A proposal of an ontology-based methodological framework for multi-platform mobile applications development

Stapić, Zlatko

Doctoral thesis / Disertacija

2014

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics Varaždin / Sveučilište u Zagrebu, Fakultet organizacije i informatike Varaždin**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:752249

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

University of Alcalá
Computer Science Department, Postgraduate School
Doctoral program “Information and Knowledge Engineering”

University of Zagreb
Faculty of Organization and Informatics
Postgraduate doctoral study "Information Sciences"

ZLATKO STAPIĆ

A PROPOSAL OF AN ONTOLOGY-BASED METHODOLOGICAL FRAMEWORK FOR MULTI-PLATFORM MOBILE APPLICATIONS DEVELOPMENT

DOCTORAL DISSERTATION

Advisors:
Prof. Vjeran Strahonja
Dr. Luis de Marcos Ortega

Alcalá de Henares & Varaždin, 2013

Sveučilište u Alcalái
Odjel računalnih znanosti, Poslijediplomska škola
Doktorski program “Inženjerstvo informacija i znanja”

Sveučilište u Zagrebu
Fakultet organizacije i informatike
Poslijediplomski doktorski studij “Informacijske znanosti”

ZLATKO STAPIĆ

PRIJEDLOG ONTOLOŠKI UTEMELJENOG METODOLOŠKOG OKVIRA ZA RAZVOJ VIŠE-PLATFORMSKIH MOBILNIH APLIKACIJA

DOKTORSKI RAD

Mentori:
Prof.dr.sc. Vjeran Strahonja
Doc.dr.sc. Luis de Marcos Ortega

Alcalá de Henares i Varaždin, 2013

Universidad de Alcalá

DEPARTAMENTO DE
CIENCIAS DE LA COMPUTACIÓN
Campus Universitario. Edificio Politécnico
28805 Alcalá de Henares (Madrid)
Teléfonos: 91 885 66 51
Fax: 91 885 66 46

UNIVERSIDAD DE ALCALÁ, PATRIMONIO DE LA HUMANIDAD

Dña. Teresa I. Díez Folledo, Profesora Titular de Universidad del Área de Lenguajes y Sistemas Informáticos, en calidad de Directora del Departamento de Ciencias de la Computación.

CERTIFICO: Que la Tesis Doctoral titulada "**A Proposal of an Ontology-Based Methodological Framework for Multi-platform Mobile Applications Development"** realizada por D. Zlatko Stapić y dirigida por el Dr. D. Vjeran Strahonja y co-dirigida por el Dr. D. Luis de Marcos Ortega, reúne los requisitos para su presentación y defensa pública habiendo recibido la conformidad del departamento en la comisión permanente celebrada el día 18 de Septiembre de 2013.

Y para que así conste, firmo la presente en Alcalá de Henares, a 26 de Septiembre de 2013

La Directora del Departamento de Ciencias de la Computación

Dña. Teresa I. Díez Folledo.

Universidad de Alcalá

Universidad de Alcalá
DEPARTAMENTO DE
CIENCIAS DE LA COMPUTACIÓN

University of Zagreb
FACULTY OF ORGANIZATION AND INFORMATICS
VARAŽDIN

Dr. D. Vjeran Strahonja, Catedrático de Universidad del Área de Ciencias de la Información y Comunicación de la Facultad de Organización e Informática de la Universidad de Zagreb.

Dr. D. Luis de Marcos Ortega, Profesor Ayudante Doctor del Área de Ciencias de la Computación e Inteligencia Artificial del Departamento de Ciencias de la Computación de la Universidad de Alcalá.

HACEN CONSTAR:

Que, una vez concluido el trabajo de tesis doctoral titulado: **"A proposal of an ontology-based methodological framework for multi-platform mobile applications development"** realizado por Zlatko Stapić, dicho trabajo tiene suficientes méritos teóricos, que se han contrastado adecuadamente mediante validaciones experimentales y que son altamente novedosos. Por todo ello consideran que procede su defensa pública.

Y para que así conste, firman la presente en Varazdin y Alcalá de Henares, a 22 de julio de 2013.

El Director de la Tesis

Dr. Vjeran Strahonja

UNIVERSITY OF ZAGREB

El Co-director de la Tesis

Dr. Luis de Marcos Ortega

Universidad de Alcalá

Onima koji su me naučili sanjati,
koji su mi omogućili krenuti,
koji su vjerovali da ću stići i s ljubavlju bili uz mene.

Mojoj obitelji.

ACKNOWLEDGMENTS

Now, when looking back, I can hardly find words to express my gratitude to those who deserve to be acknowledged and that have helped me a lot during the years of my work on this dissertation project.

First of all I want to thank my advisors Prof. Vjeran Strahonja and Dr. Luis de Marcos Ortega, for their help, patience, advices and support during all this time. Your useful recommendations, experience and motivation were of great help. Thank you for everything!

Likewise, I want to express my gratefulness to the institutions and staff of the University of Alcalá (Spain) who have kindly offered their facilities and help during my research stays in 2012 and 2013. I specially appreciate the scientific and other help provided by mentor Dr. de Marcos and Prof. José María Gutiérrez Martínez, Prof. José Javier Martínez Herráiz, Ana Maria Privado Rivera and Maria Begoña Aurrekoetxea from the Computer Science Department at the University of Alcalá.

I would like to acknowledge everyone at the Information Systems Development Department of the University of Zagreb, Faculty of Organization and Informatics for their help and support during my work on this thesis. My gratitude also goes to the Croatian Science Foundations for their financial support of my research project and my stay at the University of Alcalá. This project included important part of the research performed in this dissertation.

The dissertation language and grammar would be (at least) funny without everybody who helped me proofread the document. Thank you all, and especially thank you Tea for spending many hours in reading and suggesting the corrections to this book.

Last, but not the least, this research would never have become reality, without the love, support and motivation instilled in me by my wife Jelena, my children, mother, brother and the rest of my family. Dear Jelena, thank you for all your love and support and for always believing in me. Dear Marta and Emanuel, your smiles wiped away all exhaustion at the end of each day. Without the three of you, the sun wouldn’t shine the same for me.

ABSTRACT

Software development teams are faced with the lack of interoperability during the development of mobile applications for two or more target platforms. The development for second and every other platform means a new project with a need to repeat almost all the phases defined by the chosen methodology but with a narrow possibility of reuse of the already defined artifacts. The existing efforts of professional and scientific community to solve this problem have a similar approach ("code once, run everywhere") with similar advantages and drawbacks. Thus, this dissertation aims to propose a different solution and is concerned with: (1) analyzing the methodologies suitable for mobile applications development, (2) observing the implementation of prototype application in order to define artifacts that are created during the development process for two target platforms, (3) semantic description of artifacts and their meaning, and (4) defining unique ontological definition as a base for methodological interoperability.

The results of a systematic literature review performed on 6761 primary studies, show that current state-of-the-art literature brings only 22 development methodologies and 7 development approaches which can be identified as eligible for multi-platform mobile applications development. Among these, Mobile-D methodology accompanied with Test Driven Development was chosen and used in the observed development processes for Android and Windows Phone platforms. Total of 71 artifacts were identified and the artifacts reusability level when developing for second target platform was 66.00%. In the last research phase, the artifacts for both platforms were semantically described into a single ontological description comprising 213 classes, 14 object properties and 2213 axioms defined in ALCRIF DL expression sub-language. Having this ontology proved as correct and valid, flexible, reusable and extensible we created the basis for development of an information system to guide the development teams in a more efficient and interoperable process of multi-platform mobile applications development.

Keywords: *Methodology*, *mobile*, *multi-platform*, *development*, *ontology*.

RESUMEN

Los equipos de desarrollo de software se enfrentan al problema de la falta de interoperabilidad durante el desarrollo de aplicaciones para dos o más plataformas. El desarrollo para la segunda y subsiguientes plataformas significa un nuevo proyecto con la necesidad de repetir casi todas las fases definidas en la metodología elegida, pero con pocas posibilidades de reutilizar los artefactos definidos. Los esfuerzos realizados por la comunidad científica y profesional para solventar este problema tienen una aproximación similar ("code once, run everywhere") también con similares ventajas e inconvenientes. Esta tesis pretende proponer una solución diferente: (1) analizando las metodologías adecuadas para el desarrollo de aplicaciones móviles, (2) observando la implementación de un prototipo de aplicación que sirva para definir los artefactos creados durante el proceso de desarrollo para dos plataformas, (3) estableciendo una descripción semántica de los artefactos y su significado, y (4) creando una única definición ontológica como base para la interoperabilidad metodológica.

Los resultados de una revisión sistemática de la literatura, realizada sobre 6761 estudios primarios, mostraron que el estado del arte actual cuenta solo con 22 metodologías de desarrollo y 7 enfoques de desarrollo (development approaches) adecuados para el desarrollo de aplicaciones móviles multi-plataforma. De entre ellas se seleccionó y empleó la metodología Mobile-D junto con un enfoque dirigido por las pruebas (test driven development) para estudiar el proceso de desarrollo en las plataformas Android y Windows Phone. Se identificaron un total de 71 artefactos y el nivel de reusabilidad de los artefactos durante el desarrollo para la segunda plataforma fue del 66.00%. En la última fase de la investigación se describieron semánticamente los artefactos para ambas plataformas en una única descripción ontológica definida en el sublenguaje de expresión ALCRIF DL que cuenta con 213 clases, 14 propiedades de objeto y 2213 axiomas. Habiendo comprobado la corrección, validez, flexibilidad, reusabilidad y extensibilidad de la ontología, hemos creado la base para el desarrollo de un sistema de información que guie a los equipos de desarrollo hacia un proceso de desarrollo más eficiente e interoperable para la construcción de aplicaciones móviles multi-plataforma.

Palabras clave: *Metodología*, *móvil*, *multi-plataforma*, *desarrollo*, *ontología*

SAŽETAK

Razvojni timovi susreću se s problemom neinteroperabilnosti prilikom razvoja aplikacija za dvije ili više mobilnih platformi. Razvoj aplikacije za drugu i svaku sljedeću platformu znači novi projekt u kojem je potrebno ponovno provesti većinu faza definiranih odabranom metodikom razvoja, pri čemu se kreirani artefakti teško ili uopće ponovno ne koriste. Napori profesionalne i znanstvene zajednice za rješenjem ovog problema imaju sličan pristup („kodiraj jednom, koristi svugdje“), slične prednosti, ali i zajedničke nedostatke. Stoga ova disertacija navedenom problemu pristupa na nov način i bavi se: (1) analiziranjem metodika pogodnih za razvoj mobilnih aplikacija, (2) promatranjem razvoja prototipne aplikacije u svrhu definiranja artefakata koji nastaju pri razvoju mobilne aplikacije za dvije ciljane platforme, (3) semantičkim opisivanjem definiranih artefakata i njihovih značenja, te (4) definiranjem jedinstvene ontološke definicije kao osnove za metodološku interoperabilnost.

Rezultati sustavnog pregleda literature provedenog nad 6761 radom pokazali su da se trenutno u literaturi spominju 22 metodike i 7 pristupa koji su pogodni za razvoj više-platformskih mobilnih aplikacija. Između identificiranih metodika odabrani su Mobile-D metodika i pristup razvoju vođen testiranjem, koji su korišteni pri implementaciji prototipnog rješenja za Android i Windows Phone platformu. Ukupno je identificiran 71 artefakt pri čemu je ponovna iskoristivost artefakata pri razvoju za drugu platformu bila 66.00%. U posljednjoj su fazi istraživanja artefakti semantički opisani u zajedničku ontološku definiciju koja u konačnici sadrži 213 klasa, 14 objektnih svojstava i 2213 aksioma definiranih pomoću ALCRIF-DL jezika izraza. U radu je dokazano da je ontologija valjana, fleksibilna, ponovno iskoristiva i nadogradiva, čime je kreirana osnova za razvoj informacijskog sustava koji bi vodio razvojne timove u efikasnijem i bolje interoperabilnom procesu razvoja više-platformskih mobilnih aplikacija.

Ključne riječi: *Metodika, razvoj više-platformskih mobilnih aplikacija, ontologija*

TABLE OF CONTENTS

List of figures V

List of tables VII

List of acronyms IX

1. Introduction 1
   1.1. Outlining the problem 1
      1.1.1. Development of mobile applications 1
      1.1.2. Existing solutions 4
      1.1.3. The final remarks on platforms and tools 10
   1.2. Objectives and hypotheses 10
      1.2.1. The main goal 11
      1.2.2. Hypotheses 11
   1.3. Research scope and methodology 11
      1.3.1. Scope definition 11
      1.3.2. Research approach 13
   1.4. Dissertation disposition 15

2. Mobile applications development methodologies: a systematic review 17
   2.1. Research method 18
      2.1.1. Definition of systematic literature review (SLR) 19
      2.1.2. Steps to be performed 20
      2.1.3. Advantages and disadvantages of SLR 42
      2.1.4. Light SLR 43
      2.1.5. Conclusions on SLR 44
   2.2. Planning the review 44
      2.2.1. Defining the basic concepts 45
      2.2.2. Overview of methodologies targeting development of mobile applications 50
      2.2.3. Identification of the need for a review 54
      2.2.4. Specifying the research questions 56
      2.2.5. Developing a review protocol 57
      2.2.6. Evaluating the review protocol 62
   2.3. Conducting the review 62
      2.3.1. Identification of research 62
      2.3.2. Selection of primary studies 64

2.3.3. Study quality assessment ... 68
2.3.4. Data extraction and monitoring ... 70
2.3.5. Data synthesis ... 71
2.4. Choosing development methodology ... 74
2.5. Relevance of the chapter ... 75

3. Methodology implementation ... 77
3.1. Mobile-D overview ... 77
3.1.1. Introducing Mobile-D ... 77
3.1.2. Mobile-D process ... 78
3.1.3. Mobile-D artifacts ... 79
3.1.4. Test driven development ... 81
3.1.5. Mobile-D reference ... 81
3.2. Explore phase ... 82
3.2.1. Targeted users and stakeholders ... 82
3.2.2. Initial requirements ... 82
3.2.3. Architecture line description ... 82
3.2.4. Project plan ... 83
3.2.5. Documentation ... 84
3.2.6. Monitoring and measurement ... 85
3.2.7. Project plan checklist ... 85
3.3. Initialize phase ... 86
3.3.1. Environment setup ... 86
3.3.2. Project plan and architecture plan ... 86
3.3.3. Initial requirements analysis ... 88
3.3.4. Product backlog ... 88
3.3.5. Acceptance tests ... 89
3.3.6. User interface sketches ... 92
3.3.7. Trial Day ... 92
3.4. Productionize ... 103
3.4.1. First iteration ... 103
3.4.2. Other iterations ... 116
3.5. Stabilize ... 123
3.6. System test & fix ... 123
3.7. Development of Windows Phone application ... 125
3.7.1. Explore phase ... 125
3.7.2. Initialize phase ... 126
3.7.3. Productionize ... 128

3.7.4. Stabilize 129
3.7.5. System test & fix 129
3.8. Conclusions on implementation 130
3.9. Relevance of the chapter 132

4. Identification of the artifacts 133
4.1. Analysis setting 133
4.2. Artifacts targeting Android platform 137
4.3. Artifacts targeting Windows Phone platform 142
4.4. Cross-platform artifacts comparison 147
4.4.1. Common artifacts 148
4.4.2. Platform dependent artifacts 149
4.5. Relevance of the chapter 150

5. The ontology for methodological interoperability 153
5.1. Ontology 154
5.1.1. Definitions 154
5.1.2. Uses of ontologies 155
5.1.3. Ontologies and semantic interoperability 156
5.1.4. Ontology types 157
5.1.5. Ontology development methodologies 159
5.1.6. Ontology development tools and languages 166
5.1.7. Final remarks on ontologies 168
5.2. Android artifacts ontology 169
5.2.1. The domain and the scope of the ontology 170
5.2.2. Reuse of existing ontologies 171
5.2.3. Identified terms 171
5.2.4. Classes and class hierarchy 172
5.2.5. Properties of classes 176
5.2.6. Knowledge definition and inference 177
5.2.7. Final remarks on Android Case Ontology 182
5.3. Windows Phone artifacts ontology 182
5.3.1. Existing ontology reuse 183
5.3.2. Classes, properties and hierarchy 183
5.3.3. Updates in knowledge definition 185
5.3.4. Final remarks on Windows Phone Case Ontology 187
5.4. Common ontology for methodological interoperability 188
5.4.1. The domain and the scope of the ontology 189

5.4.2. Merging the existing ontologies ... 190
5.4.3. Updating the basic terms ... 192
5.4.4. Final class and properties hierarchy ... 193
5.4.5. Evaluating and testing the ontology ... 200
5.4.6. Final remarks on proposed ontology for methodological interoperability ... 210
5.5. Relevance of the chapter ... 211

6. Discussion of results ... 213
6.1. Methodologies for development of mobile applications ... 213
6.1.1. Performing systematic literature review in SE ... 214
6.1.2. Mobile development methodologies and approaches: SLR ... 216
6.2. Mobile-D implementation ... 218
6.3. Identification of artifacts ... 219
6.4. Ontology for methodological interoperability ... 220
6.5. Summary of the results ... 223

7. Conclusion ... 225
7.1. Research objectives revisited ... 225
7.2. Limitations of the research ... 227
7.3. Possible future research ... 228
7.4. Conclusion ... 230

References ... 233

Appendix A – Papers selected for the SLR Phase 2 analysis ... 245

Appendix B – Papers selected for the SLR Phase 3 analysis ... 260

Appendix C – Study quality assessment table ... 263

Appendix D – Filled data forms for the SLR ... 265

Appendix E – Multi-platform Case Artifacts Ontology ... 291

Extended abstract ... 369

Resumen extendido ... 389

Prošireni sažetak ... 409

Curriculum vitae ... 429

LIST OF FIGURES

Figure 1 - Problem - The Big Picture 3
Figure 2 - Architecture of some existing solutions 4
Figure 3 - MobiVine overview 6
Figure 4 - PhoneGap build process 7
Figure 5 - Architecture of some possible solutions 9
Figure 6 - Possible scope (A) 12
Figure 7 - Possible scope (B) 12
Figure 8 - Systematic Review Protocol Template 27
Figure 9 - Example of study selection process (a) 31
Figure 10 - Example of study selection process (b) 31
Figure 11 - Example of applying narrative synthesis 39
Figure 12 - Agile Risk-based Methodology 53
Figure 13 - Mobile-D process 78
Figure 14 - Artifacts in Mobile-D 79
Figure 15 - Basic project plan 84
Figure 16 - Detailed project plan 87
Figure 17 - Overall system architecture 87
Figure 18 - Mobile application architecture 88
Figure 19 - User interface sketches 92
Figure 20 - Entity users (trial day) 96
Figure 21 - Class diagram (mobile app - trial day) 98
Figure 22 - Class diagram (web service - trial day) 99
Figure 23 - Test results (trial day) 102
Figure 24 - Application screenshots (trial day) 102
Figure 25 - Data model (iteration 1) 108
Figure 26 - Mobile app class model (iteration 1) 110
Figure 27 - Web app class model (iteration 1) 111
Figure 28 - Test results (iteration 1) 114
Figure 29 - Application screenshots (iteration 1) 115
Figure 30 - Final database model 118
Figure 31 - Final class model (mobile application) 121
Figure 32 - Application screenshots 122
Figure 33 - System Test and Fix phase 123
Figure 34 - Translating user interface from Android to WP 127

Figure 35 - Automated WP unit testing 128
Figure 36 - Focusing semantic of artifacts and their origin 134
Figure 37 - Guarino's types of ontologies according to generality level..... 159
Figure 38 - De Nicola's UPON framework 165
Figure 39 - Android Case ontology top level artifacts 172
Figure 40 - Android Case ontology asserted subclasses of Inferred class 173
Figure 41 - Part of inferred model for class Artifact 181
Figure 42 - ArtifactOrigin and Artifact in WP ontology..... 184
Figure 43 - Used and Produced Documents asserted class model 187
Figure 44 - Used and Produced Documents asserted class model 187
Figure 45 - Example of automatically merged ontology..... 191
Figure 46 - Example of merged ontology 192
Figure 47 - Top level artifacts 194
Figure 48 - Asserted subclasses of Inferred class..... 195
Figure 49 - Comparing asserted and by reasoner inferred class hierarchy 204
Figure 50 - OWL 2 Validation report results 205
Figure 51 - Ontology Evaluation plug-in 206
Figure 52 - Example of defined and executed DL query with reasoning results 207

LIST OF TABLES

Table 1 - Procedures for documenting the search process 29
Table 2 - Quality concept definitions 33
Table 3 - Types of Bias 33
Table 4 - Data collection form template 35
Table 5 - Structure and Contents of Reports of Systematic Reviews 40
Table 6 - Mobile-D phases, activities and tasks 51
Table 7 - MASAM methodology phases, activities and tasks 53
Table 8 - The review protocol 58
Table 9 - Search keywords and synonyms 62
Table 10 - The list of relevant sources 63
Table 11 - Applied procedures in selection process 65
Table 12 - The list of relevant studies 66
Table 13 - Propagation of relevant studies through phases 68
Table 14 - The criteria for unbiased study identification 69
Table 15 - Quality assessment checklist 70
Table 16 - Excerpt of filled quality assessment form 70
Table 17 - Data collection form 71
Table 18 - Developed methodologies and approaches 72
Table 19 - Used methodologies and approaches 73
Table 20 - Methodologies not eligible for multiplatform development 74
Table 21 – Methodologies/approaches targeting specific mobile applications 74
Table 22 - Documents describing Mobile-D methodology 75
Table 23 - Mobile-D inputs and outputs 80
Table 24 - Project plan checklist - Explore 85
Table 25 - Product backlog 88
Table 26 - Selected feature for Trial Day 92
Table 27 - Web service (users.php) specification 97
Table 28 - Project plan checklist – 0 Iteration 103
Table 29 - Selected features for first iteration 104
Table 30 - Web service (groups.php) specification 109
Table 31 - Web service (enrolments.php) specification 109
Table 32 - Project plan checklist – 0 Iteration 116
Table 33 - Iterations plan with features selection 116
Table 34 - Performed tasks 117

Table 35 - Web services specification 119
Table 36 - Recognized system limitations 124
Table 37 - Duration of planned and real activities 130
Table 38 - Mobile-D artifacts by tasks 135
Table 39 - Template for describing the identified artifacts 137
Table 40 - Identified artifacts in development process for Android 138
Table 41 - Types of artifacts related to Android development 142
Table 42 - Identified artifacts in Windows Phone case 143
Table 43 - Common artifacts in Android in WP case 148
Table 44 - Android and WP specific artifacts 150
Table 45 - Basic terms in Android Case Ontology 171
Table 46 – Android Case ontology classes and class hierarchy 174
Table 47 - Android case ontology object properties description 176
Table 48 - DL Queries for inferred classes 180
Table 49 - WP case artifacts defined in ontology 184
Table 50 - Final list of terms used in Multiplatform ontology 193
Table 51 - Classes and class hierarchy 196
Table 52 - Object properties description 198

LIST OF ACRONYMS

ACM	Association for Computing Machinery
API	Application Programming Interface
ARR	Absolute Risk Reduction
ASD	Adaptive Software Development
AUP	Agile Unified Process
CMM	Capability Maturity Model
CMS	Centers for Medicare and Medical Services, Office of information Services
CRD	Centre for Reviews and Dissemination, University of York
CRIS	Comparative Review of Information Systems Design Methodologies
CWA	Close-World Assumptions
DL	Description Logic
DSDM	Dynamic System Development Method
DSL	Domain Specific Language
ERA	Entity-Relationships-Attribute model
EUP	Enterprise Unified Process
IEEE	Institute of Electrical and Electronics Engineers
IFIP	International Federation for Information Processing
INSPEC	Information Services for the Physics and Engineering Communities
IRI	Internationalized Resource Identifiers
JME	Java Micro Edition
JSON	JavaScript Object Notation
JSR	Java Specification Request
LSD	Lean Software Development
MASAM	Mobile Application Software Development Method
MDD	Model Driven Development
ME	Micro Edition
MVC	Model-View-Controller
MVVM	Model-View-ViewModel
NPD	New Product Development
OD101	Ontology Development 101
OR	Odds Ratio
OWL	Web Ontology Language
PDM	Platform Dependent Model

PHP	Hypertext Preprocessor: Open source scripting language
PICOC	Population, Intervention, Comparison, Outcomes, Context
PICOS	Population, Interventions, Comparators, Outcomes, Study selection
RR	Relative Risk
RAD	Rapid Application Development
REST	Representational State Transfer
RUP	Rational Unified Process
SADD	Software Architecture and Design Description (document)
SC	Story Card – An artifact in Mobile-D methodology.
SCM	Software Change Management
SDK	Software Development Kit
SDLC	Systems Development Life Cycle
SDM	Software Development Methodology
SE	Software Engineering
SLR	Systematic Literature Review
SMD	Standardized Mean Difference
SOA	Service Oriented Architecture
SPEM	Software and Systems Process Engineering Meta-model framework
SPI	Software Process Improvement
SW	Software
SWEBOK	Software Engineering Body of Knowledge
TC	Task Card
TDD	Test Driven Development
UML	Unified Modeling Language
UI	User Interface
UP	Unified Process (same as USDP)
UPON	Unified Process for ONtology building
USDP	Unified Software Development Process
XAML	EXtensible Application Markup Language
XML	EXtensible Markup Language
XP	Extreme Programming
WAC	Wholesale Applications Community
WMD	Weighted Mean Difference
WP	Windows Phone

1. INTRODUCTION

1.1. Outlining the problem

1.1.1. Development of mobile applications

The development of mobile applications differs from the development of traditional desktop or web applications in several important aspects (Rahimian and Ramsin, 2008; Spataru, 2010). According to Rahimian and Ramsin (2008), among other challenges, the designer of a software system for mobile environments has to cope with portability issues, various standards, protocols and network technologies, limited capabilities of devices and strict time-to-market requirements. Additionally, development of mobile systems is a challenging task with a high level of uncertainty, and according to Hosbond (2005), it is a result of two main sets of challenges that should be addressed in the domain of mobile systems development, namely *business* related challenges (e.g. tough competition, conflicting customer interests, establishment of revenue-share models etc.) and *development* specific challenges (e.g. rapidly changing technology, lack of standardization, integration with existing systems etc.).

When discussing the development of mobile applications, the first issue that should be addressed is the usage of methodology (Rahimian and Ramsin, 2008; Spataru, 2010; La and Kim, 2009). Classic or agile software development methodologies should be adapted for the development of mobile applications as the existing ones do not cover the specific mobile targeted requirements (La and Kim, 2009). There are several attempts from different authors to create new methodologies in order to cover the gaps in the domain of mobile applications. Some of them are *Agile Risk-based Methodology* (Rahimian and Ramsin, 2008), *MASAM* (Jeong et al., 2008), and *Mobile-D* (Abrahamsson et al., 2004).

Another issue is the use of platform specific and dependent development environments which are not interoperable in a single way (Agarwal et al., 2009). Additionally, a number of different (specific) devices which are based on the same platform (Agarwal et al., 2009; Manjunatha et al., 2010; Ridene et al., 2010) is also an important issue. This includes various hardware implementations and operating systems capabilities with support on different API levels (Agarwal et al., 2009) and which are based on different programming languages (Manjunatha et al., 2010). The problem is also known as *fragmentation problem* (Agarwal et

al., 2009; Manjunatha et al., 2010; Ridene et al., 2010), which states that a fragmentation of APIs exists even within a single platform.

Subsequently, testing becomes a great problem as simulated or emulated devices usually do not provide full functionality or are incapable of creating a real life test scenarios (Ridene et al., 2010). Testing on physical devices is usually too expensive if used to cover up all important devices and their capabilities. Several projects in this field, such as *Device Anywhere* (DeviceAnywhere, 2011) or *DSML* (Ridene et al., 2010) also do not provide full and needed functionality. Finally, the deployment and the maintenance phases should not be forgotten as well as both of them bring a fresh set of specific requirements that are mainly defined by mobile device producers and their stores.

On the other hand, the development of mobile applications also differs from the development of web or desktop applications in the number of target platforms. According to Manjunatha et al. (2010) *the fragmentation problem* forces the developers of mobile applications to focus on only specific platforms and versions. As the development of mobile applications primarily aims the wide range of users, development for only specific platforms and versions is not an option and the development teams reach for different solutions to this problem. The ideal (i.e. still nonexistent) solution would be to code once and to deploy (run) the same code to all target platforms. The fragmentation problem is the result of mobile industry being continuously highly technology-driven, which means that the focus is on innovation instead of standardization. This problem was recognized several years ago by Hosbond (2005).

Finally, it is important to notice that the development of mobile applications has some similarities with the traditional development. For example after performing an extensive literature review, Hosbond and Nielsen (2005) concluded that the scope of mobile systems development is an extension of the scope and the body of knowledge on traditional systems development. However, they also noticed that in the existing literature knowledge about traditional systems development is largely neglected. Generally, we can conclude that the reported challenges in the development of mobile applications have strong relation with the challenges that have accompanied the development in the past as some of the problems have followed the software development from the very beginning, and some have been gone and have now re-appeared again (e.g. limited capabilities of screens).

In order to define the problem in the domain of this thesis, several important concepts should be taken into consideration. The overall picture of a development playground could be presented as in Figure 1 with the following main parts:

- Teams
- Development environments
- Development methodologies

- Mediatory publishing services
- Target devices

The main characteristics of mobile applications development teams could be described in just a few words. Whether the teams are working on open source or in-house projects concerning mobile applications, they can be classified as *small, flexible*, and *keen on learning a specific technology and/or platform*. Although the classic interoperability among the team members and among different teams is not of a specific interest in this thesis, the *methodological interoperability* and the existing artifact reuse among team members or teams working on a same functionality but for a different target devices should be pointed out.

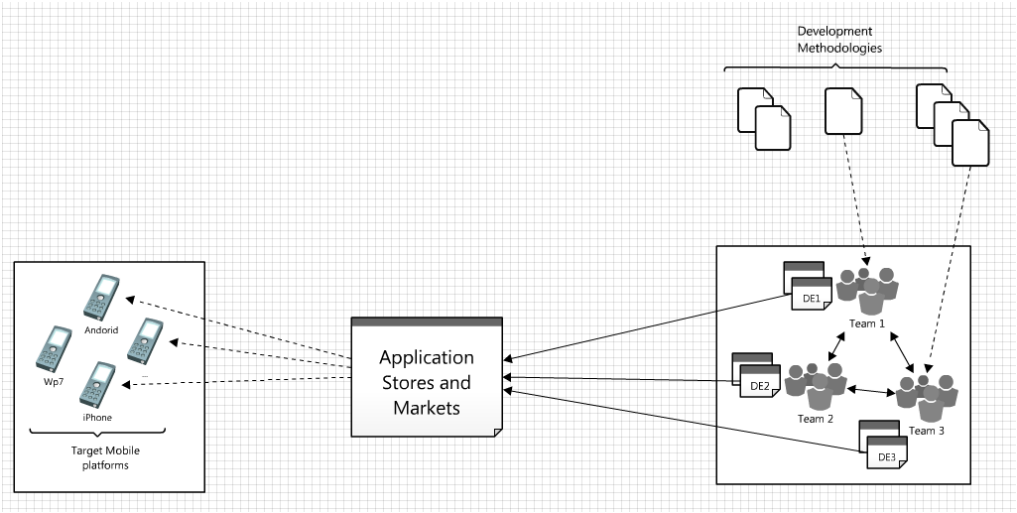


Figure 1 - Problem - The Big Picture

Let us imagine a real business scenario in which a development company wants to produce a classic business or non-business application that should be runnable on a several different mobile platforms and devices. The standard approach would be to create several different teams, each team targeting one specific platform, to adopt several development methodologies or at least different methods, each of them applicable for a specific platform and to produce characteristic outputs which will satisfy the requirements specified by the mediatory application stores or markets (see Figure 1). More experienced teams would probably try to perform as many as possible unique activities that should be similar or same across all platforms, or would even try to perform whole Model Driven Development approach through all phases except in creation of Platform Dependent Model and its implementation.

But, the big question still remains. Is it possible to make this process easier in the sense of development, interoperability and reusability? Is it possible to code once and run on different target platforms?

Unfortunately, it is not possible to code once and run on any mobile device. This slogan, according to Ridene et al. (2010), is not true even for Java, and moreover, the trends in the

mobile industry show us that this will not be possible in the short-term future, as mobile platforms are still closed, locked-in (Manjunatha et al., 2010), and devices are dependent on them. On the other hand, several different approaches aiming to propose some improvements in the multi-platform mobile applications development exist. These approaches are summarized into two main groups and shortly described in the following chapter.

1.1.2. Existing solutions

1.1.2.1. Mediatory transform engine

In the past year or two, the problem of mobile applications development for multiple target platforms became important in the scientific as well as the professional community. The results are visible in the form of several existing systems and projects that fairly enough enable the development teams to use a mediatory language or just mediatory transform engine and to code for several target platforms. Some of the most influential projects are MobiCloud (Manjunatha et al., 2010; Services Research Lab and Metadata and Languages Lab, 2011) from Kno.e.sis Research Group (Kno.e.sis Research Group, 2011), Rhodes (Rhomobile, Inc., 2011) and Amanquah & Eporwei code generator (Amanquah and Eporwei, 2009). As Figure 2 shows, reaching for this solution will bring some improvements to development teams. First of all, project team or project teams will be able to use a single proprietary or open-source programming language and could try to implement the desired functionality. The mediatory transform engine will then produce a platform specific code which can be tested and deployed through specific application store or market.

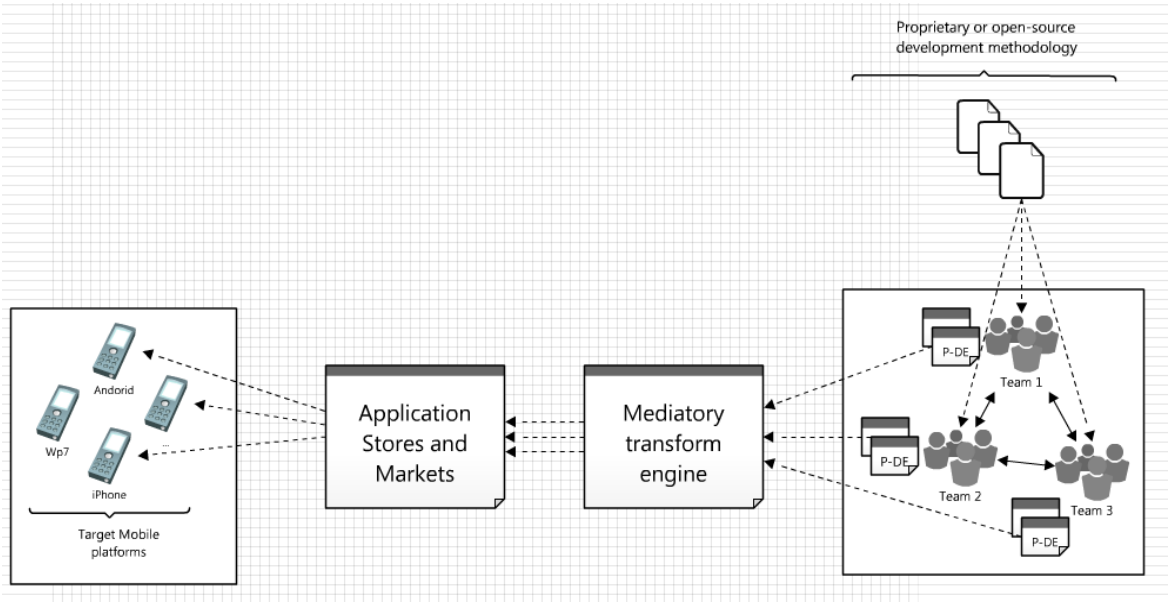


Figure 2 - Architecture of some existing solutions

```
1  #
2  # A helloworld example
3  #
4  #
5  recipe(:helloworld) do
6    metadata({:id => 'helloworld-app'})
7
8    # models
9    model(:greeting,{:message => :string})
10
11   #controllers
12    controller(:sayhello) do
13       action :retrieve,:greeting
14    end
15
16   # views
17    view :show_greeting, {:models =>[:greeting],:controller => :sayhello,:action => :retrieve}
18
19 end
20
```

Code 1 - "Hello World" application written in proprietary DSL (source: MobiCloud platform)

There are several examples of systems with described functionality. Some of them (e.g. MobiCloud) use their own domain specific language (DSL) to transform into platform specific source or, though rarely, even executable code. Other systems (Amanquah and Eporwei, 2009) transform code written in well-known languages to specific source (or executable) code. The code snippet (Code 1) shows an example written in proprietary DSL which is based on implementation of Model-View-Controller (MVC). The output could be simple "Hello World" application source code for four different platforms.

This approach, however, also has several significant drawbacks (Manjunatha et al., 2010). The idea of having mediatory transform engine that transforms source code to specific platforms depends on the efforts invested in the transform engine. The engine depends on specific platforms and available APIs, and by definition, DSL caters only to a specific domain (Manjunatha et al., 2010). Even if there is a possibility to enrich the engine with transformation procedures to all existing APIs, there is an important problem of platform incompatibilities. For example, it is not possible to use multithreading in Windows Phone 7 while, on the other hand, in other platforms it is not just possible but even desirable. Another example is Android which does not provide thread sync mechanisms as Symbian does.

Some other drawbacks of this approach are the necessity to learn a specific DSL, the boundaries defined by the use of any specific languages, the lack of control of generated source code, the lack of control of user interface design (Manjunatha et al., 2010), the problems with testing and many others.

1.1.2.2. The use of native application adapters

Another possible solution to the given problem could be the introduction of adapter applications (adapters) as native applications for every target platform (Agarwal et al., 2009). According to Agarwal et al. this is one of the two main techniques for handling fragmentation. As standardization of APIs in mobile world is still not possible, the usage of programming

techniques whereby the interface calls are wrapped, i.e. abstracted, in distinct modules which are then ported across the platforms, is left as the other solution. For example, the same authors are proposing MobiVine as a solution to handle fragmentation of platform interfaces. Specifically, the authors have identified that the fragmentation of mobile platform interfaces results in different syntax and semantics, results in usage of platform specific data structures and properties, results in throwing platform specific exceptions and is also characterized by inconsistencies in implementation by different vendors. This has bearing on the portability of mobile applications across multiple platforms. So, the proposed solution is composed on two main components: *M-Proxies* and *M-Plugins*. *M-Proxies* component helps abstract heterogeneities in interfaces across different platforms while binding to the underlying middleware stack and is used to realize platform specific blocks. The other component, called *M-Plugins*, helps integrate MobiVine with the existing tooling and deployment infrastructure and is used to override the gap between M-Proxy and platform specific APIs.

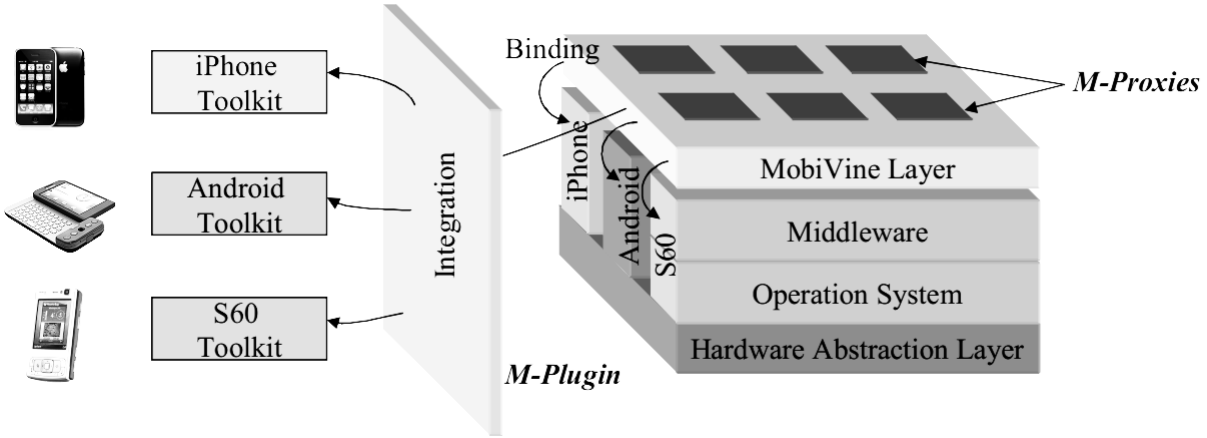


Figure 3 - MobiVine overview
(Agarwal et al., 2009)

The authors of MobiVine evaluated the usage of MobiVine as middleware layer and they discussed the achieved improvements in terms of enhancing platform and language portability, reducing code complexity, making maintenance easier and performance by a negligible fraction slower. But, they also concluded that MobiVine framework should be extended to cover other platform interfaces (like working with contact list information), to include other platforms, and to make the concept of proxy model broader by studying its applicability to other forms of mobile fragmentation, e.g. screen size and resolution.

Another well-known wrapper is PhoneGap (PhoneGap, 2011). The applications written in HTML, CSS and JavaScript are wrapped with PhoneGap and then deployed to multiple platforms. The developers could use free, open-source framework to access some of the native APIs.

After the Adobe Corporation acquired the original PhoneGap's creator Nitobi company, they also announced that they will offer developers the choice of using two powerful solutions for cross-platform development of native mobile apps, one using HTML5 and JavaScript with PhoneGap and the other using Adobe Flash® with Adobe AIR® (Adobe Corporation, 2011). On the other hand, the original PhoneGap approach has not been changed and as the application takes on extra complexity, more involved logic will require spending more time on application behavior with specific devices. Even when the same code base is used when developing for multiple platforms, the separate prepare & build and sometimes porting steps should be performed to produce the version targeting multiple platforms. According to (Lunny, 2011) more complicated applications are keen on "surprising" the developers during the porting process and in these cases, PhoneGap documentation should be consulted. In the end, there will not be a single code base Java Script file, but rather an *application.iphone.js* file containing iPhone implementation along with equivalent *application.android.js* and *application.blackberry.js* files (Lunny, 2011). Finally, there are many different guides and recommendations that should be followed while developing this way (Lunny, 2011), and we can generally conclude that taking all of them into consideration means learning a new programming and development style which is as difficult as learning a new programming language from scratch.

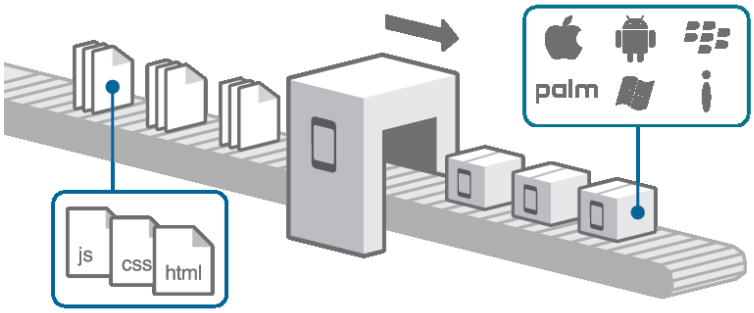


Figure 4 - PhoneGap build process
(PhoneGap, 2011)

Additionally, there are other attempts and efforts that are undertaken to over-come mobile platform and interface diversity and fragmentation. These efforts, for example, include the creation of extensions to Java platform, through Java Specification Requests (JSRs) such as JSR 248: Mobile Service Architecture (Bektesevic and Rysa, 2008) or JSR 256: Mobile Sensor API (Niemela, 2009), or the development of Wholesale Applications Community (WAC) APIs and applications (Apps).

JSRs are designed to provide the set of APIs for specifically targeted use (e.g. for mobile service architectures or mobile sensors). But, according to Agarwal et al. (2009), along with

standard Java Micro Edition (Java ME), mobile platform developers in practice choose to include different sets of JSRs which results in the diversity even among their own devices.

On the other hand, WAC is an open, global alliance of leading companies in the mobile telecommunication industry with the goal of providing a different operator network APIs through single cross-operator API platform. Specifically, this platform is built on the work of the former Open Mobile Terminal Platform Ltd.'s BONDI project[1], the Joint Innovation Lab (JIL) device APIs[2] and the GSM Association's OneAPI program[3], and currently WAC platform offers WAC Apps framework (WAC Application Services Ltd, 2012a) and WAC Payment API (WAC Application Services Ltd, 2012b). WAC Apps aims to help create the mobile apps quicker by using existing, familiar web technologies and tools through direct access to mobile device functionality. According to WAC Application Services Ltd (2012a), the types of applications that could be published currently are widgets written to the WAC specifications[4], native Android applications and HTML5 applications. WAC Payment API aims to enable developers to be able to access the operator billing capabilities through single API by using a set of developed Software Development Kits (SDKs) for multiple platforms. Although this API is useful in some cases, currently it covers only payment options and can be used for Android, PhoneGap, PHP and JavaScript/HTML5 platforms (WAC Application Services Ltd, 2012c). WAC announced that they plan to launch additional network APIs over time to provide the developers with further opportunities to create richer applications (WAC Application Services Ltd, 2012b).

So generally, the adapter-based approach requests that the adapters should be pre-developed and published in the specific application store, or as in the case of PhoneGap, deployed along with the application (PhoneGap, 2011). The general idea of creating adapter is to create a *platform specific application* that will bi-directionally convert the specific interfaces of the target platforms (left-side) into one unique interface that could be used to communicate with different applications (single, right-side). Every single adapter converts a different target interface to unique (same) interface, which means that one application really could be

---

[1] BONDI project (http://bondi.omtp.org/default.aspx) aimed to create a standardized approach for letting web applications access key local capabilities on the mobile device. [accessed: 18th of May 2012]

[2] Joint Innovation Lab was an initiative of several mobile carriers on developing device APIs and related services that build upon the W3C Widgets specification. Web page (http://www.jil.org/) is closed and redirected to WAC's page (http://www.wacapps.net/). [accessed: 18th of May 2012]

[3] "The GSMA OneAPI initiative defines a commonly supported set of lightweight and Web friendly APIs to allow mobile and other network operators to expose useful network information and capabilities to Web application developers. It aims to reduce the effort and time needed to create applications and content that is portable across mobile operators." (http://oneapi.gsma.com/) [accessed: 18th of May 2012]

[4] WAC Device API specification could be found here: http://specs.wacapps.net/index.html. [accessed: 18th of May 2012]

imported into one or more different adapters and run under one or more different platforms. The mentioned application could be stored on any web server or even on a cloud as is shown in Figure 5.

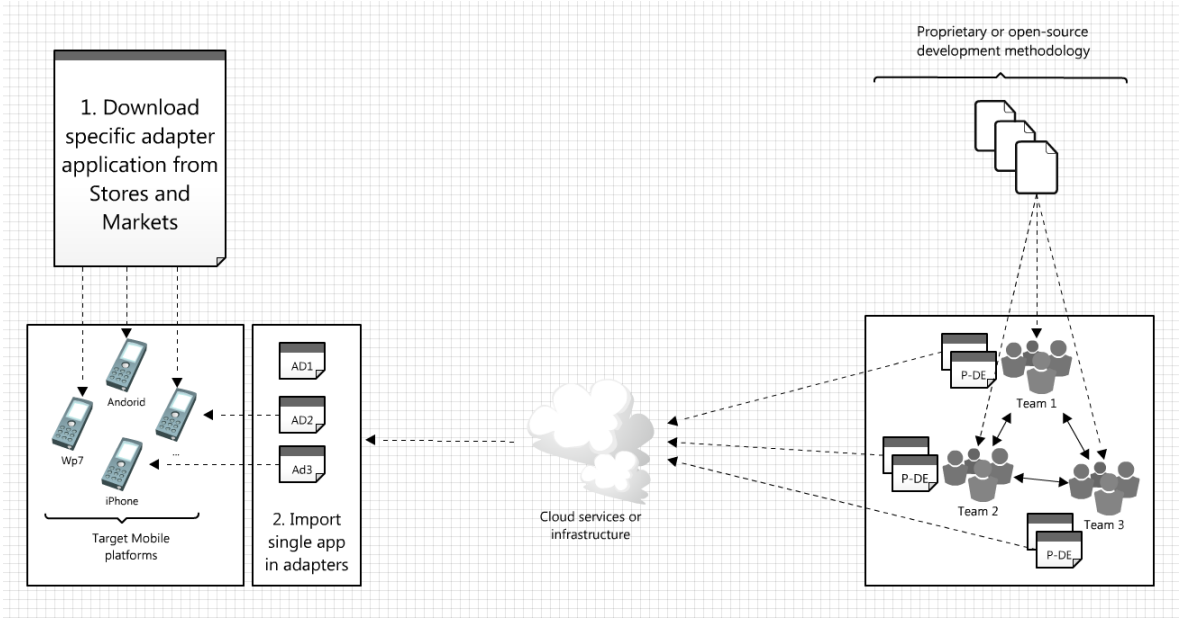


Figure 5 - Architecture of some possible solutions

There are two possible scenarios that could be implemented by adapter developers. (1) The adapters could be 100% aligned by means of common interface and this scenario would reduce the number of teams – presented in the Figure 5 – to one. This would be a great achievement, but on the other hand there is one big drawback too. The functionality of the future applications would be reduced to the common features that all target platforms support and to the common features that are implemented into the adapters for all target platforms. This brings us to the problems presented in the existing solutions and this also makes this scenario rather unlikely to be feasible. (2) The other scenario introduces some differences in the adapters by means of common (right-side) interface. If the mentioned interface is not the same for all platforms, the use of such adapters would provide a more specific functionality on mobile applications, a scenario more feasible, but also a one that would bring the need to develop more or less different applications for each target platform.

Almost all of the drawbacks stated for existing solutions that introduce transform engine are also present in this possible solution. The mentioned PhoneGap (PhoneGap, 2011) platform allows the development of native applications with web technologies (HTML5, CSS &JavaScript) enriched with a given set of APIs. According to PhoneGap Documentation[5] this

[5] PhoneGap API Reference Documentation [accessed: 15th of October 2011]: http://docs.phonegap.com/en/1.1.0/phonegap_events_events.md.html#backbutton

platform supports *back button event* only on the *Android* platform despite the fact that the event exists in several other platforms as well. Although there is some space for research in this area, especially in the field of interface transformation, the improvements that will bring the process of development of demanding applications for multiple target platforms through this approach are also hardly achievable and even feasible.

1.1.3. The final remarks on platforms and tools

As it can be seen, there are several rather different approaches that scientists and experts are taking to solve the problem of developing for multiple platforms. Each one of them has its own advantages and disadvantages. But still, one issue remains that is common to almost all of these approaches. It is impossible to create a transform engine, or adapter application that would keep all of the advantages of all target platforms and that would provide the range of possibilities as native development environments do. Also, if we want to preserve the capability of teams working on the open-source projects, it is necessary to give them the possibility to work in a native development environment and to develop by using a programming language they prefer most.

In order to provide such possibilities, this thesis will focus on proposing the solution to enhance interoperability among teams working on the same application but on different (and native) development environments. The work on the native development environments will provide the teams with the full advantages of using the native APIs, the native test environments and the native generators of the executable code.

1.2. Objectives and hypotheses

This doctoral research focuses on the analysis of this problem and on the proposal of a solution in a domain of methodological interoperability. The idea is to allow developer teams to use native development environments (that is, all their advantages for platform specific mobile application development) by raising the re-usability and interoperability to a higher, methodological level. Therefore, this dissertation will attempt to answer the following questions: (1) what methodologies and development approaches can be used in multi-platform mobile applications development; (2) what artifacts (required inputs and outputs of methodologically and methodically defined development steps) emerge during mobile applications development, (3) whether and to what extent there are similarities between these artifacts, (4) whether it is possible to ontologically describe these artifacts, and create a basis for developing a system that would support the methodological interoperability.

1.2.1. The main goal

The main goal is to ontologically describe artifacts that arise in the methodologically managed process of mobile application development targeting two or more mobile platforms, and to create the basis for more efficient and interoperable process of multi-platform mobile applications development.

1.2.2. Hypotheses

This doctoral thesis focuses on researching and proving the following hypothesis:

H_1: It is possible to create ontological description of elements of methodological interoperability containing structural and semantic aspects of sets of artifacts created in the development process of a mobile application for two or more target platforms.

1.3. Research scope and methodology

1.3.1. Scope definition

The development for mobile applications is as complex as are other fields in the domain of software engineering. There are several different perspectives that could be taken to produce a single mobile application. We can identify at least three dimensions in the space of the possible approaches the development team can take. If we include other more or less important elements the space will rapidly become multi-dimensional, and by multi we mean more than three. So to keep the thesis focused, we will take into consideration the following dimensions of space *S* as:

$S = \{M, A, P\}$ (1.)

M - Development methodology

A - Development approach

P - Target platform

The three mentioned axes could have several different values:

$M = \{m_1, m_2, \dots m_n\}$ (2.)

$A = \{a_1, a_2, \dots a_m\}$ (3.)

$P = \{p_1, p_2, \dots p_o\}$ (4.)

For example, these values could be:

M = {Extreme programming (XP), SCRUM, Rational Unified process (RUP)}

A = {Model driven development (MDD), Test driven development (TDD), Model View Controller (MVC) Implementation}

$P = \{Android, Windows\ Phone\ 7, Nokia\ Symbian\}$

While defining the scope of proposed solution it is wise to bring some logic assumptions that are based on the real life scenario and the possible usage of results gained throughout this work. Whether one team will develops multiple applications or several teams develop different applications, we can assume that the team (teams) will use the same methodology as they work together and as they want to take advantage of semantic interoperability while developing same application for different target platforms. Similar, we can assume that the development approach will be the same for development of a single application for all target platforms. Of course, the teams will develop application for one or more target platforms, so the cardinality of sets *M*, *A* and *P* can be described as:

$$|M| = 1 \quad (5.)$$

$$|A| = 1 \quad (6.)$$

$$|P| > 1 \quad (7.)$$

Subsequently, the cardinality of final space *S* that is focused in this research can be presented as in Figure 6 or in Figure 7, and can be defined as:

$$|S| = \{(1, 1, n) : n > 1\} \quad (8.)$$

The development process *DP* presented in those two figures can be described as a set of sub processes *SP* i.e. ordered triples.

$$DP = \{SP_1, SP_2, \ldots SP_n : SP_i \in S;\ SP_i = (m, a, p_i);\ 1 < n \leq |P|;$$
$$i = \{1, 2, \ldots, n\};\ m \in M;\ a \in A;\ p \in P\}. \quad (9.)$$

So for example, if we want to develop an application for Android, iPhone and Nokia, and we choose Extreme Programming supported by Model Driven Development, the development process would be described as *DP = {(3, 1, 1), (3, 1, 2), (3, 1, 3)}*. Similar, if we use SCRUM supported by Test Driven Development, the development process could be described as *DP = {(2, 2, 1), (2, 2, 2), (2, 2, 3)}*.

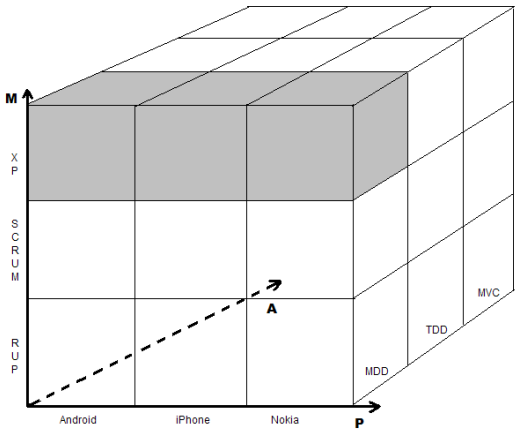


Figure 6 - Possible scope (A)

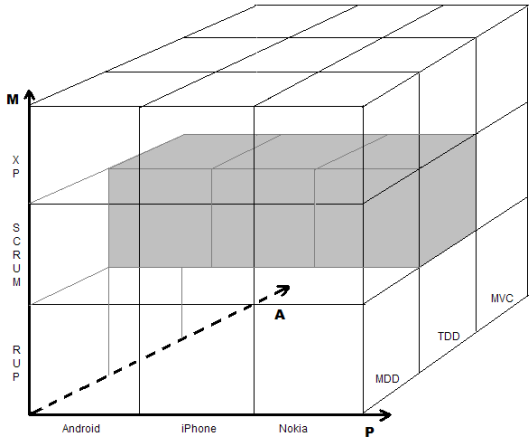


Figure 7 - Possible scope (B)

Taking into consideration all that was said, we can conclude that all ordered triples (sub processes) in one development process have the same first two elements, but different third elements. This different element makes the sub-processes (i.e. development processes for specific target platforms) rather different.

Within the presented scope, the teams will have the opportunity to work in the preferred development environments, i.e. platforms (P), and have the chance to take the advantages of the native development environment and the use of the native code: However, they will also have to obey the rule of the use of only one methodology and one development approach for the development for all the target platforms.

Note: If the teams develop an open source product, they might be interested in using specific, preferred methodology, but this scenario is not covered by this research. Additionally, the term *target platform* could be analyzed with greater granularity by defining manufacturer, platform, device and API but this is also out of the scope of this research.

1.3.2. Research approach

The overall goal of this research is to create the semantic definition of the elements of methodological interoperability containing structural and semantic aspects of the sets of artifacts created in the development process of mobile application for at least two specific target platforms. These semantic definitions can be used to create a general ontology that will be the base for interoperability and future work on the development of the framework and the supporting system. The research is divided into three main phases, each of them containing several stages. These stages, along with the used methodologies are enumerated as follows:

First phase: Choosing development methodology

- Analyze the state-of-the-art of methodologies for mobile development and choose methodology to use and describe

  $M = \{m\}$ (10.)

- Analyze the state-of-the-art of development approaches for mobile development and choose the development approach to use and describe

  $A = \{a\}$ (11.)

Second phase: Identifying artifacts sets

- Choose two specific mobile platforms to develop for according to their artifacts and development process

  $P = \{p_1, p_2\}$ (12.)

- Perform a development process *DP* by conducting *m* and *a* for p_1 and p_2 in order to create a prototype application

  $DP = \{SP_1, SP_2\} \Rightarrow DP = \{(m, a, p_1), (m, a, p_2)\}$ (13.)

- Analyze the development process and identify all obligatory and optional tasks along with the corresponding inputs and outputs:

  $IO_{p1} = \{I_{p1}, O_{p1}\} \Rightarrow IO_{p1} = \{i_{1p1}, i_{2p1}, \ldots i_{np1}, o_{1p1}, o_{2p1}, \ldots o_{mp1},\} : n, m \in N$ (14.)
  $IO_{p2} = \{I_{p2}, O_{p2}\} \Rightarrow IO_{p2} = \{i_{1p2}, i_{2p2}, \ldots i_{np2}, o_{1p2}, o_{2p2}, \ldots o_{mp2},\} : n, m \in N$ (15.)

- Define set of artifacts *R* for each target platform

  $R = \{R_{p1}, R_{p2}\} \Rightarrow R = \{(r_{1p1}, r_{2p1}, \ldots r_{np1}), (r_{1p2}, r_{2p2}, \ldots r_{mp2})$
  $: r_{ip1} \in IO_{p1}; i \leq n; r_{jp2} \in IO_{p2}; j \leq m\}$ (16.)

- If differences for p_1 and p_2 exist, find the differences in tasks, inputs or outputs on as much higher level of abstraction as possible and define a subset of artifacts that will be used for ontology definition.

  $R' = \{R'_{p1}, R'_{p2} : R'_{p1} \subset R_{p1}; R'_{p2} \subset R_{p2}\}$ (17.)

Third phase: Creating an ontology

- Analyze the state-of-the-art for ontology development and construction and choose ontology development method and ontology development language to use.
- Define all ontology elements for SP_1 and SP_2 with a special attention on the artifacts set defined in R'.

  $OE_1 = f(SP_1, R')$ (18.)
  $OE_2 = f(SP_2, R')$ (19.)

- Create specific ontologies for SP_1 and SP_2 and describe them with proper ontology definition language, with a special attention on the ontology elements defined for artifacts set defined in R'.

  $O_1 = f(OE_1, R')$ (20.)
  $O_2 = f(OE_2, R')$ (21.)

- Create a common ontology from specific ontologies by defining semantic equality and diversity; this common ontology will be the base for future interoperability on methodological level.

  $O = f(O_1, O_2, R')$ (22.)

- Look forward into a future work, framework and system development.

1.4. Dissertation disposition

After introducing the problem domain, giving an overview of existing solutions and stating the objectives, hypotheses and research scope in this chapter, the rest of this document is organized in additional six chapters as follows.

The second chapter presents the results of the Systematic Literature Review performed in order to determine the existing body of knowledge of the methodologies for mobile applications development. As the use of scientific method of SLR in the field of Software Engineering is still emerging, with a relatively small number of performed reviews, we found the existing guidelines presented in (Kitchenham and Charters, 2007) could be improved with the recommendations and inputs from other influential authors in the field, and thus first we give (in Chapter 2.1) an overview of the method along with discussion and recommendations as mentioned. Following the enhanced guidelines, that give special focus to method execution by PhD students, we continued to perform the SLR (Chapters 2.2 and 2.3) which resulted in identification of 22 development methodologies and 6 development approaches (see Table 18 and Table 19 in Chapter 2.3.5). Finally we discuss and choose Mobile-D methodology supported by Test Driven Development in Chapter 2.4 for the development of our prototype application and further analysis.

The second research phase is covered by Chapter 3 and Chapter 4 of this document. The third chapter gives an overview of Mobile-D methodology (in Chapter 3.1), and then presents the results of the multi-platform development of prototype application by using the mentioned methodology (Chapters 3.2 to 3.8). The application is developed for Android and Windows Phone target platforms, and the focus in this chapter is put on executed phases, activities and tasks along with created and used artifacts. In the fourth chapter we systemize and analyze the obtained artifacts. Chapter 4.1 gives the discussion on analysis setting, while the identified Android artifacts are presented in Chapter 4.2, the identified Windows Phone artifacts are presented in Chapter 4.3, and the cross-platform analysis is performed and reported in Chapter 4.4. A total of 71 artifacts are identified, out of which more than 70% are common to both development cases with high a reusability potential of 66% as presented in Table 43.

Chapter 5 is considered to be the most important chapter of this thesis, as it presents the taken approach along with its results in the third and the final phase of our research process. The chapter gives an overview of concepts related to ontologies and ontology development (Chapter 5.1) and then presents the created ontologies. When reporting on the development of *Android Case Artifacts Ontology* (chapter 5.2) we put focus on the usage of Ontology Development 101 methodology and implementation of its seven steps. On the other hand, when reporting on the development of the second specific ontology, namely *WindowsPhone Case Artifacts Ontology* (Chapter 5.3), we put focus on the concepts of reusing and updating

the existing ontology. Finally, Chapter 5.4 presents the development of a common ontology for both cases, and here we put focus on the concepts of merging, extending, evaluating and testing the ontologies. The created ontology is verified and validated by several different mechanisms and the results proved its semantic correctness and completeness.

The last two chapters of this document are used for extensive discussion on all research activities by reflecting on motivation, results contributions, rigor and evaluation (Chapter 6) and on summarization of contributions and conclusions which emphasize on achieved goals, open issues and possible further research directions that could be taken continuing from the results of this research (Chapter 7).

The annexes of the document bring more details on results obtained during each research phase. Thus Appendix A brings the list of all the papers that are selected for the second phase of the SLR analysis and similarly Appendix B gives the papers selected for SLR quality assessment and further analysis, while Appendix C and Appendix D respectively bring the final study quality assessment table and data extracted form for each selected study. Finally, Appendix E brings the developed ontology presented in compact and human readable Manchester syntax.

2. MOBILE APPLICATIONS DEVELOPMENT METHODOLOGIES: A SYSTEMATIC REVIEW

To goal of this chapter is to identify and choose a proper development methodology which is to be used in the rest of the research process. As, to our knowledge, there are no studies performed to identify all development methodologies suitable to mobile applications development, we performed an extensive systematic literature review of the methodologies and development approaches that are reported in the literature as being created or used specifically for mobile applications development.

As the method of systematic literature review is rather new in the field of software engineering, first the best practice in performing such time consuming and comprehensive method will be analyzed. The guidelines given by Kitchenham and Charters (2007) are followed and discussed by adding the recommendations and findings from other influential authors in the field. Special focus is given to the problem of performing the method by PhD students. This part of the chapter results with structured and detail instructions that can help researchers and PhD students to decrease the risks and biases and to increase the review quality.

Following the findings presented in the first part of the chapter we continue to plan and conduct a systematic literature review and answer two research questions: (1) what development methodologies and approaches are reported in literature as defined in theory or used in practice for mobile application development and (2) are the identified methodologies and approaches applicable in multi-platform mobile applications development? After analyzing more than 6700 initial sources we found 49 publications to be included in data extraction process which in the end resulted in identification of 22 methodologies that are used in development of mobile applications along with 7 development approaches.

Finally, we were able to establish the criteria for choosing one methodology and approach that are to be used in the rest of the research process. The chosen methodology is Mobile-D (Abrahamsson et al., 2005a) supported by Test Driven Development as Mobile-D's suggested approach.

2.1. Research method

In order to perform comprehensive and thorough analysis of existing methodologies for development of mobile applications, the systematic approach should be undertaken and existing methodologies should be reviewed in such a manner which will result in a solid basis for the rest of the research in the domain of this thesis. Such analysis could be undertaken by applying different methods and approaches, such as *systematic literature review*, *systematic mapping studies*, *tertiary reviews* discussed by (Kitchenham and Charters, 2007), or *narrative review*, *conceptual review*, *rapid review* and several other types presented by (Petticrew and Roberts, 2005). The systematic mapping study should be used when a topic is either very little or very broadly covered, and tertiary reviews are most suitable approach if several reviews in the target domain already exist and should be summarized. The narrative reviews usually do not set out the scientific methods that aim to limit systematic error. Additionally, the conceptual review should be used when aiming to provide an overview of literature in the given field and the rapid review is usually carried out within limited time or with restrictions in the scope of the research. Subsequently, taking into consideration the undertaken initial examination of the domain, we decided to use a systematic literature review (SLR) as this method has been used widely for different analysis in the field of software engineering (SE).

"A systematic literature review is a means of evaluating and interpreting all available research relevant to a particular research question, topic area, or phenomenon of interest. Systematic reviews aim to present a fair evaluation of a research topic by using a trustworthy, rigorous, and auditable methodology." (Kitchenham and Charters, 2007) The origins of systematic review can be traced back to the beginning of the 20th century, but during the 1980's, systematic research synthesis and meta-analysis reach an especially distinctive methodological status in the domain of health sciences (Williams and Carver, 2010). During this period and as a result of performing similar methods in various other fields, different synonyms of this method have been used in the literature. Some of them are *research review*, *research synthesis*, *research integration* and *systematic overview* (Biolchini et al., 2005).

In the field of software engineering during the last years several primary studies have been conducted and although these studies are accompanied by an increasing improvement in methodology, this field is still an area of investigation that remains to be explored and that could well bring many benefits in terms of mechanisms needed to assist practitioners to adopt appropriate technologies and methodologies (Biolchini et al., 2005). The guideline for systematic reviews that aimed to help software engineering researchers was proposed by (Kitchenham, 2004) and was created as adaptation of several existing guidelines from other disciplines, mainly medicine. Although the three proposed phases of systematic review, namely *planning the review*, *conducting the review* and *reporting the review*, in general were

not criticized, some authors like Biolchini et al. (2005), Mian et al. (2005) and Staples and Niazi, (2007) found that Kitchenham described them to a relatively high level which is partially inappropriate to conduct for researchers in the field of software engineering. In favor of this goes the fact that Kitchenham in 2007 published a new version of technology report (Kitchenham and Charters, 2007) with the aim to propose more comprehensive guidelines of performing a systematic literature review for researchers and PhD students in the field. The basis for this guideline remained the same: the existing guidelines used by medical researchers, but was reinforced by several books and discussions with researches from other fields.

The next sections will cover in detail the systematic literature review methodology as it is proposed in (Kitchenham and Charters, 2007). The sections will present a methodology and give summary of all phases and activities that should be performed while conducting systematic review in the field of software engineering.

2.1.1. Definition of systematic literature review (SLR)

Systematic literature review (SLR) is defined by Kitchenham and Charters (2007) as "*a form of secondary study that uses a well-defined methodology to identify, analyze and interpret all available evidence related to a specific question in a way that is unbiased and (to a degree) repeatable"*. Dybå and Dingsøyr (2008a) define SLR as "*a concise summary of the best available evidence that uses explicit and rigorous methods to identify, critically appraise, and synthesize relevant studies on a particular topic*". According to Dybå, these methods should be defined in advance and documented in a protocol so the others could critically appraise and replicate the review.

There are different reasons for performing systematic literature review. In general, whenever a literature review is performed it could be done by applying systematic (following stated procedures and steps) or unsystematic (just reading and taking notes) approach. The usual reason to use SLR is to *summarize the existing evidence concerning a treatment or a technology*. This is to say that for example, as is the case in this thesis, systematic literature review can be used to summarize the methodologies that could be used for development of mobile applications. SLR could also be used to *identify any gaps in current research in order to suggest areas for further investigation* or *to provide a framework/background in order to appropriately position new research activities.* In addition, there are other general reasons to use a systematic rather than unsystematic approach, such as the purpose of the research, the scientific approach, the quality expectations or the existence of previous researches on the selected topic.

According to Dybå and Dingsøyr (2008a) the key feature that distinguishes SLR from traditional narrative reviews is in its explicit attempt to minimize the chances of making wrong conclusions which could be the results of biases either in primary studies or in the review process itself.

2.1.2. Steps to be performed

Although the methodology of SLR is considerably upgraded if compared to the first version from 2004, the main three phases remain the same. General steps to be performed are also similar and are defined as follows:

Phase 1: Planning the review

- Identification of the need for a review
- Commissioning a review (optional)
- Specifying the research question(s)
- Developing a review protocol
- Evaluating the review protocol (recommended)

Phase 2: Conducting the review

- Identification of research
- Selection of primary studies
- Study quality assessment
- Data extraction and monitoring
- Data synthesis

Phase 3: Reporting the review

- Specifying dissemination mechanisms
- Formatting the main report
- Evaluating the report (recommended)

According to the author of the review process, Kitchenham, all mentioned activities (stages) are mandatory except *commissioning a review* as it depends on the planned commercialization of review results, as well as *evaluating the review protocol* and *evaluating the report* which are optional as they depend on the quality assurance procedures decided by the author(s) of the review. In any case, the mentioned activities are recommended.

As one can conclude from the above list, the mentioned stages and phases are sequential. However, it is important to mention that some of the stages can be repeated more than once and may involve iteration or reimplementation. For example, the negative evaluation of review protocol or negative evaluation of the report might result in the need to repeat the part or the whole review process. Or, the inclusion and exclusion criteria of the relevant studies

could be refined after quality criteria are defined. It is important to notice that even experienced scientists often have to change or adapt the review protocol. To some authors this provides a reason for criticism of the methodology of the already existing reviews for not being completely objective or even conducting a fake rational design process. However, there are authors such as Staples and Niazi (2007) who discuss the need of the protocol even if it is a subject of constant changes through the whole systematic review process. All that has been said brings us to a strong general conclusion that the protocol is needed and that it increases the quality of the process.

In the following sections, each stage of the SLR process will be discussed in detail.

2.1.2.1. Planning the review

The most important activities during the phase of review planning are definition of the review question(s) and creation of the review protocol. However, the rest of the activities should not be neglected and also deserve a serious approach. The results of this phase should be a clearly defined review protocol containing the purpose and the procedures of the review.

The summary of each stage is presented below and is based on guidelines presented in (Kitchenham and Charters, 2007) and on additional discussions from other authors cited in the text.

Identification of the need for a review is the first activity in the SLR process. It arises from the preliminary research in the topic area. When the author(s) has a firsthand knowledge in the area of interest, then it is possible to conclude whether more thorough and unbiased research is needed. It is especially important to identify and review the existing systematic reviews on the same topic. The review of existing SLRs is usually undertaken against appropriate and previously created evaluation criteria. The most common practice is to create a checklist or set of questions that should be examined for every existing SLR. There are several checklists proposed by different authors and organizations, and depending on the level of complexity, they usually operate with concepts of the quality of defined inclusion and exclusion criteria or the level of literature and relevant studies coverage along with the assessment of quality of included studies. For example Centre for Reviews and Dissemination (2009) in the book *Systematic Reviews* defines the following set of questions to use while critically appraising review articles:

- Was the review question clearly defined in terms of population, interventions, comparators, outcomes and study designs (PICOS)?
- Was the search strategy adequate and appropriate? Were there any restrictions on language, publication status or on publication date?

- Were preventative steps taken to minimize bias and errors in the study selection process?
- Were appropriate criteria used to assess the quality of the primary studies, and were preventative steps taken to minimize bias and errors in the quality assessment process?
- Were preventative steps taken to minimize bias and errors in the data extraction process?
- Were adequate details presented for each of the primary studies?
- Were appropriate methods used for data synthesis? Were differences between studies assessed? Were the studies pooled, and if so was it appropriate and meaningful to do so?
- Do the authors' conclusions accurately reflect the evidence that was reviewed?

Commissioning a review is an optional task whose inclusion in the process depends on the type and the stakeholders of the review process. If the review is commissioned by an organization that has no time or expertise to perform a review by itself, then the organization must provide a commissioning document that will contain all important information about the required work such as project name, review questions, timetable, budget or dissemination strategy.

Scientists and PhD students will not create a commissioning document while performing a systematic literature review as a part of their own work. The only issue that should be addressed in this case is that a dissemination strategy should be incorporated in the review protocol.

Specifying the research question or questions is probably the most important part of the systematic review process as it is the base for all other activities. The research question defines which primary studies to include or exclude from the review, and the data that should be extracted from the reviewed literature. The defined research question should be answered in the final systematic literature review report.

As Kitchenham emphasizes, there are several types of research questions (adapted from guidelines in the domain of health care) that can be stated in the domain of software engineering. These questions may concern, for example, effect of SE technology, cost and risk factors, the impact of technology on different concepts et cetera. The type of a question can sometimes determine the guidelines and procedures to be used (as for example in domain of health care). My opinion is that it is not necessary to create a finite set of types of research questions, but rather to use a set of guidelines on how to create a research question that has the appropriate structure. According to Kitchenham, it is important to create a *right question*, i.e. a question that *is meaningful and important to practitioners and researchers,* that *will lead either to changes in current SE practice or to increased confidence in the value of*

current practice, or that *will identify discrepancies between commonly held beliefs and reality.* Finally, the right questions can be the questions that are primarily of interest to researchers in order to identify and scope the future research activities. For example, such question could be used in a systematic review performed by a PhD student in order to identify existing basis and to identify if and where the research fits into the current body of knowledge.

Usually, authors define more than one research question or they define one high-level research question and then break it down to several more specific and concrete questions. For example, in order to characterize software architecture changes by means of a systematic review, Williams and Carver (2010) created the following high-level question: *Can a broad set of characteristics that encompass changes to software architectures be identified using the current software engineering body of knowledge and be used to create a comprehensive change assessment framework?* Additionally, the authors created five more specific questions along with accompanying motivation. The specific questions were:

- What are the attributes of the existing software change classification taxonomies?
- How are software architecture elements and relationships used when determining the effects of a software change?
- How is the architecture affected by the functional and non-functional changes to the system requirements?
- How is the impact of architecture changes qualitatively assessed?
- What types of architecture changes can be made to common architectural views?

Another approach is to create a single research question, and in order to clarify its boundaries, several complementary research questions can be created. For example, in order to review the reasons for undertaking CMM[6]-based SPI[7] initiatives in organizations, Staples and Niazi (2008) defined the following research question: *Why do organizations embark on CMM-based SPI initiatives?* And, in order to clarify the question they stated several complementary questions that were not used during the investigation:

- What motivates individuals to support the adoption of CMM-based SPI in an organization?
- Why should organizations embark on CMM-based SPI initiatives?

[6] CMM is an acronym for Capability Maturity Model. The CMM was first introduced by Humphrey W. S., as a model and practical guidance for improving the software development and maintenance process (Humphrey, 1989). CMM is applicable to other processes as well.

[7] SPI is an acronym for Software Process Improvement and referes to an approaches that are intended to improve the practice of software engineering. One of these approaches is also an CMM-based approach (Staples and Niazi, 2008).

- What reasons for embarking on CMM-based SPI are the most important to organizations?
- What benefits have organizations received from CMM-based SPI initiatives?
- How do organizations decide to embark on CMM-based SPI initiatives?
- What problems do organizations have at the time that they decide to adopt CMM-based SPI?

The research questions also depend on the type of review which, according to Noblit and Hare (1988), can be *integrative* or *interpretative*. According to Dybå and Dingsøyr (2008a) the difference between integrative and interpretative reviews is that integrative reviews are concerned with combining or summarizing data for the purpose of creating generalizations, and interpretative reviews achieve synthesis through combination of concepts identified in the primary studies into a higher-order theoretical structure. This division could be aligned with the principles of "right questions" mentioned earlier in this chapter.

According to Petticrew and Roberts (2005) it is a good way to start the question writing process by breaking it down into sub-questions. If the review aims to answer a question about the effectiveness, the authors suggest using a model called PICOC, defining a *population, intervention, comparison, outcomes* and *context.* These criteria were accepted in Kitchenham's guidelines and discussed from the viewpoint of software engineering as follows:

- *Population* in the terms of SE can assume wide range of roles or groups and even areas, from *novice testers*, *experienced software architects* to, for example, *control systems*. As the number of undertaken primary studies in the field of SE is relatively small (comparing to other fields), it is wise to avoid any restriction on the population.
- *Intervention* should define a software methodology/tool/technology/procedure that the authors are interested in reviewing and that should address specific issue that is in the focus of the research. Basically, intervention is the concept that is going to be observed in the context of the planned systematic review.
- *Comparison* is the software engineering methodology/tool/technology/procedure with which the intervention is being compared. If the comparison technology is the conventional or commonly-used technology, it is often referred to as the "control" treatment and the control situation must be adequately described.
- *Outcomes* should relate to factors of importance to practitioners. All relevant outcomes should be specified, without using surrogate measures that may be misleading.
- *Context* refers to the context in which the comparison takes place (e.g. academia or industry), participants taking part (e.g. practitioners, consultants, students) and the tasks being performed (e.g. small scale, large scale). There are many examples of

unrepresentative experiments, i.e. the experiments that are undertaken in academia using students and small scale tasks, and these should be excluded from serious systematic reviews.

Developing a review protocol is considered as the most important activity of the whole planning phase as it determines the rest of the SLR process. The output of this activity should be a detailed review protocol that specifies the methods that will be used to perform a planned systematic review. Creating a protocol prior to systematic review is necessary to reduce the possibility of researcher bias. Staples and Niazi (2007) claim that review protocol, as a concrete and formal plan of the systematic review, usually insinuates and suggests the structure of the final report.

Protocol should also describe the background context of the research, the specific research questions, the planned search strategy, criteria for publication selection, the treatment of publication quality assessment, the data extraction plan, the data synthesis plan and a project plan. Although usually it is impossible to predict all the elements and obstacles in the whole systematic review process, above mentioned parts define it in general. That is why some authors, for example Staples and Niazi (2007), argue that a protocol is a subject of constant changes through the whole systematic review process. In the guidelines, Kitchenham suggests that aspects of the protocol should be piloted during its development. In particular, the search terms, selection criteria, and data extraction procedures should be tried out before finalizing the protocol.

Although some elements of the review protocol are already stated, the full list of elements of the protocol, defined by (Kitchenham and Charters, 2007), is presented here without any changes:

- *Background*. The rationale for the survey.
- *The research questions* that the review is intended to answer.
- *The search strategy* that will be used to search for primary studies including search terms and resources to be searched. Resources include digital libraries, specific journals, and conference proceedings. An initial mapping study can help determine an appropriate strategy.
- *Study selection criteria*. Study selection criteria are used to determine which studies are included in, or excluded from, a systematic review. It is usually helpful to pilot the selection criteria on a subset of primary studies.
- *Study selection procedures*. The protocol should describe how the selection criteria will be applied e. g. how many assessors will evaluate each prospective primary study, and how disagreements among assessors will be resolved.

- *Study quality assessment checklists and procedures*. The researchers should develop quality checklists to assess the individual studies. The purpose of the quality assessment will guide the development of checklists.
- *Data extraction strategy*. This defines how the information required from each primary study will be obtained. If the data require manipulation or assumptions and inferences to be made, the protocol should specify an appropriate validation process.
- *Synthesis of the extracted data*. This defines the synthesis strategy. This should clarify whether or not a formal meta-analysis is intended and if so what techniques will be used.
- *Dissemination strategy* (if not already included in a commissioning document).
- *Project timetable*. This should define the review schedule.

Taking into considerations the discussion from other authors, several stated elements are especially important. For example Dybå and Dingsøyr (2008a) argue that explicit inclusion and exclusion criteria (which should specify the types of study designs, interventions, populations and outcomes that will be included in the review) and a systematic search strategy (which should specify the keyword strings and bibliographic sources defined in a such way to ensure good topic coverage) are of the most importance. They also state that sometimes it is even necessary to perform a search of key journal and conference proceedings by hand to identify relevant studies that are not fully indexed. On the other hand, some authors put focus on quality assurance elements and on planning, considering them to be critical in order to mitigate risks of researcher bias (Kitchenham and Charters, 2007) or in order to support the practical conduct of systematic review (Staples and Niazi, 2007).

In order to make the process of development of review protocol easier, Kitchenham gave an example of protocol for a tertiary study review. On the other hand, Biolchini et al. (2005) created a *Systematic Review Protocol Template* which, even based on the first version of the Kitchenham's guidelines, covers majority of concepts and could be used as a starting point in creating a review protocol. Except the mentioned guidelines, protocol was also based on the systematic review protocols developed in the medical area and on the example found in Protocol for Systematic Review by Mendes E. and Kitchenham B., 2004. (as cited by Biolchini). Every concept in Biolchini's template is described in detail and a pilot study was conducted in order to evaluate the developed protocol template. The results of the study showed that usage of template has significantly shortened the time spent on planning against the review execution time[8].

[8] More details on mean time spent on systematic review tasks along with simple formula to predict the needed time are presented in (Petticrew and Roberts, 2005).

The *Systematic Review Protocol Template* created by (Biolchini et al., 2005) is composed of five main parts. The original template is given in Figure 8 without any changes.

1. Question Formularization
   - 1.1. Question Focus
   - 1.2. Question Quality and Amplitude
     - Problem
     - Question
     - Keywords and Synonyms
     - Intervention
     - Control
     - Effect
     - Outcome Measure
     - Population
     - Application
     - Experimental Design
2. Sources Selection
   - 2.1. Sources Selection Criteria Definition
   - 2.2. Studies Languages
   - 2.3. Sources Identification
     - Sources Search Methods
     - Search String
     - Sources List
   - 2.4. Sources Selection after Evaluation
   - 2.5. References Checking
3. Studies Selection
   - 3.1. Studies Definition
     - Studies Inclusion and Exclusion Criteria Definition
     - Studies Types Definition
     - Procedures for Studies Selection
   - 3.2. Selection Execution
     - Initial Studies Selection
     - Studies Quality Evaluation
     - Selection Review
4. Information Extraction
   - 4.1. Information Inclusion and Exclusion Criteria Definition
   - 4.2. Data Extraction Forms
   - 4.3. Extraction Execution
     - Objective Results Extraction
       - **i)** Study Identification
       - **ii)** Study Methodology
       - **iii)** Study Results
       - **iv)** Study Problems
     - Subjective Results Extraction
       - **i)** Information through authors
       - **ii)** General Impressions and Abstractions
   - 4.4. Resolution of divergences among reviewers
5. Results Summarization
   - 5.1. Results Statistical Calculus
   - 5.2. Results Presentation in Tables
   - 5.3. Sensitivity Analysis
   - 5.4. Plotting
   - 5.5. Final Comments
     - Number of Studies
     - Search, Selection and Extraction Bias
     - Publication Bias
     - Inter-Reviewers Variation
     - Results Application
     - Recommendations

Figure 8 - Systematic Review Protocol Template (Biolchini et al., 2005)

Evaluating the review protocol is not compulsory, but is a recommended step in the SLR process in order to improve its quality as the protocol is a critical element of any systematic review. The researchers must take into consideration several aspects in order to agree on a procedure for evaluating the protocol. Important aspects are purpose of the research, desired quality, time, financial construction etc. With regards to these, there are several methods of evaluating a review protocol which can be used:

- author's review (not recommended)
- peer review
- review by supervisor (appropriate for PhD students)
- review by external experts (the best option)
- test of protocol execution

Review by external experts is probably the best option, but it usually depends on the financial construction of the review project. In this case, the group of external experts should be asked to review the protocol, and the same group can be asked to review the final report.

Test of protocol execution is a good and widely used alternative method. In this case, the review of protocol is executed by performing a full cycle of systematic review (following the

protocol) but on a reduced set of selected sources. If the gained results are not suitable, or if any phase of the review reveals unexpected problems, the new version of the protocol must be created.

2.1.2.2. Conducting the review

According to Kitchenham's guidelines, *conducting the review* phase consists of five obligatory stages. This phase takes most of the researcher's time, and although all five stages are important, identification of research and selection of primary studies will determine the rest of reviewing process. In this phase the predefined protocol should be followed and the phase should result in data extracted, summarized and ready for dissemination.

The summary of each stage is presented below and is based on guidelines presented in (Kitchenham and Charters, 2007) and on additional discussions from other authors which are cited in the text.

Identification of research, as a first step in conducting a review, it results in a list of entire population of publications relevant to the research questions and obtained by performing a search strategy.

The search strategy should be the same as stated in the review protocol, and it should be stated in such a manner that it allows the study to be replicable and open to external review. If a researcher is not experienced in a creating a search strategy, then he or she should ask for help (for example from librarian). It is also good to break down the research question and to identify initial search strings according to population, intervention, comparison, outcomes, context and study design. On top of that, it is important to create a list of synonyms, abbreviations and alternative spellings. Apart from results gained from digital libraries, other sources such as reference lists from relevant primary studies, journals, grey literature (e.g. technical reports), research registers and the Internet should also be searched (sometimes manually).

The process of definition of search strategy is usually iterative and should benefit from preliminary searches, trial searches and consultations with experts in the field.

In order to address publication bias (the problem that positive results are more likely to be published than negative) and not to allow it to become a systematic bias, Kitchenham suggests that it is important to take appropriate steps. For example scanning grey literature, conference proceedings and contacting domain experts could result in addition of studies with "negative" results.

As the number of identified primary studies may be extensive (some authors, for example Unterkalmsteiner et al. (2012) have identified more than 10.800 publications), the appropriate

reference manager software should be used to keep a record on all of them along with the links to the potentially useful full papers.

Process of performing a SLR must be transparent and replicable. This means that the whole process should be properly documented: the review and search must be documented, and unfiltered search results should be saved and retained for possible reanalysis. Many of these documents will not be presented in the final report but can also be published and a reference to them can be given in the final report. Kitchenham proposed the procedures for documenting the search process according to data source as presented in Table 1.

Table 1 - Procedures for documenting the search process

Data source	**Documentation**
Digital Library	Name of database Search strategy for the database Date of Search Years covered by search
Journal hand Searches	Name of journal Years searched Any issues not searched
Conference proceedings	Title of proceedings Name of conference (if different) Title translation (if necessary) Journal name (if published as part of a journal)
Efforts to identify unpublished studies	Research groups and researches contacted (names and contact details) Research web sites searched (date and URL)
Other sources	Date of search URL Any specific conditions pertaining to the search.

Source: (Kitchenham and Charters, 2007)

In an attempt to perform an exhaustive search Brereton et al. (2007) identified seven electronic sources as most relevant sources to Software Engineers, and they also discuss about considering the use of additional sources (*) from publishers or bibliographical databases:

- IEEExplore
- ACM Digital library
- Google scholar
- Citeseer library
- INSPEC
- ScienceDirect
- EI Compendex
- *SpringerLink
- *Web of Science
- *Scopus

Unfortunately, the search of many relevant journals can only be performed manually, but is also an important part of the search process. The usual way to identify relevant journals is to read papers reference lists or by searching the Internet. Several authors also tried to identify a list of relevant journals and conferences in the field of software engineering. For example,

combining the recommendations from (Hannay et al., 2007; Kitchenham and Charters, 2007), the list of relevant journals and conferences (ordered alphabetically) could be:

- ACM Transactions on Software Engineering Methodology (TOSEM)
- ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) [9]
- Empirical Software Engineering (EMSE)
- Evaluation and Assessment in Software Engineering (EASE)
- IEEE Computer
- IEEE Software
- IEEE Transaction on Software Engineering (TSE)
- Information and Software Technology (IST)
- International Conference on Software Engineering (ICSE)
- Journal of Software: Evolution and Process (JSEP) [10]
- Journal of Software: Practice and Experience (SP&E)
- Journal of Systems and Software (JSS)

Selection of primary studies is performed on all identified (potentially relevant) studies by applying an inclusion and exclusion criteria in order to assess their actual relevance. The selection criteria are also decided during the protocol definition but if necessary, they can be refined during this process. The identification of research will usually end up with a great number of articles that do not answer to the research question (because the keywords may have different meanings or may be used in the studies that are not in the focus of SLR research topic). The *inclusion criteria* will define which of these studies to include in the set of relevant ones, and the *exclusion criteria* can be applied on the already selected studies in order to identify those that do not meet additional conditions, or on the initial list of studies in order to remove irrelevant ones. Inclusion and exclusion criteria should be based on the research question, but could be defined based on study types. For example, only quantitative studies will be taken into consideration.

Study selection is a multistage and iterative process. If the number of initially obtained studies is large, the authors usually start with simple criteria and, for example, in the first iteration include/exclude studies only by reading the title. In the second iteration the abstract is read

[9] ESEM symposium was first held in 2007 as a merge of *IEEE International Symposium on Empirical Software Engineering (ISESE)* and *IEEE International Symposium on Software Metrics (METRICS)*, so if searching for papers prior to 2007 it is wise to check issues of ISESE and METRICS.

[10] JSEP journal was born from two parent journals, *Journal of Software Maintenance and Evolution: Research and Practice* and *Software Process: Improvement and Practice*, and the second one should be searched separately as it was issued until 2009. Issues of the first journal are available on the current JSEP home page.

and finally, full papers are read. Two study selection processes are shown in Figure 9 (Unterkalmsteiner et al., 2012) and Figure 10 (Dybå and Dingsøyr, 2008a).

However, some authors advocate a more strict approach. For example, Brereton et al. (2007) advise the researchers to exclude studies by means of reading the title and the abstract only if there are no doubts that study can be excluded. Otherwise, they point out that they have learnt from their own experience that "the standard of IT and software engineering abstract is too poor to rely on when selecting primary studies", and they advise reviewing the conclusions as well. Of course, final set of selected papers should be reviewed in detail.

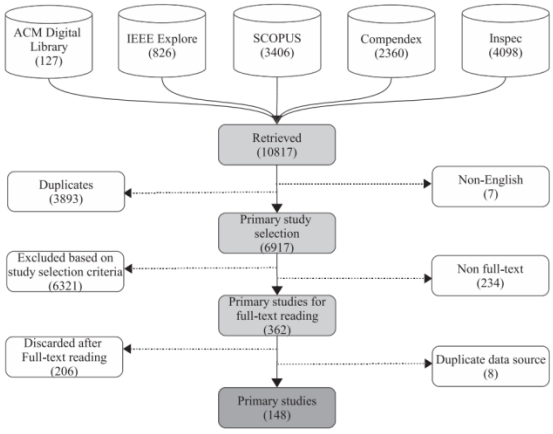


Figure 9 - Example of study selection process (a)

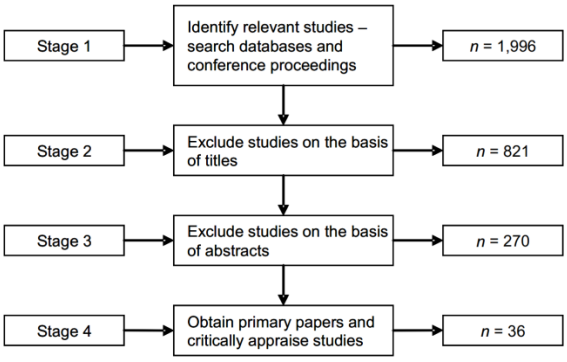


Figure 10 - Example of study selection process (b)

Kitchenham is familiar with general instructions on keeping the list of excluded papers, but she suggests that totally irrelevant papers should be excluded first (for example, papers that have nothing to do with Software Engineering) and then, while analyzing other papers, the list of exclusions should be kept updated along with the reasons of exclusion.

In order to increase the reliability of inclusion decisions it is possible to perform the same process by two or more researches. The Cohen Kappa coefficient (Cohen, 1968) can be used to measure the level of agreement between the researches[11]. If there is a disagreement then it should be discussed and resolved, but the initial value of Kappa statistics should be preserved in the final report and used for discussion and conclusions. Alternatively, using test-retest approach latter researches can evaluate a random sample of the primary studies.

[11] The Cohen Kappa coefficient (Cohen, 1968) is statistical measure of agreement between two observers rating qualitative items. The simple Kappa coefficient (from 1960) is calculated for nominal scales and it treats all disagreements between raters equally. But, the Weightet kappa, κ_w, provides the means of taking into consideration the ratio-scaled degrees of disagreement between raters. Theoretical Kappa maximum of 1.0 means perfect agreement between raters.

On the other hand, a PhD student can use one of the following methods to increase the reliability of inclusion decisions:

- consultation with advisor
- consultation with expert panel or other researcher
- re-evaluation of a random sample of the primary studies by the test-retest approach
- re-evaluation of a random sample by other researcher while publishing a paper on the subject

Advisors usually help students to choose an appropriate method and if decided so, the advisor can review the inclusion decisions or help the student find external experts or perform other stated methods.

Study quality assessment is the second most important stage in this phase. The idea of this process is to analyze and assess the quality of each primarily selected study to be finally included in data extraction and reporting process. In general, the aim of assessing the quality is to make sure that the study findings are relevant and unbiased. However, this is not a simple process as, according to Kitchenham, there is no agreed definition of study "quality". Some authors, for example Centre for Reviews and Dissemination (2009), discuss that the study quality assessment procedures mainly depend on the type of the study. For example, in health sciences, the quality assessment of a study that was conducted by using a *randomized controlled trials* method cannot be the same as the assessment of *quasi-experimental studies* or *observational studies*. The mentioned guidelines also state that the following elements should be assessed regardless of the study type:

- appropriateness of study design to the research objective
- risk of bias
- choice of outcome measure
- statistical issues
- quality of reporting and intervention
- generalizability

Mentioned elements do not have the same importance in every case, but the authors usually agree that the risk of bias (also known as *internal validity*) is pernicious as it can easily obscure intervention effects. Generalizability (also known as *applicability* or *external validity*) considers the extent to which a study is generalizable and how closely a study reflects a practice (Centre for Reviews and Dissemination, University of York, 2009). Additionally, Kitchenham states that quality assessment should be used to:

- provide more detailed inclusion/exclusion criteria
- provide explanation for differences in study results
- weigh the importance of individual studies for overall synthesis

- guide the interpretation and further research

In this process, Kitchenham also finds that three concepts are important and most closely related to the study quality. She defines them as follows:

Table 2 - Quality concept definitions

Term	Synonyms	Definition
Bias	Systematic error	A tendency to produce results that depart systematically from the 'true' results. Unbiased results are internally valid.
Internal validity	Validity	The extent to which the design and conduct of the study are likely to prevent systematic error. Internal validity is a prerequisite for external validity.
External validity	Generalizability, Applicability	The extent to which the effects observed in the study are applicable outside of the study.

Source: (Kitchenham and Charters, 2007)

The most common tool (quality instrument) used to assess the quality of studies is *checklist*. Usage of checklists ensures that all assessed studies are evaluated critically and in a standardized way. According to Centre for Reviews and Dissemination (2009) there are many different checklists and scales already available, and they can be used or adapted to meet the requirements of the review or to cover the bias and validity in the focus of specific research. In literature several types of biases are recognized that should be addressed in a *checklist*. Kitchenham adopted the division and adapted the definitions and protection mechanisms in order to address software engineering rather than medicine. The identified types of biases along with definition and protection mechanisms are as follows:

Table 3 - Types of Bias

Type	Synonyms	Definition	Protection mechanism
Selection bias	Allocation bias	Systematic differences between comparison groups with respect to treatment.	Randomization of a large number of subjects with concealment of the allocation method (e.g. allocation by computer program not experimenter choice).
Performance bias		Systematic difference is the conduct of comparison groups apart from the treatment being evaluated.	Replication of the studies using different experimenters. Use of experimenters with no personal interest in either treatment.
Measurement bias	Detection bias	Systematic difference between the groups in how outcomes are ascertained.	Blinding outcome assessors to the treatments is sometimes possible.
Attrition bias	Exclusion bias	Systematic differences between comparison groups in terms of withdrawals or exclusions of participants from the study sample.	Reporting of the reasons for all withdrawals. Sensitivity analysis including all excluded participants.

Source: (Kitchenham and Charters, 2007)

In addition to these, Higgins and Green (2011) emphasize *reporting bias* and also recognize *other biases*. By *reporting bias* they discuss systematic differences between reported and unreported findings, and by *other biases* they presume other sources of bias that are relevant in certain circumstances (for example language etc.).

According to Kitchenham, checklist should also include consideration of biases and validity problems that can occur at the different stages of the study (*design, conduct, analysis* and *conclusions*). Reviewing available papers on the subject of checklists creation for quantitative studies, and noticing that authors focus on different set of questions, Kitchenham and Charters (2007) created an accumulated list of 59 questions and organized them with respect to study stage and study type. These questions cover four mentioned stages and can be used for *quantitative empirical studies*, *correlation (observational) studies*, *surveys* and *experiments*. The same process was conducted in qualitative studies, and resulted in 18 questions that could be used. These example checklists, which we highly recommend, should not be used literally, but rather as a pool of questions. The appropriate questions could be taken from the pool for each specific study.

The review protocol should define quality instruments as well as specify how the quality data are to be used. In general, there are two rather different but not mutually exclusive ways: (1) to assist primary study selection and (2) to assist data analysis and synthesis.

There are several limitations the authors should be aware of when attempting to perform a quality analysis of different studies. First primary studies could be poorly reported, but the lack of report does not necessarily mean a leak in the procedure. According to Petticrew and Roberts (2005) the quality checklists should address methodological quality and not reporting quality. If reporting quality is poor, the researchers should attempt to obtain more information from the authors of the study. Additionally, Kitchenham argues that a limitation could be a limited evidence of the relationships between factors that are thought to affect validity and the actual study outcomes, and that sometimes it is not possible to correct the statistical analysis as there is usually no access to the original data.

Finally, authors usually point out all undertaken quality assessment procedures and measures, but only to the level of detail that is suitable for the target publication. For further reading, we recommend some simple examples of quality assessment of SE studies presented in (Dybå and Dingsøyr, 2008a), (B Kitchenham et al., 2009), (Barbara Kitchenham et al., 2009) or (Kitchenham et al., 2010) and especially (Unterkalmsteiner et al., 2012).

Data extraction and monitoring, as a next step in SLR process, aims to accurately and without bias record the appropriate information from selected papers. Researchers usually, during the protocol definition phase, define *extraction forms* which are used in this activity.

The design of data extraction forms is not a trivial task while forms should be *designed to collect all information needed to address the review questions and the study quality criteria.* As the quality criteria can be used to identify inclusion/exclusion criteria or/and as a part of the data analysis, in the first case, the data extraction forms should be separated, and in the second case, a single form can be used (Kitchenham and Charters, 2007). In any case, the same authors recommend that the forms should be piloted during the protocol definition phase, and all researchers who will use the forms should take part in the pilot study in order to assess completeness of the forms along with possible technical issues.

Basically, as mentioned before, data extraction forms should contain questions needed to answer the review questions and quality evaluation criteria. There is no firm guidance on how to define these questions as they are different for every specific SLR process. On the other hand, there are several elements that are considered to be common to all forms in order to provide standard information. According to Kitchenham these elements are:

- name of the reviewer
- date of data extraction
- title, authors, journal, publications details
- space for additional notes

Combining the examples presented in (Kitchenham and Charters, 2007) and (Jørgensen, 2007) we can conclude that in general, data extraction form could include parts (sections) as presented in Table 4.

Table 4 - Data collection form template

Data item	**Value**	**Additional notes**
Extraction information		
Data extractor		
Data checker		
Date of extraction		
General study information		
Study identifier		
Title		
Publication details		Including authors, journal etc.
Questions to answer review questions		
Question 1		These questions could aim to obtain numerical or descriptive data. Each review question could be covered by more questions in data extraction form.
Question 2		
Question *n*		
Questions to assess study quality		
Question 1		These questions should be related ONLY to data analysis. Questions related to inclusion/exclusion criteria should be
Question 2		
Question *m*		

		stated on separate form.
Data summary		
Question 1		These questions could aim to collect summary information from the observed study.
Question 2		
Question *p*		

It is important to notice that the column *Additional notes* was used to present additional info on template elements, but it should also be used in extraction forms to present additional info on the extracted data.

Similarly as in the process of applying inclusion and exclusion criteria, there are different methods that could be performed to extract the data and to fill the extraction forms. In guidelines Kitchenham recommends that data extraction should be performed by two or more researchers, but as stated in (B Kitchenham et al., 2009), in practice she finds that it is useful that one researcher extracts the data and the other one checks the extraction. If several researchers are performing a data extraction, the results should be compared, aligned and if necessary discussed. However, if researchers are performing extraction on different sets of primary studies, it is important to ensure that it is done in a consistent manner by employing some cross-checking activities. Additionally, Staples and Niazi (2007) recommend that the whole process should be done in an iterative manner. PhD students will usually need some help from advisor or other experts to randomly check their extracted data or they will perform a re-test of a part of the primary studies.

Incidentally, it is important not to include multiple studies with the same data in a systematic review in order to avoid results with bias. This could be a serious threat if different sets of publications are analyzed by different researchers. Conversely, it is also important to contact the authors if it is identified that some data are missing or were poorly reported.

Finally, the authors should consider using *electronic forms* as they proved themselves useful in subsequent data analysis, especially if the extracted data is a set of numerical values and if statistical or meta-analysis has been performed.

An interesting example of data extraction process can be found in (Unterkalmsteiner et al., 2012), an example of filled extraction forms can be found in (Jørgensen, 2007) and (Dybå and Dingsøyr, 2008b) and an example of data extraction forms with a short review on process can be found in almost all papers mentioned in this chapter.

Data synthesis is the final step in the review conduction phase. During this activity extracted data are collected and summarized. In general, there are two types of data synthesis: *descriptive (narrative) synthesis* and *quantitative synthesis* (Centre for Reviews and

Dissemination, University of York, 2009). In order to draw reliable conclusions, synthesis should consider the strength of evidence, explore consistency and discuss inconsistencies.

The synthesis approach should be defined by the protocol and is determined by the type of research questions, but also by the type of available studies and by the quality of data. For example, it is not wise to perform a statistical analysis on the numerical data if the publications used are not randomized or do not cover the whole population, or if there are studies with poor quality and with biased results. In addition, according to CRD's guidance (2009), narrative and quantitative approaches are not mutually exclusive, and according to (Brereton et al. (2007) "software engineering systematic reviews are likely to be qualitative in nature".

Regardless of the synthesis type, the synthesis should begin with a creation of a summary of included studies. The studies included in the review are usually presented in a table which covers all their important details (such as type, interventions, number and characteristics of participants, outcomes etc.). In the same (or in another) table, the elements of study quality and risk of bias could also be presented. Additionally, this descriptive process should be explicit, rigorous and should help to conclude if the studies are similar and reliable to synthesize (Centre for Reviews and Dissemination, University of York, 2009). Kitchenham and Charters (2007) also add that the extracted data should be tabulated in a manner that is consistent with the review questions and structured to highlight similarities and differences between study outcomes.

Synthetizing results of qualitative studies means an integration of materials written in natural language, with significant possibility of having to understand different meanings of the same concepts as they were used by different researchers (Kitchenham and Charters, 2007). In (Noblit and Hare, 1988) the authors propose three approaches to synthesis of qualitative studies:

- *Reciprocal transaction* – translation of cases of studies with similar objective into each of other cases in order to create an additive summary.
- *Refutational synthesis* – translation of studies along with corresponding refutational studies in order to analyze the refutations in detail.
- *Line of argument synthesis* – first, the individual studies which focus the part of some problem are analyzed and then the set is analyzed as a whole in order to get broader conclusion on the addressed problem.

According to Petticrew and Roberts (2005) the narrative synthesis can be performed in several ways, but the most common one is to separate it into three distinct steps: (1) *organizing the description into logical categories*, (2) *analyzing the findings within each of the categories* and (3) *synthesizing the findings across all included studies*. The mentioned

authors argue that there is no firm guidance on how to organize the categories and that this could be done according to: intervention, population, design, outcomes etc. The second step involves a narrative description of the findings for each study. This description may vary in length and in the level of detail. Finally, the authors discuss the cross-study synthesis and state that it usually starts with a simple description of the uncovered information, then the summary information on the effect of mediating variables (if any) can be presented, and at the end the results of the individual studies are described. The main goal of cross-study synthesis is to produce an overall summary of study findings taking into considerations the quality and other variations.

Additionally, same authors describe several other synthesis methods which could be used:

- *Best evidence synthesis* – "combines the meta-analytic approach of extracting quantitative information in a common standard format from each study with a systematic approach to the assessment of study quality and study relevance".
- *Vote counting* – the easiest approach which simply compares the number of positive and negative results on specific issue. This approach is usually inappropriate to use as it has many disadvantages.
- *Cross-design synthesis* – in theory combines the complementary strengths of experimental and non-experimental research – for example by adjusting the results of *random controlled trials (RCTs)* by standardizing RCT results to the distributions obtained from database analyses.

An example of applying a narrative synthesis is presented in (Centre for Reviews and Dissemination, University of York, 2009) and can be seen in Figure 11.

Quantitative data (as well as qualitative) should be presented in tabular form. The data must be presented in a comparable way, and according to Kitchenham, it should include:

- sample size for each intervention,
- estimated effect size for intervention with standard error for each effect,
- difference between the mean values for each intervention and the confidence interval for the difference,
- units used for measuring the effect.

Different effect measures for different types of outcome are proposed in literature. Kitchenham refers to medical literature and she presents *binary outcomes* (which can be measured by effect measures like odds, risk, odds ratio (OR), relative risk (RR), absolute risk reduction (ARR)) and *continuous data* (which can be measured by mean difference, weighted mean difference (WMD) or standardized mean difference (SMD)).

Apart from narrative description of results, qualitative results are usually presented and summarized in a table. Even though "tabulating the data is a useful means of aggregation, it is necessary to explain how the aggregated data actually answers the research questions" (Brereton et al., 2007). On the other hand, quantitative results are usually presented by *forest plot* (which presents the means and variance of the difference for each study) (Kitchenham and Charters, 2007) and, of course, additionally narratively discussed and related to the research questions.

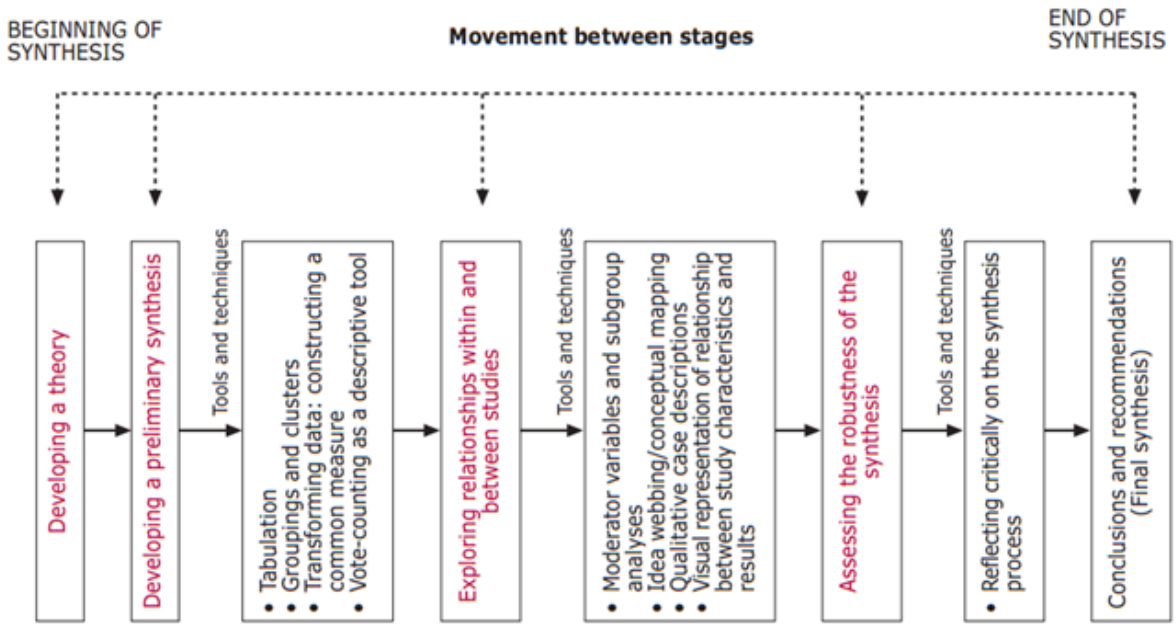


Figure 11 - Example of applying narrative synthesis
(Centre for Reviews and Dissemination, University of York, 2009)

When systematic literature review includes quantitative and qualitative studies, Kitchenham suggests that researchers should "synthetize the quantitative and qualitative studies separately, and then attempt to integrate the results by investigating whether the qualitative results can help explain the quantitative results". When there is a considerable difference in the quality of studies, Kitchenham suggests the *sensitivity analysis* to be performed in order to determine if the low quality publications have significant impact on synthesis results. Sensitivity analysis could also be performed on different subsets of primary studies to determine the robustness of the results.

Examples of different methods and approaches of presentation of systematized data can be found in Chapter 1.3.5. of (Centre for Reviews and Dissemination, University of York, 2009).

2.1.2.3. Reporting the review

The aim of the final phase of the systematic literature review process is to write the results of the review in a form suitable to dissemination channel and the target audience or parties. The results are usually written in a form of a systematic review report. The summary of possible activities is presented below and is based on the guidelines presented in (Kitchenham and Charters, 2007) and on additional discussions from other authors which are cited in the text.

Specifying dissemination strategy and mechanisms is usually performed during the project commissioning activities, or if there is no commissioning phase, then dissemination strategy and mechanisms should be defined in the review protocol. Kitchenham argues that apart from disseminating the results in academic journals and conferences, scientists should consider performing other dissemination activities that might include direct communication with affected bodies, publishing the results on web pages, posters or practitioner-oriented magazines etc.

If the results are to be published in a conference or journal, or any other publication with restricted number of pages, then the reference to a document (technical report, PhD thesis or similar) that contains all information should be provided.

Formatting the main report is the most important activity of this phase. Kitchenham adopted the suggested structure of systematic review report given in CRD's guidelines from 2001. Although the original guidelines (from 2001) are updated in (Centre for Reviews and Dissemination, University of York, 2009), the version presented by Kitchenham is sufficient in the field of software engineering. She also distinguishes reports which are to be published in technical reports and journals from the reports which are to be published in a PhD dissertation. The report structure proposed by Kitchenham is presented in Table 5 and elements marked with the (*) are usually used only in publications and not in PhD dissertations.

Table 5 - Structure and Contents of Reports of Systematic Reviews

Section	Subsection	Scope	Comments
Title*			The title should be short but informative. It should be based on the question being asked. In journal papers, it should indicate that the study is a systematic review.
Authorship*			When research is done collaboratively, criteria for determining both who should be credited as an author, and the order of author's names should be defined in advance. The contribution of workers not credited as authors should be noted in the Acknowledgements section.
Executive	Context	The importance of the	A structured summary or abstract

summary or Structured abstract*		research questions addressed by the review.	allows readers to assess quickly the relevance, quality and generality of a systematic review.
	Objectives	The questions addressed by the systematic review.	
	Methods	Data Sources, Study selection, Quality Assessment and Data extraction.	
	Results	Main finding including any meta-analysis results and sensitivity analyses.	
	Conclusions	Implications for practice and future research.	
Background		Justification of the need for the review. Summary of previous reviews.	Description of the software engineering technique being investigated and its potential importance.
Review questions		Each review question should be specified.	Identify primary and secondary review questions. Note this section may be included in the background section.
Review methods	Data sources and search strategy		This should be based on the research protocol. Any changes to the original protocol should be reported.
	Study selection		
	Study quality assessment		
	Data extraction		
	Data synthesis		
Included and excluded studies		Inclusion and exclusion criteria. List of excluded studies with rationale for exclusion.	Study inclusion and exclusion criteria can sometimes best be represented as a flow diagram because studies will be excluded at different stages in the review for different reasons.
Results	Findings	Description of primary studies. Results of any quantitative summaries. Details of any meta-analysis.	Non-quantitative summaries should be provided to summarize each of the studies and presented in tabular form. Quantitative summary results should be presented in tables and graphs.
	Sensitivity analysis		
Discussion	Principal findings		These must correspond to the findings discussed in the results section.
	Strengths and Weaknesses	Strengths and weaknesses of the evidence included in the review. Relation to other reviews, particularly considering any differences in quality and results.	A discussion of the validity of the evidence considering bias in the systematic review allows a reader to assess the reliance that may be placed on the collected evidence.
	Meaning of findings	Direction and magnitude of effect observed in summarized studies. Applicability (generalizability) of the findings.	Make clear to what extent the results imply causality by discussing the level of evidence. Discuss all benefits, adverse effects and risks. Discuss variations in effects and their reasons (for example are the treatment effects larger on larger projects).
Conclusions	Recommend-actions	Practical implications for software development.	What are the implications of the results for practitioners?
		Unanswered questions and implications for future	

		research.	
Acknowledge-ments*		All persons who contributed to the research but did not fulfill authorship criteria.	
Conflict of interest			Any secondary interest on the part of the research (e.g. a financial interest in the technology being evaluated) should be declared.
References and Appendices			Appendices can be used to list studies included and excluded from the study, to document search strategy details, and to list raw data from the included studies.

Source: (Kitchenham and Charters, 2007)

Evaluating the report is the final step in the systematic literature review process. This activity depends mainly on the type of the publication. Papers submitted to a scientific conference or scientific journal are reviewed by independent peer reviewers. Doctoral dissertations are reviewed by supervisors and by the committee during the examination process. Finally, if the publication is a technical review, it is also advisable to subject the materials to an independent evaluation. In this case, this final review could be done by the same expert panel that was created to review the research protocol. The results of the review, if negative, can require repetition of one or more phases in the systematic literature review process.

2.1.3. Advantages and disadvantages of SLR

As every other method and approach, SLR also has several advantages and disadvantages. Kitchenham identified three main groups of advantages of using systematic literature review. (1) The methodology is well-defined; (2) it enables researchers to provide the information available in the wide range of sources; (3) and in the case of quantitative data, it is possible to perform some meta-analysis and to extract information that single study cannot provide (Kitchenham and Charters, 2007). Additionally, if compared to unstructured methods, like simple literature review, the SLR has many advantages (described in the SLR process) that make the results of such analysis more reliable and more likely to be unbiased.

On the other hand, a major disadvantage of this approach is that it requires much more effort and time in comparison to simple literature review and this is exacerbated by a large number of review points: search term pilot reviews, protocol reviews, initial selection reviews, final selection reviews, data extraction reviews, and data analysis reviews (Staples and Niazi, 2007). Kitchenham also adds that the usage of meta-analysis could be a disadvantage as it can detect small and unimportant biases. Biolchini discusses that authors are supposed to perform complex activities and understand (sometimes unknown) specific concepts and terms. This is why he states that a conduction of SLR in SE is much harder than in other disciplines, for example medicine (Biolchini et al., 2005). Same authors point out that the overall process is

difficult to conduct (in order to help other researchers they prepared a systematic review conduction process and protocol template), especially the activities of protocol development, searching and evaluating studies.

Additionally, execution of this method depends on solid literature coverage of the focused phenomenon, and subsequently it cannot be used to explore new, revolutionary, phenomena which are not well covered in literature.

Finally, even experienced authors are likely to change the review protocol during the implementation phase, and that brings the problem of documenting the whole process.

2.1.4. Light SLR

The text in this chapter (Chapter 2.1) is based on the guidelines presented in (Kitchenham and Charters, 2007) and expanded with the reported feedback of the researchers, mainly from the field of software engineering. As the guidelines' authors themselves also point out, both, the guidelines and therefore this text too, are mainly created to cover the whole process of systematic literature review which is supposed to be undertaken by a large group of researchers. Although the notes for single researchers (like PhD students) throughout the text have been presented, it is important to point out that not all mentioned activities are compulsory. Kitchenham suggests that the most important steps (as *light SLR*) for PhD students to undertake are:

- Developing a protocol
- Defining the research question(s).
- Specifying what will be done to address the problem of a single researcher applying inclusion/exclusion criteria and undertaking all the data extraction.
- Defining the search strategy.
- Defining the data to be extracted from each primary study including quality data.
- Maintaining lists of included and excluded studies.
- Using the data synthesis guidelines.
- Using the reporting guidelines.

Specific recommendations are given to PhD students throughout the whole chapter while discussing specific activities. The most important for PhD students is to understand that the process should be performed with the restrictions that are normal while performing a PhD research, but research validity and rigor should not be neglected and should be achieved by employing available methods and techniques in order to get unbiased results. These include the adjustment of dissemination strategy, proper review questions that are from interest to the student, employment of supervisor to review the protocol, consultations with supervisors or other researcher to increase the reliability of inclusion decisions, implementation of test-retest

approach or asking the advisor or other researcher to randomly check the extracted data and structure the report according remarks given in the guidelines.

2.1.5. Conclusions on SLR

The process of systematic literature review is not easy to perform, but the general opinion of the authors is that this method is useful and could be used to decrease the biases and to increase the review quality. Authors also note that the usage of this method has significant obstacles in the field of software engineering in comparison to other fields, for example, the field of health sciences. The main differences are the mainly qualitative studies to be reviewed in SE, the lack of centralized index of existing systematic reviews and the overall literature searching problem raised by many different sources, with different and questionable quality. In order to overcome the mentioned obstacles, the authors who performed SLR in the field of SE suggest that the scope of the review should be limited by choosing clear and narrow research questions and that the whole process should be in advance well defined by putting a considerable effort in creation of feasible review protocol.

As SLR method still emerges in the field of software engineering, the SLR authors in the field of SE welcome the idea of publishing the replications of existing systematic reviews, along with the idea of creation of a centralized index of the existing literature reviews.

2.2. Planning the review

The previous chapter defining the research method (chapter 2.1) covers the whole SLR process as defined by Kitchenham and Charters (2007), including the phases of planning the review, conducting the review and reporting the review along with summarized and aggregated findings, observations and recommendations from other influential authors in the SE field.

The following chapters will report the whole process of performing the Systematic Literature Review in the scope of this research. Firstly, following the mentioned guidelines, the phase of planning the review will be presented in this chapter (chapter 2.2), while the chapter 2.3 will give the information on the phase of performing the review and finding the suitable methodology and chapter 2.4 brings the conclusion of this process and justifies the decision on the methodology that was used in this research.

2.2.1. Defining the basic concepts

Systems development methodologies (SDM) are of an academic interest since the early 1980s when the IFIP WG8.1[12] organized three conferences named Comparative Review of Information Systems Design Methodologies (CRIS). The first conference (Olle et al., 1982) aimed to present and compare spectrum of methodologies. The second conference (Olle et al., 1983) had a goal to analyze the features of the methodologies and the third conference (Olle et al., 1986) put the focus on the evaluation of the methodologies. These conferences also resulted in the definition and distinction of basic concepts and terms like methodology, method, tool, approach, and development cycle. However, the used concept of "methodology" was limited only to the design stage of the system development life-cycle (Gasson, 1995).

Since these origins, different definitions for the term "software development methodology" which cover full development life-cycle are created. For example, software development methodologies could be defined as (a) *"reference model for the development of software describing the various statuses of the corresponding software projects"* (Dyck and Majchrzak, 2012), as (b) *"framework for applying software engineering practices with the specific aim of providing the necessary means for timely and orderly execution of the various finer-grained techniques and methods for developing software-intensive systems"* (Ramsin and Paige, 2008), as (c) *"recommended collection of phases, procedures, rules, techniques, tools, documentation, management, and training used to develop a system"* (Avison and Fitzgerald, 2003) or (d) *"software development process by which user needs are translated into a software product by translating user needs into software requirements, transforming the software requirements into design, implementing the design in code, testing the code, and sometimes, installing and checking out the software for operational use"* (IEEE Computer Society, 1991) or as (e) *an organized and systematic approach to developing software for a target computer* (*SWEBOK V3 - Chapter 10*, 2012).

Consequently, SDM could be observed as a noun and as a verb. As a noun, "software development methodology is a **framework** that is used to structure, plan, and control the process of developing an information system" – this includes the pre-definition of specific deliverables and artifacts that are created and completed by a project team to develop or maintain an application (Centers for Medicare and Medicaid Services (CMS), Office of information Services, 2008). As a verb, the software development methodology could be considered as an **approach** used by organizations and project teams to apply the software

[12] IFIP WG8.1 – Working group of the International Federation for Information Processing on Design and Evaluation of Information Systems. The group is part of IFIP's Technical Committee on Information Systems (TC8). More information is available on the group's website: http://research.idi.ntnu.no/ifip-wg81/.

development methodology framework. Every software development methodology approach acts as a basis for applying specific frameworks to develop and maintain software. The terms Systems Development Life Cycle (SDLC) and Software Development Process are used to represent the meaning of SDM as a verb. According to Elliott (2004) the SDLC can be considered to be the oldest formalized methodology **framework**[13] for building information systems with the idea of "pursuing the development of information systems in a very deliberate, structured and methodical way, requiring each stage of the life cycle from inception of the idea to delivery of the final system, to be carried out rigidly and sequentially".

2.2.1.1. Development approaches

Although SDLC is defined as framework, with time and to manage the complexity, a number of SDLC models or methodologies as **approaches** have been created. The CMS (2008) enumerates several software development approaches which have been used since the origin of information technology. Arguably, this division could be considered as division which takes into consideration the *development cycle, the phases and their order* and according to this viewpoint, all approaches could be stated in one of the three main groups:

- *Phase oriented approach* – developed at the end of 1960s and the beginning of 1970s – states that each development phase is performed only once during the whole development project. In each phase, all required output results are finished and checked. The verification (in accordance with specification) and validation (by the user) on the results are performed.
- *Partially incremental approach* - defines approach in which only several phases are repeated incrementally, but initial set of phases is performed only once. In this model, initial phases including requirements specification are usually not repeated, and the design and subsequent phases are repeated. Other variants of the model exist (e.g. Incremental implementation only etc.).
- *Incremental approach* – states that the overall software functionality should be produced and delivered in small increments. Attention is focused only on essential features and additional functionality is added only if and when needed. The output models evolve and they are improved in every increment (iteration).

In comparison, by taking into consideration *the basic model to be used to define the product*, the development approaches could be:

[13] Initially it was a framework, but during the time the term changed meaning to specify approach!

- *Process oriented approach* (functional approach) – defines that the specification of system/software functionality is most important. Using process modeling techniques, it is possible to formally define process hierarchy, process inner logic, inter processes relationships, occurring events, and relationships between the process and the surroundings. The basic concepts that are used in this approach are functional components (such as functions, processes, sub processes, activities, operations etc.), data flows and their content, data sources and destinations, data storages and events that initiate or terminate processes.
- *Data oriented approach* – assumes that the basic model developed through the overall process of information / software system development is data model. The data model is considered to be more stable than process model and that it changes more rarely. In addition, it is considered that the data manipulation is the only important activity that is performed by some information systems processes. The basic concepts of this approach are: data structure definition concepts, data integrity preservation concepts, operators that can be used to change the state of the data.
- *Process and data oriented approach* – defines that the data models are equally important as process models and that these two models cannot be separated. This approach, which appeared in the beginning of the 1980s, also defines that every data model belongs to a specific process model, and that these two should be developed in parallel.
- *Object oriented approach* – defines the latest approach which semantically unites the data model and process model into new object models. These models represent objects, methods serving the objects and messages exchanged between the objects. They can be used to model the static and dynamic system / software properties. The basic concepts of these models are: object types, classification and built-in object structures, attributes with relationships and constraints, events and states, operations performed on objects (methods), inheritance, encapsulation, polymorphism, reusability, state pre-conditions and post-conditions, state transitions, messages…

2.2.1.2. Development methodologies

Emerging from 1960s, many different methodologies have been created and developed in theory and practice and they basically reflect the mentioned approaches. The number of these methodologies makes the categorization of SDMs not an easy task. Different authors use different viewpoints while defining categories of SDMs. Avison and Fitzgerald (2003) divide methodologies into seven broad groups: *Structured, Data-oriented, Prototyping, Object-oriented (OO), Participative, Strategic* and *Systems*. These groups are not mutually excluded. On the other side, Ramsin and Paige (2008) while focusing only on object oriented methodologies divide them into three sub-groups: *Seminal*, *Integrated* and *Agile.* In their

opinion, *seminal*[14] methodologies pioneered the unexplored field of OO analysis and design and set the basis for further evolution. Many of the concepts introduced by these methodologies are still widely used today. While the first and the second generation of OO methodologies is referred to as *seminal*, the third generation is referred to as *integrated*[15]. These methodologies are heavyweight and very complex, offering detailed process components, patterns, and management and measurement instructions. Furthermore, some of them propose ideas on seamless development, complexity management and modeling approaches. Finally, in contrast to heavyweight integrated methodologies, *agile*[16] methodologies are aiming to be lightweight, based on practices of program design, coding and testing in order to enhance software development flexibility and productivity.

Similarly, software engineering body of knowledge (*SWEBOK*, 2004) defines three basic software engineering methods topic areas, while the new version of the Report, that is now being in process of review and is soon to be published (*SWEBOK V3 - Chapter 10*, 2012), defines four topic areas as follows:

- *Heuristic methods* – those experience-based software engineering methods that have been and are fairly widely practiced in the software industry. This topic area contains three broad discussion categories: *structured analysis and design methods*, *data modeling methods*, and *object-oriented analysis and design methods*.
- *Formal methods* – are software engineering methods used to specify, develop, and verify the software through application of a rigorous mathematically based notation and language. Through the use of the specification language, the software model can be checked for consistency (in other words, lack of ambiguity), completeness, and correctness in a systematic and automated or semi-automated fashion.
- *Prototyping methods* – Software prototyping is an activity that generally creates incomplete or minimally functional versions of a software application, usually for trying out specific new features, soliciting feedback on requirements or user interfaces, further exploring requirements, design, or implementation options, and/or gaining some other useful insight into the software. The software engineer selects a prototyping method to understand the least understood aspects or components of the software first; this approach is in contrast with other development methods which usually begin development with the most understood portions first. Typically, the prototyped product does not become the final software product without extensive development rework or refactoring.

[14] i.e. influential, had a greate influence on other methodologies.
[15] i.e. combined, unified.
[16] i.e. nimble, responsive.

- *Agile methods* – Agile methods were born in the 1990s out of the need to reduce the apparent large overhead associated with heavyweight, plan-based development methods used in large-scale software-development projects. Agile methods are considered lightweight methods in that they are characterized by short, iterative development cycles, self-organizing teams, simpler designs, code refactoring, test-driven development, frequent customer involvement, and an emphasis on creating a demonstrative working product with each development cycle.

The criterion used to create this classification could be argued. Heuristic methods (a kind of approach to development based on modeling rather than on heuristics!) have *models* as primary artifacts, prototyping methods result in a *throw-away prototype* and formal methods result in a *formal specification* of the system (which should preferably be animated by using some engine). In this point of view, the main artifact of agile methods is not obvious. In eXtreme programming these are small releases that have passed unit, integration and acceptance tests while in Scrum these could be features described through product and sprint backlogs. Thus, we can conclude that common artifact denominator for agile methods could be *functionality increment* which is generated at the end of iteration.

Furthermore, according to (*SWEBOK*, 2004) at least the first three topics (but we can add and the forth one, too) are not disjoint but rather they represent distinct concerns. For example, an OO method may incorporate formal techniques and rely on prototyping for verification and validation. As methodologies continuously evolve, the SWEBOK 2004 tried as hard as possible to avoid naming particular methodologies, but new version is likely to make an exception when it comes to the *agile methods*, as the new version shortly describes *Pair programming, Rapid application development, eXtreme programming, Scrum and Feature-driven development.* Of course these are not the only agile methodologies, but according to (*SWEBOK V3 - Chapter 10*, 2012) they are the most popular ones. Finally, in the *body of knowledge* it is stated that the choice of the appropriate method could have a dramatic effect on the success of the software project.

Every methodological framework is based on some approaches or paradigms (basic model, the development cycle, the relationship of existing and future systems…) and it describes or prescribes a pattern of the development cycle, development activities and artifacts. Thus, the line between methodologies and approaches is a thin one and is often crossed by many authors, teams and organizations. That is the reason why there is no clear division between methodologies and approaches. Even Olle et al. back in (1988) pointed out that the term 'methodology' is not correctly used. Original meaning of 'a study of method' was replaced in common practice with 'method' and such practice remained till today and is followed in this dissertation as well. In general, adopting the definition from (Avison and Fitzgerald, 1988) in

this thesis, methodology will be considered as "*a collection of procedures, techniques, tools and documentation aids which will help the systems developers in their efforts to implement a new information system.*" Approach will simply be used to define the basic artifacts while conducting the chosen methodology.

2.2.2. Overview of methodologies targeting development of mobile applications

In accordance with the current state-of-the-art stream, the development of mobile applications and systems differs from traditional software development in many aspects, as it should satisfy special requirements and constraints (as elaborated in chapter 1.1.1). As already stated in previous chapters some of these requirements concern portability, standards, capabilities, privacy and time-to-market requirements and therefore, the design of mobile software systems is much more complicated and is forcing developers to reconsider the use of traditional software development methodologies. Despite the mentioned problems that could be interesting for the scientific community, a relatively few researches aimed to enhance the methodologies for mobile application development, and most of the work performed in this field has been focused on the implementation-oriented aspects of the mobile software development, while methodology-oriented issues still remain to be properly addressed (Rahimian and Ramsin, 2008). Additionally, development of mobile systems is a challenging task with a high level of uncertainty, and according to Hosbond (2005), some of the important problems are rapid technology development, lack of standardization and short time-to-market. Hosbond identified that there are two important sets of challenges that should be addressed in the domain of mobile systems development, and these are business related challenges (e.g. tough competition, conflicting customer interests, establishment of revenue-share models etc.) and development specific challenges (e.g. rapidly changing technology, lack of standardization, integration with existing systems etc.).

Reviewing the existing solutions for mobile application development, we should mention the Abrahamsson et al. (2004) and their Mobile-D methodology as an agile approach to mobile application development which is based on combination of eXtreme programming in terms of practices, Crystal family of methodologies in terms of scalability and Rational Unified Process in terms of life-cycle coverage (Supan et al., 2013). Initially, as introduced in (Abrahamsson et al., 2004), the methodology is composed of five iterations i.e. phases: *set-up, core, core2, stabilize* and *wrap-up*. According to technical documents available on the authors' web site, for example (Salo and Koskela, 2004), the methodology included 34 principal inputs and outputs (like *action point list, architecture line plan, base process description, daily status report* etc.) and 9 different roles (like *customer group, exploration team, project team, steering group*, etc.).

The method evolved and according to presently available documents such as web application presenting the methodology (VTT Technical Research Centre of Finland, 2006a) and set of documents and templates describing the methodology in detail (VTT Technical Research Centre of Finland, 2006b) the main phases, activities and tasks are presented in Table 6.

Table 6 - Mobile-D phases, activities and tasks

Mobile-D Phases	Development days / Activities	Tasks
Explore	Stakeholder establishment	Customer establishment
		Stakeholder group establishment
	Scope definition	Initial requirements collection
		Initial project planning
	Project establishment	Environment selection
		Personnel allocation
		Architecture line definition
		Process establishment
Initialize	Project set-up	Environment setup
		Training
		Customer communication establishment
	Planning day in 0 iteration	Architecture line planning
		Initial requirements analysis
	Working day in 0 iteration	
	Release day in 0 iteration	
Productionize	Planning day	Post-iteration workshop
		Requirements analysis
		Iteration planning
		Acceptance test generation
		Acceptance test review
	Working day	Wrap-up
		Test-driven development
		Pair programming
		Continuous integration
		Refactoring
		Inform customer
	Release day	System integration
		Pre-release testing
		Acceptance testing
		Release ceremonies
Stabilize	Planning day	Post-iteration workshop
		Requirements analysis
		Iteration planning
		Acceptance test generation
		Acceptance test review
	Working day	Wrap-up
		Test-driven development
		Pair programming
		Continuous integration
		Refactoring
		Inform Customer
	Documentation wrap-up	
	Release day	System integration
		Pre-release testing
		Acceptance testing
		Release ceremonies

System test & fix	System test	System test
	Planning day	Post-iteration workshop
		Requirements analysis
		Iteration planning
		Acceptance test generation
		Acceptance test review
	Working day	Wrap-up
		Test-driven development
		Pair programming
		Continuous integration
		Refactoring
		Inform customer
	Release day	System integration
		Pre-release testing
		Acceptance testing
		Release ceremonies

Source: (VTT Technical Research Centre of Finland, 2006a)

The practices included in execution of tasks during different phases and activities comprise nine principal elements which are mainly well-known agile practices specialized for mobile software development (Abrahamsson et al., 2004; VTT Technical Research Centre of Finland, 2004):

- **Phasing and pacing** – The projects are performed in iterations of which each begins with a Planning Day
- **Architecture Line** – Architecture line approach is utilized together with architectural patterns and Agile Modeling
- **Mobile Test Driven** Development – Test-first approach is utilized together with automated test cases
- **Continuous Integration** – Effective Software Change Management (SCM) practices are applied through multiple means
- **Pair Programming** – Coding, testing and refactoring are carried out in pairs
- **Metrics** – Few essential metrics are collected rigorously and utilized for feedback and process improvement purposes
- **Agile Software Process Improvement** – Post-Iteration workshops are used to continuously improve the development process
- **Off-Site Customer** – Customer participates in Planning and Release Days
- **User-Centered Focus** – Emphasis is placed on identifying and fulfilling end-user needs

Additionally, a Hybrid Method Engineering Approach was used by Rahimian and Ramsin (2008) to develop "the ideal software development methodology" named *Agile Risk-based Methodology*. The authors utilized general agile practices through New Product Development (NPD) approach and incorporated the ideas from Adaptive Software Development (ASD).

Although the part of methodology development process was based on artifact-oriented approach, this methodology is defined at the level of activity and additional research should be performed to specify the finer-grained tasks of the process (Supan et al., 2013).

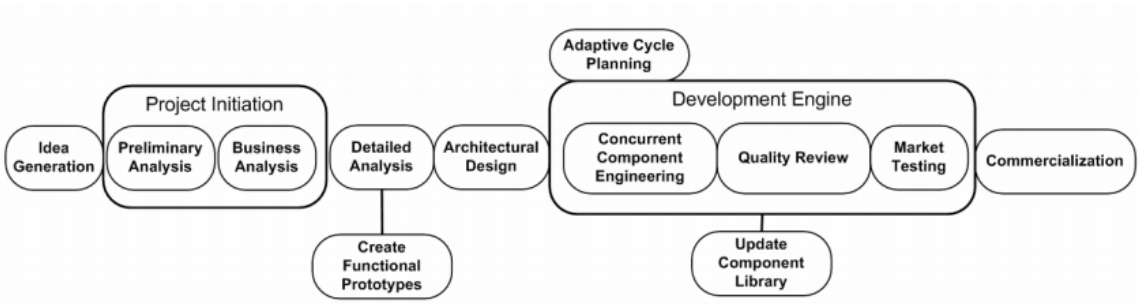


Figure 12 - Agile Risk-based Methodology (Rahimian and Ramsin, 2008)

Another methodology developed for mobile software development is MASAM (Mobile Application Software Development Method). MASAM methodology is created by Jeong et al. (2008) and it represents the proprietary methodology that was built in on the top of Software and Systems Process Engineering Meta-model (SPEM) framework.

Being based on SPEM, the MASAM is defined on three different kinds of process assets: *roles*, *tasks* and *work products.* A *role* defines a set of related skills, competencies or responsibilities (e.g. planner, manager, UI designer, developer etc.), a *task* is an assignable unit of work (e.g. initial planning, initial analysis, UI design etc.) and *work product* stands for task inputs and outputs (e.g. product summary, UI sample, task card etc.).

This agile methodology is comprised of *Development preparation phase*, *Embodiment phase*, *Product development phase* and *Commercialization phase*. The methodology defines activities and tasks for each of the four mentioned phases, as shown in Table 7.

Table 7 - MASAM methodology phases, activities and tasks

MASAM Phase	Activity	Task
Development preparation	Grasping product	Defining product summary
		Pre-planning
	Product concept sharing	User definition
		Initial product analysis
	Project Set-up	Development process coordination
		Project resource coordination
		Pre study
Embodiment	User needs understanding	Story-card workshop
		UI design
	Architecting	Non-functional requirements analysis
		Architecture definition
		Pattern management
Product development	Implementation preparation	Environment setup
		Development planning
	Release Cycle	Release planning

		Iteration cycle Iteration planning Implementation cycle Face-to-face meeting Incremental design Test Driven Development (TDD) Refactoring Pair programming Continuous integration Feedback
		Release Acceptance test Feedback
Commercialization	System Test	Acceptance test
		User test
	Product Selling	Launching test
		Product launching

Source: (Jeong et al., 2008)

To conclude, except (a) applying newly developed methods there are two other options. The company can (b) adopt and use an existing development methodology or (c) can adapt an existing development methodology to fit the specific organizational culture, company's goals and specific requirements of mobile application development. In any case, it is important to notice that implementation of the new methodological framework is a serious challenge from organizational, technical, educational and every other point of view. In fact, it is about the implementation of a new development system. Although the analysis that would cover all these concerns is out of scope of this work, the adoption or adaption of a methodology for the development of mobile applications should not be considered as an easy task and if performed, should be backed up with serious preliminary research and carefully made decisions.

This short review does not cover all methodologies, but based on this preliminary review we can conclude that the authors do agree on several facts that are important for this dissertation. (1) The development for mobile devices differs from standard development, (2) the agile approach is widely used in methodologies for mobile devices and (3) neither one of the presented methodologies is applicable without additional efforts to make the process more fine-grained or more suitable to specific development environment and mobile application requirements.

2.2.3. Identification of the need for a review

Preliminary research on the software development methodologies, presented in the previous chapters can lead us to several important conclusions. Firstly, the field of software development, during its 50-year-old history, has been interwoven with many different

software development methodologies and approaches. This also resulted in the terminology confusion as many authors mix different concepts such as methodology, approach, framework and process. Secondly, there are some attempts to create specific software development methodology that would be suitable for development of mobile applications. Surprisingly, these attempts are relatively rare, they are not aligned with the current mobile development demands which have slightly but seriously changed, especially after the introduction of the mobile application stores back in 2009, and finally some of these methodologies are still not usable in practice as being defined at relatively high level of abstraction. Thirdly, many companies have chosen to use the existing and familiar development methodologies while developing mobile applications. The trends show that agile approach is most suitable and widely used when developing mobile applications (Abrahamsson et al., 2003; Holler, 2006), but still, some companies have considerable heritage in using non-agile approaches which they still find as the most suitable.

The number and complexity of different possibilities indicate that a thorough and unbiased research method such is systematic literature review is needed in order to get the overall overview of possible methodologies that could be taken while developing applications for mobile devices.

Additionally, the preliminary research is performed to identify the existing systematic literature reviews on software development methodologies for development of mobile applications. The IEEExplore, ACM Digital library, INSPEC, CiteSeerX and GoogleScholar databases were searched by the following search query: `("literature review" OR SLR) AND (mobile development)`[17].

Almost all obtained papers[18] were excluded as not being literature reviews or not being literature reviews in mobile applications development. Only one paper (Hosbond and Nielsen, 2005) passed the inclusion criteria, but the focus of the SLR performed in this paper was to review the literature in the domain of four mobile systems development perspectives (requirements, technology, application, business) but unfortunately did not include methodologies or approaches to be used when developing mobile applications.

[17] This query implicitly includes „systematic literature review“ phrase. Additionally, more rigorous search querries, like ("literature review" OR SLR) AND (mobile development methodologies) or similar have been discarded as returning only a few or no results.

[18] The search returned following number of papers: IEEEExplore (61), ACM Digital library (624), INSPEC (62), CiteSeerX (22) and GoogleScholar (128). Additionally, the original query on GoogleScholar returned more than 22.300 results, so there was used a narrower concept serching for „mobile development“ as a phrase instead of searching for both words independently as in other databases.

To conclude, according to information available in the mentioned databases, there are no existing systematic literature reviews covering the subject of software development methodologies for mobile applications development, which makes the need for such review even bigger. As an additional proof of this claim, the results of SLRs on *Systematic Literature Reviews in Software Engineering* presented in (B Kitchenham et al., 2009) and in (Kitchenham et al., 2010) show that no literature reviews were conducted in the domain of software development methodologies or software development methodologies for mobile devices.

2.2.4. Specifying the research questions

In the previous chapter we discussed the results of preliminary researches performed in order to identify possible mobile application development methodologies and on existing SLRs identified the need for the systematic literature review. In order to address the issues determined in this analysis, this systematic review is aligned to answer the following research questions:

RQ1 – What development methodologies and approaches are reported in literature as defined in theory or used in practice for mobile application development?

RQ2 – Are the identified methodologies and approaches applicable for multi-platform mobile applications development?

Motivation for RQ1 is to identify all existing methodologies and approaches for development of mobile applications and motivation for RQ2 is to define a set of methodologies and approaches that could be used for multi-platform mobile applications development.

With respect to RQ1, several important decisions were made. Firstly, as preliminary research showed, and thus assuming that there are not so many publications in this field, it is decided not to apply any time filters on the source publications. The fields of software development methodologies and especially methodologies for development of mobile applications are considered to be young disciplines and additional time constraints are not necessary. Secondly, it is important to clearly distinguish methodologies and approaches according to definitions presented in chapter 2.2.1. Finally, only methodologies and approaches reported to be used for development of mobile applications and mobile systems should be taken as relevant and potentially selected for review.

With respect to RQ2, as methodologies or approaches by definition are not platform dependent, it is important to notice that simple decision parameters will be taken into consideration in order to determine if identified development methodologies and approaches are applicable for multi-platform mobile applications development. Actually, we assume that there might be some methodologies and approaches reported to be developed for specific

mobile target platform/s and only these methodologies or approaches (at least unchanged) will be considered as not applicable for multi-platform mobile applications development. Secondly, RQ2 is important for the other research activities in this thesis, as only the applicable methodologies could be used in the following research phases.

Although there are multiple motivations for performing this literature review, both research questions are defined with the purpose of identifying the existing body-of-knowledge basis for choosing one mobile application development methodology and one development approach that will be used in the subsequent research phases performed in this dissertation project. In order to clarify these research questions the following complementary questions are defined:

- Is the paper reporting on a software development methodology or a development approach?
- Is the reported methodology/approach properly defined with clear phases, activities, tasks, roles, inputs and outputs?
- Are there any specific instructions on how to apply the methodology/approach?
- Are there any specific techniques reported to be used while applying the methodology or approach?
- Are there any specific instructions on any organizational aspects of teams applying the methodology/approach?
- Is the methodology/approach developed for any specific mobile target platform?

Only the last complementary question targets RQ2, while all other stated complementary questions target RQ1.

2.2.5. Developing a review protocol

The review protocol defining this research is created according to instruction presented in the previous chapters. Additionally, the template used for protocol creation is proposed by (Biolchini et al., 2005) and further explained by (Mian et al., 2005).

The protocol is firstly defined during the phase of review planning, but due to the characteristic of some protocol elements to present final or intermediate results, the information on these elements is inserted in subsequent phases of the systematic literature review.

Additionally, it is important to mention, that some protocol elements like *keywords and synonyms* and *search strings* are piloted either by using English dictionary and reading the literature (in case of synonyms definition) or by performing a pilot database search (in case of search strings definition). Final version of the protocol is presented in Table 8.

Table 8 - The review protocol

1. Question formularization	
1.1. Question focus	To identify software development methodologies and approaches that could be used for multi-platform mobile applications development.
1.2. Question quality and amplitude	**Problem:** Development of mobile applications differs from development of traditional desktop or web applications. Not all software development methodologies are used for development of mobile applications. Special problem is fragmentation of mobile platforms and devices, and thus the development process should be performed more than once. None of the existing approaches to solve this problem is good enough. This research has the idea to approach the problem differently and to define methodological interoperability, i.e. interoperability on highest, methodology level. In order to do that, it is necessary to identify applicable software development methodologies and approaches that could be used in multi-platform mobile applications development. **Research questions**: RQ1: What development methodologies and approaches are reported in literature as defined in theory or used in practice for mobile application development? RQ2: Are the identified methodologies and approaches applicable for multi-platform mobile applications development? **Keywords and synonyms**: • mobile • *software development:* system development, application development, program development • *methodology:* method, approach, framework, process, procedure, model **Intervention**: Software development methodologies and approaches for mobile applications development. **Effect**: Identification of methodologies and approaches for multi-platform mobile applications development. **Control**: Methodologies defined in previous chapters. **Outcome measure**: Cardinality of identified set of methodologies. **Population**: Publications reporting on intervention and containing defined keywords. **Application**: Subsequent research in this thesis, mobile applications development companies, researchers. **Experimental design**: Statistical method will not be applied.

2. Sources selection	
2.1. Sources selection criteria definition	Sources recommended by field experts (i.e. Brereton et al. (2007), Hannay et al. (2007), Kitchenham and Charters (2007)) and enumerated in previous chapters will be included in the search process. The criteria for sources selection used by field experts are based on *source quality and overall recognition in the software engineering community*.
2.2. Studies languages	English
2.3. Sources identification	**Sources search methods:** Research through web search engines and manual search. **Search string**: *(mobile AND ("software development" OR "system development" OR "application development" OR "program development") AND (methodology OR method OR approach OR framework OR process OR procedure OR model))* **Sources list:** Relevant electronic sources in the field of Software Engineering identified by Brereton et al. (2007): 1. IEEExplore 5. INSPEC 2. ACM Digital library 6. ScienceDirect 3. Google Scholar 7. EI Compendex (not available) 4. CiteSeerX library Special focus will be put on following combined list of relevant journals and proceedings in the field of software engineering which is based on lists given by Hannay et al. (2007) and by Kitchenham and Charters (2007). Hannay et. al. explicitly state that journals and conferences chosen by them were chosen because they were considered to be leaders in software engineering in general and empirical software engineering in particular: • ACM Transactions on Software Engineering Methodology (TOSEM) • ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) • Empirical Software Engineering (EMSE) in SpringerLink (manual search) • Evaluation and Assessment in Software Engineering (EASE) in ScienceDirect • IEEE Computer • IEEE Software • IEEE Transaction on Software Engineering (TSE)

	• Information and Software Technology (IST) in ScienceDirect • International Conference on Software Engineering (ICSE) in ACM Digital Library and IEEExplore • Journal of Software: Evolution and Process (JSEP) in Wiley (manual search) • Journal of Software: Practice and Experience (SP&E) in Wiley (manual search) • Journal of Systems and Software (JSS) in ScienceDirect If some of the mentioned journals and conference proceedings are not included in the databases of the enumerated search engines, they will be searched manually.
2.4. Sources selection after evaluation	All sources listed in 2.3 satisfied quality criteria.
2.5. References checking	Sources are defined on basis of recommendations of field experts. The final list of selected sources is also approved by two supervisors.
3. Studies selection	
3.1. Studies definition	**Studies inclusion and exclusion criteria:** The primary studies describing software development methodology or approach in theory or reporting their usage in practice will be included in review process. The studies that do not provide sufficient information on the phases, activities, tasks, roles, inputs and outputs (i.e. document templates, expected results, task prerequisites etc.) will be excluded from the review. **Studies type definition:** No filter on type of studies will be applied. All kinds of studies related to the research topic will be selected. **Procedures for studies selection:** After performing an automated search based on defined keywords and search string, initial set of potential studies for inclusion will be obtained. The studies will be firstly filtered by applying inclusion criteria on the study title. The studies that meet inclusion criteria along with those with unclear or indistinct title will be included in second phase. Second phase will apply inclusion criteria on the abstract. If abstract will be unclear or fuzzy, the introduction and conclusion will also be taken in consideration. Studies that will finally be included will be reviewed in detail by reading the full text. At last, if necessary, exclusion criteria will be applied based on information obtained from full text review.
3.2. Selection execution	**Initial studies selection:** The complete list of selected studies can be found in chapter 2.3.2 (Table 12 on page 66 of this document).

	Studies quality evaluation: The list of studies that passed inclusion and exclusion criteria can be found in chapter 2.3.3 on page 68 of this document. **Selection review:** Study selection process was reviewed and approved by two supervisors and one of them is field expert.
4. Information extraction	
4.1. Information inclusion and exclusion criteria definition	The extracted information from studies must contain theoretical or practical description of phases that should be performed during the development process according to focused methodology. If studies are reporting new software development approach, then the main characteristics, values and rules which define focused approach should be contained in extracted information.
4.2. Data extraction forms	The template form for data extraction that is defined for this review can be found in chapter 2.3.4 on page 70, and complete list of filled data extraction forms on all selected primary studies can be found in Appendix D on page 265.
4.3. Extraction execution	The results of objective (study identification, study methodology, study results and study problems) and subjective (information through the authors and general impressions and abstractions) data extraction are presented in chapter 2.3.4 on page 70.
4.4. Resolution of divergences among reviewers	There were no divergences, as the extraction was performed only by one author, i.e. author of this thesis.
5. Result summarization	
5.1. Results statistical calculus	Statistical calculi were not used.
5.2. Results presentation in tables	The final results are presented in tables with the following information. • Studies reporting the creation of new methodology or approach • Studies reporting the methodology or approach usage • Methodologies/approaches not eligible for multiplatform development • Methodologies/approaches targeting specific mobile applications The stated tables with final reported results could be found in chapter 2.3.5 on page 71.
5.3. Sensitivity	There was no need for sensitivity analysis.

analysis	
5.4. Plotting	There was no need for plotting.
5.5. Final comments	**Number of studies obtained:** 6761 **Number of relevant studies:** 49 **Results application:** Mobile-D methodology supported by Test Driven Development is selected for application in this research. **Recommendations:** Identified methodologies could be separately analyzed in order to determine their quality and applicability. This was not the focus of this study.

2.2.6. Evaluating the review protocol

The review protocol is evaluated by two supervisors of this thesis project. Also, it is important to mention that one of the supervisors (prof. Strahonja) is an expert with scientific and empirical background in the field of software development methodologies. Some minor requests stated by both supervisors, regarding sources identification and final reporting were taken in consideration and implemented in the final version of the review protocol.

2.3. Conducting the review

2.3.1. Identification of research

The research is focused on the identification of software development methodologies and approaches that could be used for multi-platform mobile applications development. In order to identify primary studies relevant to the stated research questions, the following keywords with the list of relevant synonyms are used:

Table 9 - Search keywords and synonyms

Keyword	Synonyms
mobile	-
software development	system development application development program development
methodology	method approach framework process procedure model

The stated list of synonyms is created according to the results of preliminary literature review and is based on the empirical knowledge of terms used in the software engineering literature.

The target population consists of the publications reporting the software development methodologies and approaches for mobile applications development containing the defined keywords. In order to identify the initial list of publications, the search engines and manual search have been used. The following query is defined for automatic database search:

(mobile AND ("software development" OR "system development" OR "application development" OR "program development") AND (methodology OR method OR approach OR framework OR process OR procedure OR model))

The presented query has been executed on the databases and the relevant journals and proceedings in the field of software engineering which are recommended by the filed experts Brereton et al. (2007), Hannay et al. (2007), Kitchenham and Charters (2007) and as elaborated in chapter 2.1.2.2. The final list of relevant sources is given in the Table 10.

Table 10 - The list of relevant sources

Relevant databases	
IEEExplore	INSPEC
ACM Digital Library	ScienceDirect
Google Scholar	EI Compendex (excluded)
CiteSeerX library	
Relevant journals and proceedings	
ACM Transactions on Software Engineering Methodology (TOSEM)	ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)
Empirical Software Engineering (EMSE) in SpringerLink	Evaluation and Assessment in Software Engineering (EASE) in ScienceDirect
IEEE Computer	IEEE Software
IEEE Transaction on Software Engineering (TSE)	Information and Software Technology (IST) in ScienceDirect
International Conference on Software Engineering (ICSE) in ACM Digital Library and IEEExplore	Journal of Software: Evolution and Process (JSEP) in Wiley
Journal of Software: Practice and Experience (SP&E) in Wiley	Journal of Systems and Software (JSS) in ScienceDirect

The preliminary research showed that majority of mentioned journals and proceedings is indexed in the stated electronic databases, and manual search has been performed only on the following databases:

- Empirical Software Engineering (EMSE) in SpringerLink
- Journal of Software: Evolution and Process (JSEP) in Wiley
- Journal of Software: Practice and Experience (SP&E) in Wiley

Additionally, despite the best efforts, the access to the electronic database EI Compendex is available neither at the University of Alcalá nor at the University of Zagreb, and thus, this

database had to be excluded from the list. So the final list of the excluded databases includes only:

- EI Compendex

As it can be seen from the final set of relevant sources, the focus of this research is only on the scientific research community. This is mainly due to the time and "personnel" constraints. The past showed that the industry, as a source of development methodologies should not be neglected and we strongly recommend that white papers, technical reports and other unpublished materials should also be included in the future similar literature reviews.

2.3.2. Selection of primary studies

The primary studies describing software development methodology or approach in theory or reporting their usage in practice have been included in the review process. The studies that do not provide sufficient information on the phases, activities, tasks, roles, inputs and outputs (i.e. document templates, expected results, task prerequisites etc.) have been excluded from the review. The type of studies has not been filtered and all kinds of studies related to the research topic that have been found by the search have been considered for possible inclusion.

2.3.2.1. Applied procedures in selection process

After the automated search based on defined keywords and search string is performed, the initial set of the potential studies for inclusion is obtained (see Table 11). The studies are firstly filtered by applying inclusion criteria on the study title. The studies that met the inclusion criteria along with those with unclear or indistinct title are included in the second phase where the inclusion criteria were applied on the abstract. Some of the abstracts were unclear and fuzzy, and in those cases the introduction and conclusion were also taken into consideration. The final phase conducted on the included studies was performed by a detailed analysis and full text reading. During this phase, the exclusion criteria were applied based on the information obtained from full text review.

As it can be seen in Table 11, in total 6761 initial studies were obtained by automatically performed database searches. The search of Google Scholar database had to be performed with specific time constraints, as it was impossible to reach all results given by the original search query. This was not the only problem faced during the research process, but the faced problems will be discussed in later chapter. Apart from Google, some other database engines also had to be parameterized, and the used parameters, date ranges, filters and search execution date are all reported in Table 11.

Table 11 - Applied procedures in selection process

Database	Search query	Date range / other filters	Date of search	No. of results
IEEE Xplore ®	("mobile application" OR "mobile development") AND ("software development" OR "system development" OR "application development" OR "program development") AND (methodology OR method OR approach OR framework OR process OR procedure OR model)	-	05.06.2012.	68
ACM Digital Library		Searched journals, proceedings and transactions	06.06.2012.	335
CiteSeerX		Citations included	07.06.2012.	55
INSPEC		-	07.06.2012.	85
ScienceDirect		Searched fields: Computer Science, Engineering, Social Sciences	07.06.2012.	399
Google Scholar		Full text search 19xx – 2004	08.06.2012.	867
Google Scholar		Full text search; 2005 – 2006	08.06.2012.	661
Google Scholar		Full text search; 2007 – 2008	08.06.2012.	925
Google Scholar		Full text search; 2009	09.06.2012.	694
Google Scholar		Full text search; 2010	09.06.2012.	868
Google Scholar		Full text search; Filter: "+*phone*" 2011	11.06.2012.	923
Google Scholar		Full text search; Filter: "-*phone*" 2011	11.06.2012.	352
Google Scholar		Full text search; 2012	12.06.2012.	529
Manual search of Journals	Performed by reading paper titles and abstracts	2007-2012	13. - 15.06.2012.	0
Total				**6761**

The full list of all obtained papers is kept only in the reference management software, but the lists of the identified studies after applying inclusion criteria on the study title and after applying inclusion criteria on the abstract are documented in the annexes of this document (see Appendix A and Appendix B). The full text documents are obtained for almost all studies included in the second identification phase and are also stored in the reference management software. Additionally, the reference management software contains the exclusion reasons for all studies excluded in the second and the third iteration. Finally, the list of all studies considered to be relevant and included in the literature review process results is given in Table 12.

Table 12 - The list of relevant studies

Study identifier	Study
(Abrahamsson et al., 2005b)	Abrahamsson, P., Hanhineva, A., Jäälinoja, J., 2005. Improving business agility through technical solutions: A case study on test-driven development in mobile software development, in: Business Agility and Information Technology Diffusion. Presented at the IFIP TC8 WG 8.6 International Working Conference.
(Abrahamsson et al., 2009)	Abrahamsson, P., Ihme, T., Kolehmainen, K., Kyllönen, P., Salo, O., 2009. Mobile-D for Mobile Software: How to Use Agile Approaches for the Efficient Development of Mobile Applications.
(Abrahamsson et al., 2004)	Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., Koskela, J., Kyllönen, P., Salo, O., 2004. Mobile-D: an agile approach for mobile application development, in: Companion to the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages, and Applications, OOPSLA '04. ACM, New York, NY, USA, pp. 174–175.
(Alyani and Shirzad, 2011)	Alyani, N., Shirzad, S., 2011. Learning to innovate in distributed mobile application development: Learning episodes from Tehran and London, in: 2011 Federated Conference on Computer Science and Information Systems (FedCSIS). Presented at the 2011 Federated Conference on Computer Science and Information Systems (FedCSIS). IEEE., Piscataway, NJ, USA, pp. 497–504.
(Barnawi et al., 2012)	Barnawi, A., Qureshi, M., Khan, A.I., 2012. A Framework for Next Generation Mobile and Wireless Networks Application Development using Hybrid Component Based Development Model. Arxiv preprint arXiv:1202.2515.
(Bergström and Engvall, 2011)	Bergström, F., Engvall, G., 2011. Development of handheld mobile applications for the public sector in Android and iOS using agile Kanban process tool.
(Binsaleh and Hassan, 2011)	Binsaleh, M., Hassan, S., 2011. Systems Development Methodology for Mobile Commerce Applications: Agile vs. Traditional. International Journal of Online Marketing (IJOM) 1, 33–47.
(Biswas et al., 2006)	Biswas, A., Donaldson, T., Singh, J., Diamond, S., Gauthier, D., Longford, M., 2006. Assessment of mobile experience engine, the development toolkit for context aware mobile applications, in: Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, ACE '06. ACM, New York, NY, USA.
(Charaf, 2011)	Charaf, H., 2011. Developing Mobile Applications for Multiple Platforms, in: Engineering of Computer Based Systems (ECBS-EERC), 2011 2nd Eastern European Regional Conference on The. p. 2.
(Chen, 2004)	Chen, M., 2004. A methodology for building mobile computing applications. International journal of electronic business 2, 229–243.
(Cuccurullo et al., 2011)	Cuccurullo, S., Francese, R., Risi, M., Tortora, G., 2011. A Visual Approach supporting the Development of MicroApps on Mobile Phones, in: Proc. of 3rd International Symposium on End-User Development. Presented at the 3rd International Symposium on End-User Development, Brindisi, Italy, pp. 289–294.
(Ejlersen et al., 2008)	Ejlersen, A., Knudsen, M.S., Løvgaard, J., Sørensen, M.B., 2008. Using Design Science to Develop a Mobile Application.
(Forstner et al., 2005)	Forstner, B., Lengyel, L., Kelenyi, I., Levendovszky, T., Charaf, H., 2005. Supporting Rapid Application Development on Symbian Platform, in: Computer as a Tool, 2005. EUROCON 2005.The International Conference On. pp. 72 –75.
(Forstner et al., 2006)	Forstner, B., Lengyel, L., Levendovszky, T., Mezei, G., Kelenyi, I., Charaf, H., 2006. Model-based system development for embedded mobile platforms, in: Model-Based Development of Computer-Based Systems and Model-Based Methodologies for Pervasive and Embedded Software, 2006. MBD/MOMPES 2006. Fourth and Third International Workshop On. p. 10–pp.
(Gal and Topol, 2005)	Gal, V., Topol, A., 2005. Experimentation of a Game Design Methodology for Mobile Phones Games.
(Hedberg and Iisakka, 2006)	Hedberg, H., Iisakka, J., 2006. Technical Reviews in Agile Development: Case Mobile-D, in: Quality Software, 2006. QSIC 2006. Sixth International Conference On. pp. 347–353.
(Ihme and Abrahamsson, 2005)	Ihme, T., Abrahamsson, P., 2005. The Use of Architectural Patterns in the Agile Software Development of Mobile Applications.
(Jeong et al., 2008)	Jeong, Y.J., Lee, J.H., Shin, G.S., 2008. Development Process of Mobile Application SW Based on Agile Methodology, in: Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference On. pp. 362–366.
(Kaariainen et al., 2004)	Kaariainen, J., Koskela, J., Abrahamsson, P., Takalo, J., 2004. Improving requirements management in extreme programming with tool support - an improvement attempt that failed, in: Euromicro Conference, 2004. Proceedings. 30th. pp. 342 – 351.
(Khambati et al., 2008)	Khambati, A., Grundy, J., Warren, J., Hosking, J., 2008. Model-Driven Development of Mobile Personal Health Care Applications, in: Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering, ASE '08. IEEE Computer Society, Washington, DC, USA, pp. 467–470.
(Kim, 2008)	Kim, H.K., 2008. Frameworks of Process Improvement for Mobile Applications. Engineering Letters 16.
(Kim et al., 2009)	Kim, H., Choi, B., Yoon, S., 2009. Performance testing based on test-driven development for mobile applications, in: Proceedings of the 3rd International Conference on Ubiquitous

	Information Management and Communication, ICUIMC '09. ACM, New York, NY, USA, pp. 612–617.
(Korkala and Abrahamsson, 2004)	Korkala, M., Abrahamsson, P., 2004. Extreme programming: Reassessing the requirements management process for an offsite customer. Software Process Improvement 12–22.
(Maharmeh and Unhelkar, 2009)	Maharmeh, M., Unhelkar, B., 2009. A Composite Software Framework Approach for Mobile Application Development. Handbook of research in mobile business: technical, methodological, and social perspectives 194.
(Maia et al., 2010)	Maia, M.E.F., Celes, C., Castro, R., Andrade, R.M.C., 2010. Considerations on developing mobile applications based on the Capuchin project, in: Proceedings of the 2010 ACM Symposium on Applied Computing, SAC '10. ACM, New York, NY, USA, pp. 575–579.
(Manjunatha et al., 2010)	Manjunatha, A., Ranabahu, A., Sheth, A., Thirunarayan, K., 2010. Power of clouds in your pocket: An efficient approach for cloud mobile hybrid application development, in: Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference On. pp. 496–503.
(Marinho et al., 2012)	Marinho, F.G., Andrade, R.M.C., Werner, C., Viana, W., Maia, M.E.F., Rocha, L.S., Teixeira, E., Filho, J.B.F., Dantas, V.L.L., Lima, F., Aguiar, S., 2012. MobiLine: A Nested Software Product Line for the domain of mobile and context-aware applications. Science of Computer Programming
(Nyström, 2011)	Nyström, A., 2011. Agile Solo - Defining and Evaluating an Agile Software Development Process for a Single Software Developer.
(Ortiz and Prado, 2010)	Ortiz, G., Prado, A.G.D., 2010. Improving device-aware Web services and their mobile clients through an aspect-oriented, model-driven approach. Information and Software Technology 52, 1080 – 1093.
(Pauca and Guy, 2012)	Pauca, V.P., Guy, R.T., 2012. Mobile apps for the greater good: a socially relevant approach to software engineering, in: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, SIGCSE '12. ACM, New York, NY, USA, pp. 535–540.
(Rahimian and Ramsin, 2008)	Rahimian, V., Ramsin, R., 2008. Designing an agile methodology for mobile software development: A hybrid method engineering approach, in: Research Challenges in Information Science, 2008. RCIS 2008. Second International Conference On. pp. 337–342.
(Rosa and Lucena,Jr., 2011)	Rosa, R.E.V.S., Lucena,Jr., V.F., 2011. Smart composition of reusable software components in mobile application product lines, in: Proceedings of the 2nd International Workshop on Product Line Approaches in Software Engineering, PLEASE '11. ACM, New York, NY, USA, pp. 45–49.
(Rupnik, 2009)	Rupnik, R., 2009. Mobile Applications Development Methodology, in: Unhelkar, B. (Ed.), Handbook of Research in Mobile Business: Technical, Methodological, and Social Perspectives. IGI Global Snippet.
(Saifudin et al., 2011)	Saifudin, A.W.S.N., Salam, B.S., Abdullah, C.M.H.L., 2011. MMCD Framework and Methodology for Developing m-Learning Applications. Presented at the International conference on Teaching & Learning in Higher Education (ICTLHE 2011).
(Salo, 2004)	Salo, O., 2004. Improving software process in agile software development projects: results from two XP case studies, in: Euromicro Conference, 2004. Proceedings. 30th. pp. 310–317.
(Scharff, 2010)	Scharff, C., 2010. The Software Engineering of Mobile Application Development.
(Scharff, 2011)	Scharff, C., 2011. Guiding global software development projects using Scrum and Agile with quality assurance, in: Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference On. pp. 274–283.
(Scharff and Verma, 2010)	Scharff, C., Verma, R., 2010. Scrum to support mobile application development projects in a just-in-time learning context, in: Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, CHASE '10. ACM, New York, NY, USA, pp. 25–31.
(Schwieren and Vossen, 2009)	Schwieren, J., Vossen, G., 2009. A design and development methodology for mobile RFID applications based on the ID-Services middleware architecture, in: Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference On. pp. 260–266.
(Shiratuddin and Sarif, 2008)	Shiratuddin, N., Sarif, S.M., 2008. m d-Matrix: Mobile Application Development Tool. Proceedings of the International MultiConference of Engineers and Computer Scientists 1.
(Shiratuddin and Sarif, 2009)	Shiratuddin, N., Sarif, S.M., 2009. Construction of Matrix and eMatrix for Mobile Development Methodologies, in: Handbook of Research in Mobile Business: Technical, Methodological, and Social Perspectives. IGI Global, pp. 113–126.
(Su and Scharff, 2010)	Su, S.H., Scharff, C., 2010. Know Yourself and Beyond: A Global Software Development Project Experience with Agile Methodology, in: Proceedings of Student-Faculty Research Day, CSIS. Pace University.
(Thompson et al., 2010)	Thompson, C., White, J., Dougherty, B., Turner, H., Campbell, S., Zienkiewicz, K., Schmidt, D.C., 2010. Model-Driven Architectures for Optimizing Mobile Application Performance.
(Um et al., 2005)	Um, J., Hong, S., Kim, Y.T., Chung, E., Choi, K.M., Kong, J.T., Eo, S.K., 2005. ViP: A Practical Approach to Platform-based System Modeling Methodology. Journal of Semiconductor Technology and Science 5, 89.
(Walkerdine et al., 2009)	Walkerdine, J., Phillips, P., Lock, S., 2009. A Tool Supported Methodology For Developing Secure Mobile P2P Systems, in: Mobile Peer-to-peer Computing for Next Generation Distributed Environments: Advancing Conceptual and Algorithmic Applications. pp. 283–301.

(Wolkerstorfer et al., 2008)	Wolkerstorfer, P., Tscheligi, M., Sefelin, R., Milchrahm, H., Hussain, Z., Lechner, M., Shahzad, S., 2008. Probing an agile usability process, in: CHI '08 Extended Abstracts on Human Factors in Computing Systems, CHI EA '08. ACM, New York, NY, USA, pp. 2151–2158.
(Xiong and Wang, 2010)	Xiong, Y., Wang, A., 2010. A new combined method for UCD and software development and case study, in: Information Science and Engineering (ICISE), 2010 2nd International Conference On. pp. 1–4.
(Zakal et al., 2011)	Zakal, D., Lengyel, L., Charaf, H., 2011. Software Product Lines-based development, in: Applied Machine Intelligence and Informatics (SAMI), 2011 IEEE 9th International Symposium On. pp. 79–81.
(Zeidler et al., 2008)	Zeidler, C., Kittl, C., Petrovic, O., 2008. An integrated product development process for mobile software. International Journal of Mobile Communications 6, 345–356.

The propagation of relevant studies through the research process is described in Table 13.

Table 13 - Propagation of relevant studies through phases

Database	Identified studies – P1	Identified studies – P2	Identified studies – P3	Relevant studies (after QA)		
	n	*n*	*n*	*n*	%''	%
IEEE Xplore ®	68	25	3	3	4.41	6.12
ACM Digital Library	335	79	13	9	2.69	18.37
CiteSeerX	55	12	0	0	0.00	0.00
INSPEC	85	39	3	1	1.18	2.04
ScienceDirect	399	26	4	2	0.50	4.08
Google Scholar 19xx - 2004	867	40	5	3	-	-
Google Scholar 2005 - 2006	661	37	8	6	-	-
Google Scholar 2006 - 2008	925	41	7	6	-	-
Google Scholar 2009	694	31	6	6	-	-
Google Scholar 2010	868	45	6	5	-	-
Google Scholar 2011a	923	29	5	4	-	-
Google Scholar 2011b	352	21	4	3	-	-
Google Scholar 2012	529	14	3	1	-	-
Google Scholar Subtotal	5819	258	44	34	0.58	69.39
Subtotal	6761	439	-		-	
Redundant studies	*NA*	75*	-		-	
Total	**6761**	**364**	**67**	**49**	**0.73**	**100**

* Google Scholar database returned some results that were previously identified in other databases.
%'' Percentage in respect to initial studies pool from the same database
% Percentage in respect to final pool of all relevant studies

As it can be seen from the presented table, 49 studies are identified as relevant which makes it only a 0.73% of initial 6761 studies. Additionally, *Science Direct* and *Google Scholar* are the databases with the biggest waste factor as more than 99.4% of all initial studies were discarded as irrelevant. Nevertheless, Google Scholar proved to give 69.39% of all relevant studies. However, one could discuss the quality of Google Scholar studies in relation to the studies obtained from other databases, but such analysis is out of the focus of this work.

2.3.3. Study quality assessment

The activities of the study quality assessment were performed carefully through the whole process of the studies' identification. As it was impossible to apply the usage of checklists on

all initially identified studies, during the first phase, the focus was put on an unbiased study selection process, while the later phases additionally included the quality assessment of the identified studies.

During the first identification phase, considerable efforts were made in order to clearly divide studies that do not have any connection with software engineering and software development from those that do. Additionally, in order to assess the quality of each primarily selected study and to make sure that the study findings are relevant and unbiased, firm criteria were established in the second and third phase. The complete overview of these criteria is given in the Table 14.

Table 14 - The criteria for unbiased study identification

Identification of studies - P1	
Inclusion	**Exclusion**
Software engineering	Other studies undoubtedly not from research domain
Software development	
Mobile development	
Other studies connected with the topic of interest	
Identification of studies – P2	
Inclusion	**Exclusion**
Reporting the methodology or approach used in development or mobile applications development	Defining frameworks for specific purposes (i.e. security, engine development etc.)
Defining framework or approach for development of mobile applications	Defining building blocks with or without specific purpose (i.e. for user interface, tracking, reporting etc.)
Defining framework or approach for specific development phases	Defining testing frameworks, toolkits or middleware…
Defining framework or approach for development of applications in specific application area	Defining frameworks for development of part of application (e.g. adding context awareness, content awareness etc.)
	Defining or reporting the usage of platforms for mobile apps development with no concerns on development process
	Other papers not connected with inclusion criteria.
Identification of studies – P3	
Inclusion	**Exclusion**
Checklist result positive	Checklist result negative

As the studies observed in this systematic review process are oriented on software development and development methodologies and approaches, they are usually not based on the usage of experimental design and statistical methods. This means that the specific quality assessment checklist applicable for studies in the domain of software engineering and particularly for this research had to be built. This checklist was created according to approach given by (Dybå and Dingsøyr, 2008b) who defined three main issues pertaining to quality that need to be considered when appraising the qualitative studies identified in the review: *rigour*, *credibility* and *relevance*. In addition to these, the advice to include the *screening criteria* is

accepted in order to assess study rationale, aims and context. The created checklist is presented in Table 15.

Table 15 - Quality assessment checklist

ID	Quality assessment question	Possible results
Q1	Study reports existing methodology or approach used in mobile application development?	Yes/No
Q2	Study defines new methodology or approach for mobile applications development?	Yes/No
Q3	Research design is appropriate to address the study context?	Yes/Partially/No
Q4	Researches have experience in software development and mobile applications development?	Yes/Partially/No
Q5	The reported or created process is clearly defined to the applicable level?	Yes/Partially/No
Q6	The study provided value for research and practice?	Yes/Partially/No

The first two questions which define the screening criteria are used as the basis for including or excluding the studies. The studies that were answered with *No* on both questions were excluded, and of the 67 papers assessed for the quality, the number of included papers for the final data extraction and synthesis was 49 (73.13%).

Subsequently, the questions labeled Q3 to Q6 aimed to assess the quality of the study and thus included the assessment of research design, the assessment of created or reported development process, the assessment of applicability of the results and finally assessment of researchers' experience. The possible answers for these questions included mark "Partially" which was given in cases when the assessed criterion was not focused in the study, but jet could not be discarded as negative. The exception is question Q4 as the experience of researchers was assessed out of the context as only few papers included written evidence on experience.

Table 16 contains an excerpt of quality assessment form as the table containing all data on performed quality assessment is given in the Appendix C.

Table 16 - Excerpt of filled quality assessment form

Study / Question	Q1	Q2	Q3	Q4	Q5	Q6	QA score
(Charaf, 2011)	Yes	No	Yes	Yes	Partially	Partially	3.0
(Alyani and Shirzad, 2011)	Yes	Yes	Partially	Yes	Partially	Partially	2.5
(Maharmeh and Unhelkar, 2009)	No	Yes	Partially	Yes	Partially	Yes	3.0
(Schwieren and Vossen, 2009)	No	Yes	No	Partially	No	No	0.5
(Ranabahu et al., 2011)	No	No					
(Barnawi et al., 2012)	No	Yes	Yes	Yes	Yes	Yes	4.0
…							

2.3.4. Data extraction and monitoring

The data extraction forms used in this research are created by combining the examples and following the instruction given by Kitchenham and Charters (2007) and Jørgensen (2007). As

discussed in chapter 2.1.2.2, the aim of data extraction process is to accurately and without bias record the appropriate information from the selected papers. Based on the data collection form template presented in Table 4, the final developed data collection form is adapted for this particular research. Full list of all filled data extraction forms can be found in Appendix D on page 265. The example of filled data collection form with extracted data from (Xiong and Wang, 2010) is presented in Table 17.

Table 17 - Data collection form

Data item	Value	Notes
Study identifier	(Xiong and Wang, 2010)	
Title	A new combined method for UCD and software development and case study	
Publication details	Y. Xiong and A. Wang, "A new combined method for UCD and software development and case study," in Information Science and Engineering (ICISE), 2010 2nd International Conference on, 2010, pp. 1–4.	
Study type	New methodology	
Name of methodology / approach	Inter-combined Model	
Application in multi-platform development	Yes	Platform independent
Details on defined / reported methodology / approach	Inter-combined Model aims to shorten the knowledge transfer from designers to developers. The model has four parts: - Requirement analysis and user study - Model establishment and function map specification - Design and background engine implementation - System integration and coding	
Additional resources on methodology / approach description	Each phase was described in additional details, but not to the level of activities, tasks, inputs and outputs.	
Report on methodology / approach example implementation	Mobile Karaoke project.	
Organizational aspects on implementation	Researchers stated that Inter-combined Model has positive effect on human resource arrangement and cost reduction.	
Project management aspects on implementation	Some implications on human resource arrangements.	

The presented data extraction form consists of three parts. The first part aims to extract general data on each study, the second part directly responds to research questions, and the third part gives more details on the study quality but only related to data analysis and not inclusion and exclusion criteria.

2.3.5. Data synthesis

As the research questions in this systematic literature review are straightforward and easy to answer from of extracted data, the activities of the data synthesis are performed according to instructions given by Petticrew and Roberts (2005).

The data are synthesized into the following groups

- Studies reporting the creation of new methodology or approach
- Studies reporting the methodology or approach usage
- Methodologies/approaches not eligible for multiplatform development
- Methodologies/approaches targeting specific mobile applications

Lists of potential methodologies and approaches that could be used in the subsequent phases of this research process are given in Table 18 and Table 19. The total of 14 methodologies and 2 approaches are identified as new while 9 methodologies and 4 approaches are identified as being used in development of mobile applications. Methodologies are marked as type M and approaches as type A in the following tables.

Table 18 - Developed methodologies and approaches

Name	Type	Study	QA score
Agile Methodology for Mobile Software Development	M	(Rahimian and Ramsin, 2008)	3.0
Agile Solo	M	(Nyström, 2011)	2.0
Agile usability process	M	(Wolkerstorfer et al., 2008)	2.0
DEAL	M	(Alyani and Shirzad, 2011)	2.5
Integrated Product Development Process for Mobile Software	M	(Zeidler et al., 2008)	2.0
Inter-combined Model	M	(Xiong and Wang, 2010)	3.0
MASAM methodology	M	(Jeong et al., 2008)	2.5
Methodology for Building Enterprise-Wide Mobile Applications	M	(Chen, 2004)	4.0
MicroApp visual approach	M	(Cuccurullo et al., 2011)	2.5
Mobile Application Development Methodology	M	(Rupnik, 2009)	1.5
Mobile-D	M	(Abrahamsson et al., 2004)	2.5
		(Abrahamsson et al., 2009)	1.0
New media application prototyping	M	(Biswas et al., 2006)	3.0
Systems Development Methodology	M	(Binsaleh and Hassan, 2011)	4.0
ViP (Virtual Platform)	M	(Um et al., 2005)	4.0
Composite Application Software Development Process Framework	A	(Maharmeh and Unhelkar, 2009)	3.0
MobiLine	A	(Marinho et al., 2012)	4.0

Type: M - Methodology, A - Approach

There are several facts that should be pointed out and are related to the identified new methodologies and approaches. First of all, only one methodology was covered by more than one study, while all other methodologies are presented in a single identified study. Secondly, as expected, the methodologies and approaches in the mobile development field are rather new. Only 4 studies are more than 5 years old, while all the other studies date in the last five years. The overall study quality assessment score (calculated as explained in chapter 2.3.3), has the mean value of 2.735 (68.38%) with the standard deviation of 0.903. This can be interpreted as relatively low study quality with high deviation in quality. But, as the quality assessment was performed on the studies and not on the reported methodologies, without additional research it is not possible to order the methodologies according to their quality.

On the other hand, as expected, more authors reported the usage of methodology or approach. Total of 9 methodologies and 4 approaches have been reported as used. The important fact is that only one methodology (Mobile-D) identified as new was reported to have been used. The usage of this methodology was reported in five different studies, while all other new methodologies and approaches were not reported to have been used.

Table 19 - Used methodologies and approaches

Name	Type	Study	QA score
Design Science	M	(Ejlersen et al., 2008)	3.0
Dynamic Channel Model	M	(Shiratuddin and Sarif, 2008)	2.5
		(Shiratuddin and Sarif, 2009)	2.0
Extreme Programming	M	(Korkala and Abrahamsson, 2004)	3.0
		(Kaariainen et al., 2004)	2.0
		(Salo, 2004)	3.0
Kanban	A	(Bergström and Engvall, 2011)	1.5
Mobile-D	M	(Shiratuddin and Sarif, 2008)	2.5
		(Shiratuddin and Sarif, 2009)	2.0
		(Korkala and Abrahamsson, 2004)	3.0
		(Hedberg and Iisakka, 2006)	4.0
		(Ihme and Abrahamsson, 2005)	3.5
Mobile Engineering (MobE)	M	(Shiratuddin and Sarif, 2008)	2.5
		(Shiratuddin and Sarif, 2009)	2.0
Mobile RAD	M	(Shiratuddin and Sarif, 2008)	2.5
		(Shiratuddin and Sarif, 2009)	2.0
Rapid Application Development	M	(Forstner et al., 2005)	2.0
Scrum	M	(Su and Scharff, 2010)	2.0
		(Pauca and Guy, 2012)	1.0
		(Scharff and Verma, 2010)	2.5
		(Scharff, 2010)	2.5
		(Alyani and Shirzad, 2011)	2.5
		(Scharff, 2011)	2.0
Model Driven Development	A	(Charaf, 2011)	3.0
		(Kim, 2008)	2.5
		(Ortiz and Prado, 2010)	3.0
		(Forstner et al., 2006)	2.5
		(Thompson et al., 2010)	1.0
		(Khambati et al., 2008)	2.5
Model Driven Product Lines	A	(Zakal et al., 2011)	2.0
Software Product Lines	A	(Rosa and Lucena,Jr., 2011)	2.0
Test Driven Development	A	(Nyström, 2011)	2.0
		(Abrahamsson et al., 2005b)	4.0
		(Kim et al., 2009)	1.5
		(Hedberg and Iisakka, 2006)	4.0

Type: M - Methodology, A - Approach

It was hard to predict the number of methodologies that would target specific mobile platforms, and it turned out that only one methodology (see Table 20) cannot be used in multi-platform mobile application development as it targets only those platforms which support Flash technology. Actually, the paper presents a development process for interactive mobile applications based on Sony Ericssons's Capuchin project which aimed to bring together the

advantages of Java Micro Edition (JME) and Flash Lite. The methodology in particular deals with specific issues raised by this approach and this marks the stated methodology as not eligible to be used in this research process.

Table 20 - Methodologies not eligible for multiplatform development

Name	Type	Study	QA score
Development process of Caputchin applications Targeting platforms supporting Flash only	M	(Maia et al., 2010)	1.0

Type: M - Methodology, A - Approach

The stated groups are defined in accordance with the research process that has been performed in this thesis and that is the reason why some methodologies and approaches had to be separately reported as targeting only specific or specialized mobile applications (Table 21). These methodologies were also not applicable to be used in this research process, but are worth mentioning as being developed for mobile applications.

Table 21 – Methodologies/approaches targeting specific mobile applications

Name	Type	Study	QA score
Component Based Model for IP Multimedia Subsystem Targeting IP multimedia subsystems only	M	(Barnawi et al., 2012)	4.0
Design and Development Methodology for mobile RFID applications Targeting only RFID applications	M	(Schwieren and Vossen, 2009)	0.5
MMCD Methodology Targeting only m-Learning applications	M	(Saifudin et al., 2011)	1.5
PEPERS Development Methodology (PDM) Targeting only P2P applications	M	(Walkerdine et al., 2009)	3.0
2TUP - 2 Tracks Unified Process Targeting only mobile games development	M	(Gal and Topol, 2005)	3.0
MobiCloud Targeting generation of a cloud mobile hybrid applications	A	(Manjunatha et al., 2010)	2.5

Type: M - Methodology, A - Approach

2.4. Choosing development methodology

As stated before, the total of 22 development methodologies and 7 development approaches were identified as eligible to be used in the development process.

As the starting-point assumption of this research is to provide the teams with a possibility of using native development environments and preferred development methodology, the research should not be dependent on any special characteristics that a chosen methodology consists of. In the other words, any identified methodology could be used.

However, the established criterion used to choose development methodology was *reported applicability.* Cross-analysis of the results presented in Table 18 and Table 19 shows that

Mobile-D was the only methodology specifically created for mobile applications development that was reported to be used in practice. In addition, we performed a small research to identify other sources published by the methodology creators and found that this methodology is thoroughly and in detail defined. The documents that are officially available and that describe the Mobile-D development methodology are presented in the following table (Table 22).

Table 22 - Documents describing Mobile-D methodology

Year	Document
(2005a)	P. Abrahamsson, A. Hanhineva, H. Hulkko, J. Jäälinoja, K. Komulainen, M. Korkala, J. Koskela, P. Kyllönen, and O. Salo, "Agile Development of Embedded Systems: Mobile-D," ITEA, Agile Deliverable D.2.3, 2005.
(2006)	T. Kynkäänniemi and K. Komulainen, "Agile Documentation in Mobile-D Projects," 2006.
(2004)	O. Salo and J. Koskela, "Mobile-D Glossary, VTT Technical Research Centre of Finland, Available at: http://agile.vtt.fi/mobile-d.zip." VTT Technical Research Centre of Finland, 2004.
(2006a)	VTT Technical Research Centre of Finland, "Mobile-D Online Presentation (Web Application)," *AGILE Software Technologies Research Programme*, 2008. [Online]. Available: http://agile.vtt.fi/mobiled.html. [Accessed: 16-May-2012].
(2004)	P. Abrahamsson, A. Hanhineva, H. Hulkko, T. Ihme, J. Jäälinoja, M. Korkala, J. Koskela, P. Kyllönen, and O. Salo, "Mobile-D: an agile approach for mobile application development," in Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications, New York, NY, USA, 2004, pp. 174–175.

The obtained papers and other documents that include detailed guidelines are sufficient to make this methodology fully applicable and usable throughout this research. Additionally, as the Mobile-D is leaning on, and is strongly connected with, Test Driven Development approach, this approach will be used in the following phases as well.

To conclude, systematic literature review resulted in the lists of different methodologies reported to be used, or created specifically for mobile applications development. But, the analysis on reported applicability showed that Mobile-D with Test Driven Development is the only newly created methodology already used in practice and that is the reason for choosing this methodology and approach in the research phases that follow.

2.5. Relevance of the chapter

To recapitulate, first we explored the state of the art in performing a systematic literature review in the field of software engineering. The three-phase guidelines given by Kitchenham and Charters (2007) are followed and discussed by adding the recommendations and findings from other influential authors in the field. The results of the discussion are a contribution to the knowledge and could be used either by researchers or by PhD students in order to employ suitable methods and techniques and to lower the biases and increase the quality of review.

Following these recommendations, second part of the chapter presented the conduction of SLR which in the end brings the identification of 22 development methodologies and 7 development approaches that could be used for multi-platform mobile applications development. Among identified methodologies, our analysis showed that Mobile-D is the most suitable methodology and it will be used along with Test Driven Development in the rest of this research process.

Having the methodology and approach chosen, we have finished the first phase of our research process. Now we move to the second phase with the goal of identifying the artifacts arising in the methodologically driven development processes for two target platforms.

3. METHODOLOGY IMPLEMENTATION

After performing systematic literature review, identifying and choosing the development methodology to be used in this research, in this chapter we will report in detail the development process and Mobile-D methodology implementation. As the report of such process is not a trivial task, first we will introduce the basics of Mobile-D methodology and accompanying approach called Test Driven Development in order to give an overview of the performed phases. Additionally, we will define the term 'artifact' to clearly denote the point of view to be taken while reading this chapter.

The mobile application that is developed is named *KnowLedge*. It is a simple social network application designed to share knowledge among participants grouped in groups of interest. The application is designed to cover the main functional development requirements and thus to represent the vast majority of mobile applications. Such requirements in general cover distinct development concerns, including UI features, local database, device API-s, connection to web services and 3rd party features.

The report of the development process presented in this chapter focuses on the created artifacts and their connection to each other along with their connections to the performed activities. In the Android case we bring a detailed description of the whole process along with the examples of the artifacts created. Even so, in the Windows Phone case, we decided not to report the whole process in detail again, but rather to discuss the possibility of reusing the existing artifacts. We found that many artifacts can be completely or partially reused.

3.1. Mobile-D overview

3.1.1. Introducing Mobile-D

The methodology was first presented by Abrahamsson at al. (2004) and after that it slightly evolved to the final version which is in detail presented in technical specification which includes the complete glossary, the description of all phases, stages, tasks and practices along with templates (Abrahamsson et al., 2005a). Additionally, the VTT Technical Research Centre of Finland created and published a web application which can be used to easily

navigate through methodology phases and to obtain the relevant specification documents (VTT Technical Research Centre of Finland, 2006a).

3.1.2. Mobile-D process

The short overview of this methodology is already given in the chapter 2.2.2 while describing methodologies for development of mobile application. A more detailed overview of the process will be given here in order to create a basis for the implementation that follows.

Mobile-D process (see Figure 13) includes five phases that are executed in partially incremental order. The aim of the first phase, called *Explore*, is to prepare the foundation for future development. The *Initialize* phase should describe and prepare all components of the application as well as to predict possible critical issues of the project. Initialize phase is usually called a zero iteration (*0-iteration*) phase as it in addition to *project set-up* includes the stages of *planning day*, *working day* and *release day* which are also used in Productionize phase. The idea of the 0-iteration phase is to assure the functionality of the technical development environment through the implementation of some representative features. Additionally, in this phase some prototyping can be done in order to decide which technological solution would be the most appropriate for the rest of the development process.

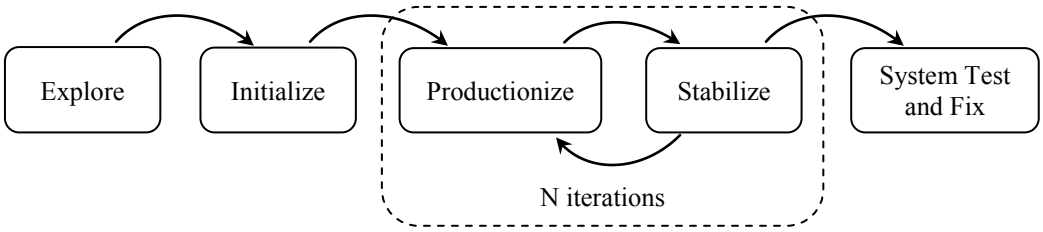


Figure 13 - Mobile-D process

The *Productionize* and *Stabilize* phases are executed iteratively in order to develop all other features of the mobile product. Iterations start with planning day in *Productionize* phase. The first activity is post-iteration workshop which aims to enhance the development process to better fit the needs of the current software development team. The requirements analysis, iteration planning and acceptance test generation tasks follow and are executed during the planning day. The working day is based on implementation through test driven development, pair programming, continuous integration and refactoring. This day ends with the task of informing the customer on new functionality. Finally, the release day includes the activities of integration and testing. The *Stabilize* phase has the goal to finalize the implementation, including integrating subsystems if needed. As this phase can contain additional programming and development, the activities are very similar to the activities in the Productionize phase.

Only additional activity concerns documentation wrap-up. Iterations should result in a working piece of functionality at the user level.

Finally, *System Test and Fix* phase aims to detect if the produced system correctly implements the customer defined functionality. It also provides the project team feedback on the systems functionality and the defect information for last fixing iteration of the Mobile-D process. This last iteration is not obligatory, but when fixing is needed it consists of the same activities as other implementation iterations already explained.

While observing the whole Mobile-D process we can conclude that it is an agile approach to mobile application development which is based on combination of eXtreme programming in terms of practices, Crystal family of methodologies in terms of scalability and Rational Unified Process in the terms of life-cycle coverage. In paper (Supan et al., 2013) we have discussed the challenges and issues that accompany the use of this methodology that companies or small teams should be aware of before introducing it in everyday practice.

3.1.3. Mobile-D artifacts

An artifact may be defined as "*an object that has been intentionally made or produced for a certain purpose*" (Hilpinen, 2011) or it may refer to "*one of many kinds of tangible byproduct produced during the development of software*" (Parker, 2011). The artifacts that arise in the process of mobile application development are from special interest in this research and thus we have adopted the definition of an artifact as "*any piece of software (i.e. models/descriptions) developed and used during software development and maintenance.*" (Conradi, 2004)

Conceptual model (Figure 14) comprises the Mobile-D process, its activities and tasks that are performed by utilizing some methods and practices and using some tools resulting in artifacts as final outputs. Thus, artifacts are results of performed activities, but they are also used as inputs to perform other activities and tasks.

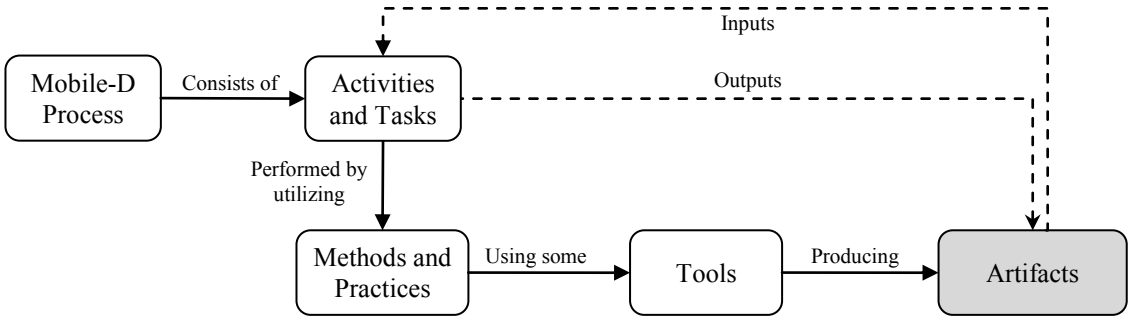


Figure 14 - Artifacts in Mobile-D

To give an overall picture, Table 23 shows all inputs and outputs that are defined by the methodology and are connected with the five mentioned phases.

Table 23 - Mobile-D inputs and outputs

Explore		Initialize		Productionize		Stabilize		System test & fix	
Inputs	**Outputs**	**Inputs**	**Outputs**	**Inputs**	**Outputs**	**Inputs**	**Outputs**	**Inputs**	**Outputs**
Product proposal	The initial requirements document	Initial requirements document	Updated project plan and architecture line plan	Updated project plan and architecture line plan	Implemented functionality	The implemented functionality of the product	The implemented functionality of the whole project software	The implemented functionality	A tested and fixed system (the final release)
Mobile-D process library	Project plan	Project plan	The first version of Software Architecture and Design Description document.	The first version of Software Architecture and Design Description document	Acceptance test documentation	Related development artifacts	The finalized documentation of the product	Acceptance test documentation	Documentation of found defects
Organizational process library	Base process description	Base process description	Implemented functionality	Plans for checking of critical development issues	Developer notes			User-defined functionality	System Test report
Contract	Measurement plan	Measurement plan	First version of the product backlog	Implemented functionality	UI – illustrations			User interface description is used to create test cases	Test log
Initial requirements document	Training plan	Training plan	The updated Initial requirements document	Product backlog containing all the identified requirements of the project	Action point list				
Project plan	Architecture line description	Architecture line description	Developer notes and UI – illustrations of each discussed requirement	Metrics data	Updated product backlog				
Relevant standards		Product backlog	Acceptance tests for each requirement	Experience of the project team	Updated project plan				
				Acceptance test documentation	Story and task cards				
				Story and task cards	Experience data of the project team tests				
				Data on spent resources	Knowledge of the system requirements and acceptance				
				Manuals, API specifications and other support material	Defect list				
				Unit tests	Release audit checklist				
					Initial requirements document				
					Daily status report				

Source: (Supan et al., 2013)

The artifacts that we are interested in this research do not concern only the direct results of performing the activities, but also the specific outputs that are connected with the development for a specific target platform.

3.1.4. Test driven development

Mobile-D strongly suggests the usage of Test Driven Development which is connected to all Mobile-D phases. The basics and the state of the art on TDD can be found in (Hammond and Umphress, 2012). To make the understanding of the following chapters easier, we bring a quick overview of this development approach.

The practice of test driven development requests the developer to write a failing automated test case and then to write the production code that will pass the test. In general TDD process can be summed up into five main steps (Beck, 2002):

1. Write a new test case.
2. Run all the test cases and see the new one fail.
3. Write just enough code to make the test pass.
4. Re-run the test cases and see them all pass.
5. Refactor code to remove duplication.

In Mobile-D, the purpose of TDD is to give the developers confidence that the code they produce works as well as to guide the design of the code to an easily testable structure. Additionally, the refactoring practice is also based on TDD to ensure that changes made to the code did not break any functionality (Abrahamsson et al., 2005a). Finally, being the main practice of any working day, test driven development is used in all phases except the first (Explore) phase.

3.1.5. Mobile-D reference

The most important source of information on how to perform Mobile-D methodology for this research is the already mentioned technical report presented in (Abrahamsson et al., 2005a). As the document contains detailed information on Mobile-D phases, stages, activities, tasks, practices, patterns and other relevant concepts, we recommend having a glance at it before reading the following sections and having it at disposal while reading. All other documents mentioned in Table 22 are also a relevant source of information and can be used to gain more comprehensive knowledge on Mobile-D.

The following sections report on the conduction of Mobile-D methodology in creation of a prototype application for two target platforms.

3.2. Explore phase

3.2.1. Targeted users and stakeholders

The application KnowLedge does not have any specific target groups. It is aimed to be distributed freely through online stores to all interested parties. Additionally, as the nature of the application was to serve as an experiment in a research process, there are no classical stakeholders recognized. The only participating individuals in the process were two thesis supervisors included as process specialists, and the researcher himself who conducted all of the activities.

3.2.2. Initial requirements

The application is intended to enable users to learn and/or share knowledge in an interactive and social manner. The basic usage should include the following functional requirements:

- Browsing through the categories to find existing knowledge on a topic
- Placing the request for new explanation/instructions/tutorial
- Creation of new knowledge (either answering unsolved requests or creating a new topic)
- Sharing the knowledge in groups
- Sharing location data among group members
- Android and Windows Phone native look and feel
- Different user privacy levels

The presented list does not include nonfunctional requirements as nonfunctional requirements analysis was not performed for this prototype application.

3.2.3. Architecture line description

<u>The goal for internal product quality:</u> an evolutionary prototype.

<u>System context</u>: the application is intended to be a standalone mobile application dependent on internet connection and on supporting web services. The optional dependency (not being a part of the core features) is fine or coarse GPS location. Only one interface to the external entities should be developed in order to join the mobile application with web services. There is no need for any other interfaces as the system does not include other enterprise, infrastructure or legacy subsystems.

Technological domain includes the nonfunctional requirements of application being runnable on any Android 2.2 or newer device. According to currently available data (Android

Developers, 2013), more than 95% of all Android devices are covered by this inclusion criterion.

Architectural risks: variety of android devices and supporting API-s. Different device capabilities and significant differences in device screen size could become problem in testing and implementation of user interface.

Using a somewhat old version of API-a (version 8) could result in constraints in application of suitable user interface and other features.

Architectural skills: sufficient, as the main researcher has been involved in mobile applications development for Android during the last several years.

Architectural training needs: not necessary.

Software architecture: multilayered software architecture with separated business logic, user interface and database connectivity layers. The idea was to capture the core architectural abstractions for the whole system as soon as possible, on the basis of the experience of the project team, and to do a constant architectural refactoring by using pattern-based core abstractions.

Software architecture documentation: described software architecture documentation process supported by developer-level models, sketches and short documents used in the development process.

Templates for SW architecture and Design Description document: Several specific templates aligned with UML modeling language were created. The architecture and design were described at least with UML Class diagrams and ERA models. As the chosen methodology specifies, some other typical agile tools were also used in order to describe the features and planned tasks. These tools include UI sketches, product backlog, story and task cards et cetera. Typical software architecture that was used is multi-layered software architecture.

3.2.4. Project plan

Due to the project's specific requirements and its background, it did not include any financial or resources constraints. The basic project plan was defined as a set of phases and stages and the overall project duration was set to 20 weeks. The team responsible for the conduction of the project was composed of a researcher and supervisors, although the supervisors' roles were very limited and included few activities during the project establishment, mainly quality checking and final validation.

The initial project plan is given in the following picture including the identified iterations and graphical representation on Gantt chart.

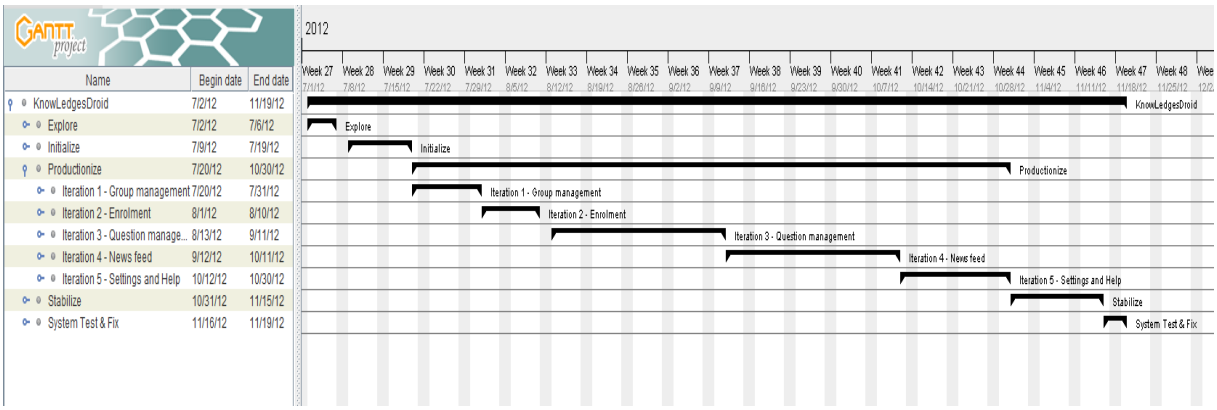


Figure 15 - Basic project plan

In this phase it was impossible to determine the iterations that will be necessary in the Fix phase as those are dependent on the overall quality of the development process, and on some unpredictable technological issues.

3.2.5. Documentation

The documentation includes two distinct sets of documents. First set considers the documents related to the project implementation and project management. Aligned with the agile practices, this set contains the documents that are considered to be the necessary minimum in every project development process. This group contains:

- Initial requirements document
- Project plan document
- Software architecture and design description document
- System test plan
- Product backlog
- System test report

The second group of documents includes documents related to the research that is conducted. This set includes the following documents:

- Identified artifacts and description
- Historical data on every document
- Notes on the development process

The iterative updating approach of producing the documents with preservation of versions was used. This approach is aligned with the agile practices and is suitable for a project of this type.

3.2.6. Monitoring and measurement

As our project did not deal with resource (human, time, money) management, the monitoring activities were not in our focus. Thus, the monitoring of the development process was conducted only by identifying the level of agreement between planed and conducted activities. Additionally, the duration of the activities was measured and noted for future comparison with subsequent development processes. The overall goal for this process was not to exceed the planned duration of the project, but this was not a crucial requirement and it did not affect the research goals.

Additionally, the quality assurance was conducted by acceptance tests, validation, usage of coding standards, process validation by supervisors and finally product verification on the market.

3.2.7. Project plan checklist

Taken from the Mobile-D process library (VTT Technical Research Centre of Finland, 2006b), the following table represents the project plan checklist for the *Explore* phase.

Table 24 - Project plan checklist - Explore

Project Plan Checklist			
Explore			
Initial requirements	**Yes**	**No**	**NA**
All the initial functional requirements have been included in the project plan	x		
All the initial non-functional requirements have been included in the project plan	x		
Schedule & Rhythm			
The overall schedule has been included in the project plan	x		
The planned rhythm (phases and its iterations) have been defined in the project plan	x		
Resources			
Project plan has been updated with the identified interest groups and their members			x
Project plan has been updated of the information concerning the selected software development tools, terminals, etc.	x		
Project plan has been updated with the identified project team members	x		
Training			
Training needs of project team have been included in the project plan			x
Schedule of training has been included in the project plan			x
Documentation			
The documents to be produced in the project have been included in the project plan	x		
The life span of each document has been included in the project plan	x		
Quality Assurance			

The quality assurance procedures have been defined in the project plan for each work product (documentation, code and product) including the actors and schedule	x		

The checklists showed that during the Explore phase, three aspects of Mobile-D methodology were not applicable (NA) in the context of this mobile project (as explained in previous chapters). All other elements are marked positively which makes this phase successfully completed.

3.3. Initialize phase

3.3.1. Environment setup

The software development environment was prepared for development of Android applications. Although the installation of base tools on the machine (including browser, PDF viewer, picture viewer etc.) and the installation of specific tools for project management (GantProject) and reporting tools (Microsoft Office) was performed during the project preparation and explore phase, the implementation tools (Case Studio, Sprintometer, Visual Paradigm for UML, SQLite Professional…) and development (Java Development Kit, Eclipse IDE, Android Development Tools, Android SDK…) had to be installed in this phase. Additionally, the drivers for testing devices were also downloaded and installed and the devices were connected to the development environments. The development environment was tested and simple Android application was produced and deployed on a mobile device. Finally, the subscription to servers for hosting database and services was obtained and tested.

All mentioned tools were free or obtained through relevant institutional subscription of the University of Zagreb and/or the University of Alcala.

There was no need for environment setup for the purpose of training or customer communication.

3.3.2. Project plan and architecture plan

The basics for overall project execution plan remained the same at the end of this phase, but taking into consideration a more detailed requirements analysis it was possible to define a more fine grained iterations including the planning, working and release days. The updated project plan can be seen in Figure 16. As there was no need for personal resources or financial planning, these tasks were skipped. Additionally, extensive risk planning which usually takes place in organizational environment was not necessary.

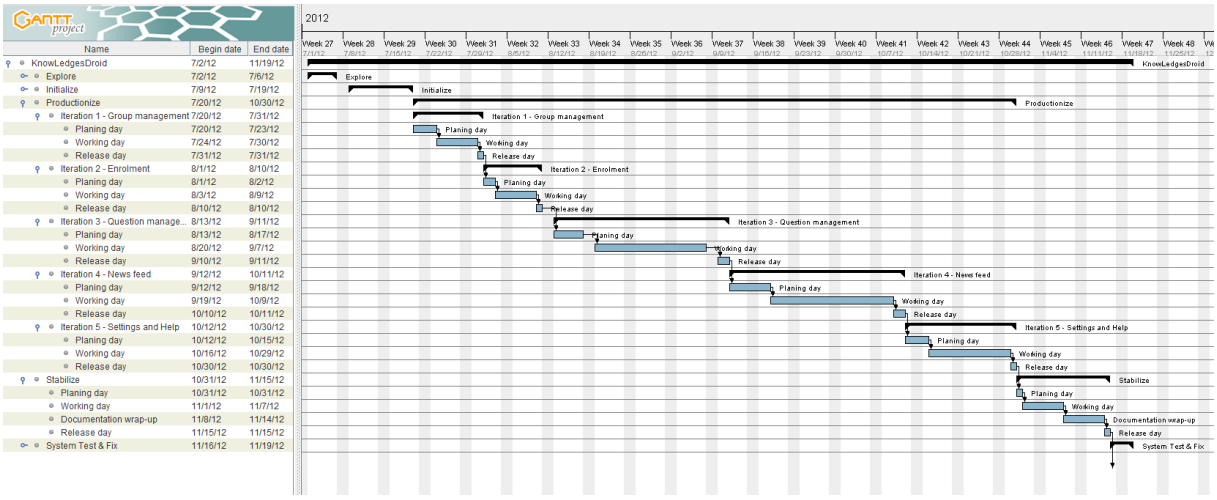


Figure 16 - Detailed project plan

The planned system architecture is defined on two abstraction levels. First (upper) abstraction level, as shown in Figure 17, presents the overall system architecture which includes the main system participants and components. The identified components are mobile application, and web and database servers, while the infrastructure is based on connectivity (Internet) and GPS data. Although, the main system functionality is not visible from this diagram, the important requirement of enabling the users to form the groups is presented here.

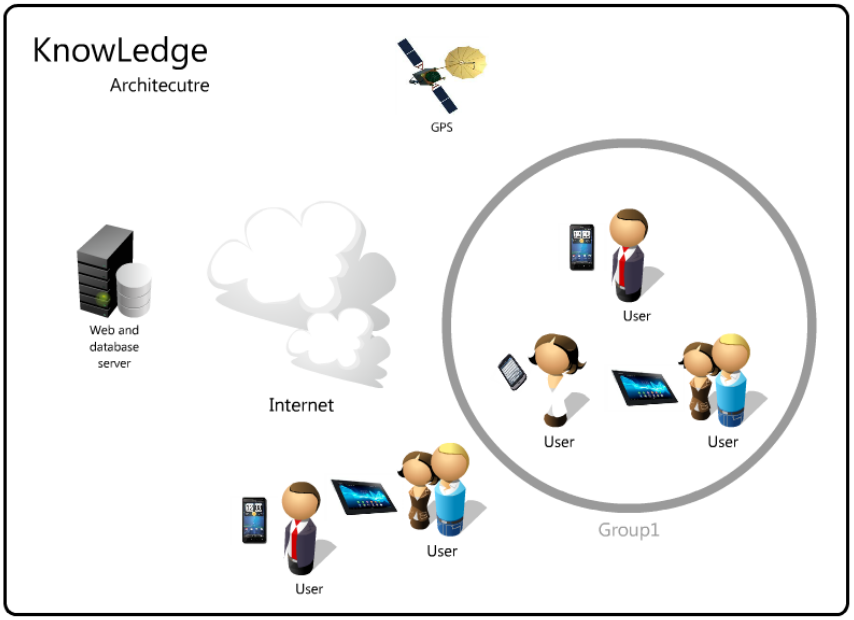


Figure 17 - Overall system architecture

The second architectural diagram shows the mobile application detailed architecture as it is presented in Figure 18. The idea was that the mobile application should, accordingly, communicate with web service and lean on native (i.e. Android) and 3rd party API-s in order to deliver the required functionality. It should be based on multi-layered architecture with

three distinct and connected layers. The internal cohesion (see (Miller, 2008)) of the presented modules should be high, and at the same time the external coupling should be kept low.

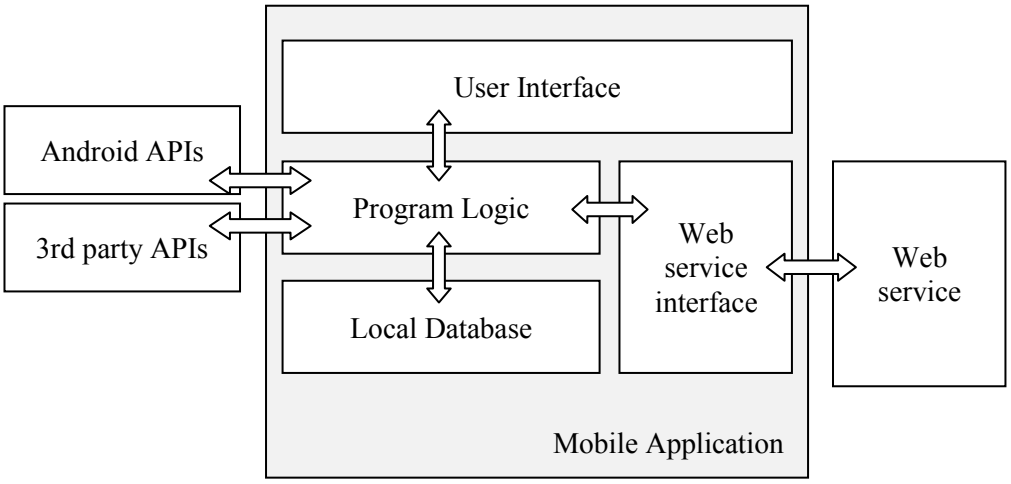


Figure 18 - Mobile application architecture

3.3.3. Initial requirements analysis

The initial requirements analysis task was performed, and the results include product backlog, the user interface sketches and the generated acceptance tests for each requirement presented in next chapter.

3.3.4. Product backlog

Product backlog describes application features presented through user stories. Every feature has an assigned importance level. They are scaled from 1 being not important to 5 being very important.

Table 25 - Product backlog

Features / stories		Importance
F1.1	When the application is started the news should be displayed. News should include any unread answers to the user's questions; news on activities in user's groups and other information important for current user.	3
F1.2	The news presented on the first application screen should be "links" to corresponding application functionality.	3
F2.1	Current user should be able to check all his questions, including those that have been answered already. Questions should be presented by title and short description. Other details about every question should be presented in new window after user clicks on it.	5
F2.2	User should be capable to add a new question. New questions should be defined in separate windows which should include all important information about the question (title, text and images). The images should be taken by the phone camera.	5
F2.3	User should be capable to delete his/her own question. The deletion should not be performed without user's explicit confirmation on deletion action.	3

F2.4	User should be capable to change his/her own question. The process of question changing should be similar to process of question adding.	1
F2.5	User should be capable to add answers to his/her own and others' questions.	5
F2.6	The owner of the question should be able to mark a question as answered.	5
F2.7	User should be able to apply the filter by root-searching the list of questions available to him.	2
F3.1	User should be able to set/change own profile. The profile should include the basic information about the user (visible) to other group members.	5
F3.2	User should be able to set/change application settings. The settings should include the possibility to deny further invitations to groups, to set privacy level (of showing or no emails to other users and of showing or no current location to other users).	2
F4.1	User should be able to see the list of all groups currently enrolled to.	5
F4.2	User should be able to apply the filter by root-searching the available groups according to their title and description. All groups should be observed by search.	2
F4.3	User should be able to see the details on any group he is enrolled to, including the list of other members. User should NOT be able to see the list of other members (except their number) for the groups he is not enrolled to.	4
F4.4	User should be able to join any existing group by sending the application to group owner.	5
F4.5	User should be able to leave any group he is enrolled to. Other group members should only be notified on that. Owner cannot leave the group and the group should be deleted manually (see F5.4).	1
F5.1	User should be able to create a new group.	5
F5.2	User should be able to invite new members to his group by inviting them via in application email.	2
F5.3	User should be able to invite new members to his group by sending them email.	1
F5.4	User should be able to delete any group he owns.	2
F6.1	User should be able to see all members of the groups he is enrolled to on the map. If group member has disabled this privacy setting, it will be excluded from the view.	3
F7.1	User should be able to read a general help about the application usage.	1

3.3.5. Acceptance tests

The template sheets for acceptance tests proposed by Mobile-D (Abrahamsson et al., 2005a) were used and the tests are defined for each application requirement defined in the product backlog. Each acceptance test was to be approved at the end of development process, and it includes the definition and remarks on the test of final functionality in different contexts. The following test descriptions are examples of acceptance tests created in this step.

Acceptance test F1.1

Displaying news for current user

When the application is started the news should be displayed. News should include any unread answers to the user's questions; news on activities in user's groups and other information important for current user.

Context 1
Application executed for the first time or user did not created his profile yet.

Expected output

Title	Description
Welcome	Welcome to KnowLedge application. To begin click to set up your user profile.

Context 2
User is not member of any group and there are no activities to display.

Expected output

Title	Description
No news	There are no news to display. Use application menu to join groups and become part of KnowLedge community.

Context 3
User actively uses the application and has news in several categories.

Expected output

Title	Description
New answer	Your question **%questionTitle** has been answered by **%firstName**.
%questionTitle	%description. [up to 50 chars]
New invitation	You have invitation by **%firstName** to join the group **%groupName**.
Application accepted	Your application to join the group **%groupName** is accepted.
New member	**%firstName** joint the group **%groupName**.

Acceptance test F1.2

Linking news
The news presented on the first application screen should be “links” to corresponding application functionality.

Context
News presented on the first screen.

Expected output

News	Link
Welcome	Users profile page.
No news	-
New answer	Question %questionTitle page.
%questionTitle	Question %questionTitle page.
New invitation	Invitation dialog followed by group page.
Application accepted	Group %groupName page.
New member	New member profile page.

Acceptance test F2.1

My questions

Current user should be able to check all his questions, including those that have been answered already. Questions should be presented by title and short description. Other details about every question should be presented in new window after user clicks on it.

Context 1

User clicks on "My questions" option.

Expected output

Question title	Question description [up to 50 chars]
Title 1	Description 1.
Title 2	This description cannot fit into 50 chars and wi…
Example question	What is the name of this bird?

Context 2

User clicks on any question presented in the list.

Expected output

Question title	Question description [full]	Asked by;	Group	Answers
Title 1	Description contained from text and images. In single description, text and image could be presented multiple times.	%firstName %lastName	% groupName	List of answers.
Title 2	This description cannot fit into 50 chars and will be shortened in list view but in question view should be written fully.	%firstName %lastName	% groupName	List of answers.
Example question	What is the name of this bird? I sow it yesterday in our park. It looks like some kind of a parrot.	John Johnson	Nature	-

Acceptance test F7.1

Help

User should be able to read a general help about the application usage.

Context

User clicks on "Help" option.

Expected output:

The new view with textual help appears. The help contains information on all application features.

3.3.6. User interface sketches

In order to align the user requirements with the technological implementation and possibilities provided by a target platform, user interface sketches were created. These sketches also enabled the team to get a full picture of the desired functionality. After several iterations, the sketches were finished. Figure 19 shows an example of the created document.

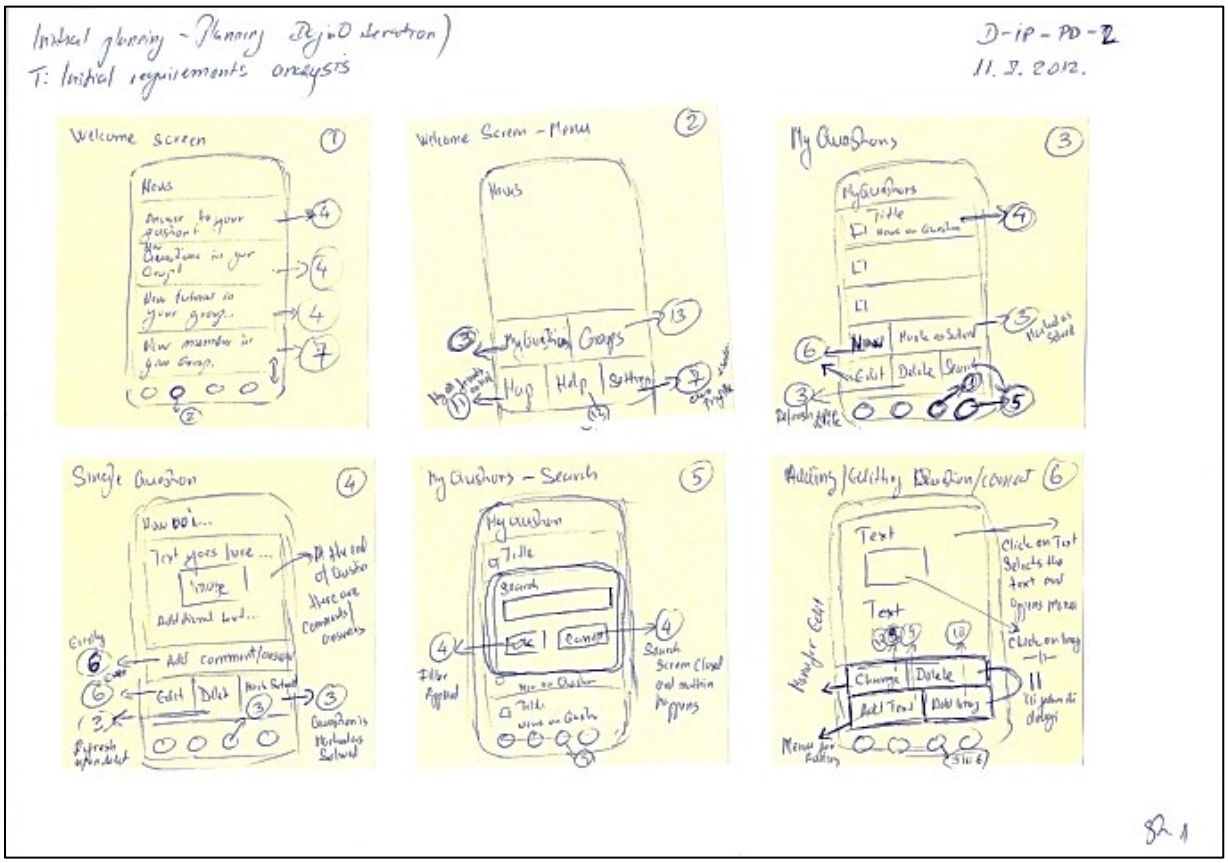


Figure 19 - User interface sketches

3.3.7. Trial Day

The selected feature that was to be implemented in this trial day is F3.1. The idea of performing trial day was to create functionality that will cover (at least in basic aspects) most of the architectural design elements and also to create the base for other features. As the application is user oriented, having information on the current user was a prerequisite for almost all other features which made this feature a core functionality of the system.

Table 26 - Selected feature for Trial Day

Features / stories		Importance
F3.1	User should be able to set/change own profile. The profile should include the basic information about the user (visible) to other group members.	5

Finally, the goal of this day was also to assure the functionality of the technical development environment through the implementation of the feature. The following tables present defined story cards (SC) and task cards (TC). These documents were defined during the planning day, but were refined during the implementation and documentation wrap-up.

3.3.7.1. Story and task cards

Story card F3.1

F3.1	Type	Difficulty		Effort		Priority	Notes
		Before	After	Estim.	Spent		
	New	H	H	4	5	5	
Description							
User should be able to set/change own profile. The profile should include the basic information about the user (visible) to other group members. The basic information about the user should include first name, last name and mail address. The information should be stored in local database and synchronized with information on web service.							
Date	**Status**	**Comment**					
11.7.2012	Defined	This story is taken to be implemented during the trial day. This will introduce the execution of tasks concerning preparation and validation activities and thus will be slightly different than in implementation of other stories.					
12.7.2012	Implementing	The implementation is taking longer than expected. There are many decisions that are to be made but after some initial research is performed. This research include prototyping and writing the code that is to be discarded, searching and reading the available sources, looking through finished projects etc.					
16.7.2012	Done	The basic architecture of this project is created. The database, business logic, user interface, web service and helping layers are established. The automatic tests including unit and integration testing are created.					
16.7.2012	Verified	All test, including unit, integration and acceptance testing are performed and successful.					

* This story card, as all other SCs, was defined during the planning day but was refined during the implementation.

Task card TC-0-1 - Create initial test cases

TC-0-1	Type	Difficulty		Confi-dence	Notes
		Before	After		
	New	5	5	3	
Description					
Initial test cases for this functionality should be created.					
Date	**Status**	**Comment**			
11.7.2012	Defined				
12.7.2012	Implementing	After choosing from several existing testing frameworks, the core functionality will be tested by native *android.test* framework, and the robotic testing of application usage will be performed by *robotium* free framework (code.google.com/p/robotium/).			
12.7.2012	Done	Some core tests are created. Other tests and robotic integration testing will be defined at the end of the stage. The problems experienced include the lack of knowledge on the platform capabilities.			
16.7.2012	Verified	All tests succeeded.			

* This task card, as all other TCs, was defined during the planning day but was refined during the implementation.

Task card TC-0-2 - Create database model

TC-0-2	Type	Difficulty		Confi-dence	Notes
		Before	After		
	New	1	1	5	

Description

The database model for mobile and web service part of the system should be created. The model should be easy as it is only a trial of whole database model that is to be implemented in later phases.

Date	Status	Comment
11.7.2012	Defined	
12.7.2012	Implementing	The part of database model important for this story is created for mobile application and for web service.
13.7.2012	Done	The database containing defined entities is up and running on hosting provider. The model on mobile application will be deployed through database layer.
16.7.2012	Verified	All tests on mobile application succeeded.

Task card TC-0-3 - Create database layer in mobile app

TC-0-3	Type	Difficulty		Confi-dence	Notes
		Before	After		
	New	3	3	5	

Description

The database layer is a set of classes that are responsible to create and maintain local SQLite database, as well as to provide the access to the data (i.e. create, read, update or delete) data.

Date	Status	Comment
11.7.2012	Defined	
12.7.2012	Implementing	The database layer is relatively easy to create but hard to test as it should be tested in context of other application functionality. This will be done while implementing task of defining synchronization layer.
13.7.2012	Done	Currently layer contains base class for accessing database, plus entity class user for accessing the information on user in database.
16.7.2012	Verified	All test succeeded.

Task card TC-0-4 - Create database layer in web app

TC-0-4	Type	Difficulty		Confi-dence	Notes
		Before	After		
	New	5	5	3	

Description

The database layer is a set of classes that are responsible to create and maintain local MySQL database, as well as to provide the access to the data (i.e. create, read, update or delete) data. The classes should be accessible through exposed web services with corresponding exposed methods.

Date	Status	Comment
11.7.2012	Defined	
12.7.2012	Implementing	Using phpMyAdmin, the database is successfully created on MySQL server. Additionally, web service and supporting classes are being developed.
13.7.2012	Done	The exposed web service along with supporting classes are created and tested locally. The security mechanisms are not included as these are not required by user requirements.
16.7.2012	Verified	Integration and acceptance tests succeeded.

Task card TC-0-5 – Implement and connect user interface

TC-0-5	Type	Difficulty		Confi-dence	Notes
		Before	After		
	New	2	2	5	

Description

Corresponding user interface for entering the data in mobile application should be created. The elements of user interface, as well as other messages communicated to the user should be language independent, but implemented in English. The functionality of user interface should through corresponding activity classes be connected to database layer.

Date	Status	Comment
11.7.2012	Defined	
12.7.2012	Implementing	As the user interface for profile is not the first screen in the application, auxiliary operations were implemented in order to be able to navigate to target page. *activity_profile.xml* is being created and should be connected to business logic layer class ProfileActivity.java.
13.7.2012	Done	The user interface is created and is language independent, screen size independent and orientation independent.
16.7.2012	Verified	All tests including acceptance test succeeded.

Task card TC-0-6 – Add synchronization layer

TC-0-6	Type	Difficulty		Confi-dence	Notes
		Before	After		
	New	3	5	4	

Description

The data stored in local database should be automatically synchronized to web service.

Date	Status	Comment
11.7.2012	Defined	
13.7.2012	Implementing	The classes and behavior necessary for data synchronization between application and web service are created. KnowledgeService.java and JsonAdapter.java are created and ProfileActivity.java is seriously improved.
13.7.2012	Done	The data cannot be stored in local database unless the user is created by web service which returns the user id. After the user is created, it can be only updated.
16.7.2012	Verified	All tests succeeded.

Task card TC-0-7 – Finalize tests

TC-0-7	Type	Difficulty		Confi-dence	Notes
		Before	After		
	Enhance	5	5	3	

Description

All created functionality should be tested thoroughly; the test for core and robotized testing should be updated and saved. If necessary, code should be updated and fixed.

Date	Status	Comment
11.7.2012	Defined	
13.7.2012	Implementing	Some tests concerning core functionality were defined in previous task. Now other tests dependent on technological specifications should be defined, and finally the test defining robotized integration testing of application is to be created.
16.7.2012	Done	17 fully automatic tests are created. Code is refactored and fixed. More than 100

		assertions in included in 16 unit (more than 85) and 1 integration (more than 15) tests.
16.7.2012	Verified	All tests succeeded.

Task card TC-0-8 – Optimize and refactor

TC-0-8	**Type**	**Difficulty**		**Confi-dence**	**Notes**
		Before	**After**		
	Enhance	1	1	5	
Description					
Created code should be optimized, commented and refactored. All tests should execute successfully at the end.					
Date	**Status**	**Comment**			
11.7.2012	Defined				
16.7.2012	Implementing	Considerable efforts were made during the implementation, so there was no much work to do during the refactoring task. Instead, the classes and methods are fully commented.			
16.7.2012	Done				
16.7.2012	Verified	All tests succeeded.			

3.3.7.2. Data model

The requirements analysis showed that this trial day concerns only the functionality regarding one entity in data model. User entity was defined as follows.

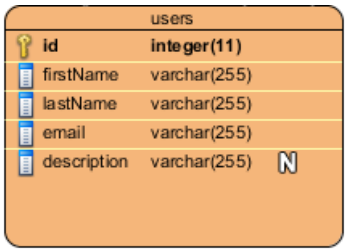


Figure 20 - Entity users (trial day)

The same data model was deployed on mobile database and on web service database hosted online.

3.3.7.3. Created web service

Exposed web service covering the functionality of managing the system users is exposed and can be accessed by the URL: http://knowledge.uphero.com/users.php. The frontend part of the web application is accessible to the mobile application through several methods that are described in Table 27. Other functionality is defined in the backend and cannot be accessed

directly, but still plays a crucial role in the functionality of the web service. The model of the whole web application (with web service) is presented in the next chapter.

Table 27 - Web service (users.php) specification

http://knowledge.uphero.com/users.php			
method	**json***	**response****	**description**
create	firstName lastName email [description]	responseId responseText [newUserId]	Creates a new user in database. Compulsory data in post include method name and the data about new user packed into JSON format. Web service will return JSON formatted string. If everything was OK the string will contain additional data on newUserId.
update	id firstName lastName email [description]	responseId responseText	Updates an existing user in the database. Compulsory data in post include method name and the updated data about user packed into JSON format. Web service returns JSON formatted string containing the operation result id and text.
delete	id	responseId responseText	Deletes and existing user from the database. Compulsory data in post include method name and user id. Web service returns JSON formatted string containing the operation result id and text.

* json – parameter name. Should contain all stated elements in JSON format.
** response – String response from web service. Contains all stated attributes in JSON format.

3.3.7.4. Created class models

As the feature selected for the trial day spans vertically through the whole system architecture, the class model designed and created during this phase is not so simple. The model contains classes for database connectivity layer, business logic layer, user interface layer plus some helper classes to connect to web service. The model of the mobile application is presented in Figure 21.

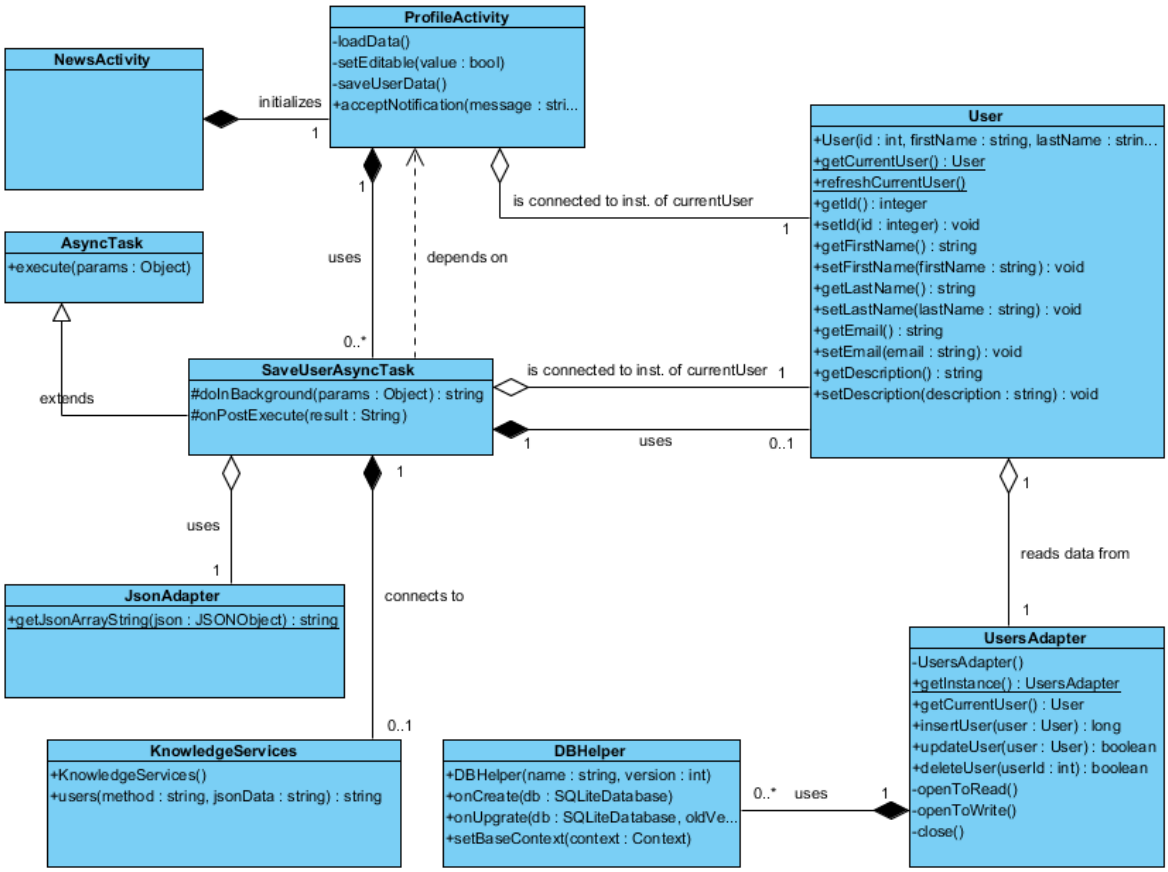


Figure 21 - Class diagram (mobile app - trial day)

Class *NewsActivity* was used only to provide the functionality of opening the target *ProfileActivity* class and thus is not defined at this phase. Additionally, some classes extend native Android classes, but these are not presented unless it was necessary in order to understand the navigability through the model (e.g. *AsyncTask* provides method execute which was called by *ProfileActivity*, as the method in *SaveUserAsyncTask* are protected and thus inaccessible from mentioned *ProfileActivity* class). The private attributes are hidden in the diagram as they are irrelevant in this report. Finally, many classes use native Android classes which are not shown in this diagram in order to make it clean and simple and direct the focus only on the architectural design.

On the other hand, as presented in Figure 22, web application comprises of one exposed web service (*users.php*) which is backed up by several classes providing the means of accessing and storing the data and loading the necessary configuration.

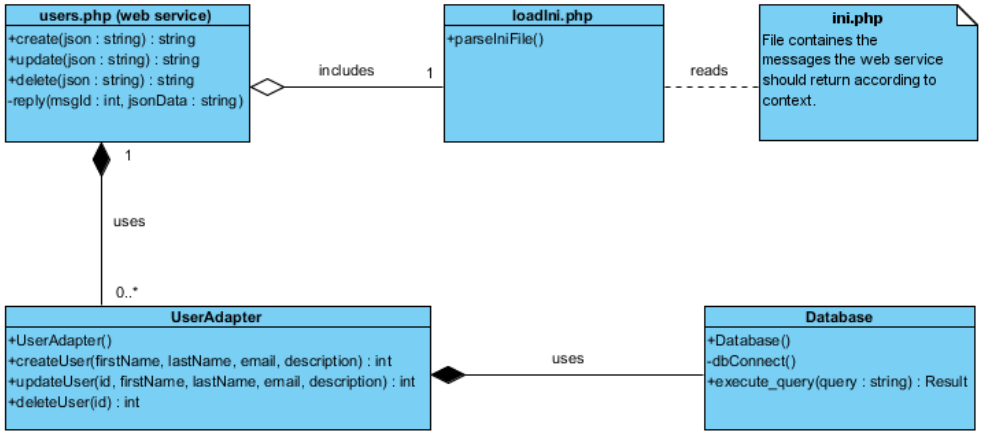


Figure 22 - Class diagram (web service - trial day)

3.3.7.5. Implementation

During the implementation tasks, the classes presented in the above figures were implemented in Java and PHP. According to the Mobile-D methodology, very strict coding standards were applied, and at the end of the implementation process, the code was commented. An example of a part of a commented class is shown in Code 2. As it can be seen, the comments include the description and the tags defining the author, date, connecting task and other elements usual for code comments (such as see also, code etc.).

```
package foi.uah.knowledge.entities;
import foi.uah.knowledge.database.UsersAdapter;


/**
 * Class represents an User entity. When ever in application the information about
 * the user should be used it should be provided by this class. As the application
 * can only have one user, the behavior of this class is some-what specific.
 *
 * @author     Zlatko
 * @date       13.7.2012.
 * @task       TC-0-2
 */
public class User {
       private static User currentUser;
       private int id;
       private String firstName = "";
       private String lastName = "";
       private String email = "";
       private String description = "";

       /**
        * Constructor which creates new user according to given parameters.
        *
        * @param id          User id. The value should be obtained from web service.
        * @param firstName   First name. Compulsory.
        * @param lastName    Last name. Compulsory.
        * @param email               Email address. Compulsory.
        * @param description An optional description of user to be created.
        *
```

```
         * @author  Zlatko
         * @date      13.7.2012.
         * @task      TC-0-2
         * @changes
         */
        public User(int id, String firstName, String lastName, String email, String
                        description)
        {
                setId(id);
                setFirstName(firstName);
                setLastName(lastName);
                setEmail(email);
                setDescription(description);
        }

        /**
         * Static method returns object with information on current user written in
         * database. If data in database is changed, the information on current user
         * will not change automatically, and thus the
         * refreshCurrentUser method should be used.
         *
         * @see #refreshCurrentUser()
         * @return An object with information on current user, if such exist in
         * database.
         *
         * @author    Zlatko
         * @date      13.7.2012.
         * @task      TC-2-2
         * @changes
         */
        public static User getCurrentUser()
        {
                if (currentUser == null)
                {
                        UsersAdapter ua = UsersAdapter.getInstance();
                        currentUser = ua.getCurrentUser();
                }
                return currentUser;
        }


        /**
         * Static method which refreshes the current object with the latest data on
         * user in database. This method should be called whenever the database
         * information is changed.
         *
         * @author    Zlatko
         * @date      13.7.2012.
         * @task      TC-0-2
         * @changes
         */
        public static void refreshCurrentUser()
        {
                currentUser = null;
                UsersAdapter ua = UsersAdapter.getInstance();
                currentUser = ua.getCurrentUser();
        }
...
...


}
```

Code 2 - Commented class

Additionally, the best practices of object oriented programming (abstraction, inheritance, encapsulation, polymorphism, error handling etc.) were used (Mitchell, 2003), which resulted in a high internal cohesion (Miller, 2008) and at the same time the external coupling was kept low. Although the trial day resulted in a relatively small number of classes, the same principles were applied during the whole development process.

3.3.7.6. Testing

As the Mobile-D methodology suggests (Abrahamsson et al., 2005a), the whole development process was based on Test Driven Development (TDD) (Hammond and Umphress, 2012). As it is visible from the defined tasks, the working day began with the activities of writing the unit tests for core functionality. As some of the technological aspects were not familiar to the implementer of this task (i.e. me, a PhD student), the task resulted with only a few basic unit tests regarding already familiar and known classes.

Other unit tests were written during the development and the TC-0-7 (Finalize tests) task. The whole process resulted in 16 unit tests which completely automatically asserted more than 85 different conditions.

The integration testing was also automatized by defining the *Robotium* test (Reda, 2012) which robotically runs the application on mobile phone or on simulator and performs all possible actions including creation of the user, inaccurate attempts of updating the user, accurate updating tests and similar. The integration testing thus included the tests of some features that were impossible to test by unit testing (like asynchronous behavior of some classes).

In the end, and after the refactoring, all 17 tests (16 unit tests + one integration test) were successfully run, and more than 100 assertions gave expected results as shown in Figure 23.

As it can be seen from the test results, only two tests were time consuming. The web service test took more than 10 seconds as it called the web service more than 15 times. Additionally, the automated integration robotic test took more than 40 seconds, as it tested the application as a user would. These results were expected and also confirmed that there were no other time-heavy objects.

Finished after 56,853 seconds

Runs: 17/17 Errors: 0 Failures: 0

- emulator-5554 [Runner: JUnit 3] (56,677 s)
  - foi.uah.knowledge.test.ProfileActivityRobotiumTest (42,496 s)
    - testRobotium_UserProfile (42,496 s)
  - foi.uah.knowledge.test.database.UsersAdapterTest (1,538 s)
    - testConstructorSingleton (0,461 s)
    - testDeleteUser (0,377 s)
    - testInsertUser (0,400 s)
    - testUpdateUser (0,200 s)
    - testAndroidTestCaseSetupProperly (0,100 s)
  - foi.uah.knowledge.test.entities.SaveUserAsyncTaskTest (0,050 s)
    - testAndroidTestCaseSetupProperly (0,050 s)
  - foi.uah.knowledge.test.entities.UserTest (0,426 s)
    - testConstructor (0,101 s)
    - testCurrentUser (0,075 s)
    - testGettersAndSetters (0,100 s)
    - testManyInstances (0,075 s)
    - testAndroidTestCaseSetupProperly (0,075 s)
  - foi.uah.knowledge.test.lib.JsonAdapterTest (0,250 s)
    - testGetJsonArrayString (0,075 s)
    - testAndroidTestCaseSetupProperly (0,175 s)
  - foi.uah.knowledge.test.services.KnowledgeServicesTest (11,917 s)
    - testUserCreate (4,745 s)
    - testUserUpdate (7,072 s)
    - testAndroidTestCaseSetupProperly (0,100 s)

Figure 23 - Test results (trial day)

Finally, all tests were designed by accepting the Mobile-D recommendations (Abrahamsson et al., 2005a) on performing the test driven development. Additionally the tests were designed in such a manner that the order of execution of tests was not important, the tests were not dependent on any existing system configuration and revert original data in local database and thus did not interfere with manual testing performed during the development.

3.3.7.7. Application screenshots

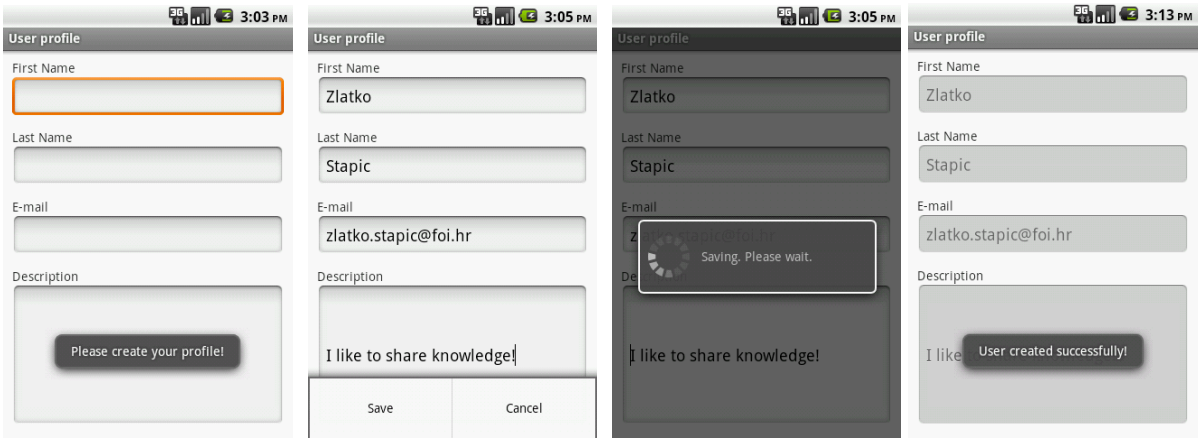


Figure 24 - Application screenshots (trial day)

Above figure shows several screenshots taken at the end of the trial day. These screenshots show only one use case which was implemented during this phase and do not cover the whole implemented functionality. The whole functionality was successfully tested during the execution of the corresponding acceptance test.

3.3.7.8. Project plan checklist

At the end of this stage there was no need for performing the usual activities of the release day. All tests including the acceptance test were performed successfully and the documentation including the artifacts of everything that was done was wrapped up. Finally in order to check if everything was done correctly, the requirements defined by the Mobile-D and stated in the check list (see Table 28) were checked.

Table 28 - Project plan checklist – 0 Iteration

0 Iteration	**Yes**	**No**	**NA**
Requirements			
The project plan has been updated concerning the selected trial requirements for 0 iteration	X		
The project plan has been updated concerning the realization of the selected trial requirements for the 0 iteration	X		
Architecture line definition has been included in the project plan	X		

3.4. Productionize

3.4.1. First iteration

The selected features to be implemented in this iteration are presented in Table 29 and mainly concern the manipulation of groups owned by user. The reason for having these features selected is that the functionality regarding group management set up the basis for other functionalities. As stated in the project backlog, the importance of F5.1 and F4.1 is very high, which also justifies the decision. Although the F5.3 is currently not important, the email invitations are easy to implement and tightly connected with F5.2 and thus this easy task is included in this iteration as well. As it can be seen in the following table, the order of the execution was slightly changed.

Table 29 - Selected features for first iteration

Features / stories		Importance
F5.1	User should be able to create a new group.	5
F4.1	User should be able to see the list of all groups currently enrolled to.	5
F5.4	User should be able to delete any group he owns.	2
F5.2	User should be able to invite new members to his group by inviting them via in application email.	2
F5.3	User should be able to invite new members to his group by sending them email.	1

3.4.1.1. Story cards and task cards

Story card F5.1

F5.1	Type	Difficulty		Effort		Priority	Notes
		Before	After	Estim.	Spent		
	New	L	M	4	5	5	
Description							
User should be able to create a new group. The basic information about the group should include name, description and creator. The information should be stored in web database and downloaded locally when necessary through web service.							
Date	**Status**	**Comment**					
17.7.2012	Defined	This functionality is prerequisite for most of other functionality of this iteration as well as of following iterations. It should be implemented by calling appropriate web service and displaying the results.					
19.7.2012	Implementing	The approach established during the trial day is taken in implementation of this feature. The only difference is that groups should not be stored in local database after created and confirmed from the web service.					
23.7.2012	Done	The functionality is created.					
26.7.2012	Enhanced	The refactoring was made and the code is significantly improved and made simple but sill functional.					
27.7.2012	Verified	All tests succeeded.					

Story card F4.1

F4.1	Type	Difficulty		Effort		Priority	Notes
		Before	After	Estim.	Spent		
	New	L	M	4	5	5	Partial implementation!
Description							
User should be able to see the list of all groups currently enrolled to. The basic information about the group should include name, description and number of members. The information should be stored in web database and downloaded locally when necessary through web service. This functionality will be partially implemented in this phase as currently there is no possibility to see invitations and to accept them and thus user will not be enrolled in any group except own groups.							
Date	**Status**	**Comment**					
17.7.2012	Defined	This functionality is prerequisite for most of other functionality of this iteration as well as of following iterations. It should be implemented by calling appropriate web service and displaying the results.					
19.7.2012	Implementing	The implementation of web role is focused in this task as it performs the most important logic. The mobile application will receive and display the data.					
24.7.2012	Done	It took us little bit longer than expected to finish this task. The web service role was hard to debug. This problem should not be neglected while preparing the implementation of other requirements.					
26.7.2012	Enhanced	The refactoring was made and the code is significantly improved and made simple but sill functional.					

27.7.2012	Verified	All tests succeeded.

Story card F5.4

F5.4	Type	Difficulty		Effort		Priority	Notes
		Before	After	Estim.	Spent		
	New	L	L	3	4	2	

Description

User should be able to delete any group he owns.
The group will not be deleted from the web service, but it will be rather marked as deleted and will stay in database for analytical purposes.

Date	Status	Comment
17.7.2012	Defined	Appropriate web service should be called and the data in database should be marked as deleted but kept for analytical purposes.
19.7.2012	Implementing	The mobile side of the system should do the majority of work including the communication with the user and preparation of data to be sent to web service.
25.7.2012	Done	The user is asked to confirm the action and after the parameters are sent to web service which logically marks the group as deleted.
26.7.2012	Enhanced	The refactoring was made and the code is significantly improved and made simple but sill functional.
27.7.2012	Verified	All tests succeeded.

Story card F5.2

F5.2	Type	Difficulty		Effort		Priority	Notes
		Before	After	Estim.	Spent		
	New	L	M	3	4	2	

Description

User should be able to invite new members to his group by inviting them via in application email.
In-application emails should be implemented through web database. This means that the email should be “sent” by marking the information in database, and “read” after the client application will ask for news feed. This news should include “emails”.

Date	Status	Comment
17.7.2012	Defined	Appropriate web service should be called and the email should marked in appropriate database entity.
19.7.2012	Implementing	The mobile side of the system should do the majority of work including the communication with the user and preparation of data to be sent to web service.
26.7.2012	Done	The data collected from UI and local objects is sent to web service.
26.7.2012	Enhanced	The refactoring was made and the code is significantly improved and made simple but sill functional.
27.7.2012	Verified	All tests succeeded.

Story card F5.3

F5.3	Type	Difficulty		Effort		Priority	Notes
		Before	After	Estim.	Spent		
	New	L	1	2	2	1	

Description

User should be able to invite new members to his group by sending them email.
The simple email should be sent from the web server and it should contain the information that there is new invitation to group. In the application, the user should see the invitation after contacting the web service for news again as described in F5.2.

Date	Status	Comment
17.7.2012	Defined	The email should be sent automatically after calling the web service in F5.2 if appropriate parameter is present.
19.7.2012	Implementing	The implementation of this requirement will be realized through the implementation of F5.2 requirement.
26.7.2012	Done	The necessary changes in existing functionality of mobile and web service are created. Web service is enhanced with the functionality of preparing and sending the e-mail messages.
26.7.2012	Enhanced	The refactoring was made and the code is significantly improved and made simple but sill functional.
27.7.2012	Verified	All tests succeeded.

By analyzing the aforementioned user stories, we concluded that the best approach is to combine all five of them into a single sequence of tasks. This decision was made as the functionality described in these user stories is strongly interconnected and interdependent. The tasks identified are described by the following task cards.

Task card TC-1-1 - Create initial test cases

TC-1-1	Type	Difficulty		Confi-dence	Notes
		Before	After		
	New	5	5	3	
Description					
Initial test cases for these functionalities should be created.					
Date	**Status**	**Comment**			
17.7.2012	Defined	The agreed and tried *android.test* and *robotium* framework should be used.			
19.7.2012	Implementing	The analysis showed that there are not many new classes in mobile application suitable for unit testing, but on the other hand the test for web services should be prepared.			
19.7.2012	Done	The unit tests concerning the functionality of mobile application classes and synchronous communication with web services are created.			
27.7.2012	Verified	The tests are finalized and successful in run.			

Task card TC-1-2 – Update database model

TC-1-2	Type	Difficulty		Confi-dence	Notes
		Before	After		
	Enhance	1	1	5	
Description					
Web application database model should be updated. It should be an easy task as there will probably be no changes on existing model. On the other hand, several more entities should be created in order to cover all functionality for this iteration.					
Date	**Status**	**Comment**			
17.7.2012	Defined				
19.7.2012	Implementing	It is not necessary to alter existing model.			
20.7.2012	Done	New model includes entities users, groups and enrolments and is capable of storing data on users and on active and inactive (canceled) groups and enrolments.			
27.7.2012	Verified	All tests succeeded and the model is suitable for current requirements.			

Task card TC-1-3 – Implement server side functionality

TC-1-3	Type	Difficulty		Confi-dence	Notes
		Before	After		
	New	4	4	4	

Description

Web service leaning on the upgraded data model should be written. It should include exposed methods as well as backend supporting functionality. The approach created during the trial day should be used.

Date	Status	Comment
17.7.2012	Defined	
20.7.2012	Implementing	All features in this iteration are counting on web service support. Thus the planned services should be carefully implemented and error free.
23.7.2012	Done	This task took longer than expected to be finished. The majority of functionality is supported by web services and the development of those is time consuming and hard to debug. In any case the planned services are developed and ready for usage.
27.7.2012	Verified	All tests succeeded.

Task card TC-1-4 – Implement mobile app functionality

TC-1-4	Type	Difficulty		Confi-dence	Notes
		Before	After		
	New	4	5	4	

Description

Using the basics of infrastructure created during the trial day, new classes should be developed and connected to display the data in appropriate user interface (see UI sketches). The information should be downloaded from the web services in real time.

Date	Status	Comment
17.7.2012	Defined	
23.7.2012	Implementing	There are several new concepts which are not tried (prototyped) but are to be developed. These concepts include the usage of custom dialogs, the handling of user actions and hardware keys etc.
26.7.2012	Done	This task also took longer than expected. The main reason is the development of not trialed concepts and little bit complicated infrastructure that resulted in asynchronous communication. This source should be refactored.
26.7.2012	Enhanced	The source is heavily refactored. The service layer is made free of business logic and is now only used for communication with web services. This reduced the number of classes in service layer.
27.7.2012	Verified	All tests succeeded.

Task card TC-1-5 – Finalize tests

TC-1-5	Type	Difficulty		Confi-dence	Notes
		Before	After		
	Enhance	5	5	3	

Description

All created functionality should be tested thoroughly; the test for core and robotized testing should be updated and saved. If necessary, code should be updated and fixed.

Date	Status	Comment
17.7.2012	Defined	The agreed and tried *android.test* and *robotium* framework should be used.
26.7.2012	Implementing	This task should include the preparation of integration tests. During the test design is concluded that isolation of test cases could be the problem.
26.7.2012	Done	All integration tests are created in one sequence. Although this is not good

		approach, the execution of isolated test cases proved to be very time consuming as every test case has to prepare the context from scratch.
27.7.2012	Verified	All tests succeeded.

Task card TC-1-6 – Optimize and refactor

TC-1-6	**Type**	**Difficulty**		**Confi-dence**	**Notes**
		Before	**After**		
	Enhance	1	2	5	
Description					
Created code should be optimized, commented and refactored. All tests should execute successfully at the end.					
Date	**Status**	**Comment**			
17.7.2012	Defined				
26.7.2012	Implementing	The asynchronous nature of the communication with web service and wrong infrastructure design made the service layer very heavy. Current class-per-service-call environment is dealing with preparation of data and business logic. This is not good.			
27.7.2012	Done	The preparation of data and business logic was moved to the real business logic layer which made the service layer very simple. This resulted in several new classes which ensure proper communication between these two layers.			
27.7.2012	Verified	All tests succeeded.			

3.4.1.2. Database model

Updated database model was initially created during the planning day, and slightly updated during the working days. The final version satisfying all requirements of this phase can be seen in the following picture. Only the database model representing server side functionality was updated.

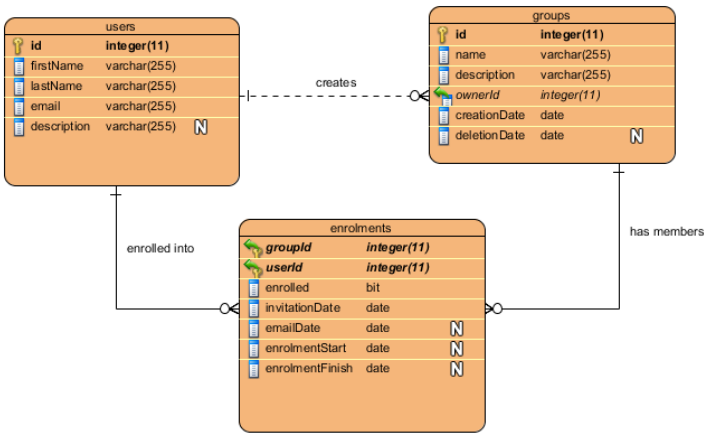


Figure 25 - Data model (iteration 1)

The important information was stored in *groups* and *enrolments* entities. These entities are designed to store information on currently active, but also on inactive groups and enrolments.

After the group is created, the owner is automatically enrolled into a group (*enrolled = 1* and *enrolmentStart = currentDate*). After the owner invites another member, a new record is added to the enrolments table, but the information keeping attribute this time is *invitationDate* which is set to *currentDate*, and other attributes await for user to accept (or reject) the invitation. After the group is deleted, its *deletionDate* is set up and for all members of that group, enrollment is canceled by setting the *enrolled* to *0* and *enrolmentFinish* to *currentDate*. Thus, the database model ensures proper navigability and information preservation and can be considered as valid.

3.4.1.3. Created web services

The following tables describe created web services, their methods and corresponding parameters sent and received in JSON format. Some of the listed web services are still not used and thus not included in any test.

Table 30 - Web service (groups.php) specification

http://knowledge.uphero.com/groups.php			
method	**json**	**response**	**description**
create	name description ownerId	responseId responseText [newGroupId]	Creates a new group in database. The owner of the group is automatically enrolled in the new group. If everything was OK the return string will contain additional data on *newGroupId*.
update	id name description	responseId responseText	Updates an existing group in the database. Only name and description are allowed to be changed. Web service returns usual response.
delete	id	responseId responseText	Logically deletes existing group from the database by setting the *deletionDate* value. All memberships are automatically canceled by setting the *enrolled* = 0 and *enrolmentFinished* valued. Web service returns usual response.
my	ownerId	responseId responseText [groups]	Returns JSON string containing an array of groups owned by given user. The information contains a number of members in every group.

Table 31 - Web service (enrolments.php) specification

http://knowledge.uphero.com/enrolments.php			
method	**json***	**response****	**description**
inviteUser	groupId inviterId email [sendEmail]	responseId responseText	Adds new enrolment invitation in database. Optional data includes parameter *sendEmail* that defines if normal email invitation should be sent or not. Only *invitationDate* and optionally *emailDate* attributes are defined. Web service returns usual response.
enroll	groupId	responseId	Enrolls user in a group. In this iteration the method is

	userId	responseText	not used, thus it is not jet tested by service or integration tests. Web service returns usual response.
cancel	groupId userId	responseId responseText	Cancels the user's enrolment by setting the *enrolled* to false and noting down the withdraw date. This service is not jet used and thus is not tested. Web service returns usual response.

3.4.1.4. Created class models

During the planning day, the technology independent class model was created, but during the working days it was slightly improved to fit the target platform. The second version of the class model included some technology dependent classes like *AsyncTask* which are specific for Android platform. In any case, the specific focus was given so the class model can be re-used during the development of application for other target platforms.

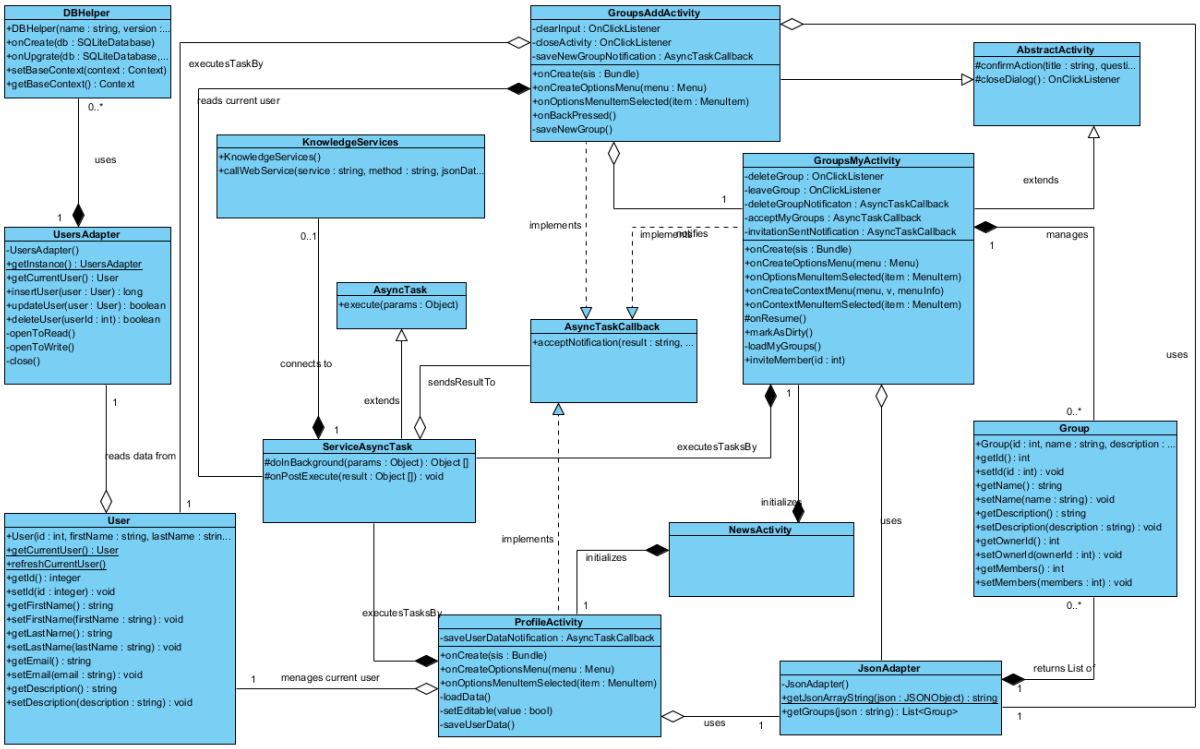


Figure 26 - Mobile app class model (iteration 1)

Although a little complicated, the architecture of the mobile application was still flexible and modular. As it can be seen, activity classes are the most important part of the functionality. Those classes execute tasks by *ServiceAsyncTask* class which asynchronously communicates with web service, and sends the result through *AsyncTaskCallback* interface that is implemented by the caller.

The new entity added in this iteration was *Group* entity. This class is simple as it is just used to encapsulate the data download from the web service.

JsonAdapter is a static class providing helpful functionality when working with JSON objects and strings, and finally, the only class that deals with local database is class *User* which provides information on the current user.

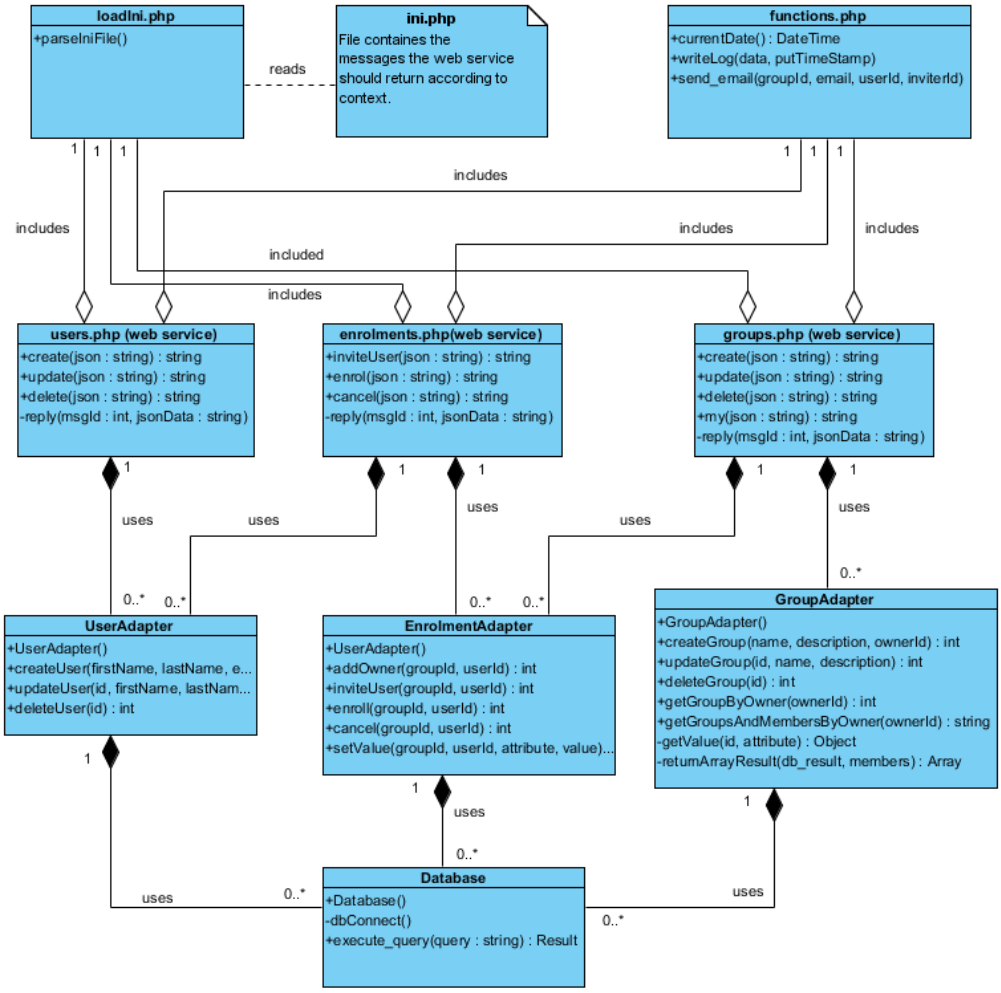


Figure 27 - Web app class model (iteration 1)

In the web application, the infrastructure was not changed. The web services were backed up with adapters which communicate with the web database.

3.4.1.5. Implementation

The most important infrastructural functionality developed in this phase concerns the communication with the web services. The implementation protocols and practices established and described during the trial day phase were closely followed in this phase as well. The model developed during the trial day was insufficiently flexible and had to be improved as there were many calls to the web services. The following example shows the new approach in solving this problem.

```java
/**
 * The method coordinates the web service call/response. The data is obtained,
 * prepared and sent to service proxy. The results will be asynchronously received
 * by AsyncCallbackTask pointed when calling the proxy.
 *
 * @author     Zlatko
 * @date       13.7.2012.
 * @task       TC-0-6
 * @changes    26.7.2012
 */
private void saveUserData()
{
        try{
                //getting the data
                strFirstName = txtFirstName.getText().toString();
                strLastName = txtLastName.getText().toString();
                strEmail = txtEmail.getText().toString();
                strDescription = txtDescription.getText().toString();

                String method = "";
                String responseAttribute = "";

                //preparing json object
                JSONObject jsonObject = new JSONObject();
                jsonObject.put("firstName", strFirstName);
                jsonObject.put("lastName", strLastName);
                jsonObject.put("email", strEmail);
                jsonObject.put("description", strDescription);

                if (User.getCurrentUser() == null) {
                        method = "create";
                        responseAttribute = "newUserId";
                }
                else {
                        method = "update";
                        jsonObject.put("id", User.getCurrentUser().getId());
                }
                String jsonString = JsonAdapter.getJsonArrayString(jsonObject);

                //calling the service and showing progress dialog
                ServiceAsyncTask asyncTask = new ServiceAsyncTask();
                ProgressDialog dialog = ProgressDialog.show(this, "",
                     getResources().getString(R.string.dialogSaving), true, true);
                Object params[] = new Object[]{this, jsonString, "users", method,
                     responseAttribute, dialog, saveUserDataNotification};
                asyncTask.execute(params);
        }
        catch (JSONException e) { }
}

/**
 * This callback task is called after web service returns the results. According
 * to the results, it is necessary to perform synchronization with local databas
 * and to inform the user on actions performed. The data will be stored in
 * local database only if web service request responds with message 100 (OK). The
 * method inserts data in local database
 * only first time and after that it only updates the data.
 *
 * @author     Zlatko
 * @date       26.7.2012.
 * @task       TC-1-6
 * @changes
 */
AsyncTaskCallback saveUserDataNotification = new AsyncTaskCallback() {

        @Override
```

```
        public void acceptNotification(String result, boolean ok) {
                if (ok) {
                        if (User.getCurrentUser() == null){
                                //create new user in local database
                                int id = Integer.parseInt(result);
                                User user = new User(id, strFirstName, strLastName,
                                    strEmail, strDescription);
                                UsersAdapter.getInstance().insertUser(user);
                                Toast.makeText(context, getResources().getString (R.
                                    string.msgUserCreated), Toast.LENGTH_LONG).show();
                        }
                        else{
                                //update data in local database
                                int id = User.getCurrentUser().getId();
                                User user = new User(id, strFirstName, strLastName,
                                    strEmail, strDescription);
                                UsersAdapter.getInstance().updateUser(user);
                                Toast.makeText(context, getResources().getString(R.
                                    string.msgUserUpdated), Toast.LENGTH_LONG).show();
                        }
                        setEditable(false);
                }else{
                        Toast.makeText(context, result, Toast.LENGTH_LONG).show();
                        }
                }
    };
```

Code 3 - Handling web service call and response

The code example shows the basic approach taken in handling web service call and response. Before calling the *asyncTask,* the data is obtained and prepared into JSON object. Additionally, other parameters are also prepared, along with JSON data packed into a single object with a predefined structure, and sent to the proxy to communicate with web service. After gaining the async callback, the results are analyzed and the data is synchronized with the local database. This approach allows similar communication with web service from any object in mobile application.

3.4.1.6. Testing

During the implementation of the respective tasks concerning testing, we faced several important challenges. The implementation resulted in few classes suitable for unit testing. Despite that, the unit tests were created in advance for all classes which were used in the application except the classes which deal with asynchronous communication with web services. Additionally, the complete suite of unit tests was created to test the web services directly.

On the other hand, asynchronous behavior was also tested, but through the sequential fully automatized integration test which additionally tests the behavior of activities. At the end of the iteration, a total of 26 tests with approximately 200 assertions were run and were completely successful.

Runs: 26/26 Errors: 0 Failures: 0

- htc-htc_wildfire-HT0C2PY06311 [Runner: JUnit 3] (222,794 s)
  - foi.uah.knowledge.test.IntegrationRobotiumTest (191,930 s)
    - testRobotiumIntegration (191,930 s)
  - foi.uah.knowledge.test.database.UsersAdapterTest (1,575 s)
    - testConstructorSingleton (0,250 s)
    - testDeleteUser (0,325 s)
    - testInsertUser (0,225 s)
    - testUpdateUser (0,350 s)
    - testAndroidTestCaseSetupProperly (0,425 s)
  - foi.uah.knowledge.test.entities.GroupTest (0,600 s)
    - testConstructor (0,275 s)
    - testGetersAndSetters (0,125 s)
    - testManyInstances (0,125 s)
    - testAndroidTestCaseSetupProperly (0,075 s)
  - foi.uah.knowledge.test.entities.UserTest (0,900 s)
    - testConstructor (0,100 s)
    - testCurrentUser (0,100 s)
    - testGettersAndSetters (0,325 s)
    - testManyInstances (0,225 s)
    - testAndroidTestCaseSetupProperly (0,150 s)
  - foi.uah.knowledge.test.lib.JsonAdapterTest (0,175 s)
    - testGetGroups (0,050 s)
    - testGetJsonArrayString (0,075 s)
    - testAndroidTestCaseSetupProperly (0,050 s)
  - foi.uah.knowledge.test.services.KnowledgeServicesTest (27,514 s)
    - testEnrolmentsInviteUser (5,151 s)
    - testGroupCreate (3,425 s)
    - testGroupDelete (3,051 s)
    - testGroupGetMy (4,600 s)
    - testUserCreate (4,562 s)
    - testUserUpdate (6,625 s)
    - testAndroidTestCaseSetupProperly (0,100 s)
  - foi.uah.knowledge.test.services.ServiceAsyncTaskTest (0,100 s)
    - testAndroidTestCaseSetupProperly (0,100 s)

Figure 28 - Test results (iteration 1)

3.4.1.7. Application screenshots

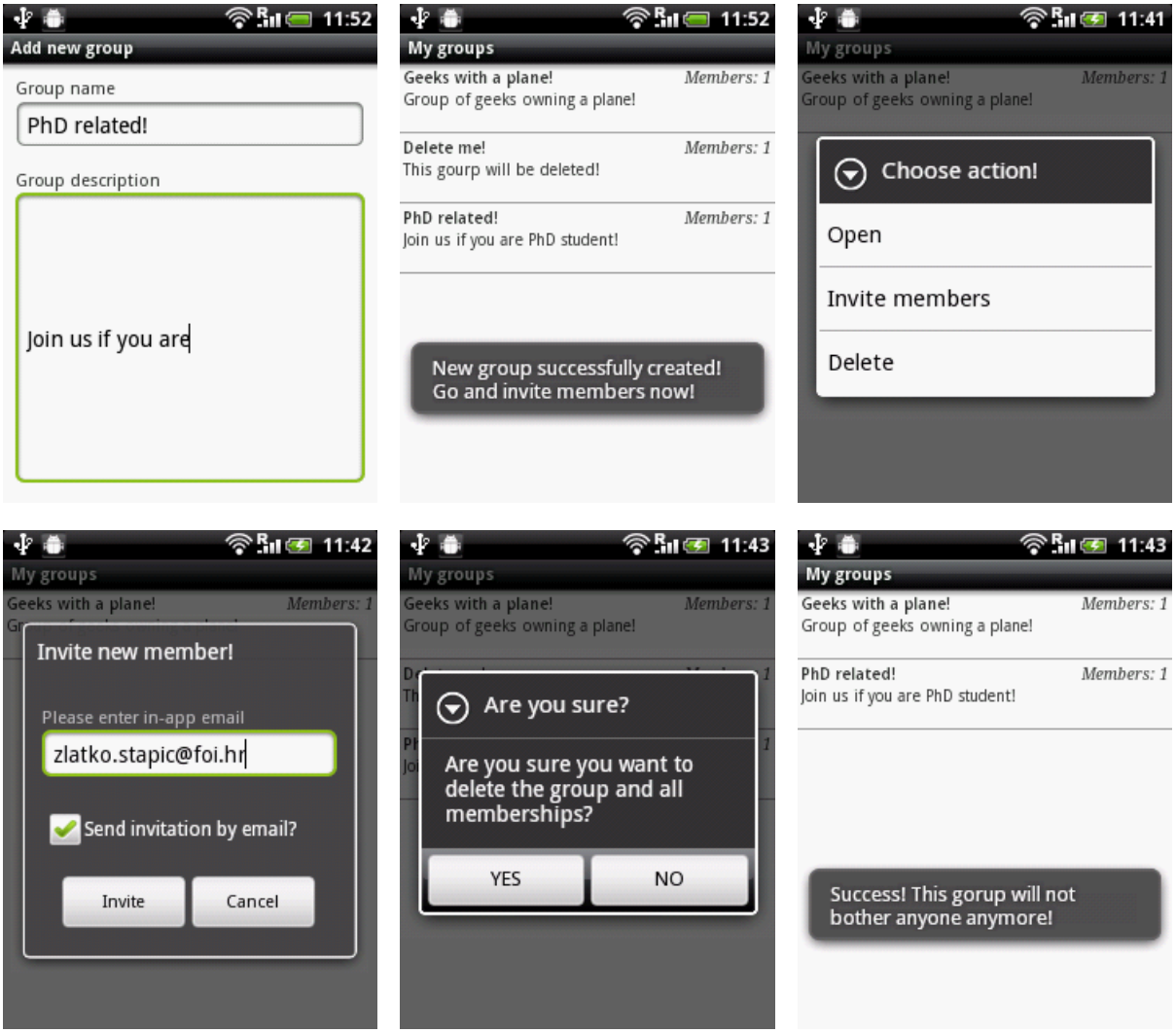


Figure 29 - Application screenshots (iteration 1)

Above figure shows several screenshots taken at the end of the first iteration.

3.4.1.8. Project plan checklist

At the end of this stage there was no need for performing the usual activities of the release day. All tests including the acceptance tests are performed successfully, the documentation including artifacts of everything that is done is wrapped up, and finally in order to check if everything is done correctly, the requirements defined by the Mobile-D and stated in the check list (see Table 32) are checked.

Table 32 - Project plan checklist – 0 Iteration

Productionize Iteration(s)			
Requirements			
The project plan has been updated concerning the selected requirements for the current iteration	X		
The project plan has been updated concerning the realization of the selected requirements for the current iteration	X		
The project plan has been updated concerning any changes in, e.g., the schedule, rhythm, requirements, and resources	X		
The project plan has been updated concerning the realization of quality assurance activities in current iteration	X		

3.4.2. Other iterations

As had been planned, all other iterations were performed in a similar manner. As the objective was to identify the artifacts, there is no need to report all the iterations in detail here. Rather, this chapter will present the summary information on the performed tasks and outputs, as well as give the final versions of some important documents.

3.4.2.1. Iterations overview

According to iterations plan which was a part of the overall project plan, the four remaining iterations included the implementation of user stories (features) as presented in Table 33.

Table 33 - Iterations plan with features selection

Features / stories		**Importance**
I2 - Second iteration - Enrollment		
F4.2	User should be able to apply the filter by root-searching the available groups according to their title and description. All groups should be observed by search.	2
F4.3	User should be able to see the details on any group he is enrolled to, including the list of other members. User should NOT be able to see the list of other members (except their number) for the groups he is not enrolled to.	4
F4.4	User should be able to join any existing group by sending the application to group owner.	5
F4.5	User should be able to leave any group he is enrolled to. Other group members should only be notified on that. Owner cannot leave the group and the group should be deleted manually (see F5.4).	1
F6.1	User should be able to see all members of the groups he is enrolled to on the map. If group member has disabled this privacy setting, it will be excluded from the view.	3
I3 - Third iteration – Questions management		
F2.2	User should be capable to add new question. New questions should be defined in separate windows which should include all important information about the question (title, text and images). The images should be taken by the phone camera.	5
F2.1	Current user should be able to check all his questions, including those that have been answered already. Questions should be presented by title and short description. Other	5

	details about every question should be presented in new window after user clicks on it.	
F2.7	User should be able to apply the filter by root-searching the list of questions available to him.	2
F2.3	User should be capable to delete own question. The deletion should not be performed without user's explicit confirmation on deletion action.	3
F2.4	User should be capable to change own question. The process of changing question should be similar to process of adding new question.	1
F2.5	User should be capable to add answers to own and others' questions.	5
F2.6	The owner of the question should be able to mark a question as answered.	5
I4 - Fourth iteration – News feed		
F1.1	When the application is started the news should be displayed. News should include any unread answers to the user's questions; news on activities in user's groups and other information important for current user.	3
F1.2	The news presented on the first application screen should be "links" to corresponding application functionality.	3
I5 - Fifth iteration – Settings and help		
F3.2	User should be able to set/change application settings. The settings should include the possibility to deny further invitations to groups, to set privacy level (of showing or no emails to other users and of showing or no current location to other users).	2
F7.1	User should be able to read a general help about the application usage.	1

All iterations included planning, working and release days. Thus, the working days were navigated through the series of predefined tasks, which described along with other documents can be found in the documents library. The summary of the performed tasks during the implementation is presented in the following table.

Table 34 - Performed tasks

Id	Task card	Type	Difficulty		Confi-dence	Date finished
			Before	After		
I2 - Second iteration - Enrollment						
TC-2-1	Create initial test cases	New	5	5	3	1.8.2012
TC-2-2	Implement supporting web services	Enhance	4	3	5	2.8.2012
TC-2-3	Implement group searching and viewing	New	5	5	4	3.8.2012
TC-2-4	Implement group enrolment and leaving	Enhance	3	3	4	6.8.2012
TC-2-5	Implement map view	New	3	4	4	7.8.2012
TC-2-6	Finalize tests	Enhance	5	5	3	8.8.2012
TC-2-7	Optimize and refactor	Enhance	2	2	5	8.8.2012
I3 - Third iteration – Questions management						
TC-3-1	Create initial test cases	New	5	5	4	17.8.2012
TC-3-2	Update database model	Enhance	1	1	5	20.8.2012
TC-3-3	Implement supporting web services	New	3	3	5	22.8.2012
TC-3-4	Develop questions management	New	5	5	5	27.8.2012
TC-3-5	Develop answers management	New	4	5	5	29.8.2012
TC-3-6	Finalize tests	Enhance	5	5	3	31.8.2012
TC-3-7	Optimize and refactor	Enhance	2	2	5	3.9.2012
I4 - Fourth iteration – News feed						
TC-4-1	Create initial test cases	New	5	5	4	11.9.2012

TC-4-2	Update database model	Enhance	1	1	5	11.9.2012
TC-4-3	Implement supporting web services	New	3	3	5	13.9.2012
TC-4-4	Implement mobile app functionality	New	5	5	5	17.9.2012
TC-4-5	Finalize tests	Enhance	5	5	3	19.9.2012
TC-4-6	Optimize and refactor	Enhance	2	2	5	20.9.2012
I5 - Fifth iteration – Settings and help						
TC-5-1	Create initial test cases	New	5	5	4	28.9.2012
TC-5-2	Update database model	Enhance	1	2	5	1.10.2012
TC-5-3	Update web services	Enhance	3	4	5	3.10.2012
TC-5-4	Implement settings management	New	3	3	5	5.10.2012
TC-5-5	Update groups management	Enhance	2	3	5	9.10.2012
TC-5-6	Update profile management	Enhance	2	3	5	11.10.2012
TC-5-7	Define help content	New	1	2	5	12.10.2012
TC-5-8	Develop help functionality	New	3	3	5	15.10.2012
TC-5-9	Finalize tests	Enhance	5	5	3	17.10.2012
TC-5-10	Optimize and refactor	Enhance	2	2	5	18.10.2012

3.4.2.2. Final database model

The final version of the database model, which has gone through tree additional iterations, is presented in the Figure 30. The presented model completely satisfies user requirements for the whole system, it is "open" and not tied to any technology, and is flexible to be updated or changed if necessary during the project lifecycle.

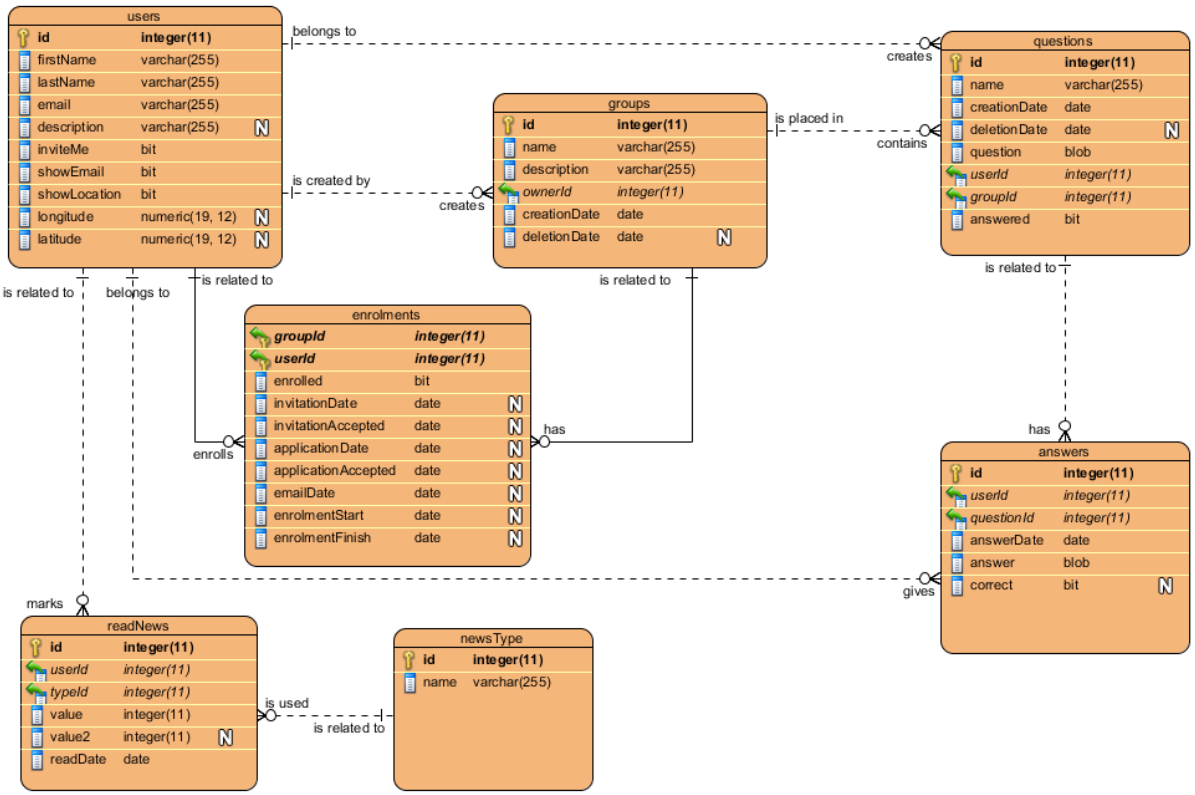


Figure 30 - Final database model

The model is created in the well-known *Crow's foot notation* (also known as James Martin's notation (Martin, 1986)). As it can be seen, three entities are considered to be weak entity types: *enrolments, readNews* and *answers*. These entity types are dependent on other strong entity types. Additionally, some relationships were made non-identifying by purpose of easier navigability and indexing, but also because of the idea of putting the read news into a specific entity in order to be excluded from the news feeds. Finally, special focus was put to relationships, role naming and cardinality in order to define those according to the best practices in data modeling.

3.4.2.3. Created web services

The final list of web services developed during the whole development process is shown in Table 35. The services developed in early development cycles were already described in detail. All other mentioned web services use the same Representational State Transfer (REST) communication protocol (Fielding, 2000), accept JSON formatted data and respond with JSON formatted response (Crockford, 2006). This approach was initially chosen as platform independent and is most likely to prove useful for other platforms as well.

Table 35 - Web services specification

Method	JSON formatted request	JSON formatted response
USERS (http://knowledge.uphero.com/users.php)		
create	firstName, lastName, email, [description]	responseId, responseText, [newUserId]
update	id, firstName, lastName, email, [description]	responseId, responseText
delete	id	responseId, responseText
position	id, longitude, latitude	responseId, responseText
settings	id, inviteMe, showEmail, showLocation	responseId, responseText
GROUPS (http://knowledge.uphero.com/groups.php)		
create	name, description, ownerId	responseId, responseText, [newGroupId]
update	id, name, description	responseId, responseText
delete	id	responseId, responseText
my	ownerId	responseId, responseText, [groups]
search	keyword	responseId, responseText, [groups]
ENROLMENTS (http://knowledge.uphero.com/enrolments.php)		
inviteUser	groupId, inviterId, email, [sendEmail]	responseId, responseText
enroll	groupId, userId, [action]	responseId, responseText
cancel	groupId, userId, [action]	responseId, responseText
members	groupId, userId	responseId, responseText, [users]
apply	groupId, userId	responseId, responseText
userLocations	userId	responseId, responseText, [users]
QUESTIONS (http://knowledge.uphero.com/questions.php)		
create	name, question, userId, groupId	responseId, responseText, [newQuestionId]
update	id, name, question, groupId	responseId, responseText
delete	id	responseId, responseText
searchByUser	userId	responseId, responseText, [questions]

searchByGroup	groupId	responseId, responseText, [questions]
searchByString	userId, keyword	responseId, responseText, [questions]
searchById	id	responseId, responseText, [questions (full)]
ANSWERS (http://knowledge.uphero.com/answers.php)		
create	answer, userId, questionId	responseId, responseText, [newAnswerId]
update	id, answer	responseId, responseText
searchByQuestion	questionId	responseId, responseText, [answers]
markAnswer	id	responseId, responseText
NEWS (http://knowledge.uphero.com/news.php)		
markRead	userId, typeId, value, [value2]	responseId, responseText
getByUser	userId	responseId, responseText, [news]

The usage of Service Oriented Architecture (SOA) in mobile application development got the acceleration during the last several years. This is a result of a wider Internet availability on mobile devices and of improved capabilities of mobile devices in terms of hardware. There are many projects that propose different SOA frameworks that could be used in development of mobile applications (Papageorgiou et al., 2009; Yee et al., 2009). Although our prototype application has Service Oriented Architecture, it is important to notice that the whole web part of this prototyping system is developed only for supporting purposes, and many concepts that should be implemented in commercial projects were not implemented here. Thus, the stated web services are stripped off of any session keeping, security checking, logging etc.

3.4.2.4. Class models

The alignment between planned and implemented system architecture can be observed through the final version of the class diagram. As it can be seen in Figure 31, it contains more than 25 classes, and it is unreasonable to present it in detail thus it is presented on the level of class names and relationships. The important conclusions that arise in this point are that during the development, the business logic layer which contains the *activity* and *service* classes become heavy but easy to maintain. The previously explained infrastructure was followed through all five iterations, and it is easy to notice that asynchronous calls to web services made the almost all activity classes to lean on *ServiceAsyncTask* and to receive the results through *AsyncTaskCallback* interface. The obtained results were later transformed into readable entity object through *JsonAdapter* object.

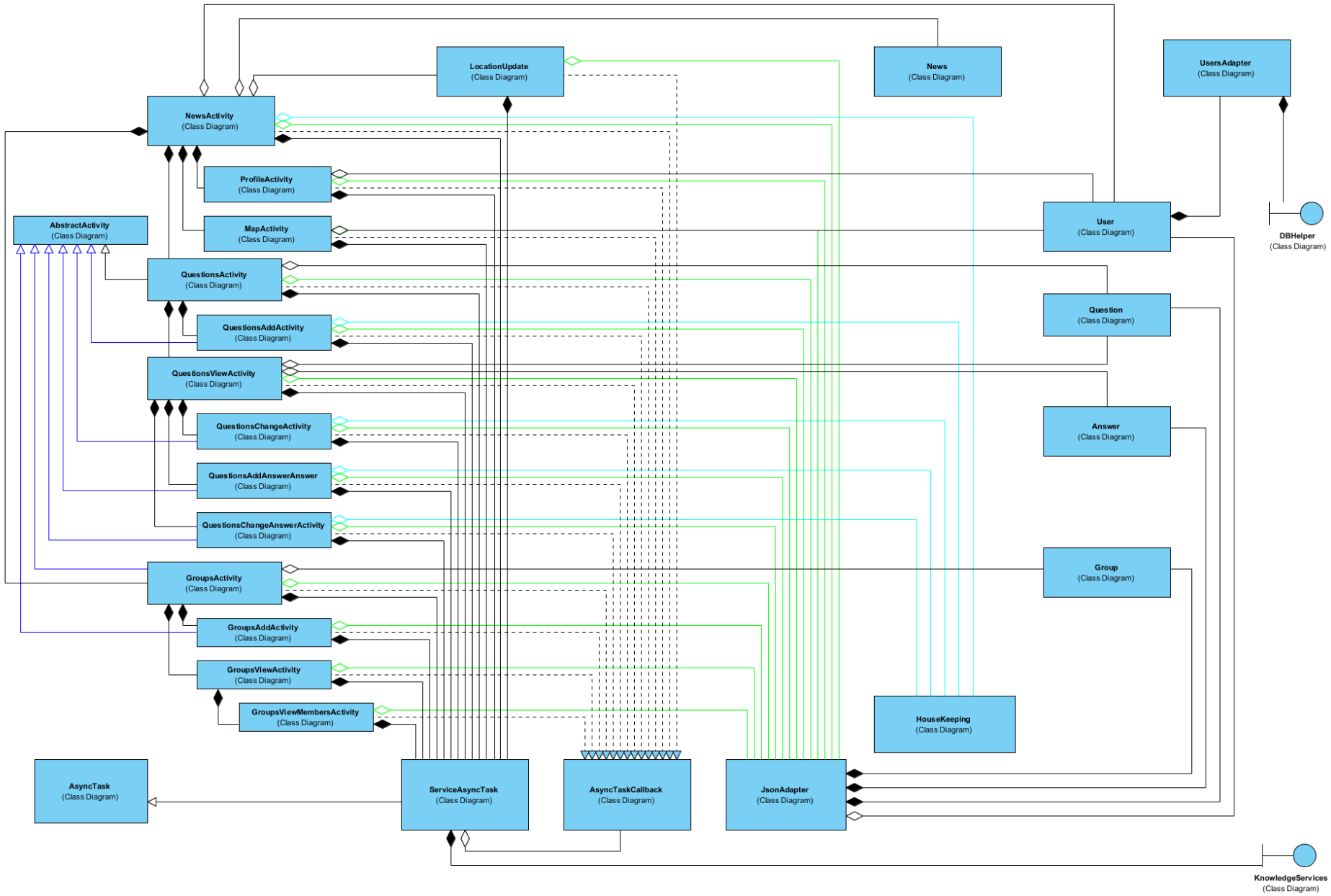


Figure 31 - Final class model (mobile application)

3.4.2.5. Application screenshots

The glimpse view of several use cases of final application functionality can be seen in the following figure containing the application screenshots. The presented functionality is fully tested, and all unit test as well as acceptance tests resulted in success.

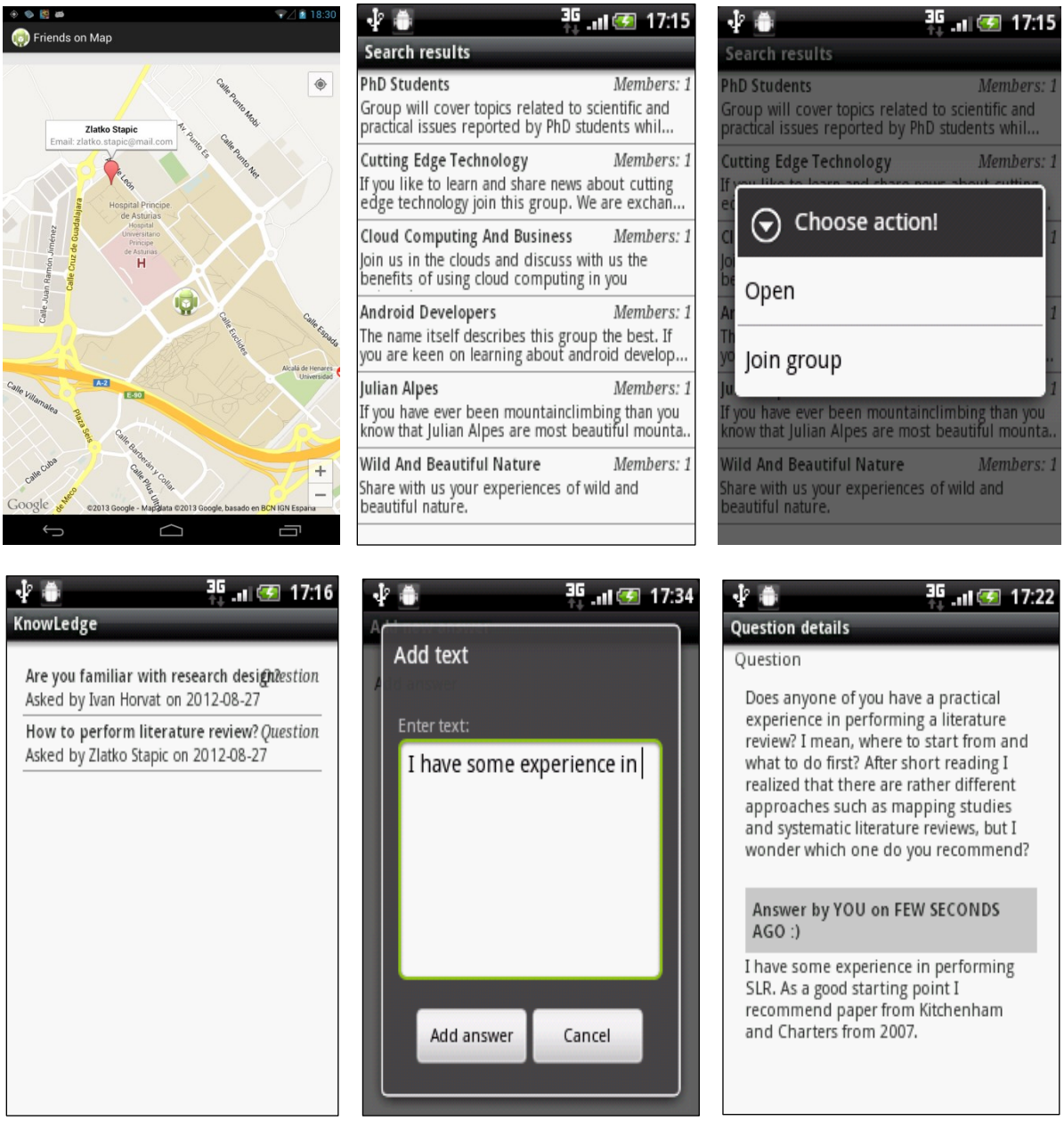


Figure 32 - Application screenshots

3.5. Stabilize

By definition, the purpose of this phase is to integrate smaller subsystems developed by different teams into a single product. Activities that were performed during this phase are exactly the same to the activities performed during the working days and thus artifacts the teams usually create are semantically same as artifacts we created in the earlier phases. As our mobile application was not divided into subsystems, there was no need to perform integration activities.

The additional task that characterizes this phase of mobile application development is called "Documentation wrap-up" task. Although the documentation was created during the whole development process, especially during the planning days of each phase and iteration, this task is specific as it produces the documentation for the project stakeholders and not for the agile team. Thus, the outputs of this task are finalized architectural, design and UI documents.

Following the rules given in (Abrahamsson et al., 2005a) we produced the mentioned documents that are salient, short and useful.

3.6. System test & fix

The important phase in the development of our project was System Test and Fix phase. As it can be seen in figure (Figure 33) taken from VTT's web application (2006a), the most important task is *System Test* task which comprises the activities of updating the test plan, executing the tests, logging the results and reporting the defects.

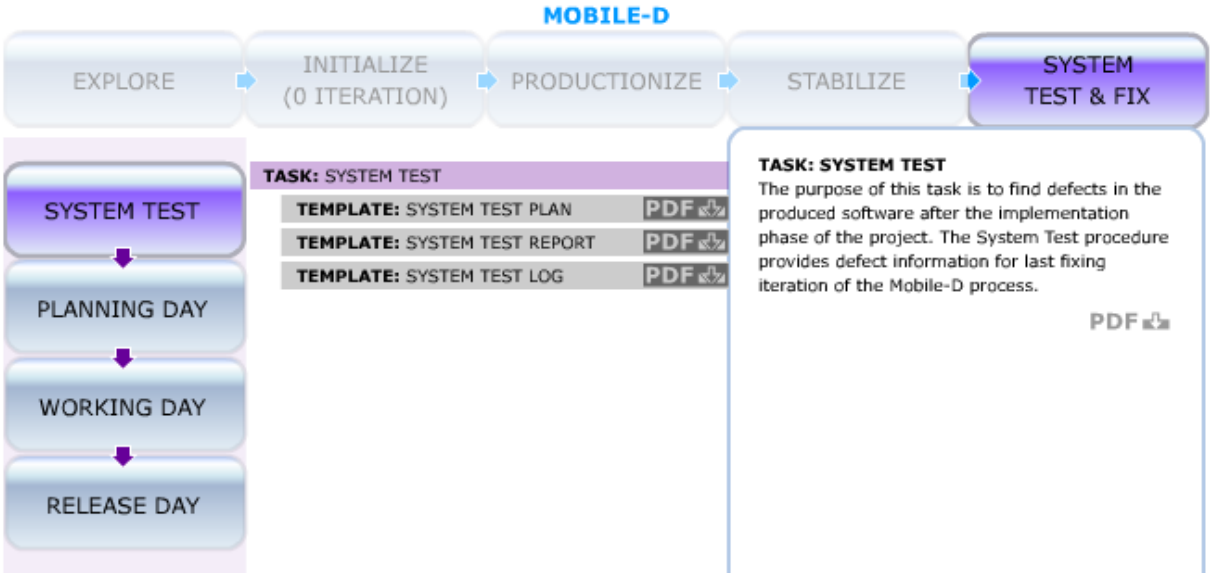


Figure 33 - System Test and Fix phase

As defined in Mobile-D methodology, this activity is performed only once (i.e. after the implementation phase of the project). The activities largely depend on the test results and sometimes no fixes are necessary. Some artifacts used in this phase were only updated as they had been already presented (UI tests, Acceptance tests, Integration Test, Unit tests) while others were newly created (final release, documentation of found defects).

As identified during the testing, and described in the minutes of the post iterations workshops, the following elements (see Table 36) of the mobile system functionality could be improved.

Table 36 - Recognized system limitations

Identified limitation of KnowLedge system	
1	The system does not treat email as unique. This might reflect on problems with sending the email invitation.
2	User cannot be invited or apply to join to a group repeatedly.
3	It is not possible to send email invitations to the users which are not already registered in KnowLedge system. This might slowdown the progression in getting new users.
4	Not all news should be canceled manually, as there are some news that should be automatically canceled (like notification of user leaving a group or similar).
5	Some data storage and data transfer optimization should be made. The existing content should not be downloaded repeatedly.
6	In some cases, the possibility of changing an existing answer could be useful. This should be carefully designed and planed with implementation of proper control.

The removal of these limitations would not have any influence on the identified set of artifacts but would significantly extend the development process. As these functionalities were not included in the user requirements, it was decided to leave them for some future versions of this system. Thus, the activities of fixing the application were not necessary.

Finally, we moved forward to publish the final version of the application on Google Play store. The process of publishing is straightforward and easy if all development activities are performed carefully and application manifest entries are correct. Google does not perform any manual application testing, and the only criteria that are to be satisfied concern the automated testing of application package. Having this in mind, we had to create an application icon in several formats, sign and publish the application by a wizard, and prepare the application screenshots and description. After uploading these documents to Google Play, our development process was officially finished. The application is available for download at http://barok.foi.hr/~zstapic/knowledge/android.

3.7. Development of Windows Phone application

The development of KnowLedge application for Windows Phone (WP) target platform was conducted after the development targeting Android platform. We used same Mobile-D methodology and same Test Driven Development approach.

Expectedly, from the methodological perspective, the development process was much easier as many artifacts developed earlier were completely or partially reused in this process. This possibility of reusing the artifacts was of our specific interest, as the overall goal of this research was to discover the similarities and to semantically describe them. While some artifacts remained the same, the other could be reused only as templates and the last group was formed from the artifacts that had to be built from scratch.

On the other hand, the development process was unexpectedly time-consuming. Although we were completely familiar with the desired application functionality, and although we reused some code templates, still the development for a new platform was a very challenging task which brought many obstacles. WP technology is very different from Android technology, and as can be seen from the description that follows, some aspects of the implementation approach (for example, in user interface, in communication with web service, in internal application structure) had to be reconsidered from scratch.

Additionally, although some artifacts were built from scratch their structure is very similar to the structure of the artifacts we have already presented. Thus we find no reason to report the whole process in detail again. Having this in mind, the following chapters discuss the performed development phases, but from the point of view focusing on the similarities and differences. Only completely new artifacts will be presented here.

3.7.1. Explore phase

The activities of stakeholder establishment, the scope definition and project establishment were almost completely omitted in the development process for the second target platform. In this phase, we didn't have to redefine the target users, stakeholders or initial requirements and architecture line description as these remained the same as for the Android target platform. The only activities that we had to perform included the definition of technological domain, redefinition of technology related risks and needed skills.

Regarding the technology, we decided to define a requirement of the application being runnable on any device running Windows Phone 7.5 (API level 7.1) or newer. The reasons for choosing this API level are guided by the principle of targeting as many devices as possible. As we do not need any capabilities of newer APIs, targeting 7.1 was a reasonable choice.

In a similar manner, the software architecture, project plan, documentation, and monitoring measures remained the same as for Android. The planned duration was not changed by purpose of making comparisons at the end of both development processes.

3.7.2. Initialize phase

The initialize phase took the same activities that we performed in the first development process. The existing virtual machine along with the set of tools not related to the development was reused, but the development environment for WP had to be established from scratch. We installed Microsoft Visual Studio, WP7.1 SDK, WP Toolkit, Microsoft Zune and connectivity software for our test devices. Finally, the testing of the development environment was performed by creating test project and deploying it to the testing device.

On the other hand, the activities that were supposed to produce updated project plan, architecture line plan and product backlog were unnecessary. All these artifacts including the system architectural diagrams, definition of features and the first version of acceptance tests remained the same and were reused. Thus again, we ended up with a product backlog containing the description of 22 features to be implemented in this development process.

The only document that we had to build again was the document containing the user interface sketches. The comparison of UI elements that are used in Android with those that could be used in WP showed that the relationships are not always direct. The in-detail analysis of the problem of automatic UI transformation was not in the focus of this research, but we found this software engineering challenge very interesting and thus tried to identify the elements that should be used in WP in order to give the user WP native look and feel along with the same functionality. In Figure 34 we can see that, for example, list (in the background of the Android sketch) can be translated to the same concept of list in the WP. But, the custom dialog does not have a WP implementation and we can either use another screen, or make changes in design of the existing form in a way that filter option will be a part of the main screen.

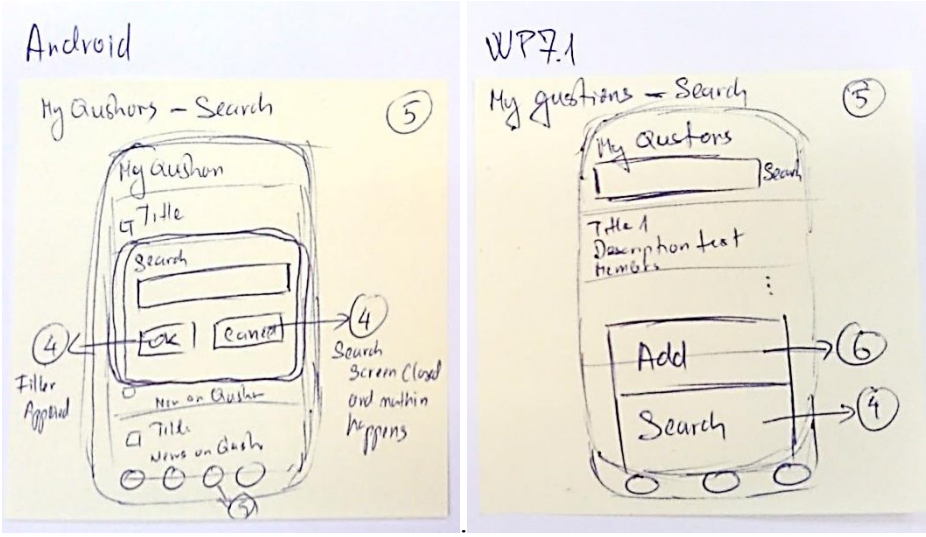


Figure 34 - Translating user interface from Android to WP

In the same sense we had to find different solutions to translate some other concepts like Android's toast message and progress dialogs.

The purpose of a trial day in this 0-iteration remained the same. The plans of features that ought to be implemented in order to trial and establish the internal application infrastructure remained the same. We also reused the data model completely and the story card and task cards as partially reused artifacts. Even without the need to design and develop the supporting backend system, the implementation of WP functionality took more time than planned and much more time than for Android. There are many reasons for this, mostly concerning platform restrictions and a narrowed set of usable features when compared to Android. Additionally, the recommended practice in development of WP applications is to use MVVM[19] pattern which requests a significant increase in development efforts. The use of this pattern helps in making a strong distinguishing line between the application layers in a multi-layered architecture.

Finally, another problem in WP development is the application of TDD approach. Although there are several 3rd party unit testing frameworks available for use, we found them to be out of date or without any maintenance and support – abandoned. The official Microsoft testing framework for Windows Phone was released very recently (as a part of Visual Studio 2012 Update 2) and targets the testing of Windows Phone 8 mobile applications. Thus, we had to use a limited functionality of Microsoft test framework that targets testing of .Net

[19] MVVM stands for Model View ViewModel architectural pattern from Microsoft. This pattern is largely based on MVC pattern, but with the focus on event-driven programming of UI development platforms.

applications. This limited the testing functionality only on Core classes and not on the user interface classes.

Test run completed Results (Group By: Class Name): 19/19 passed;

Result	Test Name	Error Message
JsonAdapterTest		
Passed	TestSingleton	
Passed	TestGetGroups	
Passed	TestGetJsonArrayString	
KnowledgeServiceAsyncTest		
Passed	TestAsymcTask	
KnowledgeServiceTest		
Passed	TestGroupCreate	
Passed	TestUserCreate	
Passed	TestGroupGetMy	
Passed	TestGroupDelete	
Passed	TestEnrolmentsInviteUser	
Passed	TestUserUpdate	
Passed	TestSingleton	
UserDataContextTest		
Passed	TestManyInstances	
Passed	TestUpdateUser	
Passed	TestInsertUser	
Passed	TestConstructor	
Passed	TestDeleteUser	
Passed	TestCurrentUser	
UserItemTest		
Passed	TestGettersAndSetters	
Passed	TestConstructor	

Figure 35 - Automated WP unit testing

The automated integration testing of WP was and still is impossible. There is no framework that might provide the features of automatic or robotized testing of Windows Phone applications, especially not for testing on devices. The only possible solution was to use a software that is capable of recording mouse and keyboard events. As this solution did not provide any possibility of making assertions we had to reject it and perform manual integration testing at the end of iteration.

3.7.3. Productionize

The approach and issues that we faced during the four *Productionize* iterations were very similar to the approach and issues we faced during the 0-iteration. We reused many artifacts which were related to project plan, iteration plans, product backlog, acceptance tests and other documentation. We also partially reused artifacts which were connected to activities of noting the current tasks such as story and task cards.

There was not need to make any changes to existing web service and remote database, which can bring us to conclude that the development process of these parts of the systems was thorough and with good quality.

While developing the WP application, we found the Android classes that were used to define entities very useful and we simply converted them to model classes in the new architecture. Additionally, some classes that were classified as libraries and were used to manipulate with JSON strings or to do housekeeping were also reused and easily translated to .Net. The process of localizing the mobile application reused all keys and values, but the original XML document had to be manually translated into a .Net resource file. We kept almost all the keys, and used exactly the same translations in both applications. Finally, the logic used to prepare the web service requests and to analyze the results was also reused and simply translated to the new programming language.

On the other hand, the existing code related to user interface manipulation, as well as the code related to web service asynchronous call and response had to be completely rejected. The .Net architecture made it easier to implement this functionality by using the events and delegates.

3.7.4. Stabilize

As the exhausting testing was performed during the development which initially included the integration with existing web services, at the end of the iterations the stabilize activities turned out to relate only to finishing of the documentation by performing wrap-ups. The final (but manual) integration testing was performed in this phase and as the results were positive we were capable of finishing the architectural, design and UI documents and move forward in the next iteration.

3.7.5. System test & fix

After having all iterations performed, the system test & fix activities were on schedule. Similar to the Android case, unit, integration and acceptance tests were positive. As the initial requirements were the same, the list of functionality that could be improved was also the same. As the removal of these limitations would not have any influence on the identified set of artifacts, we again decided to leave it for some future version of this system.

The process of publishing the finalized application on the Windows Market resulted in some new artifacts. We were obliged to use Marketplace Test Kit tool, to package application into a .XAP document and to provide the Market with icons and screenshots in different format than those for Android. After the testing process, the application will be available for download at http://barok.foi.hr/~zstapic/knowledge/wp.

3.8. Conclusions on implementation

By observing the whole development process again we can conclude that the implemented activities are well aligned with the planned activities. The following table (Table 37) displays the planned and realized activities and only differences from the Android case are in the duration of some activities while the overall project duration was shortened for 14 working days, but all activities had to be performed.

Table 37 - Duration of planned and real activities

Stage/Phase/Activity	Duration in days		
	Planned	Android	WP
KnowLedge	101	87	71
Explore	5	4	1
Stake holder establishment	2	1	0
Scope definition	2	2	0,5
Project establishment	1	1	0,5
Initialize	9	7	5
Project set-up	3	2	1
Planning day 0	3	2	0
Working day 0	3	3	4
Productionize	73	69	62
Iteration 1 – Group management	8	9	9
Planning day	2	2	1
Working day	5	6	7
Release day	1	1	1
Iteration 2 – Enrolment	8	9	10
Planning day	2	2	1
Working day	5	6	8
Release day	1	1	1
Iteration 3 – Question management	22	19	22
Planning day	5	4	2
Working day	15	13	17
Release day	2	2	3
Iteration 4 – News feed	22	12	11
Planning day	5	3	2
Working day	15	8	8
Release day	2	1	1
Iteration 5 – Settings and help	13	20	10
Planning day	2	3	1
Working day	10	16	8
Release day	1	1	1
Stabilize	12	4	2
Planning day	1	0	0
Working day	5	0	0
Documentation wrap-up	5	4	2
Release day	1	0	0
System Test & Fix	2+	1	1
System test	2	1	1

The duration of the development process in WP case is shorter for 30 working days if compared to the planned duration and is shorter for 16 working days (18.4%) if compared to

the Android development case. Such improvements in performance could be the result of the fact that we had already been familiar with the system requirements, that the backend system had already been developed and that different artifacts were partially or fully reused. On the other hand, we stated that the development time was not significantly reduced as we experienced many development issues and that the improvements could be result of our approach. As this is not important for the rest of the research we did not performed detailed analysis.

As serious testing had been done through all the iterations, the final tests were successfully executed in both development cases and there was no need for any changes in the system during the *System Test and Fix* phase. In any case, the overall development process was conducted in such manner that all activities and artifacts defined by Mobile-D methodology were performed and created.

Mobile application *KnowLedge* was designed to, by its purpose, cover the main and most common functional development requirements, and as such, it is a representative of the vast majority of mobile applications. Such requirements in general cover distinct development concerns, including *UI features*, *local database*, *device API-s*, *connection to web services* and *3rd party features.*

As Mobile-D methodology is well defined, it was not hard to follow the development process through all Mobile-D phases. Still, as the developed project was rather small and developed solely by the researcher with some minor help from his supervisors, small and acceptable divergence and misalignment with the Mobile-D was necessary. Still, we think that the performed process faithfully demonstrates the development process that would be performed by any small company developing a mobile application.

While developing Windows Phone application, the whole process was performed again. As the structure of the created artifacts along with the development process was the same as the one presented for the Android case, we found no reasons to report it again in detail. Thus, we reported the development process from the point of view in which we discussed the possibilities of reusing the existing artifacts. We found that many artifacts concerning the planning activities were reusable. Some of them concerning the product backlog, source code, resources and inner application logic were partially reusable, and of course, some had to be created from scratch. We also found that the backend part of our system requested no changes and although this lowered the overall workload the total development time was not shortened as we experienced some WP platform specific issues and some testing issues.

All empirical evidence created during the implementation was used in the next phases of this research process in order to identify their semantics, relationships and similarity between the two target platforms.

3.9. Relevance of the chapter

This section reported the development of mobile application *KnowLedge* by implementing Mobile-D methodology and Test Driven Development. First we gave a short overview of the methodology and approach and we defined the point of view in which the created artifacts took the most important role. Then, in the Android case, the performed phases were reported in detail along with the created outputs and their connections. The Mobile-D process with its clear technical specification was well documented and easy to follow and the overall development process took less time than initially planned.

In the case of Windows Phone application development, the whole process was performed again, but as the structure of the created artifacts was the same as the one presented in the Android case, we found no reason to report it again in detail. Thus, we reported the development process from the point of view in which we discussed the possibilities of reusing the existing artifacts. We found that many of the artifacts were completely or partially reusable.

We think that the performed process faithfully demonstrates the development process that would be performed by any small company developing mobile applications. The empirical evidence collected during this development was used in the subsequent research process of identifying the methodological interoperability and semantically similar artifacts.

4. IDENTIFICATION OF THE ARTIFACTS

In this chapter we will look back on the implementation results but from the *artifact identification* point of view. All artifacts that arose in the development sub-processes are enumerated and systematized in order to prepare the inputs for the next phase of the semantic description.

In order to perform a straightforward and unbiased analysis, first we defined the setting which includes the definition of artifacts, the relations with other methodological concepts that will be observed and the template that is to be used for the artifact description. As the artifacts were observed as "*any piece of software developed and used during software development and maintenance*" we found the list of Mobile-D artifacts related to the process tasks not sufficient and thus we performed our own analysis.

Thus, we observed the development process for each target platform separately and identified more than 70 artifacts that we initially grouped in 12 categories. After performing the cross-platform analysis we found that more than 70% of all identified artifacts were in common to both platforms and 66% percent of them are partially or completely reusable.

4.1. Analysis setting

In Chapter 3.1.3 we defined the conceptual model and gave a definition of artifacts that arise in the development process which utilizes some development methodology. In our case, Mobile-D methodology was chosen. For this research we adopted the Conradi's (2004) definition of the artifacts as "*any piece of software (i.e. models/descriptions) developed and used during software development and maintenance. Examples are requirements specifications, architecture and design models, source and executable code (programs), configuration directives, test data, test scripts, process models, project plans, documentation etc.*"

The conceptual model given in the mentioned chapter introduces the position of the artifacts in the overall development process. As the goal of this research was to analyze only the structural and semantic aspects of these sets of artifacts, we performed an analysis only from the semantic concept view, while other possible views, such as procedural concept view or

pragmatic concept view are not covered by it. Thus, we only observed the artifacts and their connection to the activities and tasks. The semantic of this connection was reduced to the concept of affiliation (e.g. which artifact is produced and used in which activity or task).

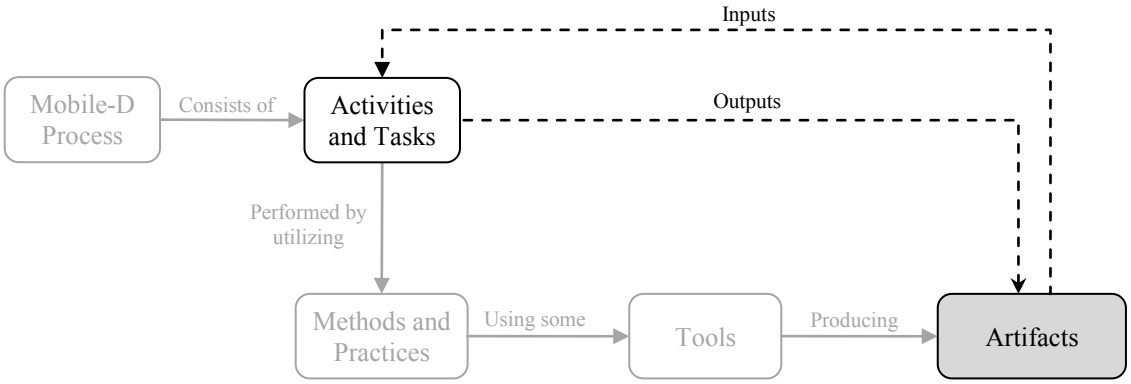


Figure 36 - Focusing semantic of artifacts and their origin

In this setting, the semantic concept view which describes the facts and the knowledge about the observed world was used. Additionally, by applying a procedural concept view, the analysis could be enhanced with procedural knowledge such as states, intentions, plans and rules and by applying a pragmatic concept view it could be additionally described by intentions, obligations or pragmatics of action. As we aimed to describe the concepts on artifacts in order to enhance the reusability while developing for second and other target platforms, the last two concept views are out of the scope of this research.

Mobile-D methodology, as described in chapter 3.1, comprises development process of five phases which are executed in combined sequential and incremental manner. Table 23 given in Chapter 3.1.3 presents inputs and outputs that were used in these phases. The list was created according to the Mobile-D process library and it includes documents and other deliverables, but also presents them at a very high level of abstraction and as completely platform-independent. After summarizing the information given in the Mobile-D process library (Abrahamsson et al., 2005a) and after correcting logical errors found in the existing overview, the mentioned artifacts were read (R), updated (U) or created (C) in tasks as presented in Table 38.

On the other hand, our analysis included only those documents that were used in the development of our prototype projects and introduced specific platform dependent deliverables. In this sense, our analysis, for example, provides a more specific description than the output “implemented functionality” states or specifies exact standards that were used rather than just specifying “relevant standards” as artifacts.

Table 38 - Mobile-D artifacts by tasks

	I					II			III																IV	V
PHASE: I - Explore II - Initialize III - Productionize IV - Stabilize V - System test & Fix	Customer establishment	Initial requirements collection	Initial project planning	Architecture line definition	Process establishment	Customer communication establishment	Architecture line planning	Initial requirements analysis	Acceptance test review	Acceptance test generation	Iteration planning	Post-iteration workshop	Requirements analysis	Continuous integration	Inform customer	Pair programming	Refactoring	Test-driven development	Wrap-ups	Acceptance testing	Pre-release testing	Release Ceremonies	System Integration	Documentation wrap-up	System test
Product proposal	R	R		R	R																				
Organizational process library					R																				
Contract	R	R	R																						
Initial requirements document		C	R	R			R	R												U					
Project plan	U		C	R	R U		R				R														
Standards		R																							
Base process description					C	R																			
Training plan					C																				
Measurement plan					C																				
Architecture line description				C			R																		
Architecture line plan							U	U																	
Software architecture and design							C																		
Product backlog								C			R		U												
Developer notes								C					C												
UI-illustrations/description								C					C												R
Acceptance tests/documentation								C	R U	C	R		U							R	R				R
Implemented functionality														R		C	R C	C			R		R C	R C	R C
Metrics data											R	R													
Experience												R C													
Story and task cards										R	C				R	R		R	R						
Action point list												C													

(Table 38 continued)

PHASE: I - Explore II - Initialize III - Productionize IV - Stabilize V - System test & Fix	I					II			III															IV	V
	Customer establishment	Initial requirements collection	Initial project planning	Architecture line definition	Process establishment	Customer communication establishment	Architecture line planning	Initial requirements analysis	Acceptance test review	Acceptance test generation	Iteration planning	Post-iteration workshop	Requiremements analysis	Continuous integration	Inform customer	Pair programming	Refactoring	Test-driven development	Wrap-ups	Acceptance testing	Pre-release testing	Release Ceremonies	System Integration	Documentation wrap-up	System test
Development artifacts											C											R		R	
Knowledge									U																
Data																			R						
Manuals, API specs and other																R									
Unit tests																	R	C							
Daily status report															C										
Defect list																				U	U				C
Release audit check list																						C			
The finalized documentation																								C	
System test report																									C
Test log																									C

- Task input
- Task output

R - Read C - Create
U - Update

Source: Based on information from (Abrahamsson et al., 2005a)

Additionally, this agile methodology uses main concepts of planning, working and release day through several phases. The activities and tasks, and thus the artifacts as well, are very similar regardless of the phase they are created or used in. This means that the approach of identifying and grouping the artifacts only according to the phases of the origin would not be a good way. Thus, while identifying the artifacts, we initially collected the data that included name, type/category, description and usage of the artifacts as presented in the following template (Table 39).

Table 39 - Template for describing the identified artifacts

Artifact name	Type	Description	I		II		III		IV		V	
			Input	Output	Input	Output	Input	Output	Input	Output	Input	Output

4.2. Artifacts targeting Android platform

After establishing the point of view we had decided to take in this research phase, we will move forward to identify and summarize the artifacts that emerged in the Android development process of our prototype mobile application. Although this has already been stated, it should be highlighted again that the development process itself was pretty much straightforward in following the Mobile-D methodology (see chapter 3.8) with only a slight misalignment in the organizational point of view – the project was not developed in an organization but by the researcher himself. Although this might have some negative and arguable influences, we assumed that the possibility of taking notes and observing the development process from the "inside" offers more advantages. We strived to follow all practices as they have been defined by the professional community and/or Mobile-D methodology, and we also developed a final and publishable product – the same as a company would do.

Thus, from the conceptual point of view, we created a solid basis for identifying not only the documents that had been created, but also other artifacts that might be hard to identify if the project was performed outside the laboratory.

The table presented below shows the list of identified artifacts, along with their initial classification, description and connection with the Mobile-D phases. We used standard CRU notation for denoting the artifacts that were created (C), used/read (R) and updated (U).

Table 40 - Identified artifacts in development process for Android

Artifact name	Type	Description	Phases inputs and outputs									
			I		II		III		IV		V	
			Input	Output	Input	Output	Input	Output	Input	Output	Input	Output
Mobile-D process library	Document	Process library describing the Mobile-D methodology in detail. Used as methodology guidelines in every phase. (Abrahamsson et al., 2005a)	R		R		R		R		R	
Product proposal	Document	Generated before the development process. Describes the initial and general idea on the product.	R									
Initial requirements document	Document	Created according to product proposal, but later updated with information on stakeholders and functional system requirements. It is also updated during the planning phase in 0-iteration and subsequent iterations.		C	R	U	R	U	R		R	
Project plan	Document	Contains all information on project including definition of customer group, scope, planned activities and their duration, plans on documentation etc. Aligned with agile practices, this document is also updated during the iterations.		C	R	U	R	U				
Project plan checklist	Document artifact	Mobile-D project plan checklist. This document is part of project plan.		C		U		U		U		U
Project plan checklist template	Template	Mobile-D project plan checklist (Abrahamsson et al., 2005a)	R									
Project plan Gantt chart	Model	Model containing the graphical information on project plan iterations, activities and their duration. It is used in Project plan document.		C		U		U				
Measurement plan	Document artifact	Includes the metrics and plan for monitoring of the project. In our case we recorded only the duration of activities and compared them with plan. This document is part of project plan.		C	R	U	R	U	R	U	R	U
Architecture line description	Document artifact	Created during the architecture line definition task and updated in architecture line planning activity. Contains the information on system context, technological scope, architectural risks etc. This document is part of project plan.		C	R	U	R					
Software architecture and design description document (SADD)	Document	Contains the technical documentation on the developed product.				C	R	U	R	U		
Architecture line plan	Document artifact	Contains the information on planned system architecture. Created after the prototyping is finished. This document is part of SADD document.				C						
UI-illustrations	Document artifact	Describes the illustrations of mobile application user interface. It is part of SADD document.				C	R	U	R		R	

Artifact name	Type	Description	Phases inputs and outputs									
			I		II		III		IV		V	
			Input	Output	Input	Output	Input	Output	Input	Output	Input	Output
Data model (mobile)	Model	Entity-Relationship-Attribute model of the mobile database. It is presented in SADD document.				C			R			
Data model (web)	Model	Entity-Relationship-Attribute model of the web application. It is presented in SADD document.				C	R	U	R			
Web service specification	Document artifact	Contains information on exposed web services along with available methods, their parameters and other communication elements. Part of SADD document.				C	R	U	R			
Class model (mobile)	Model	UML class diagram describing the mobile application internal structure and created classes. This model is used in SADD document.				C	R	U	R			
Class model (web)	Model	UML class diagram describing the web application internal structure and created classes. This model is used in SADD document.				C	R	U	R			
Class	Model element	UML model element used to describe a new class that is to be implemented.				C	R	U	R			
Android class	Model element	UML model element used to describe an existing Android class that is to be used.			R		R		R			
System Test plan	Document	Contains the information on purpose, plan and definitions of system test.		C	R	U	R	U	R		R	
Acceptance test	Document artifact	Created during initial requirements analysis. Contains the information on acceptance test of one product feature. Can include different contexts, and test scenarios with sample data. The document is part of System Test Plan document.				C	R	U	R		R	U
Acceptance test template sheet	Template	Mobile-D acceptance test template sheet (Abrahamsson et al., 2005a)			R							
Prototype functionality	Code	Developed functionality during the trial day. It prototypes some of the main application functionalities and is used to define the basic approach for implementing the similar functionalities in other iterations.				C	R					
Product backlog	Document	Contains the information on features that are (to be) implemented in the development process, through several iterations. Users can contribute in defining the features/stories.				C	R	U	R	U		
Story card	Document artifact	Basic documentation card containing information on one feature that is implemented. It is defined during the planning day but is refined during the implementation and wrap-up. It is part of the Product backlog document.				C	R	U	R	U		
Story card template	Template	Mobile-D story card template (Abrahamsson et al., 2005a)			R							

Artifact name	Type	Description	Phases inputs and outputs									
			I		II		III		IV		V	
			Input	Output	Input	Output	Input	Output	Input	Output	Input	Output
Task card	Document artifact	Basic documentation card containing the information on one task that is to be performed during the iteration. it is defined during the planning day and refined during implementation and wrap-up. It is part of the Product backlog document.				C	R	U	R	U		
Task card template	Template	Mobile-D task card template (Abrahamsson et al., 2005a)			R							
Iterations plan	Document artifact	Contains the information about planned iterations along with selected features for specific iteration. This document is part of Product backlog document.					R	C	R			
Iteration backlog	Document artifact	Contains the information on specific iteration including story and task cards. Each iteration document is created from scratch. It is part of Product backlog document.			C			C		U		
System test report	Document	Final document on testing. Contains information on performed tests and issues detected.										C
Test results	Document artifact	Results are obtained during the whole development process testing tasks. At the end this document becomes part of System test report.				C	R	U	R	U	R	U
Defect list	Document artifact	Document created after testing is performed. It contains found issues and planned activities. At the end this document becomes part of System test report document.						C	R	U	R	U
Unit test	Code	Unit test tests a single unit of code. It is created in separate project and references main project while performing different assertions.				C	R	U	R		R	
Integration test	Code	Robotized test which tests application integrated functionality.				C	R	U	R		R	
API documentation	Example	Android API documentation from developers.android.com			R		R		R			
Example code	Example	Android example code on different topics found on the internet from various sources.			R		R		R			
Development unrelated software tools	Software	These software tools support the main operations performed by project team. For example these include office suit, PDF reader, image editor etc.		C								
Project management software tool	Software	The tool used for project management.		C								
Drivers	Software	Set of drivers used to install the device connectivity for testing purposes.				C						
Development environment	Software	Set of applications used for Android development. We used Eclipse base SDK.				C						
Throw-away prototype	Code	Project created to test development environment and connected devices. This project is discarded.				C						

Artifact name	Type	Description	Phases inputs and outputs									
			I		II		III		IV		V	
			Input	Output	Input	Output	Input	Output	Input	Output	Input	Output
Web application development environment	Software	The web application development and hosting environment had to be set up.				C						
Mobile application	Product	The mobile application created in the development process.				C		U				U
Web service	Product	The web part of the system created in the development process.				C		U				
Java code	Code	Java code developed during the implementation activities.				C	R	U	R		R	
PHP code	Code	PHP code developed during the implementation activities.				C	R	U				
XML resource	Code	XML code describing application layout, menus, localized strings etc.				C	R	U				
Application manifest	Code	XML document containing the information on application. This document is most important code artifact.				C		U			R	
Google Play Services	Code	Google library containing the classes necessary if using Google Maps.					R					
Activity	Code	Represents java class that inherits Android Activity class with the purpose of controlling the application view.				C	R	U	R			
Layout	Code	Represents XML code that is used to describe user interface form or screen.				C	R	U				
Layout element	Code	Represents XML code that is used to describe any user interface element such as text box, list box, button etc.				C	R	U				
Localization strings	Code	Represent XML code that is used to provide localized translation of values according to value unique key.				C	R	U			R	
Google API Key	License	Google license identifying the developer as unique person. This key is application specific and is used when using Google Maps API.					R	C				
IEEE Standard No. RFC4627 (JSON)	Standard	Standard defining the JSON format. (Crockford, 2006)			R		R					
Application screenshot	Resource	Application screenshots are created as needed for publishing process.				C		U				U
Application icon	Resource	Application icon is designed as needed for publishing process.										C
Application description	Resource	Short but important description used for publishing process. It includes the information on application, category, authors etc.										C
Deployment package	Resource	APK file created for publishing purposes.										C

C – Created, R – Read/used, U - Updated

The identification process resulted in total of 60 different artifacts that are grouped in 12 groups according to their type. From our point of view, which is based on conceptual analysis of semantic interoperability among different target platforms, we identified the following types related to Android development:

Table 41 - Types of artifacts related to Android development

Artifact type	Description
Document	Represents used documents or created artifacts that are published as documents during or at the end of development process.
Document artifact	Represents document that could be observed as stand-alone artifact, but is usually included in some other document.
Template	Represents templates that are used to create some artifacts.
Model	Represents models that are created during the development process. Models could be observed as stand-alone artifacts, but are usually presented as a part of some document.
Model element	Represents the atomic level (i.e. integral) artifact that could be observed as stand-alone and is used to create models.
Code	Represents any artifact created during the implementation and is written in any programming or description language.
Example	Represents code artifacts created by third party and used as examples of implemented functionality or to solve some programming issue.
Software	Represents software tools used during the entire project.
License	Represents individual-specific unique key that is obtained or used during the development process.
Standard	Represents document containing formal and internationally recognized description of some concept or element.
Publishing resource	Represents resources that are created during the development process and are used in publishing purposes.
Product	Represents final product as most important project deliverable.

Although some semantic links between the identified artifact types are obvious, the detailed semantic analysis, the definition of the relationships and the hierarchy among the artifacts and the identified types was performed in the next research phase and hence they were not focused on in this phase. In order to facilitate understanding, at this point it should be pointed out that some documents contain parts (document artifact) that should be observed separately which is why we identified them as a specific (new) type. Similarly, the model element could be observed as a stand-alone artifact used to build more complex models.

4.3. Artifacts targeting Windows Phone platform

As has been reported in Chapter 3.7, the development of mobile application targeting Windows Phone (WP) platform aimed to analyze if the existing artifacts from the Android case can be reused. This resulted in the fact that several activities in the Explore phase were completely omitted and some other activities were simplified due to the artifacts partial reuse. But, although all used artifacts were not created in the windows phone development process,

we nevertheless consider them as artifacts that belong to this process and subsequently they were included in the following table.

The cross-platform comparison and analysis of the artifacts similarity was performed later and is not in focus of this chapter. We bring here the list of the identified artifacts that were used in the Windows Phone development case. Again, we used the standard CRU notation for denoting the artifacts that were created (C), used/read (R) and updated (U).

Table 42 - Identified artifacts in Windows Phone case

Artifact name	Type	Description	Phases inputs and outputs									
			I		II		III		IV		V	
			Input	Output	Input	Output	Input	Output	Input	Output	Input	Output
Mobile-D process library	Document	Process library describing the Mobile-D methodology in detail. Used as methodology guidelines in every phase. (Abrahamsson et al., 2005a)	R		R		R		R		R	
Product proposal	Document	Generated before the development process. Describes the initial and general idea on the product.	R									
Initial requirements document	Document	Created according to product proposal, but later updated with information on stakeholders and functional system requirements. It is also updated during the planning phase in 0-iteration and subsequent iterations.		C	R	U	R	U	R		R	
Project plan	Document	Contains all information on project including definition of customer group, scope, planned activities and their duration, plans on documentation etc. Aligned with agile practices, this document is also updated during the iterations.		C	R	U	R	U				
Project plan checklist	Document artifact	Mobile-D project plan checklist. This document is part of project plan.		C		U		U		U		U
Project plan checklist template	Template	Mobile-D project plan checklist (Abrahamsson et al., 2005a)	R									
Project plan Gantt chart	Model	Model containing the graphical information on project plan iterations, activities and their duration. It is used in Project plan document.		C		U		U				
Measurement plan	Document artifact	Includes the metrics and plan for monitoring of the project. In our case we recorded only the duration of activities and compared them with plan. This document is part of project plan.		C	R	U	R	U	R	U	R	U
Architecture line description	Document artifact	Created during the architecture line definition task and updated in architecture line planning activity. Contains the information on system context, technological scope, architectural risks etc. This document is part of project plan.		C	R	U	R					

Artifact name	Type	Description	Phases inputs and outputs									
			I		II		III		IV		V	
			Input	Output	Input	Output	Input	Output	Input	Output	Input	Output
Software architecture and design description document (SADD)	Document	Contains the technical documentation on the developed product.				C	R	U	R	U		
Architecture line plan	Document artifact	Contains the information on planned system architecture. Created after the prototyping is finished. This document is part of SADD document.				C						
UI-illustrations	Document artifact	Describes the illustrations of mobile application user interface. It is part of SADD document.				C	R	U	R		R	
Data model (mobile)	Model	Entity-Relationship-Attribute model of the mobile database. It is presented in SADD document.				C			R			
Data model (web)	Model	Entity-Relationship-Attribute model of the web application. It is presented in SADD document.				C	R	U	R			
Web service specification	Document artifact	Contains information on exposed web services along with available methods, their parameters and other communication elements. Part of SADD document.				C	R	U	R			
Class model (mobile)	Model	UML class diagram describing the mobile application internal structure and created classes. This model is used in SADD document.				C	R	U	R			
Class model (web)	Model	UML class diagram describing the web application internal structure and created classes. This model is used in SADD document.				C	R	U	R			
Class	Model element	UML model element used to describe a new class that is to be implemented.				C	R	U	R			
.Net class	Model element	UML model element used to describe an existing .Net class that is to be used.			R		R		R			
System test plan	Document	Contains the information on purpose, plan and definitions of tests.		C	R	U	R	U	R		R	
Acceptance test	Document artifact	Created during initial requirements analysis. Contains the information on acceptance test of one product feature. Can include different contexts, and test scenarios with sample data. The document is part of System Test Plan document.				C	R	U	R		R	U
Acceptance test template sheet	Template	Mobile-D acceptance test template sheet (Abrahamsson et al., 2005a)			R							
Prototype functionality	Code	Developed functionality during the trial day. It prototypes some of the main application functionalities and is used to define the basic approach for implementing the similar functionalities in other iterations.				C	R					

Artifact name	Type	Description	Phases inputs and outputs									
			I		II		III		IV		V	
			Input	Output	Input	Output	Input	Output	Input	Output	Input	Output
Product backlog	Document	Contains the information on features that are (to be) implemented in the development process, through several iterations. Users can contribute in defining the features/stories.				C	R	U	R	U		
Story card	Document artifact	Basic documentation card containing information on one feature that is implemented. It is defined during the planning day but is refined during the implementation and wrap-up. It is part of the Product backlog document.				C	R	U	R	U		
Story card template	Template	Mobile-D story card template (Abrahamsson et al., 2005a)			R							
Task card	Document artifact	Basic documentation card containing the information on one task that is to be performed during the iteration. it is defined during the planning day and refined during implementation and wrap-up. It is part of the Product backlog document.				C	R	U	R	U		
Task card template	Template	Mobile-D task card template (Abrahamsson et al., 2005a)			R							
Iterations plan	Document artifact	Contains the information about planned iterations along with selected features for specific iteration. This document is part of Product backlog document.					R	C	R			
Iteration backlog	Document artifact	Contains the information on specific iteration including story and task cards. Each iteration document is created from scratch. It is part of Product backlog document.			C			C		U		
System test report	Document	Final document on testing. Contains information on performed tests and issues detected.										C
Test results	Document artifact	Results are obtained during the whole development process testing tasks. At the end this document becomes part of System test report.				C	R	U	R	U	R	U
Defect list	Document artifact	Document created after testing is performed. It contains found issues and planned activities. At the end this document becomes part of System test report document.						C	R	U	R	U
Unit test	Code	Unit test tests a single unit of code. It is created in separate project and references main project while performing different assertions.				C	R	U	R		R	
Integration test	Document artifact	Represents the description and results of integration test that is performed manually. This document is part of System Test Plan document.				C	R	U	R		R	
API documentation	Example	WP API documentation from http://msdn.microsoft.com			R		R		R			

Artifact name	Type	Description	Phases inputs and outputs									
			I		II		III		IV		V	
			Input	Output	Input	Output	Input	Output	Input	Output	Input	Output
Example code	Example	WP example code on different topics found on the internet from various sources.			R		R		R			
Development unrelated software tools	Software	These software tools support the main operations performed by project team. For example these include office suit, PDF reader, image editor etc.		C								
Project management software tool	Software	The tool used for project management.		C								
Drivers	Software	Set of drivers used to install the device connectivity for testing purposes.				C						
Development environment	Software	Set of applications used for Windows Phone development and integrated in Visual Studio.				C						
Throw-away prototype	Code	Project created to test development environment and connected devices. This project is discarded.				C						
Web application development environment	Software	The web application development and hosting environment had to be set up.				C						
Mobile application	Product	The mobile application created in the development process.				C		U				U
Web service	Product	The web part of the system created in the development process.				C		U				
C# code	Code	C# code developed during the implementation activities.				C	R	U	R		R	
PHP code	Code	PHP code developed during the implementation activities.				C	R	U				
XAML description	Code	XML based XAML code describing application layout and layout elements.				C	R	U				
WMAppManifest	Code	XML document containing the information on application. It includes the information on some application resources. It is created automatically.				C					R	
Microsoft Phone Controls Toolkit	Code	Library containing the classes necessary for adding some basic and advanced controls.			R		R					
Silverlight Map Control	Code	Library containing the classes necessary for using Bing maps in WP application.					R					
Page (C#)	Code	Represents C# class that has the purpose of controlling the application view.				C	R	U	R			
Page (XAML)	Code	Represents XAML code that is used to describe user interface form or screen.				C	R	U				
Page element	Code	Represents XAML code that is used to describe any user interface element such as text box, list box, button etc.				C	R	U				
Resource file	Code	Represents code that is used to provide the application with resources (strings, images, icons, audio, files and other). We used it to provide the application with localized translation for two languages.				C	R	U			R	

Artifact name	Type	Description	Phases inputs and outputs									
			I		II		III		IV		V	
			Input	Output	Input	Output	Input	Output	Input	Output	Input	Output
Bing maps key	License	Microsoft license identifying the developer as unique person. This key is application specific and is used when using Silverlight Map Control.					R	C				
IEEE Standard No. RFC4627 (JSON)	Standard	Standard defining the JSON format. (Crockford, 2006)			R		R					
Application screenshot	Resource	Application screenshots are created as needed for publishing process.				C		U				U
Application icons	Resource	Application icons are designed as needed for publishing process.										C
Application description	Resource	Short but important description used for publishing process. It includes the information on application, category, authors etc.										C
Deployment package	Resource	XAP file created for publishing purposes.										C

C – Created, R – Read/used, U - Updated

The total of 61 artifacts were identified and described. All artifacts are classified according to the same classification of 12 different artifact types recognized in the first development case. In the following chapter, a cross-platform analysis will be performed in order to identify common, specific, and partially reusable artifacts in both development processes.

4.4. Cross-platform artifacts comparison

The undertaken activities of identifying and describing the artifacts that were used in the two development cases resulted in a list of 60 artifacts in the Android case and 61 artifacts in the Windows Phone case. The initial classification of these artifacts resulted in 12 different types. The purpose of this chapter is not to perform a detailed semantic analysis of the artifacts relations, but rather to do a cross-platform comparison in order to separate those that are common to both platforms from those that are specific to one or the other and those that are partially reusable.

We strongly believe that the order of execution of the development cases did not have any influence on the identified set of artifacts. We also believe that the artifacts that were reusable in our presented scenario would also be reusable if we developed for Windows Phone first. However, having only this development case, we cannot make strong conclusions, but the evidence collected in this scenario indicates on this characteristic. This could be another positive aspect of the approach taken in this dissertation.

4.4.1. Common artifacts

In the cross-platform analysis we found that 50 artifacts (70.42% of all identified artifacts) are common to both development cases. Thus, we named them *common artifacts.* These artifacts are enumerated in Table 43.

Table 43 - Common artifacts in Android in WP case

Artifact name	Identical	Partially reused	Different
Mobile-D process library	X		
Product proposal	X		
Initial requirements document	X		
Project plan		X	
Project plan checklist		X	
Project plan checklist template	X		
Project plan Gantt chart	X		
Measurement plan		X	
Architecture line description			X
Software architecture and design description document			X
Architecture line plan		X	
UI illustrations			X
Data model (mobile)	X		
Data model (web)	X		
Web service specification	X		
Class model (mobile)			X
Class model (web)	X		
Class		X	
System test plan			X
Acceptance tests	X		
Acceptance test template sheet	X		
Prototype functionality			X
Product backlog		X	
Story card		X	
Story card template	X		
Task card		X	
Task card template	X		
Iterations plan	X		
Iterations backlog		X	
System test report			X
Test results			X
Defect list		X	
Unit test		X	
Integration test			X
API documentation			X
Example code			X
Development unrelated software tools	X		
Project management software tool	X		
Drivers			X
Development environment			X
Throw-away prototype			X
Web application development environment	X		
Mobile application			X
Web service	X		
PHP code	X		
IEEE standard No.RFC4627 (JSON)	X		
Application screenshot			X

Application icon		X	
Application description		X	
Deployment package			X
TOTAL (50)	**20**	**13**	**17**

Additionally, many of these common artifacts are platform independent as being products of methodological approach. In total, 20 out of 50 identified artifacts (40.00%) have been created or obtained only once, as these were identical in both development processes. In this group, it is important to distinguish between those artifacts that were only used as inputs while performing the methodology (like Mobile-D process library, various templates, standards, tools) and those that had to be created by a development team, but only once (like artifacts concerning some aspects of project planning activities, testing or backend system development activities). A proper reuse of these artifacts will give the development team the first fruits of taking the approach we are proposing in this dissertation.

On the other hand, there are 13 artifacts (26.00%) that could be partially reused while performing the development process for the second or any other target platform. There are various reuse levels that we recognized in this group (from reusing artifact creation approach, reusing content inner logic, to reusing some parts of content itself). We believe that a different additional analysis should be performed in this direction and that the results could give a more specific knowledge on reusable artifact elements, which, in the end, could result in more specific and easier to follow instructions and thus better results for development teams.

Finally, we recognized 17 artifacts (34.00% of all common artifacts) with a very low level of possible reuse. They were classified as ones that should be developed from scratch for every target platform.

The results presented in this chapter are very encouraging and we can conclude that they create a strong basis and motivation for additional research and analyses. In this dissertation, we have covered only one possible approach, but as has been stated before, other approaches are also welcome.

4.4.2. Platform dependent artifacts

The artifacts that are characteristic for one target platform and are significantly different from artifacts of other target platform are classified as *platform dependent artifacts.* As presented in Table 44 there are 10 Android specific artifacts and 11 Windows phone specific artifacts that were created in this particular development case.

Table 44 - Android and WP specific artifacts

Android specific artifacts
Android class
Java code
XML resource
Application manifest
Google Play Services
Activity
Layout
Layout element
Localization strings
Google API Key
TOTAL (10)
Windows Phone specific artifacts
.Net class
C# code
XAML description
WMAppManifest
Microsoft Pone Controls Toolkit
Silverlight Map Control
Page (C#)
Page (XAML)
Page element
Resource file
Bing maps key
TOTAL (11)

If we carefully observe and compare these platform specific artifacts, we can conclude that even in this case there are some semantic similarities. For example, *Java code* and *C# code* are separate artifacts but they might have reusable parts like sequencing, iterations, algorithms etc. Thus we did not reject them as irrelevant for the rest of the research, and have used them as well in the next phase of the semantic analysis.

4.5. Relevance of the chapter

To summarize, in this chapter we have identified all artifacts that arose in our development process for two target platforms: Android and Windows Phone. The artifacts are observed as "*any piece of software developed and used during software development and maintenance*" (Conradi, 2004), and thus we first created a list of artifacts that were specific for Mobile-D methodology and then enhanced it with the artifacts identified in our development cases. The total of 71 artifacts were recognized and initially classified in 12 different categories.

Our cross-platform analysis showed that 50 artifacts (70.42%) are common to both development cases. We found that 20 artifacts are exactly the same in both cases and another 13 artifacts are partially reusable. Thus, in total the 33 artifacts (66.00% of the common

artifacts) are completely or partially reusable. This brought us to the conclusion that these results provide a solid basis and motivation for the semantic analysis that follows.

With the identification and cross-platform analysis of the artifacts we have concluded the second phase of our research process. We now move to the third phase where we will semantically and ontologically describe these artifacts.

5. THE ONTOLOGY FOR METHODOLOGICAL INTEROPERABILITY

The main goal of this research is to ontologically describe artifacts that arise in the methodologically managed process of mobile application development targeting two or more mobile platforms, and to create the basis for more efficient and interoperable process of multi-platform mobile applications development.

In the previous chapters we analyzed the state of the art in the usage of methodologies for mobile applications development, and also performed a development process for two different target platforms by utilizing Mobile-D methodology, and based on the gathered empirical evidence we identified more than 70 different artifacts that arose in these two development cases.

In this chapter we will move on to our last research phase in order to semantically describe the identified artifacts, their meaning and relations and finally to create a formal ontology containing the knowledge on possibilities of artifacts reuse in multi-platform mobile application development.

The chapter is organized in four parts. First, we will introduce and define the concept of ontology, discuss possible usages, types, development methodologies and tools, in order to determine the type of our ontology along with the environment that will be used to develop and describe the ontology. Secondly, we will develop an ontology describing the development for Android platform and in this part we will focus on ontology development by utilizing an ontology development methodology. In the third part we will define the second ontology describing the development for Windows Phone target platform and in this part we will put focus on the concepts of ontology reuse and update.

Finally, in the fourth part we will present the development of the common ontological description for both platforms, and in this chapter we will focus on the concepts of ontology merging, extension, evaluation and testing.

5.1. Ontology

5.1.1. Definitions

The term *ontology* is a philosophical term that has its roots in Greek words "*on*" (genitive "ontos") - "*being*", and "*logia*" - "*writing about, study of*". It is often stated that Greek philosophers Parmenides, who argued about nothingness, and Aristotle, who argued about theory of being in his work Metaphysics, begot the concept of ontology in the 4th century BC. Since then, many other philosophers have used the concept and the term. In philosophy ontology is defined as "*a branch of metaphysics concerned with identifying, in the most general terms, the kinds of things that actually exist. Thus, the ontological commitments of a philosophical position include both its explicit assertions and its implicit presuppositions about the existence of entities, substances or beings of particular kinds*" (Kabilan, 2007). In other words, ontology is the theory of existence.

From our perspective, we are more interested in the concept of ontology that is currently used in some other disciplines including Artificial Intelligence, Knowledge Management, Information Systems and Software Engineering. Gruber (1993a) defined ontology as "*an explicit specification of conceptualization*". To put it another way and according to Gruber, ontology is a specification of a representational vocabulary for a shared domain of discourse and it includes definitions of classes, relations, functions and other objects. According to Gong et al. (2006), ontology is a general conceptualization of a specific domain in a format readable to humans and to machines. Same authors define *Process Description Ontology* as a formal semantics to traditional process modeling elements, such as entities, objects and activities, their relationships et cetera.

Following Gruber's definition, Studer et al. (1998) defined ontology as "*a formal, explicit specification of a shared conceptualization.*" This definition includes: the term *conceptualization* as an abstract modeling of some phenomenon and identification of its relevant concepts; the term *shared* representing that the knowledge included in the ontology should be consensual and shared; the term *formal* to exclude the use of natural languages and to make the ontology machine readable: and the term *explicit* denoting that the concepts and the constraints on their use should be explicitly defined.

On the other hand, based on their experience Noy and McGuinness (2001) took the pragmatic approach and defined the ontology as "a formal explicit description of concepts in a domain of discourse (classes (sometimes called concepts)), properties of each concept describing various features and attributes of the concept (slots (sometimes called roles or properties)), and restrictions on slots (facets (sometimes called role restrictions))".

According to Hilera et al. (2010) ontology is a knowledge representation tool, and the knowledge representation tools can be classified at four different levels. Dictionaries, taxonomies, thesauri and ontologies are respective representatives of these levels. The last one, the ontology level, includes definitions of concepts (dictionaries), implicit or explicit vocabulary, as well as descriptions of specialized relationships between concepts (taxonomies), lexical and equivalence relationships (thesaurus), and combination of relationships with other more complex relationships between concepts to completely represent a certain knowledge domain.

As we can see, the term "ontology" was taken from philosophy, but its use and meaning in Computer Science got a new and adapted perspective. As there is no consensus on the definition of ontology, in the context of this research we consider ontology as *an explicit formal conceptualization of a shared understanding of the domain of interest which includes vocabulary of terms for describing the domain elements, semantics in order to define the relationships of the domain elements and pragmatics in order to define possible usages of these elements.*

5.1.2. Uses of ontologies

The use of ontologies in the domain of Computer Science grew rapidly in the last two decades. Firstly, ontologies were used mainly as tools in the area of Artificial Intelligence, but now, their usage become popular in many other fields as they provided the domain experts the possibility of categorizing the domain knowledge.

Noy and McGuinness (2001) gave a comprehensive overview of possible reasons for the use of ontologies. They found following reasons which are here shortly explained and demonstrated on our example:

- *To share common understanding of the structure of information among people or software agents*. In our case, after having the ontology of artifacts that arose in the development process defined, we created a basis for development of an automated system or software agent that could provide teams with information on requested queries or event in order to guide them in the development process.
- *To enable reuse of domain knowledge*. This is one of the strongest reasons for ontology usage. For example, if we need a detailed description of the Android operating system in our ontology, we can simply reuse the existing ontology if one exists. Additionally, we might consider using an existing general ontology and extending it to the knowledge describing our domain.
- *To make domain assumptions explicit*. Explicit assumptions bring several advantages in terms of understanding, improving or correcting knowledge. Thus, the assumptions

created in our ontology of artifacts can be changed without the need to change the system that uses them, and will still be readable to people without any knowledge about the design of the system that is based on the ontology.

- *To separate domain knowledge from the operational knowledge*. This is another common use of ontologies. In our example, we could describe the artifacts and their relationships separately from describing the operational knowledge on using those artifacts. Thus, the system built on this operational knowledge could be easily fed with some other ontology of artifacts without the need to be changed.
- *To analyze domain knowledge*. The process of creating ontologies is possible only when the domain terms are declaratively specified. The ontological description thus enhances declarative description and makes the knowledge formal and reusable.

In the end, it is important to notice that ontology should not have a purpose in itself. The ontologies should be built with an existing idea of their application. The desired application always has an influence on the ontology structure and its final form. Thus, the ontological description of artifacts that arise in the methodologically driven development process would not be the same if we build it with the idea of using the application in teaching on methodological process and if we build it with the idea of using the application to advise and help on artifact reuse when developing for different platforms.

5.1.3. Ontologies and semantic interoperability

Interoperability is in nature multilateral and can be best understood as a shared value of the community. According to European Interoperability Framework for European Public Services (EIF) (European Commission, 2010) the interoperability within the context of European Public Services delivery can be defined as "*ability of disparate and diverse organizations to interact towards mutually beneficial and agreed common goals, involving the sharing of information and knowledge between the organizations, through the business processes they support, by means of the exchange of data between their respective ICT systems.*" Also, the EIF defines Interoperability framework as "*an agreed approach to interoperability for organizations that wish to work together towards the joint delivery of public services. Within its scope of applicability, it specifies a set of common elements such as vocabulary, concepts, principles, policies, guidelines, recommendations, standards, specifications and practices.*"

In the context of this research, the IEEE definition of interoperability will be adopted and extended. The original definition (IEEE Computer Society., 1990) says that interoperability is "the ability of two or more systems or components to use the information that has been exchanged". The definition of interoperability will be extended with the methodological and social component to "*the ability of two or more systems, components, teams or team members*

to use and exchange the information and methodological artifacts that have been created during the mobile application development process".

Observing from different points of view, we can talk about several types of interoperability. The most suitable division for this research is the one that defines two types of interoperability. Several authors are talking about *semantic* and *syntactic interoperability* (Park and Ram, 2004). So, according to Park and Ram *semantic interoperability* is the knowledge-level interoperability which provides the interoperable systems with a possibility to bridge the semantic conflicts, and *syntactic interoperability* is the application-level interoperability that allows interoperable systems to cooperate regardless of their implementation techniques (Park and Ram, 2004). This thesis will deal only with semantic interoperability.

Additionally, Park and Ram define three different areas of semantic interoperability. *Mapping-based approach* creates mappings between semantically related information sources, *intermediary-based approach* depends on the use of intermediary mechanisms to achieve interoperability, and *query-oriented approach* is based on interoperable languages (Park and Ram, 2004) (Gong et al., 2006). The mapping-based approach is not designed to be independent of particular schemas and applications; the query-oriented approach requires the users to understand all underlying local databases; so the most promising approach is the intermediary-based approach as it uses intermediary mechanisms such as mediators or ontologies, which may have domain-specific knowledge, mapping knowledge, or rules specifically developed for coordinating various and autonomous information sources (Park and Ram, 2004).

According to Paulheim and Probst (2010), interoperability can be performed on different levels, and subsequently they define integration on data source level, integration on the business logic level and integration on the user interface level.

Surprisingly, interoperability on the methodological level is rarely mentioned in literature. Thus, the goal of this research is to create an ontological definition that can be used as a knowledge source for information system guiding the development teams to increase the methodological interoperability by reusing the artifacts that are created in the development process of mobile application for the second and every other target platform.

5.1.4. Ontology types

There is no single point of view which could be taken when defining ontology types. According to Lovrenčić (2007) ontologies can be grouped in accordance with their forms, the volume and the type of conceptualization structure, the conceptualization subject and the richness of described content. The same author emphasizes that the most common

classification is according to the conceptualization subject. Upon adapting the classification from (Gómez-Pérez, 2004) she describes the following eight categories of ontology types (Lovrenčić, 2007):

- *Knowledge representation ontologies* aim to represent the domain knowledge by utilizing a knowledge representation paradigm. These ontologies are built from common modeling artifacts – classes, relationships and attributes. The most commonly used knowledge representation paradigms are Frame Ontology, Resource Description Framework (RDF), RDF Schema (RDFS), Ontology Interface Layer (OIL), DARPA Agent Markup Language + OIL (DAML+OIL) and Web Ontology Language (OWL).
- *General/Common Ontologies* describe the common knowledge that can be used in different domains. These ontologies define different general concepts like time, space, events and similar.
- *Top-level Ontologies* describe abstract concepts which are related to the specific concepts used in ontologies at lower abstraction level. These ontologies should be universal and expressive. Some of well-known upper-level ontologies are Cyc (aims to describe the whole human consensual knowledge) and SUMO (Suggested Upper Merged Ontology supported by IEEE).
- *Domain Ontologies* describe concepts belonging to one specific domain. The domain should be described at the highest possible abstraction level so the ontology could be reused while developing other ontologies in the same domain. Some of the domains could be Education, Law, Knowledge Management, Medicine, Engineering et cetera. As the number of domains grew, the need for structured ontology libraries resulted in several well-known libraries like Protégé Ontology Library, DAML Ontology Library and others.
- *Task Ontologies* describe the concepts that are related to a specific task or activity and needed to solve the problems related to that task.
- *Domain Task Ontologies* are similar to Task Ontologies, but are reusable in the same domain. We consider these ontologies as more general.
- *Method Ontologies* give the description of the concepts that are used in the specification of the process of decision making in order to solve a task.
- *Application Ontologies* define the concepts related to the knowledge in a specific application. These ontologies are dependent on their appliance and usually extend other domain and task ontologies related to the observed application.

As it can be seen from the listed ontology types, the main difference between the ontologies is in the level of abstraction of the described concepts. They form a continuum that covers concepts ranging from being very specific to being very general and abstract. The level of abstraction is directly connected to the possibility of ontology reusability as general

ontologies are highly reusable and those describing specific concepts are not (Lovrenčić, 2007).

Similar ontology classification created upon ontology generality was created by Guarino (1998). He defined four types of ontologies we already mentioned: *Top-level Ontology*, *Domain Ontology*, *Task and Problem Solving Ontology* and *Application Ontology*. These types are, according to Guarino, hierarchically ordered as it is shown in Figure 37.

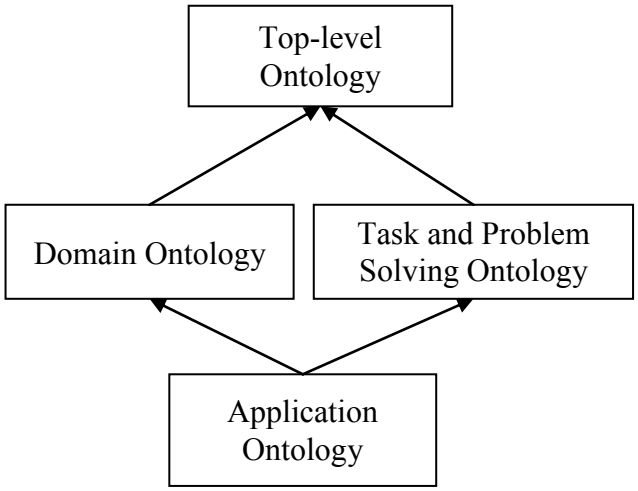


Figure 37 - Guarino's types of ontologies according to generality level

As domain ontology can be defined as a network of domain model concepts (topics, knowledge elements) that defines the elements and the semantic relationships between them (Brusilovsky et al., 2005), the use of domain ontologies is suitable to describe all content regarding development methodology and approach, and thus, the ontology that is a subject of this research is classified as domain ontology as well. In this way, the adaptive Web-based system, which we plan to develop on the base of the results of this research, will be able to select and recommend the most relevant reusable content during the development of multi-platform mobile application.

5.1.5. Ontology development methodologies

Gruber (1993b) defined five principles that became de facto standard in the ontology design not only in the Artificial Intelligence field but also in other fields where ontologies are used. These five principles include *clarity*, *coherence*, *extendibility*, *minimal encoding bias* and *minimal ontological commitment.* We will give a glance overview of these principles as they are the goals that should be achieved in every ontology development activity. According to (Gruber, 1993b) the principles can be described as:

- **Clarity**: Ontology should be able to transmit the encapsulated knowledge and the meaning to its users through objective and complete definitions. Documentation of definitions should be written in a natural language.
- **Coherence**: Ontology should be logically coherent at the level of axioms as well as informally coherent in concepts that are described for instance in a natural language or in examples. Subsequently, the inferred knowledge should be coherent to that described in the documentation.
- **Extendibility**: Ontology should be designed to anticipate the usage of a shared vocabulary in such a way that it should be possible to extend the ontology with new terms that are based on the existing vocabulary without the need of changing the existing definitions.
- **Minimal encoding bias**: The conceptualization should be specified at the knowledge level without depending upon any symbol or language encoding. This will enable the automatic transformation of ontology among different encoding styles and will enable the usage of ontology in knowledge-sharing agents implemented in different representation systems.
- **Minimal ontological commitment**: Ontology should make as few claims as possible about the world being modeled. This is done by defining only essential terms needed for communication of the knowledge. Subsequently, this will enable further specialization and instantiation of the ontology as needed.

Gruber concluded his criteria definition with discussion about the necessity of having some trade-offs among the stated criteria. Although the criteria are not diagonally opposite, some trade-offs are necessary. But, as we can see, Gruber did not give any guidelines on how to achieve these criteria in a methodological manner. He did not provide a cookbook that we can use while designing the ontology. Additionally, these criteria define only the requirements regarding the creation of ontology artifacts, but do not reflect upon the intended purpose of the ontology.

In addition to the stated, Kabilan (2007) defined specific design choices that are to be made while designing domain ontologies. She defined the following questions:

- Which concepts are relevant and necessary to be included in the proposed ontology?
- What is the optimum design architecture for the proposed ontology?
- What kind of design strategy is best suited for the given domain and given purposes?
- How to be consistent in the conceptualization of similar categories of concepts?
- How to match the functional requirements of the targeted application with the goals of ontology design? How do these functional requirements influence the ontology design choices?

- What is the minimum required level of knowledge formalization?
- Which knowledge representation formalism/language to choose?
- Once the above design decisions are taken, how should a designer actually proceed in capturing, analyzing and representing the implicit and explicit domain knowledge?
- What tools, methods, other knowledge sources, models may be chosen to help in the knowledge modeling process?

Providing answers to all of these questions is not a trivial task. It is obvious that a structured and guided approach is necessary. Thus, during these 20 years since the earlier mentioned design principles have been stated, a number of ontology development methodologies have been proposed.

There are several papers that give an extensive overview of ontology design methodologies, such as (Dahlem, 2011), (Lovrenčić, 2007) and (Kabilan, 2007). Dahlem compared sixteen ontology design methodologies and he concluded that three of them have their roots in the creation of Knowledge Based Systems (CommonKADS, Cyc and KBSI IDEF5), five of them aim at the construction of ontologies from scratch (Grűninger and Fox, Uschold and King, METHONTOLOGY, Ontology Development 101 and UPON), two of them emphasize the collaborative evolution of ontologies (DILIGENT and HCOME), three of them are focusing on reuse of existing knowledge (SENSUS, KACTUS and ONIONS) and the remaining three are inspired with database engineering (DOGMA), wiki-based systems (mOnt) and Knowledge Management (On-To-Knowledge). Although the list of compared methodologies is not an exhausting one and there are many other methodologies described in literature, in the case of our research, methodologies that aim at construction of ontologies from scratch (as it is later elaborated in Chapter 5.2.2) are from our specific interest, and they will be shortly described in the following paragraphs.

5.1.5.1. METHONTOLOGY

After identifying the lack of standardized procedures in the ontology development process, Fernandez-Lopez et al. (1997) defined an ontology development methodology – namely METHONTOLOGY – as the methodology that is based on software development process. Their method is based on the execution of the following phases which provide the activities for building an ontology from scratch:

1. Specification – The idea of this phase is to produce informal, semi-formal, or formal specification document written in natural language including information on the purpose of the ontology, users, scenarios of use, the level of formality of future ontology and the scope which includes a set of terms to be represented, its characteristics and granularity.

2. Knowledge acquisition – The activities of knowledge acquisition are independent activities in the ontology development process, but are performed simultaneously with specification and other phases.
3. Conceptualization – This phase should result in conceptually structured domain knowledge in terms of the domain vocabulary identified in the ontology specification phase. *Glossary of Terms* should be created in this phase and it should include concepts, instances, verbs and properties. The following activities include: grouping activity where concepts and terms are grouped according to their inner cohesion; the activities of concepts description, verbs description and tables of formulas and rules creation.
4. Integration – As a result of this activity, METHONTOLOGY proposes the development of an integration document, summarizing the meta-ontology that will be used along with detailed links between terms that are to be used and the terms defined in conceptual model.
5. Implementation – This phase should result in the ontology codified in a formal language. The activities of this phase should be supported by ontology development environment which should at least provide: a lexical and syntactic analyzer, translators, an editor, a browser, a searcher, evaluators and so on.
6. Evaluation – In the METHONTOLOGY, evaluation assumes the terms of verification which refer to technical process that guarantee the correctness of the ontology and validation which checks if the ontology corresponds to the system that they supposed to represent.
7. Documentation – This support activity should be done through the whole ontology development process. After mentioned phases, the documentation activities include the creation of a requirements specification document, a knowledge acquisition document, a conceptual model document, a formalization document, an integration document, an implementation document and an evaluation document.

The mentioned activities can be divided into two main groups: the technical activities and the support activities. Technical activities include *specification*, *conceptualization* and *implementation*, while the remaining are support activities.

Although the presented methodology slightly evolved during the time, its basic approach remained the same.

5.1.5.2. Ontology Development 101

Another well-known and often used methodology for ontology development is Ontology Development 101 (Noy and McGuinness, 2001). This methodology describes iterative approach in ontology development, and is created as one possible approach that can be used.

The approach gained popularity mainly because of its simplicity, clarity and focus on the results.

Basic assumptions built into the Ontology Development 101 (OD101) methodology are: there is no single correct way to model a domain and the best solution always depends on the application and the expected extensions of the ontology; ontology development is necessarily an iterative process; the concepts in the ontology should be close to objects (nouns) and relationships (verbs) in the domain of interest (in the sentences that describe the domain). The whole methodology is comprised in execution of 7 steps as described in (Noy and McGuinness, 2001):

- Step 1. Determine the domain and the scope of the ontology. In order to define a domain and the scope of the ontology, OD101 proposes the list of basic questions that should be answered. The list includes questions like: *What is the domain that the ontology will cover? For what are we going to use the ontology? Who will use and maintain the ontology?* The answers to these questions aim at limiting the scope of the model. Additionally, the OD101 authors suggest the creation of a list of competency questions that a knowledge base, based on the ontology, should be able to answer. In our case, the competency questions list could contain questions like: *What artifacts do I need in this development step? What are the outputs of this step? Is the class diagram presented in the test plan document or software design and description document? What artifacts can I reuse in this phase?*
- Step 2. Consider reusing existing ontologies. There are different libraries containing already developed ontologies that can be reusable in our particular case. Additionally, if our system needs to interact with other applications that have already committed to particular ontologies or vocabularies, it is necessary to reuse and build upon these ontologies and vocabularies.
- Step 3. Enumerate important terms in the ontology. The list of terms that arise in our domain of interest should be created. This list will be updated in all iterations and while building it we can think of: what terms we would like to talk about, what properties do those terms have and what would we like to say about those terms? For example, some terms that could be interesting to our ontology are: *artifact*, *phase*, *activity*, *task*, *input*, *output* et cetera.
- Step 4. Define the classes and the class hierarchy. This step and step 5 are closely connected and are always performed in parallel by defining a few definitions of the concepts in the hierarchy and then continue by describing properties on those concepts. These two steps are also two most important steps in the ontology design process.

  There are three basic approaches that can be taken while developing a class hierarchy:

- A *top-down* development process starts with the definition of the most general concepts in the observed domain and continues with subsequent specialization of the concepts.
- A *bottom-up* development process starts with the definition of the most specific classes and then groups them into more general concepts.
- A *combination* development process combines a top-down and bottom-up approach. The idea of this approach is to define more salient concepts first and then to make generalization or specialization as needed. The Uschold and Gruninger (1996) (who define this approach as "middle-out approach") argue that top-down and bottom-up approaches have a number of negative effects (like over-detailed ontologies, high efforts needed, less stability) and they find a middle-out approach as a balanced approach that they used successfully in practice.

  In any case, the terms are in this step converted into classes which are then organized into a hierarchy. A class should become a subclass if all instances of that class are also instances of its super class.
- Step 5. Define the properties of classes – slots. In this step the internal structure of the concepts is created. As the classes from the list of terms created in Step 2 are already selected, most of the remaining terms are properties of these classes. In general, there are several types of properties that could be created: *intrinsic properties*, *extrinsic properties*, *structure properties*, and *relationships*. The mentioned properties should be attached to the most general class that can have that property.
- Step 6. Define the facets of the properties. Each defined property should be described in detail by defining some additional restrictions like the type of its value, cardinality, domain (classes that property describes) and range (allowed classes of instances),
- Step 7. Create instances. This is the last step in an ontology creation process. It results in a list of individual instances of classes in the hierarchy.

By the characteristics of the presented methodology (simplicity, focus on results and iterative approach) we can call this methodology an *agile ontology development methodology*, and that is why we find this methodology as the most suitable for our research process and we will use it in defining our ontology.

5.1.5.3. UPON

Unified Process for ONtology building (UPON) is an ontology building methodology based on the Unified Process (UP). The methodology is proposed by De Nicola et al. (2005) who tried to show that the basic phases in developing a software system could be the same when building an ontology. They also propose the reuse of UML modeling language to model some aspects of ontologies as they find them use-case driven, iterative and incremental.

Similar to UP, UPON also defines *cycles*, *phases*, *iterations* and *workflows*. Each cycle consists of *inception*, *elaboration*, *construction* and *transition* and results in the release of a new version of the ontology. Each phase is further subdivided into iterations where five workflows take place: *requirements*, *analysis*, *design*, *implementation* and *test* (see Figure 38).

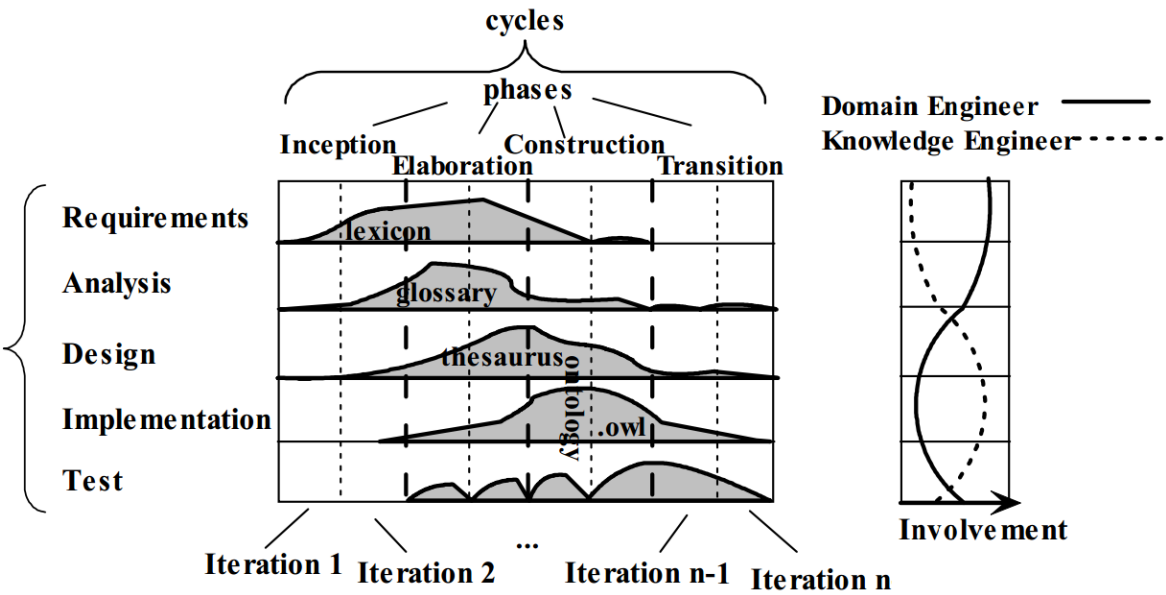


Figure 38 - De Nicola's UPON framework (De Nicola et al., 2005)

5.1.5.4. Uschold and King

Back in 1995, Uschold and King defined a skeleton for a methodology for building ontologies. The skeleton consisted of four main phases which are defined as follows (Uschold and King, 1995):

- Identify Purpose
- Building the Ontology
- Ontology capture
- Ontology coding
- Integrating Existing Ontologies
- Evaluation
- Documentation

If compared to other methodologies created later, we can conclude that this simple methodology created the basis for its successors. By describing other mentioned methodologies we already described all concepts that were focused by Uschold and King as well.

5.1.5.5. Grűninger and Fox

The ontology development methodology presented by Grűninger and Fox (1995) is based on the activities that transform *Informal Competency Questions* through specification of *Terminology in First-Order Logic*, to *Formal Competency Questions* and finally to specification of *Axioms in First-Order Logic*. The procedure is finished after the *Completeness of Theorems* is checked. This methodology defines formal approach in ontologies development and provides a framework for evaluating the adequacy of created ontologies by proving the completeness of theorems for the ontologies with respect to the formal competency questions.

Similarly to Uschold and King's methodology that highly influenced the methodologies for development of *semi-formal*[20] ontologies, this methodology highly influenced the development of other methodologies for development of *formal* (also known as *rigidly formal*) ontologies.

5.1.6. Ontology development tools and languages

Prior to moving forward in our research process we have to state what ontology representation language and what ontology development tool we will use. The ontology representation language and tools are usually related to ontology design methodology. Starting from Ontolingua which is proposed by Gruber (1993a), there are many such languages like LOOM, OCML or OWL. These languages vary in the degrees of formality and expressive power (Corcho et al., 2003). OWL – Web Ontology Language[21] created by W3C Web Ontology Working Group, became the most widely used language and is supported by most generic tools, such as editors or reasoning systems (Lumsden et al., 2011). Current version of OWL is OWL2[22].

In the same manner, many ontology development tools exist. Among many analyses and comparisons of these tools we point out the analysis performed by Youn and McLeod (2006) who compared fourteen ontology development tools by seven criteria. Although many of these tools evolved a lot during the last years, it might be important to notice the authors' conclusion that all of them have their advantages and disadvantages. The authors did not propose any tool as the best solution.

---

[20] Uschold and Gruninger (1996) classified ontologies upon their formality and complexity and they defined four major categories as follows: *highly informal* are ontologies expressed in natural language; *semi-informal* are ontologies expressed in structured form of natural language; *semi-formal* are expressed in artificially formally defined language; and *rigidly formal* are those ontologies that have terms defined with semantics, theorems and proofs.

[21] http://www.w3.org/2004/OWL/

[22] http://www.w3.org/TR/owl2-overview/

On the other hand, Khondoker and Mueller (2010) analyzed the usage of ontology editors and found that SWOOP, TopBraid Composer, OntoTrack, Internet Business Logic, Protégé and IHMC Cmap Ontology Editor are the only tools used by participants they questioned. Their results show that 75% of all participants used Protégé and 41.95% of participants created the ontologies in the domain of Information System Design. This gives us a solid basis to accept the Protégé[23] as the most commonly used tool and the one to use in our ontology development.

As Protégé natively works with Frames and OWL (and from version 4 it also supports OWL2), we had to decide whether to use Frames or OWL as our ontology representation language. According to Wang et al. (2006) the main difference between them is that Frames is used when *close-world assumptions* (CWA) are suitable and OWL otherwise. The concept of CWA represents the semantics with the presumptions that what is *not* currently known to be *true* is *false*. On the other hand, capabilities and expressiveness of OWL are needed to deliver the functional requirements, when we need Description Logic (DL) reasoning to ensure logical consistency of ontologies, when we aim to create robust terminologies or when classification is a paradigm for reasoning in applications. Although it is possible to use both languages in our case, we find the use of OWL representation language more appropriate.

OWL is a language for defining and instantiating ontologies by defining descriptions of classes, properties and their instances. It provides three increasingly expressive sublanguages (W3C Web Ontology Working Group, 2004). *OWL Lite* supports classification hierarchy and simple constrains features, *OWL DL* supports maximum expressiveness without losing computational completeness and decidability of reasoning system, while *OWL Full* is meant for users who want maximum expressiveness and freedom but with no computational guarantee and with no full reasoning support. Although OWL DL has to include constructs with some restrictions, in our ontology we need full reasoning support, and thus we will use OWL DL representation language.

Besides defining the abstract structure of the ontology, OWL provides the ways in defining their meaning in terms of formal semantic description which specifies how to derive the logical consequences out of the ontology, i.e. facts not literally presented in the ontology but entailed by the semantics. In OWL2 we can use two alternative ways of assigning meaning to the ontologies: Direct Semantics[24] and RDF-Based Semantics[25]. According to (W3C OWL Working Group, 2012), "*OWL2 DL is used informally to refer to ontologies interpreted using*

[23] Protégé is a free, open-source, plugin-based platform that provides suite of tools to construct domain models and knowledge-based applications with ontologies. It can be obtained for free from http://protege.stanford.edu/
[24] http://www.w3.org/TR/owl2-direct-semantics/
[25] http://www.w3.org/TR/owl2-rdf-based-semantics/

the Direct Semantics and OWL2 Full is used informally to refer to RDF graphs considered as ontologies and interpreted using the RDF-Based Semantics". This means that we are more interested in capabilities of Direct Semantics reasoning which assigns meaning directly to ontology structures, resulting in semantics compatible with the model theoretic semantics of the *SROIQ*[26] description logic. This also brings the necessity of placing some restrictions[27] on ontology structures in order to ensure that they can be translated into SROIQ knowledge base.

Finally, concrete syntax is needed in order to store OWL2 ontologies and to exchange them among tools and applications. The primary exchange syntax for OWL2 is RDF/XML[28] but other concrete syntaxes may also be used. These include alternative RDF serializations, such as Turtle[29]; an XML serialization[30]; and a more readable syntax, called the Manchester Syntax[31]. As Protégé supports all mentioned syntaxes along with automatic translation among them, we can later decide which of these alternatives to use while exporting our ontology into a human readable format.

5.1.7. Final remarks on ontologies

Ontologies gained a huge popularity during the last two decades and are currently used in different scientific fields. As they provide means of explicit and formal specification of knowledge and conceptualization, which is readable to humans and to machines, we also found it appropriate to use the ontologies as a tool in defining our framework for methodological interoperability in multi-platform mobile applications development.

In previous chapters, we tried to give a short overview of a several concepts that are related to ontologies and ontology development. First, for the purpose of this research we defined ontology as an explicit formal conceptualization of a shared understanding of the domain of interest which includes the vocabulary of terms in order to describe the domain elements, semantics in order to define the relationships of the domain elements and pragmatics in order to define possible usages of these elements.

---

[26] SROIQ represents fragment of first order logic with useful computational properties. An overview of DL languages can be seen in (Belcar and Lovrenčić, 2012). Belcar and Lovrenčić defined SROIQ languages as follows: S – AL and C with transitive properties; AL – base attributive language that allows atomic negation, concept intersection, universal restriction and limited existential quantification; C – complex concept negation; R – limited complex role inclusion axioms, reflexivity and irreflexivity, role disjointness; O – nominals (enumerated classes or object value restrictions); I – inverse properties; Q – qualified cardinality (number) restrictions.

[27] The details on restrictions are given in Section 3 of OWL 2 Structural Specification document which can be obtained at http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/#Ontologies

[28] http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/

[29] http://www.w3.org/TR/turtle/

[30] http://www.w3.org/TR/2012/REC-owl2-xml-serialization-20121211/

[31] http://www.w3.org/TR/2012/NOTE-owl2-manchester-syntax-20121211/

We also presented the most common reasons for the use of an ontology and we argued about their classification in accordance with different points of view. In this context we concluded that in this research we will create domain ontology in order to semantically describe concepts belonging to one specific domain – development of mobile applications for specific platforms. The goal of such ontology is to create a knowledge basis for information system that could guide the development teams in increasing the methodological interoperability by reusing the created artifacts.

In order to choose an ontology development methodology, we gave a short overview of several influencing ontology development methodologies which are either commonly used today or made a great influence on the development of other methodologies. In this context, we decided to use Noy and McGuiness' methodology, namely Ontology Development 101, which by its characteristics can be described as agile ontology development methodology. This methodology consists of seven steps which are designed as guidelines in iterative ontology development from scratch to final ontology.

Finally, we argued about the possibilities of using different ontology development tools and ontology development languages. The research performed by Khondoker and Mueller (2010) showed that by far the most widely used tool is Protégé tool developed at Stanford University. As Protégé is aligned with the OD101 methodology, and being widely used from scientists and practitioners in, among others, fields of Information Systems Development and Knowledge Management, we decided to use it in our research as well. Subsequently, as Protégé works with two ontology representation languages, Frames and OWL, we discussed both and selected OWL2 DL as the most appropriate language in our case.

Having selected the ontology development methodology, development tool and representation language we can advance to the next step in our research process – to define the ontology for Android and Windows phone artifacts created in Mobile-D managed development process.

5.2. Android artifacts ontology

This chapter presents the development process and the final ontology describing the artifacts that arose in the development of our prototype application for Android target platform by using Mobile-D methodology. As described in previous chapters, we decided to use Noy's and McGuinness's Ontology Development 101 (OD101) methodology as the guidelines for development process. We also decided to use Protégé tool and to develop OWL2 DL ontology.

The mentioned OD101 methodology is in detail described in (Noy and McGuinness, 2001) as an iterative approach in ontology development that gained popularity mainly because of its simplicity, clarity and focus on the results. Basic assumptions incorporated into the OD101 methodology include that there is no single correct way to model a domain; the best solution always depends on the application and the expected extensions of the ontology; that ontology development is necessarily an iterative process; and that concepts in the ontology should be close to objects (nouns) and relationships (verbs) in the domain of interest (in the sentences that describe the domain).

As we described in chapter 5.1.5.2, the whole methodology consists of execution of seven steps. The following sections describe the final results obtained at the end of iterative ontology development process.

5.2.1. The domain and the scope of the ontology

The domain and the scope of our ontology are clearly defined from the beginning of this research process and there was no need for us to define it from scratch. As stated in our research goals, the ontological description should describe the elements of methodological interoperability containing structural and semantic aspects of sets of artifacts created in the development process of (in this case) Android mobile application. Such ontology will be reused in subsequent research steps to develop a common ontology for two target platforms that aim to help in achieving higher methodological interoperability.

In order to precisely direct the ontology development process, we also defined a set of competency questions that a knowledge base, based on this ontology, should be able to answer:

- What are development phases, activities and tasks in Mobile-D methodology?
- As Mobile-D is an iterative process, what are the exact tasks performed in every activity?
- What artifacts arise in the development process of Android mobile application?
- What artifacts originate from the used development methodology and what from Android target platform?
- What are the categories that these artifacts can be categorized into?
- What artifacts are classified in any specific category?
- In what tasks are the specific artifacts created, updated or used?
- How are the artifacts mutually connected?
- What is the hierarchy among the identified artifacts?
- What are the final products in the development process?
- What artifacts are only used and not created in the process?

As it can be seen from the list of defined questions, the ontology should be capable of answering the questions regarding the structural aspects of methodological phases, activities and tasks, structural aspects of the identified artifacts and the semantic aspects regarding the origin, type and use of artifacts.

5.2.2. Reuse of existing ontologies

We performed research and went through several ontology libraries (including Protégé Ontology Library[32], DAML Ontology Library[33] and ONKI Ontology Library Service[34]) but were not able to find any existing ontology that deals with mobile applications development, android development, software development artifacts or software development methodologies that were suitable for reuse in our case. We have been able to reuse some vocabulary from top level (upper) ontologies, but as our vocabulary was simple and in this case we do not put specific focus to the vocabulary, we decided to build an ontology from scratch.

5.2.3. Identified terms

The list of terms that arise in our domain of interest was incrementally created during the whole ontology development process. The final list of terms that are the base for our ontology includes: *phase, activity, task, artifact, task input, task output, artifact type, artifact origin, artifact usage, artifacts hierarchy*. Mentioned terms are described in Table 45.

Table 45 - Basic terms in Android Case Ontology

Term	Context
Phase	Mobile-D phases.
Activity	Mobile-D activities structured according to phases.
Task	Mobile-D tasks structured according to activities.
Task input	Artifacts that are used as input while performing specific tasks.
Task output	Artifacts that are produced or updated while performing specific tasks.
Artifact	Any piece of software developed and used during software development and maintenance. It includes models, tools, templates, documents et cetera.
Artifact type	Characteristic types of artifacts that could be recognized in order to classify all identified artifacts.
Artifact origin	In terms of reusability, artifacts origin becomes important. It defines the origin of artifacts such as identifying those artifacts that are defined (or requested) by used methodology or those that are products specific for target platform.
Artifact usage	The most important term. It includes knowledge on creation, usage and update of the artifacts in concrete tasks.
Artifact hierarchy	Defines hierarchy among artifacts if it exists.

[32] http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library
[33] http://www.daml.org/ontologies/
[34] http://onki.fi/en/browser/

5.2.4. Classes and class hierarchy

In the process of class and hierarchy definition, we followed the advice from Uschold and Gruninger (1996) and used middle-out approach by first defining more salient concepts and then making generalizations and specializations as needed. The approach resulted in total definition of 152 classes that are organized in 7 top level classes (see Figure 39).

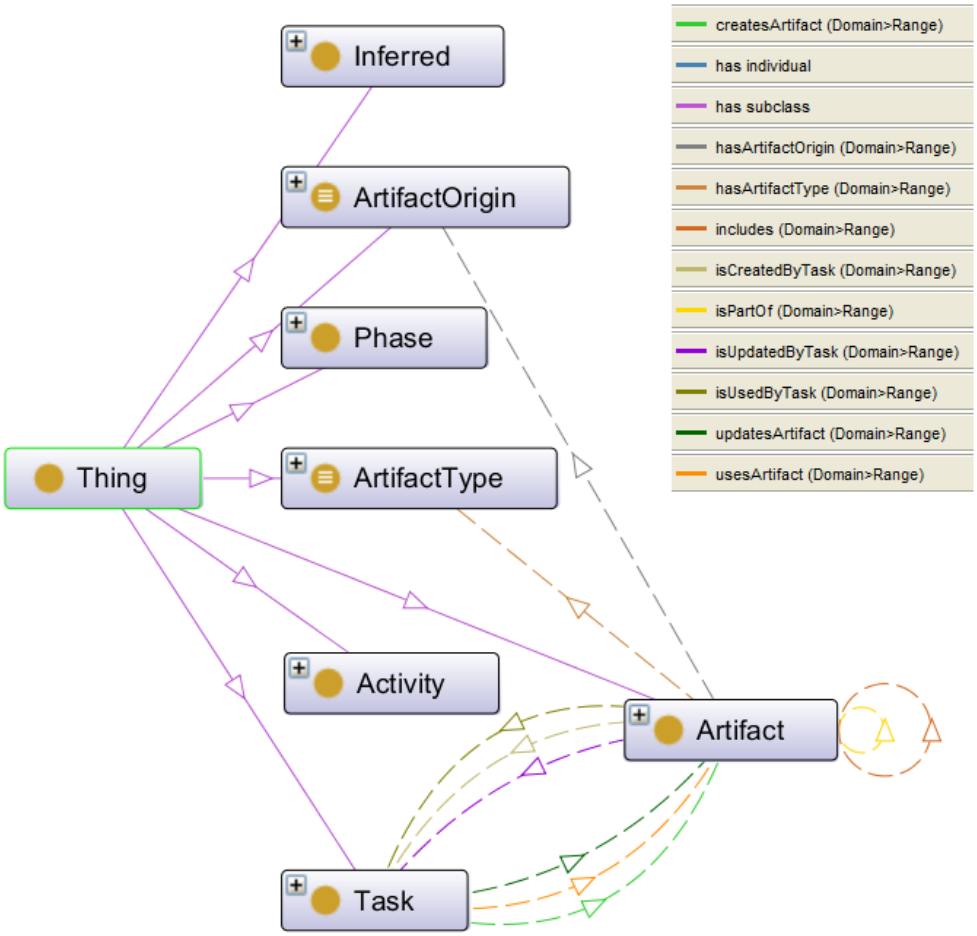


Figure 39 - Android Case ontology top level artifacts

The above figure focuses class *Artifact* which is top level class (*hasParent Thing*) but also has connections with defined classes *Task*, *ArtifactType*, *ArtifactOrigin* and itself. Although existing, the relationships among other top level classes are not presented in this figure.

We believe that at this point, two additional explanations are needed regarding the presented classes. First, class *Inferred* represents all classes defined only by using Description Logic (DL). These classes are populated by respective equivalent classes by the reasoning tool. This is one possible approach in extracting knowledge from ontology definition. Figure 40 shows asserted sub-model of *Inferred* class.

Figure 40 - Android Case ontology asserted subclasses of Inferred class

Secondly, classes *ArtifactOrigin* and *ArtifactType* presented in Figure 39 are created by using the so-called *Value Partition* pattern. This pattern uses a *covering axiom* in order to define a class with finite number of subclasses. In our case, classes have finite number of types and origins.

All other classes created and defined in the ontology, along with the class hierarchy are presented in Table 46.

Table 46 – Android Case ontology classes and class hierarchy

Thing	Phase	Activity	Task	Artifact		Artifact Type	Artifact Origin	Inferred
Phase	Explore	Documentation Wrap-up	Acceptance Test Generation	Acceptance Test	Acceptance Test Template Sheet	Code	Android Artifact	Activities by Phases (5)
Activity	Initialize	Planning Day	Acceptance Testing	Android Activity	Android Class	Document	Methodological Artifact	Android Artifacts
Task	Productionize	Planning Day In 0 Iteration	Acceptance Test Review	API Documentation	Application Description	Document Element	Other Artifact	Borrowed Artifacts
Artifact	Stabilize	Project Establishment	Architecture Line Definition	Application Icon	Application Manifest	Example	Service Artifact	Final Documentation
Artifact Type	System Text And Fix	Project Set-up	Architecture Line Planning	Application Screenshot	Architecture Line Description	License		Final Products
Artifact Origin		Release Day	Continuous Integration	Architecture Line Plan	Class Model Mobile	Model		Methodological Artifacts
Inferred		Scope Definition	Customer Communication Establishment	Class Model Web	Data Model Mobile	Model Element		Other Artifacts
		Stake Holder Establishment	Customer Establishment	Data Model Web	Defect List	Product		Service Artifacts
		System Test	Documentation Wrap-up	Deployment Package	Development Environment	Resource		Tasks by Activities (11)
		Working Day	Environment Set-up	Development Unrelated Software Tool	Driver	Software		Used and Produced Documents
		Working Day In 0 Iteration	Inform Customer	Example Code	Google API Key	Standard		
			Initial Project Planning	Google Play Services	Initial Requirements Document	Template		
			Initial Requirements Analysis	Integration Test	Iteration Backlog			
			Initial Requirements Collection	Iteration Plan	Java Code			
			Iteration Planning	JSON Standard	Layout			

			Pair Programming Practice	Layout Element	Localization String			
			Post Iteration Workshop	Measurement Plan	Mobile Application			
			Pre Release Testing	Mobile-D Process Library	PHP Code			
			Process Establishment	Product Backlog	Product Proposal			
			Publish Application	Project Management Software Tool	Project Plan			
			Refactoring Practice	Project Plan Checklist	Project Plan Checklist Template			
			Release Ceremonies	Project Plan Gantt Chart	Prototype Functionality			
			Requirements Analysis	SADD Document	Story Card			
			System Integration	Story Card Template	System Test Plan			
			System Test	System Test Report	Task Card			
			Test Driven Development Practice	Task Card Template	Test Results			
			Wrap-up	Throwaway Prototype	UI Illustrations			
				UML Class	Unit Test			
				Web Development Environment	Web Service			
				Web Service Specification	XML Resources			

All classes are presented in alphabetical order. Class names are made easier to read by removing suffixes and presenting the names in multiple-word format rather than in a single-word format (so-called CamelCase) that is used in the ontology. Additionally, in the ontology, the classes are described by several annotations including labeling and commenting. Where applicable, description of Mobile-D elements is taken from (Abrahamsson et al., 2005a), while other classes (especially artifacts) are described as presented in chapter 4.2. Additional details on defined classes and the ontology in general including description logic can be found in OWLDoc documentation available at http://barok.foi.hr/~zstapic/ont/acao/doc/.

5.2.5. Properties of classes

Defined properties are closely connected with classes. We define a concept of property as a binary relation between two things. In ontology definition, properties should be observed as relations between individuals that are described through relation between two classes of individuals. Our resulting ontology contains only *object properties* and *annotation properties*, as we had no need to use *datatype properties.*

As *annotation properties* are used to provide ways of describing other ontology elements (for human reading), in this chapter we will put focus on created *object properties*. In order to define knowledge on structure, semantics and usage of ontology elements we defined 12 object properties. Table 47 shows properties and their detailed description.

Table 47 - Android case ontology object properties description

Property	Facets	Description
consistsOf	Domain: *Activity* or *Phase* Range: *Task* or *Activity*	Property connecting individual *Activities* that are performed in specific *Phases* and individual *Tasks* that are performed during specific *Activities*. Logically, this property is inverse property of *isPerformedIn*, but we explicitly defined it in order to have the information available even in the original model.
createsArtifact	Inverse Of: *isCreatedByTask* Domain: *Task* Range: *Artifact*	Inversed property of *isCreatedByTask*. It connects Task individuals and created specific Artifact individuals.
hasArtifactOrigin	Characteristics: *Functional* Domain: *Artifact* Range: *ArtifactOrigin*	Property connecting individual *Artifact* and individual in definite class *ArtifactOrigin* which defines several possible types of Artifact origin. This property is used to classify artifacts by types but from different point of view than property *hasArtifactType*.
hasArtifactType	Characteristics: *Functional* Domain: *Artifact* Range: *ArtifactType*	Property connecting specific *Artifact* individuals with *ArtifactType* individuals. It defines type of the specific Artifact according to defined classification according to artifact usage.
includesArtifact	Characteristics: *Asymmetric* Inverse Of: *isPartOfArtifact* Domain and Range:	Inverse property of *isPartOfArtifact*. It defines individual Artifacts that are included in observed Artifact.

	Artifact	
isCreatedByTask	Inverse Of: *createsArtifact* Domain: *Artifact* Range: *Task*	Property connecting the *Task* individuals that create specific *Artifact* individuals. Creating the artifact logically means it usage even if it is not explicitly stated.
isPartOfArtifact	Characteristics: *Asymmetric* Inverse Of: *includesArtifact* Domain: *Artifact* Range: *Artifact*	Property connecting individual *Artifacts* into hierarchy. This property is *Asymmetric* as two individuals cannot be both part of each other.
isPerformedIn	Domain: *Activity* or *Task* Range: *Phase* or *Activity*	Property defines relationship between specific *Task* individuals and owning *Activity*. Logically, this property is inverse of *consistsOf* property, but we defined both separate to have the information available even in the original model.
isUpdatedByTask	Inverse Of: *updatesArtifact* Domain: *Artifact* Range: *Task*	Property connecting the *Task* individuals that update specific *Artifact* individuals.
isUsedByTask	Inverse Of: *usesArtifact* Domain: *Artifact* Range: *Task*	Property connecting the *Task* individuals that read specific *Artifact* individuals.
updatesArtifact	Inverse Of: *isUpdatedByTask* Domain: *Task* Range: *Artifact*	Inversed property of *isUpdatedByTask*. It connects *Task* individuals and updated specific *Artifact* individuals.
usesArtifact	Inverse Of: *isUsedByTask* Domain: *Task* Range: *Artifact*	Inversed property of *isUsedByTask*. It connects *Task* individuals and used specific *Artifact* individuals.

The restrictions defined by Description Logic (DL) used in OWL 2 DL had some influence on defined *object properties*. For instance, transitive properties cannot be defined as asymmetric or irreflexive, functional properties cannot be transitive etc. But, all concepts that are restricted by direct definition can be modeled alternatively and thus we had no problems that would threaten our logical model.

5.2.6. Knowledge definition and inference

Connecting the instances of classes with defined properties we had to follow OWL 2 DL restrictions, rules and syntax. Additionally, OWL DL is based on Open World Assumption (OWA) logic paradigm, and as we have already stated, the OWA paradigm assumes that we cannot conclude that something does not exist until it is explicitly stated that it does not exist. In other words, we cannot assume that something is *false* just because it is not stated to be *true*. Thus, for example, logical definition of artifact *MobileDProcessLibrary* would be insufficient as presented in Code 4 example.

```
SubClass Of:
Artifact
hasArtifactOrigin some MethodologicalArtifact
hasArtifactType some Document
isUsedByTask some Task
```

Code 4 - Insufficient class description in OWA paradigm

As stated in Code 4 we defined *MobileDProcessLibrary* artifact to be the subclass of a named class *Artifact*, but also to be a subclass of unnamed classes of things that have origin as *MethodologicalArtifact*, or that are of type *Document* or used by any *Task*. The good side of OWA is that in this case we cannot conclude that our artifact is equivalent to other artifacts that for instance have origin as *MethodologicalArtifact*. Such conclusion, even if possible, would be wrong. But, on the other hand, although we only stated that our artifact is used by a Task we cannot conclude that it was not created and was not used by some (the same or another) Task[35]. Thus, query searching for all artifacts that are only used in our process, as presented in Code 5, would not obtain the correct answer.

```
Artifact
 and (not (isCreatedByTask some Task))
 and (not (isUpdatedByTask some Task))
 and (isUsedByTask some Task)
```

Code 5 - Query searching for used but not created Artifacts

In order to completely define the mentioned artifacts we have to use *closure axioms* and to explicitly state that such artifacts were not created and not modified in our development process. Thus, the complete description looks like the one presented in Code 6. Of course, there are additional possibilities of "closing" open world logic in OWL but we will not elaborate on them here.

```
SubClass Of:
Artifact
hasArtifactOrigin only MethodologicalArtifact
hasArtifactOrigin some MethodologicalArtifact
hasArtifactType only Document
hasArtifactType some Document
isUsedByTask only Task
isUsedByTask some Task
not (isCreatedByTask some Task)
not (isUpdatedByTask some Task)
```

Code 6 - Sufficient class description in OWA paradigm

Using the same approach, we described every class defined in our ontology. Other examples are more complicated only if many properties are applied. For example (see Code 7),

[35] For instance, this would be possible in CWA paradigm.

SystemTestPlan artifact is defined by six different properties and some of them describe "more than one" cardinality relationship.

```
SubClass Of:
Artifact
hasArtifactOrigin only MethodologicalArtifact
hasArtifactOrigin some MethodologicalArtifact
hasArtifactType only Document
hasArtifactType some Document
isCreatedByTask only InitialProjectPlanningTask
isCreatedByTask some InitialProjectPlanningTask
isUpdatedByTask only
    (InitialRequirementsAnalysisTask
     or PostIterationWorkshopTask
     or ProcessEstablishmentTask
     or SystemTestTask)
isUpdatedByTask some InitialRequirementsAnalysisTask
isUpdatedByTask some PostIterationWorkshopTask
isUpdatedByTask some ProcessEstablishmentTask
isUpdatedByTask some SystemTestTask
isUsedByTask only
    (ArchitectureLineDefinitionTask
     or ArchitectureLinePlanningTask
     or DocumentationWrapUpTask
     or IterationPlanningTask
     or ProcessEstablishmentTask
     or SystemTestTask
     or TestDrivenDevelopmentPractice)
isUsedByTask some ArchitectureLineDefinitionTask
isUsedByTask some ArchitectureLinePlanningTask
isUsedByTask some DocumentationWrapUpTask
isUsedByTask some IterationPlanningTask
isUsedByTask some ProcessEstablishmentTask
isUsedByTask some SystemTestTask
isUsedByTask some TestDrivenDevelopmentPractice
not (isPartOfArtifact some Artifact)
```

Code 7 - Example class description in OWL2 DL

Similarly, DL queries are used to define the already mentioned inferred classes of objects that are from our specific interest in this ontology. We defined 24 DL queries that answer the competency questions stated earlier in this chapter. The examples of created description logic queries are presented in Table 48.

Table 48 - DL Queries for inferred classes

Inferred class	DL Query
Activities by Phases (5)	`isPerformedIn some Explore`
Android Artifacts	`hasArtifactOrigin some AndroidArtifact`

Borrowed Artifacts	`Artifact`
	` and (not (isCreatedByTask some Task))`
	` and (not (isUpdatedByTask some Task))`
	` and (isUsedByTask some Task)`

Final Documentation	`Artifact`
	` and (not (BorrowedArtifacts))`
	` and (not (isPartOfArtifact some Artifact))`
	` and (hasArtifactType some Document)`

Final Products	`Artifact`
	` and (not (BorrowedArtifacts))`
	` and (not (isPartOfArtifact some Artifact))`
	` and (hasArtifactType some Product)`

Methodological Artifacts	`hasArtifactOrigin some MethodologicalArtifact`

Other Artifacts	`Artifact`
	` and (not (AndroidArtifacts`
	` or MethodologicalArtifacts`
	` or ServiceArtifacts))`

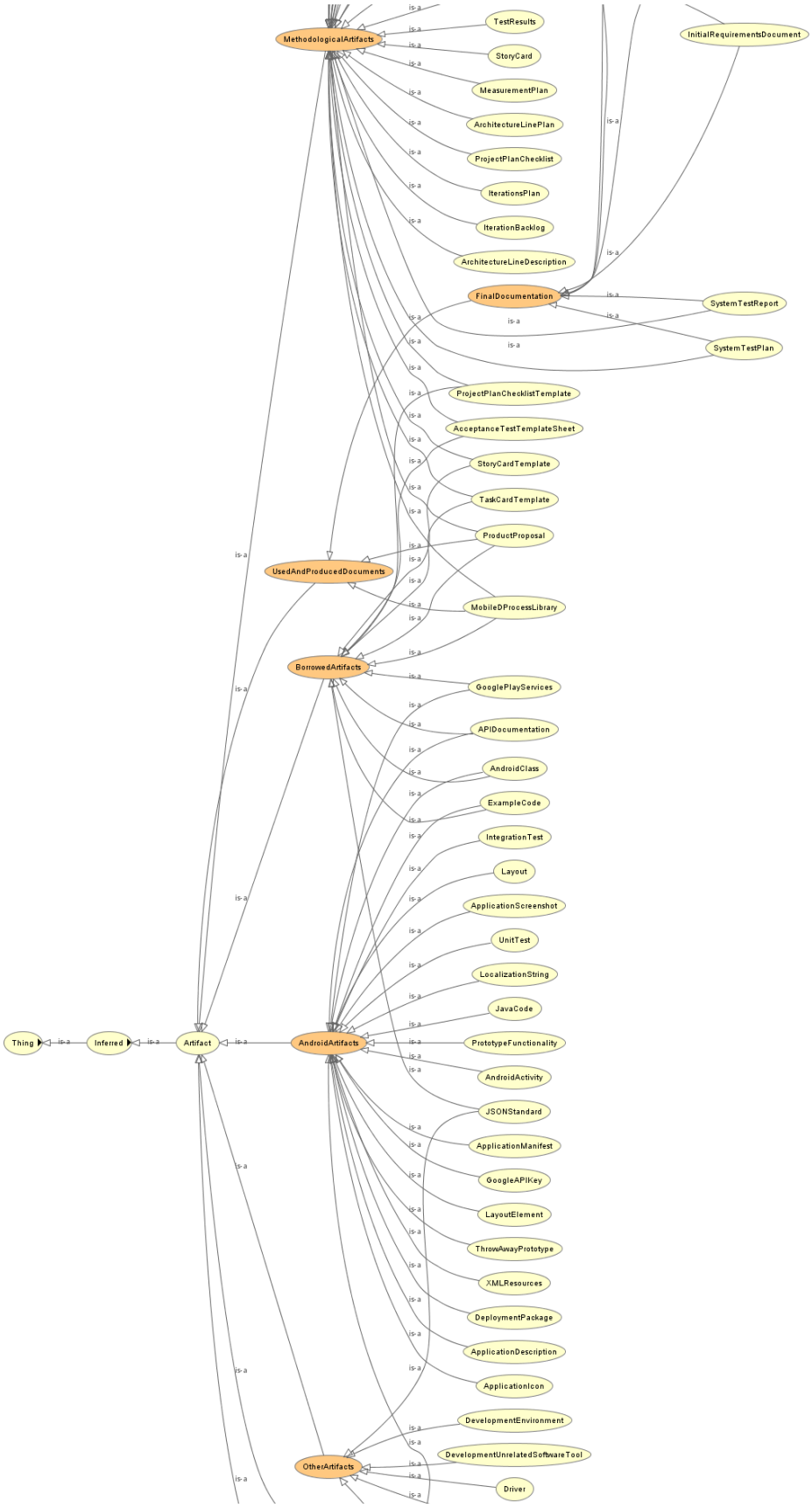
Service Artifacts	`hasArtifactOrigin some ServiceArtifact`
Tasks by Activities (11)	`isPerformedIn some PlanningDayActivity`

Used and Produced Documents	`Artifact`
	` and (not (isPartOfArtifact some Artifact))`
	` and (hasArtifactType some Document)`

A part of inferred model for class Artifact is presented in Figure 41[36]. As we can see, the reasoning system rearranged the artifacts and grouped them according to the defined classes for inference.

Full OWL Documentation for *Android Case Ontology* which contains DL description of all classes and queries is available as OWLDoc on http://barok.foi.hr/~zstapic/ont/acao/doc/.

[36] Full inferred model is available at http://barok.foi.hr/~zstapic/ont/acao/inferred/inferred.png



Full picture is available at http://barok.foi.hr/~zstapic/ont/acao/inferred/artifact.png

Figure 41 - Part of inferred model for class Artifact

5.2.7. Final remarks on Android Case Ontology

By following Ontology Development Methodology 101 (Noy and McGuinness, 2001) we have created an ontology which describes the development process of our prototype android mobile application by utilizing Mobile-D methodology. The point of view taken in this ontology development process, as argued in chapter 4.1, puts the artifacts created and used in this process in a special focus.

The resulting ontology comprises of 152 classes, 12 object properties and 1692 axioms defined by ALCRIF description logic expressivity sub-language[37]. The ALCRIF DL expressivity states that the ontology uses constructs of (AL) Attributive language atomic negation, concept intersection, universal restriction, limited existential qualification, (C) complex concept negation, (R) limited complex role inclusion axioms, reflexivity and irreflexivity, role disjointness, (I) Inverse properties and (F) functional properties.

Due to their size and complexity, we decided not to put Android and Windows Phone ontologies as appendixes to this thesis[38], but to make the ontologies and their full OWLDoc documentation available online. Android Case Artifacts Ontology OWLDoc documentation is available at http://barok.foi.hr/~zstapic/ont/acao/doc/ and ontology in OWL/XML format is available at http://barok.foi.hr/~zstapic/ont/acao.owl.

The ontology syntax and logical correctness was tested by several reasoners, including FaCT++, HermiT 1.3.8, Pellet and RacerPro. Additionally, the inferred knowledge was carefully observed and corrected by the author and the supervisors until we have got errorless results.

This ontology will, along with Windows Phone Case Artifacts Ontology, be used in the last step of our research process the goal of which is to define a common ontological description of multi-platform mobile application development with special focus on artifact reusability.

5.3. Windows Phone artifacts ontology

This chapter presents the development process and the final ontology describing the artifacts that arose in the development of our prototype application for Windows Phone target platform by using Mobile-D methodology. The development of this, second, ontology was a straight-forward task that was performed with a great level of reusability of the existing ontological description created in the Android case. Although we followed again the same ontology

[37] Results are taken from Ontology metrics Protégé plugin.
[38] Final, upperlevel ontology which aims for methodological interoperability is presented in Appendix E.

development methodology (OD101), the first two steps were skipped as the domain and the scope of this ontology are basically the same as described in the first case. In the same manner, competency questions regarding the development methodology, development process, artifacts, their classification and categorization, hierarchy, use etc., also remained unchanged. Finally, the goal of this ontology is also to reason about the mentioned questions, and to use it in the next research step while defining a common ontological description.

5.3.1. Existing ontology reuse

In contrast to the development of the first ontology from scratch, in the second case we were able to reuse our existing ontology. Due to the characteristics and the need of a later ontology merging, the unique ontology element identifiers called Internationalized Resource Identifiers (IRI) should not be changed unless a described concept is logically different from the existing concept.

Thus, we imported an existing ontology, and maximally tried to reuse it while developing the second ontology. Our approach was to change the existing Android elements into applicable Windows Phone elements rather than deleting the Android and creating a new Windows Phone element. The changed concepts got new IRIs, while physically unchanged concepts preserved IRIs created in the Android Case ontology development.

By using Protégé's tool for ontology comparisons and by comparing the first and the second ontology, we can see that 10 ontological elements were renamed, 1 element was added, 16 additional were updated and their IRIs were changed which resulted in small changes in 39 additional elements but their IRIs were not changed. These elements are mainly artifacts and concepts very strictly connected to artifacts.

Having these numbers in mind, we can conclude that 66 concepts out of 165 were changed, and that the rest were reused. Additionally, the comparison was not performed at the level of axioms, but a rough analysis shows that about less than 10% of all axioms (1708) were changed and that the rest were reused.

5.3.2. Classes, properties and hierarchy

The overall asserted class hierarchy defined in the first ontology was not changed in our second case. Only two sets of classes were updated: *ArtifactOrigin* and *Artifact*. As it can be seen in Figure 42, the context of *Artifact* did not change (we changed its subclass structure not visible in this image), while the subclass structure of *ArtifactOrigin* now includes *WindowsPhoneArtifact* class of instances.

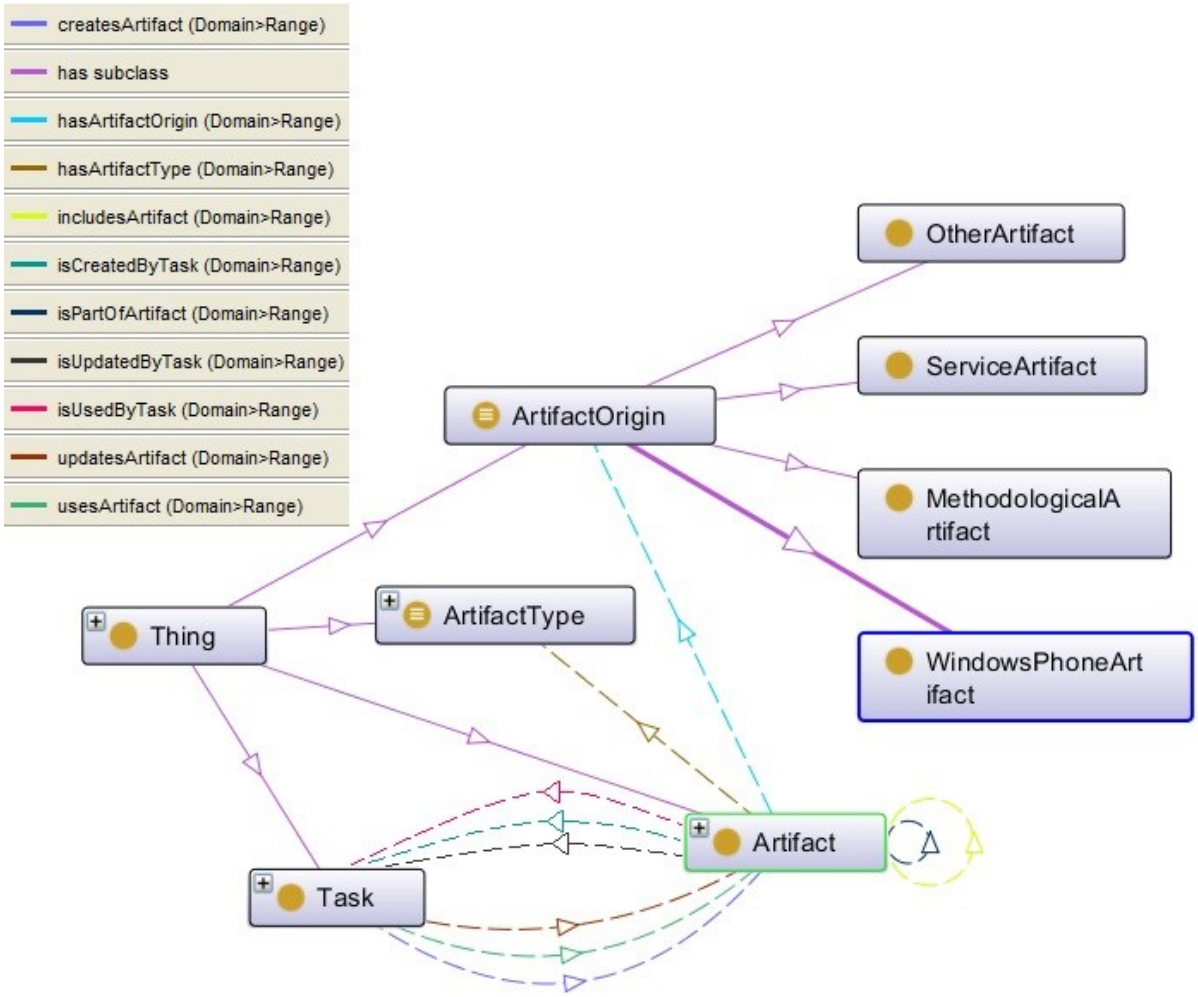


Figure 42 - ArtifactOrigin and Artifact in WP ontology

The most important changes and updates were created in the class of Artifacts, where all Android specific classes have been replaced with Windows Phone specific classes. An interesting point here is that direct mapping between similar concepts in these two platforms was done in 10 out of 11 cases. Only one completely new artifact was identified in Windows Phone environment. Table 49 brings an enumeration of all 61 artifacts that were recognized in WP development case and described in the ontology.

Table 49 - WP case artifacts defined in ontology

Artifact			
Acceptance Test	Acceptance Test Template Sheet	API Documentation	Application Description
Application Icon	Application Screenshot	Architecture Line Description	Architecture Line Plan
Bing Maps Key	Class Model Mobile	Class Model Web	CS Code
Data Model Mobile	Data Model Web	Defect List	Deployment Package
Development Environment	Development Unrelated Software Tool	DotNet Class	Driver

Example Code	Initial Requirements Document	Integration Test	Iteration Backlog
Iteration Plan	JSON Standard	Measurement Plan	Microsoft Phone Controls Toolkit
Mobile Application	Mobile-D Process Library	Page CS	Page XAML
Page XAML Element	PHP Code	Product Backlog	Product Proposal
Project Management Software Tool	Project Plan	Project Plan Checklist	Project Plan Checklist Template
Project Plan Gantt Chart	Prototype Functionality	Resource File	Silverlight Map Control
SADD Document	Story Card	Story Card Template	System Test Plan
System Test Report	Task Card	Task Card Template	Test Results
Throwaway Prototype	UI Illustrations	UML Class	Unit Test
Web Development Environment	Web Service	Web Service Specification	WMAppManifest
XAML Description			

Mapping between Android and WP concepts possible New concept in WP

On the other hand, we reused all property definitions and there was no need to change or update any property (see Table 47 for details on all 12 properties) at this point. This brings us to the conclusion that basic ontological model describing development process for single platform is well defined. This also suggests that the model could be easily reused in definition of development process for other platforms without the need for changing any infrastructural semantic constructs.

The OWLDoc document containing details on defined classes and on the Windows Phone Case Artifacts Ontology in general is available at http://barok.foi.hr/~zstapic/ont/wpcao/doc/. Additionally, figure representing asserted class model along with named DL queries is available at http://barok.foi.hr/~zstapic/ont/wpcao/asserted/full.png.

5.3.3. Updates in knowledge definition

Except the artifacts marked as completely updated or new there are several other artifacts that have undergone some semantic changes in this ontology. It is important to have these changes in mind for the preparation of the ontologies merge and the creation of a common ontology for multi-platform development, as these could be the most hidden sources of future errors and misleading logic.

For example, as shown in Code 8, *Integration Test* artifact (which was classified as *Code* artifact in Android case) is now defined as *Document Artifact* due to the fact that there are no

available automatic or robotized integration testing tools. Although the artifact name remained the same, the new definition included changes in other relations as well, including the Tasks creating, using and updating this artifact and its hierarchy in the artifacts graph.

```
SubClass Of:
Artifact
hasArtifactOrigin only MethodologicalArtifact
hasArtifactOrigin some MethodologicalArtifact
hasArtifactType only DocumentElement
hasArtifactType some DocumentElement
isCreatedByTask only TestDrivenDevelopmentPractice
isCreatedByTask some TestDrivenDevelopmentPractice
isPartOfArtifact only SystemTestPlan
isPartOfArtifact some SystemTestPlan
isUpdatedByTask only
    (ContinuousIntegrationPractice
     or PreReleaseTestingTask
     or SystemIntegrationTask)
isUpdatedByTask some ContinuousIntegrationPractice
isUpdatedByTask some PreReleaseTestingTask
isUpdatedByTask some SystemIntegrationTask
isUsedByTask only
    (ContinuousIntegrationPractice
     or PreReleaseTestingTask
     or SystemIntegrationTask
     or SystemTestTask)
isUsedByTask some ContinuousIntegrationPractice
isUsedByTask some PreReleaseTestingTask
isUsedByTask some SystemIntegrationTask
isUsedByTask some SystemTestTask
```

Code 8 - Updated Integration Test artifact

Other similar changes include different position in hierarchy of *Resource File* artifact if compared to Android artifact with similar purpose (*Localization string*) and changes in description of many artifacts. All these changes will have to be properly addressed in common multi-platform ontology.

All other semantic constructs querying knowledge from the described ontology (as presented in Table 48) remained the same and the mentioned changes in artifacts definition did not influence on them.

For example, the DL query on *Used and Produced Documents* is given in Table 48 and graphical representation of asserted class description is presented in Figure 43. These assertions are created to be populated by a reasoner using the ontologically defined knowledge.

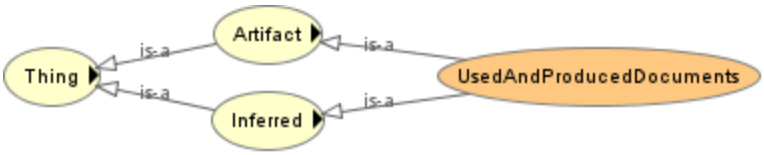


Figure 43 - Used and Produced Documents asserted class model

Thus the inferred model (obtained after performing reasoning on the ontology definition and queries) presented in Figure 44 shows that query named *Used And Produced Documents* is classified as *Artifact* and that it consists of two asserted classes defining documents that are used as inputs in whole Mobile-D process (*Mobile-D Process Library* and *Product Proposal*), and another query named *Final Documentation* that is populated by classes defining Mobile-D produced documents. Asserted classes are light-yellow in presented figures and named DL queries which aim to extract knowledge from the ontology are colored light-brown.

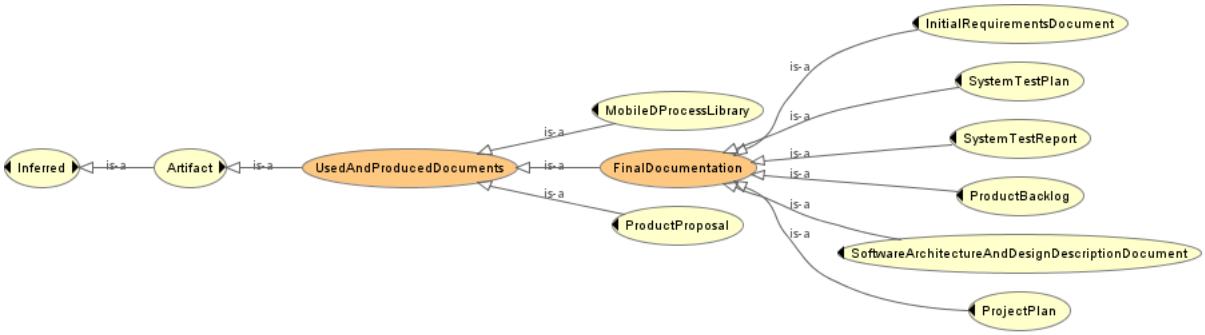


Figure 44 - Used and Produced Documents asserted class model

As the inference in this case, and in all other DL queries, resulted in semantically correct information, we can conclude that this ontology, although upgraded and updated is still logically consistent and valid. This proves extensible and updatable design of our ontology.

5.3.4. Final remarks on Windows Phone Case Ontology

In this chapter we presented the specifics of the created Windows Phone Case Artifacts Ontology. Although we followed OD101 methodology, several steps in ontology definition process were skipped and the results were reused from similar process performed for the Android Case. The most important factor in ontology development process was the possibility of partial reuse of an existing ontology. The basic ontology structure, the properties definition and knowledge extraction DL queries were completely reused, while some classes were reused and other were updated or created from scratch.

The resulting ontology comprises 153 classes, 12 object properties and 1708 axioms defined in the ALCRIF DL expressivity sub-language. Similar to the Android case ontology, due to its size and complexity, we decided not to put the ontology definition as appendix to this thesis, but to make the ontology and its full OWLDoc documentation available online. Windows

Phone Case Artifacts Ontology OWLDoc documentation is available at http://barok.foi.hr/~zstapic/ont/wpcao/doc/ and ontology in OWL/XML format is available at http://barok.foi.hr/~zstapic/ont/wpcao.owl web location.

Another important aspect of this ontology development process is that it proved the validity and flexibility of the existing Android ontology and thus it validated the conceptual model that is the base for our ontologies targeting single platforms. As we argued in the previous chapters, during the update of the existing imported ontology into a new ontology targeting different mobile platforms, there was no need for us to change or update any properties, basic ontology structure, *defined classes* or DL queries. We just had to redefine several *primitive classes* and to align the ontology with the artifact use, types and origin. The tests and the reasoning performed by several reasoners showed that the model is still valid and that the outputs and the results are as expected. This proves the extensibility and updatability of the designed ontology.

This ontology will, along with Android Case Artifacts Ontology, be used in the last step of our research process where we will define a common ontological description of multi-platform mobile application development with a special focus on artifact reusability.

5.4. Common ontology for methodological interoperability

Having the two ontologies describing the development of the same mobile application for two target platforms, we can now move forward and define a new upper-level ontology. This ontology will combine the already described existing knowledge with the new axioms on reusability and thus result in an ontological specification capable of providing the information on methodological interoperability achieved through the artifact reuse.

In this sense, this chapter presents the development process and the final ontology describing the artifacts that arose in the development of our prototype application for two target platforms by using Mobile-D methodology. The chapter presents two distinct sets of activities that were performed during this development. First we merged the two existing ontologies and then created an additional conceptualization related to artifact reusability. In this sense, we had to enhance the methodology that was used in the development of specific ontologies – Ontology Development 101 (Noy and McGuiness, 2001) – as it does not include any tasks related to ontology merging.

In the end of the ontology development process, the ontology was evaluated by seven different mechanisms, including the execution of the sequence of knowledge acquisition queries which gave semantically correct results validated by domain experts.

5.4.1. The domain and the scope of the ontology

The domain and the scope of the ontology were defined at the beginning of our research process. We aimed to ontologically describe the elements of methodological interoperability containing structural and semantic aspects of sets of artifacts created in the development process of multi-platform mobile application.

By *structural aspects* we presume the modeling and knowledge of connections and hierarchy that occur among artifacts (inter-artifact), along with those that occur in relationships of artifact-task, task-activity and activity-phase in the selected development methodology. By *semantic aspects* we imply the conceptualization of knowledge that includes artifact's meaning, its content, classification and possibility of reuse. The combined structural and semantic knowledge should provide solid basis capable of answering competency questions.

We already defined competency questions related to application development targeting any single platform. Those questions should be answerable with this ontology as well, and they include:

- What are development phases, activities and tasks in Mobile-D methodology?
- As Mobile-D is iterative process, what are the exact tasks performed in every activity?
- What artifacts arise in the development process of Android mobile application?
- What artifacts originate from used development methodology and what from Android target platform?
- What are the categories that these artifacts can be categorized into?
- What artifacts are classified in any specific category?
- In what tasks are the specific artifacts created, updated or used?
- How are the artifacts mutually connected?
- What is the hierarchy among the identified artifacts?
- What are the final products in development process?
- What artifacts are only used and not created in the process?

We updated this list with an additional set of questions regarding the artifact reusability semantics. These new questions that guided us when enhancing the existing merged ontologies are stated as follows:

- What platform specific artifacts are classified as reusable?
- What artifacts can be reused in any given development phase?
- What artifacts can be reused in any given development activity or task?
- What artifacts are reusable in accordance with their type or origin?

The list of defined questions can be extended if necessary, but for the purpose of this research and in accordance with our research goals we found it sufficient to include the knowledge

regarding the structural aspects of methodological phases, activities and tasks, structural aspects of the identified artifacts, semantic aspects regarding the origin, type, use and reuse of artifacts. Although it is not in the scope of this research, we sincerely encourage the analysis of another semantic aspect – intra-artifact aspect – which should answer questions like "*Which part of any partially reusable artifact can be reused and which does not?*" or "*How specific artifact is reusable: by its structure, content, inner logic or their combination?*"

5.4.2. Merging the existing ontologies

The development process of the upper-level ontology (namely *Multiplatform Case Artifacts Ontology*) was significantly determined by the fact that we had already developed two platform specific ontologies which should be reused and thus the ontology development process included two rather distinct tasks: reusing the existing ontologies and semantically enhancing the new one.

Although the Ontology Development 101 (Noy and McGuiness, 2001) advises the reuse of existing ontologies, it does not provide any instructions on how to implement existing ontologies into a new one. The decision is left to the developer, and in general there are two main approaches that can be taken: existing ontology/ontologies *import* or existing ontologies *merge*. The import is usually a better option if the existing ontologies are distinct (e.g. disjoint by their constructs) and if there is no need for changing them. In our case, the existing ontologies overlapped significantly semantically and even physically and additionally, it was necessary for us to add new knowledge regarding reusability in existing constructs. On top of that, while developing the *Windows Phone Case Artifacts Ontology* we put a significant effort in properly reusing the *Android Case Artifacts Ontology* in order to make the merging process easier.

The two mentioned ontologies were merged by Protégé's *Ontology merging* tool. This tool, as well as other ontology merging tools, does not provide many merging options. No effort was done to automatically resolve any conflicts, and no effort was done either to provide the user with report on these conflicts as well. The tool simply merges concepts with exactly the same IRI into one concept, and all other concepts are left intact.

However, this lack in ontology merging tools had no significant influence on our merging process, as all platform independent artifacts had the same (reused) IRI, while other, platform dependent artifacts had platform specific IRIs, which ensured that all platform specific artifacts were preserved in the new ontology. An example of automatically merged ontology is given in Figure 45.

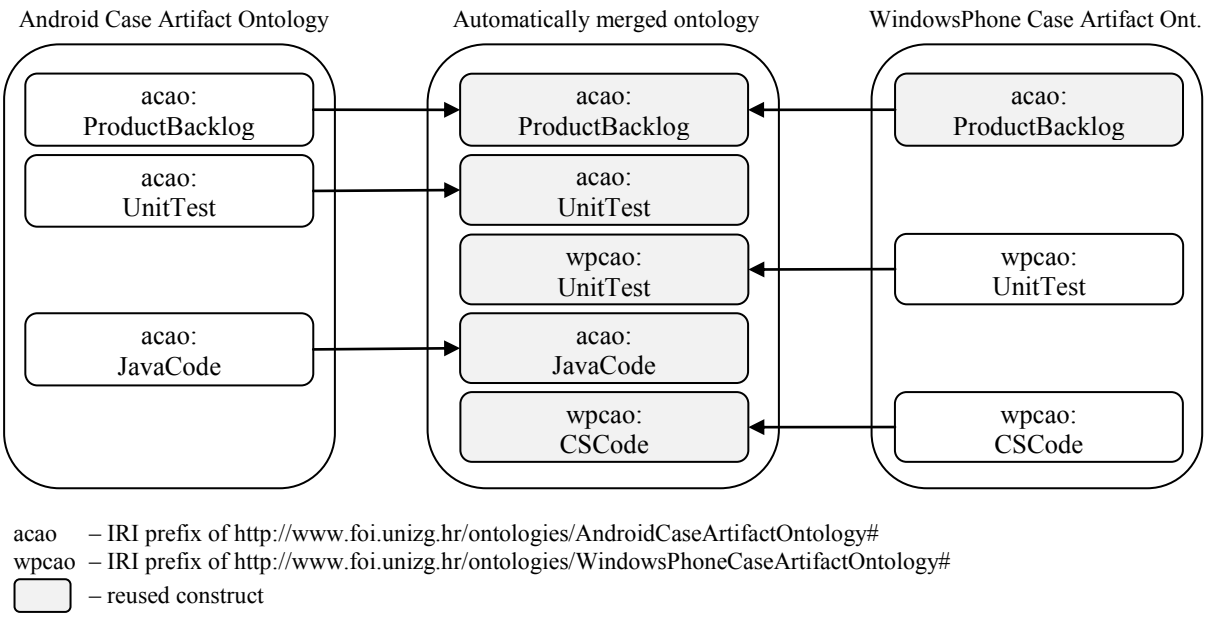


Figure 45 - Example of automatically merged ontology

As it can be seen from the above Figure, when it comes to merging of the artifacts, we had three specific cases. First, the most common case represents the merge of the two already reused constructs, which resulted in a single new construct. This case covers all classes regarding phases, activities, tasks, inferred knowledge and platform independent artifacts. In the second and the third case, we had different (but semantically similar) constructs, and in both cases, all artifacts were preserved, only this time the artifacts were reused representations of the existing artifacts. We use the word *representation* to denote that these are new artifacts in any case.

However, a semantically similar construct was still not connected by any means of class or property connection. Thus, our first step was to resolve the lack of connection between the logical pairs of artifacts and to properly describe them. Out of many possible approaches, we decided to create a super class for every pair of artifacts and to connect them by making them members of the same class. The resulting ontology, at this point, looked as it is shown in example Figure 46. Finally, we extracted the existing but common ontological description of the elements of each pair and we assigned this description to the newly defined super classes. In total 22 new classes have been created.

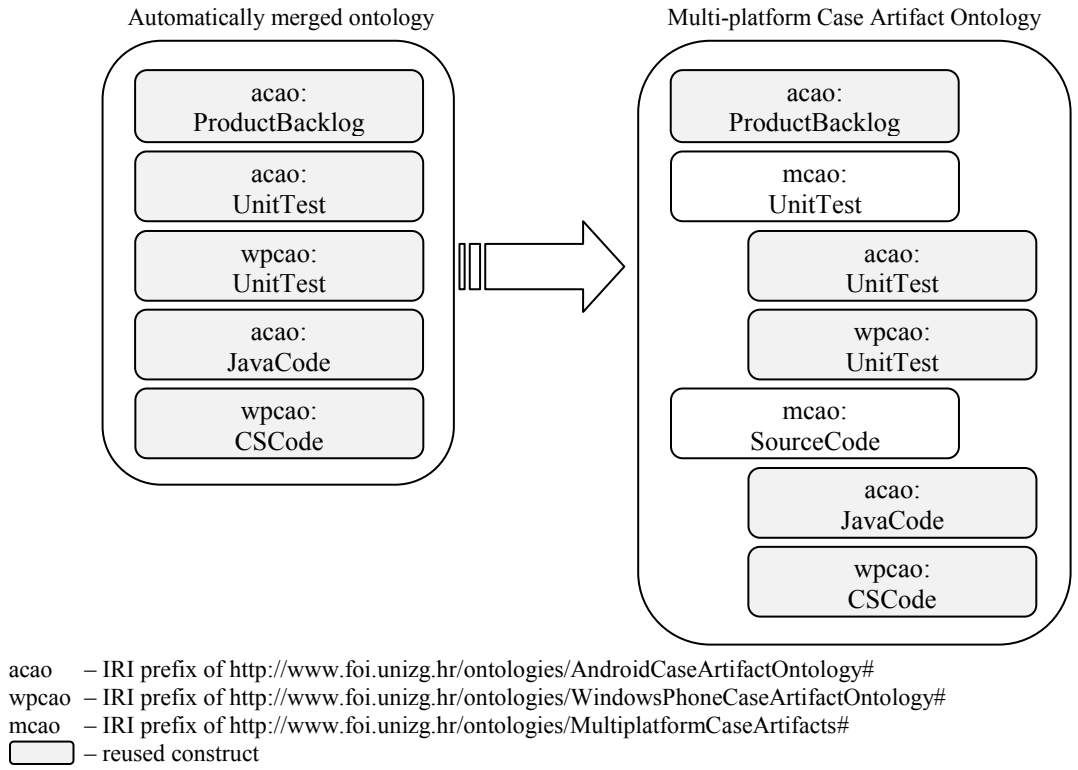


Figure 46 - Example of merged ontology

This completed our activities of merging the existing ontologies into a single upper-level ontology. As the single ontology inherited (and will enhance) all conceptualization from the previously created Android Case and WP Case ontologies, we can say that our ontologies describing specific cases are now deprecated and should not be used. In favor of this goes the fact that it is generally much easier to update upper-level ontology with the knowledge on an additional target platform than to create a new ontology from scratch.

5.4.3. Updating the basic terms

While proceeding to enhance the merged ontology with the semantic information on reusability, we continued to follow the Ontology Development 101 methodology. This process (which consists of 7 steps) was described in detail in the previous chapters (see chapters 5.1.5.2 and 5.2 on pages 162 and 169) and thus we will not discuss it here. Rather, we will present its results and point out all important aspects of the process itself and of the created ontology.

The basic terms defined for the Android Case ontology were reused in Windows Phone Case ontology and thus are included in this ontology as well. As we aim to enhance the ontology with the conceptualization on artifact reusability, we had to introduce a couple of new important terms. The final list, containing both, previously stated and the new set of terms is presented in Table 50.

Table 50 - Final list of terms used in Multiplatform ontology

Term	Context
Phase	Mobile-D phases.
Activity	Mobile-D activities structured according to phases.
Task	Mobile-D tasks structured according to activities.
Task input	Artifacts that are used as input while performing specific tasks.
Task output	Artifacts that are produced or updated while performing specific tasks.
Artifact	Any piece of software developed and used during software development and maintenance. It includes models, tools, templates, documents et cetera.
Artifact type	Characteristic types of artifacts that could be recognized in order to classify all identified artifacts.
Artifact origin	In terms of reusability, artifacts origin becomes important. It defines the origin of artifacts such as identifying those artifacts that are defined (or requested) by used methodology or those that are products specific for target platform.
Artifact usage	Term includes knowledge on creation, usage and update of the artifacts in concrete tasks.
Artifact hierarchy	Defines hierarchy among artifacts if it exists.
Reusability	Identified artifact reusability levels which denote if artifacts are completely, partially or not reusable.
Artifact similarity	Defines mutual reusability among artifacts.

As we can see, the *reusability* and *artifact similarity* are two newly added terms. The first term relates to the concepts of levels of reusability and as defined in chapter 4.4, we classified all the artifacts into three reusability levels: *partially reusable*, *completely reusable* and *not reusable artifacts*. The other concept relates to inter-artifact similarity defining pairs of similar artifacts.

5.4.4. Final class and properties hierarchy

The new model of top-level classes with the focus on the *Artifact* class is given in Figure 47. If compared to Figure 39 there are not many changes at the top level classes of the ontology. The set of top level concepts remained the same, while the only difference is addition of a new *value partition* class *ReuseLevel*. The figure describing the new ontology shows that *Artifact* is finally connected with *Task*, *ArtifactOrigin*, *ArtifactType* and *ReuseLevel*. Among these relationships, the relationship with *Task* is the strongest as it is defined with three properties (each of them having inversed property). Although existing, the relationships among other top level classes are not presented in this figure in order to make it focus on artifacts only.

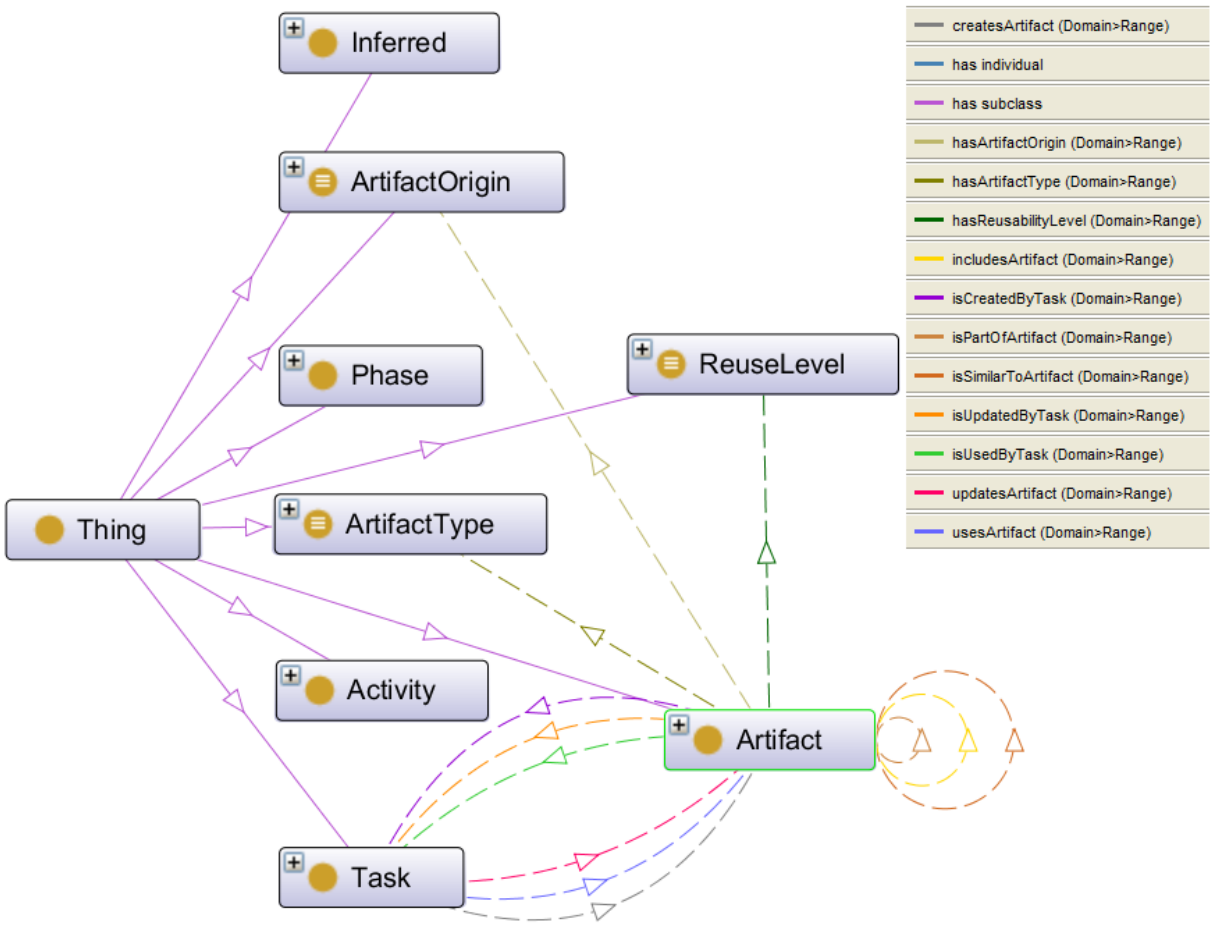


Figure 47 - Top level artifacts

The completed ontology consists of 213 classes, 14 properties and 2213 axioms. Important classes to mention here are the classes organized under *Inferred* class. As we have already discussed in the chapter on the Android Case ontology development, these classes are defined only by using Description Logic (DL). These classes are populated by their respective equivalent classes by reasoning tool, and this is one possible approach in extracting knowledge from the ontology definition. The final version of asserted sub-model of *Inferred* class is presented in Figure 48.

Secondly, classes ArtifactOrigin and ArtifactType and ReuseLevel presented in Figure 47 are created by using the so-called *Value Partition* pattern. This pattern uses a *covering axiom* in order to define a class with a finite number of subclasses. In our case, classes have finite number of types, origins and levels.

All other classes created and defined in the final ontology, along with class hierarchy are presented in Table 51. Due to space constraints and table size, we decided not to present the leafage of the platform specific artifacts and inferred classes as these have already been presented in the thesis. Instead, we present here in light gray color those artifacts that have specific subclasses for each platform. The number of subclasses is presented in braces.

Figure 48 - Asserted subclasses of Inferred class

Table 51 - Classes and class hierarchy

Thing	Phase	Activity	Task	Artifact		Artifact Type	Artifact Origin	Reuse Level	Inferred
Phase	Explore	Documentation Wrap-up	Acceptance Test Generation	Acceptance Test	Acceptance Test Template Sheet	Code	Android Artifact	Completely	Activities by Phases (5)
Activity	Initialize	Planning Day	Acceptance Testing	API Documentation (2)	APP Description (2)	Document	Methodological Artifact	None	Artifacts Origin (5)
Task	Productionize	Planning Day In 0 Iteration	Acceptance Test Review	App Icon (2)	App Manifest (2)	Document Element	Other Artifact	Partially	Artifact Reusability (4)
Artifact	Stabilize	Project Establishment	Architecture Line Definition	App Prototype Functionality (2)	App Reference (4)	Example	Service Artifact		Artifacts Usage (6)
Artifact Type	System Text And Fix	Project Set-up	Architecture Line Planning	App Resource (4)	App Screenshot (2)	License			Task by Activities (11)
Artifact Origin		Release Day	Continuous Integration	Architecture Line Description	Architecture Line Plan	Model			
Reuse Level		Scope Definition	Customer Communication Establishment	Class Model Mobile	Class Model Web	Model Element			
Inferred		Stake Holder Establishment	Customer Establishment	Data Model Mobile	Data Model Web	Product			
		System Test	Documentation Wrap-up	Defect List	Deployment Package (2)	Resource			
		Working Day	Environment Set-up	Development Environment (2)	Development Unrelated Software Tool	Software			
		Working Day In 0 Iteration	Inform Customer	Example Code (2)	Initial Requirements Document	Standard			
			Initial Project Planning	Integration Test (2)	Iteration Backlog	Template			
			Initial Requirements Analysis	Iteration Plan	JSON Standard				
			Initial Requirements Collection	Maps Key (2)	Measurement Plan				
			Iteration Planning	Mobile Application (2)	Mobile-D Process Library				
			Pair Programming Practice	PHP Code	Product Backlog				

			Post Iteration Workshop	Product Proposal	Project Management Software Tool				
			Pre Release Testing	Project Plan	Project Plan Checklist				
			Process Establishment	Project Plan Checklist Template	Project Plan Gantt Chart				
			Publish Application	SADD Document	Source Code (2)				
			Refactoring Practice	Story Card	Story Card Template				
			Release Ceremonies	System Test Plan	System Test Report				
			Requirements Analysis	Task Card	Task Card Template				
			System Integration	Test Device Driver (2)	Test Results				
			System Test	Throwaway Prototype (2)	UI Illustrations				
			Test Driven Development Practice	UML Class	UML Class SDK (2)				
			Wrap-up	Unit Test (2)	View (2)				
				View Controller (2)	View Element (2)				
				Web Development Environment	Web Service				
				Web Service Specification					

Classes having additional sub-classes not presented in this table.
Number of subclasses is denoted in braces.

The approach in class naming and description defined in development of platform specific ontologies was also reused in the merged ontology. Thus, the classes are named in *CamelCase* style and described with several annotation properties including labeling, commenting and notes making. Where applicable, description of Mobile-D elements is taken from (Abrahamsson et al., 2005a), while other classes (especially artifacts) are described as presented in Chapter 4.

In addition to the 213 classes, the conceptualization is created with 14 *object properties*. We already discussed the types of properties and concluded that our ontology does not need *datatype properties*, but only *object properties* which are defined as relationship between two classes of individuals. The properties defined for platform specific ontologies are reused and updated with *isSimilarToArtifact* and *hasReusabilityLevel* properties. The mentioned two properties are used to describe the knowledge on artifacts reusability and similarity with other artifacts. The final list of all the properties created and used in our ontology is presented in Table 52.

Table 52 - Object properties description

Property	Facets	Description
consistsOf	Domain: *Activity* or *Phase* Range: *Task* or *Activity*	Property connecting individual *Activities* that are performed in specific *Phases* and individual *Tasks* that are performed during specific *Activities*. Logically, this property is inverse property of *isPerformedIn*, but we explicitly defined it in order to have the information available even in the original model.
createsArtifact	Inverse Of: *isCreatedByTask* Domain: *Task* Range: *Artifact*	Inversed property of *isCreatedByTask*. It connects Task individuals and created specific Artifact individuals.
hasArtifactOrigin	Characteristics: *Functional* Domain: *Artifact* Range: *ArtifactOrigin*	Property connecting individual *Artifact* and individual in definite class *ArtifactOrigin* which defines several possible types of Artifact origin. This property is used to classify artifacts by types but from different point of view than property *hasArtifactType*.
hasArtifactType	Characteristics: *Functional* Domain: *Artifact* Range: *ArtifactType*	Property connecting specific *Artifact* individuals with *ArtifactType* individuals. It defines type of the specific Artifact according to defined classification according to artifact usage.
includesArtifact	Characteristics: *Asymmetric* Inverse Of: *isPartOfArtifact* Domain and Range: *Artifact*	Inverse property of *isPartOfArtifact*. It defines individual Artifacts that are included in observed Artifact.
hasReusabilityLevel	Characteristics: *Functional* Domain: *Artifact* Range: *ReuseLevel*	Property connecting specific *Artifact* individuals with one of predefined reusability levels. This property classifies artifacts into completely, partially or unreusable classes.
isCreatedByTask	Inverse Of: *createsArtifact* Domain: *Artifact* Range: *Task*	Property connecting the *Task* individuals that create specific *Artifact* individuals. Creating the artifact logically means it usage even if it is not explicitly stated.

isPartOfArtifact	Characteristics: *Asymmetric* Inverse Of: *includesArtifact* Domain: *Artifact* Range: *Artifact*	Property connecting individual *Artifacts* into hierarchy. This property is *Asymmetric* as two individuals cannot be both part of each other.
isPerformedIn	Domain: *Activity* or *Task* Range: *Phase* or *Activity*	Property defines relationship between specific *Task* individuals and owning *Activity*. Logically, this property is inverse property of *consistsOf* property, but we defined both separate to have the information available even in the original model.
isSimilarToArtifact	Characteristics: *Symmetric* Inverse Of: *isSimilarToArtifact* Domain and Range: *Artifact*	Property connecting the individuals of class Artifact with other similar individuals of the same class. Usually, all artifacts in the same class, if class is reusable, are reusable, but this is not a rule. Sometimes, pairs of artifacts in the same class can be mutually reusable, but not reusable with other artifacts of pairs.
isUpdatedByTask	Inverse Of: *updatesArtifact* Domain: *Artifact* Range: *Task*	Property connecting the *Task* individuals that update specific *Artifact* individuals.
isUsedByTask	Inverse Of: *usesArtifact* Domain: *Artifact* Range: *Task*	Property connecting the *Task* individuals that read specific *Artifact* individuals.
updatesArtifact	Inverse Of: *isUpdatedByTask* Domain: *Task* Range: *Artifact*	Inversed property of *isUpdatedByTask*. It connects *Task* individuals and updated specific *Artifact* individuals.
usesArtifact	Inverse Of: *isUsedByTask* Domain: *Task* Range: *Artifact*	Inversed property of *isUsedByTask*. It connects *Task* individuals and used specific *Artifact* individuals.

As we have argued in the chapter on Android Case ontology, there are some restrictions on property definitions defined by OWL 2 DL. Each time we broke a restriction on properties, the reasoners started to behave unexpectedly, sometimes reporting the use of unsupported logic and sometimes just crashing without any explanation. For instance, querying the knowledge out of the ontology would be much easier if there was a possibility of defining the same property to be symmetric and transitive or defining functional property to be transitive et cetera. However, when needed, we used other approaches and assured that our logical model is safe and that the ontological description is correct.

The complete ontological definition presented in *Manchester OWL Syntax* format[39] and containing the details on classes, properties, class and property description and semantics can

[39] The Manchester syntax is a user-friendly compact syntax for OWL 2 ontologies (Horridge and Patel-Schneider, 2009). Although it is frame-based, as opposed to the axiom-based other syntaxes for OWL 2, we find it to be the most compact and human readable syntax that can be easily and automatically converted in other OWL 2 syntaxes.

be found in Appendix E of this document. We also generated a full OWLDoc documentation on the created ontology and made it available for access and analysis at http://barok.foi.hr/~zstapic/ont/mcao/doc/.

5.4.5. Evaluating and testing the ontology

5.4.5.1. Ontology evaluation

Ontology *evaluation* means to judge the ontologies against a reference framework during each phase and between phases of its life cycle (Gómez-Pérez, 2001). Examples of reference frameworks (according to the same author) can be real world, a set of requirements and a set of competency questions. However, Gómez-Pérez argues, that there are few ontology development methodologies that have evaluation included throughout the entire lifetime of the ontology development process. In the terms of classifying the ontologies according to their formalization level (Uschold and Gruninger, 1996), integrated formal evaluation is possible only in development process of *rigidly formal* ontologies, while in all other ontologies, we need different and other approaches.

According to Brank et al. (2005), most evaluation approaches fall into one of the following categories:

- evaluation based on comparing the ontology to a "golden standard" which may itself be an ontology, syntax specification or any other representation that is considered to be a good representation of the concepts of the problem domain under consideration,
- evaluation based on using the ontology in an application and evaluating the results,
- evaluation involving comparison with a source of data (e.g. a collection of documents about the domain to be covered by the ontology,
- or evaluation done by humans who try to assess how well the ontology meets a set of predefined criteria, standards, requirements et cetera.

Performing a review of existing ontology evaluating techniques Brank et al. (2005) concluded that ontology evaluation is an important open problem with no single best or preferred approach to ontology evaluation. Additionally, Brank thinks that the choice of a suitable approach must depend on the purpose of evaluation, the application in which the ontology is to be used, and on what aspect of the ontology we are trying to evaluate. Finally, Brank stated that automated ontology evaluation should be the focus of future researches.

This research took place in 2005, but since then not many researches were performed. There were some tools and techniques developed, but those were developed for specific ontology development environment or representation languages. In our opinion, Protégé Frames had

good support for ontology evaluation in several tools, including those created in CO-Ode project and *OntoClean* methodology[40]. On the other hand, current support in automatic evaluation tools for Protégé OWL is insufficient. CO-Ode project developed *OWL Lint*[41] framework for defining and running tests against a set of OWL ontologies for quality control, debugging, best practices, and other purposes. Unfortunately, the project is closed and the resources on this tool are unavailable and not aligned with the current version of Protégé. Similarly, *OntoCheck*[42], a simple plugin for verifying the ontology naming conventions and metadata completeness developed at University of Freiburg, is also not aligned with the current version of Protégé.

However, there are some tools that allow basic syntax checking of the ontology, ontology alignment with the OWL standard and consistency of the ontology through check of syntactical ontology elements. In our case, we used two of them: *OWL Validator*[43] developed at the University of Manchester which is used as official W3C OWL validating tool and *Ontology Evaluation*[44], an open source plug-in developed at Aristotle University of Thessaloniki which is currently the only evaluation plugin supported by Protégé OWL version 4.3. We will come back to these tools later in this chapter.

Ontology Development 101 methodology (Noy and McGuinness, 2001), that we used in our development process, also lacks formal ontology evaluation activities and mechanisms. Instead of formal evaluation tasks, the description of the methodological steps is intertwined with recommendations and advices on performing the tasks and evaluating their results. Additionally, the *competency questions* are used as a background for development process and for the final evaluation of the results through the ontology application. As the focus through the whole methodology is placed on (1) utilization of good practices in ontology development, (2) on human checking of intermittent and final results and (3) on the assessment of the quality of the final ontology by using it in applications for which it was designed, it is hard to choose in which of the four categories defined by Brank et al. (2005) this methodology falls into.

Observing the definition of ontology evaluation again, we can conclude that complete and automatic evaluation throughout all phases is still not possible. Rather, it is a human-centric process which is done in every ontology development task with some minor help from the reasoners and syntax checking tools.

---

[40] http://protege.stanford.edu/ontologies/ontoClean/ontoCleanOntology.html
[41] http://protegewiki.stanford.edu/wiki/OWL_Lint
[42] http://protegewiki.stanford.edu/wiki/OntoCheck
[43] http://www.w3.org/2001/sw/wiki/OWL_Validator and http://owl.cs.manchester.ac.uk/validator/
[44] http://protegewiki.stanford.edu/wiki/Ontology_Evaluation

However, we should not forget that evaluation actually subsumes the execution of two steps: *verification* and *validation* (Gómez-Pérez, 2004). *Ontology verification* deals with building the ontology correctly, that is ensuring that its definitions implement correctly the requirements, and *ontology validation* refers to whether the meaning of the definitions really models the real world for which the ontology was created (Vrandečić, 2009). To make the definitions simpler we will also refer to Vrandečić who says that *ontology verification* answers if the ontology was built in the right way, whereas *ontology validation* answers if the right ontology was built.

Finally, in this short introduction to the concepts related to ontology evaluation, we have to point out the role of domain experts. As ontology validation is usually the only way to assure the correctness of ontologically described knowledge, which usually cannot be performed automatically, it is an important part of assessing the quality of an ontology to have the domain experts validating the ontology.

5.4.5.2. Used evaluation mechanisms

In order to verify and validate our ontology, throughout the whole development process lifecycle, we have performed the following seven verification and validation mechanisms:

1. Methodologically driven ontology development process
2. Followed recommendation and advices from other authors
3. Using reasoning tools to verify the ontology in each iteration
4. Using W3C OWL validating tool
5. Using the Ontology evaluation plug-in
6. Using DL queries to obtain information via inference on ontology knowledge
7. Checking the results by domain experts

The first five evaluating mechanisms are connected with ontology verification and are used to lower the risks of making any syntactical and basic semantic errors throughout the whole ontology development process.

The last two mechanisms are connected with ontology validation. These two mechanisms have been used in the end of development process to check if the created ontology represents the domain knowledge in semantically correct way.

By performing the methodologically driven ontology development process and utilizing the Ontology Development 101, we ensured that our approach was systematic and guided by the experience of researchers who already used it. As we have described and discussed in Chapter 5.2, the whole development process had seven steps which were implemented iteratively through several iterations. We followed the recommendation from Uschold and Gruninger (1996) and used middle-out approach in class and class-hierarchy definition. This enabled us

to focus on more salient classes first and then to classify them in super or subclasses as needed. This approach, however, increases the risk of omitting some classes, but we dealt with it through other verification mechanisms.

Noy and McGuinness (2001) put special focus in tasks related to classes and properties definition and they gave a set of recommendations and advices that we tried to follow in our development process. For instance, they gave us advice on measures that should been taken to ensure that the class hierarchy is correct, on analyzing siblings in a class hierarchy, on taking care of multiple inheritances, when to introduce a new class or property or instance of a class, on limiting the scope of the ontology and dealing with disjoint classes. They also gave advice on properties creation and their relationships through facets and on some general issues regarding the ontology creation like the choice of naming convention, of using singular or plural, of using prefixes and suffixes and on use of reserved names and abbreviations. We also consulted the recommendations presented in (Horridge, 2011) who took practical point of view and discussed the advantages and disadvantages of different approaches in solving the most common issues in ontology development.

Throughout the whole incremental development process we used reasoning tool to verify the newly added concepts and their influence on the already defined concepts. In general, Description Logic reasoners check the consistency of ontology and automatically compute the ontology class hierarchy. In this document we referred to computed class hierarchy as to *inferred* class hierarchy. Additionally, a reasoner can check whether or not all of the statements and definitions in the ontology are mutually consistent (Horridge, 2011). If we add the reasoners' possibility to detect and report any syntax errors, then we can conclude that a consistent use of reasoners in development process represents a solid ontology verification mechanism.

We used *FaCT++, HermiT 1.3.8* and *Pallet* reasoners which are available through Protégé installation or through standard plug-in installation procedure. All used reasoners classified our ontology in the same way and returned the same inference results. For the examples presented in this chapter, we used FaCT++ as native Protégé reasoner.

Figure 49 presents comparison of a part of asserted and a part of inferred class hierarchy. As we can see on the left hand side of the figure, asserted hierarchy does not group artifacts into specific super classes regarding their type or usage. However, we used Description Logic to define a set of *Inferred* classes (marked with icon) to access knowledge that is encoded in the ontology. During the ontology definition, some of these classes were automatically classified as sub-classes of class *Artifact*, but as we can see, they are without any child elements. Same classes, along with the rest of ontological description, were used by the

reasoner in order to create a new class hierarchy, as presented on the right hand side of the mentioned Figure 49.

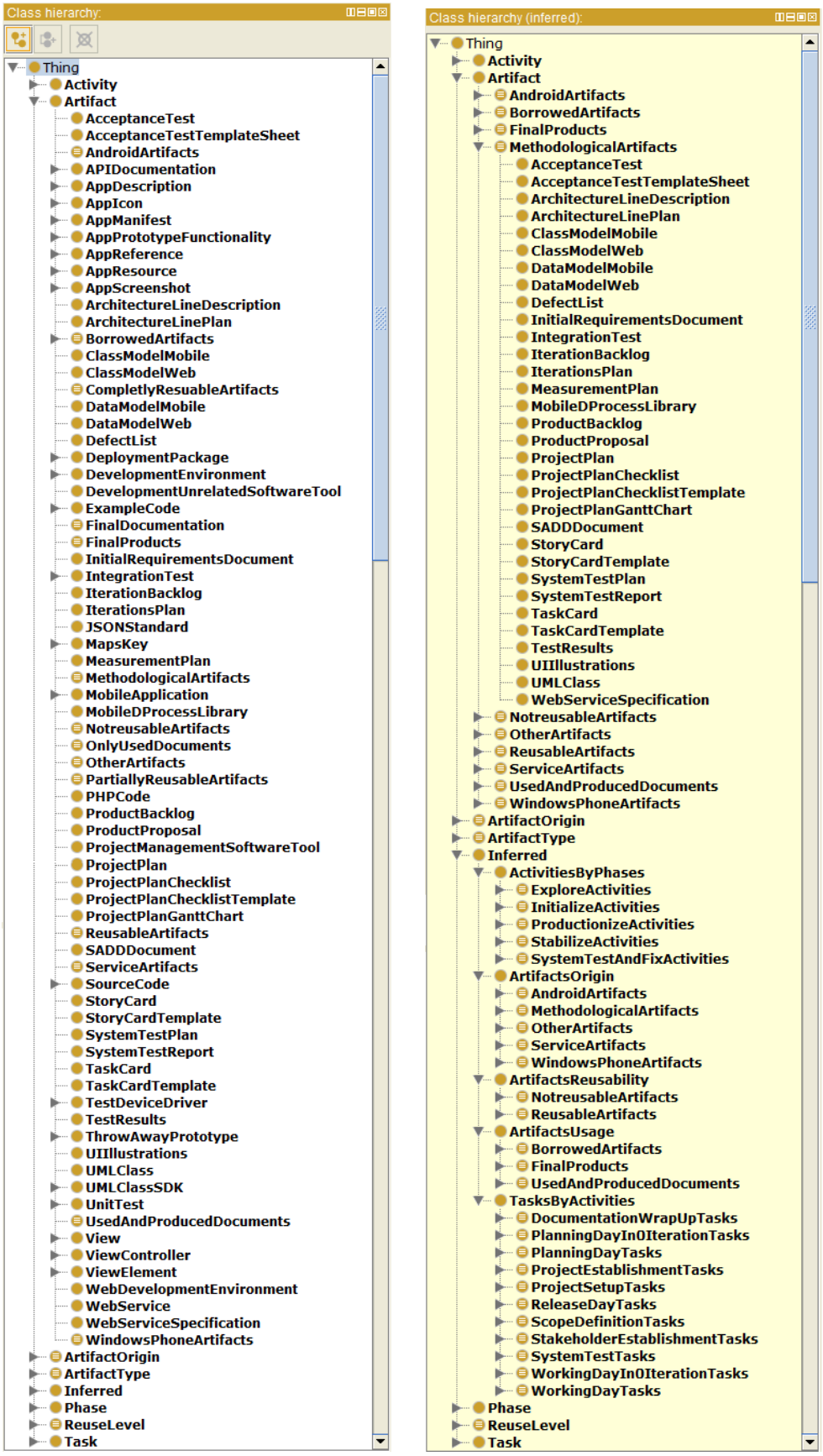


Figure 49 - Comparing asserted and by reasoner inferred class hierarchy

Additionally, as it can be seen on the right hand side, all DL defined classes are now populated with inferred subclasses. In the above example, expanded class *MethodologicalArtifacts* is populated with those artifacts that originate in Mobile-D methodology. Similarly, all other named queries and defined classes are populated with appropriate sub-classes. The asserted and inferred models were in the end assessed by the thesis supervisors who agreed on their consistency and semantic correctness.

In order to evaluate the ontology syntax, we also used two different tools that evaluate the ontology automatically. *OWL Validator* is developed at the University of Manchester, and it is currently an official W3C OWL validating tool (Horridge, 2009). Figure 50 shows the evaluation results stating that the ontology and all of its imports are in the OWL 2 DL profile.

Figure 50 - OWL 2 Validation report results

The other used tool is *Ontology Evaluation*[45] (Tantsis, 2013), a plug-in developed as a Master Thesis project at Aristotle University of Thessaloniki. Although without technical or any other formal documentation and support, except information written in the thesis itself, the plugin is currently, as far as our knowledge reaches, the only evaluation plugin supported by Protégé OWL version 4.3. Thus, even if the quality of the evaluation engine may be questionable, it can help in the evaluation of the ontology according to several parameters including *naming conventions, class hierarchy, property hierarchy, property restrictions, similar concepts, documentation and visualization, domain and range of properties and restrictions on disjointness.* An example of performed tests on class hierarchy and documentation (see Figure 51) showed that there are no problems with class hierarchy, but

[45] http://protegewiki.stanford.edu/wiki/Ontology_Evaluation

some concepts needed improvements in documentation. After additional analysis, it turned out that some mid-level classes were not documented.

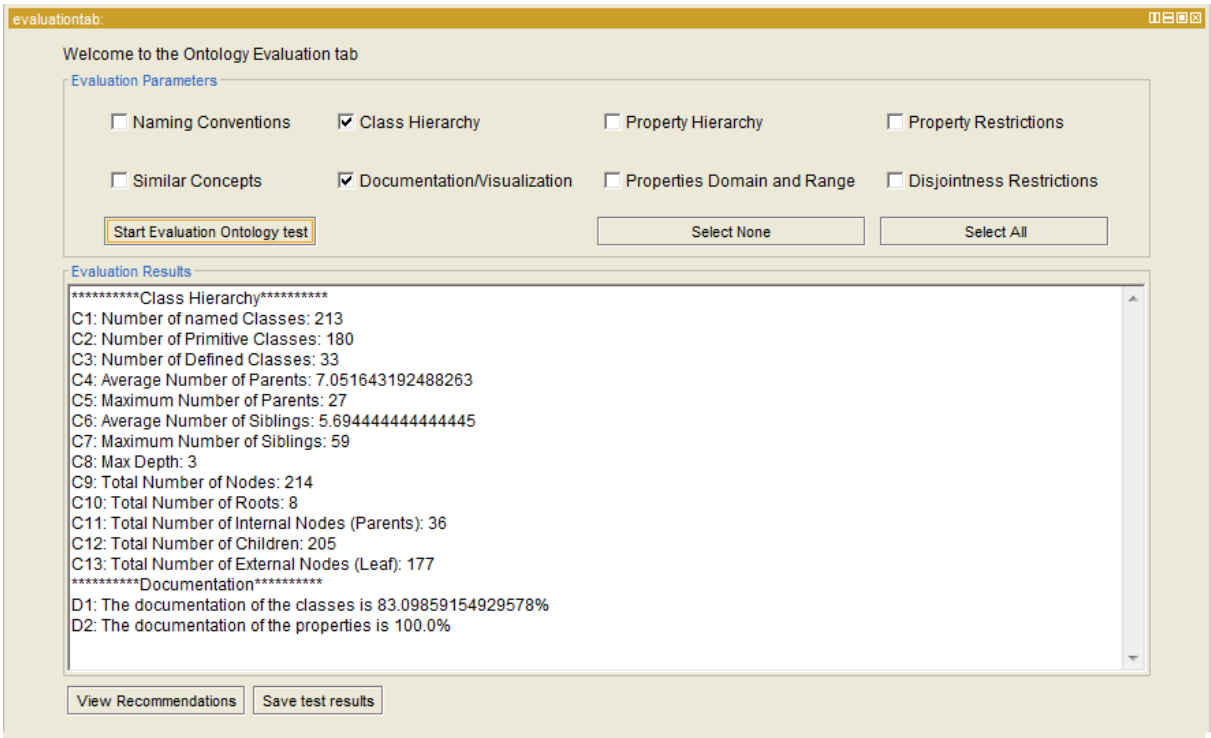


Figure 51 - Ontology Evaluation plug-in

The mentioned evaluation tool, along with the evaluation results creates a set of recommendations that could be used to improve the ontology quality. These recommendations are based on simple evaluation result parameters without any contextual input, and thus should be taken with significant precaution and placed in the context of every particular ontology. For example, the tool advised us to create “some datatype properties” just because we did not have any. In our case, as we argued in Chapter 5.2.5, these properties are not necessary and by missing them the ontology does not lose any quality. On the other hand, the advice on possible duplication of concepts was very welcomed.

Finally, in order to validate the ontology against its usage in the future application, we created a series of DL queries which aimed to extract direct and indirect knowledge out of the ontology, by using a reasoning engine. The results obtained by these queries have been validated by the supervisors of this thesis, and one of them (prof. Vjeran Strahonja) is a domain expert in the field of software engineering methodologies.

The following sections present several queries executed upon our ontology with their Description Logic representation and the finally obtained results.

➢ *What platform specific artifacts are classified as reusable?*

In order to get the artifacts that are platform specific we can create several different queries that would be based on different concepts already built into the ontology. Thus, we can use only basic classes like Artifact and their properties, or we might use already defined named queries which would, in our case, be logically connected sub queries.

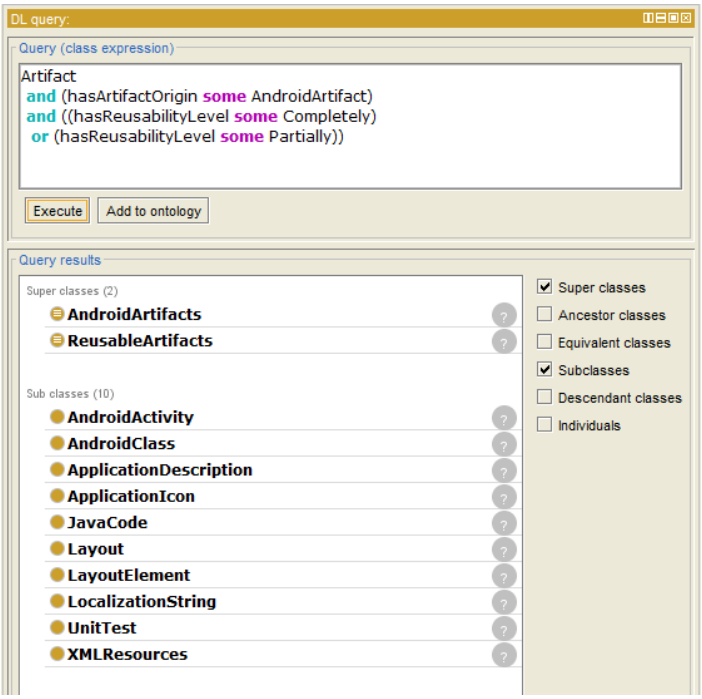


Figure 52 - Example of defined and executed DL query with reasoning results

DL query obtaining only Android reusable artifacts could look like this:

```
Artifact
 and (hasArtifactOrigin some AndroidArtifact)
 and ((hasReusabilityLevel some Completely)
   or (hasReusabilityLevel some Partially))
```

Code 9 - Android reusable artifacts

If using already defined concepts which classify all *Android* and reusable artifacts, we can use this query:

```
Artifact and (AndroidArtifacts) and (ReusableArtifacts)
```

Code 10 - Android reusable artifacts with already defined named queries

In both cases, the result is the same and it contains the following enumerated artifacts (see Figure 52).

```
AndroidActivity, AndroidClass, ApplicationDescription, ApplicationIcon,
JavaCode, Layout, LayoutElement, LocalizationString, UnitTest, XMLResources
```

In similar manner, we could ask for *Windows Phone* artifacts only or for reusable artifacts that originate from *Mobile-D* methodology et cetera.

- *What artifacts can be reused in any given development activity or task?*

For example, in order to obtain all reusable artifacts that were used, created or updated during the *Iteration Planning* task we can use a query like this:

```
Artifact
 and ((isUsedByTask some IterationPlanningTask)
   or (isCreatedByTask some IterationPlanningTask)
   or (isUpdatedByTask some IterationPlanningTask))
 and (ReusableArtifacts)
```

Code 11 - Reusable artifacts by task

The query result:

```
AcceptanceTest, IterationBacklog, IterationsPlan, MeasurementPlan,
ProductBacklog, ProjectPlan, ProjectPlanChecklist, ProjectPlanGantChart,
StoryCard, StoryCardTemplate, TaskCard, TaskCardTemplate
```

On the other hand, if we want to enumerate all artifacts that are an output of any task performed during the *Working Day* activity we can use a query like this:

```
Artifact
 and ((isUpdatedByTask some WorkingDayTasks)
   or (isCreatedByTask some WorkingDayTasks))
 and (ReusableArtifacts)
```

Code 12 - Reusable artifacts by activity

The query result:

```
AppResource, ClassModelWeb, DataModelMobile, DataModelWeb, IterationBacklog,
IterationsPlan, MeasurementPlan, PHPCode, ProductBacklog,
ProjectPlanChecklist, SourceCode, StoryCard, TaskCard, UMLClass, UnitTest,
View, ViewController, ViewElement, WebService, WebServiceSpecification
```

- *What artifacts can be reused in any given development phase?*

The following query results in a set of artifacts that are reusable and created, updated or used in *Explore phase*. The artifacts were additionally filtered with their origin in order to exclude *Other Artifacts* that are not connected to development methodology or target platform.

```
Artifact
 and((isCreatedByTask some (isPerformedIn some (isPerformedIn some Explore)))
  or (isUpdatedByTask some (isPerformedIn some (isPerformedIn some Explore)))
  or (isUsedByTask some (isPerformedIn some (isPerformedIn some Explore))))
and (ReusableArtifacts)
and (not (OtherArtifacts))
```

Code 13 - Reusable artifacts by phase and origin filter

The query result:

```
InitialRequirementsDocument, MeasurementPlan, ProductProposal, ProjectPlan,
ProjectPlanChecklist, ProjectPlanChecklistTemplate, ProjectPlanGanttChart
```

In the above example we used nested queries to reach all artifact that are created by some *Task* that was performed in some *Activity* performed in some *Phase*. Another approach in ontological modeling of such problems can be the usage of transitive properties.

- *What artifacts are reusable in accordance with their type or origin?*

The following query enumerates artifacts with specific type of *Document* that are *completely* or *partially reusable*.

```
Artifact
 and (hasArtifactType some Document)
 and ((hasReusabilityLevel some Completely)
  or (hasReusabilityLevel some Partially))
```

Code 14 - Reusable artifacts by their type

The query result:

```
InitialRequirementsDocument, MobileDProcessLibrary, ProductBacklog,
ProductProposal, ProjectPlan
```

The artifacts that are completely or partially reusable are recognized as sub class of *Reusable Artifacts*, which we used in other examples presented previously.

Except the queries that answer our competency questions stated at the beginning of the ontology development process, by using the built vocabulary of classes, properties, value partitions and named queries, we can build any other query in order to obtain other specific knowledge encoded in the ontology. These queries can be specific focusing on any particular artifact, or general and focus on groups of artifacts.

For example, the following query asks for any reusable artifact that is used in creation of *Software Architecture and Design Description Document*.

```
Artifact
 and (isPartOfArtifact some SADDDocument)
 and (ReusableArtifacts)
```

Code 15 - Reusable artifacts used in specific document

The query result:

```
AppDescription, ArchitectureLinePlan, ClassModelWeb, DataModelMobile,
DataModelWeb, WebServiceSpecification
```

By answering all competency questions defined at the beginning of our ontology development process, we proved the completeness of the created ontology. As presented in previous examples, DL queries are flexible and the ontology is capable of answering a wide range of questions regarding any concept that is used in its creation. Additionally, queries and results were observed by domain experts who finally validated the ontology and agreed on its completeness.

Such an ontology represents a solid basis for creation of information system that can guide the development team or development teams in achieving methodological interoperability by reusing artifacts created in multi-platform mobile application development process.

5.4.6. Final remarks on proposed ontology for methodological interoperability

The development process of development of an ontology for methodological interoperability, namely *Multi-platform Case Artifacts Ontology*, was performed in two phases. First, we created two specific ontologies targeting Android and Windows Phone application development and secondly, we merged these two ontologies into a new ontology which we enhanced with multi-platform and reusability conceptualization.

The created ontology comprises 213 classes, 14 object properties and 2213 axioms defined in ALCRIF DL expression sub-language. Generated in *Manchester OWL Syntax* format it can be found in Appendix E of this document. Also, the ontology in native OWL/XML format can be downloaded from http://barok.foi.hr/~zstapic/ont/mcao.owl, while full OWLDoc ontology documentation can be accessed and analyzed at http://barok.foi.hr/~zstapic/ont/mcao/doc/.

The whole development process was guided by Ontology Development 101 methodology and recommendations in ontology development given by Noy and McGuinness (2001) and Horridge (2011). We also put special focus in reusing the existing knowledge while building the second and the third (i.e. the final) ontology, and the proof of the ontology's quality was the possibility of reusing the Android ontology without the need to change any infrastructural elements while building a Windows Phone ontology. Additionally, after merging the two ontologies, we had no redundancy to deal with, and had no problems in updating the ontology with a new conceptualization. This proves that the ontology is both reusable and extendable.

A special focus was put on the ontology evaluation through its development and final testing. We used seven evaluation mechanisms, and as the most important one, we tested the ontology with series of Description Logic queries which asked different questions including all competency questions stated at the beginning of the ontology development. The results were then analyzed by the two thesis supervisors, and one of them is a domain expert. The use of evaluation mechanisms throughout the development process and positive validation are the proof of ontology's quality and completeness.

This brings us to the final conclusion that developed *Multi-platform Case Artifacts Ontology* represents a knowledge base that can be used in development of information system aiming to guide development teams in achieving methodological interoperability by reusing artifacts created in the process of multi-platform mobile application development.

5.5. Relevance of the chapter

This chapter presented the results and the approach taken in our last research phase – *Ontology Development Phase*.

As development of ontologies is not a trivial task, first we introduced the concepts of the ontologies by looking into the origins of the term in Philosophy, and then by defining it in Computer Science. Finally we agreed to use the definition of ontology saying that ontology is *an explicit formal conceptualization of a shared understanding of the domain of interest which includes the vocabulary of terms in order to describe the domain elements, semantics in order to define the relationships of the domain elements and pragmatics in order to define possible usages of these elements.*

After discussing different types of ontologies, their possible usages and presenting in detail several the most commonly used and the most important ontology development methodologies, tools and languages, we decided to create a domain ontology in order to semantically describe concepts belonging to the domain of development of mobile application for specific target platforms. Additionally, we argued the reasons for using the Noy and McGuiness' Ontology Development 101 methodology, as the best option suitable in our case, and finally, we decided to use Protégé ontology development tool and OWL2 DL as the most appropriate ontology language in our case.

The chapter also presents in detail the usage of Ontology Development 101 methodology while developing *Android Case Artifacts Ontology*. We have put focus on reusability when developing *WindowsPhone Case Artifacts Ontology*, and finally, on ontology merging, updating and evaluation when developing *Multi-platform Case Artifacts Ontology*.

The results showed that our ontologies are reusable, extensible and updatable as we performed all these tasks without the need of changing any existing infrastructural elements. The final ontology is additionally verified and validated with several automatic and manual evaluation mechanisms including the validation by domain experts who analyzed the results of the executed DL queries. The validation results showed that ontology is valid and complete and thus can be used in future development of an information system that would help

development teams to achieve methodological interoperability by reusing the artifacts created in the process of multi-platform mobile application development.

This concludes our three phase research process which resulted in (1) Systematic Literature Review performed in order to identify and choose a mobile development methodology applicable in multi-platform development, (2) the implementation of a prototype application by utilizing the selected methodology performed in order to identify all artifacts that arose in the development process, and (3) ontology development in order to ontologically describe the empirical and theoretical knowledge and thus make it usable for future development of information systems targeting the increase of methodological interoperability in the development of mobile application for multiple platforms.

6. DISCUSSION OF RESULTS

This multidisciplinary research composed of systematic literature review, analysis of artifacts created in methodologically driven mobile application development, and development of an ontological description of artifacts reusability is presented in the previous chapters. Through every research phase we gave an overview and analysis of the existing body of knowledge, performed a research and reported on the results that were obtained in it.

In this chapter we would like to review, assess and recapitulate the results that were produced during the presented research process. This discussion includes review of the results on performing Systematic Literature Review in the field of software engineering with special focus on the aspects regarding the execution of this method by doctoral students. Additionally, we discuss the identified development methodologies and approaches with special focus on multi-platform development. The artifacts that arise in the development process targeting multiple-platforms are identified during the second phase of the research as a result of performed two development cases. These artifacts are analyzed and finally ontologically described in the last research phase.

All these results are argued and assessed in this chapter where we put special focus on the research *motivation*, *results*, *contributions*, *rigor* and *evaluation.* By research *motivation* we would like to emphasize the reasons for performing the research activities. By *results* and *contribution* we aim to systematize the obtained results and the contribution to knowledge. Discussing the *research rigor* we would like to point out our approach and its main characteristics, and discussing the *evaluation* we would like to underline the evaluation mechanisms that are used in order to verify and validate the used approach and the obtained results.

Finally, we encompass the discussion with evidence on testing the stated research hypothesis.

6.1. Methodologies for development of mobile applications

In Chapter 2.1 of this thesis we gave a detailed analysis of Systematic Literature Review (SLR) methodology as it is proposed by Kitchenham and Charters (2007). We presented the methodology and gave summary of all phases and activities that should be performed while

conducting the SLR in the field of software engineering. Later, in Chapters 2.2 and 2.3 we reported our literature review on methodologies for development of mobile applications.

In this chapter we would like to emphasize several characteristics of this research phase, with the focus on mentioned views: motivation, results, contributions, rigor and evaluation.

6.1.1. Performing systematic literature review in SE

Motivation: The method of SLR is a well-known method of assessing and summarizing the existing body of knowledge on a particular research question or questions. Although the origins of SLR can be traced back to the beginning of the 20th century, it emerged in the field of software engineering (SE) during last several years. As there are important differences in performing the SLR in SE and performing it in other fields, the authors who performed the method generally agree that this field is still an area of investigation that remains to be explored and that could well bring many benefits (Biolchini et al., 2005). The guidelines presented by Kitchenham and Charters (2007) are created by adaptation of several existing guidelines from other disciplines, mainly medicine, and thus are partially inappropriate for the field of SE. Several authors, including Biolchini et al. (2005), Mian et al. (2005) and Staples and Niazi, (2007) criticized the mentioned guidelines as explained above. As the methodology of SLR as described in the guidelines is comprehensive, but time consuming, risky and inappropriate for conduction by a single researcher, we decided to perform the analysis of the reports and recommendations given by other authors and to enhance the guidelines in this manner. Specifically, we focused on possible approaches that could be taken by PhD students in order to overcome the most important obstacles they usually run on during the execution of this method.

Results: As presented in Chapter 2.1, three phases of SLR are discussed in detail and recommendations from other authors are given. In the *review planning phase*, the most important tasks are concerned with specification of research questions and development of review protocol. PhD students will usually define such research questions that aim to identify the scope of future research activities. Additionally, PhD students will usually break-down the research question into sub-questions by utilizing the PICOC model, i.e. defining the population, intervention, comparison, outcomes and context. On the other hand, the development of review protocol is not a trivial task, which according to some authors (e.g. Staples and Niazi (2007)) is a subject of constant changes throughout the whole SLR process. In this context, we found the template proposed by Biolchini et al. (2005) as an important artifact which defines structure of the protocol along with the explanation of its elements. Some protocol elements should be defined upon execution of pilot studies, and thus this task can be time consuming. Subsequently, evaluation of review protocol is a key activity that

should be done by field experts or in the case of doctoral students, at least by thesis supervisors. Other often used evaluation method is test of protocol execution.

In the *conducting the review phase* predefined protocol should be followed. This is the most time-consuming phase which ends up with data extracted, summarized and ready for dissemination. PhD students should use appropriate tools like appropriate reference manager software in order to keep record on all of the identified studies through all review phases. One of the key quality criteria is the transparency and the replicability of the review. In order to identify relevant studies, doctoral students should strictly use predefined *inclusion* and *exclusion* criteria, and lists of relevant sources for the field of software engineering can be adopted from other authors, like for example from (Hannay et al., 2007) or (Kitchenham and Charters, 2007). Depending on the number of initially obtained studies, different approaches can be taken in their filtering. Less strict approach would be to, in the first step, exclude some studies only by reading their title. This is sometimes the only approach as the number of initial studies could be more than 10.000. On the other hand, Brereton et al. (2007) advocate a more strict approach where exclusion by title should be avoided and used only if exclusion is obvious. Reliability of inclusion and exclusion decisions is important, and doctoral students can use several methods to increase it. Consultations with the advisor, the expert panel or other researchers, re-evaluation of a random sample of the primary studies by test-retest approach or re-evaluation by other researcher are some of the methods recommended for PhD students. The study quality assessment procedures mainly depend on the type of the study, but one method is particularly often used in SE – the use of *checklists* with defined quality criteria. Finally, data extraction and synthesis are the last activities of this phase. The most usual approach in data extraction is the usage of *extraction forms*. Examples of extraction form can be found in (Kitchenham and Charters, 2007) and (Jørgensen, 2007), or in Table 4 of this document.

The mentioned data synthesis can be qualitative and quantitative, but in both cases, results presented in an appropriate (e.g. table, graph or figure) manner should be narratively explained. Doctoral students will probably report their findings in their dissertation, but prior to that, proper evaluation of the results should be carried out. In this evaluation, help from a supervisor, prior to submitting the dissertation to be evaluated by committee is welcomed. On the other hand, the evaluation of scientific papers is done by scientific peer review.

Contributions: The body of knowledge on performing the systematic literature review in the field of software engineering as proposed by Kitchenham and Charters (2007) is presented and enhanced with a discussion, observation and recommendations synthetized from other influential authors in the field. The three-phase-process along with stages and tasks is analyzed in detail, and special focus is put on making the execution of this comprehensive

method easier for single researchers, like PhD students. Enhanced guidelines that can be used while performing the systematic literature review are the main result of this research activity.

Rigor: A comprehensive analysis of available papers on how to perform SLR in the field of SE was performed. The results showed that one document, the guidelines from Kitchenham from (2004) which were updated by Kitchenham and Charters in (2007) is used as the knowledge base on how to perform the method in all other reported reviews. However, we carefully analyzed and compared the mentioned document with the reports and recommendations from other influencing authors in the field. Each recommendation given in our report has theoretical or practical proof that is found in the cited literature.

Evaluation: A short paper on the results presented in this chapter is already published at the Central European Conference on Information and Intelligent Systems (Stapić et al., 2012), while the full paper is currently under the review. Additionally, the presented enhanced guidelines were evaluated by the thesis supervisors and were used in the SLR process performed in this research.

6.1.2. Mobile development methodologies and approaches: SLR

Motivation: In Chapter 2.2, we defined the basic concepts that are connected with the software development methodologies, and also we gave an overview of methodologies targeting the development of mobile applications and concluded that it differs from the standard development, that the agile approach is widely used in methodologies for mobile devices and that all presented methodologies should be more fine grained and suitable for specific development environment. Thus, even there are some attempts to create a specific software development methodology that would be suitable for development of mobile applications, these attempts are relatively rare and they are not aligned with the current mobile development demands. So, many companies choose to use the existing and familiar development methodologies in while developing mobile applications. These methodologies are often adapted and changed, and a proper analysis of all of these possibilities was needed. We also performed a research in order to identify the existing SLR from the domain of interest and found that there are no existing SLRs targeting mobile application development methodologies, which makes the need for such a review even bigger.

Results: In our systematic literature review, we aimed to answer two research questions. First, we wanted to know what development methodologies and approaches are reported in literature as defined in theory or used in practice for mobile application development, and second, we aimed to analyze if these methodologies and approaches are applicable for multi-platform mobile applications development. After having the review protocol developed and validated by the thesis supervisors, we performed automatic and manual search on the

selected sources and obtained 6761 initial studies which were then analyzed through several phases by applying strictly defined exclusion and inclusion criteria (see Table 14). The review resulted in 49 relevant studies that were analyzed and data extraction was performed on them. We finally identified 22 development methodologies and 7 development approaches that can be used in the development of mobile applications (see Table 17 and Table 18). On the other hand, only one methodology was not eligible according to the second research question, as it targeted specific platform capabilities. After analyzing the obtained results and comparing the reported use and available documentation on identified methodologies, Mobile-D methodology along with Test Driven Development emerged as the most suitable (although still not fully applicable without changes) methodology-approach pair to be used in the following research phases.

Contributions: During the time of writing this thesis and to our knowledge, there are no Systematic Literature Reviews performed in the field of Software Engineering that assess the software development methodologies in general or specifically for mobile applications development. Thus, in our research we performed SLR in order to identify development methodologies and approaches that are reported to be used in mobile applications development. Specific focus is placed on the assessment of included studies quality. Although the average study quality is not very good, the results showed that 22 methodologies and 7 approaches are reported to be used in development of mobile applications.

Rigor: The method of Systematic Literature Review is performed by consistently following the guidelines which are usually used in the process of SLR implementation in the field of Software engineering. The mentioned guidelines are additionally enforced with the recommendations from influential authors in the field. Every step is taken upon strictly defined and evaluated procedure and with explicitly defined criteria. Where applicable, references to the theoretical or practical background of all used artifacts are provided. All included studies have undergone quality assessment, which resulted in elimination of 18 (out of 67) studies. The remaining 49 studies were analyzed and data was extracted in accordance with template specifically developed to provide sufficient information regarding the research questions. All performed activities along with the results are reported as requested by SLR methodology.

Evaluation: The SLR process is by its nature sequential. But, having the evaluation procedures at every milestone, it can be considered as iterative process as well. By following the methodology requirements and systematized recommendations from other authors, strict evaluation mechanisms were applied at this research phase. Thus, the review questions, created protocol, created search string, selected sources and other elements were evaluated during or at the end of the planning phase. During the execution phase, the inclusion and

exclusion criteria were applied by the main researcher and then evaluated either by test-retest method or by evaluation of the results by the research supervisors. Finally, the report results were again evaluated in accordance with dissemination mechanisms and media.

6.2. Mobile-D implementation

The first part of the second research phase is presented in Chapter 3. First we gave an overview of the chosen development methodology and then we utilized the methodology in two development cases. The results contain documented development process for two target platforms with the focus on the used and created artifacts. These results were used in subsequent research phases.

Motivation: The identification of artifacts that arise in the development process for two or more target platforms could be done either by analyzing some existing data on development processes performed in practice (e.g. in company, or by individual developers), or by performing a development in a laboratory environment. Although both approaches have their advantages and disadvantages, we had to choose the second option, as it proves to be more flexible and fully controlled by the researcher himself. The development of fully functional application for two target platforms is a time consuming work, but it brings the benefits of executing the process with careful analysis of all performed phases, activities and tasks along with all artifacts that were created in the process.

Results: After almost 160 working days, two versions of the same application were created. During the development process we put special focus on the artifacts that were created in the process and on their reusability. Specifically, while developing the mobile application for the first target platform, the artifacts were observed from methodological point of view. The methodological approach along with the connected artifacts was reported in detail. On the other hand, while developing for the second target platform, artifacts were observed from the reusability point of view. Although we had some implementation problems which made some phases in the second development case unexpectedly long, the reusability at methodological level resulted in a development process shortened for 18.4% (see Table 37). If we remove the technology related issues, the time saved with this approach would be even bigger.

Contributions: The performed process faithfully demonstrates the development process that would be performed by any small company. The finished product with all planned functionality implemented and tested is a proof of completeness of our approach. The empirical evidence collected during such development represents valuable scientific knowledge base which we used in the rest of this research and which can be used for different additional analyzes in the future.

Rigor: The Mobile-D methodology as described in (Abrahamsson et al., 2005a) was strictly followed in both development cases. All activities were carefully noted and the development process is made transparent and reported in this document.

Evaluation: Test Driven Development represents continuous evaluation of created project throughout the whole development process. This product evaluation includes execution of unit tests on units of code, integration tests on integrated components of the system, system test on final product and acceptance test on required functionality. On the other hand, the alignment of development process with Mobile-D methodology is evaluated according to methodology implementation instructions given in (Abrahamsson et al., 2005a). The final evaluation was performed by the thesis supervisors who are experts in the field of software engineering and development.

6.3. Identification of artifacts

Chapter 4 represents the second part of second research phase where we analyze and compare the artifacts that arose in methodologically driven process mobile application development for two target platforms. This chapter uses the empirical evidence and created artifacts collected during the implementation phase and identifies Android development case artifacts and Windows Phone development case artifacts, and the analysis shows a great level of reusability.

Motivation: We consider Mobile-D as being a well-documented methodology for development of mobile applications. We used several documents describing the methodology, but the most important one is definitely a guide presented in (Abrahamsson et al., 2005a) which in detail describes the whole development process, and it also enumerates all artifacts that arise in such methodologically driven process. However, the overall picture on the use of these artifacts by phases and tasks is hard to read from the mentioned document. Additionally, these are not the only artifacts that we are interested in. From the point of view taken in this research, platform specific artifacts and development unrelated artifacts could also be reusable in different ways and on different reusability levels. Thus, comprehensive analysis of all artifacts that arise in such development process is needed. Additionally, once identified, such artifacts should be analyzed, compared, cross-platform compared and connected to the development phases, activities and tasks.

Results: In order to perform straight format and unbiased analysis, first we defined the analysis setting (see Chapter 4.1) which includes the definition of artifacts, the relations with other methodological concepts that will be observed and the template that is to be used for artifact description. As the artifacts are observed as "*any piece of software developed and*

used during software development and maintenance" we found the list of Mobile-D artifacts (see Table 38) related to process tasks not sufficient and thus we performed our own analysis. During the analysis, we observed the development process for each target platform separately and we identified 71 different artifacts that we initially grouped in 12 categories (see chapters 4.2 and 4.3). After performing a cross-platform analysis we found that more than 70% of all identified artifacts are common to both platforms and 66% of them are partially or completely reusable (see chapter 4.4).

Contributions: Our analysis included artifacts that originate from the selected methodology, from the specific target platform or are necessary as supportive in performing other development unrelated tasks like communicating, reporting or project management. Another important contribution of this research phase are the results of cross-platform analysis showing high level of reusability among artifacts created during the development for two target platforms. These results are very encouraging and we can conclude that they create a strong basis and motivation for additional research and analyses.

Rigor: This research phase is performed by a careful analysis of empirical evidence collected during the research process and by systematic analysis of the Mobile-D documentation. In cross-platform analysis, three levels of reusability were created and all artifacts were evaluated according to the same criteria in order to be placed in '*completely*', '*partially*' or '*none*' level.

Evaluation: Three different evaluation mechanisms were used in this phase. First, we compared our matrixes showing the Android and Windows Phone artifacts (Table 40 and Table 42) with the Mobile-D artifacts matrix (Table 38). Although not all artifacts are present in both matrixes, we could evaluate our results at least for methodological artifacts. Secondly, the cross-platform analysis results were compared against the development notes that had been created during the implementation process. In the end, as usual, the results were additionally evaluated by the two thesis supervisors.

6.4. Ontology for methodological interoperability

The last research phase is presented in Chapter 5. This chapter presents a background for ontology development (see chapter 5.1) by defining: the ontology, its types and usages, connections with our proposed methodological interoperability, ontology development methodologies, tools and languages. Having the background established first we developed *Android Case Artifacts Ontology* (see 5.2), then we reused it in the development of *WindowsPhone Case Artifacts Ontology* (see 5.3), and finally we merged these two ontologies

and enhanced the resulting ontology into *Multiplatform Case Artifacts Ontology* which focuses on artifacts reusability (see 5.4).

Motivation: The main goal of this research was to ontologically describe artifacts that arise in a methodologically managed process of mobile application development targeting two or more mobile platforms, and to create the basis for more efficient and interoperable process of multi-platform mobile applications development. As we argued in the Chapter 1.1 of this thesis, the development for mobile devices brings different new challenges, and although there are several rather different approaches that scientists and experts are taking to solve these problems, their common characteristic is also their main disadvantage: all of them are based on paradigm "code once – run anywhere" which is unachievable and which takes away a native development environment possibilities. This motivated us to propose a novel approach by enhancing the interoperability among teams working on the same application targeting different platforms by moving the focus to the methodological interoperability that would be achieved through the reuse of artifacts created in such process. Having this in mind, and as described in Chapter 5.1.3, ontologies are a natural solution and tool in achieving semantic interoperability.

Results: First, we tried to give a short overview of several concepts that are related to ontologies and ontology development (see Chapter 5.1). For the purpose of this research we defined ontology as *an explicit formal conceptualization of a shared understanding of the domain of interest which includes the vocabulary of terms in order to describe the domain elements, semantics in order to define the relationships of the domain elements and pragmatics in order to define possible usages of these elements*. We also presented the most common reasons for ontology usages and we argued about their classification in accordance with different points of view. As important result of this research, we created a connection between ontologies and methodological interoperability that is proposed by this thesis. Additionally, we gave a short overview of several influencing ontology development methodologies which are either commonly used today or had a great influence on the development of other methodologies. Finally, we argued about the possibilities of using different ontology development tools and ontology development languages.

By implementation of Ontology Development 101 methodology (Noy and McGuinness, 2001) we created two platform specific ontologies and one upper level common ontology for multi-platform development. The development of the first ontology was performed from scratch and the focus in the report presented in Chapter 5.2 was put on the ontology development process. During the development of the second platform specific ontology (see Chapter 5.3), we focused on the reusability and ontology update. The results showed that no infrastructural changes on the existing ontology were necessary while converting it into an

ontology targeting a different platform. On the other hand, the development of the final ontology targeting multi-platform development and reusability focused on the ontology merging, enhancing, evaluating and testing concepts. Two existing ontologies were merged and again there was no need for any infrastructural changes or conflict resolutions (see Chapter 5.4.2). The merged ontology was finally enhanced with a conceptualization regarding the artifacts reusability.

The final ontological description encodes the information on 213 classes (see Table 51), 14 properties (see Table 52) and 2213 axioms. A full ontological description is available in Appendix E of this document, in OWL/XML format at http://barok.foi.hr/~zstapic/ont/mcao.owl and as OWLDoc documentation at http://barok.foi.hr/~zstapic/ont/mcao/doc/.

Special focus in this chapter is placed on ontology testing and evaluation. The ontology is tested with series of Description Logic queries which aimed to answer all competency questions stated at the beginning of development process. More on testing and evaluation is given in subsequent *evaluation* paragraph.

Contributions: This chapter contributes to knowledge in several aspects. First we presented the most important concepts in ontology development. Although these are not new concepts, **the use of ontologies in achieving a methodological interoperability is a novel approach in solving the mobile platform and device fragmentation problem**. Additionally, we argued about the use of specific methodologies, tools, languages and approaches in ontology development. Such discussion along with the in detail presented ontology development process that was taken in this research, could be useful in future ontology development projects. **The *Multi-platform Case Artifacts Ontology* represents a unique ontological description which is created to be a knowledge base for any information system that aims to help development teams in increasing interoperability at methodological level by reusing the artifacts that arise in multi-platform development process.**

Rigor: In the development of all three ontologies we followed the OD101 methodology (Noy and McGuinness, 2001), the recommendations given in (Horridge, 2011) and middle-out approach in class development as proposed by Uschold and Gruninger (1996). Additionally, during the whole development we kept the ontology in consistent logical, syntactic and semantic state by performing the continuous evaluation by several mechanisms presented in the following paragraph. As it can be seen from the obtained results, the created ontologies are flexible, reusable and updatable.

Evaluation: As created ontology is one of the main contributions of this thesis, the special focus was put in its verification and validation throughout the whole development process (see Chapter 5.4.5). We used seven different automatic and manual evaluation mechanisms

that aimed to verify that the ontology was built correctly and to validate its content quality and completeness.

First, the ontologies were built by following methodological development process which ensured that our approach was systematic. We also followed the recommendations given in (Noy and McGuinness, 2001) and in (Horridge, 2011) in order to avoid mistakes that are often made and to solve the most common issues in ontology development. Third, throughout the whole incremental development process we used reasoning tools to verify the newly added concepts and their influence on the already defined concepts. Reasoners detect any syntax errors, check the consistency of the ontology and automatically compute the inferred class hierarchy model and as such are strong evaluation tool.

Additionally, we used two different tools that automatically evaluate the created ontologies: *OWL Validator* (Horridge, 2009) which formally validated the ontology syntax and *Ontology evaluation plugin* (Tantsis, 2013) which automatically evaluated the ontology according to eight properties and gave us some insights and recommendations in possible inconsistencies in the created ontology.

As the sixth and probably the most important evaluation mechanism, in order to validate the ontology against its usage in future application, we created a series of Description Logic queries which aimed to extract direct and indirect knowledge out of the ontology, by using a reasoning engine. We created queries to test the ontology against all competency questions that were created at the beginning of ontology development process and that were used as a ontology guiding thread. The results obtained by these queries have been validated by thesis supervisors - prof. Strahonja who is a domain expert in the field of Software Engineering Methodologies, and dr. de-Marcos as an expert in Artificial Intelligence.

The created ontology is successfully verified and positively validated, and as such it represents a solid basis for creation of an appropriate information system.

6.5. Summary of the results

Taking into consideration all what was said in the previous chapters we can conclude that the research process was performed in the planned scope and within the planned research framework defined at the beginning of the research process (see Chapter 1.3).

Following this framework we identified the methodologies that could be used for development of mobile applications; we implemented the chosen methodology and approach and created a mobile application targeting two target platforms; we identified and analyzed the artifacts that were created in this development process, and we created the ontological

definition that describes the artifacts in accordance with Mobile-D methodology and from the reusability point of view.

According to the results that were obtained during the ontology evaluation and testing, we can conclude that such ontological description, that encodes the knowledge with OWL 2 and Description Logic defined axioms and queries, represents a solid basis that can be used in development of information system aiming to guide the development teams in achieving the methodological interoperability by reusing artifacts created in the process of multi-platform mobile application development. Additionally, we proved that our ontological description is highly flexible and extensible, which allows us to update it with information on new platform specific or platform independent artifacts without the need of changing the underling infrastructure defined by the main class hierarchy elements, defined value partitions or properties. Finally, the model allows the creation of Description Logic queries which can be used to acquire direct or indirect information encoded in ontology knowledge. We showed the examples of such queries which among other aimed to reach the information regarding the competency questions stated at the beginning of the ontology development.

Therefore, we can conclude that **it is possible to create ontological description of elements of methodological interoperability containing structural and semantic aspects of sets of artifacts created in the development process of a mobile application for two or more target platforms**, which makes our H_1 hypothesis confirmed.

This opens different possibilities for further research in this field – starting from building additional ontological descriptions, building the different information systems that would utilize such knowledge, designing and creating the integrated systems that would not only guide the developers, but also provide them with interoperable artifacts management environment.

7. CONCLUSION

7.1. Research objectives revisited

As we described in the introductory chapters of this thesis, this research focuses on the analysis of the problem of multi-platform mobile applications development, and on the proposal of a novel solution in the domain of ontology-based methodological interoperability.

Thus the stated goals included the acquisition of answers to the following questions: (1) what methodologies and development approaches can be used in multi-platform mobile applications development; (2) what artifacts (required inputs and outputs of methodologically and methodically defined development steps) emerge during mobile applications development, (3) whether and to what extent there are similarities between these artifacts, (4) whether it is possible to ontologically describe these artifacts, and create a basis for developing a system that would support the methodological interoperability.

Thus, the main goal of the research is connected to the last stated question, and it was to ontologically describe artifacts that arise in the methodologically managed process of mobile application development targeting two or more mobile platforms, and to create the basis for a more efficient and interoperable process of multi-platform mobile applications development.

In this chapter we would like to have a glance look back on the performed research and to emphasize its results by answering the stated questions and by aligning the results with the main goal of this research.

- *What methodologies and development approaches can be used in multi-platform mobile applications development?*

After creating a comprehensive analysis of how to perform a Systematic Literature Review in the field of Software Engineering (Chapter 2.1) we performed an SLR with the goal to answer the stated research question (Chapters 2.2 and 2.3). Reviewing more than 6700 initially obtained studies through a set of predefined phases, we identified a total of 49 studies that are found to be relevant to our question. Finally, we identified 22 development methodologies and 7 development approaches that can be used in multi-platform mobile applications development (see Table 17 and Table 18).

- *What artifacts (required inputs and outputs of methodologically and methodically defined development steps) emerge during mobile applications development?*

Out of 22 identified methodologies, we argued and choose Mobile-D methodology to be the most suitable for development of our mobile application for two target platforms (see Chapter 2.4). In the next research phase, we performed the development in order to identify the artifacts that arise in such development process (see Chapters 3 and 4). After analyzing the empirical and theoretical evidence we identified a total of 71 artifacts (60 in Android case and 61 in WP case) that were used or created in the mentioned development process. The artifacts are enumerated and described in Table 40 and Table 42.

- *Whether and to what extent are there similarities between these artifacts?*

The cross-platform analysis of the identified artifacts showed significant similarities between the artifacts used in the two development cases (see Chapter 4.4). After performing a cross-platform analysis we found that more than 70% of all identified artifacts are common to both development cases, that 66% of these common artifacts are completely or partially reusable, and that the remaining platform specific artifacts also have some similarities.

- *Whether it is possible to ontologically describe these artifacts, and create a basis for developing a system that would support the methodological interoperability*

Having the artifacts identified, we moved to the process of their ontological description. First we created an ontological description of artifacts targeting Android development (see Chapter 5.2), then we created an ontological description targeting Windows Phone development (see Chapter 5.3), and finally we merged these two in a common ontological description that is additionally enhanced with the conceptualization of artifacts reusability (see Chapter 5.4). The whole process of creation was methodologically driven and evaluated with several evaluation mechanisms (see Chapter 5.4.5) which proved its correctness, validity and completeness.

With all this, **we can conclude that we ontologically described the artifacts that arise in a methodologically managed process of mobile application development targeting two or more mobile platforms. Having this ontology proved to be correct and valid, flexible, reusable and extensible we created the basis for development of an information system to guide the development teams in a more efficient and interoperable process of multi-platform mobile applications development, and thus the main research goal is accomplished**.

7.2. Limitations of the research

In this research several limitations can be identified. For example, the biggest challenge that we faced in the first research phase was the execution of a complicated and time-consuming scientific method of Systematic Literature Review by a single researcher. The SLR is originally created and defined to be performed by a team of researchers, and the execution by a single researcher (a doctoral student) makes the process of eliminating the research bias more complicated and, of course, very time-consuming. In order to deal with this limitation we defined very narrow research questions strictly focusing on the necessities of this thesis, and we tried to strictly follow the recommendations on performing the Light SLR that are given by the methodology creators and other influential authors. Finally, the role of the thesis supervisors in elimination of research bias was the most important as they evaluated the research results at every reached milestone.

The institutional subscriptions to the available scientific sources are very poor in Croatia and somewhat better in Spain. However, the restrictions on accessing several databases (including the newest volumes from Springer, some volumes from Wiley and the whole EI Compendex database) are also identified as limitations in this research. In the end, we believe that the lack of several sources did not significantly influence the overall literature review results as in some cases we contacted the authors of the studies who gladly sent us their findings. I would like to take this opportunity to thank all of them for this.

In the second research phase, the most important limitation was the lack of information on performed projects of development of mobile application in development companies that are targeting two or more target platforms. Our attempts to get such information for scientific purposes were politely refused and we had to turn to laboratory development environment in order to acquire empirical evidence that would be used in the later research phases. Although we performed a rigorous development process that was evaluated by several different mechanisms we find such approach as a possible limitation of this research. The main difference from the development process in a company is lack of organization hierarchy and roles, along with the lack of standard organizational processes that are intertwined with development processes. However, we had this in mind while defining the requirements of the mobile application and we tried to require the development of an application that would represent a vast majority of today's mobile applications developed by software companies. In this manner, we could talk about other differences that could be found when comparing a development performed by a single developer and development performed by a company that has a history, with its legacy systems, specific organization culture et cetera. Although, the development of a mobile application with or without a legacy system only influences the

development process and not the methodological aspects, we believe that other mentioned differences could be taken as additional limitation of this research.

Regarding the third research phase we are aware that the proposed ontology presents only the development of one application for two target platforms, and that the identified set of artifacts in general could include many other platform specific artifacts and even some methodology specific artifacts. Additionally, as stated in our scope definition (Chapter 1.3.1) we covered only one development methodology supported by one development approach and targeting for two mobile platforms. All mentioned issues can be recognized as the limitations of this research, but we have to keep in mind that this research process had the main goal of proposing a new framework or approach that can be used in solving the mobile platform fragmentation problem. As argued in the previous chapter, this goal is fully achieved.

In the next chapter we will elaborate on the possible future research directions that could be taken in order to overcome some of the above mentioned limitations or/and to enhance the framework and make it usable in a concrete information system.

7.3. Possible future research

This research presents a comprehensive set of activities which resulted in a final product that is usable in its current state. However, by extending the contexts of using such ontology we can identify other possible research activities or even research directions that could be taken.

Even though throughout the whole research, including the section on research limitations, we have pointed out the possible additional approaches that could be taken in order to enhance the results, or to take a different point of view in analyzing some concept of interest, in this chapter we would like to emphasize some of these possibilities.

In general, we recognize two main fields where this research sets the basis for future scientific and professional activities. Those fields are *Software Engineering* with particular focus on *mobile engineering* and, secondly, *Knowledge Engineering* with particular focus on *ontology development*.

Let us start with the second one. The created ontology defines the basic infrastructure and elements in the proposed framework of methodological interoperability, but currently it covers only one development methodology and one development approach and it targets two mobile platforms. As we have already discussed, the ontology is reusable and updatable but with limits on adding new artifacts targeting different mobile platforms. If we want to move to a completely new methodology, few of the existing classes would be reusable. Thus we think that some improvements in this sense could be achieved with different ontology

structure. Perhaps, building parts for the ontology should not be specific ontologies targeting specific platform, but rather distinct ontologies describing the methodology on one side and the target platform on the other side. This would raise the level of reusability and it definitely needs more scientific attention.

In addition to knowledge regarding the structural aspects of methodological phases, activities and tasks, structural aspects of the identified artifacts, semantic aspects regarding the origin, type, use and reuse of artifacts, only the inter-artifact relationships were described in the approach taken in this research. To get more fine grained results would include also an intra-artifact description describing its content in detail. Such analysis should answer questions like "*Which part of any partially reusable artifact could be reused and which does not?*" or "*How specific artifact is reusable: by its structure, content, inner logic or their combination?*" Having this information on artifact inner content, the proposed framework would have additional useful functionalities which would enable development teams to even better reuse existing outputs and to additionally reduce development time.

An interesting research activity could be to compare the existing methodologies for the development of mobile applications and to ontologically describe such acquired knowledge. Such ontological description could be used in creation of ontologies in our framework, but would also provide many different possibilities that are connected with mobile application development, like how to choose proper methodology in a specific context, or how to implement a new methodology that is unfamiliar to the team members.

On the other side, when talking about research activities in the field of software engineering, we have already mentioned the necessity of moving this research to a new phase where a proper information system for guiding the artifacts reuse would be developed. The development of such a novel system is not a trivial task and it gives many research possibilities in the domain of its design, functionality, relationship with the ontological knowledge base et cetera. We also mentioned other systems that could be developed and that are connected with artifacts management or even automatic transformation. Both these topics open a set of new research fields and possibilities.

Finally, there are different research activities that could be connected to the performed systematic literature review. As our research questions were rather narrow, similar review could be performed in order to identify the methodologies and compare their main activities, phases and tasks. Also, the data extraction forms, used in our research, contain some information that we currently did not need, but we extracted it as we presumed it would be useful for additional analysis. Such information, for example, relates to details on identified methodology, its organizational or project management aspects et cetera. The analysis of this information, along with the analysis of assessed studies quality could give new and interesting

results in this domain. As the SLR still emerges in the field of software engineering, an analysis of the performed researches along with recommendations and conclusions is also very welcomed.

In this short look into possible research directions in the future we presented only the most important research activities that could be performed, but as we have already said, many different and small enhancements of our research are possible and they are discussed throughout the dissertation text.

7.4. Conclusion

This doctoral research tried to propose a different approach in solving the mobile platform fragmentation problem with particular focus on multi-platform mobile application development. It is a multidisciplinary research positioned inside the intersection of *Software* and *Knowledge Engineering* fields. By utilizing ontologies, we proved that such formal specification of conceptualization represents a solid basis for the development of an information system that could guide development teams in a more efficient and methodologically interoperable process of multi-platform mobile application development by reusing the already created artifacts.

Three research phases were performed in order to identify the methodologies that are used for multi-platform mobile application development, to identify the artifacts that arise in such development process and to semantically describe those artifacts into a correct and valid ontological description. Thus, the overall scientific contributions of this research can be described as:

- Systematization of recommendations in performing the Systematic Literature Review process in the field of Software Engineering with special focus given to the execution of SLR by a single researcher (like doctoral students).
- Identification of available development methodologies and development approaches that are reported in literature as created or used for mobile applications development. The identification is performed by means of Systematic Literature Review.
- Systematization of knowledge and concepts in the field of application development for mobile devices, identifying artifacts created and used while developing for mobile devices with the consistent implementation of the selected development methodology.
- Classification of identified artifacts according to their reusability level, type and origin. This classification implies semantic description of the artifacts, description of

the connection between the artifact and development tasks, activities and phases along with description of inter-artifact relationships.

- A new ontological description of the artifacts that can be used as a knowledge basis for developing a system that would support methodological interoperability and therefore make development of applications for multiple mobile platforms more efficient.
- Guidelines and recommendations for improving the development of multi-platform applications for mobile devices through the utilization of an ontology-based framework proposed by this research.

Although there are ontologies defined to provide interoperability at different levels of an application development process, this novel approach aims to define interoperability at, until now unexplored, methodological level. Semantic descriptions created and evaluated in this thesis proved that the proposed approach and the supporting framework represent a solid basis for performing additional research in this field. However, developing this ontology is only the first step in the chain of activities to be implemented in order to develop a semantically supported system for methodological interoperability.

REFERENCES

Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., Koskela, J., Kyllönen, P., Salo, O., 2004. Mobile-D: an agile approach for mobile application development, in: Companion to the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages, and Applications, OOPSLA '04. ACM, New York, NY, USA, pp. 174–175.

Abrahamsson, P., Hanhineva, A., Hulkko, H., Jäälinoja, J., Komulainen, K., Korkala, M., Koskela, J., Kyllönen, P., Eporwei, O.T., 2005a. Agile Development of Embedded Systems: Mobile-D (Agile Deliverable No. D.2.3). ITEA.

Abrahamsson, P., Hanhineva, A., Jäälinoja, J., 2005b. Improving business agility through technical solutions: A case study on test-driven development in mobile software development, in: Business Agility and Information Technology Diffusion. Presented at the IFIP TC8 WG 8.6 International Working Conference.

Abrahamsson, P., Ihme, T., Kolehmainen, K., Kyllönen, P., Salo, O., 2009. Mobile-D for Mobile Software: How to Use Agile Approaches for the Efficient Development of Mobile Applications.

Abrahamsson, P., Warsta, J., Siponen, M.T., Ronkainen, J., 2003. New directions on agile methods: a comparative analysis. IEEE, pp. 244–254.

Adobe Corporation, 2011. Adobe Announces Agreement to Acquire Nitobi, Creator of PhoneGap [WWW Document]. Adobe.com - Press releases. URL http://www.adobe.com/aboutadobe/pressroom/pressreleases/201110/AdobeAcquiresNitobi.html (accessed 18-May-12).

Agarwal, V., Goyal, S., Mittal, S., Mukherjea, S., 2009. MobiVine: a middleware layer to handle fragmentation of platform interfaces for mobile applications, in: Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware, Middleware '09. Springer-Verlag New York, Inc., New York, NY, USA, pp. 24:1–24:10.

Ahtinen, A., Nurminen, J.K., Häkkilä, J., 2007. Developing a mobile reporting system for road maintenance: user research perspective, in: Proceedings of the 4th International Conference on Mobile Technology, Applications, and Systems and the 1st International Symposium on Computer Human Interaction in Mobile Technology, Mobility '07. ACM, New York, NY, USA, pp. 1–7.

Alyani, N., Shirzad, S., 2011. Learning to innovate in distributed mobile application development: Learning episodes from Tehran and London, in: 2011 Federated Conference on Computer Science and Information Systems (FedCSIS). Presented at the 2011 Federated Conference on Computer Science and Information Systems (FedCSIS). IEEE., Piscataway, NJ, USA, pp. 497–504.

Amanquah, N., Eporwei, O.T., 2009. Rapid application development for mobile terminals, in: 2nd International Conference on Adaptive Science & Technology (ICAST). Presented at the Technology (ICAST), Accra, Ghana, pp. 410–417.

Android Developers, 2013. Platform Versions [WWW Document]. Dashboards | Android Developers. URL http://developer.android.com/about/dashboards/index.html (accessed 3-Jul-13).

Avison, D.E., Fitzgerald, G., 1988. Information systems development: methodologies, techniques, and tools, Information systems series. Blackwell Scientific Publications, Oxford [England] ; Boston.

Avison, D.E., Fitzgerald, G., 2003. Where now for development methodologies? Communications of the ACM 46, 78–82.

Balagtas-Fernandez, F.T., Hussmann, H., 2008. Model-Driven Development of Mobile Applications, in: Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering, ASE '08. IEEE Computer Society, Washington, DC, USA, pp. 509–512.

Barnawi, A., Qureshi, M., Khan, A.I., 2012. A Framework for Next Generation Mobile and Wireless Networks Application Development using Hybrid Component Based Development Model. Arxiv preprint arXiv:1202.2515.

Beck, K., 2002. Test-driven development: by example, The Addison-Wesley signature series. Addison-Wesley, Boston.

Bektesevic, E., Rysa, E., 2008. JSR 248: Mobile Service Architecture [WWW Document]. The Java Community Process(SM) Program - JSRs: Java Specification Requests. URL http://jcp.org/en/jsr/detail?id=248 (accessed 17-May-12).

Belcar, T., Lovrenčić, S., 2012. Use of Description Logics Expressive Power in Ontologies, in: Proceedings of 23rd Central European Conference on Information and Intelligent Systems. Presented at the CECIIS 2012, Varaždin, pp. 23–28.

Bergström, F., Engvall, G., 2011. Development of handheld mobile applications for the public sector in Android and iOS using agile Kanban process tool.

Binsaleh, M., Hassan, S., 2011. Systems Development Methodology for Mobile Commerce Applications: Agile vs. Traditional. International Journal of Online Marketing (IJOM) 1, 33–47.

Biolchini, J., Gomes Mian, P., Candida Cruz Natali, A., Horta Travassos, G., 2005. Systematic Review in Software Engineering (Technical report No. RT - ES 679 / 05). PESC, Rio de Janeiro.

Biswas, A., Donaldson, T., Singh, J., Diamond, S., Gauthier, D., Longford, M., 2006. Assessment of mobile experience engine, the development toolkit for context aware mobile applications, in: Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, ACE '06. ACM, New York, NY, USA.

Bowen, J., Hinze, A., 2011. Supporting Mobile Application Development with Model-Driven Emulation. Electronic Communications of the EASST 45.

Brank, J., Grobelnik, M., Mladenić, D., 2005. A survey of ontology evaluation techniques, in: In In Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005.

Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M., 2007. Lessons from applying the systematic literature review process within the software engineering domain. Journal of Systems and Software 80, 571–583.

Brusilovsky, P., Sosnovsky, S., Yudelson, M., 2005. Ontology-based Framework for User Model Interoperability in Distributed Learning Environments, in: World Conference on ELearning, E-Learn 2005. AACE, pp. 2851–2855.

Centers for Medicare and Medicaid Services (CMS), Office of information Services, 2008. Selecting a development approach.

Centre for Reviews and Dissemination, University of York, 2009. Systematic reviews: CRD's guidance for undertaking reviews in health care. Centre for Reviews and Dissemination, York.

Charaf, H., 2011. Developing Mobile Applications for Multiple Platforms, in: Engineering of Computer Based Systems (ECBS-EERC), 2011 2nd Eastern European Regional Conference on The. p. 2.

Chen, M., 2004. A methodology for building mobile computing applications. International journal of electronic business 2, 229–243.

Cohen, J., 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. Psychological Bulletin 70, 213–220.

Conradi, R., 2004. Software engineering mini glossary [WWW Document]. URL http://www.idi.ntnu.no/grupper/su/publ/ese/se-defs.html (accessed 5-May-12).

Corcho, O., Fernandez-Lopez, M., Gomez-Perez, A., 2003. Methodologies, tools and languages for building ontologies. Where is their meeting point? Data & Knowledge Engineering 46, 41–64.

Cristani, M., Cuel, R., 2004. A Comprehensive Guideline for Building a Domain Ontology from Scratch, in: Proceedings of I-KNOW '04. Presented at the I-KNOW '04, Graz, Austria.

Crockford, D., 2006. The application/json Media Type for JavaScript Object Notation (JSON) (IEEE Standard No. RFC4627). Network Working Group.

Cuccurullo, S., Francese, R., Risi, M., Tortora, G., 2011. A Visual Approach supporting the Development of MicroApps on Mobile Phones, in: Proc. of 3rd International Symposium on End-User Development. Presented at the 3rd International Symposium on End-User Development, Brindisi, Italy, pp. 289–294.

Dahlem, N., 2011. OntoClippy: A User-Friendly Ontology Design and Creation Methodology. International Journal of Intelligent Information Technologies 7, 15–32.

De Nicola, A., Missikoff, M., Navigli, R., 2005. A proposal for a Unified Process for ONtology building: UPON, in: In Proceedings of 16th International Conference on Database and Expert Systems Applications (DEXA.

DeviceAnywhere, 2011. DeviceAnywhere - Test Center Enterprise Automation [WWW Document]. Automated Testing on Smartphones and Tablets. URL http://tiny.cc/DeviceAnywhere (accessed 27-Aug-11).

Dybå, T., Dingsøyr, T., 2008a. Strength of evidence in systematic reviews in software engineering. ACM Press, pp. 178–187.

Dybå, T., Dingsøyr, T., 2008b. Empirical studies of agile software development: A systematic review. Information and Software Technology 50, 833–859.

Dyck, S., Majchrzak, T.A., 2012. Identifying Common Characteristics in Fundamental, Integrated, and Agile Software Development Methodologies. IEEE, pp. 5299–5308.

Ejlersen, A., Knudsen, M.S., Løvgaard, J., Sørensen, M.B., 2008. Using Design Science to Develop a Mobile Application.

Elliott, G., 2004. Global business information technology: an integrated systems approach. Pearson Addison Wesley, Harlow, England; New York.

European Commission, 2010. European Interoperability Framework (EIF 2.0) (COM(2010) 744 final).

Fernandez-Lopez, M., Gomez-Perez, A., Juristo, N., 1997. METHONTOLOGY: from Ontological Art towards Ontological Engineering, in: Proceedings of the AAAI97 Spring Symposium. Stanford, USA, pp. 33–40.

Fielding, R.T., 2000. Architectural styles and the design of network-based software architectures. University of California, Irvine.

Fjellheim, T., Milliner, S., Dumas, M., Vayssière, J., 2007. A process-based methodology for designing event-based mobile composite applications. Data & Knowledge Engineering 61, 6 – 22.

Forstner, B., Lengyel, L., Kelenyi, I., Levendovszky, T., Charaf, H., 2005. Supporting Rapid Application Development on Symbian Platform, in: Computer as a Tool, 2005. EUROCON 2005.The International Conference On. pp. 72 –75.

Forstner, B., Lengyel, L., Levendovszky, T., Mezei, G., Kelenyi, I., Charaf, H., 2006. Model-based system development for embedded mobile platforms, in: Model-Based Development of Computer-Based Systems and Model-Based Methodologies for Pervasive and Embedded Software, 2006. MBD/MOMPES 2006. Fourth and Third International Workshop On. p. 10–pp.

Gal, V., Topol, A., 2005. Experimentation of a Game Design Methodology for Mobile Phones Games.

Gasson, S., 1995. The role of methodologies in IT-related organisational change, in: Proceedings of BCS Specialist Group on IS Methodologies, 3rd Annual Conference, The Application of Methodologies in Industrial and Business Change. Presented at the 3rd Annual Conference, The Application of Methodologies in Industrial and Business Change, North East Wales Institute, Wrexham.

Gómez-Pérez, A., 2001. Evaluation of ontologies. International Journal of Intelligent Systems 16, 391–409.

Gómez-Pérez, A., 2004. Ontological engineering: with examples from the areas of knowledge management, e-commerce and the Semantic Web, Advanced information and knowledge processing. Springer, London ; New York.

Gómez-Pérez, A., 2004. Ontology Evaluation, in: Handbook on Ontologies, International Handbooks on Information Systems. Springer, pp. 251–274.

Gong, R., Li, Q., Ning, K., Chen, Y., O'Sullivan, D., 2006. Business process collaboration using semantic interoperability: Review and framework, in: Mizoguchi, R., Shi, Z., Giunchiglia, F. (Eds.), SEMANTIC WEB - ASWC 2006, PROCEEDINGS, LECTURE NOTES IN COMPUTER SCIENCE. pp. 191–204.

Gruber, T.R., 1993a. A translation approach to portable ontology specifications. KNOWLEDGE ACQUISITION 5, 199–220.

Gruber, T.R., 1993b. Toward principles for the design of ontologies used for knowledge sharing (Technical report No. KSL-93-04). Stanford University, Stanford.

Grüninger, M., Fox, M.S., 1995. Methodology for the Design and Evaluation of Ontologies, in: Workshop on Basic Ontological Issues in Knowledge Sharing. Presented at the Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal.

Guarino, N., 1998. Formal Ontology and Information Systems, in: Proceedings of the 1st International Conference on Formal Ontology in Information Systems. Trento, Italy, pp. 3–15.

Guide to the software engineering body of knowledge (SWEBOK V3) - Software engineering models and methods (Chapter 10 - Unpublished - In Review) (Technical report No. ?), 2012.

Guide to the software engineering body of knowledge 2004 version: SWEBOK (Technical report No. ISO/IEC TR 19759), 2004. . Los Alamitos, CA.

Hammond, S., Umphress, D., 2012. Test driven development. ACM Press, p. 158.

Hannay, J., Sjoberg, D., Dyba, T., 2007. A Systematic Review of Theory Use in Software Engineering Experiments. IEEE Transactions on Software Engineering 33, 87–107.

Hedberg, H., Iisakka, J., 2006. Technical Reviews in Agile Development: Case Mobile-D, in: Quality Software, 2006. QSIC 2006. Sixth International Conference On. pp. 347–353.

Higgins, J.P.., Green, S. (Eds.), 2011. Cochrane Handbook for Systematic Reviews of Interventions. Version 5.1.0 [updated March 2011]. The Cochrane Collaboration. Available from http://www.cochrane-handbook.org/.

Hilera, J.R., Pages, C., Martinez, J.J., Gutierrez, J.A., de-Marcos, L., 2010. An evolutive process to convert glossaries into ontologies. Information Technology and Libraries 29, 195–204.

Hilpinen, R., 2011. Artifact [WWW Document]. Stanford Encyclopedia of Philosophy. URL http://plato.stanford.edu/entries/artifact/ (accessed 5-May-12).

Holler, R., 2006. Mobile Application Development: A Natural Fit with Agile Methodologies.

Horridge, M., 2009. OWL 2 Validator [WWW Document]. University of Manchester. URL http://owl.cs.manchester.ac.uk/validator/ (accessed 20-Jun-13).

Horridge, M., 2011. A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools.

Horridge, M., Patel-Schneider, P.F., 2009. Manchester Syntax - OWL (W3C Document). W3C.

Hosbond, J.H., 2005. Mobile Systems Development: Challenges, Implications and Issues, in: Krogstie, J., Kautz, K., Allen, D. (Eds.), Mobile Information Systems II, IFIP International Federation for Information Processing. Springer Boston, pp. 279–286.

Hosbond, J.H., Nielsen, P.A., 2005. Mobile Systems Development - A literature review, in: Proceedings of IFIP 8.2 Annual Conference.

Humphrey, W.S., 1989. Managing the software process. Addison-Wesley, Reading, Mass.

IEEE Computer Society, 1991. IEEE Standard Computer Dictionary. A Compilation of IEEE Standard Computer Glossaries (610-1991) (IEEE Std No. 610-1991).

IEEE Computer Society., 1990. IEEE standard computer dictionary : a compilation of IEEE standard computer glossaries, 610. Institute of Electrical and Electronics Engineers, New York NY USA.

Ihme, T., Abrahamsson, P., 2005. The Use of Architectural Patterns in the Agile Software Development of Mobile Applications.

Jeong, Y.-J., Lee, J.-H., Shin, G.-S., 2008. Development Process of Mobile Application SW Based on Agile Methodology, in: Proceedings of 10th International Conference on Advanced Communication Technology, (ICACT 2008). IEEE, Gangwon-Do, pp. 362–366.

Jørgensen, M., 2007. Estimation of Software Development Work Effort:Evidence on Expert Judgment and Formal Models. International Journal of Forecasting 23, 449–462.

Kaariainen, J., Koskela, J., Abrahamsson, P., Takalo, J., 2004. Improving requirements management in extreme programming with tool support - an improvement attempt that failed, in: Euromicro Conference, 2004. Proceedings. 30th. pp. 342 – 351.

Kabilan, V., 2007. Ontology for information systems (04IS) design methodology: conceptualizing, designing and representing domain ontologies. Data- och systemvetenskap, Kungliga Tekniska högskolan, Kista.

Kangas, E., Kinnunen, T., 2005. Applying user-centered design to mobile application development. Communications of the ACM 48, 55–59.

Khambati, A., Grundy, J., Warren, J., Hosking, J., 2008. Model-Driven Development of Mobile Personal Health Care Applications, in: Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering, ASE '08. IEEE Computer Society, Washington, DC, USA, pp. 467–470.

Khan, U.A., 2008. Improved Iterative Software Development Method for Game Design.

Khondoker, R.M., Mueller, P., 2010. Comparing Ontology Development Tools Based on an Online Survey, in: Proceedings of the World Congress on Engineering. Presented at the WCE 2010, London.

Kim, H., Choi, B., Yoon, S., 2009. Performance testing based on test-driven development for mobile applications, in: Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication, ICUIMC '09. ACM, New York, NY, USA, pp. 612–617.

Kim, H.K., 2008. Frameworks of Process Improvement for Mobile Applications. Engineering Letters 16.

Kim, W.Y., Son, H.S., Kim, J.S., Kim, R.Y., 2010. Development of Windows Mobile Applications using Model Transformation Techniques. Journal of KISS: Computing Practices 16, 1091–5.

Kitchenham, B., 2004. Procedures for Performing Systematic Reviews (Technical report No. Keele University Technical Report TR/SE-0401; NICTA Technical Report 0400011T.1). Software Engineering Group; National ICT Australia Ltd., Keele; Eversleigh.

Kitchenham, Barbara, Brereton, P., Turner, M., Niazi, M., Linkman, S., Pretorius, R., Budgen, D., 2009. The impact of limited search procedures for systematic literature reviews - A participant-observer case study. IEEE, pp. 336–345.

Kitchenham, B., Charters, S., 2007. Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3 (Technical report No. EBSE-2007-01). Keele University and University of Durham.

Kitchenham, B, Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., Linkman, S., 2009. Systematic literature reviews in software engineering – A systematic literature review. Information and Software Technology 51, 7–15.

Kitchenham, B., Pretorius, R., Budgen, D., Pearl Brereton, O., Turner, M., Niazi, M., Linkman, S., 2010. Systematic literature reviews in software engineering - A tertiary study. Information and Software Technology 52, 792–805.

Kno.e.sis Research Group, 2011. Welcome to Kno.e.sis [WWW Document]. URL http://knoesis.wright.edu/ (accessed 27-Aug-11).

Korkala, M., Abrahamsson, P., 2004. Extreme programming: Reassessing the requirements management process for an offsite customer. Software Process Improvement 12–22.

Kurschl, W., Mitsch, S., Prokop, R., Schonbock, J., 2007. Gulliver - a framework for building smart speech-based applications, in: Proceedings of the 40th Annual Hawaii International Conference on System Sciences. Waikoloa, HI, USA.

Kynkäänniemi, T., Komulainen, K., 2006. Agile Documentation in Mobile-D Projects (Agile Deliverable No. D.2.10), Agile Software Development of Embedded Systems.

La, H.J., Kim, S.D., 2009. A service-based approach to developing Android Mobile Internet Device (MID) applications. 2009 IEEE International Conference on Service-Oriented Computing and Applications (SOCA) 00, 1–7.

La, H.J., Lee, H.J., Kim, S.D., 2011. An efficiency-centric design methodology for mobile application architectures, in: Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference On. pp. 272–279.

Lovrenčić, S., 2007. Formalna ontologija sveučilišnih studija (Doctoral dissertation). University of Zagreb, Varazdin, Croatia.

Lumsden, J., Hall, H., Cruickshank, P., 2011. Ontology definition and construction, and epistemological adequacy for systems interoperability: A practitioner analysis. Journal of Information Science 37, 246–253.

Lunny, A., 2011. Phonegap beginner's guide: build cross-platform mobile applications with the PhoneGap open source development framework. Packt Publishing Limited, Birmingham, UK.

Madiraju, P., Malladi, S., Balasooriya, J., Hariharan, A., Prasad, S.K., Bourgeois, A., 2010. A methodology for engineering collaborative and ad-hoc mobile applications using SyD middleware. Journal of Network and Computer Applications 33, 542 – 555.

Maharmeh, M., Unhelkar, B., 2009. A Composite Software Framework Approach for Mobile Application Development. Handbook of research in mobile business: technical, methodological, and social perspectives 194.

Maia, M.E.F., Celes, C., Castro, R., Andrade, R.M.C., 2010. Considerations on developing mobile applications based on the Capuchin project, in: Proceedings of the 2010 ACM Symposium on Applied Computing, SAC '10. ACM, New York, NY, USA, pp. 575–579.

Makunga, L., Church, K., 2002. Software Development in Mobile Computing Applications. INFORMATION TECHNOLOGY ON THE MOVE 257.

Manjunatha, A., Ranabahu, A., Sheth, A., Thirunarayan, K., 2010. Power of Clouds in Your Pocket: An Efficient Approach for Cloud Mobile Hybrid Application Development,

in: 2010 IEEE Second International Conference on Cloud Computing Technology and Science. pp. 496–503.

Marinho, F.G., Andrade, R.M.C., Werner, C., Viana, W., Maia, M.E.F., Rocha, L.S., Teixeira, E., Filho, J.B.F., Dantas, V.L.L., Lima, F., Aguiar, S., 2012. MobiLine: A Nested Software Product Line for the domain of mobile and context-aware applications. Science of Computer Programming -.

Martin, J., 1986. Information Engineering. Savant Research Studies, Lancashire.

Mian, P., Conte, T., Natali, A., Biolchini, J., Travassos, G., 2005. A Systematic Review Process for Software Engineering, in: ESELAW '05: 2nd Experimental Software Engineering Latin American Workshop.

Miller, J., 2008. Cohesion And Coupling. MSDN Magazine - The Microsoft Journal for Developers 23.

Mitchell, J.C., 2003. Concepts in programming languages. Cambridge University Press, Cambridge, UK ; New York.

Niemela, P., 2009. JSR 256: Mobile Sensor API [WWW Document]. The Java Community Process(SM) Program - JSRs: Java Specification Requests. URL http://jcp.org/en/jsr/detail?id=248 (accessed 17-May-12).

Noblit, G.W., Hare, R.D., 1988. Meta-ethnography: synthesizing qualitative studies. SAGE, London.

Noy, N.F., McGuinness, D.L., 2001. Ontology Development 101: AGuide to Creating Your First Ontology (Technical report No. KSL-01-05; SMI-2001-0880), Stanford Knowledge Systems Laboratory and Stanfrod Medical Informatics Technical Report. Stanford University, Stanfrod.

Nyström, A., 2011. Agile Solo - Defining and Evaluating an Agile Software Development Process for a Single Software Developer.

Olle, T.W., Hagelstein, J., Macdonald, I.G., Rolland, C., Sol, H.G., Van Assche, F.J.M., Verrijn-Stuart, A.A., 1988. Information systems methodologies: a framework for understanding. Addison-Wesley Pub. Co, Wokingham, England ; Reading, Mass.

Olle, T.W., Sol, H.G., Tully, C.J. (Eds.), 1983. Information systems design methodologies: a feature analysis: Proceedings of the IFIP WG 8.1 Working Conference on Feature Analysis of Information Systems Design Methodologies, York, U.K., 5-7 July, 1983. North-Holland ; Sole distributors for the U.S.A. and Canada, Elsevier Science Pub. Co, Amsterdam ; New York : New York, N.Y.

Olle, T.W., Sol, H.G., Verrijn Stuart, A.A. (Eds.), 1982. Information systems design methodologies: a comparative review: proceedings of the IFIP WG 8.1 Working Conference on Comparative Review of Information Systems Design Methodologies, Noordwijkerhout, The Netherlands, 10-14 May 1982. North-Holland Pub. Co. ; Sole distributors for the U.S.A. and Canada, Elsevier Science Pub. Co, Amsterdam ; New York : New York, N.Y.

Olle, T.W., Sol, H.G., Verrijn Stuart, A.A. (Eds.), 1986. Information systems design methodologies: improving the practice: proceedings of the IFIP WG 8.1 Working Conference on Comparative Review of Information Systems Design Methodologies, Improving the Practice, Noordwijkerhout, The Netherlands, 5-7 May, 1986, Post-conference ed. ed. North-Holland Pub. Co. ; Sole distributors for the U.S.A. and Canada, Elsevier Science Pub. Co, Amsterdam ; New York : New York, N.Y.

Ortiz, G., Prado, A.G.D., 2010. Improving device-aware Web services and their mobile clients through an aspect-oriented, model-driven approach. Information and Software Technology 52, 1080 – 1093.

Papageorgiou, A., Leferink, B., Eckert, J., Repp, N., Steinmetz, R., 2009. Bridging the gaps towards structured mobile SOA. ACM Press, p. 288.

Park, J., Ram, S., 2004. Information systems interoperability: What lies beneath? ACM TRANSACTIONS ON INFORMATION SYSTEMS 22, 595–632.

Parker, P.M., 2011. Definition of artifact [WWW Document]. Webster's Online Dictionary. Url http://www.websters-online-dictionary.org/definitions/artifact (accessed 5-Jul-11).

Paspallis, N., Papadopoulos, G.A., 2006. An approach for developing adaptive, mobile applications with separation of concerns, in: Computer Software and Applications Conference, 2006. COMPSAC'06. 30th Annual International. pp. 299–306.

Pauca, V.P., Guy, R.T., 2012. Mobile apps for the greater good: a socially relevant approach to software engineering, in: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, SIGCSE '12. ACM, New York, NY, USA, pp. 535–540.

Paulheim, H., Probst, F., 2010. Application integration on the user interface level: An ontology-based approach. DATA & KNOWLEDGE ENGINEERING 69, 1103–1116.

Petticrew, M., Roberts, H., 2005. Systematic reviews in the social sciences: a practical guide. Blackwell Pub., Malden, MA.

PhoneGap, 2011. Take the pain out of compiling mobile apps for multiple platforms [WWW Document]. PhoneGap Build. URL https://build.phonegap.com (accessed 27-Aug-11).

Rahimian, V., Ramsin, R., 2008. Designing an agile methodology for mobile software development: A hybrid method engineering approach, in: Proceedings of Second International Conference on Research Challenges in Information Science, RCIS (2008). IEEE, Marrakech, pp. 337–342.

Ramsin, R., Paige, R.F., 2008. Process-centered review of object oriented software development methodologies. ACM Computing Surveys 40, 1–89.

Ranabahu, A.H., Maximilien, E.M., Sheth, A.P., Thirunarayan, K., 2011. A domain specific language for enterprise grade cloud-mobile hybrid applications, in: Proceedings of the Compilation of the Co-located Workshops on DSM'11, TMC'11, AGERE!'11, AOOPES'11, NEAT'11, & VMIL'11, SPLASH '11 Workshops. ACM, New York, NY, USA, pp. 77–84.

Reda, R., 2012. Robotium - The world's leading Android™ test automation framework [WWW Document]. URL http://code.google.com/p/robotium/ (accessed 6-Jun-12).

Rhomobile, Inc., 2011. Smartphone Enterprise Application Integration, White paper [WWW Document]. URL http://tiny.cc/rhomobile (accessed 20-Aug-11).

Ridene, Y., Belloir, N., Barbier, F., Couture, N., 2010. A DSML For Mobile Phone Applications Testing, in: Proceedings of 10th Workshop on Domain-Specific Modeling in SPLASH. France.

Rosa, R.E.V.S., Lucena,Jr., V.F., 2011. Smart composition of reusable software components in mobile application product lines, in: Proceedings of the 2nd International Workshop on Product Line Approaches in Software Engineering, PLEASE '11. ACM, New York, NY, USA, pp. 45–49.

Rossi, M., Tuunanen, T., 2010. A method and tool for rapid consumer application development. International Journal of Organisational Design and Engineering 1, 109–125.

Rupnik, R., 2009. Mobile Applications Development Methodology, in: Unhelkar, B. (Ed.), Handbook of Research in Mobile Business: Technical, Methodological, and Social Perspectives. IGI Global Snippet.

Saifudin, A.W.S.N., Salam, B.S., Abdullah, C.M.H.L., 2011. MMCD Framework and Methodology for Developing m-Learning Applications. Presented at the International conference on Teaching & Learning in Higher Education (ICTLHE 2011).

Salo, O., 2004. Improving software process in agile software development projects: results from two XP case studies, in: Euromicro Conference, 2004. Proceedings. 30th. pp. 310–317.

Salo, O., Koskela, J., 2004. Mobile-D Glossary, VTT Technical Research Centre of Finland, Available at: http://agile.vtt.fi/mobile-d.zip.

Scharff, C., 2010. The Software Engineering of Mobile Application Development.

Scharff, C., 2011. Guiding global software development projects using Scrum and Agile with quality assurance, in: Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference On. pp. 274–283.

Scharff, C., Verma, R., 2010. Scrum to support mobile application development projects in a just-in-time learning context, in: Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, CHASE '10. ACM, New York, NY, USA, pp. 25–31.

Schwieren, J., Vossen, G., 2009. A design and development methodology for mobile RFID applications based on the ID-Services middleware architecture, in: Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference On. pp. 260–266.

Services Research Lab, Metadata and Languages Lab, 2011. Cloud-Mobile Hybrid Application Generator [WWW Document]. MobiCloud. URL http://mobicloud-classic.knoesis.org/ (accessed 27-Aug-11).

Shah, M., Mears, B., Chakrabarti, C., Spanias, A., Center, S., Tempe, A., 2012. A Top-Down Design Methodology Using Virtual Platforms for Concept Development.

Shiratuddin, N., Sarif, S.M., 2008. m d-Matrix: Mobile Application Development Tool. Proceedings of the International MultiConference of Engineers and Computer Scientists 1.

Shiratuddin, N., Sarif, S.M., 2009. Construction of Matrix and eMatrix for Mobile Development Methodologies, in: Handbook of Research in Mobile Business: Technical, Methodological, and Social Perspectives. IGI Global, pp. 113–126.

Simonsen, A., 2004. Developing mobile applications.

Spataru, A.C., 2010. Agile Development Methods for Mobile Applications (PhD Thesis, University of Edinburgh). The University of Edinburgh, Edinburgh.

Stapić, Z., López, E.G., Cabot, A.G., de Marcos Ortega, L., Strahonja, V., 2012. Performing Systematic Literature Review in Software Engineering, in: Proceedings of 23rd Central European Conference on Information and Intelligent Systems. Presented at the

Central European Conference on Information and Intelligent Systems - CECIIS, Faculty of Organization and Informatics, Varaždin, pp. 441–447.

Staples, M., Niazi, M., 2007. Experiences using systematic review guidelines. Journal of Systems and Software 80, 1425–1437.

Staples, M., Niazi, M., 2008. Systematic review of organizational motivations for adopting CMM-based SPI. Information and Software Technology 50, 605–620.

Studer, R., Benjamins, V.R., Fensel, D., 1998. Knowledge engineering: Principles and methods. Data & Knowledge Engineering 25, 161–197.

Su, S.H., Scharff, C., 2010. Know Yourself and Beyond: A Global Software Development Project Experience with Agile Methodology, in: Proceedings of Student-Faculty Research Day, CSIS. Pace University.

Supan, D., Teković, K., Škalec, J., Stapić, Z., 2013. Using Mobile-D methodology in development of mobile applications: challenges and issues, in: Razvoj Poslovnih i Informatičkih Sustava CASE 25. Presented at the Razvoj poslovnih i informatičkih sustava CASE 25, CASE d.o.o, Rijeka, pp. 91–98.

Tantsis, G., 2013. Ontology evaluation plug-in for the Protege software (Master Thesis). Aristotle University of Thessaloniki, Thessaloniki.

Terani, N.S., 2012. IPhone Application Development Challenges and Solutions. CALIFORNIA STATE UNIVERSITY.

Thompson, C., White, J., Dougherty, B., Turner, H., Campbell, S., Zienkiewicz, K., Schmidt, D.C., 2010. Model-Driven Architectures for Optimizing Mobile Application Performance.

Um, J., Hong, S., Kim, Y.T., Chung, E., Choi, K.M., Kong, J.T., Eo, S.K., 2005. ViP: A Practical Approach to Platform-based System Modeling Methodology. Journal of Semiconductor Technology and Science 5, 89.

Unterkalmsteiner, M., Gorschek, T., Islam, A.K.M.M., Cheng, C.K., Permadi, R.B., Feldt, R., 2012. Evaluation and Measurement of Software Process Improvement - A Systematic Literature Review. IEEE Transactions on Software Engineering.

Uschold, M., Gruninger, M., 1996. Ontologies: Principles, methods and applications. Knowledge Engineering Review 11, 93–136.

Uschold, M., King, M., 1995. Towards a Methodology for Building Ontologies, in: In Workshop on Basic Ontological Issues in Knowledge Sharing, Held in Conjunction with IJCAI-95.

Vrandečić, D., 2009. Ontology Evaluation, in: Handbook on Ontologies, International Handbooks on Information Systems. Springer, pp. 293–313.

VTT Technical Research Centre of Finland, 2004. Mobile-D Product Description [WWW Document]. AGILE Software Technologies Research Programme. URL http://agile.vtt.fi/prodserv.html (accessed 16-May-12).

VTT Technical Research Centre of Finland, 2006a. Mobile-D Online Presentation (Web Application) [WWW Document]. AGILE Software Technologies Research Programme. URL http://agile.vtt.fi/mobiled.html (accessed 16-May-12).

VTT Technical Research Centre of Finland, 2006b. Mobile-D Description and Templates (ZIP Archive Document), Available at: http://agile.vtt.fi/mobile-d.zip.

W3C OWL Working Group, 2012. OWL 2 Web Ontology Language Document Overview (Second Edition) (W3C Recommendation No. REC-owl2-overview-20121211).

W3C Web Ontology Working Group, 2004. OWL Web Ontology Language Guide (W3C Recommendation No. REC-owl-guide-20040210).

WAC Application Services Ltd, 2012a. WAC Apps [WWW Document]. WAC Apps - Developer Website. URL http://www.wacapps.net/wac-apps (accessed 18-May-12).

WAC Application Services Ltd, 2012b. WAC APIs [WWW Document]. WAC APIs - Developer Website. URL http://www.wacapps.net/wac-apis (accessed 18-May-12).

WAC Application Services Ltd, 2012c. WAC Payment API SDKs [WWW Document]. WAC Payment API Resources - Developer Website. URL http://www.wacapps.net/sdks (accessed 18-May-12).

Walkerdine, J., Phillips, P., Lock, S., 2009. A Tool Supported Methodology For Developing Secure Mobile P2P Systems, in: Mobile Peer-to-peer Computing for Next Generation Distributed Environments: Advancing Conceptual and Algorithmic Applications. pp. 283–301.

Wang, H.H., Noy, N.F., Rector, A., Musen, M., Redmond, T., Rubin, D., Tu, S., Tudorache, T., Drummond, N., Horridge, M., Seidenberg, J., 2006. Frames and OWL Side by Side. Presented at the The Ninth International Protégé Conference, Stanford University, Stanford.

Wasserman, A.I., 2010. Software engineering issues for mobile application development, in: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, FoSER '10. ACM, New York, NY, USA, pp. 397–400.

Williams, B.J., Carver, J.C., 2010. Characterizing software architecture changes: A systematic review. Information and Software Technology 52, 31–51.

Wolkerstorfer, P., Tscheligi, M., Sefelin, R., Milchrahm, H., Hussain, Z., Lechner, M., Shahzad, S., 2008. Probing an agile usability process, in: CHI '08 Extended Abstracts on Human Factors in Computing Systems, CHI EA '08. ACM, New York, NY, USA, pp. 2151–2158.

Xiong, Y., Wang, A., 2010. A new combined method for UCD and software development and case study, in: Information Science and Engineering (ICISE), 2010 2nd International Conference On. pp. 1–4.

Yee, K.Y., Tiong, A.W., Tsai, F.S., Kanagasabai, R., 2009. OntoMobiLe: A Generic Ontology-Centric Service-Oriented Architecture for Mobile Learning. IEEE, pp. 631–636.

Youn, S., McLeod, D., 2006. Ontology Development Tools for Ontology Based Knowledge Management (CREATE Reserach Archive. Non-published Research Reports No. Paper 100).

Zakal, D., Lengyel, L., Charaf, H., 2011. Software Product Lines-based development, in: Applied Machine Intelligence and Informatics (SAMI), 2011 IEEE 9th International Symposium On. pp. 79–81.

Zeidler, C., Kittl, C., Petrovic, O., 2008. An integrated product development process for mobile software. International Journal of Mobile Communications 6, 345–356.

APPENDIXES

Appendix A – Papers selected for the SLR Phase 2 analysis

S1. Abrahamsson P, Salo O, Ronkainen J, Warsta J, 2002. Agile software development methods - Review and Analysis (Report No. VTT Publ. 478). VTT Technical Research Centre of Finland.

S2. Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., Koskela, J., Kyllönen, P., Salo, O., 2004. Mobile-D: an agile approach for mobile application development, in: Companion to the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages, and Applications, OOPSLA '04. ACM, New York, NY, USA, pp. 174–175.

S3. Abrahamsson, P., Hanhineva, A., Jäälinoja, J., 2005. Improving business agility through technical solutions: A case study on test-driven development in mobile software development, in: Business Agility and Information Technology Diffusion. Presented at the IFIP TC8 WG 8.6 International Working Conference.

S4. Abrahamsson, P., Ihme, T., Kolehmainen, K., Kyllönen, P., Salo, O., 2009. Mobile-D for Mobile Software: How to Use Agile Approaches for the Efficient Development of Mobile Applications.

S5. Abrahamsson, P., Still, J., 2007. Agile software development: theoretical and practical outlook. Product-Focused Software Process Improvement 410–411.

S6. Abrahamsson, P., Warsta, J., Siponen, M.T., Ronkainen, J., 2003. New directions on agile methods: a comparative analysis, in: Software Engineering, 2003. Proceedings. 25th International Conference On. pp. 244–254.

S7. Acharya, S., Mohanty, H., Shyamasundar, R., 2003. MOBICHARTS: a notation to specify mobile computing applications, in: System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference On. p. 11–pp.

S8. Ahlgren, R., Markkula, J., 2005. Design patterns and organisational memory in mobile application development. Product Focused Software Process Improvement 1–35.

S9. Ahtinen, A., Nurminen, J.K., Häkkilä, J., 2007. Developing a mobile reporting system for road maintenance: user research perspective, in: Proceedings of the 4th International Conference on Mobile Technology, Applications, and Systems and the 1st International Symposium on Computer Human Interaction in Mobile Technology, Mobility '07. ACM, New York, NY, USA, pp. 1–7.

S10. Aini, Q., La Katjong, B., dan Kartika Sari Puteri, E.R., 2011. Application development of mobile Transjakarta route map: (case study: Jakarta Indonesia), in: Proceedings of the 9th International Conference on Advances in Mobile Computing and Multimedia, MoMM '11. ACM, New York, NY, USA, pp. 264–267.

S11. Al Bar, A., Mohamed, E., Akhtar, M.K., Abuhashish, F., 2011. A preliminary review of implementing Enterprise Mobile Application in ERP environment.

S12. Alahuhta, P., Löthman, H., Helaakoski, H., Koskela, A., Röning, J., 2006. Experiences in developing mobile applications using the Apricot Agent Platform. Personal and Ubiquitous Computing 11, 1–10.

S13. Alatalo, P., Järvenoja, J., Karvonen, J., Keronen, A., Kuvaja1, P., 2002. Mobile application architectures. Product Focused Software Process Improvement 572–586.

S14. Algan, F., Tuğlular, T., 2005. Test Driven Software Development.

S15. Ali, N., Ramos, I., Solís, C., 2010. Ambient-PRISMA: Ambients in mobile aspect-oriented software architecture. Journal of Systems and Software 83, 937 – 958.

S16. Ali, N.N., Mansoor, H., 2011. Cross Platform Mobile Application Development Framework.

S17. Al-Maharmeh, M., Unhelkar, B., 2009. Applying a Composite Process Framework (CPF) in Real Life Software Development Project, in: Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference On. pp. 1384–1389.

S18. Alyani, N., Shirzad, S., 2011. Learning to innovate in distributed mobile application development: Learning episodes from Tehran and London, in: 2011 Federated Conference on Computer Science and Information Systems (FedCSIS). Presented at the 2011 Federated Conference on Computer Science and Information Systems (FedCSIS). IEEE., Piscataway, NJ, USA, pp. 497–504.

S19. Amanquah, N., Eporwei, O.T., 2009. Rapid application development for mobile terminals, in: Adaptive Science Technology, 2009. ICAST 2009. 2nd International Conference On. pp. 410–417.

S20. Amoroso, D.L., Ogawa, M., 2011. Japan's Model of Mobile Ecosystem Success: The Case of NTT DoCoMo. Journal of Emerging Knowledge on Emerging Markets 3, 27.

S21. Andersson, B., Henningsson, S., 2010. Developing Mobile Information Systems: Managing Additional Aspects.

S22. Andes, D., Cremer, J., Draxler, B., Dudley, N., Haldeman, L., Hsieh, H., Likarish, P., Nguyen, D.T., Sarnelli, C., Winet, J., 2011. UCOL – Iowa City UNESCO City of Literature: mobile application research & development, in: Proceedings of the 2011 iConference, iConference '11. ACM, New York, NY, USA, pp. 636–637.

S23. Aslan, I., Leichtenstern, K., Holleis, P., Wasinger, R., Stahl, C., 2010. Tool-support for mobile and pervasive application development - issues and challenges, in: Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services, MobileHCI '10. ACM, New York, NY, USA, pp. 499–502.

S24. Ayob, N., Hussin, A.R.C., Dahlan, H.M., 2009. Three layers design guideline for mobile application, in: Information Management and Engineering, 2009. ICIME'09. International Conference On. pp. 427–431.

S25. Azhari, S., Wardoyo, R., Hartati, S., 2008. Development of distribution application using intelligent mobile agent approach for accessing the progress status of enterprise projects. Proceeding The 4th International Conference on Information & Communication Technology and System (ICTS).

S26. B'far, R., 2005. Mobile Computing Principles: designing and developing mobile applications with UML and XML. Cambridge Univ Pr.

S27. Balagtas-Fernandez, F., Hussmann, H., 2009. A Methodology and Framework to Simplify Usability Analysis of Mobile Applications, in: Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering, ASE '09. IEEE Computer Society, Washington, DC, USA, pp. 520–524.

S28. Balagtas-Fernandez, F., Tafelmayer, M., Hussmann, H., 2010. Mobia Modeler: easing the creation process of mobile applications for non-technical users, in: Proceedings of the 15th International Conference on Intelligent User Interfaces, IUI '10. ACM, New York, NY, USA, pp. 269–272.

S29. Balagtas-Fernandez, F.T., Hussmann, H., 2008. Model-Driven Development of Mobile Applications, in: Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering, ASE '08. IEEE Computer Society, Washington, DC, USA, pp. 509–512.

S30. Ballagas, R., Memon, F., Reiners, R., Borchers, J., 2007. iStuff mobile: rapidly prototyping new mobile phone interfaces for ubiquitous computing, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '07. ACM, New York, NY, USA, pp. 1107–1116.

S31. Baloian, N., Zurita, G., Antunez, P., Baytelman, F., 2007. A Flexible, Lightweight Middleware Supporting the Development of Distributed Applications across Platforms, in: Computer Supported Cooperative Work in Design, 2007. CSCWD 2007. 11th International Conference On. pp. 92–97.

S32. Bareiss, R., Sedano, T., 2011. Improving Mobile Application Development, in: Proceedings of 2nd Annual Workshop on Software Engineering for Mobile Applications Development. Presented at the 2nd Annual Workshop on Software Engineering for Mobile Applications Development, Santa Monica, CA, USA, pp. 5–8.

S33. Barnawi, A., Al-Talhi, A.H., Qureshi, M., Khan, A.I., 2012. Novel Component Based Development Model For Sip-Based Mobile Application. Arxiv preprint arXiv:1202.2516.

S34. Barnawi, A., Qureshi, M., Khan, A.I., 2012. A Framework for Next Generation Mobile and Wireless Networks Application Development using Hybrid Component Based Development Model. Arxiv preprint arXiv:1202.2515.

S35. Behrens, H., 2010. MDSD for the iPhone: developing a domain-specific language and IDE tooling to produce real world applications for mobile devices, in: Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion, SPLASH '10. ACM, New York, NY, USA, pp. 123–128.

S36. Bellotti, F., Berta, R., Gloria, A.D., Margarone, M., 2003. MADE: developing edutainment applications on mobile computers. Computers & Graphics 27, 617 – 634.

S37. Bellotti, F., Berta, R., Margarone, M., De Gloria, A., 2008. oDect: an RFID-based object detection API to support applications development on mobile devices. Software: Practice and Experience 38, 1241–1259.

S38. Benou, P., Bitos, V., 2009. Developing mobile commerce applications. Selected readings on electronic commerce technologies: contemporary applications.

S39. Bergström, F., Engvall, G., 2011. Development of handheld mobile applications for the public sector in Android and iOS using agile Kanban process tool.

S40. Bertolli, C., Buono, D., Mencagli, G., Vanneschi, M., 2010. An Approach to Mobile Grid Platforms for the Development and Support of Complex Ubiquitous Applications. International Journal of Advanced Pervasive and Ubiquitous Computing (IJAPUC) 2, 24–38.

S41. Bhattacharyya, S., 2003. Framework for Developing Adaptable Applications in Pervasive Environments.

S42. Binsaleh, M., Hassan, S., 2011. Systems Development Methodology for Mobile Commerce Applications: Agile vs. Traditional. International Journal of Online Marketing (IJOM) 1, 33–47.

S43. Biswas, A., Donaldson, T., Singh, J., Diamond, S., Gauthier, D., Longford, M., 2006. Assessment of mobile experience engine, the development toolkit for context aware mobile applications, in:

Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, ACE '06. ACM, New York, NY, USA.

S44. Blanco, P., Camarero, J., Fumero, A., Werterski, A., Rodríguez, P., 2009. Metodología de desarrollo ágil para sistemas móviles Introducción al desarrollo con Android y el iPhone.

S45. Blom, S., Book, M., Gruhn, V., Hrushchak, R., Kohler, A., 2008. Write Once, Run Anywhere A Survey of Mobile Runtime Environments, in: Grid and Pervasive Computing Workshops, 2008. GPC Workshops' 08. The 3rd International Conference On. pp. 132–137.

S46. Boonma, P., Suzuki, J., 2011. Model-driven performance engineering for wireless sensor networks with feature modeling and event calculus, in: Proceedings of the 3rd Workshop on Biologically Inspired Algorithms for Distributed Systems, BADS '11. ACM, New York, NY, USA, pp. 17–24.

S47. Bowen, J., Hinze, A., 2011. Supporting Mobile Application Development with Model-Driven Emulation. Electronic Communications of the EASST 45.

S48. Braun, P., Eckhaus, R., 2008. Experiences on model-driven software development for mobile applications, in: Engineering of Computer Based Systems, 2008. ECBS 2008. 15th Annual IEEE International Conference and Workshop on The. pp. 490–493.

S49. Breivold, H.P., Sundmark, D., Wallin, P., Larsson, S., 2010. What Does Research Say about Agile and Architecture?, in: Software Engineering Advances (ICSEA), 2010 Fifth International Conference On. pp. 32–37.

S50. Bungert, A., 2009. Developing for Mobile Platforms.

S51. Burke, S., Hatfield, A., Mosunov, A., Sajwani, F., Shalaby, A., 2012. Open-Source Software Development.

S52. Burton, B., 2011. Mobile App Development with Corona: Getting Started. Burtons Media Group.

S53. Carbon, R., Hess, S., 2011. Mobile Business Applications must be thoroughly engineered!, in: 2nd Annual Workshop on Software Engineering for Mobile Application Development.

S54. Carlson, D., Schrader, A., 2011. A wide-area context-awareness approach for Android, in: Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services, iiWAS '11. ACM, New York, NY, USA, pp. 383–386.

S55. Carter, S.A., Mankoff, J., 2005. Momento: Early-Stage Prototyping and Evaluation for Mobile Applications (in submission) (Technical report No. UCB/CSD-05-1380). EECS Department University of California, Berkeley.

S56. Cha, S., Kurz, J.B., Du, W., 2009. Toward a unified framework for mobile applications, in: Communication Networks and Services Research Conference, 2009. CNSR'09. Seventh Annual. pp. 209–216.

S57. Chapter, X., 2009. Mobile Applications Development Methodology. Handbook of research in mobile business: technical, methodological, and social perspectives 160.

S58. Charaf, H., 2011. Developing Mobile Applications for Multiple Platforms, in: Engineering of Computer Based Systems (ECBS-EERC), 2011 2nd Eastern European Regional Conference on The. p. 2.

S59. Charland, A., Leroux, B., 2011. Mobile application development: web vs. native. Communications of the ACM 54, 49–53.

S60. Chaudhary, A., Bharathan, K., 2011. Component Based Software Reuse in Mobile Application Development.

S61. Chen, G., Kotz, D., 2005. Solar: An open platform for context-aware mobile applications. DTIC Document.

S62. Chen, M., 2004. A methodology for building mobile computing applications. International journal of electronic business 2, 229–243.

S63. Cheng, M.C., Yuan, S.M., 2005. An adaptive mobile application development framework. Embedded and Ubiquitous Computing–EUC 2005 765–774.

S64. Cheng, M.C., Yuan, S.M., 2007. An Adaptive and Unified Mobile Application Development Framework for Java. Journal of Information Science and Engineering 23, 1391.

S65. Cheung, A., Grandison, T., Johnson, C., Schönauer, S., 2007. Ïnfïnïty: a generic platform for application development and information sharing on mobile devices, in: Sixth International ACM Workshop on Data Engineering for Wireless and Mobile Access (MobiDE), MobiDE '07. ACM, New York, NY, USA, pp. 25–32.

S66. Choi, M., 2012. A Platform-Independent Smartphone Application Development Framework. Computer Science and Convergence 787–794.

S67. Choi, Y., Yang, J.S., Jeong, J., 2009. Application framework for multi platform mobile application software development, in: Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference On. pp. 208–213.

S68. Chowdhary, V., 2011a. Mobile Web Application Development [WWW Document]. Refulz, Web Developer's Blog. URL http://php.refulz.com/mobile-web-application-development/

S69. Chowdhary, V., 2011b. XUI–Mobile Application Development Library [WWW Document]. Refulz, Web Developer's Blog. URL http://php.refulz.com/xui-mobile-application-development-library/

S70. Christensen, J.H., 2009. Using RESTful web-services and cloud computing to create next generation mobile applications, in: Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented

Programming Systems Languages and Applications, OOPSLA '09. ACM, New York, NY, USA, pp. 627–634.

S71. Coelho, H.A. de O., Anido, R. de O., Drummond, R., 2006. QuickFrame - A Fast Development Tool for Mobile Applications, in: Innovations in Information Technology, 2006. pp. 1 –5.

S72. Corral, L., Sillitti, A., Succi, G., 2011. Preparing Mobile Software Development Processes to Meet Mission-Critical Requirements, in: 2nd Annual Workshop on Software Engineering for Mobile Application Development.

S73. Corral, L., Sillitti, A., Succi, G., Garibbo, A., Ramella, P., 2011. Evolution of Mobile Software Development from Platform-Specific to Web-Based Multiplatform Paradigm, in: Proceedings of the 10th SIGPLAN Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, ONWARD '11. ACM, New York, NY, USA, pp. 181–183.

S74. Cota, É., Carro, L., Duarte, L., Ribeiro, L., Wagner, F., 2011. XModel: an Unified Effort Towards the Development of High-Quality Mobile Applications, in: 2nd Annual Workshop on Software Engineering for Mobile Application Development. Presented at the 2nd Annual Workshop on Software Engineering for Mobile Application Development, Santa Monica, CA, USA, p. 1.

S75. Cuccurullo, S., Francese, R., Risi, M., Tortora, G., 2011. A Visual Approach supporting the Development of MicroApps on Mobile Phones, in: Proc. of 3rd International Symposium on End-User Development. Presented at the 3rd International Symposium on End-User Development, Brindisi, Italy, pp. 289–294.

S76. Cunha, T.F.V., Dantas, V.L.L., Andrade, R., 2011. SLeSS: a Scrum and Lean Six Sigma integration approach for the development of sofware customization for mobile phones, in: Software Engineering (SBES), 2011 25th Brazilian Symposium On. pp. 283–292.

S77. Dagtas, S., Natchetoi, Y., Wu, H., Hamdi, L., 2008. An Integrated Lightweight Software Architecture for Mobile Business Applications, in: Software Architecture, 2008. WICSA 2008. Seventh Working IEEE/IFIP Conference On. pp. 41–50.

S78. Dastani, M., El Fallah, S.A., Hubner, J., Leite, J. (Eds.), 2011. Languages, Methodologies, and Development Tools for Multi-Agent Systems. Third International Workshop, LADS. Revised Selected Papers.

S79. Davis, V., Gray, J., Jones, J., 2005. Generative approaches for application tailoring of mobile devices, in: Proceedings of the 43rd Annual Southeast Regional Conference - Volume 2, ACM-SE 43. ACM, New York, NY, USA, pp. 237–241.

S80. De Florio, V., Blondia, C., 2008. On the requirements of new software development. International Journal of Business Intelligence and Data Mining 3, 330–349.

S81. de Sá, M., Carriço, L., 2006. Low-fi prototyping for mobile devices, in: CHI '06 Extended Abstracts on Human Factors in Computing Systems, CHI EA '06. ACM, New York, NY, USA, pp. 694–699.

S82. de Souza, C.R.B., Redmiles, D.F., 2009. On The Roles of APIs in the Coordination of Collaborative Software Development. Computer Supported Cooperative Work (CSCW) 18, 445–475.

S83. Degrandsart, S., Demeyer, S., Van den Bergh, J., Mens, T., 2012. A transformation-based approach to context-aware modelling. Software and Systems Modeling 1–18.

S84. Dehlinger, J., Dixon, J., 2011. Mobile Application Software Engineering: Challenges and Research Directions, in: 2nd Annual Workshop on Software Engineering for Mobile Application Development.

S85. Desruelle, H., Blomme, D., Gielen, F., 2011. Adaptive mobile web applications: a quantitative evaluation approach. Web Engineering 375–378.

S86. Di Capua, M., Costagliola, G., De Rosa, M., Fuccella, V., 2011. Rapid prototyping of mobile applications for augumented reality interactions, in: Visual Languages and Human-Centric Computing (VL/HCC), 2011 IEEE Symposium On. pp. 249–250.

S87. Dickson, P.E., 2012. Cabana: a cross-platform mobile development system, in: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, SIGCSE '12. ACM, New York, NY, USA, pp. 529–534.

S88. Diewald, S., Roalter, L., Möller, A., Kranz, M., 2011. Towards a holistic approach for mobile application development in intelligent environments, in: Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia, MUM '11. ACM, New York, NY, USA, pp. 73–80.

S89. Dingsøyr, T., Dybå, T., Moe, N.B. (Eds.), 2010. Agile software development: Current research and future directions. Springer-Verlag New York Inc.

S90. Dodda, S.R., 2010. The Use of SCRUM in Global Software Development: An Exploratory Study (Master thesis).

S91. Doherty, G., McKnight, J., Luz, S., 2010. Fieldwork for requirements: Frameworks for mobile healthcare applications. International Journal of Human-Computer Studies 68, 760 – 776.

S92. Dombroviak, K.M., Ramnath, R., 2007. A taxonomy of mobile and pervasive applications, in: Proceedings of the 2007 ACM Symposium on Applied Computing, SAC '07. ACM, New York, NY, USA, pp. 1609–1615.

S93. Dörflinger, J., Friedland, C., Merz, C., de Louw, R., 2009. Requirements of a mobile procurement framework for rural South Africa, in: Proceedings of the 6th International Conference on Mobile Technology, Application & Systems, Mobility '09. ACM, New York, NY, USA, pp. 3:1–3:4.

S94. Driver, C., Clarke, S., 2008. An application framework for mobile, context-aware trails. Pervasive and Mobile Computing 4, 719 – 736.

S95. Dulipala, J., Ramachandran, V., 2009. SCA for context-aware mobile applications.

S96. Dunkel, J., Bruns, R., 2007. Model-driven architecture for mobile applications, in: Business Information Systems. pp. 464–477.

S97. Dustdar, S., Gall, H., 2003. Architectural concerns in distributed and mobile collaborative systems. Journal of Systems Architecture 49, 457 – 473.

S98. Dwomoh-Tweneboah, M., 2004. Building applications for mobile devices with microsoft visual Studio.NET: tutorial presentation. J. Comput. Sci. Coll. 20, 179–180.

S99. Ejlersen, A., Knudsen, M.S., Løvgaard, J., Sørensen, M.B., 2008. Using Design Science to Develop a Mobile Application.

S100. Emmanouilidis, C., Koutsiamanis, R.-A., Tasidou, A., 2012. Mobile guides: Taxonomy of architectures, context awareness, technologies and applications. Journal of Network and Computer Applications -.

S101. Esfahani, H.C., Cabot, J., Yu, E., 2010. Adopting Agile Methods: Can Goal-Oriented Social Modeling Help, in: 4th International Conference on Research Challenges in Information Science (RCIS). IEEE, France.

S102. Feigin, B., 2009. Mobile Application Development.

S103. Feijóo, C., Gómez-Barroso, J.L., Ramos, S., 2010. An analysis of mobile gaming development, in: Intelligence in Next Generation Networks (ICIN), 2010 14th International Conference On. pp. 1–7.

S104. Felker, C., Slamova, R., Davis, J., 2012. Integrating UX with scrum in an undergraduate software development project, in: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, SIGCSE '12. ACM, New York, NY, USA, pp. 301–306.

S105. Fernando, N., Loke, S.W., Rahayu, W., 2012. Mobile cloud computing: A survey. Future Generation Computer Systems -.

S106. Ferscha, A., Hechinger, M., Mayrhofer, R., Oberhauser, R., 2004. A light-weight component model for peer-to-peer applications, in: Distributed Computing Systems Workshops, 2004. Proceedings. 24th International Conference On. pp. 520–527.

S107. Fjellheim, T., Milliner, S., Dumas, M., Vayssière, J., 2007. A process-based methodology for designing event-based mobile composite applications. Data & Knowledge Engineering 61, 6 – 22.

S108. Forgue, M.-C., Hazaël-Massieux, D., 2012. Mobile web applications: bringing mobile apps and web together, in: Proceedings of the 21st International Conference Companion on World Wide Web, WWW '12 Companion. ACM, New York, NY, USA, pp. 255–258.

S109. Forstner, B., Lengyel, L., Kelenyi, I., Levendovszky, T., Charaf, H., 2005. Supporting Rapid Application Development on Symbian Platform, in: Computer as a Tool, 2005. EUROCON 2005.The International Conference On. pp. 72 –75.

S110. Forstner, B., Lengyel, L., Levendovszky, T., Mezei, G., Kelenyi, I., Charaf, H., 2006. Model-based system development for embedded mobile platforms, in: Model-Based Development of Computer-Based Systems and Model-Based Methodologies for Pervasive and Embedded Software, 2006. MBD/MOMPES 2006. Fourth and Third International Workshop On. p. 10–pp.

S111. Franke, D., Elsemann, C., Kowalewski, S., Weise, C., 2011. Reverse Engineering of Mobile Application Lifecycles, in: Reverse Engineering (WCRE), 2011 18th Working Conference On. pp. 283–292.

S112. Frantz, C., Nowostawski, M., Purvis, M.K., 2012. Augmenting android with AOSE principles for enhanced functionality reuse in mobile applications, in: Proceedings of the 10th International Conference on Advanced Agent Technology, AAMAS'11. Springer-Verlag, Berlin, Heidelberg, pp. 187–211.

S113. Fraunholz, B., Hoffman, J., Jung, J., 2003. Evaluation of mobile frameworks-conceptual and technological aspects, in: Proceedings of the 10th European Conference on Information Technology Evaluation-2003. p. 245.

S114. Gaffar, A., 2009. Enumerating mobile enterprise complexity 21 complexity factors to enhance the design process, in: Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research, CASCON '09. ACM, New York, NY, USA, pp. 270–282.

S115. Gal, V., Topol, A., 2005. Experimentation of a Game Design Methodology for Mobile Phones Games.

S116. Gao, J., Koronios, A., 2010. Mobile Application Development for Senior Citizens, in: Proceedings of PACIS 2010, 14th Pacific Asia Conference on Information Systems, 9-12 July 2010; Taipei, Taiwan. pp. 214–223.

S117. Gasimov, A., Tan, C.H., Phang, C.W., Sutanto, J., 2010. Visiting mobile application development: What, how and where, in: Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR), 2010 Ninth International Conference On. pp. 74–81.

S118. Gavalas, D., Bellavista, P., Cao, J., Issarny, V., 2011. Mobile applications: Status and trends. Journal of Systems and Software 84, 1823 – 1826.

S119. Gavrilovska, A., 2009. Methodology for mobile application product development: A Case Study for Wemlin. Presented at the ICT Innovations 2009, Association for Information and Communication Technologies ICT-ACT, Ohrid, Macedonia.

S120. Gestwicki, P., Ahmad, K., 2011. App inventor for Android with studio-based learning. J. Comput. Sci. Coll. 27, 55–63.

S121. Grassi, V., Mirandola, R., Sabetta, A., 2004. UML based modeling and performance analysis of mobile systems, in: Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of

Wireless and Mobile Systems, MSWiM '04. ACM, New York, NY, USA, pp. 95–104.

S122. Green, R., Mazzuchi, T., Sarkani, S., 2010. Communication and Quality in Distributed Agile Development: An Empirical Case Study. Proceeding in World Academy of Science, Engineering and Technology 61, 322–328.

S123. Grønli, T.-M., Hansen, J., Ghinea, G., 2011. A cloud on the horizon: the challenge of developing applications for Android and iPhone, in: Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments, PETRA '11. ACM, New York, NY, USA, pp. 64:1–64:2.

S124. Guha, P., Shah, K., Shukla, S.S.P., Singh, S., 2011. Incorporating Agile with MDA Case Study: Online Polling System. Arxiv preprint arXiv:1110.6879.

S125. Guo, B., Zhang, D., Imai, M., 2010. Enabling user-oriented management for ubiquitous computing: The meta-design approach. Computer Networks 54, 2840–2855.

S126. Guthery, S.B., Cronin, M.J., 2002. Mobile Application Development with SMS and the SIM Toolkit. McGraw-Hill.

S127. HALSE, S., PATIL, S., 2011. Paper on Aspect-Oriented Software Development and its Usage. Journal of Computer and Mathematical Sciences Vol 2, 581–692.

S128. Hammershoj, A., Sapuppo, A., Tadayoni, R., 2010. Challenges for mobile application development, in: Intelligence in Next Generation Networks (ICIN), 2010 14th International Conference On. pp. 1–8.

S129. Harjula, E., Ylianttila, M., Ala-Kurikka, J., Riekki, J., Sauvola, J., 2004. Plug-and-play application platform: towards mobile peer-to-peer, in: Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia, MUM '04. ACM, New York, NY, USA, pp. 63–69.

S130. Hartmann, G., Stead, G., DeGani, A., 2011. Cross-platform mobile development.

S131. Harun, H., Jailani, N., Bakar, M.A., Zakaria, M.S., Abdullah, S., 2009. A generic framework for developing map-based mobile application, in: Electrical Engineering and Informatics, 2009. ICEEI '09. International Conference On. pp. 434 –440.

S132. Hashim, A.S., Ahmad, W.F.W., Rohiza, A., 2010. A study of design principles and requirements for the m-learning application development, in: User Science and Engineering (i-USEr), 2010 International Conference On. pp. 226–231.

S133. Hedberg, H., Iisakka, J., 2006. Technical Reviews in Agile Development: Case Mobile-D, in: Quality Software, 2006. QSIC 2006. Sixth International Conference On. pp. 347–353.

S134. Hemel, Z., Visser, E., 2011. Declaratively programming the mobile web with Mobl, in: Proceedings of the 2011 ACM International Conference on Object Oriented Programming Systems Languages and Applications, OOPSLA '11. ACM, New York, NY, USA, pp. 695–712.

S135. Ho, H.K., 2004. Mobile application using J2ME.

S136. Holleis, P., 2009. Integrating usability models into pervasive application development.

S137. Holleis, P., Schmidt, A., 2008. Makeit: Integrate user interaction times in the design process of mobile applications. Pervasive Computing 56–74.

S138. Holzer, A., Ondrus, J., 2009. Trends in mobile application development, in: Mobile Wireless Middleware, Operating Systems, and Applications-Workshops. pp. 55–64.

S139. Honda, S., Tomiyama, H., Takada, H., 2007. RTOS and Codesign Toolkit for Multiprocessor Systems-on-Chip, in: Proceedings of the 2007 Asia and South Pacific Design Automation Conference, ASP-DAC '07. IEEE Computer Society, Washington, DC, USA, pp. 336–341.

S140. Hosalkar, A., 2002. Building Mobile Applications with J2EE, J2EE-J2ME and J2EE Extended Application Servers. Proc. of MASPLAS 2.

S141. Hosbond, J., Nielsen, P., 2005. Mobile systems development: a literature review. Designing Ubiquitous Information Environments: Socio-Technical Issues and Challenges 215–232.

S142. Houssos, N., Alonistioti, N., Merakos, L., 2005. Specification and dynamic introduction of 3rd party, service-specific adaptation policies for mobile applications. Mob. Netw. Appl. 10, 405–421.

S143. Hu, X., Du, W., Spencer, B., 2011. A Multi-Agent Framework for Ambient Systems Development. Procedia Computer Science 5, 82 – 89.

S144. Huang, J., Luo, Z., 2010. Research on the Architecture of Mobile Application Development. Computer 11.

S145. Huang, W.C.D., 2007. Design and implementation of a mobile wiki: mobile RikWik.

S146. Huopaniemi, A., 2005. Software Lifecycle Management in Java Environments.

S147. Hussain, Z., Lechner, M., Milchrahm, H., Shahzad, S., Slany, W., Umgeher, M., Vlk, T., Wolkerstorfer, P., 2008. User Interface Design for a Mobile Multimedia Application: An Iterative Approach. IEEE, pp. 189–194.

S148. Hussain, Z., Lechner, M., Milchrahm, H., Shahzad, S., Slany, W., Umgeher, M., Wolkerstorfer, P., 2008. Agile user-centered design applied to a mobile multimedia streaming application. HCI and Usability for Education and Work 313–330.

S149. Ihme, T., Abrahamsson, P., 2005. The Use of Architectural Patterns in the Agile Software Development of Mobile Applications.

S150. Im, T.S., Guimaraes, M., Kennesaw, G., 2004. Component based programming in mobile devices: The future of mobile device development? Inst Informatics &Systemic 255–259.

S151. Jackson, S., Ellis, H., Postner, L., Kurkovsky, S., Mustafaraj, E., 2012. Mobile application development in computing curricula. J. Comput. Sci. Coll. 27, 110–112.

S152. Jacob, J.T.P.N., Coelho, A.F., 2011. Geo Wars–The development of a location-based game. Revista Prisma. Com.

S153. Jadhav, A., Anand, S., Dhangare, N., Wagh, K., 2012. Universal Mobile Application Development (UMAD) On Home Automation. Network and Complex Systems 2, 38–45.

S154. Jang, S., Lee, E., 2009. Reliable Mobile Application Modeling Based on Open API. Advances in Software Engineering 168–175.

S155. Jeong, Y.J., Lee, J.H., Shin, G.S., 2008. Development Process of Mobile Application SW Based on Agile Methodology, in: Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference On. pp. 362–366.

S156. Jiang, M., Yang, Z., 2007. A Model-Driven Approach for Dependable Software Systems, in: Quality Software, 2007. QSIC'07. Seventh International Conference On. pp. 100–106.

S157. Jong-Won Ko, Sung-Ho Sim, Young-Jae Song, 2011. Test Based Model Transformation Framework for Mobile Application. IEEE, pp. 1–7.

S158. Joseph, A.D., Kaashoek, M.F., 1997. Building reliable mobile-aware applications using the Rover toolkit. Wirel. Netw. 3, 405–419.

S159. Juell, M.A., Nordhaug, G.L., 2011. An approach to rapid development of modern ubiquitous Internet applications.

S160. Jugel, U., Preußner, A., 2011. A case study on API generation, in: Proceedings of the 6th International Conference on System Analysis and Modeling: About Models, SAM'10. Springer-Verlag, Berlin, Heidelberg, pp. 156–172.

S161. Julien, C., Roman, G.C., 2006. Egospaces: Facilitating rapid development of context-aware mobile applications. Software Engineering, IEEE Transactions on 32, 281–298.

S162. Julien, C., Roman, G.C., Huang, Q., 2004. Network abstractions for simplifying mobile application development. Technical Report WUCSE-04-37, Washington University.

S163. Kaariainen, J., Koskela, J., Abrahamsson, P., Takalo, J., 2004. Improving requirements management in extreme programming with tool support - an improvement attempt that failed, in: Euromicro Conference, 2004. Proceedings. 30th. pp. 342 – 351.

S164. Kadytė, V., Tétard, F., 2004. The role of usability evaluation and usability testing techniques in the development of a mobile system.

S165. Kähkönen, T., 2011. The effect of service oriented architecture and cloud computing on software testing (Master thesis).

S166. Kangas, E., Kinnunen, T., 2005. Applying user-centered design to mobile application development. Communications of the ACM 48, 55–59.

S167. Kantee, A., Vuolteenaho, H., 2006. Experiences in Portable Mobile Application Development. Advanced Software Engineering: Expanding the Frontiers of Software Technology 138–152.

S168. Karvonen, J., Warsta, J., 2004. Mobile multimedia services development: value chain perspective, in: Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia, MUM '04. ACM, New York, NY, USA, pp. 171–178.

S169. Kaufmann, B., Buechley, L., 2010. Amarino: a toolkit for the rapid prototyping of mobile ubiquitous computing, in: Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services, MobileHCI '10. ACM, New York, NY, USA, pp. 291–298.

S170. Kemper, H.G., Wolf, E., 2002. Iterative process models for mobile application systems: A framework, in: Proceedings of the 23th International Conference on Information System. pp. 401–413.

S171. Keranen, H., Abrahamsson, P., 2005. Naked objects versus traditional mobile platform development: a comparative case study, in: Software Engineering and Advanced Applications, 2005. 31st EUROMICRO Conference On. pp. 274 – 281.

S172. Khambati, A., Grundy, J., Warren, J., Hosking, J., 2008. Model-Driven Development of Mobile Personal Health Care Applications, in: Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering, ASE '08. IEEE Computer Society, Washington, DC, USA, pp. 467–470.

S173. Khan, F.H., Khan, Z.H., 2010. A Systematic Approach for Developing Mobile Information System based on Location Based Services. Network Protocols and Algorithms 2, 54–65.

S174. Khan, U.A., 2008. Improved Iterative Software Development Method for Game Design.

S175. Kim, H., Choi, B., Yoon, S., 2009. Performance testing based on test-driven development for mobile applications, in: Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication, ICUIMC '09. ACM, New York, NY, USA, pp. 612–617.

S176. Kim, H.K., 2008. Frameworks of Process Improvement for Mobile Applications. Engineering Letters 16.

S177. Kim, M., Jeong, J., Park, S., 2005. From product lines to self-managed systems: an architecture-based runtime reconfiguration framework, in: Proceedings of the 2005 Workshop on Design and Evolution of Autonomic Application Software, DEAS '05. ACM, New York, NY, USA, pp. 1–7.

S178. Kim, W.Y., Son, H.S., Kim, J.S., Kim, R.Y., 2010. Development of Windows Mobile Applications using

Model Transformation Techniques. Journal of KISS: Computing Practices 16, 1091–5.

S179. Kinzel, J., 2010. A model driven approach to build high effective and ergonomic mobile business applications, in: Consumer Electronics (ISCE), 2010 IEEE 14th International Symposium On. pp. 1–5.

S180. Koch, F., 2005. Towards a Framework for Intelligent Mobile Service Applications. INFOCOMP Journal of Computer Science 4, 1–10.

S181. Kokkoniemi, J.K., 2008. Gathering Experience Knowledge from Iterative Software Development Processes, in: Hawaii International Conference on System Sciences, Proceedings of the 41st Annual. pp. 333–333.

S182. Kolko, B., Putnam, C., Rose, E., Johnson, E., 2011. Reflection on research methodologies for ubicomp in developing contexts. Personal Ubiquitous Comput. 15, 575–583.

S183. König-Ries, B., 2009. Challenges in mobile application development. it-Information Technology 51, 69–71.

S184. Korkala, M., Abrahamsson, P., 2004. Extreme programming: Reassessing the requirements management process for an offsite customer. Software Process Improvement 12–22.

S185. Kouici, N., Sabri, N., Conan, D., Bernard, G., 2004. MADA, a mobile application development approach, in: Proceedings of the 1st French-speaking Conference on Mobility and Ubiquity Computing, UbiMob '04. ACM, New York, NY, USA, pp. 78–85.

S186. Kraemer, F.A., 2011. Engineering android applications based on UML activities, in: Proceedings of the 14th International Conference on Model Driven Engineering Languages and Systems, MODELS'11. Springer-Verlag, Berlin, Heidelberg, pp. 183–197.

S187. Kramer, D., Clark, T., Oussena, S., 2010. MobDSL: A Domain Specific Language for multiple mobile platform deployment, in: Networked Embedded Systems for Enterprise Applications (NESEA), 2010 IEEE International Conference On. pp. 1–7.

S188. Krevl, A., Vidmar, T., Pancur, M., Ciglaric, M., Tomazic, S., Zavec, A., Ciglaric, S., 2006. A Framework for Developing Mobile Location Based Applications. DTIC Document.

S189. Kulkarni, H., Dascalu, S.M., Harris, F.C., 2009. Software Development Aspects of a Mobile Food Ordering System, in: Proceedings of the ISCA 18th International Conference on Software Engineering and Data Engineering (SEDE '09). Las Vegas, Nevada, pp. 67–72.

S190. Kurschl, W., Mitsch, S., Prokop, R., Schonbock, J., 2007. Gulliver - a framework for building smart speech-based applications, in: Proceedings of the 40th Annual Hawaii International Conference on System Sciences. Waikoloa, HI, USA.

S191. Kynkäänniemi, T., Komulainen, K., 2006. Agile Software Development of Embedded Systems Version: 1.0 Date: 2006.03. 09.

S192. La, H.J., Kim, S.D., 2010. Balanced MVC Architecture for Developing Service-based Mobile Applications, in: e-Business Engineering (ICEBE), 2010 IEEE 7th International Conference On. pp. 292–299.

S193. La, H.J., Lee, H.J., Kim, S.D., 2011. An efficiency-centric design methodology for mobile application architectures, in: Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference On. pp. 272–279.

S194. La, H.J., Lee, H.M., Lee, H.J., Kim, S.D., 2010. Technical issues and lessons learned in developing service-based mobile applications. IEEE, pp. 1–4.

S195. Laakko, T., Leppanen, J., Lahteenmaki, J., Nummiaho, A., 2008. Mobile health and wellness application framework. Methods of Information in Medicine 47, 217–22.

S196. Laitinen, M., Nuckchady, V., Nelimarkka, M., 2008. MUPE as a Rapid Development Architecture – Case Wireless Educational Platform. Presented at the The Nordic Conference of Serious Games.

S197. Lee, S., 2010. Mobile agent based framework for mobile ubiquitous application development, in: Information Science and Applications (ICISA), 2010 International Conference On. pp. 1–5.

S198. Lee, V., Schneider, H., Schell, R., 2004. Mobile applications: Architecture, design, and development. Prentice Hall PTR.

S199. Leichtenstern, K., André, E., 2010. MoPeDT: features and evaluation of a user-centred prototyping tool, in: Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS '10. ACM, New York, NY, USA, pp. 93–102.

S200. Lengyel, L., Levendovszky, T., Charaf, H., 2008. Validated model transformation-driven software development. International Journal of Computer Applications in Technology 31, 106–119.

S201. Li, Z., Steenkamp, A.L., 2010. Mobile Enterprise Architecture Framework. International Journal of Information Technologies and Systems Approach (IJITSA) 3, 1–20.

S202. Lim, W.M., 2005. Towards More Usable Mobile Application Development. IEEE, pp. 1–6.

S203. Lin, H.F., 2012. Design and implementation of a mobile application for personal learning analytics.

S204. Liu, J.J., 2002. Mobile map: A case study in the design & implementation of a mobile application.

S205. Love, S., 2005. Design issues for mobile systems, in: Understanding Mobile Human-Computer Interaction. Butterworth-Heinemann, Oxford, pp. 75 – 98.

S206. Lunn, K., Gidlow, J., Heelas, C., 2002. Mobile application development, a case study in order capture. ICWI 669–672.

S207. Lutes, K., 2004. Software development for mobile computers. Pervasive Computing, IEEE 3, 10–14.

S208. Maaløe, L., Wiboe, M., 2011. A Platform-Independent Framework for Application Development for Smart Phones.

S209. MacVittie, D., 2004. Crossfire targets multiplatform development. Network Computing 15, 32–4.

S210. Madiraju, P., Malladi, S., Balasooriya, J., Hariharan, A., Prasad, S.K., Bourgeois, A., 2010. A methodology for engineering collaborative and ad-hoc mobile applications using SyD middleware. Journal of Network and Computer Applications 33, 542 – 555.

S211. Magdaleno, A.M., Werner, C.M.L., Araujo, R.M. de, 2012. Reconciling software development models: A quasi-systematic review. Journal of Systems and Software 85, 351 – 369.

S212. Maharmeh, M., Unhelkar, B., 2009. A Composite Software Framework Approach for Mobile Application Development. Handbook of research in mobile business: technical, methodological, and social perspectives 194.

S213. Maia, M.E.F., Celes, C., Castro, R., Andrade, R.M.C., 2010. Considerations on developing mobile applications based on the Capuchin project, in: Proceedings of the 2010 ACM Symposium on Applied Computing, SAC '10. ACM, New York, NY, USA, pp. 575–579.

S214. Makunga, L., Church, K., 2002. Software Development in Mobile Computing Applications. INFORMATION TECHNOLOGY ON THE MOVE 257.

S215. Malek, S., Edwards, G., Brun, Y., Tajalli, H., Garcia, J., Krka, I., Medvidovic, N., Mikic-Rakic, M., Sukhatme, G.S., 2010. An architecture-driven software mobility framework. Journal of Systems and Software 83, 972–989.

S216. Manninen, T., 2002. Contextual Virtual Interaction as Part of Ubiquitous Game Design and Development. Personal Ubiquitous Comput. 6, 390–406.

S217. Manjunatha, A., Ranabahu, A., Sheth, A., Thirunarayan, K., 2010. Power of clouds in your pocket: An efficient approach for cloud mobile hybrid application development, in: Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference On. pp. 496–503.

S218. March, V., Gu, Y., Leonardi, E., Goh, G., Kirchberg, M., Lee, B.S., 2011. µCloud: Towards a New Paradigm of Rich Mobile Applications. Procedia Computer Science 5, 618 – 624.

S219. Marinho, F.G., Andrade, R.M.C., Werner, C., Viana, W., Maia, M.E.F., Rocha, L.S., Teixeira, E., Filho, J.B.F., Dantas, V.L.L., Lima, F., Aguiar, S., 2012. MobiLine: A Nested Software Product Line for the domain of mobile and context-aware applications. Science of Computer Programming -.

S220. Marius, P., 2010. Audit Process during Projects for Development of New Mobile IT Application. Informatica Economica 14, 34–46.

S221. Martin, S., Diaz, G., Plaza, I., Ruiz, E., Castro, M., Peire, J., 2011. State of the art of frameworks and middleware for facilitating mobile and ubiquitous learning development. Journal of Systems and Software 84, 1883 – 1891.

S222. Martin, S., Diaz, G., Sancristobal, E., Gil, R., Castro, M., Peire, J., Boticki, I., 2010. M2Learn Open Framework: Developing Mobile Collaborative and Social Applications, in: UBICOMM 2010, The Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies. pp. 59–62.

S223. Mathew, J., 2010. Cross-Platform Application Development on Symbian.

S224. Matthews, M., Doherty, G., Coyle, D., Sharry, J., 2008. Designing mobile applications to support mental health interventions. Handbook of Research on User Interface Design and Evaluation for Mobile Technology 635–656.

S225. Mayuk, O., Torabi, T., 2006. Framework for Mobile Application Development and Content Integration, in: Wireless, Mobile and Ubiquitous Technology in Education, 2006. WMUTE '06. Fourth IEEE International Workshop On. pp. 69–73.

S226. Mazhelis, O., Markkula, J., Jakobsson, M., 2005. Specifying patterns for mobile application domain using general architectural components, in: Product Focused Software Process Improvement. 6th International Conference, PROFES 2005. Proceedings (Lecture Notes in Computer Science Vol. 3547).

S227. Meads, A., Warren, I., 2011. OdinTools–Model-Driven Development of Intelligent Mobile Services, in: Services Computing (SCC), 2011 IEEE International Conference On. pp. 448–455.

S228. Medvidovic, N., Edwards, G., 2010. Software architecture and mobility: A roadmap. Journal of Systems and Software 83, 885 – 898.

S229. Meijles, E., Rip, F., Bakker, M., Epema, G., 2005. Do we speak each others' language? A methodology for developing generic GI-competencies, in: 8th AGILE Conference on GI Science. F. Toppen and M. Painho. Estoril, Portugal, Universidade Nova De Lisboa, Lisboa, Portugal.

S230. Miravet, P., Marín, I., Ortín, F., Rionda, A., 2009. DIMAG: a framework for automatic generation of mobile applications for multiple platforms, in: Proceedings of the 6th International Conference on Mobile Technology, Application & Systems, Mobility '09. ACM, New York, NY, USA, pp. 23:1–23:8.

S231. Mishra, J., Dash, S.K., Dash, S., 2012. Mobile-Cloud: A Framework of Cloud Computing for Mobile Application. Advances in Computer Science and Information Technology. Computer Science and Information Technology 347–356.

S232. Mnaouer, A.B., Shekhar, A., Liang, Z.Y., 2004. A generic framework for rapid application development of mobile Web services with dynamic workflow management, in: Services Computing, 2004.(SCC 2004). Proceedings. 2004 IEEE International Conference On. pp. 165–171.

S233. Morales-Aranda, A.H., Mayora-Ibarra, O., Negrete-Yankelevich, S., 2004. M-Modeler: a framework implementation for modeling m-commerce applications, in: Proceedings of the 6th International Conference on Electronic Commerce. pp. 596–602.

S234. Motes, G., 2011. US Army Mobile Application Development: A Coder's Perspective. DTIC Document.

S235. Munson, J.P., Dewan, P., 1997. Sync: a Java framework for mobile collaborative applications. Computer 30, 59–66.

S236. Murthy, V.K., 2001. Seamless mobile transaction processing: Models, protocols and software tools, in: Parallel and Distributed Systems, 2001. ICPADS 2001. Proceedings. Eighth International Conference On. pp. 147–154.

S237. Naevdal, S., 2007. Agile development methodologies introduced to Norwegian ICT companies. (No. TDT4520). Norwegian University of Science and Technology.

S238. Natchetoi, Y., Kaufman, V., Shapiro, A., 2008. Service-oriented architecture for mobile applications, in: Proceedings of the 1st International Workshop on Software Architectures and Mobility, SAM '08. ACM, New York, NY, USA, pp. 27–32.

S239. Nguyen, N.T., 2010. How software process improvement standards and agile methods co-exist in software organisations?

S240. Northern, C., Mayfield, K., Benito, R., Casagni, M., 2011. Handbook for Implementing Agile in Department of Defense Information Technology Acquisition.

S241. Nugroho, L.E., 2001a. A context-based approach for mobile application development.

S242. Nugroho, L.E., 2001b. A specification language for mobile application development, in: Proceedings of 3rd International Conference on Information Integration and Web Based Applications and Services. (IIWAS 2001). Presented at the Third International Conference on Information Integration and Web-based Applications and Services., pp. 357–64.

S243. Nyström, A., 2011. Agile Solo-Defining and Evaluating an Agile Software Development Process for a Single Software Developer.

S244. O'Leary, P., Thiel, S., Botterweck, G., Richardson, I., 2008. Towards a product derivation process framework.

S245. ObjectGraph, L., 2010. Creating Mapping Applications for the iPhone. Cartographic Perspectives 71.

S246. Ocampo, A., Bella, F., MJ" nch, J., 2006. Software Development Processes. Developing Services for the Wireless Internet 9–32.

S247. Ogunleye, S., 2009. MobiNET: A framework for supporting Java mobile application developers through contextual inquiry.

S248. Olaniyi, O., Ajose, S., Adegoke, M., 2010. Development of a mobile airline reservation and payment system. International Journal of Electronic Finance 4, 372–389.

S249. Olivé, A., Cabot, J., 2007. A research agenda for conceptual schema-centric development. Conceptual Modelling in Information Systems Engineering 3, 319.

S250. Omar, S.H., 2000. A mobile code toolkit for adaptive mobile applications.

S251. Ortiz, G., Prado, A.G.D., 2010. Improving device-aware Web services and their mobile clients through an aspect-oriented, model-driven approach. Information and Software Technology 52, 1080 – 1093.

S252. Palviainen, M., Laakko, T., 2005. Using modular and generative approaches for implementing adaptable mobile browser applications, in: Proceedings of the IADIS International Conference WWW/Internet 2005. pp. 101–109.

S253. Papageorgiou, A., Leferink, B., Eckert, J., Repp, N., Steinmetz, R., 2009. Bridging the gaps towards structured mobile SOA, in: Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia, MoMM '09. ACM, New York, NY, USA, pp. 288–294.

S254. Paspallis, N., Papadopoulos, G.A., 2006. An approach for developing adaptive, mobile applications with separation of concerns, in: Computer Software and Applications Conference, 2006. COMPSAC'06. 30th Annual International. pp. 299–306.

S255. Patel, C., Ramachandran, M., 2010. Best Practices Guidelines for Agile Requirements Engineering Practices.

S256. Pauca, V.P., Guy, R.T., 2012. Mobile apps for the greater good: a socially relevant approach to software engineering, in: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, SIGCSE '12. ACM, New York, NY, USA, pp. 535–540.

S257. Pfleging, B., Valderrama Bahamondez, E. del C., Schmidt, A., Hermes, M., Nolte, J., 2010. MobiDev: a mobile development kit for combined paper-based and in-situ programming on the mobile phone, in: Proceedings of the 28th of the International Conference Extended Abstracts on Human Factors in Computing Systems, CHI EA '10. ACM, New York, NY, USA, pp. 3733–3738.

S258. Picco, G.P., Murphy, A.L., Roman, G.-C., 2000. Developing mobile computing applications with LIME, in: Proceedings of the 22nd International Conference

on Software Engineering, ICSE '00. ACM, New York, NY, USA, pp. 766–769.

S259. Pikkarainen, M., 2005. Agile Software Development of Embedded Systems Version: 1.0 Date: 2005.06. 13.

S260. Pikkarainen, M., Passoja, U., 2005. An approach for assessing suitability of agile solutions: A case study. Extreme Programming and Agile Processes in Software Engineering 1202–1206.

S261. Pikkarainen, M., Salo, O., Kuusela, R., Abrahamsson, P., 2011. Strengths and barriers behind the successful agile deployment—insights from the three software intensive companies in Finland. Empirical Software Engineering 1–28.

S262. Platzer, E., Petrovic, O., 2011. A learning environment for developers of mobile apps, in: Global Engineering Education Conference (EDUCON), 2011 IEEE. pp. 14–19.

S263. Pocatilu, P., Doinea, M., Ciurea, C., 2010. Development of distributed mobile learning systems, in: The 9th WSEAS International Conference on Circuits, Systems, Electronics, Control & Signal Processing (CSECS'10), Vouliagmeni, Athens, Greece.

S264. Pohl, T., Kothandaraman, R., Seshasai, V.S., 2007. Developing Mobile Applications Using SAP NetWever Mobile. SAP Press.

S265. Pokraev, S., Koolwaaij, J., van Setten, M., Broens, T., Costa, P.D., Wibbels, M., Ebben, P., Strating, P., 2005. Service platform for rapid development and deployment of context-Aware, mobile applications, in: Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference On.

S266. Polo, J., Delgado, J., 2005. An easy way to develop mobile and wireless applications. Presented at the The 7th IFIP International Conference on Mobile and Wireless Communications Networks, Marrakech, Marocco.

S267. Pulli, K., Vaarala, J., Miettinen, V., Aarnio, T., Callow, M., 2005. Developing mobile 3D applications with OpenGL ES and M3G, in: ACM SIGGRAPH 2005 Courses, SIGGRAPH '05. ACM, New York, NY, USA.

S268. Qin, Z., Zhang, J., Zhang, X., 2012. An Effective Partition Approach for Elastic Application Development on Mobile Cloud Computing. Advances in Grid and Pervasive Computing 46–53.

S269. Quinton, C., Mosser, S., Parra, C., Duchien, L., 2011. Using multiple feature models to design applications for mobile phones, in: Proceedings of the 15th International Software Product Line Conference, Volume 2, SPLC '11. ACM, New York, NY, USA, pp. 23:1–23:8.

S270. Rahimian, V., Ramsin, R., 2008. Designing an agile methodology for mobile software development: A hybrid method engineering approach, in: Research Challenges in Information Science, 2008. RCIS 2008. Second International Conference On. pp. 337–342.

S271. Ranabahu, A., Sheth, A., Manjunatha, A., Thirunarayan, K., 2010. Towards Cloud Mobile Hybrid Application Generation using Semantically Enriched Domain Specific Languages, in: International Workshop on Mobile Computing and Clouds (MobiCloud 2010).

S272. Ranabahu, A.H., Maximilien, E.M., Sheth, A.P., Thirunarayan, K., 2011. A domain specific language for enterprise grade cloud-mobile hybrid applications, in: Proceedings of the Compilation of the Co-located Workshops on DSM'11, TMC'11, AGERE!'11, AOOPES'11, NEAT'11, & VMIL'11, SPLASH '11 Workshops. ACM, New York, NY, USA, pp. 77–84.

S273. Rashid, O., Thompson, R., Coulton, P., Edwards, R., 2004. A comparative study of mobile application development in symbian and J2ME using example of a live football results service operating over GPRS, in: Consumer Electronics, 2004 IEEE International Symposium On. pp. 203 – 207.

S274. Reinhartz-Berger, I., 2003. Developing web applications with object-oriented approaches and object-process methodology.

S275. Ren, H., Duan, Z., 2012. The Study on Device Application Development and DataSynchonization. Procedia Engineering 29, 415–419.

S276. Rizvi, S., Hussain, S.Z., Hassan, S.I., 2011. Simplifying Mobile Application Development with Model-View-Controller, in: Proceedings of the 5th National Conference; INDIACom-2011. Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi, India.

S277. Rodger, R., 2011. Beginning Building Mobile Application Development in the Cloud. Wrox.

S278. Rogers, R., 2010. Developing portable mobile web applications. Linux J. 2010.

S279. Rogov, P., Borisov, N., 2007. Developing a Mobile Distance Learning System.

S280. Roman, G.C., Picco, G.P., Murphy, A.L., 2000. Software engineering for mobility: a roadmap, in: Proceedings of the Conference on the Future of Software Engineering. pp. 241–258.

S281. Rosa, R.E.V.S., Lucena,Jr., V.F., 2011. Smart composition of reusable software components in mobile application product lines, in: Proceedings of the 2nd International Workshop on Product Line Approaches in Software Engineering, PLEASE '11. ACM, New York, NY, USA, pp. 45–49.

S282. Rosado, D.G., Fernández-Medina, E., López, J., Piattini, M., 2011. Systematic design of secure Mobile Grid systems. Journal of Network and Computer Applications 34, 1168 – 1183.

S283. Rossi, M., Tuunanen, T., 2010. A method and tool for rapid consumer application development. International Journal of Organisational Design and Engineering 1, 109–125.

S284. Roth, J., 2005. The resource framework for mobile applications. Enterprise Information Systems V 300–307.

S285. Rukzio, E., Rohs, M., Wagner, D., Hamard, J., 2005. Development of interactive applications for mobile devices, in: Proceedings of the 7th International Conference on Human Computer Interaction with Mobile Devices and Services, MobileHCI '05. ACM, New York, NY, USA, pp. 365–366.

S286. Rusu, L., Sarbu, M., Podean, M., 2009. Multilayer solution using multimap for develope a mobile application, in: Proceedings of the International Conference on e-Business. Presented at the International Conference on e-Business, Milan, Italy, pp. 135–8.

S287. Saifudin, A.W.S.N., Salam, B.S., Abdullah, C.M.H.L., 2011. MMCD Framework and Methodology for Developing m-Learning Applications. Presented at the International conference on Teaching & Learning in Higher Education (ICTLHE 2011).

S288. Salim, A., Mehdi, Q., 2006. Investigation into Mobile Development Tools and Technology for Mobile Games and Application.

S289. Salo, O., 2004. Improving software process in agile software development projects: results from two XP case studies, in: Euromicro Conference, 2004. Proceedings. 30th. pp. 310–317.

S290. Salo, O., Abrahamsson, P., 2007. An iterative improvement process for agile software development. Software Process: Improvement and Practice 12, 81–100.

S291. Salvaneschi, G., Ghezzi, C., Pradella, M., 2012. Context-oriented programming: A software engineering perspective. Journal of Systems and Software 85, 1801 – 1817.

S292. Sambasivan, D., John, N., Udayakumar, S., Gupta, R., 2011. Generic framework for mobile application development, in: Internet (AH-ICI), 2011 Second Asian Himalayas International Conference On. pp. 1 –5.

S293. Sánchez, P., Jiménez, M., Rosique, F., Álvarez, B., Iborra, A., 2011. A framework for developing home automation systems: From requirements to code. Journal of Systems and Software 84, 1008 – 1021.

S294. Santi, A., Guidi, M., Ricci, A., 2010. Exploiting agent-oriented programming for developing Android applications, in: Proc. Of.

S295. Sato, D., Goldman, A., Kon, F., 2007. Tracking the evolution of object-oriented quality metrics on agile projects, in: Proceedings of the 8th International Conference on Agile Processes in Software Engineering and Extreme Programming, XP'07. Springer-Verlag, Berlin, Heidelberg, pp. 84–92.

S296. Satoh, I., 2000. MobileSpaces: A framework for building adaptive distributed applications using a hierarchical mobile agent system, in: Distributed Computing Systems, 2000. Proceedings. 20th International Conference On. pp. 161–168.

S297. Scharff, C., 2010. The Software Engineering of Mobile Application Development.

S298. Scharff, C., 2011. Guiding global software development projects using Scrum and Agile with quality assurance, in: Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference On. pp. 274–283.

S299. Scharff, C., Verma, R., 2010. Scrum to support mobile application development projects in a just-in-time learning context, in: Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, CHASE '10. ACM, New York, NY, USA, pp. 25–31.

S300. Schuster, C., Appeltauer, M., Hirschfeld, R., 2011. Context-oriented programming for mobile devices: JCop on Android, in: Proceedings of the 3rd International Workshop on Context-Oriented Programming, COP '11. ACM, New York, NY, USA, pp. 5:1–5:5.

S301. Schwieren, J., Vossen, G., 2009. A design and development methodology for mobile RFID applications based on the ID-Services middleware architecture, in: Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference On. pp. 260–266.

S302. Seifert, J., Pfleging, B., del Carmen Valderrama Bahamóndez, E., Hermes, M., Rukzio, E., Schmidt, A., 2011. Mobidev: a tool for creating apps on mobile phones, in: Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services, MobileHCI '11. ACM, New York, NY, USA, pp. 109–112.

S303. Sen, R., 2009. Developing Parallel Programs. TechEd Special Edition 17.

S304. Serhani, M.A., Benharref, A., Dssouli, R., Mizouni, R., 2009. Toward an Efficient Framework for Designing, Developing, and Using Secure Mobile Applications. the Proceedings of World Academy of Science, Engineering and Technology 40.

S305. Serm, T., Blanchfield, P., Su, K., 2006. Mobile Newspaper Development Framework: Guidelines for newspaper companies for creating usable mobile news portals, in: Computing & Informatics, 2006. ICOCI'06. International Conference On. pp. 1–8.

S306. Serral, E., Valderas, P., Pelechano, V., 2010. Towards the Model Driven Development of context-aware pervasive systems. Pervasive and Mobile Computing 6, 254 – 280.

S307. Session 4 Abstract: Mobile Application Development, 2008. , in: Mobile Business, 2008. ICMB '08. 7th International Conference On. p. xv.

S308. Shah, M., Mears, B., Chakrabarti, C., Spanias, A., Center, S., Tempe, A., 2012. A Top-Down Design Methodology Using Virtual Platforms for Concept Development.

S309. Shen, J., Sun, P., Guo, C., Yin, Y., Song, S., 2005. Delivering mobile enterprise applications on iMMS

framework, in: Proceedings of the 6th International Conference on Mobile Data Management, MDM '05. ACM, New York, NY, USA, pp. 289–293.

S310. Shen, M., Yang, W., Rong, G., Shao, D., 2012. Applying Agile Methods to Embedded Software Development: A Systematic Review. Presented at the The 2nd International Workshop on Software Engineering for Embedded Systems, Zurich, Switzerland.

S311. Shetty, K.S., Singh, S., 2011. Cloud Based Application Development for Mobile Devices for Accessing LBS. Advances in Parallel Distributed Computing 532–543.

S312. Shiratuddin, N., Sarif, S.M., 2008. m d-Matrix: Mobile Application Development Tool. Proceedings of the International MultiConference of Engineers and Computer Scientists 1.

S313. Shiratuddin, N., Sarif, S.M., 2009. Construction of Matrix and eMatrix for Mobile Development Methodologies. Handbook of research in mobile business: technical, methodological, and social perspectives.

S314. Shrestha, A., 2010. MobileSOA Framework for Context-Aware Mobile Applications, in: Proceedings of the 2010 Eleventh International Conference on Mobile Data Management, MDM '10. IEEE Computer Society, Washington, DC, USA, pp. 297–298.

S315. Simon, R., Fröhlich, P., 2007. A mobile application framework for the geospatial web, in: Proceedings of the 16th International Conference on World Wide Web, WWW '07. ACM, New York, NY, USA, pp. 381–390.

S316. Simonsen, A., 2004. Developing mobile applications.

S317. Simula, K., 2007. Intelligent software agent framework for customized mobile services, in: Proceedings of the 4th on Middleware Doctoral Symposium, MDS '07. ACM, New York, NY, USA, pp. 15:1–15:6.

S318. Singh, M., Rahmatabadi, G.Y., Ahamed, S.I., 2004. User Interface and application development experience on handheld devices, in: Electro/Information Technology Conference, 2004. EIT 2004. IEEE. pp. 125–137.

S319. Soroker, D., Cáceres, R., Dig, D., Schade, A., Spraragen, S., Tiwari, A., 2006. Pegboard: a framework for developing mobile applications, in: Proceedings of the 4th International Conference on Mobile Systems, Applications and Services, MobiSys '06. ACM, New York, NY, USA, pp. 138–150.

S320. Soumaya, D., Tabbane, M.S., Jemai, M.A., 2007. Development of a software mobile banking solution for S60 phones.

S321. Spataru, A.C., 2010. Agile development methods for mobile applications.

S322. Srinivasa, K.G., Harish Raddi, C.S., Mohan Krishna, S.H., Venkatesh, N., 2011. MeghaOS: Cloud based operating system and a framework for mobile application development, in: Information and Communication Technologies (WICT), 2011 World Congress On. pp. 858 –863.

S323. Srinivasan, J., Dobrin, R., Lundqvist, K., 2009. "State of the Art"in Using Agile Methods for Embedded Systems Development, in: Computer Software and Applications Conference, 2009. COMPSAC'09. 33rd Annual IEEE International. pp. 522–527.

S324. Su, S.H., Scharff, C., 2010. Know Yourself and Beyond: A Global Software Development Project Experience with Agile Methodology, in: Proceedings of Student-Faculty Research Day, CSIS. Pace University.

S325. Sung, M., Lee, J., 2004. Desirable mobile networking method for formulating an efficient mobile conferencing application. Embedded and Ubiquitous Computing 46–151.

S326. Tang, L., Yu, Z., Zhou, X., Wang, H., Becker, C., 2011. Supporting rapid design and evaluation of pervasive applications: challenges and solutions. Personal Ubiquitous Comput. 15, 253–269.

S327. Tanuan, M., 2007. Using Sybase WorkSpace to build service oriented architecture (SOA) applications quickly, in: Companion to the 22nd ACM SIGPLAN Conference on Object-oriented Programming Systems and Applications Companion, OOPSLA '07. ACM, New York, NY, USA, pp. 848–849.

S328. Tarnacha, A., Maitland, C.F., 2006. Entrepreneurship in mobile application development, in: Proceedings of the 8th International Conference on Electronic Commerce: The New E-commerce: Innovations for Conquering Current Barriers, Obstacles and Limitations to Conducting Successful Business on the Internet, ICEC '06. ACM, New York, NY, USA, pp. 589–593.

S329. Teng, C.C., Helps, R., 2010. Mobile Application Development: Essential New Directions for IT, in: Information Technology: New Generations (ITNG), 2010 Seventh International Conference On. pp. 471 – 475.

S330. Terani, N.S., 2012. IPhone Application Development Challenges and Solutions.

S331. Thompson, C., White, J., Dougherty, B., Schmidt, D., 2009. Optimizing mobile application performance with model–driven engineering. Software Technologies for Embedded and Ubiquitous Systems 36–46.

S332. Thompson, C., White, J., Dougherty, B., Turner, H., Campbell, S., Zienkiewicz, K., Schmidt, D.C., 2010. Model-Driven Architectures for Optimizing Mobile Application Performance.

S333. Titica, D., Fratu, O., Stanescu, E., Halunga-Fratu, S., 2007. Simple Location-based Application Development for Mobile Phones, in: Telecommunications in Modern Satellite, Cable and Broadcasting Services, 2007. TELSIKS 2007. 8th International Conference On. pp. 15–18.

S334. TRIF, S., VIŞOIU, A., 2011. A Windows Phone 7 Oriented Secure Architecture for Business Intelligence

Mobile Applications. Informatica Economică 15, 119–129.

S335. Ueyama, J., Pinto, V.P.V., Madeira, E.R.M., Grace, P., Jonhson, T.M.M., Camargo, R.Y., 2009. Exploiting a generic approach for constructing mobile device applications, in: Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE, COMSWARE '09. ACM, New York, NY, USA, pp. 12:1–12:12.

S336. Um, J., Hong, S., Kim, Y.T., Chung, E., Choi, K.M., Kong, J.T., Eo, S.K., 2005. ViP: A Practical Approach to Platform-based System Modeling Methodology. Journal of Semiconductor Technology and Science 5, 89.

S337. Unhelkar, B., Murugesan, S., 2010. The Enterprise Mobile Applications Development Framework. IT professional 12, 33–39.

S338. Vara, J.M., Marcos, E., 2012. A framework for model-driven development of information systems: Technical decisions and lessons learned. Journal of Systems and Software -.

S339. Vazquez-Briseno, M., Vincent, P., Nieto-Hipolito, J.I., de Dios Sanchez-Lopez, J., 2012. Applying a Modular Framework to Develop Mobile Applications and Services. Journal of Universal Computer Science 18, 704–727.

S340. Viana, W., Andrade, R., 2008. XMobile: A MB-UID environment for semi-automatic generation of adaptive applications for mobile devices. Journal of Systems and Software 81, 382–394.

S341. Walkerdine, J., Phillips, P., Lock, S., 2009. A Tool Supported Methodology For Developing Secure Mobile P2P Systems, in: Mobile Peer-to-peer Computing for Next Generation Distributed Environments: Advancing Conceptual and Algorithmic Applications. pp. 283–301.

S342. Walter, T., Bussard, L., Roudier, Y., Haller, J., Kilian-Kehr, R., Posegga, J., Robinson, P., 2004. Secure mobile business applications – framework, architecture and implementation. Information Security Technical Report 9, 6 – 21.

S343. Wang, A., Sørensen, C.F., Ramampiaro, H., Le, H., Conradi, R., Nyg\aard, M., 2005. Using the MOWAHS characterisation framework for development of mobile work applications. Product Focused Software Process Improvement 111–127.

S344. Wang, M., Hunger, I.A., 2007. Support Agile Development Process: Exploring Windows Presentation Foundation Technology Under the Conceptual Framework of Model-View-Controller.

S345. Wang, Y., 2004. An FSM model for situation-aware mobile application software systems, in: Proceedings of the 42nd Annual Southeast Regional Conference, ACM-SE 42. ACM, New York, NY, USA, pp. 52–57.

S346. Wasserman, A.I., 2010. Software engineering issues for mobile application development, in: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, FoSER '10. ACM, New York, NY, USA, pp. 397–400.

S347. Weerasekera, P., Abeysinghe, S., 2005. Modular mobile application development framework for resource constrained devices.

S348. Wesson, J.L., van der Walt, D.F., 2005. Implementing mobile services: does the platform really make a difference?, in: Proceedings of the 2005 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries, SAICSIT '05. South African Institute for Computer Scientists and Information Technologists, Republic of South Africa, pp. 208–216.

S349. Wichmann, D., Pielot, M., Boll, S., 2009. Companion Platform-Modular Software Platform for Rapid Development of Mobile Applications. IT-Information Technology 51, 72–78.

S350. Wikman, J., Nurminen, J.K., 2008. Open Source Web Application Development Stack for Symbian-Based Mobile Phones, in: Next Generation Mobile Applications, Services and Technologies, 2008. NGMAST'08. The Second International Conference On. pp. 607–612.

S351. Wolkerstorfer, P., Tscheligi, M., Sefelin, R., Milchrahm, H., Hussain, Z., Lechner, M., Shahzad, S., 2008. Probing an agile usability process, in: CHI '08 Extended Abstracts on Human Factors in Computing Systems, CHI EA '08. ACM, New York, NY, USA, pp. 2151–2158.

S352. Wooldridge, D., Schneider, M., 2011. Keys to the Kingdom: The App Store Submission Process, in: The Business of iPhone and iPad App Development. pp. 353–398.

S353. Xiong, Y., Wang, A., 2010. A new combined method for UCD and software development and case study, in: Information Science and Engineering (ICISE), 2010 2nd International Conference On. pp. 1–4.

S354. Yang, B., 2009. Design and implementation of a novel mobile application for SMS on demand, in: Management of e-Commerce and e-Government, 2009. ICMECG'09. International Conference On. pp. 412–415.

S355. Yang, K., Todd, C., Ou, S., 2006. Model-based service discovery for future generation mobile systems, in: Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing, IWCMC '06. ACM, New York, NY, USA, pp. 973–978.

S356. Yao, L., 2003. An Adaptive Mobile Application Development Framework (CITATION) (Master thesis).

S357. Yu, P., Yu, H., 2004. Lessons learned from the practice of mobile health application development, in: Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International. pp. 58–59.

S358. Yuen, S.L., 2003. Postponement Strategies for Mobile Application Development–A Framework. BLED 2003 Proceedings 45.

S359. Zabri, S.N., Awang, A.H., Salahuddin, L., Said, M.M., 2011. Application development with J2ME for mobile phone, in: Advanced Communication Technology (ICACT), 2011 13th International Conference On. pp. 1420 –1423.

S360. Zakál, D., Lengyel, L., 2010. Feature model-driven software development.

S361. Zakal, D., Lengyel, L., Charaf, H., 2011. Software Product Lines-based development, in: Applied Machine Intelligence and Informatics (SAMI), 2011 IEEE 9th International Symposium On. pp. 79–81.

S362. Zeidler, C., Kittl, C., Petrovic, O., 2008. An integrated product development process for mobile software. International Journal of Mobile Communications 6, 345–356.

S363. Zheng, P., Ni, L., 2006. Mobile Application Challenges, in: Smart Phone and Next Generation Mobile Computing. Morgan Kaufmann, Burlington, pp. 407 – 512.

S364. Zimmerman, J.B., 1999. Mobile Computing: Characteristics, Business benefits, and the mobile framework. University of Maryland, Browie state, INSS 960.

Appendix B – Papers selected for the SLR Phase 3 analysis

S1. Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., Koskela, J., Kyllönen, P., Salo, O., 2004. Mobile-D: an agile approach for mobile application development, in: Companion to the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages, and Applications, OOPSLA '04. ACM, New York, NY, USA, pp. 174–175.

S2. Abrahamsson, P., Hanhineva, A., Jäälinoja, J., 2005. Improving business agility through technical solutions: A case study on test-driven development in mobile software development, in: Business Agility and Information Technology Diffusion. Presented at the IFIP TC8 WG 8.6 International Working Conference.

S3. Abrahamsson, P., Ihme, T., Kolehmainen, K., Kyllönen, P., Salo, O., 2009. Mobile-D for Mobile Software: How to Use Agile Approaches for the Efficient Development of Mobile Applications.

S4. Ahtinen, A., Nurminen, J.K., Häkkilä, J., 2007. Developing a mobile reporting system for road maintenance: user research perspective, in: Proceedings of the 4th International Conference on Mobile Technology, Applications, and Systems and the 1st International Symposium on Computer Human Interaction in Mobile Technology, Mobility '07. ACM, New York, NY, USA, pp. 1–7.

S5. Alyani, N., Shirzad, S., 2011. Learning to innovate in distributed mobile application development: Learning episodes from Tehran and London, in: 2011 Federated Conference on Computer Science and Information Systems (FedCSIS). Presented at the 2011 Federated Conference on Computer Science and Information Systems (FedCSIS). IEEE., Piscataway, NJ, USA, pp. 497–504.

S6. Balagtas-Fernandez, F.T., Hussmann, H., 2008. Model-Driven Development of Mobile Applications, in: Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering, ASE '08. IEEE Computer Society, Washington, DC, USA, pp. 509–512.

S7. Barnawi, A., Qureshi, M., Khan, A.I., 2012. A Framework for Next Generation Mobile and Wireless Networks Application Development using Hybrid Component Based Development Model. Arxiv preprint arXiv:1202.2515.

S8. Bergström, F., Engvall, G., 2011. Development of handheld mobile applications for the public sector in Android and iOS using agile Kanban process tool.

S9. Binsaleh, M., Hassan, S., 2011. Systems Development Methodology for Mobile Commerce Applications: Agile vs. Traditional. International Journal of Online Marketing (IJOM) 1, 33–47.

S10. Biswas, A., Donaldson, T., Singh, J., Diamond, S., Gauthier, D., Longford, M., 2006. Assessment of mobile experience engine, the development toolkit for context aware mobile applications, in: Proceedings of the 2006 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, ACE '06. ACM, New York, NY, USA.

S11. Bowen, J., Hinze, A., 2011. Supporting Mobile Application Development with Model-Driven Emulation. Electronic Communications of the EASST 45.

S12. Charaf, H., 2011. Developing Mobile Applications for Multiple Platforms, in: Engineering of Computer Based Systems (ECBS-EERC), 2011 2nd Eastern European Regional Conference on The. p. 2.

S13. Chen, M., 2004. A methodology for building mobile computing applications. International journal of electronic business 2, 229–243.

S14. Cuccurullo, S., Francese, R., Risi, M., Tortora, G., 2011. A Visual Approach supporting the Development of MicroApps on Mobile Phones, in: Proc. of 3rd International Symposium on End-User Development. Presented at the 3rd International Symposium on End-User Development, Brindisi, Italy, pp. 289–294.

S15. Ejlersen, A., Knudsen, M.S., Løvgaard, J., Sørensen, M.B., 2008. Using Design Science to Develop a Mobile Application.

S16. Fjellheim, T., Milliner, S., Dumas, M., Vayssière, J., 2007. A process-based methodology for designing event-based mobile composite applications. Data & Knowledge Engineering 61, 6 – 22.

S17. Forstner, B., Lengyel, L., Kelenyi, I., Levendovszky, T., Charaf, H., 2005. Supporting Rapid Application Development on Symbian Platform, in: Computer as a Tool, 2005. EUROCON 2005.The International Conference On. pp. 72 –75.

S18. Forstner, B., Lengyel, L., Levendovszky, T., Mezei, G., Kelenyi, I., Charaf, H., 2006. Model-based system development for embedded mobile platforms, in: Model-Based Development of Computer-Based Systems and Model-Based Methodologies for Pervasive and Embedded Software, 2006. MBD/MOMPES 2006. Fourth and Third International Workshop On. p. 10–pp.

S19. Gal, V., Topol, A., 2005. Experimentation of a Game Design Methodology for Mobile Phones Games.

S20. Hedberg, H., Iisakka, J., 2006. Technical Reviews in Agile Development: Case Mobile-D, in: Quality Software, 2006. QSIC 2006. Sixth International Conference On. pp. 347–353.

S21. Ihme, T., Abrahamsson, P., 2005. The Use of Architectural Patterns in the Agile Software Development of Mobile Applications.

S22. Jeong, Y.J., Lee, J.H., Shin, G.S., 2008. Development Process of Mobile Application SW Based on Agile

Methodology, in: Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference On. pp. 362–366.

S23. Kaariainen, J., Koskela, J., Abrahamsson, P., Takalo, J., 2004. Improving requirements management in extreme programming with tool support - an improvement attempt that failed, in: Euromicro Conference, 2004. Proceedings. 30th. pp. 342 – 351.

S24. Kangas, E., Kinnunen, T., 2005. Applying user-centered design to mobile application development. Communications of the ACM 48, 55–59.

S25. Khambati, A., Grundy, J., Warren, J., Hosking, J., 2008. Model-Driven Development of Mobile Personal Health Care Applications, in: Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering, ASE '08. IEEE Computer Society, Washington, DC, USA, pp. 467–470.

S26. Khan, U.A., 2008. Improved Iterative Software Development Method for Game Design.

S27. Kim, H., Choi, B., Yoon, S., 2009. Performance testing based on test-driven development for mobile applications, in: Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication, ICUIMC '09. ACM, New York, NY, USA, pp. 612–617.

S28. Kim, H.K., 2008. Frameworks of Process Improvement for Mobile Applications. Engineering Letters 16.

S29. Kim, W.Y., Son, H.S., Kim, J.S., Kim, R.Y., 2010. Development of Windows Mobile Applications using Model Transformation Techniques. Journal of KISS: Computing Practices 16, 1091–5.

S30. Korkala, M., Abrahamsson, P., 2004. Extreme programming: Reassessing the requirements management process for an offsite customer. Software Process Improvement 12–22.

S31. Kurschl, W., Mitsch, S., Prokop, R., Schonbock, J., 2007. Gulliver - a framework for building smart speech-based applications, in: Proceedings of the 40th Annual Hawaii International Conference on System Sciences. Waikoloa, HI, USA.

S32. La, H.J., Lee, H.J., Kim, S.D., 2011. An efficiency-centric design methodology for mobile application architectures, in: Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference On. pp. 272–279.

S33. Madiraju, P., Malladi, S., Balasooriya, J., Hariharan, A., Prasad, S.K., Bourgeois, A., 2010. A methodology for engineering collaborative and ad-hoc mobile applications using SyD middleware. Journal of Network and Computer Applications 33, 542 – 555.

S34. Maharmeh, M., Unhelkar, B., 2009. A Composite Software Framework Approach for Mobile Application Development. Handbook of research in mobile business: technical, methodological, and social perspectives 194.

S35. Maia, M.E.F., Celes, C., Castro, R., Andrade, R.M.C., 2010. Considerations on developing mobile applications based on the Capuchin project, in: Proceedings of the 2010 ACM Symposium on Applied Computing, SAC '10. ACM, New York, NY, USA, pp. 575–579.

S36. Makunga, L., Church, K., 2002. Software Development in Mobile Computing Applications. INFORMATION TECHNOLOGY ON THE MOVE 257.

S37. Manjunatha, A., Ranabahu, A., Sheth, A., Thirunarayan, K., 2010. Power of clouds in your pocket: An efficient approach for cloud mobile hybrid application development, in: Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference On. pp. 496–503.

S38. Marinho, F.G., Andrade, R.M.C., Werner, C., Viana, W., Maia, M.E.F., Rocha, L.S., Teixeira, E., Filho, J.B.F., Dantas, V.L.L., Lima, F., Aguiar, S., 2012. MobiLine: A Nested Software Product Line for the domain of mobile and context-aware applications. Science of Computer Programming -.

S39. Nyström, A., 2011. Agile Solo - Defining and Evaluating an Agile Software Development Process for a Single Software Developer.

S40. Ortiz, G., Prado, A.G.D., 2010. Improving device-aware Web services and their mobile clients through an aspect-oriented, model-driven approach. Information and Software Technology 52, 1080 – 1093.

S41. Paspallis, N., Papadopoulos, G.A., 2006. An approach for developing adaptive, mobile applications with separation of concerns, in: Computer Software and Applications Conference, 2006. COMPSAC'06. 30th Annual International. pp. 299–306.

S42. Pauca, V.P., Guy, R.T., 2012. Mobile apps for the greater good: a socially relevant approach to software engineering, in: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, SIGCSE '12. ACM, New York, NY, USA, pp. 535–540.

S43. Rahimian, V., Ramsin, R., 2008. Designing an agile methodology for mobile software development: A hybrid method engineering approach, in: Research Challenges in Information Science, 2008. RCIS 2008. Second International Conference On. pp. 337–342.

S44. Ranabahu, A.H., Maximilien, E.M., Sheth, A.P., Thirunarayan, K., 2011. A domain specific language for enterprise grade cloud-mobile hybrid applications, in: Proceedings of the Compilation of the Co-located Workshops on DSM'11, TMC'11, AGERE!'11, AOOPES'11, NEAT'11, & VMIL'11, SPLASH '11 Workshops. ACM, New York, NY, USA, pp. 77–84.

S45. Rosa, R.E.V.S., Lucena,Jr., V.F., 2011. Smart composition of reusable software components in mobile application product lines, in: Proceedings of the 2nd International Workshop on Product Line Approaches in Software Engineering, PLEASE '11. ACM, New York, NY, USA, pp. 45–49.

S46. Rossi, M., Tuunanen, T., 2010. A method and tool for rapid consumer application development. International Journal of Organisational Design and Engineering 1, 109–125.

S47. Rupnik, R., 2009. Mobile Applications Development Methodology, in: Unhelkar, B. (Ed.), Handbook of Research in Mobile Business: Technical, Methodological, and Social Perspectives. IGI Global Snippet.

S48. Saifudin, A.W.S.N., Salam, B.S., Abdullah, C.M.H.L., 2011. MMCD Framework and Methodology for Developing m-Learning Applications. Presented at the International conference on Teaching & Learning in Higher Education (ICTLHE 2011).

S49. Salo, O., 2004. Improving software process in agile software development projects: results from two XP case studies, in: Euromicro Conference, 2004. Proceedings. 30th. pp. 310–317.

S50. Scharff, C., 2010. The Software Engineering of Mobile Application Development.

S51. Scharff, C., 2011. Guiding global software development projects using Scrum and Agile with quality assurance, in: Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference On. pp. 274–283.

S52. Scharff, C., Verma, R., 2010. Scrum to support mobile application development projects in a just-in-time learning context, in: Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, CHASE '10. ACM, New York, NY, USA, pp. 25–31.

S53. Schwieren, J., Vossen, G., 2009. A design and development methodology for mobile RFID applications based on the ID-Services middleware architecture, in: Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference On. pp. 260–266.

S54. Shah, M., Mears, B., Chakrabarti, C., Spanias, A., Center, S., Tempe, A., 2012. A Top-Down Design Methodology Using Virtual Platforms for Concept Development.

S55. Shiratuddin, N., Sarif, S.M., 2008. m d-Matrix: Mobile Application Development Tool. Proceedings of the International MultiConference of Engineers and Computer Scientists 1.

S56. Shiratuddin, N., Sarif, S.M., 2009. Construction of Matrix and eMatrix for Mobile Development Methodologies, in: Handbook of Research in Mobile Business: Technical, Methodological, and Social Perspectives. IGI Global, pp. 113–126.

S57. Simonsen, A., 2004. Developing mobile applications.

S58. Su, S.H., Scharff, C., 2010. Know Yourself and Beyond: A Global Software Development Project Experience with Agile Methodology, in: Proceedings of Student-Faculty Research Day, CSIS. Pace University.

S59. Terani, N.S., 2012. IPhone Application Development Challenges and Solutions. CALIFORNIA STATE UNIVERSITY.

S60. Thompson, C., White, J., Dougherty, B., Turner, H., Campbell, S., Zienkiewicz, K., Schmidt, D.C., 2010. Model-Driven Architectures for Optimizing Mobile Application Performance.

S61. Um, J., Hong, S., Kim, Y.T., Chung, E., Choi, K.M., Kong, J.T., Eo, S.K., 2005. ViP: A Practical Approach to Platform-based System Modeling Methodology. Journal of Semiconductor Technology and Science 5, 89.

S62. Walkerdine, J., Phillips, P., Lock, S., 2009. A Tool Supported Methodology For Developing Secure Mobile P2P Systems, in: Mobile Peer-to-peer Computing for Next Generation Distributed Environments: Advancing Conceptual and Algorithmic Applications. pp. 283–301.

S63. Wasserman, A.I., 2010. Software engineering issues for mobile application development, in: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, FoSER '10. ACM, New York, NY, USA, pp. 397–400.

S64. Wolkerstorfer, P., Tscheligi, M., Sefelin, R., Milchrahm, H., Hussain, Z., Lechner, M., Shahzad, S., 2008. Probing an agile usability process, in: CHI '08 Extended Abstracts on Human Factors in Computing Systems, CHI EA '08. ACM, New York, NY, USA, pp. 2151–2158.

S65. Xiong, Y., Wang, A., 2010. A new combined method for UCD and software development and case study, in: Information Science and Engineering (ICISE), 2010 2nd International Conference On. pp. 1–4.

S66. Zakal, D., Lengyel, L., Charaf, H., 2011. Software Product Lines-based development, in: Applied Machine Intelligence and Informatics (SAMI), 2011 IEEE 9th International Symposium On. pp. 79–81.

S67. Zeidler, C., Kittl, C., Petrovic, O., 2008. An integrated product development process for mobile software. International Journal of Mobile Communications 6, 345–356.

Appendix C – Study quality assessment table

ID	Quality assessment question	Possible results
Q1	Study reports methodology or approach used in mobile application development?	Yes/No
Q2	Study defines new methodology or approach for mobile applications development?	Yes/No
Q3	Research design is appropriate to address the study context?	Yes/Partially/No
Q4	Researches have experience in software development and mobile applications development?	Yes/Partially/No
Q5	The reported or created process is clearly defined to the applicable level?	Yes/Partially/No
Q6	The study provided value for research and practice?	Yes/Partially/No

Study / Question	Q1	Q2	Q3	Q4	Q5	Q6	Score
(Charaf, 2011)	Yes	No	Yes	Yes	Partially	Partially	3.0
(Alyani and Shirzad, 2011)	Yes	Yes	Partially	Yes	Partially	Partially	2.5
(Maharmeh and Unhelkar, 2009)	No	Yes	Partially	Yes	Partially	Yes	3.0
(Schwieren and Vossen, 2009)	No	Yes	No	Partially	No	No	0.5
(Ranabahu et al., 2011)	No	No					
(Barnawi et al., 2012)	No	Yes	Yes	Yes	Yes	Yes	4.0
(Rossi and Tuunanen, 2010)	No	No					
(Chen, 2004)	No	Yes	Yes	Yes	Yes	Yes	4.0
(Madiraju et al., 2010)	No	No					
(Xiong and Wang, 2010)	No	Yes	Yes	Yes	Partially	Partially	3.0
(Fjellheim et al., 2007)	No	No					
(Walkerdine et al., 2009)	No	Yes	Yes	Yes	Partially	Partially	3.0
(Shah et al., 2012)	No	No					
(Cuccurullo et al., 2011)	No	Yes	Partially	Yes	Partially	Partially	2.5
(Nyström, 2011)	Yes	Yes	Partially	Partially	Partially	Partially	2.0
(Paspallis and Papadopoulos, 2006)	No	No					
(La et al., 2011)	No	No					
(Zeidler et al., 2008)	No	Yes	Partially	Yes	Partially	No	2.0
(Kangas and Kinnunen, 2005)	No	No					
(Biswas et al., 2006)	No	Yes	Yes	Yes	Yes	Partially	3.0
(Maia et al., 2010)	No	Yes	No	Yes	No	No	1.0
(Shiratuddin and Sarif, 2009)	Yes	No	Yes	Yes	No	No	2.0
(Rahimian and Ramsin, 2008)	No	Yes	Yes	Yes	Partially	Partially	3.0
(Ahtinen et al., 2007)	No	No					
(Simonsen, 2004)	No	No					
(Bergström and Engvall, 2011)	Yes	No	Partially	Partially	Partially	No	1.5
(Kim et al., 2010)	No	No					
(Jeong et al., 2008)	No	Yes	Yes	Yes	Partially	No	2.5
(Korkala and Abrahamsson, 2004)	Yes	No	Yes	Yes	Partially	Partially	3.0
(Gal and Topol, 2005)	Yes	No	Yes	Yes	Partially	Partially	3.0
(Kim, 2008)	Yes	No	Partially	Yes	Partially	Partially	2.5
(Scharff, 2011)	Yes	No	Yes	Yes	No	No	2.0
(Kurschl et al., 2007)	No	No					
(Khan, 2008)	No	No					
(Abrahamsson et al., 2005b)	Yes	No	Yes	Yes	Yes	Yes	4.0
(Ortiz and Prado, 2010)	Yes	No	Yes	Yes	Partially	Partially	3.0
(Kaariainen et al., 2004)	Yes	No	Partially	Yes	Partially	No	2.0
(Salo, 2004)	Yes	No	Partially	Yes	Partially	No	3.0
(Terani, 2012)	No	No					
(Su and Scharff, 2010)	Yes	No	Partially	Partially	Partially	Partially	2.0
(Shiratuddin and Sarif, 2008)	Yes	No	Yes	Yes	Partially	No	2.5
(Saifudin et al., 2011)	No	Yes	Partially	Yes	No	No	1.5
(Rupnik, 2009)	No	Yes	Partially	Partially	Partially	No	1.5

(Pauca and Guy, 2012)	Yes	No	No	Yes	No	No	1.0
(Abrahamsson et al., 2009)	No	Yes	No	Yes	No	No	1.0
(Abrahamsson et al., 2004)	No	Yes	Partially	Yes	Partially	Partially	2.5
(Marinho et al., 2012)	No	Yes	Yes	Yes	Yes	Yes	4.0
(Forstner et al., 2006)	Yes	No	Partially	Yes	Partially	Partially	2.5
(Thompson et al., 2010)	Yes	No	No	Yes	No	No	1.0
(Balagtas-Fernandez and Hussmann, 2008)	No	No					
(Khambati et al., 2008)	Yes	No	Partially	Yes	Partially	Partially	2.5
(Kim et al., 2009)	Yes	No	Partially	Yes	No	No	1.5
(Manjunatha et al., 2010)	No	Yes	Partially	Yes	Partially	Partially	2.5
(Wolkerstorfer et al., 2008)	No	Yes	Partially	Yes	Partially	No	2.0
(Scharff and Verma, 2010)	Yes	No	Yes	Yes	Partially	No	2.5
(Rosa and Lucena,Jr., 2011)	Yes	No	Partially	Yes	Partially	No	2.0
(Makunga and Church, 2002)	No	No					
(Wasserman, 2010)	No	No					
(Zakal et al., 2011)	Yes	No	Partially	Yes	Partially	No	2.0
(Bowen and Hinze, 2011)	No	No					
(Forstner et al., 2005)	Yes	No	Partially	Yes	Partially	No	2.0
(Binsaleh and Hassan, 2011)	Yes	Yes	Yes	Yes	Yes	Yes	4.0
(Hedberg and Iisakka, 2006)	Yes	Yes	Yes	Yes	Yes	Yes	4.0
(Scharff, 2010)	Yes	No	Partially	Yes	Partially	Partially	2.5
(Ejlersen et al., 2008)	Yes	No	Yes	Yes	Partially	Partially	3.0
(Ihme and Abrahamsson, 2005)	Yes	No	Partially	Yes	Partially	Partially	3.5
(Um et al., 2005)	No	Yes	Yes	Yes	Yes	Yes	4.0

The study quality score is calculated by summarizing the columns Q3 to Q6 by valuing each positive answer (*Yes*) with score of 1 and each answer *Partially* with the score of 0.5.

Appendix D – Filled data forms for the SLR

Data item	Value	Notes
Study identifier	(Charaf, 2011)	
Title	Developing Mobile Applications Using SAP NetWever Mobile	
Publication details	T. Pohl, R. Kothandaraman, and V. S. Seshasai. Developing Mobile Applications Using SAP NetWever Mobile. SAP Press, 2007.	
Study type	Approach usage	
Name of methodology / approach	Model Driven Development	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	This paper introduces the problem of the software development for incompatible mobile platforms. Moreover, it provides a Model-Driven Architecture (MDA) and Domain Specific Modeling Language (DSML)-based solution.	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	Usage of: Visual Modeling and Transformation System (VMTS)	
Organizational aspects on implementation	None	
Project management aspects on implementation	None	

Data item	Value	Notes
Study identifier	(Alyani and Shirzad, 2011)	
Title	Learning to innovate in distributed mobile application development: Learning episodes from Tehran and London	
Publication details	N. Alyani and S. Shirzad, "- Learning to innovate in distributed mobile application development: Learning episodes from Tehran and London," in 2011 Federated Conference on Computer Science and Information Systems (FedCSIS)., Piscataway, NJ, USA, 2011, pp. 497–504.	
Study type	Methodology usage / New Methodology	
Name of methodology / approach	Scrum / DEAL	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	At the heart of the activities however, we noted a range of processes which we labeled as DEAL, as an acronym that stands for the cycle of Design, Execute, Adjust and Learn. Within the DEAL model, various activities were enhanced via formal and informal knowledge brokering and knowledge sourcing.	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	Usage: Real life projects in several years long period. Proposal: No	
Organizational aspects on implementation	Small and medium sized companies are reffered	
Project management	None	

aspects on implementation		

Data item	Value	Notes
Study identifier	(Maharmeh and Unhelkar, 2009)	
Title	A Composite Software Framework Approach for Mobile Application Development	
Publication details	M. Maharmeh and B. Unhelkar, "A Composite Software Framework Approach for Mobile Application Development," Handbook of research in mobile business: technical, methodological, and social perspectives, p. 194, 2009.	
Study type	New approach	
Name of methodology / approach	Composite Application Software Development Process Framework (CASDPF)	
Application in multi-platform development	Yes	Platform independent
Details on defined / reported methodology / approach	This framework for software development, as its name suggests, is made up of the waterfall, iterative, and agile approaches to software development.	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	No	
Organizational aspects on implementation	The composite process framework combines the business rules and processes that are involved in mobile application development.	
Project management aspects on implementation	A composite software development process framework retains the flexible aspects of the agile development approach and, at the same time, facilitates exchange of information between project stakeholders (such as business users, developers and testers) during the project life-cycle. Therefore, the CASDPF increases the chance of project success.	

Data item	Value	Notes
Study identifier	(Schwieren and Vossen, 2009)	
Title	A design and development methodology for mobile RFID applications based on the ID-Services middleware architecture	
Publication details	J. Schwieren and G. Vossen. "A design and development methodology for mobile RFID applications based on the ID-Services middleware architecture," in Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference on, 2009, pp. 260–266.	
Study type	New methodology	
Name of methodology / approach	Design and Development Methodology for mobile RFID applications	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	Basic process model of the proposed design and development methodology consists of three phases: Analysis, Design and Implementation. The authors propose basic activities at very high abstraction level.	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	SPCS - Sentry Patrol Control System	

Organizational aspects on implementation	None	
Project management aspects on implementation	None	

Data item	Value	Notes
Study identifier	(Barnawi et al., 2012)	
Title	A Framework for Next Generation Mobile and Wireless Networks Application Development using Hybrid Component Based Development Model	
Publication details	A. Barnawi, M. Qureshi, and A. I. Khan. "A Framework for Next Generation Mobile and Wireless Networks Application Development using Hybrid Component Based Development Model," Arxiv preprint arXiv:1202.2515, 2012.	
Study type	New methodology	
Name of methodology / approach	Component Based Model for IP Multimedia Subsystem	CBD Model for the IMS
Application in multi-platform development	Yes	Platform independent
Details on defined / reported methodology / approach	A new component-based development (CBD) model has been proposed for an IMS-based mass mobile examination system as a solution for the research problem. A CBD model is a process model that provides a framework to develop software from previously developed components. The main phases of the improved CBD are 'Project Planning', 'Analysis', 'Adaptation, Engineering & Integration' and 'Testing'.	
Additional resources on methodology / approach description	The phases are described in detail. The document used in the process are also presented and described.	
Report on methodology / approach example implementation	MObile Mass EXamination (MOMEX)	
Organizational aspects on implementation	None	
Project management aspects on implementation	None	

Data item	Value	Notes
Study identifier	(Chen, 2004)	
Title	A methodology for building mobile computing applications	
Publication details	M. Chen, "A methodology for building mobile computing applications," International journal of electronic business, vol. 2, no. 3, pp. 229–243, 2004.	
Study type	New methodology	
Name of methodology / approach	A Methodology for Building Enterprise-Wide Mobile Applications	
Application in multi-platform development	Yes	Platform independent
Details on defined / reported methodology / approach	The five major phases for building mobile computing applications are described as follows: 1. Develop enterprise-wide mobile strategies 2. Analyze the mobility of business processes 3. Develop an enterprise-wide mobile technical architecture 4. Build mobile applications	

	5. Deploy mobile applications	
Additional resources on methodology / approach description	Each of stated phases is described in more details.	
Report on methodology / approach example implementation	No	
Organizational aspects on implementation	The proposed methodology in this paper is an attempt to identify some guidelines and formulate a life-cycle approach to assisting enterprises in planning and developing enterprise-wide mobile strategies and applications.	
Project management aspects on implementation	No	

Data item	**Value**	**Notes**
Study identifier	(Xiong and Wang, 2010)	
Title	A new combined method for UCD and software development and case study	
Publication details	Y. Xiong and A. Wang, "A new combined method for UCD and software development and case study," in Information Science and Engineering (ICISE), 2010 2nd International Conference on, 2010, pp. 1–4.	
Study type	New methodology	
Name of methodology / approach	Inter-combined Model	
Application in multi-platform development	Yes	Platform independent
Details on defined / reported methodology / approach	Inter-combined Model aims to shorten the knowledge transfer from designers to developers. The model has four parts: - Requirement analysis and user study - Model establishment and function map specification - Design and background engine implementation - System integration and coding	
Additional resources on methodology / approach description	Each phase was described in additional details, but not to the level of activities, tasks, inputs and outputs.	
Report on methodology / approach example implementation	Mobile Karaoke project.	
Organizational aspects on implementation	Researchers stated that Inter-combined Model has positive effect on human resource arrangement and cost reduction.	
Project management aspects on implementation	Some implications on human resource arrangements.	

Data item	**Value**	**Notes**
Study identifier	(Walkerdine et al., 2009)	
Title	A Tool Supported Methodology For Developing Secure Mobile P2P Systems	
Publication details	J. Walkerdine, P. Phillips, and S. Lock. "A Tool Supported Methodology For Developing Secure Mobile P2P Systems," in Mobile peer-to-peer computing for next generation distributed environments: advancing conceptual and algorithmic applications, 2009, pp. 283–301.	
Study type	New methodology	
Name of methodology / approach	PEPERS Development Methodology (PDM)	

Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	PEPERS Development Methodology (PDM), is a tool-supported methodology that aims to assist designers in developing secure mobile P2P systems, and encourages them to consider specific mobile P2P design issues from an early stage. The PDM is based on a 5-stage spiral model. • Requirements Elicitation • Propose P2P system architecture • Propose sub-system design • System Implementation • Verification and Validation	
Additional resources on methodology / approach description	BANKSEC project P2P ARCHITECT project PEPERS project	
Report on methodology / approach example implementation	Case study - The Security firm pilot	
Organizational aspects on implementation	Workshops were held with local mobile phone software companies to obtain additional third-party feedback. These companies were typically small in size, and so provided a different perspective to the software development process. Overall the developers found the PDM and supporting tool to offer significant help in guiding the development of their secure mobile P2P applications. The smaller industrial companies were less sure about its use to them, mainly because they do not have the resources to follow a traditional development process and time to market is critical to them.	
Project management aspects on implementation	None	

Data item	**Value**	**Notes**
Study identifier	(Cuccurullo et al., 2011)	
Title	A Visual Approach supporting the Development of MicroApps on Mobile Phones	
Publication details	S. Cuccurullo, R. Francese, M. Risi, and G. Tortora, "A Visual Approach supporting the Development of MicroApps on Mobile Phones," in Proc. of 3rd International Symposium on End-User Development, Brindisi, Italy, 2011, pp. 289–294.	
Study type	New methodology	
Name of methodology / approach	MicroApp visual approach	
Application in multi-platform development	Yes	Current implement. in Android
Details on defined / reported methodology / approach	In this paper, we present a visual approach to enable End-Users to compose visually their own applications directly on their mobile phone. It is composed of: - MicroApp Definition - MicroApp Modeling - MicroApp Deployment	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	No	
Organizational aspects	None	User centric

on implementation		method.
Project management aspects on implementation	None	

Data item	Value	Notes
Study identifier	(Nyström, 2011)	
Title	Agile Solo - Defining and Evaluating an Agile Software Development Process for a Single Software Developer	
Publication details	A. Nyström. "Agile Solo - Defining and Evaluating an Agile Software Development Process for a Single Software Developer," 2011.	Master thesis
Study type	New methodology / Approach usage	
Name of methodology / approach	Agile Solo / Test Driven Development	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	The development process was intended to be helpful for any single programmer in any project. The defined practices are: - Weekly Presentations and Updated Priorities, Monthly Deliveries and Customer Test, Planning an iteration, Test Driven Development, The Pomodoro Technique, Peer Code Review, Auto Code Review, Visual Control, Modeling, Compensating for pair programming, Iteration Task Management	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	Case study	
Organizational aspects on implementation	No	Single developer
Project management aspects on implementation	Yes. Agile project management.	

Data item	Value	Notes
Study identifier	(Zeidler et al., 2008)	
Title	An integrated product development process for mobile software	
Publication details	C. Zeidler, C. Kittl, and O. Petrovic, "An integrated product development process for mobile software," International Journal of Mobile Communications, vol. 6, no. 3, pp. 345–356, 2008.	
Study type	New methodology	
Name of methodology / approach	An Integrated Product Development Process for Mobile Software	
Application in multi-platform development	Yes	Platform independent
Details on defined / reported methodology / approach	Based on the extensive research coverage on the new product development process, we have adapted a holistic product development approach for mobile services and applications. The resulting process considers a more dynamic competitive environment and the use of common tools for strategic analysis and product development. Consists of five pages: - Idea generation - Business model development - Legal aspects - Market research and user experience design	

	- Implementation	
Additional resources on methodology / approach description	Phases are described at relatively the high level of abstraction. Although, the activities are enumerated.	
Report on methodology / approach example implementation	Case study: HEROLD mobile	
Organizational aspects on implementation	The process included the organizational aspects.	
Project management aspects on implementation	The process includes the project management aspects.	

Data item	**Value**	**Notes**
Study identifier	(Biswas et al., 2006)	
Title	Assessment of mobile experience engine, the development toolkit for context aware mobile applications	
Publication details	A. Biswas, T. Donaldson, J. Singh, S. Diamond, D. Gauthier, and M. Longford, "Assessment of mobile experience engine, the development toolkit for context aware mobile applications," in Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology, New York, NY, USA, 2006.	
Study type	Methodology usage	
Name of methodology / approach	New media application prototyping	
Application in multi-platform development	Yes	Platform independent
Details on defined / reported methodology / approach	Prototyping with multiple iterations is an expensive solution to break this deadlock. The key bottlenecks in such prototyping are: contextual/user behavior research; design idea generation; design transfer from designer/artist to the technologists; system design, development and testing; and situated validation	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	Trickster game application Deer & Bear game application Situated editor mobile application	
Organizational aspects on implementation	None	
Project management aspects on implementation	None	

Data item	**Value**	**Notes**
Study identifier	(Maia et al., 2010)	
Title	Considerations on developing mobile applications based on the Capuchin project	
Publication details	M. E. F. Maia, C. Celes, R. Castro, and R. M. C. Andrade. "Considerations on developing mobile applications based on the Capuchin project," in Proceedings of the 2010 ACM Symposium on Applied Computing, New York, NY, USA, 2010, pp. 575–579.	
Study type	New methodology	
Name of methodology / approach	Development process of Caputchin applications	Name is not formally defined
Application in multi-	No	Platforms

platform development		supporting Flash only
Details on defined / reported methodology / approach	Paper shows an initial development process for mobile applications based on the Capuchin project. The defined phases are: - Application requirements elicitation and user interface draft - Implement and test the View component based on Flash UI - Flash/JME division and data transfer format specification - Implement the controller and model components	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	Case study: Weather application	
Organizational aspects on implementation	Some organizational aspects are discussed	
Project management aspects on implementation	None	

Data item	**Value**	**Notes**
Study identifier	(Shiratuddin and Sarif, 2009)	
Title	Construction of Matrix and eMatrix for Mobile Development Methodologies	
Publication details	N. Shiratuddin and S. M. Sarif, "Construction of Matrix and eMatrix for Mobile Development Methodologies," in Handbook of research in mobile business: technical, methodological, and social perspectives, 2nd ed., IGI Global, 2009, pp. 113–126.	
Study type	Methodology usage	
Name of methodology / approach	Mobile-D Mobile RAD Dynamic Channel Model Mobile Engineering (MobE)	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	The study compares the mentioned methodologies in systematic manner.	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	Yes. The methodologies are compared based on example projects.	
Organizational aspects on implementation	Partially included in comparison.	
Project management aspects on implementation	Partially included in comparison.	

Data item	**Value**	**Notes**
Study identifier	(Rahimian and Ramsin, 2008)	
Title	Designing an agile methodology for mobile software development: A hybrid method engineering approach	
Publication details	V. Rahimian and R. Ramsin, "Designing an agile methodology for mobile software development: A hybrid method engineering approach," in Research Challenges in Information Science, 2008. RCIS	

	2008. Second International Conference on, 2008, pp. 337–342.	
Study type	New methodology	
Name of methodology / approach	Agile Methodology for Mobile Software Development	Formally not defined
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	Paper identifies the main requirements of a mobile software development methodology, based on which a highlevel methodology framework was built using the Hybrid Methodology Design approach. Proposed methodology is an agile risk-based methodology, highly influenced by the Adaptive Software Development method and New Product Development approaches.	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	No	
Organizational aspects on implementation	Included in methodology.	
Project management aspects on implementation	Agile project management should be used.	

Data item	**Value**	**Notes**
Study identifier	(Bergström and Engvall, 2011)	
Title	Development of handheld mobile applications for the public sector in Android and iOS using agile Kanban process tool	
Publication details	F. Bergström and G. Engvall, "Development of handheld mobile applications for the public sector in Android and iOS using agile Kanban process tool," 2011.	
Study type	Approach usage	
Name of methodology / approach	Kanban	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	Kanban a lean approach to agile software development and a part of the lean thinking. The approach is invented by Toyota which used this process for the visual and physical signaling system that ties together the whole Lean Production System. However, Kanban in software development can be divided into three main parts. - Visualize the workflow - Limit work in process - Measure the lead time	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	Prototype application	
Organizational aspects on implementation	Not well defined.	
Project management aspects on implementation	Not well defined.	

Data item	Value	Notes
Study identifier	(Jeong et al., 2008)	
Title	Development Process of Mobile Application SW Based on Agile Methodology	
Publication details	Y. J. Jeong, J. H. Lee, and G. S. Shin, "Development Process of Mobile Application SW Based on Agile Methodology," in Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on, 2008, vol. 1, pp. 362–366.	
Study type	New Methodology	
Name of methodology / approach	MASAM methodology	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	The objective of this proprietary methodology is to provide the process for developing the application SW operated on mobile platform. Standard process of THE MASAM is comprised of 4 phases: - Development Preparation Phase, - Embodiment Phase, - Product developing Phase, and - Commercialization Phase.	Paper written in poor English.
Additional resources on methodology / approach description	The phases are briefly described.	
Report on methodology / approach example implementation	No.	
Organizational aspects on implementation	Partially covered.	
Project management aspects on implementation	Agile approach should be used.	

Data item	Value	Notes
Study identifier	(Korkala and Abrahamsson, 2004)	
Title	Extreme programming: Reassessing the requirements management process for an offsite customer	
Publication details	M. Korkala and P. Abrahamsson, "Extreme programming: Reassessing the requirements management process for an offsite customer," Software Process Improvement, pp. 12–22, 2004.	
Study type	Methodology usage	
Name of methodology / approach	Extreme Programming, Mobile-D	
Application in multi-platform development	Yes	Platform independent
Details on defined / reported methodology / approach	Brief description is provided on executed process: - Identify essential requirements - Evaluation and implementation of enhanced User Storries - Implement, Report and Feedback - Iteration Acceptance	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	zOmbie project	
Organizational aspects on implementation	No	
Project management	Agile approach used.	

aspects on implementation		

Data item	Value	Notes
Study identifier	(Gal and Topol, 2005)	
Title	Experimentation of a Game Design Methodology for Mobile Phones Games	
Publication details	V. Gal and A. Topol, "Experimentation of a Game Design Methodology for Mobile Phones Games," 2005.	
Study type	New methodology	
Name of methodology / approach	2TUP - 2 Tracks Unified Process	
Application in multi-platform development	Yes	Platform independent
Details on defined / reported methodology / approach	Paper presents the 2TUP method, "2 Tracks Unified Process". It is based upon a SPEM modeling architecture in order to conceive elegant and adapted solutions but also to take advantage of the new techniques and technologies. 2TUP is a unified process (i.e. a software development process) built on the UML modeling language. According to 2TUP the process is modeled by two branches (tracks): - A functional track (capitalization of knowledge trade) - A technical track (re-use of a technical knowhow).	
Additional resources on methodology / approach description	The fair description is given on implementation on own project.	
Report on methodology / approach example implementation	Case study.	
Organizational aspects on implementation	No	
Project management aspects on implementation	No	

Data item	Value	Notes
Study identifier	(Kim, 2008)	
Title	Frameworks of Process Improvement for Mobile Applications	
Publication details	H. K. Kim, "Frameworks of Process Improvement for Mobile Applications," Engineering Letters, vol. 16, 2008.	
Study type	Approach usage	
Name of methodology / approach	Model Driven Development	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	Paper goes through mobile development process and architectural structures and analysis of these with empirical mobile application development.	Poor paper structure.
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	Case study.	
Organizational aspects on implementation	No	
Project management	No	

aspects on implementation		

Data item	Value	Notes
Study identifier	(Scharff, 2011)	
Title	Guiding global software development projects using Scrum and Agile with quality assurance	
Publication details	C. Scharff, "Guiding global software development projects using Scrum and Agile with quality assurance," in Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference on, 2011, pp. 274–283.	
Study type	Methodology usage	
Name of methodology / approach	Scrum	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	The paper describes the usage of Scrum in distributed development teams as well as for development for different target platforms. The developers are students.	
Additional resources on methodology / approach description	Brief description of methodology.	
Report on methodology / approach example implementation	Android application Blackberry application Java ME Team	
Organizational aspects on implementation	Partially covered.	
Project management aspects on implementation	Partially covered.	

Data item	Value	Notes
Study identifier	(Abrahamsson et al., 2005b)	
Title	Improving business agility through technical solutions: A case study on test-driven development in mobile software development	
Publication details	[1]P. Abrahamsson, A. Hanhineva, and J. Jäälinoja, "Improving business agility through technical solutions: A case study on test-driven development in mobile software development," in Business Agility and Information Technology Diffusion, 2005.	
Study type	Approach usage	
Name of methodology / approach	Test Driven Development	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	Thorough research was performed on empirical evidence of using the Test Driven Development in mobile application development process.	
Additional resources on methodology / approach description	Test Driven Development described. The references on other researches are given.	
Report on methodology / approach example implementation	Case study.	
Organizational aspects on implementation	Included.	
Project management	Included.	

aspects on implementation		

Data item	Value	Notes
Study identifier	(Ortiz and Prado, 2010)	
Title	Improving device-aware Web services and their mobile clients through an aspect-oriented, model-driven approach	
Publication details	G. Ortiz and A. G. D. Prado, "Improving device-aware Web services and their mobile clients through an aspect-oriented, model-driven approach," Information and Software Technology, vol. 52, no. 10, pp. 1080 – 1093, 2010.	
Study type	Approach usage.	
Name of methodology / approach	Model Driven Development	
Application in multi-platform development	Yes	Platform independent
Details on defined / reported methodology / approach	Aspect-Oriented Programming and model-driven development have been used to reduce both the impact of service and client code adaptation for multiple devices as well as to facilitate the developer's task.	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	Case study	
Organizational aspects on implementation	No	
Project management aspects on implementation	No	

Data item	Value	Notes
Study identifier	(Kaariainen et al., 2004)	
Title	Improving requirements management in extreme programming with tool support - an improvement attempt that failed	
Publication details	J. Kaariainen, J. Koskela, P. Abrahamsson, and J. Takalo, "Improving requirements management in extreme programming with tool support - an improvement attempt that failed," in Euromicro Conference, 2004. Proceedings. 30th, 2004, pp. 342 – 351.	
Study type	Methodology usage	
Name of methodology / approach	Extreme Programming	
Application in multi-platform development	Yes	Platform independent
Details on defined / reported methodology / approach	The paper mainly focusses on other aspects than on methodology itself.	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	zOmbie project	
Organizational aspects on implementation	No	

Project management aspects on implementation	No	

Data item	Value	Notes
Study identifier	(Salo, 2004)	
Title	Improving software process in agile software development projects: results from two XP case studies	
Publication details	O. Salo, "Improving software process in agile software development projects: results from two XP case studies," in Euromicro Conference, 2004. Proceedings. 30th, 2004, pp. 310–317.	
Study type	Methodology usage	
Name of methodology / approach	Extreme Programming	
Application in multi-platform development	Yes	Platform independent
Details on defined / reported methodology / approach	The paper mainly focusses on other aspects than on methodology itself.	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	eXpert project zOmbie project	
Organizational aspects on implementation	Included in the analysis.	
Project management aspects on implementation	Included in the analysis.	

Data item	Value	Notes
Study identifier	(Su and Scharff, 2010)	
Title	Know Yourself and Beyond: A Global Software Development Project Experience with Agile Methodology	
Publication details	S. H. Su and C. Scharff, "Know Yourself and Beyond: A Global Software Development Project Experience with Agile Methodology," in Proceedings of Student-Faculty Research Day, CSIS, 2010.	
Study type	Methodology usage	
Name of methodology / approach	Scrum	
Application in multi-platform development	Yes	Platform independent
Details on defined / reported methodology / approach	The paper describes the usage of Scrum process in a case study development performed by students.	
Additional resources on methodology / approach description	Scrum was partially described.	
Report on methodology / approach example implementation	Case study: TargetFirstGrade project	
Organizational aspects on implementation	Partially included	
Project management aspects on	Included	

implementation		

Data item	Value	Notes
Study identifier	(Shiratuddin and Sarif, 2008)	
Title	m^d-Matrix: Mobile Application Development Tool	
Publication details	N. Shiratuddin and S. M. Sarif, "m d-Matrix: Mobile Application Development Tool," Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 1, 2008.	
Study type	Methodology usage	
Name of methodology / approach	Mobile-D Mobile RAD Dynamic Channel Model Mobile Engineering (MobE)	
Application in multi-platform development	Yes	Platform independent
Details on defined / reported methodology / approach	Paper describes the tool that helps novices to choose development methodology. In that manner, the four mentioned methodologies are compared.	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	Yes. The methodologies are compared based on example projects.	
Organizational aspects on implementation	Included in analysis.	
Project management aspects on implementation	Included in analysis.	

Data item	Value	Notes
Study identifier	(Saifudin et al., 2011)	
Title	MMCD Framework and Methodology for Developing m-Learning Applications	
Publication details	A. W. S. N. Saifudin, B. S. Salam, and C. M. H. L. Abdullah, "MMCD Framework and Methodology for Developing m-Learning Applications," presented at the International conference on Teaching & Learning in Higher Education (ICTLHE 2011), 2011.	
Study type	New methodology	
Name of methodology / approach	MMCD Methodology	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	The proposed MMCD Methodology focuses only m-Learning applications. It comprises of five main components: - application idea creation stage, - structure analysis stage, - process design stage, - main function development stages, and - testing stage	
Additional resources on methodology / approach description	Stages are described on abstract level	
Report on methodology / approach example implementation	M-Nations m-learning application	
Organizational aspects	No	

on implementation		
Project management aspects on implementation	No	

Data item	Value	Notes
Study identifier	(Rupnik, 2009)	
Title	Mobile Applications Development Methodology	
Publication details	R. Rupnik, "Mobile Applications Development Methodology," in Handbook of research in mobile business: technical, methodological, and social perspectives, Second Edition., B. Unhelkar, Ed. IGI Global Snippet, 2009.	
Study type	New methodology	
Name of methodology / approach	Mobile Application Development Methodology	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	The book chapter defines new methodology and roughly defines the main phases, but it lacks the precise and detailed description on methodology itself. The defined phases are: - strategy, - analysis, - design - implementation	
Additional resources on methodology / approach description	Some elements of the stated phases are described.	
Report on methodology / approach example implementation	Two projects.	
Organizational aspects on implementation	No.	
Project management aspects on implementation	No.	

Data item	Value	Notes
Study identifier	(Pauca and Guy, 2012)	
Title	Mobile apps for the greater good: a socially relevant approach to software engineering	
Publication details	V. P. Pauca and R. T. Guy, "Mobile apps for the greater good: a socially relevant approach to software engineering," in Proceedings of the 43rd ACM technical symposium on Computer Science Education, New York, NY, USA, 2012, pp. 535–540.	
Study type	Methodology usage	
Name of methodology / approach	Scrum	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	Paper only mentions the usage of Scrum and nothing else.	
Additional resources on methodology / approach description	No	
Report on methodology / approach example	Case study.	

implementation		
Organizational aspects on implementation	No	
Project management aspects on implementation	No	

Data item	Value	Notes
Study identifier	(Abrahamsson et al., 2009)	
Title	Mobile-D for Mobile Software: How to Use Agile Approaches for the Efficient Development of Mobile Applications	
Publication details	P. Abrahamsson, T. Ihme, K. Kolehmainen, P. Kyllönen, and O. Salo, "Mobile-D for Mobile Software: How to Use Agile Approaches for the Efficient Development of Mobile Applications." 2009.	Tutorial.
Study type	New methodology	
Name of methodology / approach	Mobile-D	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	This tutorial seeks to provide an overview on the special characteristics of mobile software development and introduce a development approach called Mobile D, which combines several agile approaches to meet the needs of volatile mobile application development.	Tutorial materials unavailable
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	No	
Organizational aspects on implementation	No	
Project management aspects on implementation	No	

Data item	Value	Notes
Study identifier	(Abrahamsson et al., 2004)	
Title	Mobile-D: an agile approach for mobile application development	
Publication details	P. Abrahamsson, A. Hanhineva, H. Hulkko, T. Ihme, J. Jäälinoja, M. Korkala, J. Koskela, P. Kyllönen, and O. Salo, "Mobile-D: an agile approach for mobile application development," in Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications, New York, NY, USA, 2004, pp. 174–175.	First publication presenting Mobile-D
Study type	New methodology	
Name of methodology / approach	Mobile-D	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	Mobile-D approach is based on Extreme Programming (development practices), Crystal methodologies (method scalability), and Rational Unified Process (life-cycle coverage). A development project, following the Mobile-D approach, is divided into five iterations. These phases are: set-up, core, core2, stabilize, and wrap-up.	The phases are later renamed.
Additional resources on methodology / approach description	In this paper the methodology was not well defined.	

Report on methodology / approach example implementation	The Mobile-D approach has been empirically tested and further developed in four case studies within the ENERGI laboratory at VTT, The Technical Research Centre of Finland. These cases were concerned with new mobile phone extensions of database systems.	
Organizational aspects on implementation	The Mobile-D approach is optimized for a team of less than ten developers working in a co-located office space aiming at delivering a fully functional mobile application in a short time frame. Mobile-D has been developed in co-operation with three companies developing mobile software products and services.	
Project management aspects on implementation	Not included.	

Data item	**Value**	**Notes**
Study identifier	(Marinho et al., 2012)	
Title	MobiLine: A Nested Software Product Line for the domain of mobile and context-aware applications	
Publication details	[1]F. G. Marinho, R. M. C. Andrade, C. Werner, W. Viana, M. E. F. Maia, L. S. Rocha, E. Teixeira, J. B. F. Filho, V. L. L. Dantas, F. Lima, and S. Aguiar, "MobiLine: A Nested Software Product Line for the domain of mobile and context-aware applications," Science of Computer Programming, p. -, 2012.	
Study type	New approach	
Name of methodology / approach	MobiLine	
Application in multi-platform development	Yes	Platform independant
Details on defined / reported methodology / approach	This paper discusses an approach for the development of mobile and context-aware software using the Software Product Line (SPL) paradigm. MobiLine - A Nested Software Product Line for the domain of mobile and context-aware applications.	
Additional resources on methodology / approach description	MobiLine development approach is well defined.	
Report on methodology / approach example implementation	Case studies	
Organizational aspects on implementation	Partially covered.	
Project management aspects on implementation	Partially covered.	

Data item	**Value**	**Notes**
Study identifier	(Forstner et al., 2006)	
Title	Model-based system development for embedded mobile platforms	
Publication details	B. Forstner, L. Lengyel, T. Levendovszky, G. Mezei, I. Kelenyi, and H. Charaf, "Model-based system development for embedded mobile platforms," in Model-Based Development of Computer-Based Systems and Model-Based Methodologies for Pervasive and Embedded Software, 2006. MBD/MOMPES 2006. Fourth and Third International Workshop on, 2006, p. 10–pp.	
Study type	Approach usage	
Name of methodology / approach	Model Driven Development	
Application in multi-	Yes	

platform development		
Details on defined / reported methodology / approach	Paper discuss the relevance of the model-based approach that facilitates a more efficient software development. Additionally, paper describes several tools that support model driven development. .	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	No	
Organizational aspects on implementation	None	
Project management aspects on implementation	None	

Data item	**Value**	**Notes**
Study identifier	(Thompson et al., 2010)	
Title	Model-Driven Architectures for Optimizing Mobile Application Performance	
Publication details	C. Thompson, J. White, B. Dougherty, H. Turner, S. Campbell, K. Zienkiewicz, and D. C. Schmidt, "Model-Driven Architectures for Optimizing Mobile Application Performance." 2010.	Introduction to the book
Study type	Approach usage	
Name of methodology / approach	Model Driven Development	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	Scarce	
Additional resources on methodology / approach description	Pointing to the Book that was unavailable.	
Report on methodology / approach example implementation	No	
Organizational aspects on implementation	No	
Project management aspects on implementation	No	

Data item	**Value**	**Notes**
Study identifier	(Khambati et al., 2008)	
Title	Model-Driven Development of Mobile Personal Health Care Applications	
Publication details	A. Khambati, J. Grundy, J. Warren, and J. Hosking, "Model-Driven Development of Mobile Personal Health Care Applications," in Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering, Washington, DC, USA, 2008, pp. 467–470.	
Study type	Approach usage	
Name of methodology / approach	Model Driven Development	

Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	The focus of the paper was not on MDD but rather on tool that was used to perform MDD.	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	Case study	
Organizational aspects on implementation	No	
Project management aspects on implementation	No	

Data item	**Value**	**Notes**
Study identifier	(Kim et al., 2009)	
Title	Performance testing based on test-driven development for mobile applications	
Publication details	H. Kim, B. Choi, and S. Yoon, "Performance testing based on test-driven development for mobile applications," in Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication, New York, NY, USA, 2009, pp. 612–617.	
Study type	Approach usage	
Name of methodology / approach	Test Driven Development	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	The goal of this study is to develop a mobile performance unit testing tool that not only supports the functional testing in the development process of unit testing environment but also supports performance unit testing generation and performance automation in order to improve the quality and reliability of mobile applications.	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	MOPAD project	
Organizational aspects on implementation	No	
Project management aspects on implementation	No	

Data item	**Value**	**Notes**
Study identifier	(Manjunatha et al., 2010)	
Title	Power of clouds in your pocket: An efficient approach for cloud mobile hybrid application development	
Publication details	A. Manjunatha, A. Ranabahu, A. Sheth, and K. Thirunarayan, "Power of clouds in your pocket: An efficient approach for cloud mobile hybrid application development," in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, 2010, pp. 496–503.	
Study type	New Approach	

Name of methodology / approach	MobiCloud (A cloud mobile hybrid application generation)	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	The objective of this research, therefore, is to provide a disciplined approach to mobile applications development centered around a DSL based platform agnostic application development paradigm for CMH applications.	
Additional resources on methodology / approach description	Approach is well defined.	
Report on methodology / approach example implementation	Case study	
Organizational aspects on implementation	No	
Project management aspects on implementation	No	

Data item	**Value**	**Notes**
Study identifier	(Wolkerstorfer et al., 2008)	
Title	Probing an agile usability process	
Publication details	P. Wolkerstorfer, M. Tscheligi, R. Sefelin, H. Milchrahm, Z. Hussain, M. Lechner, and S. Shahzad, "Probing an agile usability process," in CHI '08 extended abstracts on Human factors in computing systems, New York, NY, USA, 2008, pp. 2151–2158.	
Study type	New methodology	
Name of methodology / approach	Agile usability process	
Application in multi-platform development	Yes	Platform independent
Details on defined / reported methodology / approach	Paper describes adaptations to the classical Extreme Programming (XP) process. The approach described integrates HCI (human computer interaction) instruments. The implemented HCI instruments are: user studies, extreme personas (a variation of the personas approach), usability expert evaluations, usability tests, and automated usability evaluations. By combining XP and UCD (user centered development) processes it takes advantages of both approaches.	
Additional resources on methodology / approach description	Short description of the process.	
Report on methodology / approach example implementation	No	
Organizational aspects on implementation	Some aspects included.	
Project management aspects on implementation	Some aspects included.	

Data item	**Value**	**Notes**
Study identifier	(Scharff and Verma, 2010)	
Title	Scrum to support mobile application development projects in a just-in-time learning context	
Publication details	C. Scharff and R. Verma, "Scrum to support mobile application development projects in a just-in-time learning context," in	

	Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, New York, NY, USA, 2010, pp. 25–31.	
Study type	Methodology usage	
Name of methodology / approach	Scrum	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	During this project, we attempted to provide students with a real experience with Scrum on mobile application development projects. We defined a model of working to be used in a classroom setting that involved Scrum teams, a certified Scrum Master, a Product Owner and a Client.	
Additional resources on methodology / approach description	Scrum process described.	
Report on methodology / approach example implementation	Case study	
Organizational aspects on implementation	Some elements included.	
Project management aspects on implementation	Some elements included.	

Data item	**Value**	**Notes**
Study identifier	(Rosa and Lucena,Jr., 2011)	
Title	Smart composition of reusable software components in mobile application product lines	
Publication details	R. E. V. S. Rosa and V. F. Lucena,Jr., "Smart composition of reusable software components in mobile application product lines," in Proceedings of the 2nd International Workshop on Product Line Approaches in Software Engineering, New York, NY, USA, 2011, pp. 45–49.	
Study type	Approach usage	
Name of methodology / approach	Software Product Lines	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	The Software Product Lines (SPL) approach seems to be an useful technique to support mobile application development. A way to make SPL more effective is automating the software components composition for building mobile applications.	
Additional resources on methodology / approach description	Scarce	
Report on methodology / approach example implementation	AppSpotter project	
Organizational aspects on implementation	No	
Project management aspects on implementation	No	

Data item	**Value**	**Notes**
Study identifier	(Zakal et al., 2011)	

Title	Software Product Lines-based development	
Publication details	D. Zakal, L. Lengyel, and H. Charaf, "Software Product Lines-based development," in Applied Machine Intelligence and Informatics (SAMI), 2011 IEEE 9th International Symposium on, 2011, pp. 79–81.	
Study type	Approach usage	
Name of methodology / approach	Model Driven Product Lines (Software Product Lines)	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	Current paper has presented a model-driven approach of Software Product Lines, suggesting the use of feature models as integral parts of product family specifications.	
Additional resources on methodology / approach description	Model Driven Product Lines	
Report on methodology / approach example implementation	No	
Organizational aspects on implementation	No	
Project management aspects on implementation	No	

Data item	**Value**	**Notes**
Study identifier	(Forstner et al., 2005)	
Title	Supporting Rapid Application Development on Symbian Platform	
Publication details	B. Forstner, L. Lengyel, I. Kelenyi, T. Levendovszky, and H. Charaf, "Supporting Rapid Application Development on Symbian Platform," in *Computer as a Tool, 2005. EUROCON 2005.The International Conference on*, 2005, vol. 1, pp. 72 –75.	
Study type	Methodology usage	
Name of methodology / approach	Rapid Application Development	
Application in multi-platform development	Yes	Symbian reported
Details on defined / reported methodology / approach	The paper introduces rapid application development tool set for Symbian OS. The focus of the paper is not on RAD methodology, but rather on presented tool set.	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	Case study	
Organizational aspects on implementation	No	
Project management aspects on implementation	No	

Data item	**Value**	**Notes**
Study identifier	(Binsaleh and Hassan, 2011)	
Title	Systems Development Methodology for Mobile Commerce Applications: Agile vs. Traditional	

Publication details	M. Binsaleh and S. Hassan, "Systems Development Methodology for Mobile Commerce Applications: Agile vs. Traditional," International Journal of Online Marketing (IJOM), vol. 1, no. 4, pp. 33–47, 2011.	The full paper not completely available to the researcher
Study type	New methodology / Methodology usage	
Name of methodology / approach	Systems Development Methodology / N/A	
Application in multi-platform development	Yes	Platform independent
Details on defined / reported methodology / approach	Only portion of the paper is available to the researcher due to the lack of subscription to Igi-global publishing. The acquired materials state that comprehensive research was performed in order to determine the customs of mobile application developers and that the methodology was proposed based on these results.	
Additional resources on methodology / approach description	N/A	
Report on methodology / approach example implementation	N/A	
Organizational aspects on implementation	N/A	
Project management aspects on implementation	N/A	

Data item	**Value**	**Notes**
Study identifier	(Hedberg and Iisakka, 2006)	
Title	Technical Reviews in Agile Development: Case Mobile-D	
Publication details	H. Hedberg and J. Iisakka, "Technical Reviews in Agile Development: Case Mobile-D," in Quality Software, 2006. QSIC 2006. Sixth International Conference on, 2006, pp. 347–353.	
Study type	Methodology usage / Approach usage	
Name of methodology / approach	Mobile-D / Test Driven Development	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	The short description on Mobile-D is available. Although the paper focuses on another topic it gives lots of references on Mobile-D process.	
Additional resources on methodology / approach description	ENERGY laboratory VTT Research center	
Report on methodology / approach example implementation	No	
Organizational aspects on implementation	Partially included.	
Project management aspects on implementation	Partially included.	

Data item	**Value**	**Notes**
Study identifier	(Scharff, 2010)	
Title	The Software Engineering of Mobile Application Development	

Publication details	C. Scharff, "The Software Engineering of Mobile Application Development," Pace University, NY, USA, 2010.	Presentation
Study type	Methodology usage	
Name of methodology / approach	Scrum	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	Presentation covers several topics and one of them is Scrum. The process is described and the examples are given.	
Additional resources on methodology / approach description	Scrum described in detail.	
Report on methodology / approach example implementation	Several projects: RestoMobile, TargetFirstGrade, No Ink…	
Organizational aspects on implementation	Partially included.	
Project management aspects on implementation	Partially included.	

Data item	**Value**	**Notes**
Study identifier	(Ejlersen et al., 2008)	
Title	Using Design Science to Develop a Mobile Application	
Publication details	A. Ejlersen, M. S. Knudsen, J. Løvgaard, and M. B. Sørensen, "Using Design Science to Develop a Mobile Application," 2008.	
Study type	Methodology usage	
Name of methodology / approach	Design Science	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	The special framework is developed to help the usage of Design Science components in mobile application development.	
Additional resources on methodology / approach description	Design Science partially described.	
Report on methodology / approach example implementation	Friend Finder mobile application	
Organizational aspects on implementation	Partially included	
Project management aspects on implementation	No	

Data item	**Value**	**Notes**
Study identifier	(Ihme and Abrahamsson, 2005)	
Title	The Use of Architectural Patterns in the Agile Software Development of Mobile Applications	
Publication details	T. Ihme and P. Abrahamsson, "The Use of Architectural Patterns in the Agile Software Development of Mobile Applications," 2005.	
Study type	Methodology usage	
Name of methodology / approach	Mobile-D	

Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	Paper reports the usage of Mobile-D methodology, but only in accordance with design phase in development process.	
Additional resources on methodology / approach description	No	
Report on methodology / approach example implementation	Case studies and projects.	
Organizational aspects on implementation	No	
Project management aspects on implementation	No	

Data item	**Value**	**Notes**
Study identifier	(Um et al., 2005)	
Title	ViP: A Practical Approach to Platform-based System Modeling Methodology	
Publication details	J. Um, S. Hong, Y. T. Kim, E. Chung, K. M. Choi, J. T. Kong, and S. K. Eo, "ViP: A Practical Approach to Platform-based System Modeling Methodology," *Journal of Semiconductor Technology and Science*, vol. 5, no. 2, p. 89, 2005.	
Study type	New methodology	
Name of methodology / approach	ViP (Virtual Platform)	
Application in multi-platform development	Yes	
Details on defined / reported methodology / approach	Paper proposes a new transaction-level system modeling methodology, called ViP (Virtual Platform). ViP has a two-step approach: - create a ViP for early stage software development - refine the ViP to increase the cycle accuracy for system performance analysis and software optimization The following phases are executed - IP Modeling - IP Model verification - Bus Subsystem Modeling - Integration	
Additional resources on methodology / approach description	The special case study for implementation for mobile devices is created.	
Report on methodology / approach example implementation	Case study	
Organizational aspects on implementation	Partially included in report	
Project management aspects on implementation	Partially included in report	

Appendix E – Multi-platform Case Artifacts Ontology

The appendix shows *Multi-platform Case Artifacts Ontology* presented in *Manchester OWL Syntax* format. The Manchester syntax is a user-friendly compact syntax for OWL 2 ontologies (Horridge and Patel-Schneider, 2009). Although it is frame-based, as opposed to the axiom-based other syntaxes for OWL 2, we find it to be the most compact and human readable syntax. The document presented in this appendix is available at http://barok.foi.hr/~zstapic/ont/mcao_m.owl, and the same ontology in OWL/XML syntax is available at http://barok.foi.hr/~zstapic/ont/mcao.owl.

```
Prefix: : <http://www.w3.org/2002/07/owl#>
Prefix: owl: <http://www.w3.org/2002/07/owl#>
Prefix: rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
Prefix: xml: <http://www.w3.org/XML/1998/namespace>
Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix: acao: <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#>
Prefix: mcao: <http://www.foi.unizg.hr/ontologies/MultiplatformCaseArtifacts#>
Prefix: rdfs: <http://www.w3.org/2000/01/rdf-schema#>
Prefix: wpcao: <http://www.foi.unizg.hr/ontologies/WindowsPhoneCaseArtifacts#>

Ontology: <http://www.foi.unizg.hr/ontologies/MultiplatformCaseArtifacts>

Annotations:
    rdfs:comment "The ontology describing the artifacts that arise in the
         development of multi-platform prototype mobile application by using
         Mobile-D methodology."@en,
    rdfs:isDefinedBy "Zlatko Stapić"

AnnotationProperty: rdfs:isDefinedBy
AnnotationProperty: rdfs:label
AnnotationProperty: mcao:NOTICE
AnnotationProperty: rdfs:comment
AnnotationProperty: acao:inActivity
AnnotationProperty: acao:inPhase

Datatype: rdf:PlainLiteral

ObjectProperty: acao:createsArtifact
   Annotations:
        rdfs:comment "Inversed property of isCreatedByTask. It connects Task
             individuals and created specific Artifact individuals."@en
    Domain: acao:Task
    Range: acao:Artifact
    InverseOf: acao:isCreatedByTask

ObjectProperty: acao:isCreatedByTask
    Annotations:
        rdfs:comment "Property connecting the Task individuals that create specific
             Artifact individuals. Creating the artifact logically means it usage
             even if it is not explicitly stated."@en
    Domain: acao:Artifact
    Range: acao:Task
```

```
    InverseOf: acao:createsArtifact

ObjectProperty: acao:isPartOfArtifact
    Annotations:
        rdfs:comment "Property connecting individual artifacts into hierarchy. This
             property is Asymmetric as two individuals cannot be both part of each
             other. "@en
    Characteristics: Asymmetric
    Domain: acao:Artifact
    Range: acao:Artifact
    InverseOf: acao:includesArtifact

ObjectProperty: mcao:isSimilarToArtifact
    Annotations:
        rdfs:comment "Property connecting the individuals of class Artifact with
             other similar individuals of the same class. Usually, all artifacts in
             the same class, if class is reusable, are reusable, but this is not a
             rule. Sometimes, pairs of artifacts in the same class can be mutually
             reusable, but not reusable with other artifacts of pairs."@en
    Characteristics: Symmetric
    Domain: acao:Artifact
    Range: acao:Artifact

ObjectProperty: acao:isPerformedIn
    Annotations:
        rdfs:comment "Property defines relationship between specific Task
             individuals and owning Activity. Logically, this property is inverse
             of consistsOf property, but we defined both separate to have the
             information available even in the original model."@en
    Domain: acao:Activity or acao:Task
    Range: acao:Activity or acao:Phase

ObjectProperty: acao:isUpdatedByTask
    Annotations:
        rdfs:comment "Property connecting the Task individuals that update specific
             Artifact individuals."@en
    Domain: acao:Artifact
    Range: acao:Task
    InverseOf: acao:updatesArtifact

ObjectProperty: acao:isUsedByTask
    Annotations:
        rdfs:comment "Property connecting the Task individuals that read specific
             Artifact individuals."@en
    Domain: acao:Artifact
    Range: acao:Task
    InverseOf: acao:usesArtifact

ObjectProperty: acao:usesArtifact
    Annotations:
        rdfs:comment "Inversed property of isUsedByTask. It connects Task
             individuals and used specific Artifact individuals."@en
    Domain: acao:Task
    Range: acao:Artifact
    InverseOf: acao:isUsedByTask

ObjectProperty: mcao:hasReusabilityLevel
    Annotations:
```

```
        rdfs:comment "Property connecting specific Artifact individuals with one of
             predefined reusability levels. This property classifies artifacts into
             completely, partially or unreusable classes."@en
    Characteristics: Functional
    Domain: acao:Artifact
    Range: mcao:ReuseLevel

ObjectProperty: acao:updatesArtifact
    Annotations:
        rdfs:comment "Inversed property of isUpdatedByTask. It connects Task
             individuals and updated specific Artifact individuals."@en
    Domain:
        acao:Task
    Range:
        acao:Artifact
    InverseOf:
        acao:isUpdatedByTask

ObjectProperty: acao:hasArtifactType

    Annotations:
        rdfs:comment "Property connecting specific Artifact individuals with
             ArtifactType individuals. It defines type of the specific Artifact
             according to defined classification according to artifact usage."@en
    Characteristics: Functional
    Domain: acao:Artifact
    Range: acao:ArtifactType

ObjectProperty: acao:includesArtifact
    Annotations:
        rdfs:comment "Inverse property of isPartOfArtifact. It defines individual
             Artifacts that are included in observed Artifact."@en
    Characteristics: Asymmetric
    Domain: acao:Artifact
    Range: acao:Artifact
    InverseOf: acao:isPartOfArtifact

ObjectProperty: acao:consistsOf
    Annotations:
        rdfs:comment "Property connecting individual Activities that are performed
             in specific Phases and individual Tasks that are performed during
             specific Activities. Logically, this property is inverse property of
             isPerformedIn, but we explicitly defined it in order to have the
             information available even in the original model."@en
    Domain: acao:Activity or acao:Phase
    Range: acao:Activity or acao:Task

ObjectProperty: acao:hasArtifactOrigin
    Annotations:
        rdfs:comment "Property connecting individual Artifact and individual in
             definite class ArtifactOrigin which defines several possible types of
             Artifact origin. This property is used to classify artifacts by types
             but from different point of view than property hasArtifactType."@en
    Characteristics: Functional
    Domain: acao:Artifact
    Range: acao:ArtifactOrigin

Class: acao:ReleaseCeremoniesTask
    Annotations:
```

```
        rdfs:comment "The purpose of this task is to confirm that everything has
             been done right in the current iteration and the basis for further
             development is ensured. Release ceremonies are the final steps before
             making a release of the software. In practice, release ceremonies
             consist of two essential activities; release audit and baseline
             creation."@en,
        acao:inActivity
<http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#ReleaseDayActivity>,
        rdfs:label "Release Ceremonies Task"

    SubClassOf:
        acao:isPerformedIn only acao:ReleaseDayActivity,
        acao:isPerformedIn some acao:ReleaseDayActivity,
        acao:Task


Class: acao:ClassModelMobile

    Annotations:
        rdfs:comment "UML class diagram describing the mobile application internal
             structure and created classes. This model is used in SADD
             document."@en,
        rdfs:label "Class Model (Mobile)"

    SubClassOf:
        acao:isUpdatedByTask some acao:RequirementsAnalysisTask,
        acao:hasArtifactType some acao:Model,
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:isPartOfArtifact some acao:SADDDocument,
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:hasArtifactType only acao:Model,
        mcao:hasReusabilityLevel some mcao:None,
        acao:isUsedByTask some acao:RefactoringPractice,
        acao:isPartOfArtifact only acao:SADDDocument,
        acao:isUpdatedByTask some acao:PairProgrammingPractice,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        mcao:hasReusabilityLevel only mcao:None,
        acao:Artifact,
        acao:isUsedByTask some acao:DocumentationWrapUpTask,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:DocumentationWrapUpTask
             or acao:PairProgrammingPractice
             or acao:RefactoringPractice
             or acao:TestDrivenDevelopmentPractice),
        not (mcao:isSimilarToArtifact some acao:Artifact),
        acao:isCreatedByTask only acao:InitialRequirementsAnalysisTask,
        acao:isUpdatedByTask some acao:RefactoringPractice,
        acao:isUpdatedByTask only
            (acao:PairProgrammingPractice
             or acao:RefactoringPractice
             or acao:RequirementsAnalysisTask),
        acao:isUsedByTask some acao:ContinuousIntegrationPractice,
        acao:isCreatedByTask some acao:InitialRequirementsAnalysisTask


Class: acao:ApplicationIcon
```

```
    Annotations:
        rdfs:comment "Application icon is designed as needed for publishing
             process."@en,
        rdfs:label "Application Icon Android"

    SubClassOf:
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        acao:isPartOfArtifact some acao:DeploymentPackage,
        mcao:AppIcon,
        acao:isPartOfArtifact only acao:DeploymentPackage,
        acao:hasArtifactOrigin some acao:AndroidArtifact


Class: acao:ProductionizeActivities

    EquivalentTo:
        acao:isPerformedIn some acao:Productionize

    SubClassOf:
        acao:ActivitiesByPhases


Class: acao:Software

    Annotations:
        rdfs:comment "Represents software tools used during the entire project."@en

    SubClassOf:
        acao:ArtifactType


Class: mcao:ThrowAwayPrototype

    Annotations:
        rdfs:comment "Platform specific project created to test development
             environment and connected devices. This project is discarded."@en,
        rdfs:label "Throw-away Prototype"

    SubClassOf:
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:hasArtifactType only acao:Code,
        not (acao:isUsedByTask some acao:Task),
        acao:hasArtifactType some acao:Code,
        mcao:hasReusabilityLevel only mcao:None,
        acao:Artifact,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        not (acao:isUpdatedByTask some acao:Task),
        acao:isCreatedByTask some acao:EnvironmentSetUpTask,
        mcao:hasReusabilityLevel some mcao:None,
        acao:isCreatedByTask only acao:EnvironmentSetUpTask


Class: acao:InitialRequirementsAnalysisTask

    Annotations:
        rdfs:label "Initial Requirements Analysis Task",
        rdfs:comment "The purpose of this task is to carefully prioritize and
             analyze the requirements for finding a set of requirements that force
             tocreate the most important components and interfaces of the system. A
```

```
            working architectural skeleton should be found not later than by the
            end of the first iteration. "@en,
        acao:inActivity
<http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#PlanningDayIn0IterationAct
ivity>

    SubClassOf:
        acao:isPerformedIn only acao:PlanningDayIn0IterationActivity,
        acao:Task,
        acao:isPerformedIn some acao:PlanningDayIn0IterationActivity


Class: acao:PlanningDayIn0IterationTasks

    EquivalentTo:
        acao:isPerformedIn some acao:PlanningDayIn0IterationActivity

    SubClassOf:
        acao:TasksByActivities


Class: acao:ProductProposal

    Annotations:
        rdfs:label "Product Proposal",
        rdfs:comment "Generated before the development process. Describes the
            initial and general idea on the product."@en

    SubClassOf:
        acao:isUsedByTask some acao:InitialRequirementsCollectionTask,
        mcao:hasReusabilityLevel some mcao:Completely,
        mcao:hasReusabilityLevel only mcao:Completely,
        acao:isUsedByTask some acao:ArchitectureLineDefinitionTask,
        not (acao:isUpdatedByTask some acao:Task),
        acao:isUsedByTask only
            (acao:ArchitectureLineDefinitionTask
             or acao:CustomerEstablishmentTask
             or acao:InitialRequirementsCollectionTask
             or acao:ProcessEstablishmentTask),
        acao:isUsedByTask some acao:CustomerEstablishmentTask,
        acao:hasArtifactType some acao:Document,
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        mcao:isSimilarToArtifact some acao:ProductProposal,
        mcao:isSimilarToArtifact only acao:ProductProposal,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:Artifact,
        acao:hasArtifactType only acao:Document,
        acao:isUsedByTask some acao:ProcessEstablishmentTask,
        not (acao:isCreatedByTask some acao:Task)


Class: acao:UnitTest

    Annotations:
        rdfs:label "Unit Test Android",
        rdfs:comment "Unit test tests a single unit of code. It is created in
            separate project and references main project while performing
            different assertions."@en
```

```
    SubClassOf:
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        mcao:UnitTest,
        acao:hasArtifactOrigin some acao:AndroidArtifact


Class: mcao:ArtifactsOrigin

    SubClassOf:
        acao:Inferred


Class: acao:WorkingDayIn0IterationTasks

    EquivalentTo:
        acao:isPerformedIn some acao:WorkingDayIn0IterationActivity

    SubClassOf:
        acao:TasksByActivities


Class: wpcao:PageCS

    Annotations:
        rdfs:label "Page (C#)",
        rdfs:comment "Represents C# class that has the purpose of controlling the
             application view."@en

    SubClassOf:
        mcao:ViewController,
        acao:isPartOfArtifact some wpcao:CSCode,
        acao:isPartOfArtifact only wpcao:CSCode,
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact


Class: mcao:SourceCode

    Annotations:
        rdfs:label "Source Code",
        rdfs:comment "Platform specific source code developed during the
             implementation activities."@en

    SubClassOf:
        acao:hasArtifactType only acao:Code,
        acao:isUsedByTask some acao:SystemIntegrationTask,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:DocumentationWrapUpTask
             or acao:PairProgrammingPractice
             or acao:PreReleaseTestingTask
             or acao:PublishApplicationTask
             or acao:RefactoringPractice
             or acao:SystemIntegrationTask
             or acao:SystemTestTask
             or acao:TestDrivenDevelopmentPractice),
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:hasArtifactType some acao:Code,
```

```
        acao:isUpdatedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:PreReleaseTestingTask
             or acao:RefactoringPractice
             or acao:SystemIntegrationTask),
        acao:isUpdatedByTask some acao:SystemIntegrationTask,
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:isUpdatedByTask some acao:ContinuousIntegrationPractice,
        acao:isUsedByTask some acao:SystemTestTask,
        acao:isUsedByTask some acao:RefactoringPractice,
        mcao:hasReusabilityLevel only mcao:Partially,
        acao:isCreatedByTask some acao:PairProgrammingPractice,
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:Artifact,
        acao:isUsedByTask some acao:DocumentationWrapUpTask,
        mcao:isSimilarToArtifact some mcao:SourceCode,
        acao:isCreatedByTask only acao:PairProgrammingPractice,
        acao:isUpdatedByTask some acao:RefactoringPractice,
        acao:isUsedByTask some acao:PublishApplicationTask,
        acao:isUsedByTask some acao:PreReleaseTestingTask,
        acao:isUpdatedByTask some acao:PreReleaseTestingTask,
        acao:isUsedByTask some acao:ContinuousIntegrationPractice,
        mcao:isSimilarToArtifact only mcao:SourceCode


Class: acao:XMLResources

    Annotations:
        rdfs:comment "XML code describing application layout, menus, localized
             strings etc."@en,
        rdfs:label "XML Resources Android"

    SubClassOf:
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        mcao:isSimilarToArtifact only wpcao:XAMLDescription,
        mcao:AppResource,
        mcao:isSimilarToArtifact some wpcao:XAMLDescription,
        acao:isPartOfArtifact only acao:MobileApplication,
        acao:hasArtifactOrigin some acao:AndroidArtifact,
        acao:isPartOfArtifact some acao:MobileApplication


Class: acao:Resource

    Annotations:
        rdfs:comment "Represents resources that are created during the development
             process and are used in publishing purposes."@en

    SubClassOf:
        acao:ArtifactType


Class: acao:Document

    Annotations:
        rdfs:comment "Represents used documents or created artifacts that are
             published as documents during or at the end of development
             process."@en
```

```
    SubClassOf:
        acao:ArtifactType


Class: acao:SystemTestTasks

    EquivalentTo:
        acao:isPerformedIn some acao:SystemTestActivity

    SubClassOf:
        acao:TasksByActivities


Class: mcao:Completely

    SubClassOf:
        mcao:ReuseLevel


Class: acao:DocumentElement

    Annotations:
        rdfs:comment "Represents document that could be observed as stand-alone
             artifact, but is usually included in some other document."@en

    SubClassOf:
        acao:ArtifactType


Class: acao:APIDocumentation

    Annotations:
        rdfs:comment "Android API documentation from
             http://developers.android.com"@en,
        rdfs:label "API Documentation Android"

    SubClassOf:
        mcao:APIDocumentation,
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        acao:hasArtifactOrigin some acao:AndroidArtifact


Class: acao:PrototypeFunctionality

    Annotations:
        rdfs:label "Prototype Functionality Android",
        rdfs:comment "Developed functionality during the trial day. It prototypes
             some of the main application functionalities and is used to define the
             basic approach for implementing the similar functionalities in other
             iterations."@en

    SubClassOf:
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        acao:hasArtifactOrigin some acao:AndroidArtifact,
        mcao:AppPrototypeFunctionality


Class: acao:DocumentationWrapUpActivity
```

```
    Annotations:
        rdfs:label "Stabilize",
        acao:inPhase
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#SystemTestAnd
             Fix>,
        acao:inPhase
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#Stabilize>,
        rdfs:comment "The purpose of the Stabilize phase pattern is to ensure the
             quality of the implementation of the project."@en

    SubClassOf:
        acao:isPerformedIn only
            (acao:Stabilize
             or acao:SystemTestAndFix),
        acao:isPerformedIn some acao:SystemTestAndFix,
        acao:isPerformedIn some acao:Stabilize,
        acao:consistsOf only acao:DocumentationWrapUpTask,
        acao:Activity,
        acao:consistsOf some acao:DocumentationWrapUpTask


Class: acao:SystemTestReport

    Annotations:
        rdfs:comment "Final document on testing. Contains information on performed
             tests and issues detected."@en,
        rdfs:label "System Test Report"

    SubClassOf:
        not (acao:isUsedByTask some acao:Task),
        acao:isCreatedByTask some acao:SystemTestTask,
        not (acao:isUpdatedByTask some acao:Task),
        acao:isCreatedByTask only acao:SystemTestTask,
        mcao:hasReusabilityLevel some mcao:None,
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:hasArtifactType some acao:Document,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:Artifact,
        mcao:hasReusabilityLevel only mcao:None,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        acao:hasArtifactType only acao:Document


Class: acao:ClassModelWeb

    Annotations:
        rdfs:label "Class Model (Web)",
        rdfs:comment "UML class diagram describing the web application internal
             structure and created classes. This model is used in SADD
             document."@en

    SubClassOf:
        acao:isUpdatedByTask some acao:RequirementsAnalysisTask,
        mcao:hasReusabilityLevel some mcao:Completely,
        mcao:hasReusabilityLevel only mcao:Completely,
        acao:hasArtifactType some acao:Model,
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:isPartOfArtifact some acao:SADDDocument,
```

```
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:hasArtifactType only acao:Model,
        acao:isUsedByTask some acao:RefactoringPractice,
        acao:isPartOfArtifact only acao:SADDDocument,
        acao:isUpdatedByTask some acao:PairProgrammingPractice,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        mcao:isSimilarToArtifact some acao:ClassModelWeb,
        acao:Artifact,
        acao:isUsedByTask some acao:DocumentationWrapUpTask,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:DocumentationWrapUpTask
             or acao:PairProgrammingPractice
             or acao:RefactoringPractice
             or acao:TestDrivenDevelopmentPractice),
        mcao:isSimilarToArtifact only acao:ClassModelWeb,
        acao:isUpdatedByTask some acao:RefactoringPractice,
        acao:isCreatedByTask only acao:InitialRequirementsAnalysisTask,
        acao:isCreatedByTask some acao:InitialRequirementsAnalysisTask,
        acao:isUsedByTask some acao:ContinuousIntegrationPractice,
        acao:isUpdatedByTask only
            (acao:PairProgrammingPractice
             or acao:RefactoringPractice
             or acao:RequirementsAnalysisTask)


Class: acao:ActivitiesByPhases

    SubClassOf:
        acao:Inferred


Class: acao:PairProgrammingPractice

    Annotations:
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#WorkingDayAct
             ivity>,
        rdfs:comment "The purpose of Pair Programming is to improve communication,
             enhance process fidelity and spread knowledge within the team, and
             ensure quality of the code."@en,
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#WorkingDayIn0
             IterationActivity>,
        rdfs:label "Pair Programming Practice"

    SubClassOf:
        acao:isPerformedIn some acao:WorkingDayIn0IterationActivity,
        acao:Task,
        acao:isPerformedIn only
            (acao:WorkingDayActivity
             or acao:WorkingDayIn0IterationActivity),
        acao:isPerformedIn some acao:WorkingDayActivity


Class: acao:IterationsPlan

    Annotations:
```

```
        rdfs:comment "Contains the information about planned iterations along with
             selected features for specific iteration. This document is part of
             Product backlog document."@en,
        rdfs:label "Iterations Plan"

    SubClassOf:
        acao:isUsedByTask some acao:PostIterationWorkshopTask,
        acao:isUpdatedByTask some acao:WrapUpTask,
        mcao:hasReusabilityLevel some mcao:Completely,
        mcao:hasReusabilityLevel only mcao:Completely,
        acao:isCreatedByTask only acao:IterationPlanningTask,
        acao:isPartOfArtifact only acao:ProductBacklog,
        acao:isUsedByTask some acao:WrapUpTask,
        acao:isPartOfArtifact some acao:ProductBacklog,
        acao:isUsedByTask only
            (acao:DocumentationWrapUpTask
             or acao:PostIterationWorkshopTask
             or acao:WrapUpTask),
        mcao:isSimilarToArtifact some acao:IterationsPlan,
        mcao:isSimilarToArtifact only acao:IterationsPlan,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:hasArtifactType some acao:DocumentElement,
        acao:hasArtifactType only acao:DocumentElement,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:Artifact,
        acao:isUsedByTask some acao:DocumentationWrapUpTask,
        acao:isUpdatedByTask only acao:WrapUpTask,
        acao:isCreatedByTask some acao:IterationPlanningTask


Class: acao:TasksByActivities

    SubClassOf:
        acao:Inferred


Class: acao:BorrowedArtifacts

    EquivalentTo:
        acao:Artifact
         and (not (acao:isCreatedByTask some acao:Task))
         and (not (acao:isUpdatedByTask some acao:Task))
         and (acao:isUsedByTask some acao:Task)

    SubClassOf:
        mcao:ArtifactsUsage


Class: mcao:PartiallyReusableArtifacts

    EquivalentTo:
        acao:Artifact
         and (mcao:hasReusabilityLevel some mcao:Partially)

    SubClassOf:
        mcao:ArtifactsReusability


Class: acao:ProjectPlanChecklist
```

```
    Annotations:
        rdfs:comment "Mobile-D project plan checklist. This document is part of
             project plan."@en,
        rdfs:label "Project Plan Checklist"

    SubClassOf:
        acao:isUpdatedByTask some acao:WrapUpTask,
        not (acao:isUsedByTask some acao:Task),
        acao:isUpdatedByTask only
            (acao:DocumentationWrapUpTask
             or acao:IterationPlanningTask
             or acao:PostIterationWorkshopTask
             or acao:WrapUpTask),
        acao:isCreatedByTask some acao:InitialProjectPlanningTask,
        mcao:isSimilarToArtifact some acao:ProjectPlanChecklist,
        acao:isPartOfArtifact only acao:ProjectPlan,
        mcao:hasReusabilityLevel only mcao:Partially,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:isPartOfArtifact some acao:ProjectPlan,
        acao:isUpdatedByTask some acao:DocumentationWrapUpTask,
        acao:hasArtifactType some acao:DocumentElement,
        acao:hasArtifactType only acao:DocumentElement,
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:Artifact,
        mcao:isSimilarToArtifact only acao:ProjectPlanChecklist,
        acao:isCreatedByTask only acao:InitialProjectPlanningTask,
        acao:isUpdatedByTask some acao:IterationPlanningTask,
        acao:isUpdatedByTask some acao:PostIterationWorkshopTask


Class: acao:DataModelWeb

    Annotations:
        rdfs:comment "Entity-Relationship-Attribute model of the web application.
             It is presented in SADD document."@en,
        rdfs:label "Data Model (Web)"

    SubClassOf:
        acao:isUpdatedByTask some acao:RequirementsAnalysisTask,
        mcao:hasReusabilityLevel some mcao:Completely,
        mcao:hasReusabilityLevel only mcao:Completely,
        acao:hasArtifactType some acao:Model,
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:isPartOfArtifact some acao:SADDDocument,
        acao:hasArtifactType only acao:Model,
        acao:isUsedByTask some acao:RefactoringPractice,
        mcao:isSimilarToArtifact some acao:DataModelWeb,
        mcao:isSimilarToArtifact only acao:DataModelWeb,
        acao:isPartOfArtifact only acao:SADDDocument,
        acao:isUpdatedByTask some acao:PairProgrammingPractice,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:Artifact,
        acao:isUsedByTask some acao:DocumentationWrapUpTask,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
```

```
            or acao:DocumentationWrapUpTask
            or acao:PairProgrammingPractice
            or acao:RefactoringPractice
            or acao:TestDrivenDevelopmentPractice),
        acao:isUpdatedByTask some acao:RefactoringPractice,
        acao:isCreatedByTask only acao:InitialRequirementsAnalysisTask,
        acao:isCreatedByTask some acao:InitialRequirementsAnalysisTask,
        acao:isUpdatedByTask only
            (acao:PairProgrammingPractice
            or acao:RefactoringPractice
            or acao:RequirementsAnalysisTask),
        acao:isUsedByTask some acao:ContinuousIntegrationPractice


Class: acao:ProjectPlanChecklistTemplate

    Annotations:
        rdfs:label "Project Plan Checklist Template",
        rdfs:comment "Mobile-D project plan checklist."@en

    SubClassOf:
        mcao:isSimilarToArtifact some acao:ProjectPlanChecklistTemplate,
        mcao:hasReusabilityLevel some mcao:Completely,
        mcao:hasReusabilityLevel only mcao:Completely,
        acao:isUsedByTask only acao:InitialProjectPlanningTask,
        acao:isUsedByTask some acao:InitialProjectPlanningTask,
        acao:hasArtifactType some acao:Template,
        not (acao:isUpdatedByTask some acao:Task),
        acao:isPartOfArtifact some acao:ProjectPlanChecklist,
        acao:isPartOfArtifact only
            (acao:MobileDProcessLibrary
            or acao:ProjectPlanChecklist),
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:isPartOfArtifact some acao:MobileDProcessLibrary,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:Artifact,
        acao:hasArtifactType only acao:Template,
        mcao:isSimilarToArtifact only acao:ProjectPlanChecklistTemplate,
        not (acao:isCreatedByTask some acao:Task)


Class: acao:AndroidArtifact

    Annotations:
        rdfs:label "Android Artifact",
        rdfs:comment "Defines class of artifacts that are created in relation to
            Android development."@en

    SubClassOf:
        acao:ArtifactOrigin


Class: acao:IterationBacklog

    Annotations:
        rdfs:label "Iteration Backlog",
        rdfs:comment "Contains the information on specific iteration including
            story and task cards. Each iteration document is created from scratch.
            It is part of Product backlog document."@en
```

```
    SubClassOf:
        acao:isUpdatedByTask some acao:WrapUpTask,
        acao:isUsedByTask some acao:PostIterationWorkshopTask,
        acao:isUpdatedByTask some acao:RequirementsAnalysisTask,
        acao:isCreatedByTask only acao:IterationPlanningTask,
        acao:isPartOfArtifact only acao:ProductBacklog,
        acao:isUsedByTask some acao:WrapUpTask,
        acao:isPartOfArtifact some acao:ProductBacklog,
        mcao:isSimilarToArtifact only acao:IterationBacklog,
        acao:isUsedByTask only
            (acao:DocumentationWrapUpTask
             or acao:PostIterationWorkshopTask
             or acao:WrapUpTask),
        mcao:isSimilarToArtifact some acao:IterationBacklog,
        mcao:hasReusabilityLevel only mcao:Partially,
        acao:isUpdatedByTask some acao:PairProgrammingPractice,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:hasArtifactType only acao:DocumentElement,
        acao:hasArtifactType some acao:DocumentElement,
        acao:isUpdatedByTask only
            (acao:PairProgrammingPractice
             or acao:RequirementsAnalysisTask
             or acao:WrapUpTask),
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:Artifact,
        acao:isUsedByTask some acao:DocumentationWrapUpTask,
        acao:isCreatedByTask some acao:IterationPlanningTask


Class: acao:ProjectSetUpActivity

    Annotations:
        rdfs:comment "The purpose of this stage is to 1) set-up the physical and
             technical resources for the project as well as the environment for
             project monitoring, 2) train the project team as necessary, and 3)
             establish the project specific ways to communicate with the customer
             group. All the tasks of Project Set-Up include the participation of
             project team."@en,
        rdfs:label "Project SetUp",
        acao:inPhase
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#Initialize>

    SubClassOf:
        acao:consistsOf only
            (acao:CustomerCommunicationEstablishmentTask
             or acao:EnvironmentSetUpTask),
        acao:consistsOf some acao:CustomerCommunicationEstablishmentTask,
        acao:isPerformedIn some acao:Initialize,
        acao:Activity,
        acao:consistsOf some acao:EnvironmentSetUpTask,
        acao:isPerformedIn only acao:Initialize


Class: acao:IterationPlanningTask

    Annotations:
```

```
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#PlanningDayAc
             tivity>,
        rdfs:comment "The purpose of this task is to generate the schedule and
             contents for the iteration to execute. The contents are defined in
             terms of tasks which are work orders for the team."@en,
        rdfs:label "Iteration Planning Task",
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#PlanningDayIn
             0IterationActivity>

    SubClassOf:
        acao:isPerformedIn only
            (acao:PlanningDayActivity
             or acao:PlanningDayIn0IterationActivity),
        acao:Task,
        acao:isPerformedIn some acao:PlanningDayActivity,
        acao:isPerformedIn some acao:PlanningDayIn0IterationActivity


Class: wpcao:PageXAMLElement

    Annotations:
        rdfs:label "Page (XAML) Element",
        rdfs:comment "Represents XAML code that is used to describe any user
             interface element such as text box, list box, button etc."@en

    SubClassOf:
        mcao:ViewElement,
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact,
        acao:isPartOfArtifact only wpcao:PageXAML,
        acao:isPartOfArtifact some wpcao:PageXAML


Class: wpcao:PrototypeFunctionality

    Annotations:
        rdfs:label "Prototype Functionality WP",
        rdfs:comment "Developed functionality during the trial day. It prototypes
             some of the main application functionalities and is used to define the
             basic approach for implementing the similar functionalities in other
             iterations."@en

    SubClassOf:
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact,
        mcao:AppPrototypeFunctionality


Class: mcao:IntegrationTest

    Annotations:
        rdfs:label "Integration Test",
        mcao:NOTICE "Closure axiom for some properties are created in leaf
             elements."@en,
        rdfs:comment "Platform specific, robotized or manual integration test
             document."@en
```

```
    SubClassOf:
        acao:isUsedByTask some acao:SystemIntegrationTask,
        acao:isUpdatedByTask some acao:SystemIntegrationTask,
        acao:isCreatedByTask some acao:TestDrivenDevelopmentPractice,
        acao:isUpdatedByTask some acao:ContinuousIntegrationPractice,
        mcao:hasReusabilityLevel some mcao:None,
        acao:isUsedByTask some acao:SystemTestTask,
        acao:Artifact,
        mcao:hasReusabilityLevel only mcao:None,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        acao:isCreatedByTask only acao:TestDrivenDevelopmentPractice,
        acao:isUsedByTask some acao:PreReleaseTestingTask,
        acao:isUpdatedByTask some acao:PreReleaseTestingTask,
        acao:isUsedByTask some acao:ContinuousIntegrationPractice


Class: mcao:ExampleCode

    Annotations:
        rdfs:comment "Example Code",
        rdfs:comment "Platform specific example code on different topics found on
             the internet from various sources."@en

    SubClassOf:
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:hasArtifactType only acao:Example,
        acao:hasArtifactType some acao:Example,
        mcao:hasReusabilityLevel only mcao:None,
        acao:Artifact,
        acao:isUsedByTask only acao:PairProgrammingPractice,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        not (acao:isUpdatedByTask some acao:Task),
        acao:isUsedByTask some acao:PairProgrammingPractice,
        mcao:hasReusabilityLevel some mcao:None,
        not (acao:isCreatedByTask some acao:Task)


Class: acao:ThrowAwayPrototype

    Annotations:
        rdfs:label "Throw Away Prototype Android",
        rdfs:comment "Project created to test development environment and connected
             devices. This project is discarded."@en

    SubClassOf:
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        acao:hasArtifactOrigin some acao:AndroidArtifact,
        mcao:ThrowAwayPrototype


Class: acao:IntegrationTest

    Annotations:
        rdfs:label "Integration Test Android",
        rdfs:comment "Robotized test which tests application integrated
             functionality."@en

    SubClassOf:
        not (acao:isPartOfArtifact some acao:Artifact),
```

```
        acao:isUpdatedByTask some acao:PairProgrammingPractice,
        acao:hasArtifactType only acao:Code,
        acao:isUsedByTask only
            (acao:AcceptanceTestingTask
             or acao:ContinuousIntegrationPractice
             or acao:PairProgrammingPractice
             or acao:PreReleaseTestingTask
             or acao:RefactoringPractice
             or acao:SystemIntegrationTask
             or acao:SystemTestTask),
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        acao:isUsedByTask some acao:AcceptanceTestingTask,
        acao:isUpdatedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:PairProgrammingPractice
             or acao:PreReleaseTestingTask
             or acao:RefactoringPractice
             or acao:SystemIntegrationTask),
        acao:hasArtifactType some acao:Code,
        mcao:IntegrationTest,
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:hasArtifactOrigin some acao:AndroidArtifact,
        acao:isUsedByTask some acao:RefactoringPractice,
        acao:isUpdatedByTask some acao:RefactoringPractice


Class: wpcao:WindowsPhoneArtifacts

    EquivalentTo:
        acao:Artifact
         and (acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact)

    SubClassOf:
        mcao:ArtifactsOrigin


Class: acao:Initialize

    Annotations:
        rdfs:label "Initialize Phase",
        rdfs:comment  "The  Initialize  phase  should  describe  and  prepare  all
             components  of  the  application  as  well  as  to  predict  the  possible
             critical issues of the project. Initialize phase is usually called a
             zero iteration (0-iteration) phase as it in addition to project set-up
             includes the stages of planning day, working day and release day which
             are also used in productionize phase. The idea of the 0-iteration
             phase is to assure the functionality of the technical development
             environment  through  the  implementation  of  some  representative
             features. Additionally, in this phase some prototyping could be done
             in order to decide which technological solution would be the most
             appropriate for the rest of the development process."@en

    SubClassOf:
        acao:consistsOf some acao:WorkingDayIn0IterationActivity,
        acao:consistsOf some acao:ProjectSetUpActivity,
        acao:Phase,
        acao:consistsOf some acao:PlanningDayIn0IterationActivity,
        acao:consistsOf only
            (acao:PlanningDayIn0IterationActivity
```

```
            or acao:ProjectSetUpActivity
            or acao:WorkingDayIn0IterationActivity)


Class: acao:InformCustomerTask

    Annotations:
        rdfs:label "Inform Customer Task",
        rdfs:comment "The purpose of this task is to provide an honest view of the
             progress to the customer, and to give the customer a possibility to
             give feedback about the implemented features and to guide the
             development."@en,
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#WorkingDayAct
             ivity>

    SubClassOf:
        acao:isPerformedIn only acao:WorkingDayActivity,
        acao:Task,
        acao:isPerformedIn some acao:WorkingDayActivity


Class: acao:Standard

    Annotations:
        rdfs:comment "Represents document containing formal and internationally
             recognized description of some concept or element."@en

    SubClassOf:
        acao:ArtifactType


Class: mcao:Partially

    SubClassOf:
        mcao:ReuseLevel


Class: mcao:ReusableArtifacts

    EquivalentTo:
        acao:Artifact
         and (mcao:CompletlyResuableArtifacts
         or mcao:PartiallyReusableArtifacts)

    SubClassOf:
        mcao:ArtifactsReusability

    DisjointWith:
        mcao:NotreusableArtifacts


Class: acao:DefectList

    Annotations:
        rdfs:label "Defect List",
        rdfs:comment "Document created after testing is performed. It contains
             found issues and planned activities. At the end this document becomes
             part of System test report document."@en
```

```
    SubClassOf:
        not (acao:isUsedByTask some acao:Task),
        acao:isCreatedByTask only acao:AcceptanceTestingTask,
        acao:isCreatedByTask some acao:AcceptanceTestingTask,
        mcao:isSimilarToArtifact only acao:DefectList,
        mcao:hasReusabilityLevel only mcao:Partially,
        acao:isUpdatedByTask some acao:SystemTestTask,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:hasArtifactType only acao:DocumentElement,
        acao:hasArtifactType some acao:DocumentElement,
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        mcao:isSimilarToArtifact some acao:DefectList,
        acao:Artifact,
        acao:isPartOfArtifact only acao:SystemTestReport,
        acao:isPartOfArtifact some acao:SystemTestReport,
        acao:isUpdatedByTask only
            (acao:PreReleaseTestingTask
             or acao:SystemTestTask),
        acao:isUpdatedByTask some acao:PreReleaseTestingTask


Class: acao:Example

    Annotations:
        rdfs:comment "Represents code artifacts created by third party and used as
             examples of implemented functionality or to solve some programming
             issue."@en

    SubClassOf:
        acao:ArtifactType


Class: acao:LocalizationString

    Annotations:
        rdfs:label "Localization String Android",
        rdfs:comment "Represent XML code that is used to provide localized
             translation of values according to value unique key."@en

    SubClassOf:
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        mcao:isSimilarToArtifact some wpcao:ResourceFile,
        acao:isPartOfArtifact some acao:XMLResources,
        mcao:AppResource,
        acao:hasArtifactOrigin some acao:AndroidArtifact,
        mcao:isSimilarToArtifact only wpcao:ResourceFile,
        acao:isPartOfArtifact only acao:XMLResources


Class: acao:ProductBacklog

    Annotations:
        rdfs:comment "Contains the information on features that are (to be)
             implemented in the development process, through several iterations.
             Users can contribute in defining the features/stories."@en,
        rdfs:label "Product Backlog"
```

```
    SubClassOf:
        acao:isUpdatedByTask some acao:WrapUpTask,
        acao:isUsedByTask some acao:PostIterationWorkshopTask,
        acao:isUpdatedByTask some acao:RequirementsAnalysisTask,
        acao:isUsedByTask only
            (acao:IterationPlanningTask
             or acao:PostIterationWorkshopTask),
        mcao:hasReusabilityLevel only mcao:Partially,
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:isUsedByTask some acao:IterationPlanningTask,
        acao:hasArtifactType some acao:Document,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:isUpdatedByTask only
            (acao:RequirementsAnalysisTask
             or acao:WrapUpTask),
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        mcao:isSimilarToArtifact some acao:ProductBacklog,
        acao:Artifact,
        acao:hasArtifactType only acao:Document,
        acao:isCreatedByTask only acao:InitialRequirementsAnalysisTask,
        mcao:isSimilarToArtifact only acao:ProductBacklog,
        acao:isCreatedByTask some acao:InitialRequirementsAnalysisTask


Class: wpcao:ApplicationDescription

    Annotations:
        rdfs:label "Application Description WP",
        rdfs:comment "Short but important description used for publishing process.
             It includes the information on application, category, authors etc."@en

    SubClassOf:
        mcao:AppDescription,
        acao:isPartOfArtifact some wpcao:DeploymentPackage,
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact,
        acao:isPartOfArtifact only
            (acao:SADDDocument
             or wpcao:DeploymentPackage)


Class: mcao:OnlyUsedDocuments

    EquivalentTo:
        acao:Artifact
         and acao:BorrowedArtifacts
         and (not (acao:isPartOfArtifact some acao:Artifact))
         and (acao:hasArtifactType some acao:Document)

    SubClassOf:
        mcao:ArtifactsUsage


Class: acao:AcceptanceTest

    Annotations:
        rdfs:label "Acceptance Test",
```

```
        rdfs:comment "Created during initial requirements analysis. Contains the
             information on acceptance test of one product feature. Can include
             different contexts, and test scenarios with sample data."@en

    SubClassOf:
        acao:isPartOfArtifact only acao:SystemTestPlan,
        mcao:hasReusabilityLevel some mcao:Completely,
        acao:isUpdatedByTask some acao:RequirementsAnalysisTask,
        mcao:hasReusabilityLevel only mcao:Completely,
        acao:isPartOfArtifact some acao:SystemTestPlan,
        acao:isUsedByTask some acao:AcceptanceTestingTask,
        acao:isUpdatedByTask some acao:AcceptanceTestingTask,
        acao:isUpdatedByTask only
            (acao:AcceptanceTestGenerationTask
             or acao:AcceptanceTestReviewTask
             or acao:AcceptanceTestingTask
             or acao:DocumentationWrapUpTask
             or acao:RequirementsAnalysisTask),
        mcao:isSimilarToArtifact some acao:AcceptanceTest,
        acao:isUsedByTask some acao:SystemTestTask,
        mcao:isSimilarToArtifact only acao:AcceptanceTest,
        acao:isUsedByTask some acao:IterationPlanningTask,
        acao:isUsedByTask only
            (acao:AcceptanceTestReviewTask
             or acao:AcceptanceTestingTask
             or acao:IterationPlanningTask
             or acao:PreReleaseTestingTask
             or acao:SystemTestTask),
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:hasArtifactType some acao:DocumentElement,
        acao:isUpdatedByTask some acao:DocumentationWrapUpTask,
        acao:hasArtifactType only acao:DocumentElement,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:Artifact,
        acao:isUpdatedByTask some acao:AcceptanceTestReviewTask,
        acao:isUsedByTask some acao:AcceptanceTestReviewTask,
        acao:isUsedByTask some acao:PreReleaseTestingTask,
        acao:isCreatedByTask only acao:InitialRequirementsAnalysisTask,
        acao:isCreatedByTask some acao:InitialRequirementsAnalysisTask,
        acao:isUpdatedByTask some acao:AcceptanceTestGenerationTask


Class: wpcao:PageXAML

    Annotations:
        rdfs:comment "Represents XAML code that is used to describe user interface
             form or screen."@en,
        rdfs:label "Page (XAML)"

    SubClassOf:
        mcao:View,
        acao:isPartOfArtifact some wpcao:XAMLDescription,
        acao:isPartOfArtifact only wpcao:XAMLDescription,
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact


Class: acao:DevelopmentEnvironment
```

```
    Annotations:
        rdfs:comment "Set of applications used for Android development. We used
             Eclipse base SDK."@en,
        rdfs:label "Development Environment Android"

    SubClassOf:
        mcao:DevelopmentEnvironment


Class: acao:SystemTestTask

    Annotations:
        rdfs:comment "The purpose of this task is to find defects in the produced
             software after the implementation phase of the project. The System
             Test procedure provides defect information for last fixing iteration
             of the Mobile-D process."@en,
        rdfs:label "System Test Task",
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#SystemTestAct
             ivity>

    SubClassOf:
        acao:isPerformedIn only acao:SystemTestActivity,
        acao:Task,
        acao:isPerformedIn some acao:SystemTestActivity


Class: acao:PlanningDayTasks

    EquivalentTo:
        acao:isPerformedIn some acao:PlanningDayActivity

    SubClassOf:
        acao:TasksByActivities


Class: mcao:ArtifactsUsage

    SubClassOf:
        acao:Inferred


Class: mcao:ViewElement

    Annotations:
        rdfs:label "View Element",
        rdfs:comment "Represents, usually XML based, code that is used to describe
             any user interface element such as text box, list box, button etc."@en

    SubClassOf:
        acao:hasArtifactType only acao:Code,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:PairProgrammingPractice
             or acao:PreReleaseTestingTask
             or acao:RefactoringPractice
             or acao:SystemIntegrationTask
             or acao:SystemTestTask
             or acao:TestDrivenDevelopmentPractice),
```

```
        acao:isUsedByTask some acao:SystemIntegrationTask,
        acao:hasArtifactType some acao:Code,
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:isUpdatedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:PreReleaseTestingTask
             or acao:RefactoringPractice
             or acao:SystemIntegrationTask),
        acao:isUpdatedByTask some acao:SystemIntegrationTask,
        acao:isUsedByTask some acao:PairProgrammingPractice,
        mcao:isSimilarToArtifact some mcao:ViewElement,
        acao:isUpdatedByTask some acao:ContinuousIntegrationPractice,
        acao:isUsedByTask some acao:SystemTestTask,
        acao:isUsedByTask some acao:RefactoringPractice,
        mcao:hasReusabilityLevel only mcao:Partially,
        mcao:isSimilarToArtifact only mcao:ViewElement,
        acao:isCreatedByTask some acao:PairProgrammingPractice,
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:Artifact,
        acao:isCreatedByTask only acao:PairProgrammingPractice,
        acao:isUsedByTask some acao:PreReleaseTestingTask,
        acao:isUpdatedByTask some acao:RefactoringPractice,
        acao:isUsedByTask some acao:ContinuousIntegrationPractice,
        acao:isUpdatedByTask some acao:PreReleaseTestingTask


Class: wpcao:CSCode

    Annotations:
        rdfs:label "CS Code",
        rdfs:comment "C# code developed during the implementation activities."@en

    SubClassOf:
        acao:isPartOfArtifact only wpcao:MobileApplication,
        acao:isPartOfArtifact some wpcao:MobileApplication,
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact,
        mcao:SourceCode


Class: acao:OtherArtifacts

    EquivalentTo:
        acao:Artifact
         and (acao:hasArtifactOrigin some acao:OtherArtifact)

    SubClassOf:
        mcao:ArtifactsOrigin


Class: acao:AcceptanceTestingTask

    Annotations:
        rdfs:comment "The purpose of this task is to verify that the requirements
             the customer has set for the software are implemented correctly."@en,
        rdfs:label "Acceptance Testing Task",
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#ReleaseDayAct
             ivity>
```

```
    SubClassOf:
        acao:isPerformedIn only acao:ReleaseDayActivity,
        acao:isPerformedIn some acao:ReleaseDayActivity,
        acao:Task


Class: acao:ReleaseDayTasks

    EquivalentTo:
        acao:isPerformedIn some acao:ReleaseDayActivity

    SubClassOf:
        acao:TasksByActivities


Class: acao:DataModelMobile

    Annotations:
        rdfs:comment "Entity-Relationship-Attribute model of the mobile database.
             It is presented in SADD document."@en,
        rdfs:label "Data Model (Mobile)"

    SubClassOf:
        mcao:hasReusabilityLevel some mcao:Completely,
        mcao:hasReusabilityLevel only mcao:Completely,
        acao:hasArtifactType some acao:Model,
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:isUpdatedByTask only
            (acao:PairProgrammingPractice
             or acao:RefactoringPractice),
        acao:isPartOfArtifact some acao:SADDDocument,
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:hasArtifactType only acao:Model,
        acao:isUsedByTask some acao:RefactoringPractice,
        acao:isUpdatedByTask some acao:PairProgrammingPractice,
        acao:isPartOfArtifact only acao:SADDDocument,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:Artifact,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:DocumentationWrapUpTask
             or acao:PairProgrammingPractice
             or acao:RefactoringPractice
             or acao:TestDrivenDevelopmentPractice),
        acao:isUsedByTask some acao:DocumentationWrapUpTask,
        acao:isUpdatedByTask some acao:RefactoringPractice,
        acao:isCreatedByTask only acao:InitialRequirementsAnalysisTask,
        acao:isCreatedByTask some acao:InitialRequirementsAnalysisTask,
        acao:isUsedByTask some acao:ContinuousIntegrationPractice,
        mcao:isSimilarToArtifact only acao:DataModelMobile,
        mcao:isSimilarToArtifact some acao:DataModelMobile


Class: mcao:MobileApplication

    Annotations:
```

```
        rdfs:comment "The mobile application targeting one specific platform that
             is created in the development process."@en,
        rdfs:label "Mobile Application"

    SubClassOf:
        not (acao:isPartOfArtifact some acao:Artifact),
        not (acao:isUsedByTask some acao:Task),
        acao:isUpdatedByTask only acao:PublishApplicationTask,
        acao:isUpdatedByTask some acao:PublishApplicationTask,
        acao:isCreatedByTask some acao:PairProgrammingPractice,
        acao:Artifact,
        mcao:hasReusabilityLevel only mcao:None,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        mcao:hasReusabilityLevel some mcao:None,
        acao:isCreatedByTask only acao:PairProgrammingPractice,
        acao:hasArtifactType some acao:Product,
        acao:hasArtifactType only acao:Product


Class: acao:ScopeDefinitionTasks

    EquivalentTo:
        acao:isPerformedIn some acao:ScopeDefinitionActivity

    SubClassOf:
        acao:TasksByActivities


Class: acao:ApplicationDescription

    Annotations:
        rdfs:comment "Short but important description used for publishing process.
             It includes the information on application, category, authors
             etc."@en,
        rdfs:label "Application Description Android"

    SubClassOf:
        mcao:AppDescription,
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        acao:isPartOfArtifact some acao:DeploymentPackage,
        acao:hasArtifactOrigin some acao:AndroidArtifact,
        acao:isPartOfArtifact only
            (acao:DeploymentPackage
             or acao:SADDDocument)


Class: acao:WorkingDayActivity

    Annotations:
        acao:inPhase
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#Productionize
             >,
        acao:inPhase
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#SystemTestAnd
             Fix>,
        acao:inPhase
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#Stabilize>,
        rdfs:label "Working Day",
```

```
        rdfs:comment "The purpose of this stage is to implement the system
             functionality planned during the planning day. The development team
             focuses on highest priority functionality as defined by the customer.
             Working Days are used in Productionize, Stabilize and System Test &
             Fix phases. One iteration may contain 1-n Working days. Working Days
             form the actual development days of the iteration."@en

    SubClassOf:
        acao:consistsOf only
            (acao:ContinuousIntegrationPractice
             or acao:InformCustomerTask
             or acao:PairProgrammingPractice
             or acao:RefactoringPractice
             or acao:TestDrivenDevelopmentPractice
             or acao:WrapUpTask),
        acao:consistsOf some acao:RefactoringPractice,
        acao:isPerformedIn some acao:Productionize,
        acao:isPerformedIn some acao:SystemTestAndFix,
        acao:isPerformedIn some acao:Stabilize,
        acao:consistsOf some acao:WrapUpTask,
        acao:isPerformedIn only
            (acao:Productionize
             or acao:Stabilize
             or acao:SystemTestAndFix),
        acao:consistsOf some acao:ContinuousIntegrationPractice,
        acao:consistsOf some acao:PairProgrammingPractice,
        acao:consistsOf some acao:TestDrivenDevelopmentPractice,
        acao:consistsOf some acao:InformCustomerTask,
        acao:Activity


Class: acao:UsedAndProducedDocuments

    EquivalentTo:
        acao:Artifact
         and (not (acao:isPartOfArtifact some acao:Artifact))
         and (acao:hasArtifactType some acao:Document)

    SubClassOf:
        mcao:ArtifactsUsage


Class: acao:FinalProducts

    EquivalentTo:
        acao:Artifact
         and (not (acao:BorrowedArtifacts))
         and (not (acao:isPartOfArtifact some acao:Artifact))
         and (acao:hasArtifactType some acao:Product)

    SubClassOf:
        mcao:ArtifactsUsage


Class: acao:WorkingDayTasks

    EquivalentTo:
        acao:isPerformedIn some acao:WorkingDayActivity
```

```
    SubClassOf:
        acao:TasksByActivities


Class: acao:SystemTestAndFixActivities

    EquivalentTo:
        acao:isPerformedIn some acao:SystemTestAndFix

    SubClassOf:
        acao:ActivitiesByPhases


Class: mcao:NotreusableArtifacts

    EquivalentTo:
        acao:Artifact
         and (mcao:hasReusabilityLevel some mcao:None)

    SubClassOf:
        mcao:ArtifactsReusability

    DisjointWith:
        mcao:ReusableArtifacts


Class: acao:ReleaseDayActivity

    Annotations:
        acao:inPhase
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#Productionize
             >,
        acao:inPhase
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#SystemTestAnd
             Fix>,
        acao:inPhase
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#Stabilize>,
        rdfs:comment "The purpose in this stage is to make a fully working release
             of the system under development."@en,
        rdfs:label "Release Day"

    SubClassOf:
        acao:isPerformedIn some acao:Productionize,
        acao:consistsOf some acao:AcceptanceTestingTask,
        acao:consistsOf some acao:ReleaseCeremoniesTask,
        acao:consistsOf some acao:SystemIntegrationTask,
        acao:isPerformedIn some acao:SystemTestAndFix,
        acao:isPerformedIn some acao:Stabilize,
        acao:isPerformedIn only
            (acao:Productionize
             or acao:Stabilize
             or acao:SystemTestAndFix),
        acao:Activity,
        acao:consistsOf only
            (acao:AcceptanceTestingTask
             or acao:PreReleaseTestingTask
             or acao:ReleaseCeremoniesTask
             or acao:SystemIntegrationTask),
        acao:consistsOf some acao:PreReleaseTestingTask
```

```
Class: wpcao:ThrowAwayPrototype

    Annotations:
        rdfs:label "Throw Away Prototype WP",
        rdfs:comment "Project created to test development environment and connected
             devices. This project is discarded."@en

    SubClassOf:
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact,
        mcao:ThrowAwayPrototype


Class: mcao:UMLClassSDK

    Annotations:
        rdfs:label "SDK UML Class ",
        rdfs:comment "UML model element used to describe an existing platform
             specific class that is to be used."@en

    SubClassOf:
        acao:isUsedByTask some acao:RequirementsAnalysisTask,
        acao:isPartOfArtifact only acao:ClassModelMobile,
        acao:hasArtifactType only acao:ModelElement,
        mcao:isSimilarToArtifact some mcao:UMLClassSDK,
        acao:isPartOfArtifact some acao:ClassModelMobile,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:DocumentationWrapUpTask
             or acao:InitialRequirementsAnalysisTask
             or acao:PairProgrammingPractice
             or acao:RefactoringPractice
             or acao:RequirementsAnalysisTask
             or acao:TestDrivenDevelopmentPractice),
        acao:isUsedByTask some acao:InitialRequirementsAnalysisTask,
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        not (acao:isUpdatedByTask some acao:Task),
        acao:isUsedByTask some acao:PairProgrammingPractice,
        mcao:isSimilarToArtifact only mcao:UMLClassSDK,
        acao:isUsedByTask some acao:RefactoringPractice,
        mcao:hasReusabilityLevel only mcao:Partially,
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:Artifact,
        acao:isUsedByTask some acao:DocumentationWrapUpTask,
        acao:hasArtifactType some acao:ModelElement,
        acao:isUsedByTask some acao:ContinuousIntegrationPractice,
        not (acao:isCreatedByTask some acao:Task)


Class: acao:PreReleaseTestingTask

    Annotations:
        acao:inActivity "The purpose of this task is to make sure that the software
             being produced is ready for the Acceptance Testing and release."@en,
        rdfs:label "Pre-release Testing Task",
```

```
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#ReleaseDayAct
             ivity>

    SubClassOf:
        acao:isPerformedIn only acao:ReleaseDayActivity,
        acao:isPerformedIn some acao:ReleaseDayActivity,
        acao:Task


Class: wpcao:ExampleCode

    Annotations:
        rdfs:comment "WP example code on different topics found on the internet
             from various sources."@en,
        rdfs:label "Example Code WP"

    SubClassOf:
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact,
        mcao:ExampleCode


Class: wpcao:XAMLDescription

    Annotations:
        rdfs:label "XAML Description WP",
        rdfs:comment "XML based XAML code describing application layout and layout
             elements."@en

    SubClassOf:
        mcao:AppResource,
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact,
        acao:isPartOfArtifact only acao:MobileApplication,
        acao:isPartOfArtifact some acao:MobileApplication,
        mcao:isSimilarToArtifact some acao:XMLResources,
        mcao:isSimilarToArtifact only acao:XMLResources


Class: acao:MobileApplication

    Annotations:
        rdfs:comment "The mobile application created in the development
             process."@en,
        rdfs:label "Mobile Application Android"

    SubClassOf:
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        mcao:MobileApplication,
        acao:hasArtifactOrigin some acao:AndroidArtifact


Class: acao:ScopeDefinitionActivity

    Annotations:
        rdfs:label "Scope Definition",
        acao:inPhase
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#Explore>,
```

```
        rdfs:comment "The purpose of this stage is to define the goals for the
             incipient project regarding both the contents as well as the timeline
             of the project."@en

    SubClassOf:
        acao:consistsOf only
            (acao:InitialProjectPlanningTask
             or acao:InitialRequirementsCollectionTask),
        acao:isPerformedIn some acao:Explore,
        acao:consistsOf some acao:InitialRequirementsCollectionTask,
        acao:Activity,
        acao:consistsOf some acao:InitialProjectPlanningTask,
        acao:isPerformedIn only acao:Explore


Class: acao:Inferred

    Annotations:
        rdfs:comment "Inferred knowledge from the ontology description. "@en


Class: mcao:AppIcon

    Annotations:
        rdfs:label "App Icon",
        rdfs:comment "Application icon is designed as needed for publishing
             process. It is platform specific artifact."@en

    SubClassOf:
        not (acao:isUsedByTask some acao:Task),
        acao:isCreatedByTask some acao:PublishApplicationTask,
        mcao:isSimilarToArtifact only mcao:AppIcon,
        acao:hasArtifactType only acao:Resource,
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:isCreatedByTask only acao:PublishApplicationTask,
        acao:Artifact,
        not (acao:isUpdatedByTask some acao:Task),
        acao:hasArtifactType some acao:Resource,
        mcao:isSimilarToArtifact some mcao:AppIcon,
        mcao:hasReusabilityLevel only mcao:Partially


Class: acao:ProjectEstablishmentActivity

    Annotations:
        rdfs:label "Project Establishment",
        acao:inPhase
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#Explore>,
        rdfs:comment "The purpose of this stage is to define and allocate the
             resources (both technical and human) needed for incipient software
             development project. Also the establishment of the baseline process
             for the project is an important task of this stage. The Project
             Establishment phase is in order to make sure that the project team can
             start the actual software development without delays caused by, for
             example, missing tools and proper training. "@en

    SubClassOf:
        acao:isPerformedIn some acao:Explore,
        acao:consistsOf some acao:ProcessEstablishmentTask,
```

```
        acao:consistsOf only
            (acao:ArchitectureLineDefinitionTask
             or acao:ProcessEstablishmentTask),
        acao:Activity,
        acao:isPerformedIn only acao:Explore,
        acao:consistsOf some acao:ArchitectureLineDefinitionTask


Class: mcao:ArtifactsReusability

    SubClassOf:
        acao:Inferred


Class: acao:WebServiceSpecification

    Annotations:
        rdfs:label "Web Service Specification",
        rdfs:comment "Contains information on exposed web services along with
             available methods, their parameters and other communication
             elements."@en

    SubClassOf:
        acao:isUpdatedByTask some acao:RequirementsAnalysisTask,
        mcao:hasReusabilityLevel some mcao:Completely,
        mcao:hasReusabilityLevel only mcao:Completely,
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:isPartOfArtifact some acao:SADDDocument,
        acao:isUsedByTask some acao:RefactoringPractice,
        acao:isUpdatedByTask some acao:PairProgrammingPractice,
        acao:isPartOfArtifact only acao:SADDDocument,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:hasArtifactType some acao:DocumentElement,
        acao:hasArtifactType only acao:DocumentElement,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:Artifact,
        acao:isUsedByTask some acao:DocumentationWrapUpTask,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:DocumentationWrapUpTask
             or acao:PairProgrammingPractice
             or acao:RefactoringPractice
             or acao:TestDrivenDevelopmentPractice),
        acao:isCreatedByTask only acao:InitialRequirementsAnalysisTask,
        mcao:isSimilarToArtifact some acao:WebServiceSpecification,
        acao:isUpdatedByTask some acao:RefactoringPractice,
        acao:isUsedByTask some acao:ContinuousIntegrationPractice,
        acao:isCreatedByTask some acao:InitialRequirementsAnalysisTask,
        acao:isUpdatedByTask only
            (acao:PairProgrammingPractice
             or acao:RefactoringPractice
             or acao:RequirementsAnalysisTask),
        mcao:isSimilarToArtifact only acao:WebServiceSpecification


Class: acao:ExploreActivities

    EquivalentTo:
```

```
        acao:isPerformedIn some acao:Explore

    SubClassOf:
        acao:ActivitiesByPhases


Class: acao:ApplicationScreenshot

    Annotations:
        rdfs:comment "Application screenshots are created as needed for publishing
             process."@en,
        rdfs:label "Application Screenshot Android"

    SubClassOf:
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        mcao:AppScreenshot,
        acao:isPartOfArtifact some acao:DeploymentPackage,
        acao:hasArtifactOrigin some acao:AndroidArtifact,
        acao:isPartOfArtifact only
            (acao:DeploymentPackage
             or acao:SADDDocument)


Class: acao:StabilizeActivities

    EquivalentTo:
        acao:isPerformedIn some acao:Stabilize

    SubClassOf:
        acao:ActivitiesByPhases


Class: acao:ArtifactOrigin

    Annotations:
        rdfs:comment "Classification of artifacts in types according to their
             origin."@en

    EquivalentTo:
        acao:AndroidArtifact
         or acao:MethodologicalArtifact
         or acao:OtherArtifact
         or acao:ServiceArtifact
         or wpcao:WindowsPhoneArtifact


Class: acao:AcceptanceTestReviewTask

    Annotations:
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#PlanningDayAc
             tivity>,
        rdfs:comment "The purpose of this task is to make sure that the team
             understands the requirements of the system correctly. This task also
             allows the team members to comment on the Acceptance Tests to improve
             their quality."@en,
        rdfs:label "Acceptance Test Review Task"

    SubClassOf:
```

```
        acao:isPerformedIn only acao:PlanningDayActivity,
        acao:Task,
        acao:isPerformedIn some acao:PlanningDayActivity


Class: mcao:DevelopmentEnvironment

    Annotations:
        rdfs:label "Development Environment",
        rdfs:comment "Each platform requests specific andd native development
             environment for best results."@en

    SubClassOf:
        acao:hasArtifactOrigin only acao:OtherArtifact,
        not (acao:isUsedByTask some acao:Task),
        not (acao:isUpdatedByTask some acao:Task),
        acao:isCreatedByTask some acao:EnvironmentSetUpTask,
        mcao:hasReusabilityLevel some mcao:None,
        acao:hasArtifactType only acao:Software,
        acao:hasArtifactType some acao:Software,
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:Artifact,
        mcao:hasReusabilityLevel only mcao:None,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        acao:isCreatedByTask only acao:EnvironmentSetUpTask,
        acao:hasArtifactOrigin some acao:OtherArtifact


Class: acao:ServiceArtifact

    Annotations:
        rdfs:comment "Defines class of artifacts that are created in relation to
             Web Service development."@en,
        rdfs:label "Service Artifact"

    SubClassOf:
        acao:ArtifactOrigin


Class: acao:SADDDocument

    Annotations:
        rdfs:comment "Contains the technical documentation on the developed
             product."@en,
        rdfs:label "Software Architecture And Design Description Document"

    SubClassOf:
        acao:isUpdatedByTask some acao:WrapUpTask,
        acao:isUpdatedByTask only
            (acao:DocumentationWrapUpTask
             or acao:WrapUpTask),
        acao:isUsedByTask some acao:WrapUpTask,
        mcao:hasReusabilityLevel some mcao:None,
        acao:hasArtifactType some acao:Document,
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:isUpdatedByTask some acao:DocumentationWrapUpTask,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        mcao:hasReusabilityLevel only mcao:None,
```

```
        acao:Artifact,
        acao:isUsedByTask some acao:DocumentationWrapUpTask,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        acao:hasArtifactType only acao:Document,
        acao:isUsedByTask only
            (acao:DocumentationWrapUpTask
             or acao:WrapUpTask),
        acao:isCreatedByTask some acao:ArchitectureLinePlanningTask,
        acao:isCreatedByTask only acao:ArchitectureLinePlanningTask


Class: acao:MeasurementPlan

    Annotations:
        rdfs:label "Measurement Plan",
        rdfs:comment "Includes the metrics and plan for monitoring of the project.
             In our case we recorded only the duration of activities and compared
             them with plan. This document is part of project plan."@en

    SubClassOf:
        acao:isUpdatedByTask some acao:WrapUpTask,
        acao:isUsedByTask some acao:PostIterationWorkshopTask,
        acao:isUsedByTask only
            (acao:IterationPlanningTask
             or acao:PostIterationWorkshopTask),
        acao:isPartOfArtifact only acao:ProjectPlan,
        mcao:hasReusabilityLevel only mcao:Partially,
        acao:isUsedByTask some acao:IterationPlanningTask,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:isPartOfArtifact some acao:ProjectPlan,
        acao:hasArtifactType only acao:DocumentElement,
        acao:hasArtifactType some acao:DocumentElement,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:Artifact,
        mcao:isSimilarToArtifact only acao:MeasurementPlan,
        acao:isCreatedByTask some acao:ProcessEstablishmentTask,
        acao:isCreatedByTask only acao:ProcessEstablishmentTask,
        acao:isUpdatedByTask only acao:WrapUpTask,
        mcao:isSimilarToArtifact some acao:MeasurementPlan


Class: acao:InitializeActivities

    EquivalentTo:
        acao:isPerformedIn some acao:Initialize

    SubClassOf:
        acao:ActivitiesByPhases


Class: acao:AcceptanceTestTemplateSheet

    Annotations:
        rdfs:label "Acceptance Test Template Sheet",
        rdfs:comment "Mobile-D acceptance test template sheet "@en

    SubClassOf:
        acao:isPartOfArtifact some acao:AcceptanceTest,
```

```
        mcao:hasReusabilityLevel some mcao:Completely,
        mcao:hasReusabilityLevel only mcao:Completely,
        acao:isUsedByTask some acao:InitialRequirementsAnalysisTask,
        not (acao:isUpdatedByTask some acao:Task),
        acao:hasArtifactType some acao:Template,
        acao:isUsedByTask some acao:AcceptanceTestGenerationTask,
        acao:isUsedByTask only
            (acao:AcceptanceTestGenerationTask
             or acao:InitialRequirementsAnalysisTask),
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:isPartOfArtifact some acao:MobileDProcessLibrary,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:Artifact,
        acao:isPartOfArtifact only
            (acao:AcceptanceTest
             or acao:MobileDProcessLibrary),
        acao:hasArtifactType only acao:Template,
        mcao:isSimilarToArtifact some acao:AcceptanceTestTemplateSheet,
        mcao:isSimilarToArtifact only acao:AcceptanceTestTemplateSheet,
        not (acao:isCreatedByTask some acao:Task)


Class: acao:WrapUpTask

    Annotations:
        rdfs:comment "The purpose of Wrap-up is to improve communication within the
             team and to measure the progress of the iteration. Each working day
             starts with a Wrap-up meeting, where the tasks to be implemented are
             decided and discussed. Another Wrap-up meeting is held at the end of
             day to reviewthe progress of teams and tasks."@en,
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#WorkingDayAct
             ivity>,
        rdfs:label "Wrap-up Task",
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#WorkingDayIn0
             IterationActivity>

    SubClassOf:
        acao:isPerformedIn some acao:WorkingDayIn0IterationActivity,
        acao:Task,
        acao:isPerformedIn some acao:WorkingDayActivity,
        acao:isPerformedIn only
            (acao:WorkingDayActivity
             or acao:WorkingDayIn0IterationActivity)


Class: wpcao:MicrosoftPhoneControlsToolkit

    Annotations:
        rdfs:label "Microsoft Phone Controls Toolkit",
        rdfs:comment "Library containing the classes necessary for adding some
             basic and advanced controls in Windows Phone application."@en

    SubClassOf:
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        mcao:AppReference,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact
```

```
Class: wpcao:ApplicationIcon

    Annotations:
        rdfs:comment "Application icon is designed as needed for publishing
             process."@en,
        rdfs:label "Application Icon WP"

    SubClassOf:
        mcao:AppIcon,
        acao:isPartOfArtifact some wpcao:DeploymentPackage,
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact,
        acao:isPartOfArtifact only wpcao:DeploymentPackage


Class: mcao:UnitTest

    Annotations:
        rdfs:comment "Unit test tests a single unit of platform specific code. It
             is created in separate project and references main project while
             performing different assertions."@en,
        rdfs:label "Unit Test"

    SubClassOf:
        acao:hasArtifactType only acao:Code,
        acao:hasArtifactType some acao:Code,
        mcao:isSimilarToArtifact some mcao:UnitTest,
        acao:isCreatedByTask some acao:TestDrivenDevelopmentPractice,
        acao:isUsedByTask some acao:SystemTestTask,
        acao:isUsedByTask some acao:RefactoringPractice,
        mcao:hasReusabilityLevel only mcao:Partially,
        acao:isUsedByTask only
            (acao:PreReleaseTestingTask
             or acao:RefactoringPractice
             or acao:SystemTestTask),
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:isUpdatedByTask some acao:PairProgrammingPractice,
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:Artifact,
        mcao:isSimilarToArtifact only mcao:UnitTest,
        acao:isCreatedByTask only acao:TestDrivenDevelopmentPractice,
        acao:isUsedByTask some acao:PreReleaseTestingTask,
        acao:isUpdatedByTask only acao:PairProgrammingPractice


Class: acao:AndroidArtifacts

    EquivalentTo:
        acao:Artifact
         and (acao:hasArtifactOrigin some acao:AndroidArtifact)

    SubClassOf:
        mcao:ArtifactsOrigin


Class: mcao:ViewController

    Annotations:
```

```
        rdfs:label "View Controller",
        rdfs:comment  "Represents  platform  specific  class  that  controlls  the
             application view."@en

    SubClassOf:
        acao:hasArtifactType only acao:Code,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:DocumentationWrapUpTask
             or acao:PairProgrammingPractice
             or acao:PreReleaseTestingTask
             or acao:RefactoringPractice
             or acao:SystemIntegrationTask
             or acao:SystemTestTask
             or acao:TestDrivenDevelopmentPractice),
        acao:isUsedByTask some acao:SystemIntegrationTask,
        acao:hasArtifactType some acao:Code,
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:isUpdatedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:PreReleaseTestingTask
             or acao:RefactoringPractice
             or acao:SystemIntegrationTask),
        acao:isUpdatedByTask some acao:SystemIntegrationTask,
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:isUpdatedByTask some acao:ContinuousIntegrationPractice,
        acao:isUsedByTask some acao:SystemTestTask,
        acao:isUsedByTask some acao:RefactoringPractice,
        mcao:hasReusabilityLevel only mcao:Partially,
        acao:isCreatedByTask some acao:PairProgrammingPractice,
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:Artifact,
        acao:isUsedByTask some acao:DocumentationWrapUpTask,
        mcao:isSimilarToArtifact only mcao:ViewController,
        acao:isCreatedByTask only acao:PairProgrammingPractice,
        acao:isUpdatedByTask some acao:RefactoringPractice,
        acao:isUsedByTask some acao:PreReleaseTestingTask,
        acao:isUsedByTask some acao:ContinuousIntegrationPractice,
        mcao:isSimilarToArtifact some mcao:ViewController,
        acao:isUpdatedByTask some acao:PreReleaseTestingTask


Class: mcao:ReuseLevel

    EquivalentTo:
        mcao:Completely
         or mcao:None
         or mcao:Partially


Class: wpcao:SilverlightMapControl

    Annotations:
        rdfs:comment "Library containing the classes necessary if using Bing Maps
             in WP application."@en,
        rdfs:label "Silverlight Map Control"

    SubClassOf:
        mcao:MapsSDK,
```

```
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact


Class: acao:Layout

    Annotations:
        rdfs:comment "Represents XML code that is used to describe user interface
             form or screen."@en,
        rdfs:label "Layout"

    SubClassOf:
        mcao:View,
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        acao:isPartOfArtifact some acao:XMLResources,
        acao:hasArtifactOrigin some acao:AndroidArtifact,
        acao:isPartOfArtifact only acao:XMLResources


Class: acao:DevelopmentUnrelatedSoftwareTool

    Annotations:
        rdfs:comment "These software tools support the main operations performed by
             project team. For example these include office suit, pdf reader, image
             editor etc."@en,
        rdfs:label "Development Unrelated Software Tool"

    SubClassOf:
        acao:hasArtifactOrigin only acao:OtherArtifact,
        mcao:hasReusabilityLevel some mcao:Completely,
        not (acao:isUsedByTask some acao:Task),
        mcao:hasReusabilityLevel only mcao:Completely,
        mcao:isSimilarToArtifact only acao:DevelopmentUnrelatedSoftwareTool,
        not (acao:isUpdatedByTask some acao:Task),
        acao:hasArtifactType only acao:Software,
        mcao:isSimilarToArtifact some acao:DevelopmentUnrelatedSoftwareTool,
        acao:hasArtifactType some acao:Software,
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:Artifact,
        acao:isCreatedByTask some acao:ProcessEstablishmentTask,
        acao:isCreatedByTask only acao:ProcessEstablishmentTask,
        acao:hasArtifactOrigin some acao:OtherArtifact


Class: wpcao:DevelopmentEnvironment

    Annotations:
        rdfs:comment "Set of applications used for Windows Phone development and
             integrated in Visual Studio."@en,
        rdfs:label "Development Environment WP"

    SubClassOf:
        mcao:DevelopmentEnvironment


Class: acao:Activity

    Annotations:
```

```
        rdfs:comment "In Mobile-D, activities are sometimes called stages. The
             activity represents set of tasks that should be done in order to
             achieve the goals that are defined by that activity/stage."


Class: acao:StakeholderEstablishmentTasks

    EquivalentTo:
        acao:isPerformedIn some acao:StakeholderEstablishmentActivity

    SubClassOf:
        acao:TasksByActivities


Class: mcao:AppPrototypeFunctionality

    Annotations:
        rdfs:label "App Prototype Functionality",
        rdfs:comment "Developed platform specific functionality during the trial
             day. It prototypes some of the main application functionalities and is
             used to define the basic approach for implementing the similar
             functionalities in other iterations."@en

    SubClassOf:
        acao:isUpdatedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:RefactoringPractice),
        acao:isUsedByTask some acao:PostIterationWorkshopTask,
        acao:hasArtifactType only acao:Code,
        acao:isUsedByTask some acao:RequirementsAnalysisTask,
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:hasArtifactType some acao:Code,
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:isUpdatedByTask some acao:ContinuousIntegrationPractice,
        mcao:hasReusabilityLevel some mcao:None,
        acao:isUsedByTask some acao:RefactoringPractice,
        acao:isPartOfArtifact some acao:MobileApplication,
        acao:isCreatedByTask some acao:PairProgrammingPractice,
        mcao:hasReusabilityLevel only mcao:None,
        acao:Artifact,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        acao:isCreatedByTask only acao:PairProgrammingPractice,
        acao:isUpdatedByTask some acao:RefactoringPractice,
        acao:isPartOfArtifact only acao:MobileApplication,
        acao:isUsedByTask some acao:ContinuousIntegrationPractice,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:PairProgrammingPractice
             or acao:PostIterationWorkshopTask
             or acao:RefactoringPractice
             or acao:RequirementsAnalysisTask
             or acao:TestDrivenDevelopmentPractice)


Class: acao:SystemTestActivity

    Annotations:
        rdfs:comment "The purpose of System Test & Fix is to see if the produced
             system implements the customer defined functionality correctly,
```

```
            provide the project team feedback on the systems functionality and fix
            the found defects."@en,
        acao:inPhase
            <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#SystemTestAnd
            Fix>,
        rdfs:label "System Test & Fix"

    SubClassOf:
        acao:consistsOf only acao:SystemTestTask,
        acao:isPerformedIn some acao:SystemTestAndFix,
        acao:consistsOf some acao:SystemTestTask,
        acao:Activity,
        acao:isPerformedIn only acao:SystemTestAndFix


Class: acao:ArchitectureLinePlanningTask

    Annotations:
        rdfs:label "Architecture Line Planning Task",
        rdfs:comment "The purpose of this task is to prepare all
            criticalarchitectural issues so that so that they all are in full
            readiness for a systematicarchitectural growth when implementing
            requirements selected by the customerduring forthcoming project
            phases."@en,
        acao:inActivity
            <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#PlanningDayIn
            0IterationActivity>

    SubClassOf:
        acao:isPerformedIn only acao:PlanningDayIn0IterationActivity,
        acao:Task,
        acao:isPerformedIn some acao:PlanningDayIn0IterationActivity


Class: acao:GooglePlayServices

    Annotations:
        rdfs:comment "Google library containing the classes necessary if using
            Google Maps."@en,
        rdfs:label "Google Play Services"

    SubClassOf:
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        mcao:MapsSDK,
        acao:hasArtifactOrigin some acao:AndroidArtifact


Class: acao:Driver

    Annotations:
        rdfs:label "Driver Android",
        rdfs:comment "Set of drivers used to install the device connectivity for
            testing purposes."@en

    SubClassOf:
        mcao:TestDeviceDriver


Class: wpcao:BingMapsKey
```

```
    Annotations:
        rdfs:label "Bing Maps Key",
        rdfs:comment "Microsoft license identifying the developer as unique person.
             This key is application specific and is used when using Silverlight
             Map Control."@en

    SubClassOf:
        mcao:MapsKey,
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact


Class: acao:MethodologicalArtifact

    Annotations:
        rdfs:label "Methodological Artifact",
        rdfs:comment "Defines class of artifacts that are created in relation to
             Mobile-D implementation."@en

    SubClassOf:
        acao:ArtifactOrigin


Class: acao:RequirementsAnalysisTask

    Annotations:
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#PlanningDayAc
             tivity>,
        rdfs:label "Requirements Analysis Task",
        rdfs:comment "The purpose of this task is to carefully prioritize and
             analyze the requirements selected for each iteration."@en

    SubClassOf:
        acao:isPerformedIn only acao:PlanningDayActivity,
        acao:Task,
        acao:isPerformedIn some acao:PlanningDayActivity


Class: mcao:None

    SubClassOf:
        mcao:ReuseLevel


Class: acao:ProjectSetupTasks

    EquivalentTo:
        acao:isPerformedIn some acao:ProjectSetUpActivity

    SubClassOf:
        acao:TasksByActivities


Class: mcao:CompletlyResuableArtifacts

    EquivalentTo:
        acao:Artifact
```

```
         and (mcao:hasReusabilityLevel some mcao:Completely)

    SubClassOf:
        mcao:ArtifactsReusability


Class: acao:Stabilize

    Annotations:
        rdfs:label "Stabilize Phase",
        rdfs:comment "The Productionize and Stabilize phases are executed
             iteratively in order to develop all other features of the mobile
             product. The Stabilize phase has the goal to finalize the
             implementation, including integrating subsystems if needed. As this
             phase can contain additional programing and development, the
             activities are very similar to activities in productionize phase. Only
             additional activity concerns documentation wrap-up. Iterations should
             result in working piece of functionality at the user level."@en

    SubClassOf:
        acao:consistsOf some acao:ReleaseDayActivity,
        acao:consistsOf only
            (acao:DocumentationWrapUpActivity
             or acao:PlanningDayActivity
             or acao:ReleaseDayActivity
             or acao:WorkingDayActivity),
        acao:consistsOf some acao:DocumentationWrapUpActivity,
        acao:consistsOf some acao:PlanningDayActivity,
        acao:Phase,
        acao:consistsOf some acao:WorkingDayActivity


Class: acao:ArchitectureLinePlan

    Annotations:
        rdfs:comment "Contains the information on planned system architecture.
             Created after the prototyping is finished. This document is part of
             SADD document."@en,
        rdfs:label "Architecture Line Plan"

    SubClassOf:
        not (acao:isUsedByTask some acao:Task),
        acao:isPartOfArtifact some acao:SADDDocument,
        mcao:hasReusabilityLevel only mcao:Partially,
        acao:isPartOfArtifact only acao:SADDDocument,
        acao:isUpdatedByTask some acao:InitialRequirementsAnalysisTask,
        mcao:isSimilarToArtifact some acao:ArchitectureLinePlan,
        acao:isUpdatedByTask only acao:InitialRequirementsAnalysisTask,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        mcao:isSimilarToArtifact only acao:ArchitectureLinePlan,
        acao:hasArtifactType only acao:DocumentElement,
        acao:hasArtifactType some acao:DocumentElement,
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:Artifact,
        acao:isCreatedByTask some acao:ArchitectureLinePlanningTask,
        acao:isCreatedByTask only acao:ArchitectureLinePlanningTask
```

```
Class: acao:OtherArtifact

    Annotations:
        rdfs:comment "Defines class of artifacts that are not related to Android
             development or Mobile-D implementation or Web Service
             development."@en,
        rdfs:comment "Defines class of artifacts that are not related to Windows
             Phone development or Mobile-D implementation or Web Service
             development."@en,
        rdfs:label "Other Artifact"

    SubClassOf:
        acao:ArtifactOrigin


Class: acao:JavaCode

    Annotations:
        rdfs:label "Java Code",
        rdfs:comment "Java code developed during the implementation activities."@en

    SubClassOf:
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        mcao:SourceCode,
        acao:isPartOfArtifact only acao:MobileApplication,
        acao:hasArtifactOrigin some acao:AndroidArtifact,
        acao:isPartOfArtifact some acao:MobileApplication


Class: acao:Code

    Annotations:
        rdfs:comment "Represents any artifact created during the implementation and
             is written in any programming or description language."@en

    SubClassOf:
        acao:ArtifactType


Class: wpcao:DeploymentPackage

    Annotations:
        rdfs:label "Deployment Package WP",
        rdfs:comment "XAP file created for publishing purposes."@en

    SubClassOf:
        mcao:DeploymentPackage,
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact


Class: acao:ExampleCode

    Annotations:
        rdfs:comment "Android example code on different topics found on the
             internet from various sources."@en,
        rdfs:label "Example Code Android"

    SubClassOf:
```

```
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        mcao:ExampleCode,
        acao:hasArtifactOrigin some acao:AndroidArtifact


Class: mcao:AppDescription

    Annotations:
        rdfs:label "App Description",
        rdfs:comment "Short but important description used for publishing process.
             It includes the information on application, category, authors etc. Due
             to different app store requirements, there might be some differences
             among platforms."@en,
        mcao:NOTICE "Closure axiom for isPartOfArtifact is used in leaf
             elements."@en

    SubClassOf:
        acao:isCreatedByTask only acao:DocumentationWrapUpTask,
        acao:isPartOfArtifact some acao:SADDDocument,
        not (acao:isUpdatedByTask some acao:Task),
        acao:hasArtifactType some acao:Resource,
        mcao:hasReusabilityLevel only mcao:Partially,
        acao:isCreatedByTask some acao:DocumentationWrapUpTask,
        acao:hasArtifactType only acao:Resource,
        mcao:isSimilarToArtifact only mcao:AppDescription,
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:Artifact,
        acao:isUsedByTask only acao:PublishApplicationTask,
        mcao:isSimilarToArtifact some mcao:AppDescription,
        acao:isUsedByTask some acao:PublishApplicationTask


Class: wpcao:UnitTest

    Annotations:
        rdfs:label "Unit Test WP",
        rdfs:comment "Unit test tests a single unit of code. It is created in
             separate project and references main project while performing
             different assertions."@en

    SubClassOf:
        mcao:UnitTest,
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact


Class: acao:GoogleAPIKey

    Annotations:
        rdfs:comment "Google license identifying the developer as unique person.
             This key is application specific and is used when using Google Maps
             API."@en,
        rdfs:label "Google API Key"

    SubClassOf:
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        mcao:MapsKey,
        acao:hasArtifactOrigin some acao:AndroidArtifact
```

```
Class: mcao:AppReference

    Annotations:
        rdfs:comment "Referenced platform specific libraries providing additional
             development functionality."@en,
        rdfs:label "App Reference"

    SubClassOf:
        acao:hasArtifactType only acao:Code,
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:hasArtifactType some acao:Code,
        not (acao:isUpdatedByTask some acao:Task),
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:EnvironmentSetUpTask
             or acao:PairProgrammingPractice
             or acao:PublishApplicationTask
             or acao:RefactoringPractice
             or acao:TestDrivenDevelopmentPractice),
        mcao:hasReusabilityLevel some mcao:None,
        acao:isUsedByTask some acao:RefactoringPractice,
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:Artifact,
        mcao:hasReusabilityLevel only mcao:None,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        acao:isUsedByTask some acao:PublishApplicationTask,
        acao:isUsedByTask some acao:EnvironmentSetUpTask,
        acao:isUsedByTask some acao:ContinuousIntegrationPractice,
        not (acao:isCreatedByTask some acao:Task)


Class: acao:ProjectPlanGanttChart

    Annotations:
        rdfs:label "Project Plan Gantt Chart",
        rdfs:comment "Model containing the graphical information on project plan
             iterations, activities and their duration. It is used in Project plan
             document."@en

    SubClassOf:
        mcao:isSimilarToArtifact some acao:ProjectPlanGanttChart,
        mcao:hasReusabilityLevel some mcao:Completely,
        acao:hasArtifactType some acao:Model,
        mcao:hasReusabilityLevel only mcao:Completely,
        acao:isUpdatedByTask only
            (acao:InitialRequirementsAnalysisTask
             or acao:PostIterationWorkshopTask
             or acao:ProcessEstablishmentTask),
        mcao:isSimilarToArtifact only acao:ProjectPlanGanttChart,
        acao:isUsedByTask some acao:ArchitectureLineDefinitionTask,
        acao:isCreatedByTask some acao:InitialProjectPlanningTask,
        acao:hasArtifactType only acao:Model,
        acao:isUpdatedByTask some acao:ProcessEstablishmentTask,
        acao:isPartOfArtifact only acao:ProjectPlan,
        acao:isUsedByTask some acao:ArchitectureLinePlanningTask,
        acao:isUsedByTask some acao:IterationPlanningTask,
        acao:isUpdatedByTask some acao:InitialRequirementsAnalysisTask,
```

```
        acao:isPartOfArtifact some acao:ProjectPlan,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:isUsedByTask only
            (acao:ArchitectureLineDefinitionTask
             or acao:ArchitectureLinePlanningTask
             or acao:IterationPlanningTask
             or acao:ProcessEstablishmentTask),
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:Artifact,
        acao:isUsedByTask some acao:ProcessEstablishmentTask,
        acao:isCreatedByTask only acao:InitialProjectPlanningTask,
        acao:isUpdatedByTask some acao:PostIterationWorkshopTask


Class: wpcaoDriver

    Annotations:
        rdfs:label "Driver WP",
        rdfs:comment "Set of drivers used to install the device connectivity for
             testing purposes."@en

    SubClassOf:
        mcao:TestDeviceDriver


Class: mcao:TestDeviceDriver

    Annotations:
        rdfs:label "Test Device Driver",
        rdfs:comment "Driver used to install the specific device connectivity for
             testing purposes."@en

    SubClassOf:
        acao:hasArtifactOrigin only acao:OtherArtifact,
        not (acao:isUsedByTask some acao:Task),
        not (acao:isUpdatedByTask some acao:Task),
        mcao:hasReusabilityLevel some mcao:None,
        acao:isCreatedByTask some acao:EnvironmentSetUpTask,
        acao:hasArtifactType only acao:Software,
        acao:hasArtifactType some acao:Software,
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:Artifact,
        mcao:hasReusabilityLevel only mcao:None,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        acao:isCreatedByTask only acao:EnvironmentSetUpTask,
        acao:hasArtifactOrigin some acao:OtherArtifact


Class: acao:ProjectPlan

    Annotations:
        rdfs:label "Project Plan",
        rdfs:comment "Contains all information on project including definition of
             customer group, scope, planned activities and their duration, plans on
             documentation etc. Aligned with agile practices, this document is also
             updated during the iterations."@en

    SubClassOf:
        acao:isUsedByTask some acao:ArchitectureLineDefinitionTask,
```

```
        acao:isCreatedByTask some acao:InitialProjectPlanningTask,
        acao:isUpdatedByTask some acao:ProcessEstablishmentTask,
        acao:isUpdatedByTask only
            (acao:CustomerEstablishmentTask
             or acao:InitialRequirementsAnalysisTask
             or acao:PostIterationWorkshopTask
             or acao:ProcessEstablishmentTask),
        mcao:hasReusabilityLevel only mcao:Partially,
        acao:hasArtifactType some acao:Document,
        acao:isUsedByTask some acao:ArchitectureLinePlanningTask,
        acao:isUsedByTask some acao:IterationPlanningTask,
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:isUpdatedByTask some acao:InitialRequirementsAnalysisTask,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:isUsedByTask only
            (acao:ArchitectureLineDefinitionTask
             or acao:ArchitectureLinePlanningTask
             or acao:IterationPlanningTask
             or acao:ProcessEstablishmentTask),
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:Artifact,
        acao:hasArtifactType only acao:Document,
        mcao:isSimilarToArtifact only acao:ProjectPlan,
        mcao:isSimilarToArtifact some acao:ProjectPlan,
        acao:isUsedByTask some acao:ProcessEstablishmentTask,
        acao:isCreatedByTask only acao:InitialProjectPlanningTask,
        acao:isUpdatedByTask some acao:CustomerEstablishmentTask,
        acao:isUpdatedByTask some acao:PostIterationWorkshopTask


Class: acao:UIIllustrations

    Annotations:
        rdfs:label "UI Ilustrations",
        rdfs:comment "Describes the illustrations of mobile application user
             interface. It is part of SADD document."@en

    SubClassOf:
        acao:isUpdatedByTask some acao:RequirementsAnalysisTask,
        acao:isUsedByTask some acao:AcceptanceTestingTask,
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:isUpdatedByTask only acao:RequirementsAnalysisTask,
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:isPartOfArtifact some acao:SADDDocument,
        acao:isUsedByTask some acao:AcceptanceTestGenerationTask,
        mcao:hasReusabilityLevel some mcao:None,
        acao:isUsedByTask some acao:SystemTestTask,
        acao:isUsedByTask some acao:RefactoringPractice,
        acao:isPartOfArtifact only acao:SADDDocument,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:hasArtifactType only acao:DocumentElement,
        acao:hasArtifactType some acao:DocumentElement,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        mcao:hasReusabilityLevel only mcao:None,
        acao:Artifact,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        acao:isUsedByTask only
            (acao:AcceptanceTestGenerationTask
```

```
            or acao:AcceptanceTestingTask
            or acao:PairProgrammingPractice
            or acao:RefactoringPractice
            or acao:SystemTestTask
            or acao:TestDrivenDevelopmentPractice),
        acao:isCreatedByTask only acao:InitialRequirementsAnalysisTask,
        acao:isCreatedByTask some acao:InitialRequirementsAnalysisTask


Class: wpcao:WindowsPhoneArtifact

    Annotations:
        rdfs:label "Windows Phone Artifact",
        rdfs:comment "Defines class of artifacts that are created in relation to
             Windows Phone development."@en

    SubClassOf:
        acao:ArtifactOrigin


Class: acao:WebDevelopmentEnvironment

    Annotations:
        rdfs:comment "The web application development and hosting environment had
             to be set up."@en,
        rdfs:label "Web Development Environment"

    SubClassOf:
        acao:hasArtifactOrigin only acao:OtherArtifact,
        not (acao:isUsedByTask some acao:Task),
        mcao:hasReusabilityLevel some mcao:Completely,
        mcao:hasReusabilityLevel only mcao:Completely,
        not (acao:isUpdatedByTask some acao:Task),
        acao:isCreatedByTask some acao:EnvironmentSetUpTask,
        acao:hasArtifactType only acao:Software,
        mcao:isSimilarToArtifact only acao:WebDevelopmentEnvironment,
        acao:hasArtifactType some acao:Software,
        not (acao:isPartOfArtifact some acao:Artifact),
        mcao:isSimilarToArtifact some acao:WebDevelopmentEnvironment,
        acao:Artifact,
        acao:isCreatedByTask only acao:EnvironmentSetUpTask,
        acao:hasArtifactOrigin some acao:OtherArtifact


Class: acao:StoryCard

    Annotations:
        rdfs:label "Story Card",
        rdfs:comment "Basic documentation card containing information on one
             feature that is implemented. It is defined during the planning day but
             is refined during the implementation and wrap-up. It is part of the
             Product backlog document."@en

    SubClassOf:
        acao:isUpdatedByTask some acao:WrapUpTask,
        acao:isCreatedByTask only acao:IterationPlanningTask,
        mcao:isSimilarToArtifact only acao:StoryCard,
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:isPartOfArtifact only acao:ProductBacklog,
```

```
        acao:isPartOfArtifact some acao:ProductBacklog,
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:isUsedByTask some acao:WrapUpTask,
        acao:isUsedByTask some acao:AcceptanceTestGenerationTask,
        mcao:isSimilarToArtifact some acao:StoryCard,
        mcao:hasReusabilityLevel only mcao:Partially,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:hasArtifactType some acao:DocumentElement,
        acao:hasArtifactType only acao:DocumentElement,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:Artifact,
        acao:isUsedByTask only
            (acao:AcceptanceTestGenerationTask
             or acao:PairProgrammingPractice
             or acao:TestDrivenDevelopmentPractice
             or acao:WrapUpTask),
        acao:isUpdatedByTask only acao:WrapUpTask,
        acao:isCreatedByTask some acao:IterationPlanningTask


Class: wpcao:MobileApplication

    Annotations:
        rdfs:comment "The mobile application created in the development
             process."@en,
        rdfs:label "Mobile Application WP"

    SubClassOf:
        mcao:MobileApplication,
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact


Class: acao:InitialRequirementsDocument

    Annotations:
        rdfs:comment "Created according to product proposal, but later updated with
             information on stakeholders and functional system requirements. It is
             also updated during the planning phase in 0-iteration and subsequent
             iterations."@en,
        rdfs:label "Initial Requirements Document"

    SubClassOf:
        mcao:hasReusabilityLevel only mcao:Completely,
        acao:isUsedByTask some acao:ArchitectureLineDefinitionTask,
        acao:isUsedByTask some acao:InitialRequirementsAnalysisTask,
        mcao:isSimilarToArtifact some acao:InitialRequirementsDocument,
        acao:isUsedByTask some acao:InitialProjectPlanningTask,
        acao:isUsedByTask some acao:AcceptanceTestGenerationTask,
        acao:isUsedByTask some acao:SystemTestTask,
        acao:isUsedByTask some acao:ArchitectureLinePlanningTask,
        acao:hasArtifactType some acao:Document,
        acao:isUpdatedByTask some acao:InitialRequirementsAnalysisTask,
        acao:Artifact,
        acao:hasArtifactType only acao:Document,
        acao:isCreatedByTask some acao:InitialRequirementsCollectionTask,
        mcao:hasReusabilityLevel some mcao:Completely,
        acao:isUsedByTask some acao:RequirementsAnalysisTask,
```

```
        acao:isUpdatedByTask some acao:AcceptanceTestingTask,
        mcao:isSimilarToArtifact only acao:InitialRequirementsDocument,
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:isUsedByTask only
            (acao:AcceptanceTestGenerationTask
             or acao:AcceptanceTestReviewTask
             or acao:ArchitectureLineDefinitionTask
             or acao:ArchitectureLinePlanningTask
             or acao:DocumentationWrapUpTasks
             or acao:InitialProjectPlanningTask
             or acao:InitialRequirementsAnalysisTask
             or acao:RequirementsAnalysisTask
             or acao:SystemTestTask),
        acao:isCreatedByTask only acao:InitialRequirementsCollectionTask,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:isUpdatedByTask only
            (acao:AcceptanceTestingTask
             or acao:InitialRequirementsAnalysisTask),
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:isUsedByTask some acao:DocumentationWrapUpTasks,
        acao:isUsedByTask some acao:AcceptanceTestReviewTask


Class: acao:AcceptanceTestGenerationTask

    Annotations:
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#PlanningDayAc
             tivity>,
        rdfs:comment "The purpose of this task is to support the verification of
             the requirements the customer has set for the software. This task also
             acts as a communication tool between the customer and the development
             team. "@en,
        rdfs:label "Acceptance Test Review Task"

    SubClassOf:
        acao:isPerformedIn only acao:PlanningDayActivity,
        acao:Task,
        acao:isPerformedIn some acao:PlanningDayActivity


Class: acao:Template

    Annotations:
        rdfs:comment "Represents templates that are used to create some artifacts.
             "@en

    SubClassOf:
        acao:ArtifactType


Class: wpcao:ResourceFile

    Annotations:
        rdfs:label "Resource File WP",
        rdfs:comment "Represent code that is used to provide application with
             resources (strings, images, icons, audio, files and other). We used it
             to provide the application with localized translation for two
             languages."@en
```

```
    SubClassOf:
        mcao:isSimilarToArtifact some acao:LocalizationString,
        acao:isPartOfArtifact some wpcao:CSCode,
        acao:isPartOfArtifact only wpcao:CSCode,
        mcao:AppResource,
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact,
        mcao:isSimilarToArtifact only acao:LocalizationString


Class: mcao:APIDocumentation

    Annotations:
        rdfs:label "API Documentation",
        rdfs:comment "API Documentation is platform specific set of materials and
             code examples that could be used by developers."@en

    SubClassOf:
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        not (acao:isUpdatedByTask some acao:Task),
        acao:isUsedByTask some acao:PairProgrammingPractice,
        mcao:hasReusabilityLevel some mcao:None,
        acao:isUsedByTask some acao:RefactoringPractice,
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:hasArtifactType only acao:Example,
        acao:hasArtifactType some acao:Example,
        mcao:hasReusabilityLevel only mcao:None,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:PairProgrammingPractice
             or acao:RefactoringPractice
             or acao:TestDrivenDevelopmentPractice),
        acao:Artifact,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        acao:isUsedByTask some acao:ContinuousIntegrationPractice,
        not (acao:isCreatedByTask some acao:Task)


Class: mcao:DeploymentPackage

    Annotations:
        rdfs:comment "Packege containing all files (including the application
             itself) necessary for publishing purposes. The artifact is platform
             specific."@en,
        rdfs:label "Deployment Package"

    SubClassOf:
        acao:isCreatedByTask some acao:PublishApplicationTask,
        not (acao:isUsedByTask some acao:Task),
        acao:hasArtifactType only acao:Resource,
        acao:isCreatedByTask only acao:PublishApplicationTask,
        acao:Artifact,
        mcao:hasReusabilityLevel only mcao:None,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        not (acao:isUpdatedByTask some acao:Task),
        acao:hasArtifactType some acao:Resource,
        mcao:hasReusabilityLevel some mcao:None,
        acao:isPartOfArtifact only acao:MobileApplication,
```

```
        acao:isPartOfArtifact some acao:MobileApplication


Class: acao:AndroidActivity

    Annotations:
        rdfs:label "Android Activity",
        rdfs:comment "Represents java class that inherits Android Activity class
             with the purpose of controlling the application view."@en

    SubClassOf:
        mcao:ViewController,
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        acao:isPartOfArtifact some acao:JavaCode,
        acao:hasArtifactOrigin some acao:AndroidArtifact,
        acao:isPartOfArtifact only acao:JavaCode


Class: acao:ContinuousIntegrationPractice

    Annotations:
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#WorkingDayAct
             ivity>,
        rdfs:comment "The purpose of Continuous integration is to continuously
             integrate new code with the existing code in a code repository. By
             integrating continuously massive integrations can be avoided that
             would otherwise take a lot of time and effort. Continuous integration
             is a practice which allows developers to achieve rapid feedback on
             progress of the whole development project. It helps to control the
             constant change of software."@en,
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#WorkingDayIn0
             IterationActivity>,
        rdfs:label "Continuous Integration Practice"

    SubClassOf:
        acao:isPerformedIn some acao:WorkingDayIn0IterationActivity,
        acao:Task,
        acao:isPerformedIn only
            (acao:WorkingDayActivity
             or acao:WorkingDayIn0IterationActivity),
        acao:isPerformedIn some acao:WorkingDayActivity


Class: acao:JSONStandard

    Annotations:
        rdfs:label "JSON Standard",
        rdfs:comment "IEEE Standard No. RFC4627 Standard defining the JSON
format."@en

    SubClassOf:
        acao:hasArtifactOrigin only acao:OtherArtifact,
        mcao:hasReusabilityLevel some mcao:Completely,
        mcao:hasReusabilityLevel only mcao:Completely,
        mcao:isSimilarToArtifact some acao:JSONStandard,
        mcao:isSimilarToArtifact only acao:JSONStandard,
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
```

```
        acao:isUsedByTask some acao:PairProgrammingPractice,
        not (acao:isUpdatedByTask some acao:Task),
        acao:isUsedByTask some acao:RefactoringPractice,
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:hasArtifactType only acao:Standard,
        acao:Artifact,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:PairProgrammingPractice
             or acao:RefactoringPractice
             or acao:TestDrivenDevelopmentPractice),
        acao:hasArtifactType some acao:Standard,
        acao:isUsedByTask some acao:ContinuousIntegrationPractice,
        not (acao:isCreatedByTask some acao:Task),
        acao:hasArtifactOrigin some acao:OtherArtifact


Class: wpcao:WMAppManifest

    Annotations:
        rdfs:label "WMAppManifest",
        rdfs:comment "XML document containing the information on application. It
             includes the information on some application resources. It is created
             automatically."@en

    SubClassOf:
        acao:isPartOfArtifact some wpcao:MobileApplication,
        acao:isPartOfArtifact some wpcao:DeploymentPackage,
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact,
        acao:isPartOfArtifact only
            (wpcao:DeploymentPackage
             or wpcao:MobileApplication),
        mcao:AppManifest


Class: acao:SystemTestAndFix

    Annotations:
        rdfs:label "System Test & Fix Phase",
        rdfs:comment "System Test and Fix phase aims to detect if the produced
             system implements the customer defined functionality correctly, it
             provides the project team feedback on the systems functionality and
             provides the defect information for last fixing iteration of the
             Mobile-D process. This last iteration is not obligatory, but when
             fixing is needed it consists of same activities as other
             implementation iterations already explained."@en

    SubClassOf:
        acao:consistsOf some acao:ReleaseDayActivity,
        acao:consistsOf some acao:SystemTestActivity,
        acao:consistsOf some acao:DocumentationWrapUpActivity,
        acao:consistsOf only
            (acao:DocumentationWrapUpActivity
             or acao:PlanningDayActivity
             or acao:ReleaseDayActivity
             or acao:SystemTestActivity
             or acao:WorkingDayActivity),
        acao:consistsOf some acao:PlanningDayActivity,
```

```
        acao:Phase,
        acao:consistsOf some acao:WorkingDayActivity


Class: acao:PlanningDayActivity

    Annotations:
        acao:inPhase
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#Productionize
             >,
        rdfs:label "Planning Day",
        acao:inPhase
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#SystemTestAnd
             Fix>,
        acao:inPhase
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#Stabilize>,
        rdfs:comment "The purpose in planning day is to select and plan the work
             contents for the iteration. By participating actively to planning
             activities, customer ensures that the requirements providing most
             business value is identified and those requirements are correctly
             understood."@en

    SubClassOf:
        acao:consistsOf some acao:PostIterationWorkshopTask,
        acao:isPerformedIn some acao:Productionize,
        acao:consistsOf some acao:IterationPlanningTask,
        acao:isPerformedIn some acao:SystemTestAndFix,
        acao:consistsOf some acao:AcceptanceTestGenerationTask,
        acao:isPerformedIn only
            (acao:Productionize
             or acao:Stabilize
             or acao:SystemTestAndFix),
        acao:isPerformedIn some acao:Stabilize,
        acao:consistsOf some acao:AcceptanceTestReviewTask,
        acao:Activity,
        acao:consistsOf only
            (acao:AcceptanceTestGenerationTask
             or acao:AcceptanceTestReviewTask
             or acao:IterationPlanningTask
             or acao:PostIterationWorkshopTask
             or acao:RequirementsAnalysisTask),
        acao:consistsOf some acao:RequirementsAnalysisTask


Class: acao:Phase

    Annotations:
        rdfs:comment "Mobile-D methodology comprises development process of five
             phases which are executed in combined sequential and incremental
             manner."@en


Class: acao:InitialRequirementsCollectionTask

    Annotations:
        rdfs:label "Initial Requirements Collection Task",
        rdfs:comment "The purpose of this task is to produce aninitial overall
             definition of the product’s scope, purpose and functionality. This is
             needed to enable the further planning and establishment of the project
```

```
            (size, technical issues, architecture, etc.). Also, the documented
            requirements will be the starting point for the project team to build
            an overall view on the product at hand. The customer and the steering
            group should agree and document the central functionality of the
            product as seen from the customer point of view. Additionally, also
            other requirements, such as organization's own business requirements,
            and constraints to the product development should be identified,
            agreed upon and documented."@en,
        acao:inActivity
            <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#ScopeDefiniti
            onActivity>

    SubClassOf:
        acao:Task,
        acao:isPerformedIn some acao:ScopeDefinitionActivity,
        acao:isPerformedIn only acao:ScopeDefinitionActivity


Class: acao:PlanningDayIn0IterationActivity

    Annotations:
        rdfs:label "Initial Planning (Planning Day in 0 Itteration)",
        acao:inPhase
            <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#Initialize>,
        rdfs:comment "The purpose of the Initial Planning pattern is to gain a good
            overall understanding of the product to be developed, to prepare and
            refine plans for forthcoming project phases and to prepare plans for
            verifying and solving all critical development issues by the end of
            the current phase."

    SubClassOf:
        acao:consistsOf some acao:IterationPlanningTask,
        acao:consistsOf some acao:ArchitectureLinePlanningTask,
        acao:consistsOf only
            (acao:ArchitectureLinePlanningTask
             or acao:InitialRequirementsAnalysisTask
             or acao:IterationPlanningTask),
        acao:isPerformedIn some acao:Initialize,
        acao:Activity,
        acao:consistsOf some acao:InitialRequirementsAnalysisTask,
        acao:isPerformedIn only acao:Initialize


Class: acao:StakeholderEstablishmentActivity

    Annotations:
        rdfs:comment "The purpose of this stage is to identify and establish the
            stakeholder groups that are needed in various tasks of Explore phase
            as well as in supporting activities during the software development -
            excluding the software development team itself. Wide variety of
            expertise and co-operation is neededto plan a controlled and effective
            implementation of the software product. The goals of the Stakeholder
            Establishment are to identify and establish different stakeholder
            groups needed in different taskthroughout the project."@en,
        acao:inPhase
            <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#Explore>,
        rdfs:label "Stakeholder Establishment"

    SubClassOf:
```

```
        acao:isPerformedIn some acao:Explore,
        acao:Activity,
        acao:consistsOf some acao:CustomerEstablishmentTask,
        acao:consistsOf only acao:CustomerEstablishmentTask,
        acao:isPerformedIn only acao:Explore


Class: acao:ArtifactType

    Annotations:
        rdfs:comment "Classification of artifacts in types according to their
             purpose."@en

    EquivalentTo:
        acao:Code
         or acao:Document
         or acao:DocumentElement
         or acao:Example
         or acao:Licence
         or acao:Model
         or acao:ModelElement
         or acao:Product
         or acao:Resource
         or acao:Software
         or acao:Standard
         or acao:Template


Class: acao:AndroidClass

    Annotations:
        rdfs:label "Android Class",
        rdfs:comment "UML model element used to describe an existing Android class
             that is to be used."@en

    SubClassOf:
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        mcao:UMLClassSDK,
        acao:hasArtifactOrigin some acao:AndroidArtifact


Class: acao:StoryCardTemplate

    Annotations:
        rdfs:comment "Mobile-D story card template."@en,
        rdfs:label "Story Card Template"

    SubClassOf:
        acao:isUsedByTask some acao:IterationPlanningTask,
        mcao:hasReusabilityLevel some mcao:Completely,
        mcao:hasReusabilityLevel only mcao:Completely,
        acao:isUsedByTask only acao:IterationPlanningTask,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        mcao:isSimilarToArtifact only acao:StoryCardTemplate,
        acao:isPartOfArtifact some acao:MobileDProcessLibrary,
        acao:isPartOfArtifact only
            (acao:MobileDProcessLibrary
             or acao:StoryCard),
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
```

```
        acao:Artifact,
        acao:hasArtifactType only acao:Template,
        mcao:isSimilarToArtifact some acao:StoryCardTemplate,
        acao:hasArtifactType some acao:Template,
        not (acao:isUpdatedByTask some acao:Task),
        not (acao:isCreatedByTask some acao:Task),
        acao:isPartOfArtifact some acao:StoryCard


Class: acao:ProjectManagementSoftwareTool

    Annotations:
        rdfs:comment "The tool used for project management."@en,
        rdfs:label "Project Management Software Tool"

    SubClassOf:
        acao:hasArtifactOrigin only acao:OtherArtifact,
        not (acao:isPartOfArtifact some acao:Artifact),
        not (acao:isUsedByTask some acao:Task),
        mcao:hasReusabilityLevel some mcao:Completely,
        mcao:hasReusabilityLevel only mcao:Completely,
        mcao:isSimilarToArtifact only acao:ProjectManagementSoftwareTool,
        mcao:isSimilarToArtifact some acao:ProjectManagementSoftwareTool,
        acao:Artifact,
        not (acao:isUpdatedByTask some acao:Task),
        acao:isCreatedByTask some acao:ProcessEstablishmentTask,
        acao:hasArtifactType only acao:Software,
        acao:isCreatedByTask only acao:ProcessEstablishmentTask,
        acao:hasArtifactOrigin some acao:OtherArtifact,
        acao:hasArtifactType some acao:Software


Class: acao:FinalDocumentation

    EquivalentTo:
        acao:Artifact
         and (not (acao:BorrowedArtifacts))
         and (not (acao:isPartOfArtifact some acao:Artifact))
         and (acao:hasArtifactType some acao:Document)

    SubClassOf:
        mcao:ArtifactsUsage


Class: acao:ApplicationManifest

    Annotations:
        rdfs:label "Application Manifest",
        rdfs:comment "XML document containing the information on application. This
             document is most important code artifact."@en

    SubClassOf:
        acao:isPartOfArtifact only
            (acao:DeploymentPackage
             or acao:MobileApplication),
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        acao:isPartOfArtifact some acao:DeploymentPackage,
        acao:hasArtifactOrigin some acao:AndroidArtifact,
        acao:isPartOfArtifact some acao:MobileApplication,
```

```
        mcao:AppManifest


Class: acao:Artifact

    Annotations:
        rdfs:comment "Artifact - Any piece of software developed and used during
             software development and maintenance."@en


Class: acao:ProcessEstablishmentTask

    Annotations:
        rdfs:comment "The purpose of this task is to establish the baseline process
             for the forthcoming software development project and to train the
             project team on using it. The aim of the Process Establishment is to
             make sure that the project team has all the needed competence
             regarding both the process and the technical aspects of the project.
             Thus, the need for training should be identified during this task.
             Also, deciding upon how the project's progress will be monitored and
             product's quality assured are important issues in every project, yet
             they may differ largely depending on the organization and product at
             hand. Thus, the monitoring, including the definition of metrics to be
             collected, and quality assurance tasks are to be planned based on
             these issues. For example, due to the high criticality of the end
             product it may be important to increase review practices in the
             process. Also, the life-cycle of the product effects on how quality
             issues such as variability should be perceived in the process or what
             is the criteria for product completion."@en,
        rdfs:label "Process Establishment Task",
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#ProjectEstabl
             ishmentActivity>

    SubClassOf:
        acao:isPerformedIn only acao:ProjectEstablishmentActivity,
        acao:Task,
        acao:isPerformedIn some acao:ProjectEstablishmentActivity


Class: acao:ArchitectureLineDefinitionTask

    Annotations:
        rdfs:comment "The purpose of this task is to get enough confidence in the
             architectural issues that the project can be successfully carried
             out."@en,
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#ProjectEstabl
             ishmentActivity>,
        rdfs:label "Architecture Line Definition Task"

    SubClassOf:
        acao:isPerformedIn only acao:ProjectEstablishmentActivity,
        acao:Task,
        acao:isPerformedIn some acao:ProjectEstablishmentActivity


Class: acao:UMLClass
```

```
    Annotations:
        rdfs:label "Class",
        rdfs:comment "UML model element used to describe a new class that is to be
             implemented."@en

    SubClassOf:
        acao:isUpdatedByTask some acao:RequirementsAnalysisTask,
        mcao:isSimilarToArtifact only acao:UMLClass,
        acao:hasArtifactType only acao:ModelElement,
        acao:isPartOfArtifact some acao:ClassModelMobile,
        acao:isPartOfArtifact only
            (acao:ClassModelMobile
             or acao:ClassModelWeb),
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:isUsedByTask some acao:RefactoringPractice,
        mcao:hasReusabilityLevel only mcao:Partially,
        acao:isUpdatedByTask some acao:PairProgrammingPractice,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:isPartOfArtifact some acao:ClassModelWeb,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:Artifact,
        acao:isUsedByTask some acao:DocumentationWrapUpTask,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:DocumentationWrapUpTask
             or acao:PairProgrammingPractice
             or acao:RefactoringPractice
             or acao:TestDrivenDevelopmentPractice),
        acao:isUpdatedByTask some acao:RefactoringPractice,
        acao:isCreatedByTask only acao:InitialRequirementsAnalysisTask,
        acao:isUsedByTask some acao:ContinuousIntegrationPractice,
        acao:isCreatedByTask some acao:InitialRequirementsAnalysisTask,
        acao:hasArtifactType some acao:ModelElement,
        acao:isUpdatedByTask only
            (acao:PairProgrammingPractice
             or acao:RefactoringPractice
             or acao:RequirementsAnalysisTask),
        mcao:isSimilarToArtifact some acao:UMLClass


Class: acao:ModelElement

    Annotations:
        rdfs:comment "Represents the atomic level (i.e. integral) artifact that
             could be observed as stand-alone and is used to create models."@en

    SubClassOf:
        acao:ArtifactType


Class: acao:LayoutElement

    Annotations:
        rdfs:comment "Represents XML code that is used to describe any user
             interface element such as text box, list box, button etc."@en,
        rdfs:label "Layout Element"
```

```
    SubClassOf:
        mcao:ViewElement,
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        acao:isPartOfArtifact only acao:Layout,
        acao:isPartOfArtifact some acao:Layout,
        acao:hasArtifactOrigin some acao:AndroidArtifact


Class: acao:MethodologicalArtifacts

    EquivalentTo:
        acao:Artifact
         and (acao:hasArtifactOrigin some acao:MethodologicalArtifact)

    SubClassOf:
        mcao:ArtifactsOrigin


Class: mcao:MapsKey

    Annotations:
        rdfs:label "Maps Key",
        rdfs:comment "Platform specific requirement needed for use of map
             services."@en

    SubClassOf:
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:hasArtifactType some acao:Licence,
        acao:isCreatedByTask some acao:PairProgrammingPractice,
        mcao:hasReusabilityLevel only mcao:None,
        acao:hasArtifactType only acao:Licence,
        acao:Artifact,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        not (acao:isUpdatedByTask some acao:Task),
        acao:isUsedByTask only acao:PublishApplicationTask,
        acao:isCreatedByTask only acao:PairProgrammingPractice,
        mcao:hasReusabilityLevel some mcao:None,
        acao:isUsedByTask some acao:PublishApplicationTask


Class: mcao:AppManifest

    Annotations:
        rdfs:label "App Manifest",
        rdfs:comment "Platform specific document containing the formated
             information on application. This document is most important code
             artifact."@en

    SubClassOf:
        acao:hasArtifactType only acao:Code,
        acao:isCreatedByTask some acao:PairProgrammingPractice,
        acao:hasArtifactType some acao:Code,
        acao:Artifact,
        mcao:hasReusabilityLevel only mcao:None,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        not (acao:isUpdatedByTask some acao:Task),
        acao:isUsedByTask only acao:PublishApplicationTask,
        mcao:hasReusabilityLevel some mcao:None,
        acao:isCreatedByTask only acao:PairProgrammingPractice,
```

```
        acao:isUsedByTask some acao:PublishApplicationTask


Class: acao:WorkingDayIn0IterationActivity

    Annotations:
        rdfs:label "Trial Day (Working Day in 0 Iteration)",
        acao:inPhase
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#Initialize>,
        rdfs:comment "The purpose of this stage is to trial and further set-up the
             technical development environment and to make sure that everything is
             ready for implementing the software development product. Also, the
             purpose is to implement some core functionality of the system (e.g.
             client-server communication) or solve some critical development issue
             without producing any working code. Also further technological
             investigations are possible in this stage. If the development decides
             to implement some functionality at this point, it need not to be the
             highest  priority functionality as defined by the customer but rather
             have been selected based on their importance concerning, for example,
             the architectural structure of the product. Trial Days form the pre-
             phase for the actual development days."@en

    SubClassOf:
        acao:consistsOf some acao:RefactoringPractice,
        acao:consistsOf only
            (acao:ContinuousIntegrationPractice
             or acao:PairProgrammingPractice
             or acao:RefactoringPractice
             or acao:TestDrivenDevelopmentPractice
             or acao:WrapUpTask),
        acao:consistsOf some acao:WrapUpTask,
        acao:consistsOf some acao:ContinuousIntegrationPractice,
        acao:consistsOf some acao:PairProgrammingPractice,
        acao:consistsOf some acao:TestDrivenDevelopmentPractice,
        acao:isPerformedIn some acao:Initialize,
        acao:Activity,
        acao:isPerformedIn only acao:Initialize


Class: wpcao:IntegrationTest

    Annotations:
        rdfs:comment "Represents the description and results of integration test
             that is performed manually. This document is part of System Test Plan
             document."@en,
        rdfs:label "Integration Test WP"

    SubClassOf:
        acao:isPartOfArtifact only acao:SystemTestPlan,
        acao:isPartOfArtifact some acao:SystemTestPlan,
        acao:isUpdatedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:PreReleaseTestingTask
             or acao:SystemIntegrationTask),
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:PreReleaseTestingTask
             or acao:SystemIntegrationTask
```

```
            or acao:SystemTestTask),
        acao:hasArtifactType some acao:DocumentElement,
        acao:hasArtifactType only acao:DocumentElement,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        mcao:IntegrationTest


Class: acao:EnvironmentSetUpTask

    Annotations:
        rdfs:label "Environment Set-up Task",
        rdfs:comment "The purpose of this task is to set-up development and other
             environment needed for project team in development process."@en,
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#ProjectSetUpA
             ctivity>

    SubClassOf:
        acao:Task,
        acao:isPerformedIn only acao:ProjectSetUpActivity,
        acao:isPerformedIn some acao:ProjectSetUpActivity


Class: acao:TaskCard

    Annotations:
        rdfs:comment "Basic documentation card containing the information on one
             task that is to be performed during the iteration. it is defined
             during the planning day and refined during implementation and wrap-up.
             It is part of the Product backlog document."@en,
        rdfs:label "Task Card"

    SubClassOf:
        acao:isUpdatedByTask some acao:WrapUpTask,
        acao:isCreatedByTask only acao:IterationPlanningTask,
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:isPartOfArtifact only acao:ProductBacklog,
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:isUsedByTask some acao:WrapUpTask,
        acao:isPartOfArtifact some acao:ProductBacklog,
        acao:isUsedByTask some acao:AcceptanceTestGenerationTask,
        mcao:isSimilarToArtifact some acao:TaskCard,
        mcao:isSimilarToArtifact only acao:TaskCard,
        mcao:hasReusabilityLevel only mcao:Partially,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:hasArtifactType some acao:DocumentElement,
        acao:hasArtifactType only acao:DocumentElement,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:Artifact,
        acao:isUsedByTask only
            (acao:AcceptanceTestGenerationTask
             or acao:PairProgrammingPractice
             or acao:TestDrivenDevelopmentPractice
             or acao:WrapUpTask),
        acao:isUpdatedByTask only acao:WrapUpTask,
        acao:isCreatedByTask some acao:IterationPlanningTask
```

```
Class: mcao:AppResource

    Annotations:
        rdfs:label "App Resource",
        rdfs:comment "Platform specific, usually XML based, code which describes
             different application resorces, including values, controls etc."@en,
        mcao:NOTICE "Axioms for isSimilarToArtifact are used in leaf elements."@en

    SubClassOf:
        acao:hasArtifactType only acao:Code,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:PairProgrammingPractice
             or acao:PreReleaseTestingTask
             or acao:RefactoringPractice
             or acao:SystemIntegrationTask
             or acao:SystemTestTask
             or acao:TestDrivenDevelopmentPractice),
        acao:isUsedByTask some acao:SystemIntegrationTask,
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:hasArtifactType some acao:Code,
        acao:isUpdatedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:PreReleaseTestingTask
             or acao:RefactoringPractice
             or acao:SystemIntegrationTask),
        acao:isUpdatedByTask some acao:SystemIntegrationTask,
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:isUpdatedByTask some acao:ContinuousIntegrationPractice,
        acao:isUsedByTask some acao:RefactoringPractice,
        acao:isUsedByTask some acao:SystemTestTask,
        mcao:hasReusabilityLevel only mcao:Partially,
        acao:isCreatedByTask some acao:PairProgrammingPractice,
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:Artifact,
        acao:isCreatedByTask only acao:PairProgrammingPractice,
        acao:isUsedByTask some acao:PreReleaseTestingTask,
        acao:isUpdatedByTask some acao:RefactoringPractice,
        acao:isUpdatedByTask some acao:PreReleaseTestingTask,
        acao:isUsedByTask some acao:ContinuousIntegrationPractice


Class: acao:CustomerEstablishmentTask

    Annotations:
        rdfs:label "Customer Establishment Task",
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#StakeholderEs
             tablishmentActivity>,
        rdfs:comment "The purpose of this task is to establish the customer
             interest group that has the ultimate expertise, domain knowledge and
             authority of the requirements for the software product."@en

    SubClassOf:
        acao:isPerformedIn only acao:StakeholderEstablishmentActivity,
        acao:Task,
        acao:isPerformedIn some acao:StakeholderEstablishmentActivity
```

```
Class: acao:WebService

    Annotations:
        rdfs:label "Web Service",
        rdfs:comment "The web part of the system created in the development
             process."@en

    SubClassOf:
        not (acao:isUsedByTask some acao:Task),
        mcao:hasReusabilityLevel some mcao:Completely,
        acao:isUpdatedByTask only acao:SystemIntegrationTask,
        acao:hasArtifactOrigin only acao:ServiceArtifact,
        mcao:hasReusabilityLevel only mcao:Completely,
        acao:isUpdatedByTask some acao:SystemIntegrationTask,
        mcao:isSimilarToArtifact some acao:WebService,
        mcao:isSimilarToArtifact only acao:WebService,
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:isCreatedByTask some acao:PairProgrammingPractice,
        acao:Artifact,
        acao:isCreatedByTask only acao:PairProgrammingPractice,
        acao:hasArtifactType some acao:Product,
        acao:hasArtifactType only acao:Product,
        acao:hasArtifactOrigin some acao:ServiceArtifact


Class: acao:TestResults

    Annotations:
        rdfs:comment "Results are obtained during the whole development process
             testing tasks. At the end this document becomes part of System test
             report."@en,
        rdfs:label "Test Results"

    SubClassOf:
        acao:isUpdatedByTask some acao:AcceptanceTestingTask,
        acao:isUpdatedByTask only
            (acao:AcceptanceTestingTask
             or acao:PreReleaseTestingTask
             or acao:SystemIntegrationTask
             or acao:SystemTestTask),
        acao:isUsedByTask only acao:PairProgrammingPractice,
        acao:isUpdatedByTask some acao:SystemIntegrationTask,
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:isCreatedByTask some acao:TestDrivenDevelopmentPractice,
        mcao:hasReusabilityLevel some mcao:None,
        acao:isUpdatedByTask some acao:SystemTestTask,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:hasArtifactType only acao:DocumentElement,
        acao:hasArtifactType some acao:DocumentElement,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:Artifact,
        mcao:hasReusabilityLevel only mcao:None,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        acao:isPartOfArtifact some acao:SystemTestReport,
        acao:isPartOfArtifact only acao:SystemTestReport,
        acao:isCreatedByTask only acao:TestDrivenDevelopmentPractice,
        acao:isUpdatedByTask some acao:PreReleaseTestingTask
```

```
Class: wpcao:APIDocumentation

    Annotations:
        rdfs:comment "WP API documentation from http://msdn.microsoft.com"@en,
        rdfs:label "API Documentation WP"

    SubClassOf:
        mcao:APIDocumentation,
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact


Class: acao:RefactoringPractice

    Annotations:
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#WorkingDayAct
             ivity>,
        rdfs:comment "The purpose of refactoring is to improve existing software's
             internal structure without modifying its external behavior. With small
             improvements to code, refactoring ensures that software is more
             modifiable, extendable, and readable. When refactoring is a regular
             habit during development it reduces the need to design up front.
             Instead the software is evolved by adapting to changes and improving
             the design of existing software. "@en,
        rdfs:label "Refactoring Practice",
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#WorkingDayIn0
             IterationActivity>

    SubClassOf:
        acao:isPerformedIn some acao:WorkingDayIn0IterationActivity,
        acao:Task,
        acao:isPerformedIn only
            (acao:WorkingDayActivity
             or acao:WorkingDayIn0IterationActivity),
        acao:isPerformedIn some acao:WorkingDayActivity


Class: acao:InitialProjectPlanningTask

    Annotations:
        rdfs:label "Initial Project Planning Task",
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#ScopeDefiniti
             onActivity>,
        rdfs:comment "The purpose of this task is to establish the initial plan for
             the forthcoming software development project regarding the timeline,
             rhythm and investments of the project. This is done in order to enable
             the further establishment of the project."@en

    SubClassOf:
        acao:Task,
        acao:isPerformedIn only acao:ScopeDefinitionActivity,
        acao:isPerformedIn some acao:ScopeDefinitionActivity


Class: acao:Model
```

```
    Annotations:
        rdfs:comment "Represents models that are created during the development
             process. Models could be observed as stand-alone artifacts, but are
             usually presented as a part of some document."@en

    SubClassOf:
        acao:ArtifactType


Class: acao:TestDrivenDevelopmentPractice

    Annotations:
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#WorkingDayAct
             ivity>,
        rdfs:comment "The purpose of TDD is to give the developers confidence that
             code they produce works and guide the design of the code to clearer
             more easily testable structure. TDD is also tightly coupled with
             refactoring practice because the test set that is produced with TDD is
             used while refactoring to ensurethat the change did not break the
             existing functionality of the system. In TDD the unit tests are
             written before the program code. The program code is then developed to
             work with the already written tests."@en,
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#WorkingDayIn0
             IterationActivity>,
        rdfs:label "Test Driven Development Practice"

    SubClassOf:
        acao:isPerformedIn some acao:WorkingDayIn0IterationActivity,
        acao:Task,
        acao:isPerformedIn only
            (acao:WorkingDayActivity
             or acao:WorkingDayIn0IterationActivity),
        acao:isPerformedIn some acao:WorkingDayActivity


Class: acao:MobileDProcessLibrary

    Annotations:
        rdfs:label "Mobile-D Process Library",
        rdfs:comment "Process library describing the Mobile-D methodology in
             detail. Used as methodology guidelines in every phase."@en

    SubClassOf:
        mcao:hasReusabilityLevel some mcao:Completely,
        mcao:hasReusabilityLevel only mcao:Completely,
        not (acao:isUpdatedByTask some acao:Task),
        acao:hasArtifactType some acao:Document,
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:isUsedByTask only acao:Task,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        mcao:isSimilarToArtifact only acao:MobileDProcessLibrary,
        mcao:isSimilarToArtifact some acao:MobileDProcessLibrary,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:isUsedByTask some acao:Task,
        acao:Artifact,
        acao:hasArtifactType only acao:Document,
        not (acao:isCreatedByTask some acao:Task)
```

```
Class: acao:DocumentationWrapUpTasks

    EquivalentTo:
        acao:isPerformedIn some acao:DocumentationWrapUpActivity

    SubClassOf:
        acao:TasksByActivities


Class: acao:SystemTestPlan

    Annotations:
        rdfs:label "System Test Plan",
        rdfs:comment "Contains the information on purpose, plan and definitions of
             tests."@en

    SubClassOf:
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:isUsedByTask some acao:ArchitectureLineDefinitionTask,
        acao:isCreatedByTask some acao:InitialProjectPlanningTask,
        acao:isUpdatedByTask only
            (acao:InitialRequirementsAnalysisTask
             or acao:PostIterationWorkshopTask
             or acao:ProcessEstablishmentTask
             or acao:SystemTestTask),
        mcao:hasReusabilityLevel some mcao:None,
        acao:isUpdatedByTask some acao:ProcessEstablishmentTask,
        acao:isUsedByTask some acao:SystemTestTask,
        acao:isUpdatedByTask some acao:SystemTestTask,
        acao:isUsedByTask some acao:ArchitectureLinePlanningTask,
        acao:isUsedByTask some acao:IterationPlanningTask,
        acao:hasArtifactType some acao:Document,
        not (acao:isPartOfArtifact some acao:Artifact),
        acao:isUpdatedByTask some acao:InitialRequirementsAnalysisTask,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:Artifact,
        mcao:hasReusabilityLevel only mcao:None,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        acao:isUsedByTask some acao:DocumentationWrapUpTask,
        acao:hasArtifactType only acao:Document,
        acao:isUsedByTask only
            (acao:ArchitectureLineDefinitionTask
             or acao:ArchitectureLinePlanningTask
             or acao:DocumentationWrapUpTask
             or acao:IterationPlanningTask
             or acao:ProcessEstablishmentTask
             or acao:SystemTestTask
             or acao:TestDrivenDevelopmentPractice),
        acao:isUsedByTask some acao:ProcessEstablishmentTask,
        acao:isCreatedByTask only acao:InitialProjectPlanningTask,
        acao:isUpdatedByTask some acao:PostIterationWorkshopTask


Class: acao:Product

    Annotations:
```

```
        rdfs:comment "Represents final product as most important project
             deliverable. "@en

    SubClassOf:
        acao:ArtifactType


Class: acao:TaskCardTemplate

    Annotations:
        rdfs:label "Task Card Template",
        rdfs:comment "Mobile-D task card template."@en

    SubClassOf:
        mcao:hasReusabilityLevel some mcao:Completely,
        mcao:hasReusabilityLevel only mcao:Completely,
        acao:isPartOfArtifact some acao:TaskCard,
        acao:hasArtifactType some acao:Template,
        not (acao:isUpdatedByTask some acao:Task),
        mcao:isSimilarToArtifact some acao:TaskCardTemplate,
        acao:isUsedByTask some acao:IterationPlanningTask,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:isUsedByTask only acao:IterationPlanningTask,
        acao:isPartOfArtifact some acao:MobileDProcessLibrary,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
        acao:Artifact,
        acao:isPartOfArtifact only
            (acao:MobileDProcessLibrary
             or acao:TaskCard),
        acao:hasArtifactType only acao:Template,
        mcao:isSimilarToArtifact only acao:TaskCardTemplate,
        not (acao:isCreatedByTask some acao:Task)


Class: acao:PostIterationWorkshopTask

    Annotations:
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#PlanningDayAc
             tivity>,
        rdfs:comment "The purpose of this task is to iteratively enhance the
             software development process to better fit the needs of current
             software project team. "@en,
        rdfs:label "Post-iteration Workshop Task"

    SubClassOf:
        acao:isPerformedIn only acao:PlanningDayActivity,
        acao:Task,
        acao:isPerformedIn some acao:PlanningDayActivity


Class: mcao:AppScreenshot

    Annotations:
        rdfs:comment "Application screenshots showcasing final or intermediate
             application look. Screenshot is usually platform specific due to
             different native look and feel of every platform. Some exceptions are
             possible."@en,
```

```
        mcao:NOTICE "Closure axiom for isPartOfArtifact is used in leaf
             elements."@en,
        rdfs:label "App Screenshot"

    SubClassOf:
        acao:isUpdatedByTask only
            (acao:DocumentationWrapUpTask
             or acao:SystemIntegrationTask),
        not (acao:isUsedByTask some acao:Task),
        acao:hasArtifactType only acao:Resource,
        acao:isCreatedByTask only acao:AcceptanceTestingTask,
        acao:isUpdatedByTask some acao:DocumentationWrapUpTask,
        acao:isCreatedByTask some acao:AcceptanceTestingTask,
        mcao:hasReusabilityLevel only mcao:None,
        acao:Artifact,
        acao:isUpdatedByTask some acao:SystemIntegrationTask,
        not (mcao:isSimilarToArtifact some acao:Artifact),
        acao:isPartOfArtifact some acao:SADDDocument,
        acao:hasArtifactType some acao:Resource,
        mcao:hasReusabilityLevel some mcao:None


Class: wpcao:DotNetClass

    Annotations:
        rdfs:comment "UML model element used to describe an existing .Net class
             that is to be used."@en,
        rdfs:label ".Net Class"

    SubClassOf:
        mcao:UMLClassSDK,
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact


Class: acao:DeploymentPackage

    Annotations:
        rdfs:label "Deployment Package Android",
        rdfs:comment "APK file created for publishing purposes."@en

    SubClassOf:
        acao:hasArtifactOrigin only acao:AndroidArtifact,
        mcao:DeploymentPackage,
        acao:hasArtifactOrigin some acao:AndroidArtifact


Class: acao:Productionize

    Annotations:
        rdfs:comment "The Productionize and Stabilize phases are executed
             iteratively in order to develop all other features of the mobile
             product. Iterations start with planning day in Productionize phase.
             The first activity is post-iteration workshop which aims to enhance
             the development process to better fit the needs of the current
             software development team. The requirements analysis, iteration
             planning and acceptance test generation tasks follow and are executed
             during the planning day. The working day is based on implementation
             through test driven development, pair programming, continuous
```

```
            integration and refactoring. This day ends with the task of informing
            the customer on new functionality. Finally, the release day includes
            the activities of integration and testing."@en,
        rdfs:label "Productionize Phase"

    SubClassOf:
        acao:consistsOf some acao:ReleaseDayActivity,
        acao:consistsOf some acao:PlanningDayActivity,
        acao:consistsOf only
            (acao:PlanningDayActivity
             or acao:ReleaseDayActivity
             or acao:WorkingDayActivity),
        acao:Phase,
        acao:consistsOf some acao:WorkingDayActivity


Class: acao:Task

    Annotations:
        rdfs:comment "Tasks are performed in order to achieve defined goals."@en


Class: acao:Explore

    Annotations:
        rdfs:comment "The aim of the first phase, called Explore, is to prepare the
             foundation for future development."@en,
        rdfs:label "Explore Phase"

    SubClassOf:
        acao:consistsOf some acao:ProjectEstablishmentActivity,
        acao:consistsOf some acao:ScopeDefinitionActivity,
        acao:Phase,
        acao:consistsOf some acao:StakeholderEstablishmentActivity,
        acao:consistsOf only
            (acao:ProjectEstablishmentActivity
             or acao:ScopeDefinitionActivity
             or acao:StakeholderEstablishmentActivity)


Class: acao:PHPCode

    Annotations:
        rdfs:label "PHP Code",
        rdfs:comment "PHP code developed during the implementation activities."@en

    SubClassOf:
        acao:isUsedByTask some acao:SystemIntegrationTask,
        acao:hasArtifactOrigin only acao:ServiceArtifact,
        mcao:hasReusabilityLevel only mcao:Completely,
        acao:hasArtifactType some acao:Code,
        mcao:isSimilarToArtifact some acao:PHPCode,
        acao:isUsedByTask some acao:SystemTestTask,
        mcao:isSimilarToArtifact only acao:PHPCode,
        acao:Artifact,
        acao:isUsedByTask some acao:DocumentationWrapUpTask,
        acao:isCreatedByTask only acao:PairProgrammingPractice,
        acao:isUpdatedByTask some acao:RefactoringPractice,
        acao:isUsedByTask some acao:PreReleaseTestingTask,
```

```
        acao:isUsedByTask some acao:ContinuousIntegrationPractice,
        acao:isPartOfArtifact some acao:WebService,
        acao:hasArtifactOrigin some acao:ServiceArtifact,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:DocumentationWrapUpTask
             or acao:PairProgrammingPractice
             or acao:PreReleaseTestingTask
             or acao:RefactoringPractice
             or acao:SystemIntegrationTask
             or acao:SystemTestTask
             or acao:TestDrivenDevelopmentPractice),
        mcao:hasReusabilityLevel some mcao:Completely,
        acao:hasArtifactType only acao:Code,
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:isUpdatedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:PreReleaseTestingTask
             or acao:RefactoringPractice
             or acao:SystemIntegrationTask),
        acao:isUpdatedByTask some acao:SystemIntegrationTask,
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:isUpdatedByTask some acao:ContinuousIntegrationPractice,
        acao:isUsedByTask some acao:RefactoringPractice,
        acao:isCreatedByTask some acao:PairProgrammingPractice,
        acao:isPartOfArtifact only acao:WebService,
        acao:isUpdatedByTask some acao:PreReleaseTestingTask


Class: mcao:View

    Annotations:
        rdfs:comment "Represents platform specific, usually XML based, code that is
             used to describe user interface form or screen."@en,
        rdfs:label "View"

    SubClassOf:
        acao:hasArtifactType only acao:Code,
        acao:isUsedByTask some acao:SystemIntegrationTask,
        acao:isUsedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:PairProgrammingPractice
             or acao:PreReleaseTestingTask
             or acao:RefactoringPractice
             or acao:SystemIntegrationTask
             or acao:SystemTestTask
             or acao:TestDrivenDevelopmentPractice),
        acao:isUsedByTask some acao:TestDrivenDevelopmentPractice,
        acao:hasArtifactType some acao:Code,
        mcao:isSimilarToArtifact some mcao:View,
        acao:isUpdatedByTask only
            (acao:ContinuousIntegrationPractice
             or acao:PreReleaseTestingTask
             or acao:RefactoringPractice
             or acao:SystemIntegrationTask),
        acao:isUpdatedByTask some acao:SystemIntegrationTask,
        acao:isUsedByTask some acao:PairProgrammingPractice,
        acao:isUpdatedByTask some acao:ContinuousIntegrationPractice,
        acao:isUsedByTask some acao:SystemTestTask,
```

```
        acao:isUsedByTask some acao:RefactoringPractice,
        mcao:hasReusabilityLevel only mcao:Partially,
        acao:isCreatedByTask some acao:PairProgrammingPractice,
        mcao:hasReusabilityLevel some mcao:Partially,
        acao:Artifact,
        acao:isCreatedByTask only acao:PairProgrammingPractice,
        acao:isUsedByTask some acao:PreReleaseTestingTask,
        acao:isUpdatedByTask some acao:RefactoringPractice,
        acao:isUpdatedByTask some acao:PreReleaseTestingTask,
        acao:isUsedByTask some acao:ContinuousIntegrationPractice,
        mcao:isSimilarToArtifact only mcao:View


Class: acao:CustomerCommunicationEstablishmentTask

    Annotations:
        rdfs:comment "The purpose of this task is to agree on the customs of how
             the project manager/team will communicate with the customer during the
             software development. The aim is to ensure the appropriate,
             informative and intensive communication between the team and the
             customer to assure that all the stakeholders can access the
             information they need as soon as possible. Thus, it enables the fluent
             implementation of correct requirements. The effective communication
             isneeded between the customer group as well as the software
             developers, especially in the case of off-site customer."@en,
        rdfs:label "Customer Communication Establishment Task",
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#ProjectSetUpA
             ctivity>

    SubClassOf:
        acao:isPerformedIn only acao:ProjectSetUpActivity,
        acao:Task,
        acao:isPerformedIn some acao:ProjectSetUpActivity


Class: acao:ArchitectureLineDescription

    Annotations:
        rdfs:label "Architecture Line Description",
        rdfs:comment "Created during the architecture line planning task and
             updated in subsequent iterations. Contains the information on system
             context, technological scope, architectural risks etc. This document
             is part of project plan."@en

    SubClassOf:
        acao:isUsedByTask some acao:RequirementsAnalysisTask,
        acao:isUpdatedByTask some acao:ArchitectureLinePlanningTask,
        acao:isCreatedByTask only acao:ArchitectureLineDefinitionTask,
        acao:isPartOfArtifact only acao:ProjectPlan,
        mcao:hasReusabilityLevel some mcao:None,
        acao:isUsedByTask some acao:ArchitectureLinePlanningTask,
        acao:isUsedByTask some acao:IterationPlanningTask,
        acao:isCreatedByTask some acao:ArchitectureLineDefinitionTask,
        acao:isPartOfArtifact some acao:ProjectPlan,
        acao:hasArtifactOrigin only acao:MethodologicalArtifact,
        acao:hasArtifactType some acao:DocumentElement,
        acao:hasArtifactType only acao:DocumentElement,
        acao:hasArtifactOrigin some acao:MethodologicalArtifact,
```

```
        mcao:hasReusabilityLevel only mcao:None,
        acao:Artifact,
        acao:isUpdatedByTask only acao:ArchitectureLinePlanningTask,
        acao:isUsedByTask only
            (acao:ArchitectureLinePlanningTask
             or acao:IterationPlanningTask
             or acao:RequirementsAnalysisTask),
        not (mcao:isSimilarToArtifact some acao:Artifact)


Class: acao:Licence

    Annotations:
        rdfs:comment "Represents individual-specific unique key that is obtained or
             used during the development process."@en

    SubClassOf:
        acao:ArtifactType


Class: acao:PublishApplicationTask

    Annotations:
        rdfs:comment "Although Mobile-D does not explicitly define Publish
             Application Task (as methodology is defined prior to concept of mobile
             stores is introduced) it can be done during the System Test and Fix
             phase as a part of Working day. In that manner we add this task to the
             ontology as it is crutial for some artifacts that are strictly
             connected to application publishment."@en,
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#SystemTestAct
             ivity>,
        rdfs:label "Publish Application Task",
        rdfs:label "Process Establishment Task"

    SubClassOf:
        acao:isPerformedIn only acao:SystemTestActivity,
        acao:Task,
        acao:isPerformedIn some acao:SystemTestActivity


Class: mcao:MapsSDK

    Annotations:
        rdfs:comment "Referenced platform specific libraries providing map
             component and use of maps in mobile application.."@en,
        rdfs:label "Maps SDK"

    SubClassOf:
        mcao:AppReference


Class: acao:ProjectEstablishmentTasks

    EquivalentTo:
        acao:isPerformedIn some acao:ProjectEstablishmentActivity

    SubClassOf:
        acao:TasksByActivities
```

```
Class: wpcao:ApplicationScreenshot

    Annotations:
        rdfs:comment "Application screenshots are created as needed for publishing
             process."@en,
        rdfs:label "Application Screenshot WP"

    SubClassOf:
        mcao:AppScreenshot,
        acao:isPartOfArtifact some wpcao:DeploymentPackage,
        acao:hasArtifactOrigin only wpcao:WindowsPhoneArtifact,
        acao:hasArtifactOrigin some wpcao:WindowsPhoneArtifact,
        acao:isPartOfArtifact only
            (acao:SADDDocument
             or wpcao:DeploymentPackage)


Class: acao:DocumentationWrapUpTask

    Annotations:
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#Documentation
             WrapUpActivity>,
        rdfs:label "Documentation Wrap-up Task",
        rdfs:comment "The purpose of this task is to produce documentation.
             Software without documentation is a disaster. Source code is not the
             ideal medium for communicating the rationale, structure and interfaces
             of a system. Documentation will be produced for project stakeholders
             and not for the agile team."@en

    SubClassOf:
        acao:isPerformedIn only acao:DocumentationWrapUpActivity,
        acao:Task,
        acao:isPerformedIn some acao:DocumentationWrapUpActivity


Class: acao:SystemIntegrationTask

    Annotations:
        rdfs:label "System Integration Task",
        rdfs:comment "Complex products may require that the systemis divided into
             smaller subsystems. In the case of multi-team project, the purpose
             ofthis task is to integrate subsystems, which are generated inseparate
             teams, into a single product."@en,
        acao:inActivity
             <http://www.foi.unizg.hr/ontologies/AndroidCaseArtifacts#ReleaseDayAct
             ivity>

    SubClassOf:
        acao:isPerformedIn only acao:ReleaseDayActivity,
        acao:isPerformedIn some acao:ReleaseDayActivity,
        acao:Task


Class: acao:ServiceArtifacts

    EquivalentTo:
```

```
        acao:Artifact
         and (acao:hasArtifactOrigin some acao:ServiceArtifact)

    SubClassOf:
        mcao:ArtifactsOrigin


DisjointClasses:

acao:DocumentationWrapUpActivity,acao:PlanningDayActivity,acao:PlanningDayIn0Iterat
ionActivity,acao:ProjectEstablishmentActivity,acao:ProjectSetUpActivity,acao:Releas
eDayActivity,acao:ScopeDefinitionActivity,acao:StakeholderEstablishmentActivity,aca
o:SystemTestActivity,acao:WorkingDayActivity,acao:WorkingDayIn0IterationActivity

DisjointClasses:

acao:Code,acao:Document,acao:DocumentElement,acao:Example,acao:Licence,acao:Model,a
cao:ModelElement,acao:Product,acao:Resource,acao:Software,acao:Standard,acao:Templa
te

DisjointClasses:

acao:Explore,acao:Initialize,acao:Productionize,acao:Stabilize,acao:SystemTestAndFi
x

DisjointClasses:

mcao:CompletlyResuableArtifacts,mcao:NotreusableArtifacts,mcao:PartiallyReusableArt
ifacts

DisjointClasses:

acao:AcceptanceTest,acao:AcceptanceTestTemplateSheet,acao:ArchitectureLineDescripti
on,acao:ArchitectureLinePlan,acao:ClassModelMobile,acao:ClassModelWeb,acao:DataMode
lMobile,acao:DataModelWeb,acao:DefectList,acao:DevelopmentUnrelatedSoftwareTool,aca
o:InitialRequirementsDocument,acao:IterationBacklog,acao:IterationsPlan,acao:JSONSt
andard,acao:MeasurementPlan,acao:MobileDProcessLibrary,acao:PHPCode,acao:ProductBac
klog,acao:ProductProposal,acao:ProjectManagementSoftwareTool,acao:ProjectPlan,acao:
ProjectPlanChecklist,acao:ProjectPlanChecklistTemplate,acao:ProjectPlanGanttChart,a
cao:SADDDocument,acao:StoryCard,acao:StoryCardTemplate,acao:SystemTestPlan,acao:Sys
temTestReport,acao:TaskCard,acao:TaskCardTemplate,acao:TestResults,acao:UIIllustrat
ions,acao:UMLClass,acao:WebDevelopmentEnvironment,acao:WebService,acao:WebServiceSp
ecification,mcao:APIDocumentation,mcao:AppDescription,mcao:AppIcon,mcao:AppManifest
,mcao:AppPrototypeFunctionality,mcao:AppReference,mcao:AppResource,mcao:AppScreensh
ot,mcao:DeploymentPackage,mcao:DevelopmentEnvironment,mcao:ExampleCode,mcao:Integra
tionTest,mcao:MapsKey,mcao:MobileApplication,mcao:SourceCode,mcao:TestDeviceDriver,
mcao:ThrowAwayPrototype,mcao:UMLClassSDK,mcao:UnitTest,mcao:View,mcao:ViewControlle
r,mcao:ViewElement

DisjointClasses:

acao:Activity,acao:Artifact,acao:ArtifactOrigin,acao:ArtifactType,acao:Phase,acao:T
ask

DisjointClasses:

acao:AcceptanceTestGenerationTask,acao:AcceptanceTestReviewTask,acao:AcceptanceTest
ingTask,acao:ArchitectureLineDefinitionTask,acao:ArchitectureLinePlanningTask,acao:
ContinuousIntegrationPractice,acao:CustomerCommunicationEstablishmentTask,acao:Cust
```

```
omerEstablishmentTask,acao:DocumentationWrapUpTask,acao:EnvironmentSetUpTask,acao:I
nformCustomerTask,acao:InitialProjectPlanningTask,acao:InitialRequirementsAnalysisT
ask,acao:InitialRequirementsCollectionTask,acao:IterationPlanningTask,acao:PairProg
rammingPractice,acao:PostIterationWorkshopTask,acao:PreReleaseTestingTask,acao:Proc
essEstablishmentTask,acao:PublishApplicationTask,acao:RefactoringPractice,acao:Rele
aseCeremoniesTask,acao:RequirementsAnalysisTask,acao:SystemIntegrationTask,acao:Sys
temTestTask,acao:TestDrivenDevelopmentPractice,acao:WrapUpTask

DisjointClasses:

acao:AndroidArtifact,acao:MethodologicalArtifact,acao:OtherArtifact,acao:ServiceArt
ifact,wpcao:WindowsPhoneArtifact

DisjointClasses:

acao:AndroidArtifacts,acao:MethodologicalArtifacts,acao:OtherArtifacts,acao:Service
Artifacts,wpcao:WindowsPhoneArtifacts
```

EXTENDED ABSTRACT

1. Introduction

Development of mobile systems is a challenging task which differs from traditional development in several important aspects. According to Hosbond (2005), the two main sets of challenges should be addressed in the domain of mobile systems development, namely *business* related challenges and *development* specific challenges. In this research we will focus on development specific challenges and will give special attention to the usage of methodologies which according to some authors, like Rahimian and Ramsin (2008), Spataru (2010) or La and Kim (2009), should be firstly addressed.

Classic or agile software development methodologies should be adapted for the development of mobile applications as the existing ones do not cover the specific mobile targeted requirements (La and Kim, 2009). There are several attempts from different authors to create new methodologies in order to cover the gaps in the domain of mobile applications. Some of them are *Agile Risk-based Methodology* (Rahimian and Ramsin, 2008), *MASAM* (Jeong et al., 2008), and *Mobile-D* (Abrahamsson et al., 2004). But still, there is no comprehensive research that answers the question like which existing or new methodologies are suitable for development of mobile applications.

On top of the list of methodology problems, the *fragmentation problem* forces the developers of mobile applications to focus on only specific platforms and versions (Manjunatha et al., 2010), but as the development of mobile applications primarily aims a wide range of users, such approach is not the preferable option and the development teams reach for different solutions of the problem proposed by professional and scientific community. First, we would like to mention the approach that enables the development teams to use a *mediatory language* or just *mediatory transform engine* to code for several target platforms. Some of the most influential projects are MobiCloud (Manjunatha et al., 2010), Rhodes (Rhomobile, Inc., 2011) and Amanquah & Eporwei code generator (Amanquah and Eporwei, 2009). These attempts have several advantages but also have significant drawbacks, like their dependability on the efforts invested in the transform engine, specific APIs and specific domain, the lack of control over generated source code and similar. Another possible solution to the problem could be the introduction of adapter applications (adapters) as native applications for every target platform. According to Agarwal et al. (2009) this is one of the two main techniques for handling fragmentation. As standardization of APIs in mobile world is still not possible, the usage of programming techniques whereby the interface calls are wrapped, i.e. abstracted, in distinct

modules which are then ported across the platforms, is left as the other solution. The representatives of this approach are MobiVine (Agarwal et al., 2009), PhoneGap (PhoneGap, 2011) or Adobe AIR® (Adobe Corporation, 2011). Almost all of the drawbacks stated for existing solutions that introduce a transform engine are also present in this solution. Finally, the third approach is to use web technologies and to develop cross-platform web applications, but this approach is out of our scope as it differs in many aspects (which also have their own drawbacks) from the basic assumptions taken in this research.

Therefore, in this research we focused on proposing a solution that would enhance methodological interoperability among teams working on the same application but in different (and native) development environments. The research answers the following questions: (1) what methodologies and development approaches can be used in multi-platform mobile applications development; (2) what artifacts (required inputs and outputs of methodologically and methodically defined development steps) emerge during mobile applications development, (3) whether and to what extent there are similarities between these artifacts, and (4) whether it is possible to ontologically describe these artifacts, and create a basis for development of a system that would support the methodological interoperability. Thus, the main goal is to ontologically describe artifacts that arise in the methodologically managed process of mobile application development targeting two or more mobile platforms, and to create the basis for more efficient and interoperable process of multi-platform mobile applications development.

In that sense the research aims to prove the following hypothesis: H_1 - *It is possible to create an ontological description of elements of methodological interoperability containing structural and semantic aspects of sets of artifacts created in the development process of a mobile application for two or more target platforms.*

The chapters of this summary are organized in accordance with the stated research questions. The second chapter brings a systematic review of mobile applications development methodologies; the results of the methodology implementation for two platforms are presented in the third chapter and in the fourth chapter we identify and cross-compare the artifacts that emerged in the process; in the fifth chapter the ontological definition of artifacts is given. In the last chapter we discuss the results and draw conclusions.

2. Mobile applications development methodologies: a systematic review

"A systematic literature review (SLR) is a means of evaluating and interpreting all available research relevant to a particular research question, topic area, or phenomenon of interest. Systematic reviews aim to present a fair evaluation of a research topic by using a trustworthy,

rigorous, and auditable methodology" (Kitchenham and Charters, 2007). As the method of SLR is rather new in the field of software engineering (SE), first we analyzed the best practices in performing such time consuming and comprehensive method. The guidelines given by Kitchenham and Charters (2007) are followed and discussed by adding the recommendations and findings from other influential authors in the field. Special focus is given to the problem of performing the method by PhD students. The findings of this research phase are followed during the execution of SLR which is reported in the next sections.

2.1. Performing the SLR

After performing a short and preliminary review of the existing methodologies we concluded that the development for mobile devices differs from the standard development, the agile approach is widely used in methodologies for mobile development and none of the observed methodologies is applicable without additional efforts to make the process more fine-grained or more suitable to specific development environment and mobile application requirements. This indicates that a thorough and unbiased research is needed in order to get an overall overview of possible methodologies that could be followed while developing applications for mobile devices.

Additionally, another preliminary research is performed to identify the existing systematic literature reviews on software development methodologies for development of mobile applications. The IEEExplore, ACM Digital library, INSPEC, CiteSeerX and GoogleScholar databases were searched by the following search query: *("literature review" OR SLR) AND (mobile development)*. According to information available in the mentioned databases, there are no existing systematic literature reviews covering the subject of software development methodologies for mobile applications development, which makes the need for such review even bigger.

In order to address the issues determined in this analysis, this systematic review is aligned to answer the following research questions:

- **RQ1** – What development methodologies and approaches are reported in literature as defined in theory or used in practice for mobile application development?
- **RQ2** – Are the identified methodologies and approaches applicable for multi-platform mobile applications development?

The review protocol was defined according to the instructions given in (Kitchenham and Charters, 2007) and the template used for the protocol is proposed by (Biolchini et al., 2005) and further explained by (Mian et al., 2005). Search string defined for the main research was *(mobile AND ("software development" OR "system development" OR "application development" OR "program development") AND (methodology OR method OR approach OR framework OR process OR procedure OR model))* and it was executed on available relevant

electronic sources (journals and proceedings) in the field of SE as identified by the field experts Brereton et al. (2007), Hannay et al. (2007) and by Kitchenham and Charters (2007).

The literature review was performed through several phases including the identification, inclusion and exclusion criteria application and quality assessment. Finally, 49 studies out of 6761 were identified as relevant for data extraction and synthesis.

As presented in Table 1 and Table 2 the total of 22 development methodologies and 7 development approaches were identified as newly developed or used and eligible for multi-platform mobile applications development.

Table 1 - Developed methodologies and approaches

Name	Type
Agile Methodology for Mobile Software Development	M
Agile Solo	M
Agile usability process	M
DEAL	M
Integrated Product Development Process for Mobile Software	M
Inter-combined Model	M
MASAM methodology	M
Methodology for Building Enterprise-Wide Mobile Applications	M
MicroApp visual approach	M
Mobile Application Development Methodology	M
Mobile-D	M
New media application prototyping	M
Systems Development Methodology	M
ViP (Virtual Platform)	M
Composite Application Software Development Process Framework	A
MobiLine	A

Type: M - Methodology, A - Approach

Table 2 - Used methodologies and approaches

Name	Type
Design Science	M
Dynamic Channel Model	M
Extreme Programming	M
Kanban	A
Mobile-D	M
Mobile Engineering (MobE)	M
Mobile RAD	M
Rapid Application Development	M
Scrum	M
Model Driven Development	A
Model Driven Product Lines	A
Software Product Lines	A
Test Driven Development	A

Type: M - Methodology, A - Approach

Only one methodology is covered by more than one study, while all other methodologies are presented in a single identified study. Additionally, as expected, the methodologies and approaches in the mobile development field are rather new. Only 4 studies are more than 5 years old, while all the other studies date in the last five years. The overall study quality

assessment score has the mean value of 2.735 out of 5 (68.38%) with the standard deviation of 0.903. This can be interpreted as relatively low study quality with high deviation in quality.

On the other hand, more authors reported the usage of methodology or approach than the creation of new methodology. Total of 9 methodologies and 4 approaches have been reported as used. The important fact is that only one methodology (Mobile-D) identified as newly created was reported to have been used. The usage of this methodology was reported in five different studies, while all other new methodologies and approaches were not reported as ever being used.

2.2. Choosing development methodology

As the basic assumption of the research is that methodological interoperability is platform and methodology independent (i.e. it can be performed on any methodology ontologically described), we can choose any of the 22 identified methodologies. To avoid random selection, the criterion used to choose development methodology was *reported applicability of newly developed methodologies.* Cross-analysis of the SLR results shows that *Mobile-D* is the only methodology specifically created for mobile applications development that was reported to be used in practice. In addition, we performed a research to identify other gray-literature sources published by the methodology creators and found that this methodology is thoroughly and in detail defined in several publications and the most important one is (Abrahamsson et al., 2005a).

3. Methodology implementation

Mobile-D process (see Figure 1) includes five phases that are executed in partially incremental order. The aim of the first phase, called *Explore*, is to prepare the foundation for future development. The *Initialize* phase should describe and prepare all components of the application as well as to predict possible critical issues of the project. This phase is usually called *zero iteration* (*0-iteration*) phase as it, in addition to *project set-up,* includes the stages of *planning day*, *working day* and *release day* which are also used in *Productionize* phase. The idea of the 0-iteration phase is to assure the functionality of the technical development environment through the implementation of some representative features or through prototyping. The *Productionize* and *Stabilize* phases are executed iteratively in order to develop all other features of the mobile product. Iterations start with *planning day* in *Productionize* phase. The first activity is *post-iteration workshop* which aims to enhance the development process to better fit the needs of the current software development team. *Requirements analysis*, *iteration planning* and *acceptance test generation* tasks follow and are executed during the *planning day*. *Working day* is based on implementation through *test*

driven development, *pair programming*, *continuous integration* and *refactoring*. This *day* ends with the task of *informing the customer* on new functionality. Finally, the *release day* includes the activities of *integration* and *testing*. The *Stabilize* phase has the goal to *finalize the implementation* along with *integrating subsystems* if necessary. As this phase can contain additional programming and development, the activities are very similar to the activities in the *Productionize* phase. The only additional activity concerns *documentation wrap-up*. Iterations should result in a working piece of functionality at user level.

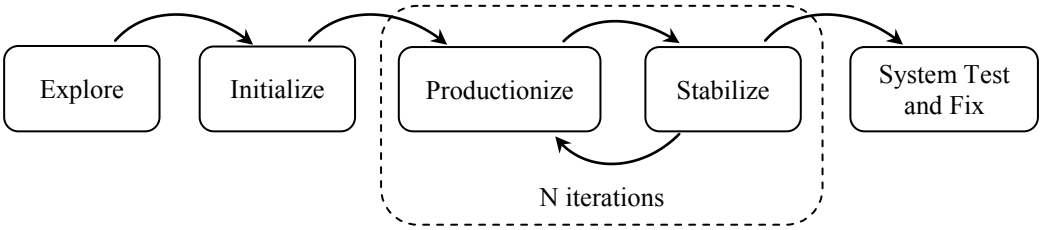


Figure 1 - Mobile-D process

Finally, *System Test and Fix* phase aims to detect if the produced system correctly implements the customer defined functionality. It also provides the project team feedback on the systems functionality and the defect information for the last fixing iteration of the Mobile-D process. This last iteration is not obligatory, but when fixing is needed it consists of the same activities as other implementation iterations already explained.

Mobile-D strongly suggests the usage of Test Driven Development (TDD) which is connected to all Mobile-D phases. The basics and the state of the art in TDD can be found in (Hammond and Umphress, 2012). The purpose of TDD is to give the developers confidence that the code they produce works, as well as to guide the design of the code towards an easily testable structure. Additionally, the refactoring practice is also based on TDD to ensure that changes made to the code do not break any functionality (Abrahamsson et al., 2005a).

In order to systematically observe the development process and to identify the artifacts created during it, we developed a prototype application, namely KnowLedge, for Android and Windows Phone target platform. The application intends to enable users learn and/or share knowledge in an interactive and social manner. Among others, the basic usage included functional requirements like browsing through categories to find existing knowledge on a topic or placing a request for a new explanation/instructions/tutorial, sharing knowledge in groups etc.

The overall system architecture comprises service oriented architecture, mobile application, remote database and usage of the global positioning system. In addition, as it can be seen in Figure 2, the mobile application architecture is also intended to be multi-layered with three

distinct but connected layers. The internal cohesion (see (Miller, 2008)) of the presented modules should be high, and at the same time the external coupling should be kept low.

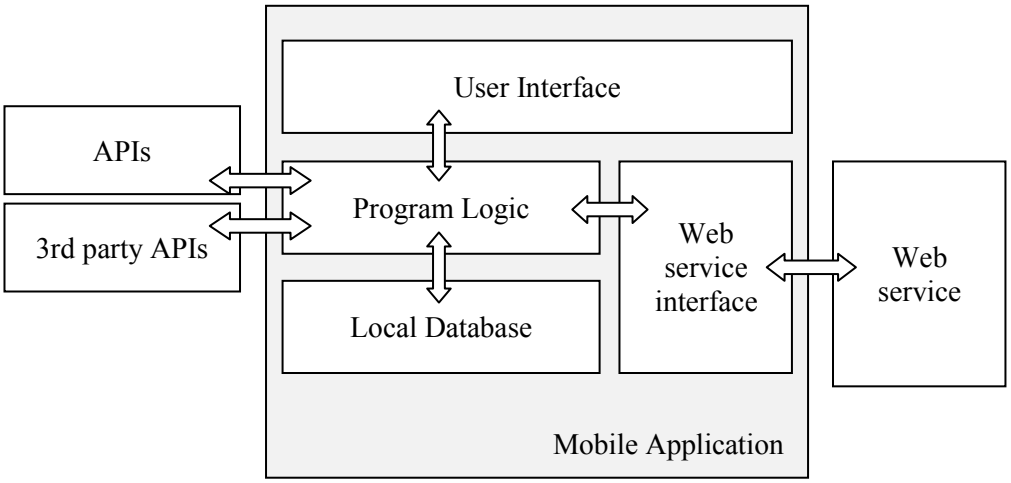


Figure 2 - Mobile application architecture

The Mobile-D process with its clear technical specification was well documented and easy to follow and the overall development process took less time than initially planned. A few screenshots of the created application are visible in Figure 3.

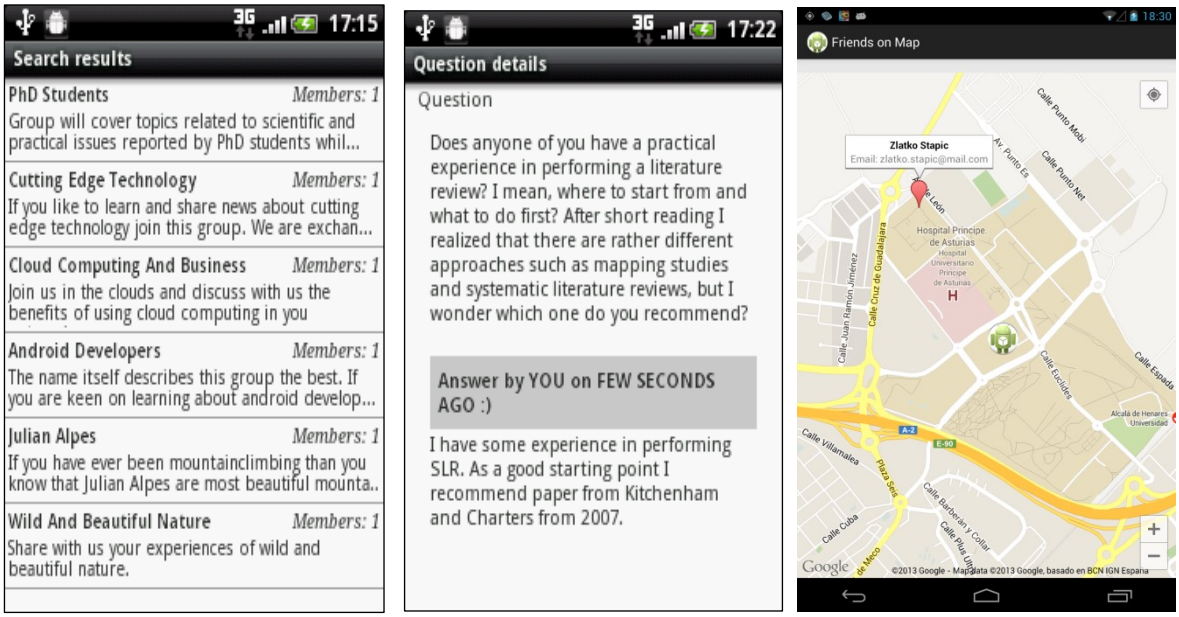


Figure 3 - Application screenshots

In the case of Windows Phone application development, the whole process was performed again, but as the structure of the created artifacts was the same as in the Android case, the focus in this development process was put on identifying the means and possibilities of reusing the existing artifacts. Although we expected some similarities among the artifacts, the results were surprising: we found that many of the artifacts were completely or partially reusable. Even though we experienced some WP platform specific issues and some testing

issues, the duration of the development process in WP case was 30 working days shorter when compared to the planned duration and 16 working days (18.4%) shorter if compared to the Android development case.

4. Identification of artifacts

As there are many definitions of artifact (e.g. from Hilpinen (2011) or from Parker (2011)), we have adopted the definition from Conradi (2004) who says that artifact is "*any piece of software (i.e. models/descriptions/code) developed and used during software development and maintenance*". As the goal of this research was to analyze only the structural and semantic aspects of the sets of artifacts, we performed an analysis only from the semantic concept view, while other possible views, such as procedural concept view or pragmatic concept view are not covered by it. Thus, we only observed the artifacts and their connection to the activities and tasks as it can be seen in Figure 4.

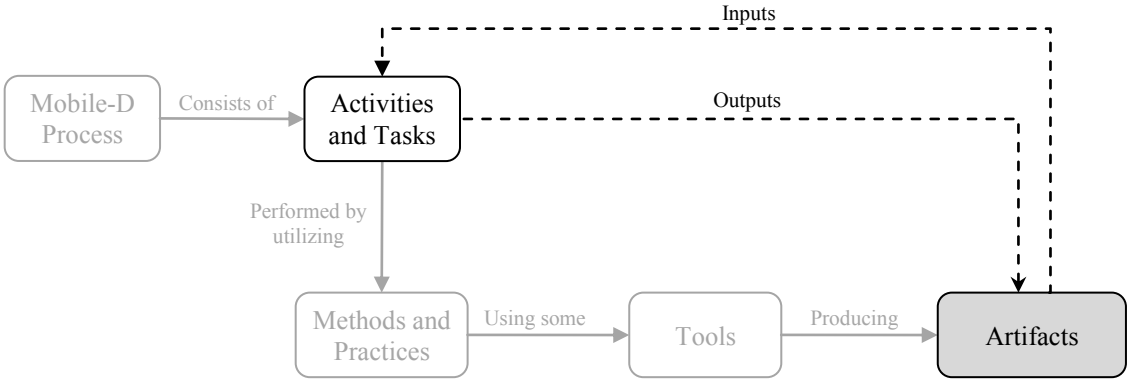


Figure 4 - Focusing semantic of artifacts and their origin

We performed the artifacts analysis in two steps. Firstly, we analyzed the Mobile-D process library (Abrahamsson et al., 2005a) and identified the documents and other platform-independent deliverables at a high level of abstraction. Secondly, as the approach of identifying and grouping the artifacts only according to the phases of the origin would not be a good way, and as during the implementation phase we collected the additional data on the artifacts, we systematized and described all identified artifacts for both target platforms using the template presented in Table 3.

Table 3 - Template for describing the identified artifacts

Artifact name	Type	Description	Phases inputs and outputs									
			I		II		III		IV		V	
			Input	Output	Input	Output	Input	Output	Input	Output	Input	Output

Thus, from the conceptual point of view, we created a solid basis for identifying not only the documents that had been created, but also other artifacts that might be hard to identify if the project was performed outside the laboratory.

Table 4 shows a part of the list of the identified artifacts, along with their initial classification, description and connection with the Mobile-D phases. We used standard CRU notation for denoting the artifacts that were created (C), used/read (R) and updated (U).

Table 4 – Part of list of identified artifacts in development process for Android

Artifact name	Type	Description	Phases inputs and outputs									
			I		II		III		IV		V	
			Input	Output	Input	Output	Input	Output	Input	Output	Input	Output
Mobile-D process library	Document	Process library describing the Mobile-D methodology in detail. Used as methodology guidelines in every phase. (Abrahamsson et al., 2005a)	R		R		R		R		R	
Product proposal	Document	Generated before the development process. Describes the initial and general idea on the product.	R									
Project plan	Document	Contains all information on project including definition of customer group, scope, planned activities and their duration, plans on documentation etc. Aligned with agile practices, this document is also updated during the iterations.		C	R	U	R	U				
…	…	…										

The identification process resulted in total of 60 different artifacts for Android development process and 61 artifacts for Windows Phone development process. The union of these two sets resulted in total of 71 identified artifacts that we grouped in 12 groups according to their type.

In the cross-platform analysis we found that 50 artifacts (70.42% of all identified artifacts) are common to both development cases. Additionally, many of these common artifacts are platform independent as being products of methodological approach. In total, 20 out of 50 identified common artifacts (40.00%) had been created or obtained only once, as these were identical in both development processes. On the other hand, there are 13 artifacts (26.00%) that could be partially reused while performing the development process for the second or any other target platform. Finally, we recognized 17 artifacts (34.00% of all common artifacts) with a very low level of possible reuse. They were classified as ones that should be developed from scratch for every target platform. A preview of results of the cross-platform analysis can be seen in Table 5. All other artifacts were classified as platform dependent artifacts, which also have some reusable semantic or syntactic parts like sequencing, iterations, algorithms etc.

Table 5 – A part of list of common artifacts in Android in WP case

Artifact name	Identical	Partially reused	Different
Mobile-D process library	X		
Product proposal	X		
Initial requirements document	X		
Project plan		X	
Project plan checklist		X	
Project plan checklist template	X		
Project plan Gantt chart	X		
Measurement plan		X	
Architecture line description			X
…			

In total, 33 artifacts (66.00% of the common artifacts) are completely or partially reusable which encouraged us and provided a solid basis and motivation for semantic analysis that followed.

5. The ontology for methodological interoperability

The term "ontology" was taken from philosophy, but its use and meaning in computer science got a new and adapted perspective. As there is no consensus on the definition of ontology, in the context of this research we consider ontology as *an explicit formal conceptualization of a shared understanding of the domain of interest which includes vocabulary of terms for describing the domain elements, semantics in order to define the relationships of the domain elements and pragmatics in order to define possible usages of these elements.*

5.1. Positioning the ontology development approach

Noy and McGuinness (2001) gave a comprehensive overview of possible reasons for the use of ontologies. The authors recognized the usage of ontologies to: *share common understanding of the structure of information among people or software agents*, *enable reuse of domain knowledge, make domain assumptions explicit*, *separate domain knowledge from the operational knowledge, and analyze domain knowledge.* Additionally, the ontologies are used as intermediary mechanisms in *intermediary-based approach* to achieve *semantic interoperability* (Park and Ram, 2004) which is of special interest in this research. Such interoperability, according to Paulheim and Probst (2010), can be performed on different levels, and subsequently they define integration on the data source level, integration on the business logic level and integration on the user interface level, but surprisingly, interoperability on the methodological level is rarely mentioned in literature.

Although there are different ontology types (see Lovrenčić (2007)), the ontology that is a subject of this research is classified as domain ontology. Domain ontology can be defined as a

network of domain model concepts (topics, knowledge elements) that defines the elements and the semantic relationships between them (Brusilovsky et al., 2005). The usage of domain ontologies is suitable to describe all content regarding chosen development methodology and approach. Similarly, there are several papers that give an extensive overview of ontology design methodologies, such as (Dahlem, 2011), (Lovrenčić, 2007) and (Kabilan, 2007). However, due to its characteristics of simplicity, focus on results and iterative approach, we can call the methodology proposed by Noy and McGuinness (2001), namely Ontology Development 101, as an *agile ontology development methodology*. Hence we found it as the most suitable for our research process and we used it in defining our ontology. Finally, there are various possibilities of using different ontology development tools and ontology development languages. The research performed by Khondoker and Mueller (2010) showed that by far the most widely used tool is Protégé. As Protégé is aligned with the OD101 methodology, and widely used from scientists and practitioners in (among others) fields of Information Systems Development and Knowledge Management, we decided to use it in our research as well. Subsequently, as Protégé works with two ontology representation languages, Frames and OWL, we discussed both and selected OWL2 DL as the most appropriate language in our case.

5.2. Developing the ontologies

The ontology development process was performed in three steps. First we developed the Android case ontology and then the Windows Phone case ontology. Finally, we merged these two into a single ontology definition.

The list of terms specific to our domain of interest was incrementally created during the whole ontology development process. The final list includes terms that are the base of our ontology: *phase, activity, task, artifact, task input, task output, artifact type, artifact origin, artifact usage, artifacts hierarchy, reusability, artifact similarity*. In the process of class and hierarchy definition, we followed the advice from Uschold and Gruninger (1996) and used the *middle-out approach* by first defining more salient concepts and then making generalizations and specializations as needed. The approach resulted in total definition of 152 classes organized in 7 top level classes for Android, 153 classes similarly organized for Windows Phone and 213 classes in the final merged ontology. The top level classes of the merged ontology are presented in Figure 5.

In order to define knowledge on structure, semantics and usage of the ontology elements we defined 12 object properties for the two specific ontologies and 14 object properties for the final merged ontology. These properties are: *consistsOf, createsArtifact, hasArtifactOrigin, hasArtifactType, includesArtifact, hasReusabilityLevel, isCreatedByTask, isPartOfArtifact,*

isPerformedIn, isSimilarToArtifact, isUpdatedByTask, isUsedByTask, updatesArtifact, usesArtifact.

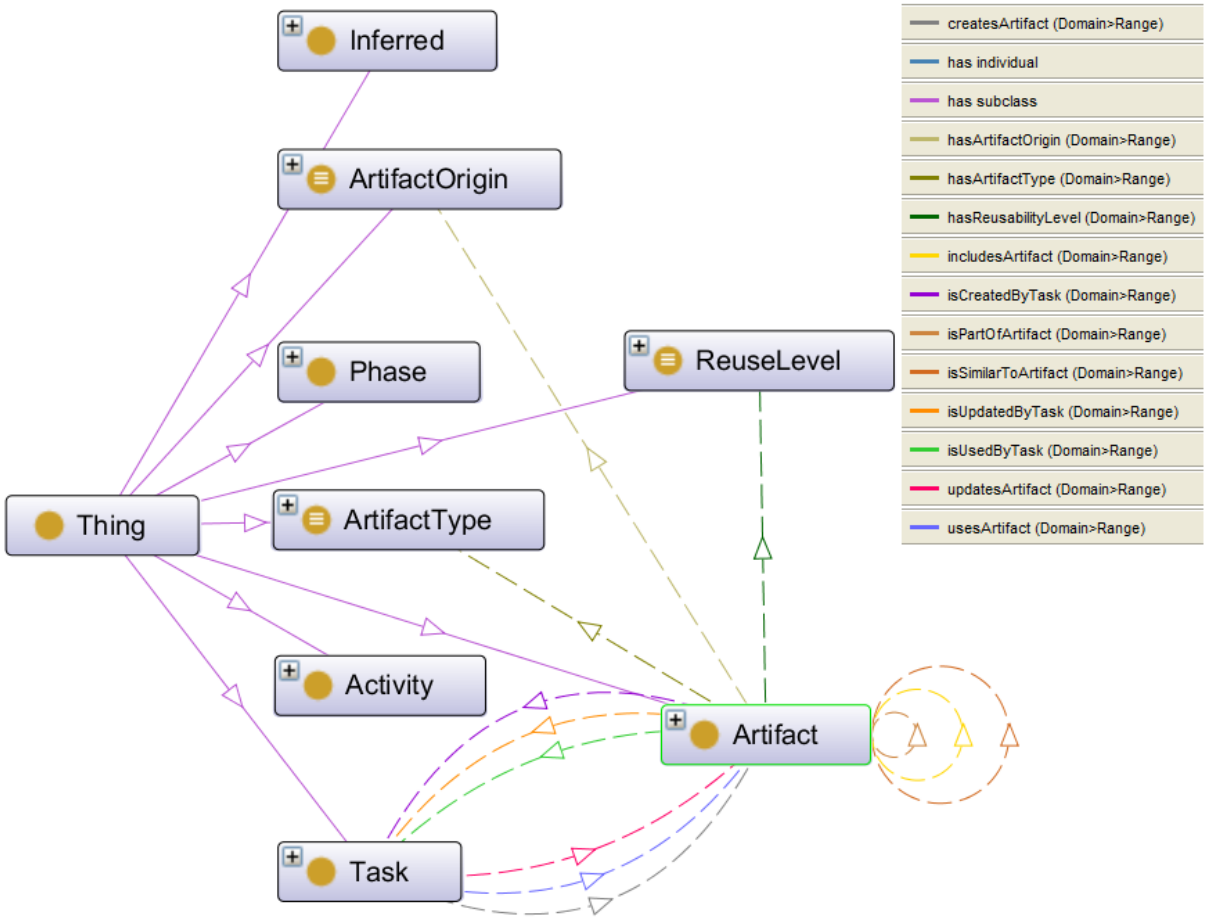


Figure 5 - Top level classes in final ontology

The figure describing a part of the final ontology shows that *Artifact* is finally connected with *Task*, *ArtifactOrigin*, *ArtifactType* and *ReuseLevel*. Among these relationships, the relationship with *Task* is the strongest as it is defined by three properties (each of them having their inversed property). Although existing, the relationships among other top level classes are not presented in this figure in order to maintain the focus on Artifact class only.

Connecting the instances of classes with the defined properties we had to follow OWL 2 DL restrictions, rules and syntax. Additionally, OWL DL is based on Open World Assumption (OWA) logic paradigm, and the OWA paradigm assumes that we cannot conclude that something does not exist until it is explicitly stated that it does not exist. For example, in order to completely define the *methodological artifacts* we had to use *closure axioms* and to explicitly state that methodological artifacts were not created and not modified in our development process but just used. The example of such description is given in Code 1.

```
SubClass Of:
Artifact
hasArtifactOrigin only MethodologicalArtifact
hasArtifactOrigin some MethodologicalArtifact
hasArtifactType only Document
hasArtifactType some Document
isUsedByTask only Task
isUsedByTask some Task
not (isCreatedByTask some Task)
not (isUpdatedByTask some Task)
```

Code 1 - Sufficient class description in OWA paradigm

During the development of the Android case ontology we put the focus on the ontology development process guided by selected development methodology and we developed the ontology from scratch. In the second iteration we put the focus on reuse of the existing ontology which proved its validity and flexibility and thus it validated the conceptual model that is the base for our ontologies targeting single platforms.

In the development of unique ontological description, the focus was put on the ontology merging, updating and evaluation. Most of the merging process was done automatically (see Figure 6). After merging the two ontologies, we had no redundancy to deal with, and had no problems in updating the ontology with a new conceptualization. This proves that the ontology is both reusable and extendable.

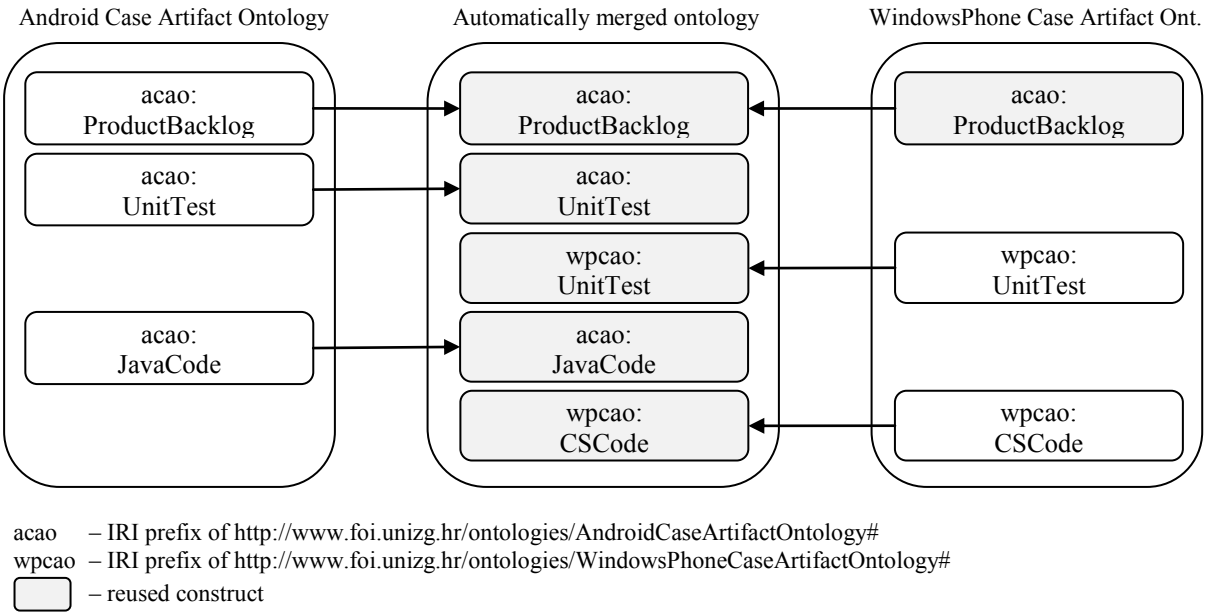


Figure 6 - Example of automatically merged ontology

The basic terms defined for the Android Case ontology were reused in Windows Phone Case ontology and thus are included in the final ontology as well. As we aimed to enhance the

ontology with the conceptualization on artifact reusability, we had to introduce a couple of new important terms (*reusability* and *artifact similarity)*.

The created ontology comprises 213 classes, 14 object properties and 2213 axioms defined in ALCRIF DL expression sub-language. The ontology in native OWL/XML format can be downloaded from http://barok.foi.hr/~zstapic/ont/mcao.owl, while full OWLDoc ontology documentation can be accessed and analyzed at http://barok.foi.hr/~zstapic/ont/mcao/doc/.

5.3. Evaluating the final ontology

In order to verify and validate our ontology, throughout the whole development process lifecycle, we have performed the following seven verification and validation mechanisms:

1. Methodologically driven ontology development process
2. Followed recommendation and advices from other authors
3. Using reasoning tools to verify the ontology in each iteration
4. Using W3C OWL validating tool
5. Using the Ontology evaluation plug-in
6. Using DL queries to obtain information via inference on ontology knowledge
7. Checking the results by domain experts

The first five evaluating mechanisms are connected with ontology verification and are used to lower the risks of making any syntactical and basic semantic errors throughout the whole ontology development process.

The last two mechanisms are connected with ontology validation. These two mechanisms have been used at the end of the development process to check if the created ontology represents the domain knowledge in semantically correct way. Queries were created and executed upon the final ontology in order to answer all competency questions related to application development targeting any single platform and reusability semantics defined at the beginning of the ontology creation process. For example, in order to obtain all reusable artifacts that were used, created or updated during the *Iteration Planning* task we can use a query like this:

```
Artifact
 and ((isUsedByTask some IterationPlanningTask)
   or (isCreatedByTask some IterationPlanningTask)
   or (isUpdatedByTask some IterationPlanningTask))
 and (ReusableArtifacts)
```

Code 2 - Reusable artifacts by task

The query result:

```
AcceptanceTest, IterationBacklog, IterationsPlan, MeasurementPlan,
ProductBacklog, ProjectPlan, ProjectPlanChecklist, ProjectPlanGantChart,
StoryCard, StoryCardTemplate, TaskCard, TaskCardTemplate
```

The following query enumerates artifacts with specific type of *Document* that are *completely* or *partially reusable*.

```
Artifact
 and (hasArtifactType some Document)
 and ((hasReusabilityLevel some Completely)
  or (hasReusabilityLevel some Partially))
```

Code 3 - Reusable artifacts by their type

The query result:

```
InitialRequirementsDocument, MobileDProcessLibrary, ProductBacklog,
ProductProposal, ProjectPlan
```

All other queries were stated in a similar manner and results were analyzed by a domain expert. The use of evaluation mechanisms throughout the development process and positive validation are the proof of the quality and completeness of the ontology. This brings us to the final conclusion that the developed *Multi-platform Case Artifacts Ontology* represents a knowledge base that can be used in the development of information system aiming to guide development teams in achieving methodological interoperability by reusing artifacts created in the process of multi-platform mobile applications development.

6. Discussion and conclusion

Throughout the research we aimed to clearly point out at least five important aspects which should make the research process transparent and repeatable. We put special focus on the research *motivation*, *results*, *contributions*, *rigor* and *evaluation.* By research *motivation* we wanted to emphasize the reasons for performing the research activities. By *results* and *contribution* we aimed to systematize the obtained results and the contribution to knowledge. Discussing the *research rigor* we wanted to point out our approach and its main characteristics, and discussing the *evaluation* we wanted to underline the evaluation mechanisms that are used in order to verify and validate the used approach and the obtained results.

In this research several limitations can be identified. For example, the biggest challenges that we faced in the first research phase were: the execution of a complicated and time-consuming scientific method of Systematic Literature Review by a single researcher; the institutional

subscriptions to the available scientific sources are very poor in Croatia but somewhat better in Spain; the lack of information about the performed projects on development of mobile applications in development companies targeting two or more target platforms made us develop a prototype application in laboratory; the proposed ontology presents only the development of one application for two target platforms; and we covered only one development methodology supported by one development approach. All mentioned issues can be recognized as the limitations of this research, but we have to keep in mind that this research process had the main goal of proposing a new framework or approach that can be used in solving the mobile platform fragmentation problem.

Following the research goals defined at the beginning of the research process we identified methodologies that could be used for development of mobile applications; we implemented the chosen methodology and approach and created a mobile application targeting two target platforms; we identified and analyzed the artifacts that were created in this development process, and we created an ontological definition that describes the artifacts in accordance with Mobile-D methodology and from the reusability point of view.

According to the results obtained during the ontology evaluation and testing, we can conclude that such ontological description represents a solid basis that can be used in development of an information system aiming to guide development teams in achieving methodological interoperability by reusing artifacts created in the process of multi-platform mobile application development. Additionally, we proved that our ontological description is highly flexible and extensible. This allows us to update it with information on new *platform specific* or *platform independent* artifacts without the need of changing the underling infrastructure which is defined by the main class hierarchy elements, value partitions and properties. Finally, the model allows the creation of Description Logic queries which can be used to acquire direct or indirect information encoded in the ontology knowledge. We showed examples of such queries which, among others, aimed to reach the information regarding the competency questions stated at the beginning of the ontology development.

Therefore, we can conclude that **it is possible to create an ontological description of the elements of methodological interoperability containing structural and semantic aspects of sets of artifacts created in the development process of a mobile application for two or more target platforms**, which makes our H_1 hypothesis confirmed.

This research presents a comprehensive set of activities which resulted in a final product that is usable in its current state. However, by extending the contexts of using such ontology we can identify other possible research activities or even research directions that could be taken. In general, we recognize two main fields where this research sets the basis for future scientific and professional activities. Those fields are *Software Engineering* with particular focus on

mobile engineering and, secondly, *Knowledge Engineering* with particular focus on *ontology development*. The created ontology defines the basic infrastructure and elements in the proposed framework of methodological interoperability, which is stable for adding other platforms, but should be reanalyzed and redefined when it comes to using it for completely different methodologies. On the other hand, when talking about research activities in the field of software engineering, we have already mentioned the necessity of moving this research towards a new phase where a proper information system for guiding the artifacts reuse would be developed. The development of such a novel system is not a trivial task and it gives many research possibilities in domains of its design, functionality, relationships with the ontological knowledge base et cetera.

Although there are ontologies defined to provide interoperability at different levels of an application development process, this novel approach aims to define interoperability at, until now unexplored, methodological level. Semantic descriptions created and evaluated in this research proved that the proposed approach and the supporting framework represent a solid basis for performing additional research in this field. However, developing this ontology is only the first step in the chain of activities to be implemented in order to develop a semantically supported system for methodological interoperability.

References

Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., Koskela, J., Kyllönen, P., Salo, O., 2004. Mobile-D: an agile approach for mobile application development, in: Companion to the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages, and Applications, OOPSLA '04. ACM, New York, NY, USA, pp. 174–175.

Abrahamsson, P., Hanhineva, A., Hulkko, H., Jäälinoja, J., Komulainen, K., Korkala, M., Koskela, J., Kyllönen, P., Eporwei, O.T., 2005a. Agile Development of Embedded Systems: Mobile-D (Agile Deliverable No. D.2.3). ITEA.

Adobe Corporation, 2011. Adobe Announces Agreement to Acquire Nitobi, Creator of PhoneGap [WWW Document]. Adobecom - Press Releases. URL http://www.adobe.com/aboutadobe/pressroom/pressreleases/201110/AdobeAcquiresNitobi.html (accessed 18-May-12).

Agarwal, V., Goyal, S., Mittal, S., Mukherjea, S., 2009. MobiVine: a middleware layer to handle fragmentation of platform interfaces for mobile applications, in: Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware, Middleware '09. Springer-Verlag New York, Inc., New York, NY, USA, pp. 24:1–24:10.

Amanquah, N., Eporwei, O.T., 2009. Rapid application development for mobile terminals, in: 2nd International Conference on Adaptive Science & Technology (ICAST). Presented at the Technology (ICAST), Accra, Ghana, pp. 410–417.

Biolchini, J., Gomes Mian, P., Candida Cruz Natali, A., Horta Travassos, G., 2005. Systematic Review in Software Engineering (Technical report No. RT - ES 679 / 05). PESC, Rio de Janeiro.

Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M., 2007. Lessons from applying the systematic literature review process within the software engineering domain. J. Syst. Softw. 80, 571–583.

Brusilovsky, P., Sosnovsky, S., Yudelson, M., 2005. Ontology-based Framework for User Model Interoperability in Distributed Learning Environments, in: World Conference on ELearning, E-Learn 2005. AACE, pp. 2851–2855.

Conradi, R., 2004. Software engineering mini glossary [WWW Document]. URL http://www.idi.ntnu.no/grupper/su/publ/ese/se-defs.html (accessed 5-May-12).

Dahlem, N., 2011. OntoClippy: A User-Friendly Ontology Design and Creation Methodology. Int. J. Intell. Inf. Technol. 7, 15–32.

Hammond, S., Umphress, D., 2012. Test driven development. ACM Press, p. 158.

Hannay, J., Sjoberg, D., Dyba, T., 2007. A Systematic Review of Theory Use in Software Engineering Experiments. IEEE Trans. Softw. Eng. 33, 87–107.

Hilpinen, R., 2011. Artifact [WWW Document]. Stanf. Encycl. Philos. URL http://plato.stanford.edu/entries/artifact/ (accessed 5-May-12).

Hosbond, J.H., 2005. Mobile Systems Development: Challenges, Implications and Issues, in: Krogstie, J., Kautz, K., Allen, D. (Eds.), Mobile Information Systems II, IFIP International Federation for Information Processing. Springer Boston, pp. 279–286.

Jeong, Y.-J., Lee, J.-H., Shin, G.-S., 2008. Development Process of Mobile Application SW Based on Agile Methodology, in: Proceedings of 10th International Conference on Advanced Communication Technology, (ICACT 2008). IEEE, Gangwon-Do, pp. 362–366.

Kabilan, V., 2007. Ontology for information systems (04IS) design methodology: conceptualizing, designing and representing domain ontologies. Data- och systemvetenskap, Kungliga Tekniska högskolan, Kista.

Khondoker, R.M., Mueller, P., 2010. Comparing Ontology Development Tools Based on an Online Survey, in: Proceedings of the World Congress on Engineering. Presented at the WCE 2010, London.

Kitchenham, B., Charters, S., 2007. Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3 (Technical report No. EBSE-2007-01). Keele University and University of Durham.

La, H.J., Kim, S.D., 2009. A service-based approach to developing Android Mobile Internet Device (MID) applications. 2009 IEEE Int. Conf. Serv.-Oriented Comput. Appl. SOCA 00, 1–7.

Lovrenčić, S., 2007. Formalna ontologija sveučilišnih studija (Doctoral dissertation). University of Zagreb, Varazdin, Croatia.

Manjunatha, A., Ranabahu, A., Sheth, A., Thirunarayan, K., 2010. Power of Clouds in Your Pocket: An Efficient Approach for Cloud Mobile Hybrid Application Development, in: 2010 IEEE Second International Conference on Cloud Computing Technology and Science. pp. 496–503.

Mian, P., Conte, T., Natali, A., Biolchini, J., Travassos, G., 2005. A Systematic Review Process for Software Engineering, in: ESELAW '05: 2nd Experimental Software Engineering Latin American Workshop.

Miller, J., 2008. Cohesion And Coupling. MSDN Mag. - Microsoft J. Dev. 23.

Noy, N.F., McGuinness, D.L., 2001. Ontology Development 101: AGuide to Creating Your First Ontology (Technical report No. KSL-01-05; SMI-2001-0880), Stanford Knowledge Systems Laboratory and Stanfrod Medical Informatics Technical Report. Stanford University, Stanfrod.

Park, J., Ram, S., 2004. Information systems interoperability: What lies beneath? ACM Trans. Inf. Syst. 22, 595–632.

Parker, P.M., 2011. Definition of artifact [WWW Document]. Websters Online Dict. URL http://www.websters-online-dictionary.org/definitions/artifact (accessed 5-Jul-11).

Paulheim, H., Probst, F., 2010. Application integration on the user interface level: An ontology-based approach. DATA Knowl. Eng. 69, 1103–1116.

PhoneGap, 2011. Take the pain out of compiling mobile apps for multiple platforms [WWW Document]. PhoneGap Build. URL https://build.phonegap.com (accessed 27-Aug-11).

Rahimian, V., Ramsin, R., 2008. Designing an agile methodology for mobile software development: A hybrid method engineering approach, in: Proceedings of Second International Conference on Research Challenges in Information Science, RCIS (2008). IEEE, Marrakech, pp. 337–342.

Rhomobile, Inc., 2011. Smartphone Enterprise Application Integration, White paper [WWW Document]. URL http://tiny.cc/rhomobile (accessed 20-Aug-11).

Spataru, A.C., 2010. Agile Development Methods for Mobile Applications (PhD Thesis, University of Edinburgh). The University of Edinburgh, Edinburgh.

Uschold, M., Gruninger, M., 1996. Ontologies: Principles, methods and applications. Knowl. Eng. Rev. 11, 93–136.

RESUMEN EXTENDIDO

1. Introducción

El desarrollo de sistemas móviles es una tarea exigente que se diferencia del desarrollo tradicional en diversos aspectos importantes. Según Hosbond (2005), los dos principales desafíos a superar en el dominio del desarrollo de sistemas móviles son los desafíos relacionados con el *negocio* y los desafíos específicos del *desarrollo*. En esta investigación nos centraremos en los desafíos específicos del desarrollo, con especial atención al uso de metodologías del software, ya que, según autores como Rahimian and Ramsin (2008), Spataru (2010) or La and Kim (2009), se trata de uno de los primeros aspectos a abordar.

Las metodologías clásicas y ágiles para el desarrollo de software deben ser adaptadas al desarrollo de aplicaciones móviles ya que las actuales no cubren las necesidades específicas de este tipo de proyectos (La and Kim, 2009). Ha habido varios intentos de diferentes autores por crear nuevas metodologías con el objetivo de cubrir las lagunas existentes en el dominio de las aplicaciones móviles. Algunas de ellas son *Agile Risk-based Methodology* (Rahimian and Ramsin, 2008), *MASAM* (Jeong et al., 2008), y *Mobile-D* (Abrahamsson et al., 2004). En todo caso aun no existe ninguna investigación exhaustiva que responda a preguntas como cuales de las metodologías nuevas o existentes son apropiadas para el desarrollo de aplicaciones móviles.

Uno de los problemas más graves que debe afrontar la mencionada metodología es el de la *fragmentación*, que obliga a los desarrolladores de aplicaciones móviles a enfocarse únicamente en plataformas y versiones específicas (Manjunatha et al., 2010), cuando en realidad su principal interés es abarcar el mayor abanico posible de usuarios. Esta aproximación al desarrollo es poco deseable, lo que provoca que los equipos de desarrollo busquen diferentes soluciones de manos de la comunidad de profesionales y de la comunidad científica. Un primer tipo de solución es la que permite a los equipos de desarrollo utilizar un *lenguaje intermedio* o un *motor de transformación intermedio* para programar para distintas plataformas al mismo tiempo. Algunos de los proyectos más influyentes de este tipo son MobiCloud (Manjunatha et al., 2010), Rhodes (Rhomobile, Inc., 2011) y el generador de código de Amanquah & Eporwei (Amanquah and Eporwei, 2009). Estas propuestas tienen varias ventajas, pero también importantes inconvenientes como serían la dependencia en el esfuerzo invertido en el motor de transformación, en APIs específicas y en el dominio específico; la falta de control sobre el código fuente generado; así como otros problemas similares. El segundo tipo de solución sería la introducción de aplicaciones "adaptador"

nativas en todas las plataformas. Según Agarwal et al. (2009) esta es una de las técnicas disponibles para manejar la fragmentación. Como la estandarización de APIs en el mundo de los móviles resulta aun imposible, la única vía disponible para aplicarla es el uso de técnicas de programación en las que se hacen llamadas a las interfaces abstractas de módulos que han sido portados a todas las plataformas. Los casos más representativos de esta aproximación son MobiVine (Agarwal et al., 2009), PhoneGap (PhoneGap, 2011) ó Adobe AIR® (Adobe Corporation, 2011). Prácticamente todas los inconvenientes que se mencionaron en el caso de los lenguajes o motores de transformación siguen estando presentes en este segundo tipo de solución. Finalmente, una tercera solución sería el uso de tecnologías web para desarrollar aplicaciones web multi-plataforma, pero esta aproximación está fuera del alcance de esta investigación ya que difiere en muchos aspectos de las otras dos soluciones propuestas (e igualmente tiene sus propias desventajas).

Por tanto, se centrá en la propuesta de soluciones para mejorar la interoperabilidad metodológica entre equipos que trabajan en la misma aplicación pero en diferentes entornos de desarrollo nativos. La investigación responderá a las siguientes preguntas: (1) qué metodologías y aproximaciones[46] al desarrollo pueden utilizarse en el desarrollo de aplicaciones móviles multi-plataforma; (2) qué artefactos (entradas y salidas de las distintas fases del desarrollo que se han definido metodológica y metódicamente) surgen durante el desarrollo de aplicaciones móviles, (3) hasta qué punto existen similitudes entre estos artefactos y (4) si es posible describir estos artefactos ontológicamente y crear una base para el desarrollo de un sistema que soporte la interoperabilidad metodológica. En consecuencia, el objetivo principal es la descripción ontológica de los artefactos que surgen en el proceso de desarrollo de aplicaciones móviles para dos o más plataformas gestionado metodológicamente, y la creación de una base para un proceso más eficiente e interoperable de desarrollo multi-plataforma para móviles.

En ese sentido, la investigación pretende probar la siguiente hipótesis: H_1 – *Es posible crear una descripción ontológica de los elementos de interoperabilidad metodológica que contenga aspectos estructurales y semánticos de los conjuntos de artefactos creados en el proceso de desarrollo de una aplicación móvil para dos o más plataformas.*

Los capítulos de este resumen están organizados de acuerdo con las preguntas de investigación planteadas. En el segundo capítulo se aborda la revisión sistemática de las metodologías de desarrollo de aplicaciones para móviles; el tercer capítulo muestra los resultados de la implementación de una metodología en dos plataformas distintas y en el

[46] Nota del traductor: De las dos posibles traducciones al castellano del término 'development approach', que son 'aproximación al/del desarrollo' y 'enfoque de desarrollo', en este texto se ha optado por la primera.

cuarto capítulo identificamos y comparamos los artefactos que surgieron en dicho proceso; en el quinto capítulo se crea la definición ontológica de los artefactos y en el último capítulo se discuten los resultados y se alcanzan las conclusiones.

2. Metodologías de desarrollo para aplicaciones móviles: Una revisión sistemática

"La revisión sistemática de literatura es una forma de evaluar e interpretar todas las investigaciones relevantes sobre una pregunta de investigación, un tema o un fenómeno de interés. La revisión sistemática busca una evaluación justa de un tema de investigación mediante el uso de una metodología rigurosa, auditable y de confianza" (Kitchenham and Charters, 2007). Como el método de SLR (systematic literature review) es novedoso en el campo de la ingeniería del software, en primer lugar hemos analizado las mejores prácticas a la hora de aplicar un método tan exigente en horas de dedicación, y tan exhaustivo. Las guias proporcionadas por Kitchenham and Charters (2007) serán seguidas y discutidas a la vez que se incluyen las recomendaciones y hallazgos de otros autores influyentes en este ámbito. Hemos puesto especial interés en el problema de la aplicación de este método por estudiantes de doctorado. Los resultados de esta fase de investigación se muestran a continuación de la ejecución del método SLR en las siguientes secciones.

2.1. Realización del SLR

Tras realizar una breve revisión preliminar de las metodologías existentes hemos concluido que el desarrollo para móviles difiere del desarrollo estandar, que la aproximación ágil es ampliamente utilizada en metodologías para desarrollo móvil, y que ninguna de las metodologías observadas es aplicable sin esfuerzos extra para hacer el proceso más detallado y adecuado a los entornos de desarrollo y los requisitos específicos de las aplicaciones para móvil. Esto indica que es necesario realizar una investigación meticulosa e imparcial con el objetivo de conseguir una visión general de las posibles metodlogías que podrían utilizarse a la hora de desarrollar aplicaciones para dispositivos móviles.

Además, se ha realizado una investigación preliminar adicional para identificar las revisiones sistematicas de literatura ya existentes en el ámbito de las metodologías de desarrollo de aplicaciones para móvil. Se han realizado busquedas en las bases de datos IEEExplore, ACM Digital library, INSPEC, CiteSeerX y GoogleScholar con la siguiente consulta: *("literature review" OR SLR) AND (mobile development)*. Según la información disponible en las citadas bases de datos, no existen revisiones sistemáticas de literatura que cubran el tema de las metodologías de desarrollo de aplicaciones móviles, lo cual hace que dicha revisión sea aun más necesaria.

En un intento de resolver los problemas descubiertos en este análisis, la revisión sistemática intentará contestar a las siguientes preguntas de investigación:

RQ1 – ¿Qué metodologías de desarrollo y aproximaciones aparecen en la literatura, que hayan sido definidas teóricamente o aplicadas en la práctica para el desarrollo de aplicaciones para móvil?

RQ2 – ¿Son las metodologías y aproximaciones aplicables en el desarrollo multi-plataforma de aplicaciones móviles?

El protocolo de revisión se definió de acuerdo a las instrucciones dadas en (Kitchenham and Charters, 2007) y usando la plantilla para el protocolo propuesta en (Biolchini et al., 2005) y explicada en más detalle por (Mian et al., 2005). La principal cadena de búsqueda usada en esta investigación fue *(mobile AND ("software development" OR "system development" OR "application development" OR "program development") AND (methodology OR method OR approach OR framework OR process OR procedure OR model))*, la cual se ejecutó sobre las fuentes de datos (revistas y actas de congresos) relevantes y disponibles en el ámbito de la ingeniería del software identificadas por los expertos de este campo Brereton et al. (2007), Hannay et al. (2007) and by Kitchenham and Charters (2007).

La revisión de la literatura se llevó a cabo en diferentes fases incluyendo la identificación, la aplicación de criterios de inclusión y exclusión, y la evaluación de la calidad. Finalmente 49 estudios de 6761 se identificaron como relevantes de cara a la extracción y síntesis de datos.

Tal y como se presenta en las tablas 1 y 2, se identificaron un total de 22 metodologías de desarrollo y 7 aproximaciones al desarrollo, tanto de nueva creación como empleadas, y por lo tanto aplicables en el desarrollo de aplicaciones móviles multi-plataforma.

Tabla 1 – Metodologías y aproximaciones desarrolladas

Nombre	Tipo
Agile Methodology for Mobile Software Development	M
Agile Solo	M
Agile usability process	M
DEAL	M
Integrated Product Development Process for Mobile Software	M
Inter-combined Model	M
MASAM methodology	M
Methodology for Building Enterprise-Wide Mobile Applications	M
MicroApp visual approach	M
Mobile Application Development Methodology	M
Mobile-D	M
New media application prototyping	M
Systems Development Methodology	M
ViP (Virtual Platform)	M
Composite Application Software Development Process Framework	A
MobiLine	A

Tipo: M - Metodología, A - Aproximación

Tabla 2 - Metodologías y aproximaciones empleadas

Nombre	Tipo
Design Science	M
Dynamic Channel Model	M
Extreme Programming	M
Kanban	A
Mobile-D	M
Mobile Engineering (MobE)	M
Mobile RAD	M
Rapid Application Development	M
Scrum	M
Model Driven Development	A
Model Driven Product Lines	A
Software Product Lines	A
Test Driven Development	A

Tipo: M - Metodología, A - Aproximación

Una única metodología se trata en más de un estudio, mientras que el resto de metodologías se presenten en un único estudio de los identificados. Adicionalmente, y tal y como se esperaba, las metodologías y aproximaciones son bastante nuevas. Solo 4 estudios tienen más de 5 años de antigüedad, mientras que el resto de estudios datan de los últimos 5 años. La puntuación global de evaluación de la calidad obtenida en el estudio es de 2,735 sobre 5 (68,38%) con una desviación estándar de 0,903. Esto indica que se trata de un estudio con una calidad relativamente baja y una alta desviación en calidad.

Por otro lado, más autores informan sobre la utilización de una metodología y/o aproximación. Un total de 9 metodologías y 4 aproximaciones se presentaron como empleadas en los distintos estudios. El hecho más relevante es que solo una metodología (Mobile-D) de nueva creación de las identificadas ha sido utilizada. Esta metodología ha sido empleada en cinco estudios diferentes, mientras que no se ha halló evidencia en los estudios sobre uso del resto de nuevas metodologías y nuevas aproximaciones.

2.2. Elección de la metodología de desarrollo

Como la asunción básica de esta investigación es que la interoperabilidad metodológica es independiente de la plataforma y de la metodología (i.e. que puede conseguirse con cualquier metodología ontológicamente definida), podría elegirse cualquiera de las 22 metodologías identificadas. Para evitar una decisión aleatoria, el criterio empleado para elegir la metodología de desarrollo fue la *novedad y aplicabilidad de la metodología de desarrollo según la literatura*. El análisis cruzado de los resultados del SLR muestra que *Mobile-D* es la única metodología creada específicamente para el desarrollo de aplicaciones móviles que se utiliza en la práctica. Además, se realizó una búsqueda para identificar otras fuentes de literatura gris publicadas por los creadores de la metodología, y se encontró que esta metodología está documentada a fondo y en detalle contando con diferentes publicaciones de la cuáles al más relevante es (Abrahamsson et al., 2005a)

3. Implementación de la metodología

El proceso Mobile-D (Figura 1) incluye cinco fases que se ejecutan en orden parcialmente incremental. El objetivo de la primera fase, llamada *Explore*, es preparar las bases para el futuro desarrollo. La fase *Initialize* debe describir y preparar todos los componentes de la aplicación a la vez que se predicen posibles problemas críticos en el proyecto. La fase Initialize a veces se denomina también fase de iteración cero (*0-iteration*) ya que además de la puesta en marcha del proyecto (*project set-up*), se incluyen fases adicionales como el *planning day*, *working day* y *release day*, que también se utilizan en la fase *Productionize*. La idea clave de la iteración 0 es asegurar la funcionalidad del entorno de desarrollo a través de la implementación de algunas de las características más representativas o a través de la creación de un prototipo. Las fases *Productionize* y *Stabilize* se ejecutan iterativamente y en orden para desarrollar el resto de características del producto. Cada iteración comienza con el *planning day* en la fase *Productionize*. La primera actividad es el taller post-iteración en donde se busca mejorar el proceso de desarrollo para que se ajuste mejor a las necesidades del equipo de desarrollo actual. Las siguientes fases que se ejecutan durante el *planning day* son el análisis de requisitos, la planificación de la iteración y la generación de pruebas de aceptación. En el *working day* se trabaja en la implementación mediante desarrollo guiado por pruebas, programación por pares, integración continua y refactorización. Este día finaliza con una tarea en la que se informa al cliente de la nueva funcionalidad desarrollada. Finalmente el *release day* incluye las actividades de integración y pruebas. La fase *Stabilize* tiene como meta finalizar la implementación, incluyendo la integración de subsistemas si fuese necesario. Como esta fase puede contener programación y desarrollo adicional, sus actividades son muy similares a las de la fase *Productionize*. La única actividad adicional es la relacionada con el empaquetado de la documentación. Cada iteración completa debería resultar en una pieza de software funcional a nivel de usuario.

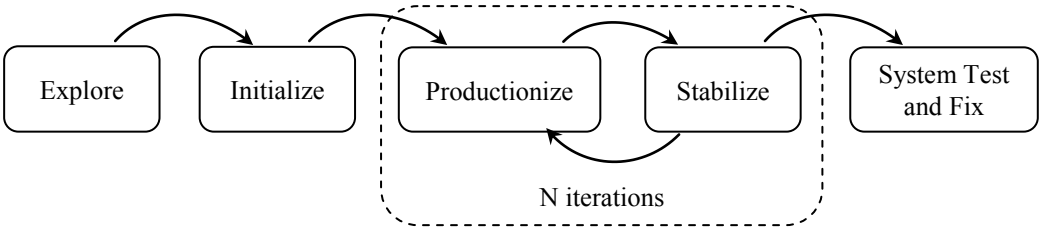


Figura 1 – Proceso Mobile-D

Finalmente, la fase de *System Test and Fix* tiene la función de detectar si el producto implementa correctamente su funcionalidad tal y como la ha definido el cliente. También proporciona *feedback* al equipo de desarrollo sobre la funcionalidad del sistema e información sobre errores que resultará necesaria para la última iteración de reparaciones en el proceso

Mobile-D. Esta última iteración no es obligatoria salvo que el sistema necesite alguna reparación, en cuyo caso se realizarán las mismas actividades que en otras iteraciones de implementación que ya se han explicado.

El proceso Mobile-D recomienda encarecidamente la aplicación del desarrollo guiado por pruebas ó TDD (del inglés, Test Driven Development), ya que está conectado con todas las fases Mobile-D. Las bases y el estado del arte del TDD pueden encontrarse en (Hammond and Umphress, 2012). El propósito del TDD es dar a los desarrolladores confianza en que el código que producen funcionará correctamente, así como guiar el diseño de la estructura del código para que resulte sencillo de probar. Además la práctica de la refactorización también se basa en el TDD, para asegurar que los cambios realizados sobre el código no producen errores en la funcionalidad ya programada (Abrahamsson et al., 2005a).

Para poder observar sistemáticamente el proceso de desarrollo y para identificar los artefactos que se crean en él, hemos desarrollado una aplicación prototipo llamada KnowLedge, para las plataformas Android y Windows Phone. La aplicación pretende dar la posibilidad a sus usuarios de aprender y compartir conocimiento de una forma interactiva y social. Entre otros, el uso básico de la aplicación incluye requisitos funcionales como explorar categorías para encontrar una fuente de conocimiento sobre un tema particular; enviar peticiones para obtener nuevas explicaciones, instrucciones o tutoriales; compartir el conocimiento dentro de un grupo de usuarios; etc.

La arquitectura general del sistema incluye una parte basada en servicios, la aplicación móvil, la base de datos remota y el uso de un sistema de posicionamiento global. Además, como puede observarse en la Figura 2, la arquitectura de la aplicación móvil también pretende ser multicapa con tres capas distintas interconectadas. La cohesión interna (ver (Miller, 2008)) de los módulos debe ser alta, mientras que el acoplamiento externo debe mantenerse bajo.

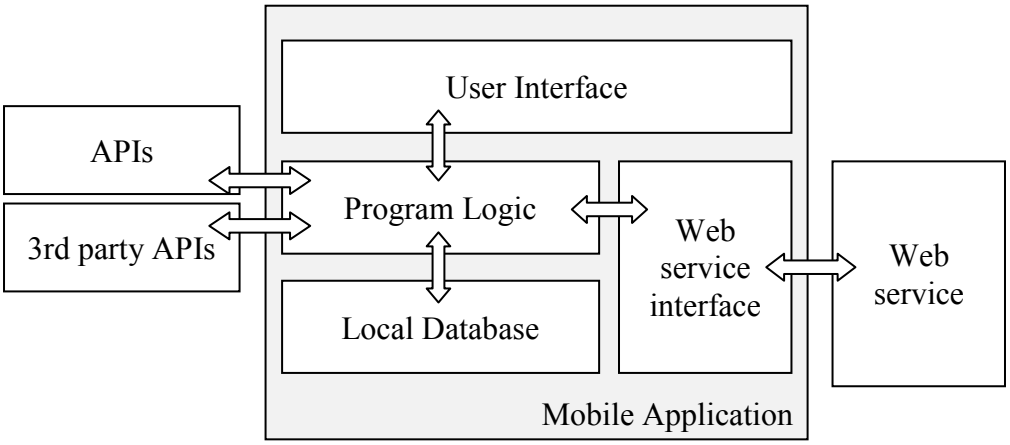


Figure 2 - Arquitectura de la aplicación móvil

El proceso Mobile-D resultó sencillo de seguir gracias a su clara especificación técnica y a su buena documentación, y el proceso de desarrollo completo fue más rápido de lo inicialmente planeado. En la Figura 3 se muestran algunas capturas de pantalla de la aplicación.

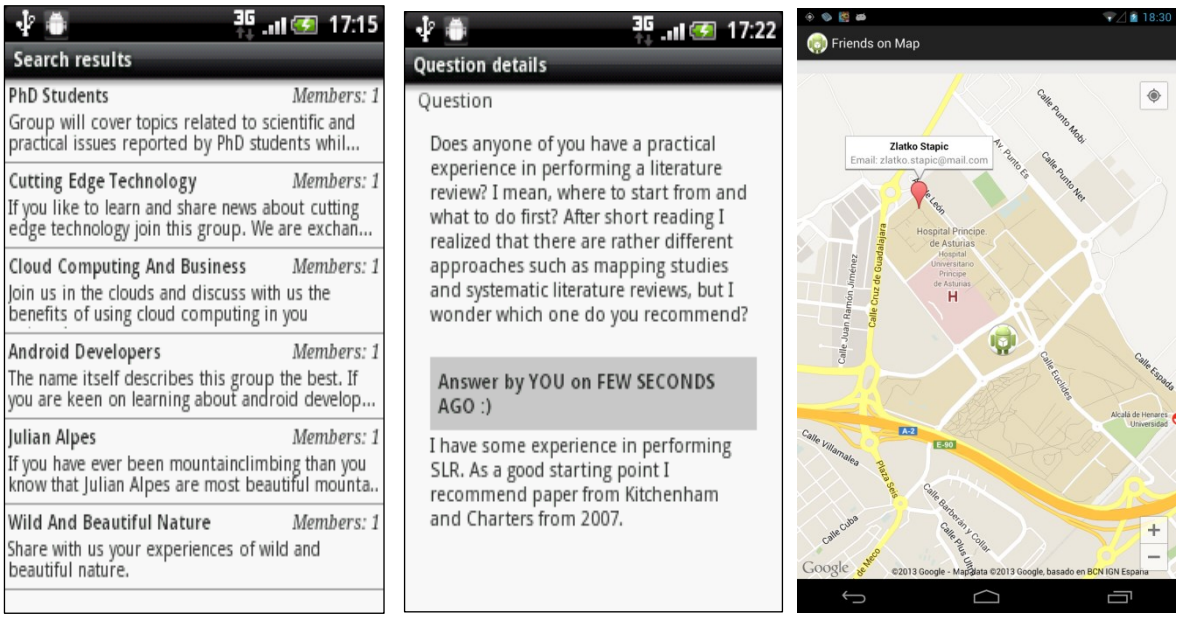


Figura 3 – Capturas de pantalla de la aplicación KnowLedge

En el caso del desarrollo de la aplicación de Windows Phone el proceso se aplicó de nuevo por completo, pero como la estructura de los artefactos generados era la misma que en el caso de Android, en este caso nos centramos en la identificación del significado (semántica) de los distintos artefactos y en sus posibilidades de reutilización. Aunque esperábamos algunas similitudes entre los artefactos, los resultados fueron sorprendentes: gran parte de los artefactos resultaron completamente o parcialmente reutilizables. A pesar de que se experimentaron algunos problemas específicos para la plataforma Windows Phone y algunos problemas durante las pruebas, la duración del proceso de desarrollo en esta plataforma se acortó en 30 días de trabajo comparándolo con la duración planificada, y en 16 días de trabajo (18.4%) comparándolo con el caso de Android.

4. Identificación de artefactos

Al haber numerosas definiciones de artefacto (p. ej. de Hilpinen (2011) ó de Parker (2011)), hemos adoptado la definición de Conradi (2004) que dice que un artefacto es "*cualquier pieza de software (i.e. modelos/descripciones) desarrollada y utilizada durante el desarrollo y el mantenimiento de software*". Dado que la meta de esta investigación es analizar únicamente los aspectos estructurales y semánticos del conjunto de artefactos, hemos realizado un análisis desde el punto de vista del concepto semántico, mientras que otros posibles puntos de vista como el del concepto procedural o el del concepto pragmático no se han cubierto. Por tanto,

únicamente hemos observado los artefactos y sus conexiones con las actividades y tareas, tal y como se muestra en la Figura 4.

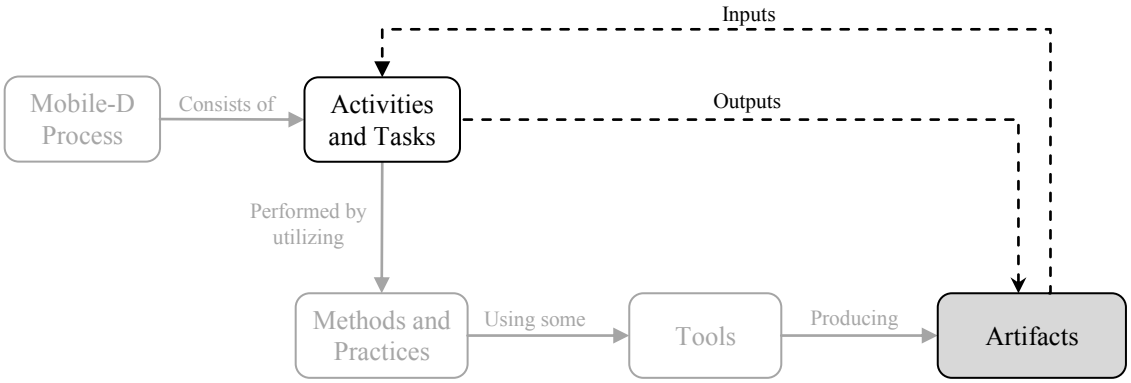


Figura 4 - Semántica de los artefactos y en su origen

El análisis de artefactos se realizó en dos pasos. En primer lugar analizamos la librería del proceso Mobile-D (Abrahamsson et al., 2005a) e identificamos a alto nivel los documentos y el resto de entregables independientes de la plataforma. En segundo lugar, como la aproximación de identificar y agrupar los artefactos en función de las fases de origen no sería adecuada, y como durante la fase de implementación se recolectaron datos adicionales de los artefactos, decidimos sistematizar y describir todos los artefactos identificados para ambas plataformas usando la plantilla presentada en la Tabla 3.

Tabla 3 – Plantilla para describer los artefactos identificados

Nombre del artefacto	**Tipo**	**Descripción**	**Input y output de cada fase**									
			I		**II**		**III**		**IV**		**V**	
			Input	**Output**	**Input**	**Output**	**Input**	**Output**	**Input**	**Output**	**Input**	**Output**

Por tanto, desde un punto de vista conceptual, hemos creado una base solida para identificar no solo los documentos que han sido creados, si no también otros artefactos que podrían ser difíciles de identificar si el proyecto fuese desarrollado fuera del laboratorio.

La Tabla 4 muestra parte de la lista de artefactos identificados, junto con su clasificación inicial, descripción y conexión con las fases del proceso Mobile-D. Hemos usado la notación estándar CRU para denotar los artefactos creados (C), usados/leídos (R) y actualizados (U).

Tabla 4 – Lista parcial de artefactos identificados en el proceso de desarrollo para Android

Nombre del artefacto	Tipo	Descripción	Input y output de cada fase									
			I		II		III		IV		V	
			Input	Output	Input	Output	Input	Output	Input	Output	Input	Output
Mobile-D process library	Documento	La Mobile-D process library describe la metodología Mobile-D en detalle. Se utiliza como guía de la metodología en todas sus fases. (Abrahamsson et al., 2005a)	R		R		R		R		R	
Product proposal	Documento	Se genera antes de comenzar el proceso de desarrollo. Describe la idea general inicial del producto.	R									
Project plan	Documento	Contiene toda la información sobre el proyecto incluyendo la definición del cliente, alcance, actividades planificadas y su duración, planes de documentación, etc. Si se utiliza junto con prácticas ágiles, este documento también se actualiza durante las iteraciones.		C	R	U	R	U				
…	…	…										

El proceso de identificación resultó en un total de 60 artefactos diferentes para el proceso de desarrollo en Android y 61 artefactos para el de Windows Phone. La unión de estos dos conjuntos resulto en un total de 71 artefactos identificados que se pueden agrupar en 12 grupos según su tipo.

En el análisis multi-plataforma encontramos que 50 artefactos (70,42% de todos los identificados) son comunes a ambos desarrollos. Además muchos de los artefactos comunes son independientes de la plataforma ya que se trata de productos propios de la aproximación metodológica. En total, 20 de los 50 artefactos comunes identificados (40,00%) han sido creados y obtenidos una única vez, ya que eran idénticos en ambos procesos de desarrollo. Por otra parte, hay 13 artefactos (26,00%) que solo pueden ser reutilizados parcialmente mientras se realiza el desarrollo en la segunda o posteriores plataformas. Finalmente, reconocimos 17 artefactos (34,00% de todos los artefactos comunes) con un nivel muy bajo de posible reutilización. Estos han sido identificados como los artefactos que deben ser desarrollados desde el principio para cada plataforma de destino. La pre-visualización de resultados del análisis multi-plataforma puede encontrarse en la Tabla 5. El resto de artefactos han sido clasificados como artefactos dependientes de la plataforma, los cuales tienen algunas partes semánticas o sintácticas reutilizables como las secuencias de instrucciones, iteraciones, algoritmos, etc.

Tabla 5 – Lista parcial de artefactos comunes en Android y Windows Phone

Nombre del artefacto	Idéntico	Parcialmente Reutilizable	Diferente
Mobile-D process library	X		
Product proposal	X		
Initial requirements document	X		
Project plan		X	
Project plan checklist		X	
Project plan checklist template	X		
Project plan Gantt chart	X		
Measurement plan		X	
Architecture line description			X
…			

En total 33 artefactos (66.00% de los artefactos comunes) son completamente o parcialmente reutilizables. Esto nos lleva a concluir que los resultados obtenidos motivan y proporcionan una sólida base para el análisis semántico que se muestra a continuación.

5. La ontología para la interoperabilidad metodológica

El término "ontología" ha sido tomado del ámbito de la filosofía, pero su uso y significado en informática toma una perspectiva nueva y adaptada. Como no existe consenso sobre la definición de ontología, en el contexto de esta investigación consideramos que una ontología es *una conceptualización explícita y formal de un conocimiento común compartido en un dominio de interés que incluye un vocabulario de términos para describir los elementos del dominio, la semántica para definir las relaciones de los elementos del dominio, y la pragmática para definir los posibles usos de estos elementos*.

5.1. Enfoque para el desarrollo de la ontología

Noy and McGuinness (2001) ofrecen un revisión exhaustiva de las posibles razones para usar ontologías. Estos autores reconocen el uso de ontologías para: *compartir conocimiento común sobre la estructura de cierta información entre personas o agentes software, permitir la reutilización de conocimiento del dominio, hacer explícitas las suposiciones de un dominio, separar el conocimiento de un dominio del conocimiento operacional, y analizar el conocimiento de un dominio*. Adicionalmente, las ontologías se emplean como mecanismos intermediadores en el enfoque centrado en la intermediación (*intermediary-based approach*) para conseguir *interoperabilidad semántica* (Park and Ram, 2004) que es de especial interés en esta investigación. Tal interoperabilidad, de acuerdo a Paulheim and Probst (2010), puede realizarse a diferentes niveles que después definen la integración en el nivel de datos, la integración en el nivel de lógica de negocio y la integración en el nivel de interfaz de usuario, pero sorprendentemente, la interoperabilidad a nivel metodológico rara vez se menciona en la literatura.

Aunque existen diferentes tipos de ontologías (veáse Lovrenčić (2007)), la ontología objeto de esta investigación se clasifica como una ontología del dominio. Una ontología del dominio puede definirse como una red de conceptos del modelo del dominio (temas, elementos del conocimiento) que definen los elementos y las relaciones semánticas entre ellos (Brusilovsky et al., 2005). El uso de ontologías del dominio es adecuado para describir todo tipo de contenido relacionado con la metodología de desarrollo y la aproximación al desarrollo. Igualmente, existen diversos estudios que proporcionan una visión general sobre metodologías para el diseño de ontologías, como son (Dahlem, 2011), (Lovrenčić, 2007) y (Kabilan, 2007). Sin embargo, por sus características en lo relativo a simplicidad, enfoque en los resultados y aproximación iterativa, podemos citar la metodología propuesta por Noy and McGuinness (2001), es decir Ontology Development 101 (OD101), como una *metodología ágil de desarrollo de ontologías*. Esta es la razón por la que la consideramos como la más conveniente para nuestro proceso de desarrollo y por la que la emplearemos para definir nuestra ontología. Finalmente, existen distintas posibilidades en cuanto a las herramientas de desarrollo de ontologías y los lenguajes de desarrollo de ontologías. La investigación realizada por Khondoker y Mueller (2010) mostró que con diferencia Protégé es la herramienta más empleada. Como Protégé está alineada con la metodología OD101 y es ampliamente utilizada por científicos y profesionales en campos como el Desarrollo de Sistemas de Información y la Gestión del Conocimiento entre otros, decidimos usarla en nuestra investigación también. Con posterioridad y dado que Protégé trabaja con dos lenguajes de representación, Frames y OWL, comparamos ambos y seleccionamos OWL2 DL como el más apropiado en nuestro caso.

5.2. Desarrollo de las ontologías

El proceso de desarrollo de la ontología se llevó a cabo en tres pasos. Primero se desarrolló una ontología para el caso de Android, después se desarrolló una metodología para el caso de Windows Phone y finalmente se fusionaron ambas ontologías en una única.

La lista de términos que aparecen en el dominio de interés se creó incrementalmente durante todo el proceso de desarrollo. La lista final de términos que son la base de la ontología incluye: *phase, activity, task, artifact, task input, task output, artifact type, artifact origin, artifact usage, artifacts hierarchy, reusability, artifact similarity*. Para el proceso de definición de clases y su jerarquía, seguimos las directrices de Uschold y Gruninger (1996) y usamos una aproximación del centro hacía afuera (middle-out) definiendo en primer lugar los conceptos más destacados para luego realizar las generalizaciones y especializaciones necesarias. Esta aproximación produjo un total de 152 clases de definición organizadas en 7 clases de alto nivel para Android; 153 clases igualmente organizadas para Windows Phone y

213 clases en la ontología fusionada final. Los artefactos de alto nivel de la ontología final se presentan en la figura 5.

Para definir el conocimiento de la estructura, semántica y uso de los elementos de la ontología se definieron 12 propiedades de objeto para las dos ontologías específicas y 14 propiedades de objeto para la ontología fusionada final. Estas propiedades son: *consistsOf, createsArtifact, hasArtifactOrigin, hasArtifactType, includesArtifact, hasReusabilityLevel, isCreatedByTask, isPartOfArtifact, isPerformedIn, isSimilarToArtifact, isUpdatedByTask, isUsedByTask, updatesArtifact, usesArtifact*.

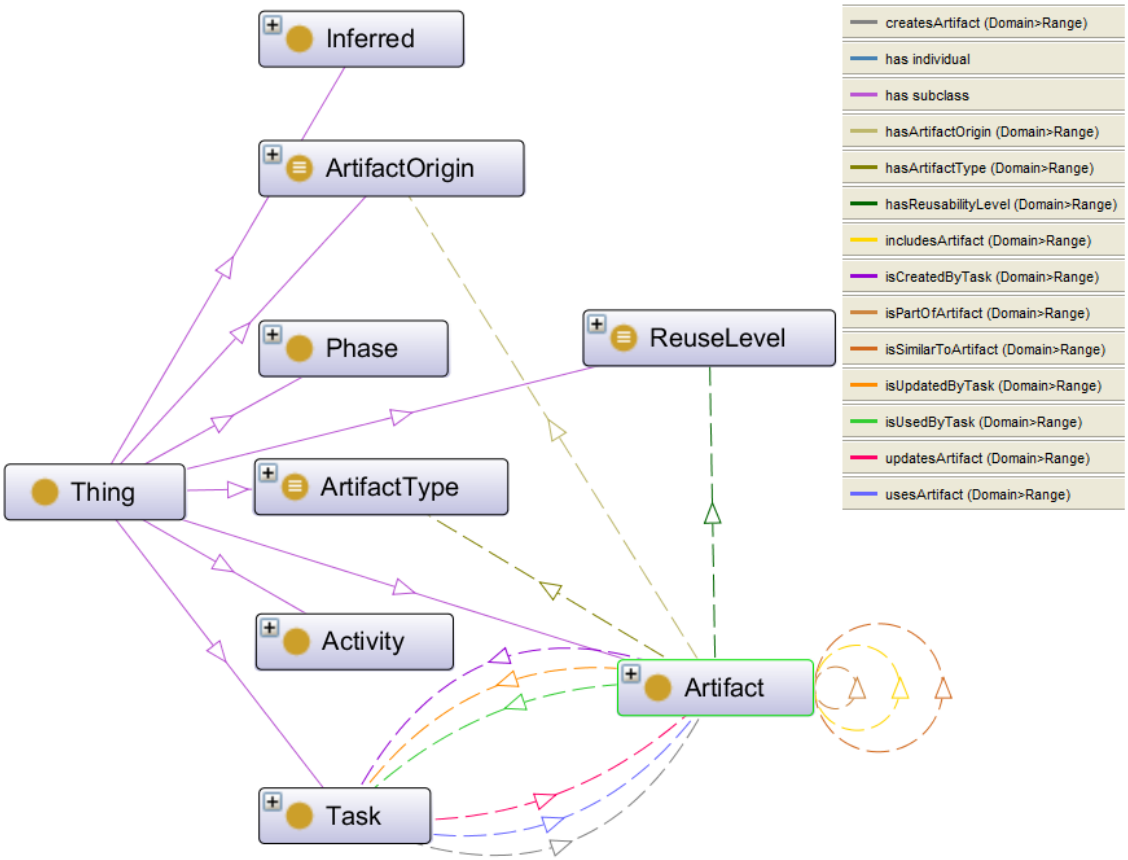


Figura 5 – Artefactos de alto nivel de la ontología

La figura que describe parte de la ontología final muestra que *Artifact* está finalmente conectado con *Task*, *ArtifactOrigin*, *ArtifactType* y *ReuseLevel*. De entre estas relaciones, la relación con *Task* es la más fuerte dado que está definida con tres propiedades (cada una de las cuales tiene la propiedad invertida). Aunque existen, las relaciones entre el resto de clases de alto nivel no se presentan en la figura con el fin de que esta se centre en los artefactos únicamente.

Para conectar las instancias de las clases con las propiedades definidas se siguieron las restricciones, reglas y sintaxis de OWL 2 DL. Adicionalmente, OWL DL se basa en paradigma lógico Open World Assumption (OWA). Este paradigma asume que no se puede

concluir que algo no existe hasta que se afirma explícitamente que no existe. Por ejemplo, para definir completamente los artefactos metodológicos tuvimos que usar axiomas de clausura y declarar explícitamente que tales artefactos no se crean ni se modifican durante el proceso de desarrollo. Simplemente se utilizan. Un ejemplo de dicha descripción se muestra en el fragmento de código 1.

```
SubClass Of:
Artifact
hasArtifactOrigin only MethodologicalArtifact
hasArtifactOrigin some MethodologicalArtifact
hasArtifactType only Document
hasArtifactType some Document
isUsedByTask only Task
isUsedByTask some Task
not (isCreatedByTask some Task)
not (isUpdatedByTask some Task)
```

Código 1 - Descripción de clase suficiente en el paradigma OWA

Durante el desarrollo de la ontología para el caso de Android nos centramos en el proceso de desarrollo guiado por la metodología de desarrollo seleccionada desarrollando la ontología desde cero. En la segunda interacción nos centramos en reutilizar la ontología existente probando su validez y flexibilidad. Esto validó por lo tanto el modelo conceptual que es la base de las ontologías dirigidas hacía una única plataforma.

Durante el desarrollo de la descripción ontológica unificada nos centramos en la fusión, actualización y evaluación. La mayoría del proceso de fusión se realizó automáticamente (veáse Figura 6). Tras la fusión de las dos ontologías, no hubo redundancia de la que ocuparse al igual que tampoco problemas al actualizar la ontología con nuevos conceptos. Esto prueba que la metodología es la vez reusable y extensible.

Los términos básicos definidos para la ontología del caso de Android se reutilizaron en la ontología para el caso de Windows Phone y por lo tanto se incluyen también en la ontología final. Como pretendíamos mejorar nuestra metodología con la conceptualización referente a la reutilización de artefactos, tuvimos que introducir un par de términos nuevos importantes (*reusability* y *artifact similarity*).

La ontología creada consta de 213 clases, 14 propiedades de objeto y 2213 axiomas definidos en el sub-lenguaje de expresión ALCRIF DL. La ontología en el formato nativo OWL/XML puede descargarse de http://barok.foi.hr/~zstapic/ont/mcao.owl, mientras que la documentación completa OWLDoc de la ontología puede encontrase en http://barok.foi.hr/~zstapic/ont/mcao/doc/.

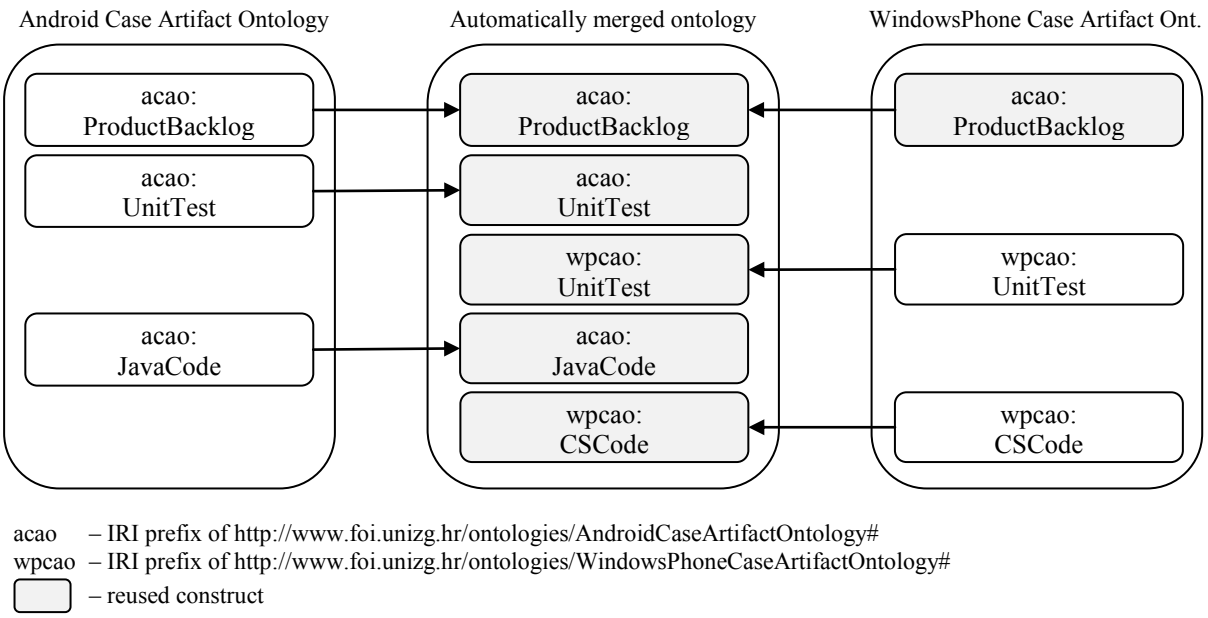


Figura 6 – Ejemplo de fusión automatica de ontologías

5.3. Evaluación de la ontología final

Para verificar y validar la ontología, durante el ciclo de vida del proceso de desarrollo, se han llevado a cabo los siete mecanismos siguientes de verificación y validación:

1. Proceso de desarrollo de la ontología dirigido metodológicamente
2. Seguir las recomendaciones y consejos de otros autores
3. Uso de herramientas de razonamiento para verificar la ontología en cada iteración
4. Uso de la herramienta de validación para W3C OWL
5. Uso del plug-in de evaluación de ontologías
6. Uso de consultas DL para obtener información vía inferencia sobre el conocimiento de la ontología
7. Comprobación de los resultados por expertos en el dominio

Los primeros cinco mecanismos de evaluación están relacionados con la verificación de la ontología y se emplearon para reducir el riesgo de cometer errores sintácticos y semánticos durante todo el proceso de desarrollo de la ontología.

Los últimos dos mecanismos están relacionados con la validación de la ontología. Estos dos mecanismos se emplearon al final del proceso de desarrollo para comprobar si la ontología representa el domino de conocimiento de una forma semánticamente correcta. Las consultas se crearon y ejecutaron sobre la ontología para responder las cuestiones relativas al desarrollo para una única plataforma y a la reusabilidad semántica definidas al principio del proceso de creación de la ontología. Por ejemplo, para obtener todos los artefactos reutilizables que se usaron, crearon o actualizaron durante la tarea *Iteration Planning*, se puede usar una consulta como la siguiente:

```
Artifact
 and ((isUsedByTask some IterationPlanningTask)
   or (isCreatedByTask some IterationPlanningTask)
   or (isUpdatedByTask some IterationPlanningTask))
 and (ReusableArtifacts)
```

Código 2 – Artefactos reutilizables por tarea

El resultado de la consulta:

```
AcceptanceTest, IterationBacklog, IterationsPlan, MeasurementPlan,
ProductBacklog, ProjectPlan, ProjectPlanChecklist, ProjectPlanGantChart,
StoryCard, StoryCardTemplate, TaskCard, TaskCardTemplate
```

La siguiente consulta enumera los artefactos del tipo específico *Document* que son *completa* o *parcialmente reutilizables*.

```
Artifact
 and (hasArtifactType some Document)
 and ((hasReusabilityLevel some Completely)
  or (hasReusabilityLevel some Partially))
```

Código 3 – Artefactos reutilizables por tipo

El resultado de la consulta es el siguiente:

```
InitialRequirementsDocument, MobileDProcessLibrary, ProductBacklog,
ProductProposal, ProjectPlan
```

El resto de consultas se declararon de forma similar y los resultados fueron analizados por un experto en el dominio. El uso de mecanismos de evaluación a lo largo del proceso de desarrollo junto con la validación positiva son prueba de la calidad y completitud de la ontología. Esto nos lleva a la conclusión final que es que *la ontología de artefactos del caso multi-plataforma* representa una base de conocimiento que puede ser empleada en el desarrollo de un sistema de información con el objetivo de guiar a los equipos de desarrollo para conseguir interoperabilidad metodológica mediante la reutilización de los artefactos que se creen en el proceso de desarrollo de aplicaciones móviles multi-plataforma.

6. Discusión y conclusiones

A lo largo de esta investigación hemos señalado al menos cinco aspectos importantes que deben hacer que el proceso de investigación sea transparente y repetible. Hemos puesto especial énfasis en la *motivación, resultados, contribuciones, rigor* y *evaluación* de la investigación. En cuanto a la *motivación* hemos querido destacar las razones para realizar la investigación. En lo relativo a *resultados* y *contribución* hemos tenido como objetivo sistematizar los resultados obtenidos y la contribución al conocimiento. En la discusión sobre el *rigor de la investigación* hemos querido señalar nuestro enfoque y sus principales

características, y en la discusión sobre la *evaluación* queríamos hacer especial hincapié en los mecanismos de evaluación empleados para verificar y validar el método empleado y los resultados obtenidos.

En esta investigación se pueden identificar varias limitaciones. Entre ellas se mencionan las siguientes: (1) El mayor reto al que se hizo frente en la primera fase de la investigación fue que la realización de la revisión sistemática de la literatura fue llevada a cabo por un único investigador resultando complicada y consumiendo mucho tiempo. (2) Las suscripciones institucionales a las fuentes de datos científicas disponibles son muy pobres en Croacia y algo mejores en España. (3) La falta de información sobre proyectos desarrollados para el desarrollo de aplicaciones móviles en compañías de desarrollo para dos o más plataformas nos obligó a desarrollar una aplicación prototipo de laboratorio. (4) La ontología presenta solo el desarrollo para dos plataformas objetivo. Y (5), solo se ha cubierto una metodología de desarrollo y una aproximación al desarrollo. Todos los problemas mencionados pueden reconocerse como limitaciones de esta investigación, pero debemos tener en cuenta que el objetivo principal de la investigación es proponer un nuevo marco o aproximación que pueda emplearse para afrontar el problema de la fragmentación en plataformas móviles.

Siguiendo los objetivos de investigación definidos al principio del proceso, hemos identificado las metodologías que se podrían emplear para el desarrollo de aplicaciones móviles; hemos implementado la metodología y la aproximación seleccionadas y hemos creado una aplicación móvil para dos plataformas; hemos identificado y analizado los artefactos creados durante el proceso, y hemos creado una definición ontológica que describe los artefactos conforme a la metodología Mobile-D desde el punto de vista de la reutilización.

De acuerdo a los resultados obtenidos durante la evaluación y prueba de la ontología, podemos concluir que la representación ontológica representa una base sólida que puede ser empleada en el desarrollo de un sistema de información que tenga el objetivo de guiar a los equipos de desarrollo a que consigan interoperabilidad metodológica reutilizando los artefactos creados en el proceso de desarrollo de aplicaciones móviles multi-plataforma. Además,

Además, hemos probado que la descripción ontológica es altamente flexible y extensible, lo que permite actualizarla con información sobre nuevos artefactos, dependientes o independientes de la plataforma, sin necesidad de cambiar la infraestructura subyacente dada por la jerarquía principal de clases y las particiones de valor o propiedades definidas. Finalmente, el modelo permite la creación de consultas en Lógica Descriptiva (Description Logic) que pueden emplearse para obtener información codificada en el conocimiento de la ontología directa o indirectamente. Hemos mostrado ejemplos de tales consultas destinadas,

entre otras cosas, a obtener información sobre las cuestiones de competencia declaradas al principio del desarrollo de la ontología.

Por lo tanto, podemos concluir que **es posible crear una descripción ontológica de los elementos de interoperabilidad metodológica que contenga aspectos estructurales y semánticos de los conjuntos de artefactos creados en el proceso de desarrollo de una aplicación móvil para dos o más plataformas**, lo que confirma nuestra hipótesis H_1.

Esta investigación presenta un amplio conjunto de actividades que han dado lugar a un producto final que se puede utilizar en su estado actual. Sin embargo, mediante la ampliación de los contextos de uso de dicha ontología podemos identificar otras actividades de investigación posibles o incluso líneas de investigación que podrían adoptarse. En general, reconocemos dos campos principales en los que esta investigación sienta las bases para futuras actividades científicas y profesionales. Esos campos son la *Ingeniería de Software* con especial énfasis en la *Ingeniería Móvil* y, en segundo lugar, la Ingeniería del Conocimiento, con especial énfasis en el *desarrollo de ontologías*. La ontología creada define la infraestructura básica y los elementos del framework propuesto para la interoperabilidad metodológica, que es estable para añadir nuevas plataformas, pero que debe reanalizarse y redefinirse cuando se trate de utilizarlo para metodologías completamente distintas. Por otro lado, cuando se habla de las actividades de investigación en el campo de la ingeniería de software, ya hemos mencionado la necesidad de trasladar la investigación a una nueva fase en la que se desarrollará un sistema de información adecuado para guiar en la reutilización de artefactos. El desarrollo de un sistema tan novedoso no es una tarea trivial y da muchas posibilidades de investigación en el ámbito de su diseño, su funcionalidad, y su relación con la base de conocimiento ontológico entre otras cosas.

Aunque existen ontologías definidas para proporcionar interoperabilidad a diferentes niveles del proceso de desarrollo de aplicaciones, este nuevo enfoque tiene por objetivo definir la interoperabilidad al, hasta ahora inexplorado, nivel metodológico. Las descripciones semánticas creadas y evaluadas en esta investigación prueban que el enfoque y el framework que los sustenta representan una base sólida para llevar a cabo más investigación en este ámbito. Sin embargo, el desarrollo de esta ontología es sólo el primer paso en la cadena de actividades que se deberían implementar a fin de desarrollar un sistema de apoyo semántico para la interoperabilidad metodológica.

Referencias

Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., Koskela, J., Kyllönen, P., Salo, O., 2004. Mobile-D: an agile approach for mobile application development, in: Companion to the 19th Annual ACM SIGPLAN Conference on

Object-oriented Programming Systems, Languages, and Applications, OOPSLA '04. ACM, New York, NY, USA, pp. 174–175.

Abrahamsson, P., Hanhineva, A., Hulkko, H., Jäälinoja, J., Komulainen, K., Korkala, M., Koskela, J., Kyllönen, P., Eporwei, O.T., 2005a. Agile Development of Embedded Systems: Mobile-D (Agile Deliverable No. D.2.3). ITEA.

Adobe Corporation, 2011. Adobe Announces Agreement to Acquire Nitobi, Creator of PhoneGap [WWW Document]. Adobecom - Press Releases. URL http://www.adobe.com/aboutadobe/pressroom/pressreleases/201110/AdobeAcquiresNitobi.html (accessed 18-May-12).

Agarwal, V., Goyal, S., Mittal, S., Mukherjea, S., 2009. MobiVine: a middleware layer to handle fragmentation of platform interfaces for mobile applications, in: Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware, Middleware '09. Springer-Verlag New York, Inc., New York, NY, USA, pp. 24:1–24:10.

Amanquah, N., Eporwei, O.T., 2009. Rapid application development for mobile terminals, in: 2nd International Conference on Adaptive Science & Technology (ICAST). Presented at the Technology (ICAST), Accra, Ghana, pp. 410–417.

Biolchini, J., Gomes Mian, P., Candida Cruz Natali, A., Horta Travassos, G., 2005. Systematic Review in Software Engineering (Technical report No. RT - ES 679 / 05). PESC, Rio de Janeiro.

Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M., 2007. Lessons from applying the systematic literature review process within the software engineering domain. J. Syst. Softw. 80, 571–583.

Brusilovsky, P., Sosnovsky, S., Yudelson, M., 2005. Ontology-based Framework for User Model Interoperability in Distributed Learning Environments, in: World Conference on ELearning, E-Learn 2005. AACE, pp. 2851–2855.

Conradi, R., 2004. Software engineering mini glossary [WWW Document]. URL http://www.idi.ntnu.no/grupper/su/publ/ese/se-defs.html (accessed 5-May-12).

Dahlem, N., 2011. OntoClippy: A User-Friendly Ontology Design and Creation Methodology. Int. J. Intell. Inf. Technol. 7, 15–32.

Hammond, S., Umphress, D., 2012. Test driven development. ACM Press, p. 158.

Hannay, J., Sjoberg, D., Dyba, T., 2007. A Systematic Review of Theory Use in Software Engineering Experiments. IEEE Trans. Softw. Eng. 33, 87–107.

Hilpinen, R., 2011. Artifact [WWW Document]. Stanf. Encycl. Philos. URL http://plato.stanford.edu/entries/artifact/ (accessed 5-May-12).

Hosbond, J.H., 2005. Mobile Systems Development: Challenges, Implications and Issues, in: Krogstie, J., Kautz, K., Allen, D. (Eds.), Mobile Information Systems II, IFIP International Federation for Information Processing. Springer Boston, pp. 279–286.

Jeong, Y.-J., Lee, J.-H., Shin, G.-S., 2008. Development Process of Mobile Application SW Based on Agile Methodology. Proceedings of 10th International Conference on Advanced Communication Technology. IEEE, Gangwon-Do, pp. 362–366.

Kabilan, V., 2007. Ontology for information systems (04IS) design methodology: conceptualizing, designing and representing domain ontologies. Data- och systemvetenskap, Kungliga Tekniska högskolan, Kista.

Khondoker, R.M., Mueller, P., 2010. Comparing Ontology Development Tools Based on an Online Survey, in: Proceedings of the World Congress on Engineering. Presented at the WCE 2010, London.

Kitchenham, B., Charters, S., 2007. Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3 (Technical report No. EBSE-2007-01). Keele University and University of Durham.

La, H.J., Kim, S.D., 2009. A service-based approach to developing Android Mobile Internet Device (MID) applications. 2009 IEEE Int. Conf. Serv.-Oriented Comput. Appl. SOCA 00, 1–7.

Lovrenčić, S., 2007. Formalna ontologija sveučilišnih studija (Doctoral dissertation). University of Zagreb, Varazdin, Croatia.

Manjunatha, A., Ranabahu, A., Sheth, A., Thirunarayan, K., 2010. Power of Clouds in Your Pocket: An Efficient Approach for Cloud Mobile Hybrid Application Development, in: 2010 IEEE Second International Conference on Cloud Computing Technology and Science. pp. 496–503.

Mian, P., Conte, T., Natali, A., Biolchini, J., Travassos, G., 2005. A Systematic Review Process for Software Engineering, in: ESELAW '05: 2nd Experimental Software Engineering Latin American Workshop.

Miller, J., 2008. Cohesion And Coupling. MSDN Mag. - Microsoft J. Dev. 23.

Noy, N.F., McGuinness, D.L., 2001. Ontology Development 101: AGuide to Creating Your First Ontology (Technical report No. KSL-01-05; SMI-2001-0880), Stanford Knowledge Systems Laboratory and Stanfrod Medical Informatics Technical Report. Stanford University, Stanfrod.

Park, J., Ram, S., 2004. Information systems interoperability: What lies beneath? ACM Trans. Inf. Syst. 22, 595–632.

Parker, P.M., 2011. Definition of artifact [WWW Document]. Websters Online Dict. URL http://www.websters-online-dictionary.org/definitions/artifact (accessed 5-Jul-11).

Paulheim, H., Probst, F., 2010. Application integration on the user interface level: An ontology-based approach. DATA Knowl. Eng. 69, 1103–1116.

PhoneGap, 2011. Take the pain out of compiling mobile apps for multiple platforms [WWW Document]. PhoneGap Build. URL https://build.phonegap.com (accessed 27-Aug-11).

Rahimian, V., Ramsin, R., 2008. Designing an agile methodology for mobile software development: A hybrid method engineering approach, in: Proceedings of Second International Conference on Research Challenges in Information Science, RCIS (2008). IEEE, Marrakech, pp. 337–342.

Rhomobile, Inc., 2011. Smartphone Enterprise Application Integration, White paper [WWW Document]. URL http://tiny.cc/rhomobile (accessed 20-Aug-11).

Spataru, A.C., 2010. Agile Development Methods for Mobile Applications (PhD Thesis, University of Edinburgh). The University of Edinburgh, Edinburgh.

Uschold, M., Gruninger, M., 1996. Ontologies: Principles, methods and applications. Knowl. Eng. Rev. 11, 93–136.

PROŠIRENI SAŽETAK

1. Uvod

Razvoj mobilnih aplikacija je izazovan zadatak koji se razlikuje od tradicionalnog razvoja u nekoliko važnih aspekata. Prema Hosbondu (2005), dva glavna skupa izazova trebala bi se rješavati u domeni razvoja mobilnih sustava. To su izazovi povezani s poslovanjem i izazovi specifični za razvoj. U ovom istraživanju usredotočit ćemo se na izazove specifične za razvoj, te ćemo posebnu pozornost obratiti na korištenje metodike razvoja. Neki autori, kao što su Rahimian i Ramsin (2008), Spataru (2010) ili La i Kim (2009), upravo korištenje metodike razvoja smatraju prioritetom pri razvoju mobilnih programskih proizvoda.

Postojeće klasične i agilne metodike razvoja softvera ne pokrivaju posebne zahtjeve razvoja mobilnih aplikacija te bi u tu svrhu trebale biti prilagođene (La i Kim, 2009). Postoji nekoliko pokušaja različitih autora koji su kreirali nove metodike kako bi uklonili nedostatke u domeni razvoja mobilnih aplikacija, a neke od njih su *Agilna metodika bazirana na uklanjanju riziku* (Rahimian i Ramsin, 2008), *MASAM* (Jeong et al., 2008) i *Mobile-D* (Abrahamsson et al., 2004). Ipak, nismo pronašli sveobuhvatno istraživanje koje odgovara na bitna pitanja kao na primjer koje postojeće ili nove metodike su pogodne za razvoj mobilnih aplikacija.

Povrh problema sa primjenom metodike, *problem fragmentacije* prisiljava programere mobilnih aplikacija da se u razvoju usredotoče samo na određene platforme i verzije (Manjunatha et al., 2010). Budući da su mobilne aplikacije prvenstveno usmjerene na širok spektar korisnika, takav pristup nije poželjan i razvojni timovi posežu za različitim rješenjima i pristupima koji su predloženi od strane stručne i znanstvene zajednice. Prvo, želimo spomenuti pristup koji omogućuje razvojnim timovima koristiti *posrednički jezik* ili pak *posrednički sustav za transformaciju kôda* kako bi pisali jedan kôd za nekoliko ciljanih platformi. Neki od najutjecajnijih projekata su *MobiCloud* (Manjunatha et al., 2010), *Rhodes* (Rhomobile, Inc., 2011) i *Amanquah & Eporwei generator koda* (Amanquah i Eporwei, 2009). Ovi pokušaji imaju nekoliko prednosti, ali također imaju i značajne nedostatke, kao što su ovisnost o naporima uloženim u sustav za transformaciju, korištenje specifičnih API-ja i primjena u samo specifičnim domenama, nedostatak kontrole nad generiranim izvornim kodom i slično. Drugo moguće rješenje problema fragmentacije moglo bi biti uvođenje *adapter aplikacija (prilagodnika)* kao izvornih aplikacija za svaku ciljanu platformu. Prema Agarwal et al. (2009) ovo je jedan od dva glavna pristupa rješavanju problema fragmentacije. Budući da standardizacija API-ja u mobilnom svijetu još uvijek nije moguća, korištenje tehnika programiranja u kojima su pozivi prema sučeljima omotani, to jest apstrahirani, u

odvojene module koji su potom prilagođeni različitim platformama ostaje kao jedino rješenje. Predstavnici ovakvog pristupa su *MobiVine* (Agarwal et al., 2009), *PhoneGap* (PhoneGap, 2011) i *Adobe AIR®* (Adobe Corporation, 2011). Gotovo svi nedostaci navedeni za rješenja koja su temeljena na transformaciji koda također postoje i u ovom pristupu. Na posljetku, treći pristup je korištenje web tehnologija i razvoj više-platformskih web aplikacija, ali kako se u mnogim aspektima razlikuje od pretpostavki ovog istraživanja, ovaj pristup (koji također ima svoje nedostatke) nije u domeni ovog istraživanja.

Stoga, ovo istraživanje fokusira se na prijedlog rješenja koje bi omogućilo veću metodološku interoperabilnost između timova koji razvijaju istu aplikaciju ali na različitim (urođenim) razvojnim okruženjima. Istraživanje odgovara na sljedeća pitanja: (1) koje metodike i razvojni pristupi mogu biti korišteni pri razvoju mobilnih aplikacija: (2) koji artefakti (zahtijevani ulazi i rezultati provedbe aktivnosti) nastaju pri razvoju mobilnih aplikacija, (3) postoje li i kako su velike sličnosti između artefakata i (4) je li moguće ontološki opisati ove artefakte i kreirati osnovu za razvoj sustava koji bi omogućio metodološku interoperabilnost. Stoga, osnovni cilj istraživanja je ontološki opisati artefakte koji nastaju u metodički upravljanom procesu razvoja mobilne aplikacije za dvije ili više platformi te time kreirati osnovu za efikasniji i interoperabilniji proces razvoja više-platformskih mobilnih aplikacija.

S tim u vezi, definirana je i hipoteza istraživanja koja glasi: H_1 - *Moguće je definirati ontološki opis elemenata metodološke interoperabilnosti takav da sadrži strukturne i semantičke aspekte u skupovima artefakata koji nastaju u procesima razvoja mobilne aplikacije za dvije ili više mobilnih platformi.*

Poglavlja ovog sažetka su organizirana sukladno postavljenim istraživačkim pitanjima. Drugo poglavlje prikazuje sustavni pregled literature o metodikama razvoja mobilnih aplikacija; treće poglavlje prikazuje rezultate implementacije metodike pri razvoju za dvije platforme; u četvrtom poglavlju prikazana je ontološka definicija artefakata, a u posljednjem poglavlju prikazani su diskusija rezultata i zaključak.

2. Metodike razvoja mobilnih aplikacija: sustavni pregled literature

"Sustavni pregled literature (SLR) predstavlja način vrednovanja i interpretiranja svih dostupnih rezultata istraživanja relevantnih za definirano istraživačko pitanje, područje ili fenomen od interesa. Sustavni pregled ima za cilj prikazati objektivno vrednovanja istraživačke teme korištenjem vjerodostojne, stroge i provjerljive metodike" (Kitchenham i Charters, 2007). Budući da je SLR metoda nova u području softverskog inženjerstva (SE), prvo smo analizirali najbolju praksu u provođenju ove složene i vremenski zahtjevne metode. Pri tome smo slijedili opis provedbe metode dan u (Kitchenham i Charters, 2007), koji smo

upotpunili preporukama i rezultatima drugih utjecajnih autora u području SE. Posebna pozornost je usmjerena na provođenje metode od strane doktorskih studenata. Rezultati ovog istraživanja su korišteni pri provedbi SLR-a, kako je prikazano u sljedećem poglavlju.

2.1. Provedba SLR-a

Rezultati preliminarnog istraživanja o postojećim metodikama razvoja pokazali su da se razvoj za mobilne uređaje razlikuje od razvoja ostalih aplikacija, da je agilni pristup najčešće korišten u metodikama razvoja mobilnih aplikacija te da niti jedna od promatranih metodika nije primjenjiva bez dodatnih napora kako bi se proces detaljnije opisao ili prilagodio specifičnim razvojnim okruženjima ili zahtjevima mobilnih aplikacija. Ovo ukazuje na potrebu sveobuhvatnog i objektivnog istraživanja kako bi se dobio uvid u metodike koje se mogu koristiti za razvoj aplikacija za mobilne uređaje.

Također, provedeno je i drugo preliminarno istraživanje kako bi se identificirali postojeći sustavni pregledi literature o metodikama razvoja mobilnih aplikacija. Pretražene su IEEExplore, ACM digitalna biblioteka, INSPEC, CiteSeerX i GoogleScholar baze podataka korištenjem sljedećeg upita: *("literature review" OR SLR) AND (mobile development)*. Informacije dostupne u spomenutim bazama podataka pokazuju da ne postoji sustavni pregled literature koji pokriva područje metodika razvoja mobilnih aplikacija. To potvrđuje potrebu za ovakvim istraživanjem.

Kako bi istraživanjem obuhvatili navedene ciljeve, definirana su sljedeća istraživačka pitanja:

- **RQ1** – Koje metodike i pristupi razvoja su prikazani u literature kao definirani u teoriji ili korišteni u praksi razvoja mobilnih aplikacija?
- **RQ2** – Jesu li identificirane metodike i pristupi primjenjivi za razvoj više-platformskih mobilnih aplikacija?

Protokol sustavnog pregleda je definiran sukladno naputcima danim u (Kitchenham i Charters, 2007) dok je predložak korišten za izradu protokola definiran u (Biolchini et al., 2005) i dodatno pojašnjen u (Mian et al., 2005). Upit korišten u glavnom istraživanju je bio *(mobile AND ("software development" OR "system development" OR "application development" OR "program development") AND (methodology OR method OR approach OR framework OR process OR procedure OR model))* i izvršen je na dostupnim relevantnim elektronskim izvorima (časopisi i zbornici) u polju softverskog inženjerstva kako su predložili Brereton et al. (2007), Hannay et al. (2007) te Kitchenham i Charters (2007).

Pregled literature proveden je kroz nekoliko faza uključujući identifikaciju primarnih izvora literature, primjenu kriterija uključivanja i isključivanja te procjenu kvalitete. To je na kraju iz početnih 6761 izvor rezultiralo odabirom 49 relevantnih naslova na kojima je provedena faza dohvata podataka i sinteze rezultata.

Kako je prikazano u tabelama 1 i 2, ukupno su identificirane 22 metodike i 7 pristupa razvoju. Ove metodike i pristupi su novo-kreirani (tabela 1) ili postojeći (tabela 2), a pogodni su za razvoj više-platformskih mobilnih aplikacija.

Tabela 1 – Novo-kreirane metodike i pristupi razvoju

Naziv	Tip
Agile Methodology for Mobile Software Development	M
Agile Solo	M
Agile usability process	M
DEAL	M
Integrated Product Development Process for Mobile Software	M
Inter-combined Model	M
MASAM methodology	M
Methodology for Building Enterprise-Wide Mobile Applications	M
MicroApp visual approach	M
Mobile Application Development Methodology	M
Mobile-D	M
New media application prototyping	M
Systems Development Methodology	M
ViP (Virtual Platform)	M
Composite Application Software Development Process Framework	P
MobiLine	P

Tip: M - Metodika, P - Pristup

Tabela 2 – Korištene metodike i pristupi

Naziv	Tip
Design Science	M
Dynamic Channel Model	M
Extreme Programming	M
Kanban	P
Mobile-D	M
Mobile Engineering (MobE)	M
Mobile RAD	M
Rapid Application Development	M
Scrum	M
Model Driven Development	P
Model Driven Product Lines	P
Software Product Lines	P
Test Driven Development	P

Tip: M - Metodika, P - Pristup

Samo je jedna metodika spomenuta u nekoliko izvora literature, dok su sve druge metodike prisutne isključivo u jednom izvoru. Također, kao što se i očekivalo, metodike i pristupi u području razvoja mobilnih aplikacija su novi. Samo 4 izvora literature su stariji od 5 godina, dok su svi drugi izvori mlađi. Provedena procjena kvalitete literature rezultirala je prosječnom ocjenom 2,735 od 5 (68,38%) uz standardnu devijaciju od 0,903 iz čega se može zaključiti da je kvaliteta literature relativno niska s velikim razlikama od izvora do izvora.

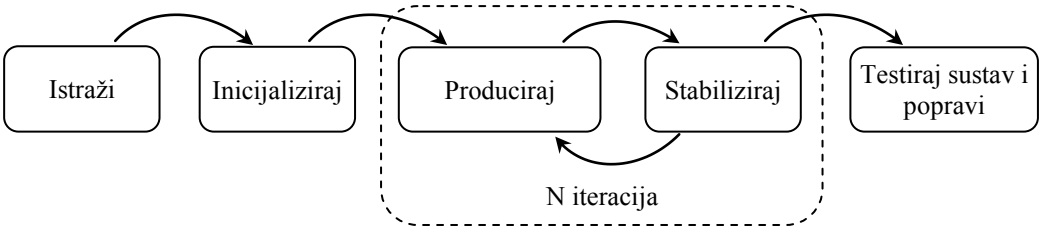
S druge strane, više autora je prikazivalo korištenje postojećih metodika ili pristupa. Ukupno 9 metodika i 4 pristupa su korišteni. Važno je spomenuti da je samo jedna metodika identificirana kao novo-kreirana i ujedno korištena u nekom drugom primjeru. Korištenje ove metodike je prikazano u pet različitih izvora, dok ostale novo-kreirane metodike i pristupi nisu prikazani kao korišteni osim u originalnim studijama.

2.2. Odabir razvojne metodike

Jedna od osnovnih pretpostavki ovog istraživanja povećanja metodološke interoperabilnosti je neovisnost o platformi i primijenjenoj metodici (to jest, može se primijeniti na bilo koju metodiku ontološki opisanu). Stoga, za nastavak istraživanja možemo odabrati bilo koju od ukupno 22 identificirane metodike. Kako bi izbjegli nasumični odabir, definiran je kriterij za odabir koji je temeljen na učestalosti korištenja novo-kreirane metodike u nekom drugom primjeru osim u originalnim studijama. Analiza SLR rezultata je pokazala da je samo *Mobile-D* metodika specifično kreirana za razvoj mobilnih aplikacija i ujedno u drugim izvorima prikazana kao korištena u praksi. Također, provedeno je dodatno istraživanje takozvane *sive literature* kako bi se pronašli dodatni materijali o odabranoj metodici. Rezultati su pokazali da je metodika detaljno opisana od strane njenih autora u nekoliko studija od kojih se važnošću ističe (Abrahamsson et al., 2005).

3. Implementacija metodike

Mobile-D proces (pogledaj sliku 1) uključuje pet faza koje se izvode u djelomično inkrementalnom pristupu. Cilj prve faze, pod nazivom *Istraži*, je pripremiti osnove za budući razvoj. Faza *Inicijaliziraj* bi trebala opisati i pripremiti sve komponente aplikacije, te prepoznati kritične točke projekta. Faza inicijalizacije se obično naziva i nulta iteracija (0-iteracija) budući da osim *postavljanja projekta* uključuje i *planiranje, izradu i isporuku* koji se inače koriste u fazi produkcije. Ideja 0-iteracije je osigurati funkcionalnost razvojnog okruženja na način da se implementiraju određeni reprezentativni dijelovi funkcionalnosti tehnikom prototipiranja. Faze *Produkcije* i *Stabiliziranja* se izvode iterativno sve dok se ne razviju sve funkcionalnosti mobilnog proizvoda. Iteracija počinje *planiranjem* u fazi produkcije, a prva aktivnost je *poslije-iteracijska radionica* koja ima za cilj poboljšati razvojni proces kako bi bolje odgovarao novonastaloj situaciji i potrebama tima. Nakon toga slijede zadaci *analize zahtjeva*, *planiranja iteracije* i *kreiranja testova prihvatljivosti* i izvršavaju se tijekom *dana planiranja* (eng. Planning day). *Radni dan* (eng. Working day) se temelji na implementaciji temeljenoj na paradigmama razvoja vođenog testiranjem, programiranja u paru, neprestane integracije i optimizacije programskog kôda. Ovaj *dan* završava zadatkom *obavještavanja naručitelja* o novim funkcionalnostima. Konačno, *dan isporuke* (eng. Release day) uključuje aktivnosti *integriranja* i *testiranja* rješenja. Faza *Stabiliziranja* ima za cilj dovršiti implementaciju te ukoliko je potrebno integrirati podsustave. Kako ova faza može također uključivati razvoj i programiranje, aktivnosti su slične aktivnostima faze produkcije. Jedina dodatna aktivnost se odnosi na pripremu dokumentacije. Svaka iteracija bi trebala završiti novom funkcionalnošću koja je spremna za isporuku korisniku.



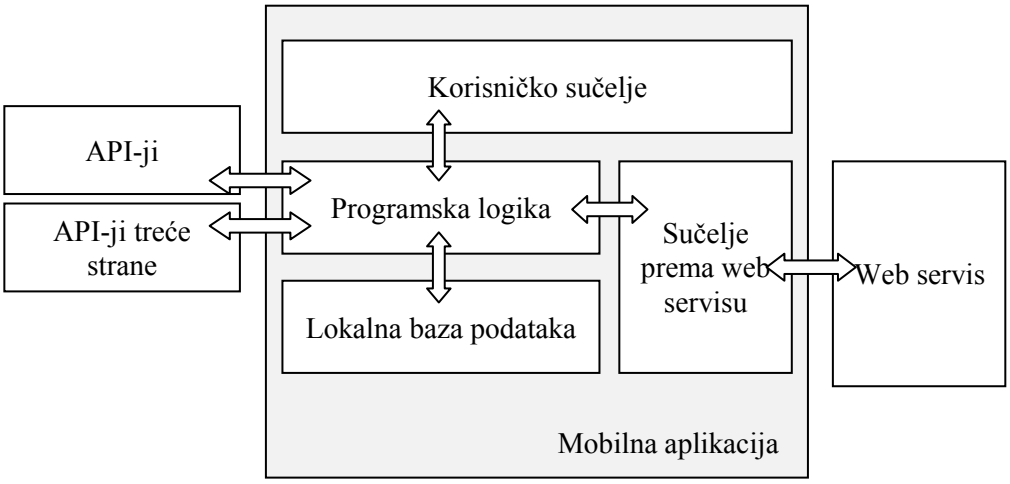
Slika 1 - Mobile-D proces

Posljednja, faza *Sistemskog testiranja i ispravke pogrešaka* služi za provjeru funkcionalnosti kreiranog sustava u usporedbi s korisničkim zahtjevima. Također projektnom timu daje povratnu informaciju o funkcionalnosti sustava i uočenim pogreškama kako bi se provelo ispravljanje pogrešaka kao posljednja iteracija Mobile-D procesa. Ova posljednja iteracija nije obvezna, ali kad se provodi onda sadrži sve aktivnosti kao i ostale već pojašnjene faze koje sadrže implementaciju.

Mobile-D sugerira korištenje *razvoja upogonjenog testiranjem* (eng. Test Driven Development – TDD) koji je sastavni dio svih Mobile-D faza. Osnovne i napredne koncepte TDD-a može se pronaći u (Hammond i Umphress, 2012). Svrha TDD-a je dati programerima sigurnost da je kôd koji kreiraju ispravan te voditi dizajn programskog kôda u strukturu koja je lako provjerljiva testovima. Također, optimizacija i restrukturiranje kôda (eng. refactoring) se temelje na TDD-u kako bi se osiguralo da promjene nastale na postojećem kôdu nisu pokvarile postojeće funkcionalnosti (Abrahamsson et al., 2005).

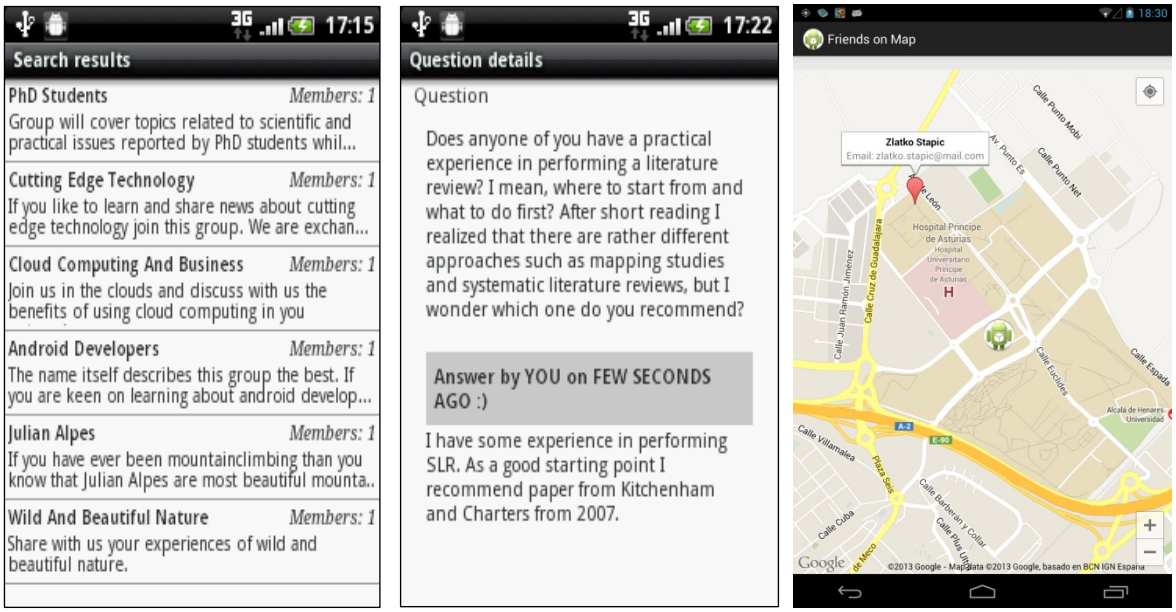
Kako bi semantički promotrili proces razvoja te identificirali artefakte koji se u njemu koriste i nastaju, razvili smo prototipnu aplikaciju, nazvanu KnowLedge, za Android i Windows Phone platforme. Aplikacija ima za cilj omogućiti korisnicima učenje i/ili dijeljenje znanja na interaktivan način u obliku društvene mreže. Između ostalih, osnovne funkcionalnosti aplikacije uključuju pregledavanje postojećih kategorija i pronalazak postojećeg znanja o određenoj temi, slanje zahtjeva za novim pojašnjenjem/instrukcijama/uputama, dijeljenje znanja u grupama i slično.

Sustav je temeljen na servisno orijentiranoj arhitekturi, mobilnoj aplikaciji, udaljenoj bazi, te korištenju globalnog sustava za pozicioniranje (GPS). Također, kako se može vidjeti na slici 2, arhitektura mobilne aplikacije je također višeslojna s tri odvojena ali povezana sloja. Unutarnja kohezija (pogledaj (Miller, 2008)) prikazanih modula je visoka nasuprot vanjskoj (međusobnoj) povezanosti koja je niska.



Slika 2 – Arhitektura mobilne aplikacije

Metodika Mobile-D ima jasnu tehničku dokumentaciju i jako je dobro dokumentirana te ju je bilo jednostavno slijediti pri razvoju koji je u konačnici trajao kraće nego je bilo inicijalno planirano. Nekoliko slika ekrana kreirane aplikacije su vidljive na slici 3.



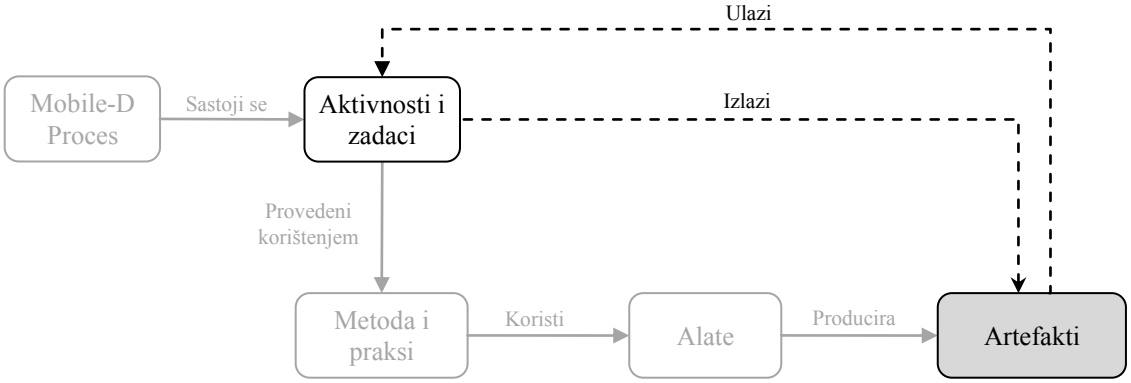
Slika 3 – Slike ekrana aplikacije

U slučaju razvoja za Windows Phone, cijeli proces je ponovljen, ali budući je struktura kreiranih artefakata ostala ista kao u Android scenariju, tijekom ovog procesa razvoja pokušalo se fokusirati na načine i mogućnosti ponovnog iskorištavanja postojećih artefakata. Iako smo očekivali određene sličnosti između artefakata, rezultati su bili iznenađujući. Zaključili smo da su mnogi artefakti u potpunosti ili djelomično iskoristivi. Stoga, iako smo tijekom razvoja Windows Phone aplikacije imali određenih problema specifičnih za WP platformu te određenih problema s testiranjem, trajanje procesa razvoja je u WP slučaju

skraćeno za 30 radnih dana u usporedbi s planom, te za 16 radnih dana (18,4%) u usporedbi s procesom razvoja za Android.

4. Identifikacija artefakata

Budući da postoji više definicija artefakta (npr. od Hilpinena (2011) ili od Parkera (2011)), za potrebe ovog istraživanja najprikladnija je definicija od Conradija (2004) koji kaže da je artefakt „*bilo koji dio softvera (to jest model/opis/kôd) kreiran i korišten tijekom razvoja i održavanja računalnog programa*“. Kako je cilj ovog istraživanja bio analizirati strukturalne i semantičke aspekte niza artefakata, proveli smo analizu samo promatrajući semantičke koncepte, dok drugi pristupi, kao promatranje proceduralnih koncepata ili pragmatičnih koncepata nisu uključeni. Stoga, samo smo promatrali artefakte i njihove veze prema aktivnostima i zadacima, kako je prikazano na slici 4.



Slika 4 – Fokusiranje na semantiku i izvor artefakata

Analiza artefakata je provedena u dva koraka. Prvo, analizirali smo Mobile-D procesnu biblioteku (Abrahamsson et al., 2005) i identificirali dokumente i druge isporuke koje su neovisne o platformi, a definirani su na visokoj razini apstrakcije. Zatim drugo, budući da pristup u identificiranju i grupiranju artefakata isključivo temeljem faze izvornog nastanka nije dobar, te budući da smo tijekom implementacije prikupili dodatne informacije o artefaktima, sistematizirali smo i opisali sve identificirane artefakte za obje platforme koristeći predložak kao u tabeli 3.

Tabela 3 – Predložak opisa artefakata

Naziv artefakta	Tip	Opis	Ulazi i izlazi po fazama									
			I		II		III		IV		V	
			Ulaz	Izlaz	Ulaz	Izlaz	Ulaz	Izlaz	Ulaz	Izlaz	Ulaz	Izlaz

Stoga, konceptualno, kreirali smo temelje identificiranja ne samo kreiranih dokumenata već i onih artefakata koje bi bilo teško identificirati da je projekt realiziran izvan laboratorija.

Tablica 4 prikazuje dio liste identificiranih artefakata, uključujući i inicijalnu klasifikaciju, opis i vezu prema fazama Mobile-D procesa. Koristili smo klasičnu CRU notaciju kako bi smo označili artefakte koji su bili kreirani (C), korišteni/čitani (R) i ažurirani (U).

Tabela 4 – Dio liste identificiranih artefakata u procesu razvoja za Android

Naziv artefakta	Tip	Opis	Ulazi i izlazi po fazama									
			I		II		III		IV		V	
			Ulaz	Izlaz	Ulaz	Izlaz	Ulaz	Izlaz	Ulaz	Izlaz	Ulaz	Izlaz
Mobile-D procesna biblioteka	Dokument	Procesna biblioteka koja detaljno opisuje Mobile-D metodiku. Korištena je kao vodić za implementaciju svake faze. (Abrahamsson et al., 2005)	R		R		R		R		R	
Prijedlog projekta	Dokument	Kreiran prije procesa razvoja. Opisuje inicijalnu ideju i osnovne funkcionalnosti proizvoda.	R									
Projektni plan	Dokument	Sadrži sve informacije o projektu uključujući podatke o korisnicima, domenu projekta, planirane aktivnosti i njihovo trajanje, planove dokumentacije i slično. U skladu je s agilnom praksom, te je ažuriran tijekom iteracija.		C	R	U	R	U				
…	…	…										

Proces identificiranja rezultirao je s ukupno 60 različitih artefakata za Android slučaj te s 61 artefakt za Windows Phone slučaj. Spoj ova dva niza artefakata rezultirao je s ukupno 71 identificirani artefakt grupiran u 12 grupa sukladno tipu.

U analizi artefakata obiju platformi zaključili smo da 50 artefakata (70,42% svih identificiranih artefakata) su zajednički za oba razvojna procesa. Također, mnogi od ovih zajedničkih artefakata su neovisni o platformi jer su rezultat metodičkog pristupa. Ukupno, 20 od 50 identificiranih zajedničkih artefakata (40,00%) su kreirani ili korišteni samo jedanput jer su bili identični u oba razvojna procesa. Također, 13 artefakata (26,00%) se moglo djelomično ponovno iskoristiti pri procesu razvoja za drugu (i svaku sljedeću) platformu. Konačno, prepoznali smo 17 artefakata (34,00% svih zajedničkih artefakata) s veoma malom razinom moguće ponovne iskoristivosti. Ti artefakti su klasificirani kao oni koje je potrebno ponovno razviti za svaku novu platformu. Dio rezultata unakrsne analize može se vidjeti u tablici 5. Svi ostali artefakti su klasificirani kao ovisni o platformi, te također imaju određene ponovno iskoristive semantičke ili sintaktičke elemente kao što su slijed, iteracije, algoritmi i slično.

Tabela 5 – Dio zajedničkih artefakata za Android i WP proces razvoja

Naziv artefakta	Isti	Djelomično ponovno korišten	Različit
Mobile-D procesna biblioteka	X		
Prijedlog projekta	X		
Dokument inicijalnih zahtjeva	X		
Projektni plan		X	
Lista provjere projektnog plana		X	
Predložak liste provjere projektnog plana	X		
Gantogram	X		
Plan mjerenja		X	
Opis arhitekture sustava			X
…			

Ukupno, 33 artefakta (66,00% zajedničkih artefakata) su potpuno ili djelomično ponovno iskoristiva. Stoga, možemo zaključiti da su ovi rezultati ohrabrujući te predstavljaju čvrste temelje i motivaciju semantičkoj analizi koja slijedi.

5. Ontologija za metodološku interoperabilnost

Izraz "ontologija" preuzet je iz filozofije, ali su njegova uporaba i značenje u računalnoj znanosti dobili novu i prilagođenu dimenziju. S obzirom da ne postoji konsenzus o definiciji ontologije, u kontekstu ovog istraživanja, pojam ontologija promatramo kao *eksplicitnu formalnu konceptualizaciju dogovorenog razumijevanja promatrane domene koja uključuje rječnik pojmova koji opisuju elemente domene, značenje kako bi se definirale veze elemenata domene i pragmatiku u cilju definiranja moguće uporabe tih elemenata.*

5.1. Definiranje pristupa razvoju ontologije

Noy i McGuinness (2001) su dali sveobuhvatan pregled mogućih razloga za korištenje ontologija. Autori su prepoznali mogućnost korištenja ontologija za: *dijeljenje uobičajenog razumijevanja strukture informacija među ljudima ili softverskim agentima, omogućavanje ponovnog korištenja znanja određene domene, eksplicitno navođenje pretpostavki domene, odvajanje znanja o domeni od operativnog znanja, analiziranje znanja o određenoj domeni*. Osim toga, ontologije se koriste kao posredni mehanizam u specifičnom pristupu za postizanje *semantičke interoperabilnost* koji je temeljen na *posrednicima* (eng. intermediary-based approach) (Park i Ram, 2004) što je od posebnog značenja u ovom istraživanju. Takva interoperabilnost prema Paulheimu i Probstu (2010), može se definirati na različitim razinama: semantička interoperabilnost na razini izvora podataka, na razini poslovne logike i na razini korisničkog sučelja, ali začudo, interoperabilnost na metodičkoj razini se rijetko spominje u literaturi.

Iako postoje različite vrste ontologija (vidi Lovrenčić (2007)), ontologija koja je predmet ovog istraživanja je klasificirana kao ontologija domene. Ontologija domene može se definirati kao mreža pojmova modela domene (teme, elementi znanja) koji definiraju elemente i značenjske odnose među njima (Brusilovsky et al., 2005). Korištenje ontologije domene pogodno je za opisivanje cjelokupnog sadržaja vezanog za metodiku i pristup razvoja aplikacija. Isto tako, postoji nekoliko radova koji daju opsežan pregled metodika razvoja ontologije, kao što su (Dahlem, 2011), (Lovrenčić, 2007) i (Kabilan, 2007). Međutim, zbog svojih karakteristika kao što su jednostavnost, fokusiranje na rezultate i iterativni pristup, možemo Noy i McGuinnessovu (2001) metodiku *Ontology development 101* nazvati *agilnom metodikom razvoja ontologije*, a to je razlog zašto ju smatramo kao najpogodniju za korištenje tijekom ovog istraživanja. Konačno, tu su i mogućnosti korištenja različitih alata i jezika za razvoj ontologije. Istraživanje provedeno od strane Khondokera i Muellera (2010) pokazuje da se za razvoj ontologija daleko najviše koristi alat pod nazivom *Protégé*. Budući da je Protégé usklađen s metodikom OD101 te se naširoko koristi od strane znanstvenika i stručnjaka u područjima razvoja informacijskih sustava i upravljanja znanjem, odlučili smo ga koristiti i u našem istraživanju. Napokon, s obzirom da Protégé radi s dva jezika prikaza ontologije, okviri i OWL, oba smo razmotrili i odabrali OWL2 DL kao primjereniji jezik u našem slučaju.

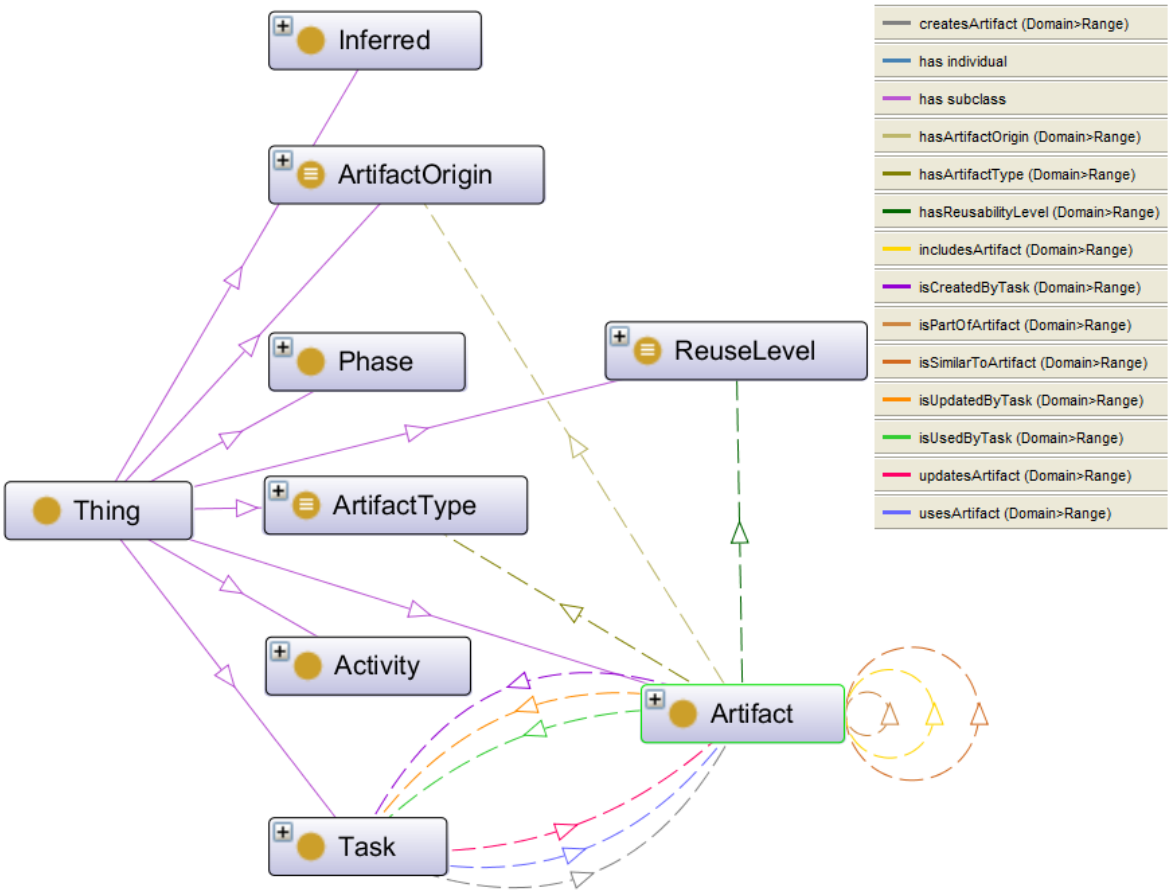
5.2. Razvoj ontologija

Proces razvoja ontologije je izveden u tri koraka. Prvo smo razvili ontologiju za Android platformu, a zatim i ontologiju za Windows Phone platformu. Na kraju smo spojili kreirane ontologije u konačnu, zajedničku, ontološku definiciju.

Popis pojmova koji se pojavljuju u našoj domeni interesa postupno je nastajao tijekom cijelog procesa razvoja ontologije. Konačan popis koji su temelj za našu ontologiju uključuje sljedeće pojmove: *faza, aktivnost, zadatak, artefakt, ulazi u zadatak, rezultati zadatka, tip artefakta, podrijetlo artefakta, korištenje artefakta, hijerarhija artefakata, ponovna iskoristivost, sličnost artefakata*. U procesu definiranja klasa i hijerarhije, slijedili smo savjet od Uscholda i Gruningera (1996) i koristili pristup *od sredine prema vani* (eng. middle-out approach) tako što smo prvo definirali važnije koncepte a zatim po potrebi stvorili generalizacije i specijalizacije. Pristup je rezultirao s ukupno definirane 152 klase koje su organizirane u 7 vršnih klasa za Android platformu, 153 klase slično organizirane za Windows Phone platformu i 213 klasa u završnoj spojenoj ontologiji. Vršne klase iz konačne ontologije su prikazane na slici 5.

U cilju definiranja znanja o strukturi, semantici i uporabi elemenata ontologije definirali smo 12 svojstava objekata za dvije specifične ontologije i 14 svojstava objekata za konačnu spojenu ontologiju.

Ta svojstva su: *sastojiSeOd, stvaraArtefakt, imaPodrijetloIzArtefakta, imaTipArtefakta, uključujeArtefakt, imaRazinuPonovneIskoristivosti, stvorenJeUZadatku, dioJeArtefakta, izvodiSeU, sličanJeArtefaktu, ažuriranJeUZadatku, korištenJeUZadatku, ažuriraArtefakt, koristiArtefakt (eng.: consistsOf, createsArtifact, hasArtifactOrigin, hasArtifactType, includesArtifact, hasReusabilityLevel, isCreatedByTask, isPartOfArtifact, isPerformedIn, isSimilarToArtifact, isUpdatedByTask, isUsedByTask, updatesArtifact, usesArtifact).*



Slika 5 – Vršne klase u konačnoj ontologiji

Slika koja opisuje dio završne ontologije pokazuje da je *artefakt* u konačnici povezan sa *Zadatkom (eng. Task), PodrijetlomArtefakta (eng. ArtifactOrigin), TipomArtefakta (eng. ArtifactType) i RazinomPonovneIskoristivosti (eng. ReuseLevel)*. Među tim odnosima, veza sa *Zadatkom (Task)* je najjače jer je definirana s tri svojstva pri čemu svako svojstvo ima odgovarajuće povratno svojstvo. Iako postoje, odnosi među ostalim vršnim klasama oni nisu prikazani na ovoj slici kako bi se fokusirali samo na *Artefakt*.

Za spajanje instanci klasa s utvrđenim svojstvima morali smo slijediti OWL 2 DL sintaksu, ograničenja i pravila. Osim toga, OWL DL temelji se na paradigmi *logike otvorenog svijeta* (eng. Open world assumtion), a OWA paradigma polazi od toga da ne možemo zaključiti da nešto ne postoji dok nije eksplicitno navedeno da to ne postoji. Na primjer, kako bi se u potpunosti definirali metodički artefakti moramo koristiti *završne aksiome* i izrijekom navesti

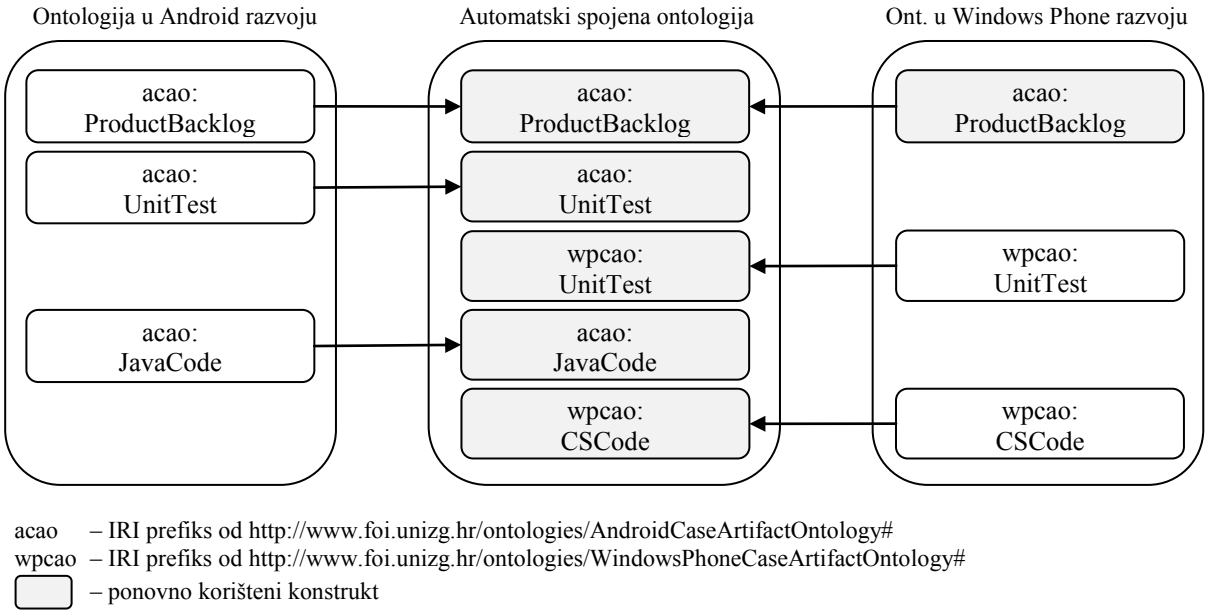
da takvi artefakti nisu stvoreni niti modificirani u našem razvojnom procesu. Oni su samo korišteni. Primjer takvog opisa je dat u kôdu 1.

```
SubClass Of:
Artifact
hasArtifactOrigin only MethodologicalArtifact
hasArtifactOrigin some MethodologicalArtifact
hasArtifactType only Document
hasArtifactType some Document
isUsedByTask only Task
isUsedByTask some Task
not (isCreatedByTask some Task)
not (isUpdatedByTask some Task)
```

Kôd 1 – Dovoljan opis klase u OWA paradigmi

Tijekom razvoja ontologije za Android platformu stavili smo naglasak na proces razvoja ontologije vođen odabranom metodikom razvoja. U ovom slučaju proces razvoja je proveden iz početka. U drugoj iteraciji smo stavili naglasak na ponovnu iskoristivost postojeće ontologije što je dokazalo njenu valjanost i fleksibilnost. Time je potvrđen i konceptualni model koji je osnova naših ontologija koje opisuju razvoj za jednu platformu.

U razvoju jedinstvenog ontološkog opisa, naglasak je stavljen na spajanje, ažuriranje i evaluaciju ontologije. Veći dio procesa spajanja je učinjen automatski (vidi sliku 6). Nakon spajanja dvaju ontologija, nije bilo redundancije kojom bi se morali baviti, te nismo imali problema u nadogradnji ontologije s novim znanjem. To dokazuje da je ontologija ponovno iskoristiva i da se može proširiti.



Slika 6 – Primjer automatski spojene ontologije

Osnovni pojmovi definirani u ontologiji razvoja za Android su korišteni u izradi ontologije razvoja za Windows Phone i na taj način su također uključeni i u konačnu ontologiju. Kako smo se usmjerili na nadogradnju ontologije sa znanjem o ponovnoj iskoristivosti artefakta, morali smo uvesti dva nova važna pojma (*ponovna iskoristivost i sličnost artefakata*).

Kreirana ontologija sastoji se od 213 klasa, 14 objektnih svojstava i 2213 aksioma definiranih pomoću ALCRIF-DL jezika izraza. Ontologija u izvornom OWL/XML formatu može se pronaći na http://barok.foi.hr/~zstapic/ont/mcao.owl, dok punoj OWLDoc dokumentaciji ontologije može se pristupiti i analizirati ju na http://barok.foi.hr/~zstapic/ont/mcao/doc/.

5.3. Vrednovanje završne ontologije

Kako bismo provjerili ispravnost i valjanost naše ontologije, tijekom trajanja cijelog procesa razvoja, koristili smo sljedećih sedam mehanizama provjere ispravnosti i valjanosti:

1. Metodički vođen proces razvoja ontologije
2. Implementacija preporuka i savjeta drugih autora
3. Korištenje alata za zaključivanje kako bismo provjerili ontologiju u svakoj iteraciji
4. Korištenje W3C OWL alata za provjeru
5. Korištenje Protégé dodatka za vrednovanje ontologije
6. Korištenje DL upita za dohvat informacije zaključivanjem nad opisanim znanjem
7. Provjera rezultata od strane stručnjaka

Prvih pet mehanizama za vrednovanje su povezani s verifikacijom ontologije i koriste se kako bi se smanjio rizik kreiranja sintaktičkih ili osnovnih semantičkih pogrešaka tijekom cijelog procesa razvoja ontologije.

Posljednja dva mehanizma su povezana s validacijom ontologije. Ova dva mehanizma koriste se na kraju razvojnog procesa kako bi provjerili predstavlja li kreirana ontologija domenu znanja na semantički ispravan način. Upiti su kreirani i izvršeni na konačnoj ontologiji kako bi se odgovorilo na sva unaprijed definirana pitanja povezana s razvojem aplikacija za odredišnu platformu i pitanja vezana uz ponovnu iskoristivost artefakata kako je definirano na početku procesa izrade ontologije. Na primjer, kako bi dobili sve ponovno iskoristive artefakte koji su korišteni, stvoreni ili ažurirani tijekom zadatka planiranja iteracije možemo koristiti ovakav upit:

```
Artifact
 and ((isUsedByTask some IterationPlanningTask)
   or (isCreatedByTask some IterationPlanningTask)
   or (isUpdatedByTask some IterationPlanningTask))
 and (ReusableArtifacts)
```

Kôd 2 – Artefakti koji se mogu ponovo koristiti u nekom zadatku

Rezultat upita:

```
AcceptanceTest, IterationBacklog, IterationsPlan, MeasurementPlan,
ProductBacklog, ProjectPlan, ProjectPlanChecklist, ProjectPlanGantChart,
StoryCard, StoryCardTemplate, TaskCard, TaskCardTemplate
```

Sljedeći upit nabraja artefakte određene vrste *dokumenta* koji se *potpuno* ili *djelomično* mogu *ponovo koristiti*.

```
Artifact
 and (hasArtifactType some Document)
 and ((hasReusabilityLevel some Completely)
  or (hasReusabilityLevel some Partially))
```

Kôd 3 – Ponovno iskoristivi artefakti određenog tipa

Rezultat upita:

```
InitialRequirementsDocument, MobileDProcessLibrary, ProductBacklog,
ProductProposal, ProjectPlan
```

Svi drugi upiti su kreirani na sličan način, a rezultate su analizirali stručnjaci iz domene razvoja softvera. Korištenje mehanizama provjere ontologije tijekom cjelokupnog razvojnog procesa, te pozitivni rezultati vrjednovanja su dokaz kvalitete i dovršenosti ontologije. To nas dovodi do konačnog zaključka da kreirana *Ontologija razvoja za više-platformi* predstavlja bazu znanja koja se može koristiti pri razvoju informacijskog sustava koji bi imao za cilj voditi razvojne timove kako bi povećali metodološku interoperabilnost ponovnim korištenjem artefakata kreiranih u procesu razvoja više-platformskih mobilnih aplikacija.

6. Rasprava i zaključak

Tijekom istraživanja željeli smo jasno istaknuti pet važnih aspekata kako bi proces istraživanja učiniti transparentnim i ponovljivim. Pri provedbi svake aktivnosti poseban naglasak smo stavili na *motivaciju*, *rezultate*, *doprinos*, *strogost* i *evaluaciju* istraživanja. Pod aspektom *motivacije* željeli smo u svakoj fazi naglasiti razloge za obavljanje istraživačkih aktivnosti. *Rezultatima* i *doprinosom* željeli smo po fazama sistematizirati dobivene rezultate i doprinos znanju. Raspravljajući o istraživačkoj *strogosti* htjeli smo ukazati na naš pristup i njegove glavne karakteristike te raspravljajući o *evaluaciji* htjeli smo naglasiti mehanizme koji su korišteni kako bi provjerili i potvrdili korišteni znanstveni pristup i dobivene rezultate.

U ovom istraživanju može se identificirati nekoliko ograničenja. Na primjer, najveći izazovi s kojima smo se suočili u prvoj fazi istraživanja bili su izvršenje komplicirane i dugotrajne znanstvene metode sustavnog pregleda literature od strane jednog istraživača, nepostojanje institucionalnih pretplata na dostupne znanstvene izvore u Hrvatskoj te nešto malo bolje

stanje u Španjolskoj, nedostatak informacija o završenim projektima o razvoju mobilnih aplikacija u razvojnim tvrtkama koje su orijentirane na dvije ili više ciljanih platformi što nas je natjeralo da razvijemo prototipnu aplikaciju u laboratoriju, predložena ontologija predstavlja razvoj samo jedne aplikacije za dvije platforme te pokriva samo jednu razvojnu metodiku podržanu jednim razvojnim pristupom. Svi navedeni problemi mogu biti prepoznati kao ograničenja ovog istraživanja, ali moramo imati na umu da je ovaj istraživački proces imao za glavni cilj predložiti novi pristup (okvir) koji se može koristiti u rješavanju problema fragmentacije mobilnih platformi.

Slijedeći istraživačke ciljeve definirane na početku istraživačkog procesa identificirali smo metodike koje se mogu koristiti za razvoj mobilnih aplikacija; implementirali smo izabranu metodiku i pristup i stvorili mobilnu aplikaciju za dvije mobilne platforme; identificirali smo i analizirali artefakte koji su nastali u ovom razvojnom procesu te stvorili ontološku definiciju koja opisuje mogućnost ponovne iskoristivosti artefakata u skladu s Mobile-D metodikom.

Prema rezultatima koji su dobiveni tijekom provjere i testiranja ontologije možemo zaključiti da ovakav ontološki opis predstavlja čvrstu osnovu za razvoj informacijskog sustava koji bi vodio razvojne timove prema postizanju metodološke interoperabilnosti uz ponovno korištenje artefakata stvorenih u procesu razvoja više-platformskih mobilnih aplikacija. Osim toga, dokazali smo da je naš ontološki opis fleksibilan i proširiv što nam omogućuje njegovo ažuriranje informacijama o novim artefaktima bez potrebe za promjenom infrastrukture definirane elementima hijerarhije klasa, pobrojanim vrijednostima (eng. value partitions) i objektnim svojstvima. Konačno, model omogućuje kreiranje DL upita koji se mogu koristiti za stjecanje izravne ili neizravne informacije ugrađene u znanje opisano ontologijom. U prethodnom poglavlju pokazali smo primjere takvih upita kojima smo između ostalog odgovorili na pitanja postavljena na samom početku razvoja ontologije.

Dakle, možemo zaključiti **da je moguće definirati ontološki opis elemenata metodološke interoperabilnosti takav da sadrži strukturne i semantičke aspekte u skupovima artefakata nastalih u procesu razvoja mobilnih aplikacija za dvije ili više mobilnih platformi**, čime je H_1 hipoteza potvrđena.

Ovo istraživanje predstavlja sveobuhvatan skup aktivnosti koje su rezultirale konačnim i uporabivim proizvodom. Međutim, proširujući kontekste korištenja takve ontologiju, možemo prepoznati druge moguće istraživačke aktivnosti pa čak i istraživačke smjerove koji se mogu poduzeti. U principu, prepoznajemo dva glavna područja gdje ovo istraživanje postavlja temelj za buduće znanstvene i stručne aktivnosti. Ta područja su *programsko inženjerstvo* s posebnim naglaskom na *mobilno inženjerstvo* i *inženjerstvo znanja* s posebnim naglaskom na *razvoj ontologije*. Stvorena ontologija definira osnovnu infrastrukturu i elemente u predloženom okviru metodološke interoperabilnosti koji je stabilan za dodavanje znanja o

drugim platformama ali bi ga trebalo ponovno analizirati i redefinirati kada je u pitanju korištenje za opis novih i potpuno različitih metodika. S druge strane, kada se govori o istraživačkim aktivnostima u području programskog inženjerstva, već smo spomenuli potrebu prelaska na sljedeću fazu istraživanja tijekom koje bi bio razvijen odgovarajući informacijski sustav za podršku metodološkoj interoperabilnosti i ponovnom korištenju artefakata. Razvoj takvog novog sustava nije trivijalan zadatak i daje mnoge mogućnosti istraživanja u području dizajna, funkcionalnosti, povezanosti s ontološkom definicijom znanja i tako dalje.

Iako postoje ontologije definirane da osiguraju interoperabilnost na različitim razinama u procesu razvoja aplikacija, ovaj novi pristup ima za cilj definirati interoperabilnost na, do sada ne istraženoj, metodičkoj razini. Semantički opisi kreirani i provjereni ovim istraživanjem dokazali su da predloženi pristup i infrastrukturalni okvir predstavljaju solidnu osnovu za nastavak istraživanja u ovom području. Stoga, razvoj ove ontologije je samo prvi korak u nizu aktivnosti koje treba provesti kako bi se razvio cjeloviti semantički podržan informacijski sustav za metodološku interoperabilnost.

Reference

Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., Koskela, J., Kyllönen, P., Salo, O., 2004. Mobile-D: an agile approach for mobile application development, in: Companion to the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages, and Applications, OOPSLA '04. ACM, New York, NY, USA, pp. 174–175.

Abrahamsson, P., Hanhineva, A., Hulkko, H., Jäälinoja, J., Komulainen, K., Korkala, M., Koskela, J., Kyllönen, P., Eporwei, O.T., 2005a. Agile Development of Embedded Systems: Mobile-D (Agile Deliverable No. D.2.3). ITEA.

Adobe Corporation, 2011. Adobe Announces Agreement to Acquire Nitobi, Creator of PhoneGap [Internet izvor]. Adobecom - Press Releases. URL http://www.adobe.com/aboutadobe/pressroom/pressreleases/201110/AdobeAcquiresNitobi.html (pristupano 18.05.12).

Agarwal, V., Goyal, S., Mittal, S., Mukherjea, S., 2009. MobiVine: a middleware layer to handle fragmentation of platform interfaces for mobile applications, in: Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware, Middleware '09. Springer-Verlag New York, Inc., New York, NY, USA, pp. 24:1–24:10.

Amanquah, N., Eporwei, O.T., 2009. Rapid application development for mobile terminals, in: 2nd International Conference on Adaptive Science & Technology (ICAST). Presented at the Technology (ICAST), Accra, Ghana, pp. 410–417.

Biolchini, J., Gomes Mian, P., Candida Cruz Natali, A., Horta Travassos, G., 2005. Systematic Review in Software Engineering (Technical report No. RT - ES 679 / 05). PESC, Rio de Janeiro.

Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M., 2007. Lessons from applying the systematic literature review process within the software engineering domain. J. Syst. Softw. 80, 571–583.

Brusilovsky, P., Sosnovsky, S., Yudelson, M., 2005. Ontology-based Framework for User Model Interoperability in Distributed Learning Environments, in: World Conference on ELearning, E-Learn 2005. AACE, pp. 2851–2855.

Conradi, R., 2004. Software engineering mini glossary [Internet izvor]. URL http://www.idi.ntnu.no/grupper/su/publ/ese/se-defs.html (pristupano 5.5.12).

Dahlem, N., 2011. OntoClippy: A User-Friendly Ontology Design and Creation Methodology. Int. J. Intell. Inf. Technol. 7, 15–32.

Hammond, S., Umphress, D., 2012. Test driven development. ACM Press, p. 158.

Hannay, J., Sjoberg, D., Dyba, T., 2007. A Systematic Review of Theory Use in Software Engineering Experiments. IEEE Trans. Softw. Eng. 33, 87–107.

Hilpinen, R., 2011. Artifact [Internet izvor]. Stanf. Encycl. Philos. URL http://plato.stanford.edu/entries/artifact/ (pristupano 5.5.12).

Hosbond, J.H., 2005. Mobile Systems Development: Challenges, Implications and Issues, in: Krogstie, J., Kautz, K., Allen, D. (Eds.), Mobile Information Systems II, IFIP International Federation for Information Processing. Springer Boston, pp. 279–286.

Jeong, Y.-J., Lee, J.-H., Shin, G.-S., 2008. Development Process of Mobile Application SW Based on Agile Methodology, in: Proceedings of 10th International Conference on Advanced Communication Technology, (ICACT 2008). IEEE, Gangwon-Do, pp. 362–366.

Kabilan, V., 2007. Ontology for information systems (04IS) design methodology: conceptualizing, designing and representing domain ontologies. Data- och systemvetenskap, Kungliga Tekniska högskolan, Kista.

Khondoker, R.M., Mueller, P., 2010. Comparing Ontology Development Tools Based on an Online Survey, in: Proceedings of the World Congress on Engineering. Presented at the WCE 2010, London.

Kitchenham, B., Charters, S., 2007. Guidelines for performing Systematic Literature reviews in Software Engineering Version 2.3 (Technical report No. EBSE-2007-01). Keele University and University of Durham.

La, H.J., Kim, S.D., 2009. A service-based approach to developing Android Mobile Internet Device (MID) applications. 2009 IEEE Int. Conf. Serv.-Oriented Comput. Appl. SOCA 00, 1–7.

Lovrenčić, S., 2007. Formalna ontologija sveučilišnih studija [Doktorska disertacija]. Sveučilište u Zagrebu, Varaždin, Hrvatska.

Manjunatha, A., Ranabahu, A., Sheth, A., Thirunarayan, K., 2010. Power of Clouds in Your Pocket: An Efficient Approach for Cloud Mobile Hybrid Application Development, in: 2010 IEEE Second International Conference on Cloud Computing Technology and Science. pp. 496–503.

Mian, P., Conte, T., Natali, A., Biolchini, J., Travassos, G., 2005. A Systematic Review Process for Software Engineering, in: ESELAW '05: 2nd Experimental Software Engineering Latin American Workshop.

Miller, J., 2008. Cohesion And Coupling. MSDN Mag. - Microsoft J. Dev. 23.

Noy, N.F., McGuinness, D.L., 2001. Ontology Development 101: AGuide to Creating Your First Ontology (Technical report No. KSL-01-05; SMI-2001-0880), Stanford Knowledge Systems Laboratory and Stanfrod Medical Informatics Technical Report. Stanford University, Stanfrod.

Park, J., Ram, S., 2004. Information systems interoperability: What lies beneath? ACM Trans. Inf. Syst. 22, 595–632.

Parker, P.M., 2011. Definition of artifact [Internet izvor]. Websters Online Dict. URL http://www.websters-online-dictionary.org/definitions/artifact (pristupano 7.5.11).

Paulheim, H., Probst, F., 2010. Application integration on the user interface level: An ontology-based approach. DATA Knowl. Eng. 69, 1103–1116.

PhoneGap, 2011. Take the pain out of compiling mobile apps for multiple platforms [Internet izvor]. PhoneGap Build. URL https://build.phonegap.com/ (pristupano 27.8.11).

Rahimian, V., Ramsin, R., 2008. Designing an agile methodology for mobile software development: A hybrid method engineering approach, in: Proceedings of Second International Conference on Research Challenges in Information Science, RCIS (2008). IEEE, Marrakech, pp. 337–342.

Rhomobile, Inc., 2011. Smartphone Enterprise Application Integration, White paper [Internet izvor]. URL http://tiny.cc/rhomobile (pristupano 20.08.11).

Spataru, A.C., 2010. Agile Development Methods for Mobile Applications [Doktorska disertacija]. Sveučilište u Edinburghu, Edinburgh.

Uschold, M., Gruninger, M., 1996. Ontologies: Principles, methods and applications. Knowl. Eng. Rev. 11, 93–136.

CURRICULUM VITAE

Zlatko Stapić was born on October 17th 1983 in Zenica (Bosnia and Herzegovina), where he received high school diploma from *Gimnazija KSC Sv. Pavao*. In 2006 he finished his undergraduate and graduate studies *at Faculty of Organization and Informatics* and obtained a Master of Informatics degree from *University of Zagreb*. Since 2006 he works at the same institution as Teaching Assistant at *Information Systems Development Department* where he currently teaches courses on Software Engineering and Software Analysis and Development.

Zlatko's research interests include methodologies for software development and development of mobile applications. During the last nine years, Zlatko participated in or managed 15 scientific and professional projects, published 30 scientific and professional papers and he is a member of IEEE society and reviewer for several scientific journals and conferences. His special interest in work with students resulted in collaboration with more than 50 students in different scientific and professional projects.

In 2011 he enrolled into a dual degree doctoral program on Information Sciences at *University of Zagreb* (Croatia) and Information and Knowledge Engineering at *University of Alcalá* (Spain). This resulted in *cotuelle* on this thesis and in various collaboration activities including research stays in 2012 and 2013 at Department of Computer Science at *University of Alcalá*.

During the studying and working experience Zlatko received more than 10 different awards, including the 3rd award for organizing the best online course at University of Zagreb from 2010, the special recognition for work with students from 2009, the silver medal from 2007 and golden medal from 2006 for innovations in SmartEcg project, Rectors award in 2006 et cetera.

Since 2002 Zlatko Stapić lives and works in Varaždin (Croatia). He is married and has two children.

Published scientific papers

Stapić Z.. Dealing with mobile platforms fragmentation problem: ontology oriented approach // IDS 2013. Proceedings of the 8th International Doctoral Seminar. Dubrovnik. 2013. pp. 326-331.

Stapić Z., de Marcos Ortega L., Gutiérrez Martínez J. M. Approaches in Development of Multi-platform Mobile Applications: State of the Art // Proceedings of IV International Conference on Application of Advanced Information and Communication Technologies. Loja (Ecuador). 2012. pp. 429-436

Stapić Z., García López E., García Cabot A., de Marcos Ortega L., Strahonja V. Performing Systematic Literature Review in Software Engineering // Proceedings of 23rd Central European Conference on Information and Intelligent Systems. Varaždin. 2012. pp. 441-447.

Radošević D., Orehovački T., Stapić Z. Automatic On-line Generation of Student's Exercises in Teaching Programming // Proceedings of the 21st Central European Conference on Information and Intelligent Systems. Varaždin. 2010. pp. 87-93.

Orehovački T., Stapić Z., Bubaš G. Mobile location based service for positioning and presentation of cultural heritage objects and Web 2.0 technologies. // Informatologia. 42 (2009) , 2, pp. 110-117 (Scopus).

Stapić Z., Orehovački T., Lovrenčić A. In Search of an Improved BoM and MRP Algorithm // Proceedings of the 31st International Conference on Information Technology Interfaces. Zagreb. 2009. pp. 665-670 (ISI Conference Proceedings Citation Index – Science, Scopus, Inspec).

Đanić M., Orehovački T., Stapić Z. Introducing CaCM: toward new students collaboration model // Proceedings of the 19th Central European Conference on Information and Intelligent Systems. Varaždin. 2008. pp. 267-273 (Inspec, CSA)

Stapić Z., Orehovački T., Đanić M.. Determination of optimal security settings for LMS Moodle // Proceedings of the 31st MIPRO International Convention on Information Systems Security. Rijeka. 2008. pp. 84-89.

Stapić Z., Orehovački T., Vrček N. Modular Approach in Integration of ICT Technologies into Mobile Heart-Work Monitoring System // Proceedings of the 27th International Conference on Organizational Science Development. Kranj. 2008. pp. 2793-2799.

Stapić Z., Vrček N., Hajdin G.. Evaluation of Security and Privacy Issues in Integrated Mobile Telemedical System // Proceedings of the 30th International Conference on Information Technology Interfaces (ITI 2008). Dubrovnik 2008. pp. 295-300 (Scopus, Inspec).

Stapić Z., Vrček N., Hajdin G. Legislative Framework for Telemedicine // Proceedings of the 19th Central European Conference on Information and Intelligent Systems. Varaždin, 2008. pp. 605-611 (Inspec).

Konecki M., Orehovački T., Stapić Z. IT users' awareness about the need of strong passwords creation // DAAAM International Scientific Book 2008. Vienna. 2008. pp. 387-394.

Velić M., Novak M., Orešković M., Padavić I., Pedljo H., Stapić Z., Car S. Smart ECG, solution for mobile heart work analysis and medical intervention in case of heart work problems // Abstracts of The 56th Annual Scientific Session of the American College of Cardiology : Special Topics in: Journal of the American College of Cardiology 50 (2007) S8. Elsevier, 2007. pp. 280A-281A (abstract in CC)

Vrček N., Velić M., Stapić Z. Integrated mobile electrocardiography // Proceedings of the 30th MIPRO International Convention on Computers in Technical Systems. Opatija. 2007. pp. 44-47.

Radošević D., Dobša J., Mladenić D., Stapić Z., Novak M. Genre Document Classification Using Flexible Length Phrares // Proceedings of 17th International Conference on Information and Intelligent Systems. Varaždin. 2006. pp. 23-28.

Dobša J., Radošević D., Stapić Z., Zubac M.. Automatic Categorisation of Croatian Web Sites // Proceedings of 25th International Convention MIPRO 2005. Rijeka. 2005. pp. 144-149.

Published professional papers

Stapić Z., Mijač M., Tomaš B. Monetizing mobile applications // Razvoj poslovnih i informacijskih sustava CASE 25. Rijeka, 2013.

Stapić Z., Patekar Bahun D., Maslić D. Comparing native Android and jQuery Mobile capabilities // Razvoj poslovnih i informacijskih sustava CASE 25. Rijeka. 2013.

Obuljen L., Filipaj D., Nađ N., Stapić Z. Challenges in development of RPG mobile application // Razvoj poslovnih i informacijskih sustava CASE 25. Rijeka. 2013.

Supan D., Teković K., Škalec J., Stapić Z. Using Mobile-D methodology in development of mobile applications: challenges and issues // Razvoj poslovnih i informacijskih sustava CASE 25. Rijeka. 2013.

Rendulić B., Mirković I., Laktašić A., Stojanović D., Stapić Z. Using 3D Augmented Reality in Windows Phone 7 Mobile Applications // Razvoj poslovnih i informacijskih sustava CASE 24. Rijeka. 2012.

Stapić Z., Ribičić O. Unutar-aplikacijska kupovina uz primjer implementacije na Android platformi // Razvoj poslovnih i informacijskih sustava CASE 24. Rijeka. 2012.

Šaško Z., Mijač M., Stapić Z., Domínguez Díaz A., Saenz de Navarrete Royo J. Windows Phone 7 Applications development using Windows Azure Cloud // Razvoj poslovnih i informacijskih sustava CASE 24. Rijeka. 2012.

Švogor I., Tušek T., Stapić Z. Component development approach for Android // Razvoj poslovnih i informacijskih sustava CASE 24. Rijeka. 2012. 169-181.

Ribičić O., Tomaš B., Stapić Z. Razvoj aplikacija za Android platformu // Razvoj poslovnih i informatičkih sustava CASE23. Zagreb. 2011. 107-116.

Stapić Z., Besednik M., Curić I., Kralj K., Samošćanec K. Integracija Windows Phone7 aplikacija s Azure oblakom putem CSLA.Net poslovne logike // Razvoj poslovnih i informatičkih sustava CASE23. Zagreb. 2011. 171-177.

Stapić Z., Vincek J., Šaško Z., Zver M. Usporedba razvoja mobilne aplikacije na Nokia Qt i Microsoft WP7 platformama // Razvoj poslovnih i informatičkih sustava CASE23. Zagreb. 2011. 93-100.

Stapić Z., Vrček N. Izazov sinkronizacije podataka između mobilnih i standardnih baza podataka // CASE 22 - Metode i alati za razvoj poslovnih i informatičkih sustava. Rijeka. 2010. 107-112.

Konecki M., Stapić Z., Orehovački T. Razvoj aplikacija korištenjem uzoraka dizajna // CASE 20 - metode i alati za razvoj poslovnih i informatičkih sustava. Rijeka. 2008. 203-209.

Orehovački T., Konecki M., Stapić Z. Primjena Web 2.0 tehnologija u poslovanju // CASE 20 - metode i alati za razvoj poslovnih i informatičkih sustava. Rijeka. 2008. 197-202.

Stapić Z., Orehovački T., Konecki M. OCL kao preduvjet automatiziranom generiranju programskog koda // CASE 20 - metode i alati za razvoj poslovnih i informatičkih sustava. Rijeka. 2008. 87-94