O_HAI 4 Games - D4.3. Case Study 3 - Serious Games & Autonomous Vehicles

Schatten, Markus

O_HAI(4)Games

Orchestration of Hybrid Artificial Intelligence Methods for Computer Games

Case Study 3 - Serious Games & Autonomous Vehicles

This project was funded by the Croatian Science Foundation

Principal investigator:

Markus Schatten

PUBLISHED BY ARTIFICIAL INTELLIGENCE LABORATORY,
FACULTY OF ORGANIZATION AND INFORMATICS, UNIVERSITY OF ZAGREB

HTTP://AI.FOI.HR/OHAI4GAMES

Some of the results presented in this deliverable have been published in [59, 68, 69].

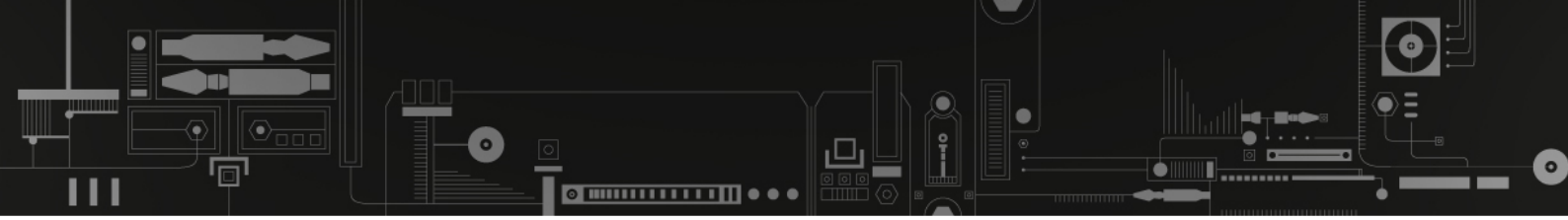
Technical Report No. AIL202301 – First release, May 2023, edit December 2023

Document compiled by: Markus Schatten with inputs from other project team members

This work has been supported in full by the Croatian Science Foundation under the project number IP-2019-04-5824.

The following graphics under CC licences have been used:

- https://en.m.wikipedia.org/wiki/File:Smartphone_icon_-_Noun_Project_283536.svg
- https://commons.wikimedia.org/wiki/File:Toll_Collect_Automat.jpg



hrzz
Hrvatska zaklada za znanost

foi
SVEUČILIŠTE U ZAGREBU
FAKULTET
ORGANIZACIJE I
INFORMATIKE
VARAŽDIN

lab

Contents

1 Project Description 1
1.1 Abstract 1
1.2 Introduction 1
1.3 Team Members 4

2 Digital Twins and Autonomous Vehicles 5
2.1 Introduction 5
2.2 Digital Twins in Autonomous Vehicles 6
2.3 Awkward Π-nguin Orchestration Infrastructure 6
2.4 Digital Twins as Game Actors 8
2.5 Discussion 10
2.6 Conclusion 11

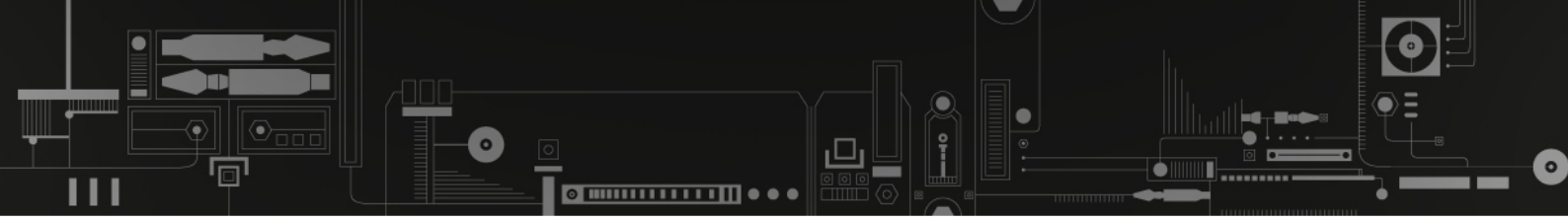
3 Cognitive Agents & Smart Mobility 13
3.1 Introduction 13
3.2 B.A.R.I.C.A. Infrastructure 14
3.3 Possible Applications to Smart Mobility 15
3.4 Conclusion & Future Research 17

4 Game Streaming & Intelligent Transport 19
4.1 Introduction 19
4.2 Related Work 20
4.3 System Architecture 21

4.4 Application in Transport and Mobility Research 22
4.5 Advantages of Game Streaming Systems 23
4.6 Conclusion and Future Work 23

5 Student Projects 25
5.1 Introduction 25
5.2 Building a Self-Driving RC Car 25
5.3 Autonomous Vehicles as a Multi-Agent System 27
5.4 Application of Artificial Intelligence in Racing Games 28

Bibliography 31



hrzz
Hrvatska zaklada za znanost

foi
SVEUČILIŠTE U ZAGREBU
FAKULTET
ORGANIZACIJE I
INFORMATIKE
VARAŽDIN

lab

1. Project Description

1.1 Abstract

Hybrid artificial intelligence (HAI) methods, which can be defined as the orchestration of complementary heterogeneous both symbolic and statistical AI methods to acquire more precise results, are omnipresent in contemporary scientific literature. Still, the methodology of developing such systems is in the most cases ad-hoc and depends from project to project. Computer games have always been connected to the development of AI. From the earliest chess minmax algorithm by Claude Shannon in 1949 to the more recent AlphaGo in 2015, computer games provide an ideal testing environment for AI methods. Similarly, AI has always been an important part of computer games, which have often been judged by the quality of their AI and praised if they used an innovative approach. Computer games allow us to test AI methods, not only for fun and leisure, but also for numerous other fields of human activity through the fields of serious games and gamification. The project proposes to establish a new framework for the orchestration of hybrid artificial intelligence methods with a special application to computer games. Therefore an ontology of hybrid AI methods as well as a meta-model shall be developed that would allow for creating models (ensembles) of hybrid AI methods. This meta-model would be implemented into a modular distributed orchestration platform which would be further enriched with a number of modules to be tested in four gaming related environments: (1) MMORPG games, (2) gamified learning platform, (3) serious game related to autonomous vehicles and (4) a game for a holographic/volumetric gaming console which would also be developed during the project.

1.2 Introduction

The application of HAI which can be defined as the orchestration of heterogeneous artificial intelligence (AI) methods including both statistical and symbolic approaches in various domains is omnipresent in current scientific literature. It is largely overlapping with the term hybrid intelligence (HI) that has been defined as *"the combination of complementary heterogeneous intelligences (...) to create a socio-technological ensemble that is able to overcome the current limitations of (artificial) intelligence."* [16]. HI lies at the intersection of human, collective and

artificial intelligence, with the intent of taking the best of each.

There have been numerous studies recently addressing issues related to HAI and HI methods in a multitude of application domains including but not limited to land-slide prediction [33], drug testing [12], forecasting crude oil prices [78], prediction of wildfire [24], evaluation of slope stability [31], modeling of hydro-power dam [9], wind energy resource analysis [21], industry 4.0 and production automation [5], airblast prediction [4], heart disease diagnosis [36] and these are just a few references from 2018 until the time of writing this proposal. Most of these and such studies report building HAI systems by combining various AI methods to acquire better and more precise results. However, when it comes to methodology of the actual orchestration of HAI methods the usual approach is ad-hoc and depends from project to project. The lack of methodology in orchestrating HAI shall be addressed in the proposed project.

In a previous project sponsored by the Croatian Science Foundation (Installation Project No. HRZZ-UIP-2013-11-8537 entitled Large-Scale Multi-Agent Modelling of Massively Multi-Player On-Line Playing Games - ModelMMORPG - see [67] for details) a comprehensive methodology for modelling large-scale intelligent distributed systems has been developed that includes a graphical modelling tool and code generator (described in [66] and in more detail in [43]). The implemented toolset allows for modelling complex multi-agent organizations and could be applied to numerous applications domains [60, 61]. Herein, we would like to apply and incorporate this methodology to the development of the HAI orchestration platform.

Computer games have always been connected to the development of AI. From the earliest chess minmax algorithm by Claude Shannon in 1949 to the more recent AlphaGo™ in 2015, computer games provide an ideal testing environment for AI methods. Similarly, AI has always been an important part of computer games. Computer games have often been judged by the quality of their AI and praised if they used an innovative approach like the ghosts in Pacman™ which had individual personality traits (1980), Creatures™ which used neural networks for character development (1996), Black & White™ which used the belief-desire-intention (BDI) model (2000), F.E.A.R.™ which used automated planning algorithms (2005) and many others (see [82, pp. 8–15] for a very detailed overview). Artificial intelligence in games is not only used for non-player character (NPC) or opponent implementation, but also for various other parts of games [82, pp. 151–203] including but not limited to **generation of content** (graphics including levels and maps, sound, narratives, rules and mechanics or even whole games like the Angelina game-generating system [14]), **player behaviour and experience modeling** [82, pp. 203–259], as well as **bot development and automated game testing** [82, pp. 91–151]. Due to their complex nature and endless possibilities of creative design, computer games present us with an excellent opportunity to study the orchestration of HAI in various scenarios – not only for fun and leisure but also for other domains in form of serious games and/or gamification.

In the previously mentioned ModelMMORPG project, we have already used an open source massively multi-player on-line role-playing game (MMORPG) called The Mana World (TMW) for which we have implemented a high-level interface to test intelligent agents playing the game. Additionally a number of connected game quests have been developed for various scenarios which allowed us to build an automated game testing system [65]. Herein we would like to use this interface to test orchestrated HAI methods, but also develop other testbeds for the planned platform.

Therefore, the main contribution of the proposed project shall be: (1) a comprehensive framework for the orchestration of hybrid artificial intelligence methods for computer games allowing to define models of HAI for various purposes, (2) an open source distributed cloud platform that will allow to implement such models based on existing HAI methods and connect them directly from game development platforms, (3) a set of best practices in developing HAI ensemble models tested in at least four specific testbeds.

The integration of HAI methods, which combines both symbolic and statistical approaches,

offers significant advancements in various domains, particularly in computer games and smart mobility. This project aims to develop a framework for the orchestration of HAI methods and test it across multiple gaming-related environments. The chapters that follow will delve into specific applications and developments within this framework, demonstrating the versatility and effectiveness of HAI in different contexts.

In the next chapter, we will explore the concept of Digital twin (DT) in the context of autonomous vehicle (AV). This chapter lays the groundwork by discussing how DTs can benefit from the orchestration of HAI methods, particularly in creating accurate and responsive virtual models that interact with real-world data.

1.3 Team Members

Markus Schatten (Principal investigator)
Head of Artificial Intelligence Laboratory,
Faculty of Organization and Informatics,
University of Zagreb

Jaime Andres Rincon Arango
Grupo de Tecnología Informática e Inteligencia Artificial (GTI-IA),
Departamento de Sistemas Informáticos y Computación,
Universitat Politècnica de València

Bogdan Okreša Đurić
Artificial Intelligence Laboratory,
Faculty of Organization and Informatics,
University of Zagreb

Carlos Carrascosa
Grupo de Tecnología Informática e Inteligencia Artificial (GTI-IA),
Departamento de Sistemas Informáticos y Computación,
Universitat Politècnica de València

Damir Horvat
Department of Quantitative Methods
Faculty of Organization and Informatics,
University of Zagreb

Vicente Julian
Grupo de Tecnología Informática e Inteligencia Artificial (GTI-IA),
Departamento de Sistemas Informáticos y Computación,
Universitat Politècnica de València

Tomislav Peharda
Artificial Intelligence Laboratory,
Faculty of Organization and Informatics,
University of Zagreb

Glenn Smith
College of Education,
University of South Florida

Igor Tomičić
Center for Forensics, Biometrics and Privacy
Faculty of Organization and Informatics,
University of Zagreb

Neven Vrček
Department of Information Systems Development
Faculty of Organization and Informatics,
University of Zagreb

2. Digital Twins and Autonomous Vehicles

In this chapter, we offer a game development perspective on DTs, focusing on smart and autonomous vehicles. A DT, which includes a virtual representation of a system throughout its lifecycle, utilizes real-time input, simulation, reasoning, and machine learning to facilitate decision-making about the actual system. This concept is described as an agent or actor within a simulated (game) environment. We provide a conceptual model for developing DTs within an orchestration platform for hybrid artificial intelligence services designed for complex game engines.

2.1 Introduction

DTs refer to virtual models that describe observed physical systems as accurately as possible. The distinction between a DT and a simple simulation lies in the robustness of a DT concerning the variety of processes that can be analyzed, as well as DTs often working with real-world data from the simulated system. For instance, a simulation might target only a single parameterized use-case for analysis, whereas a DT acts as a virtual environment capable of running all use-cases that a physical model might encounter [49, 79]. Classically defined, a digital twin is "a virtual replica of a real-world product, system, being, communities, even cities, that are continuously updated with data from its physical counterpart, as well as its environment" [25]. Compared to a "classical" model, which is defined as a simplification or abstraction of the real system, a DT can be seen as a complete mirror image model.

A DT model could be developed for any complex system. Examples include wind turbines, AVs, aircraft tracking, and more. Several authors consider DTs as the backbone of industry [13, 25, 29, 39]. Although the term was coined about 20 years ago, it has recently drawn significant attention due to digital infrastructure becoming more integrated into industry, cities, communities, and daily activities [6].

In this chapter, we will present the benefits and challenges of using an orchestrated game development platform for implementing DTs of AVs, including the elimination of duplicated and "boilerplate" code as well as the reuse potential of the original AV system code due to the introduced modularity and process distribution. However, such benefits might come with the cost of additional computing resources.

2.2 Digital Twins in Autonomous Vehicles

Recently, there has been a continuous increase in research papers focused on using DTs for AVs. For instance, [49] investigated how the DT concept can be advantageous for managing propulsion in an electric vehicle. The concept involves a physical vehicle equipped with various sensors and measurement systems that share propulsion information with a service system connected to the DT environment. The service system conducts simulations within the DT environment and uses the feedback to control the vehicle. Similarly, [79] proposed the use of the DT concept with a focus on traffic modeling. They emphasize that accurate traffic simulations are crucial for AVs's safety and performance, as different traffic dynamics require the vehicle to adapt its driving style accordingly. Additionally, the DT concept can be utilized for improving the economics of vehicle aspects, such as fuel consumption, driver assistance systems, battery management systems, and vehicle power electronics [8].

Vehicle safety and security is another critical domain requiring significant attention [2]. Research highlights scenarios where a DT ensures consistent vehicle behavior in the event of a cyber attack that could compromise data integrity and consistency, potentially leading to traffic accidents. DTs are proposed to simulate data collection, data processing, and analytics to address these concerns.

In the realm of autonomous vehicles alone, numerous aspects necessitate proper monitoring and forecasting, which can be supported by the use of DTs. This indicates that developing DTs is a complex task requiring meticulous planning and system architecture. AI and particularly machine learning (ML) are highly advocated in this domain as they help predict and propose future actions based on current circumstances. However, the use of these methods adds another layer of complexity. As components evolve, it is essential to ensure that DT components are extendable and modular to adapt to new requirements easily. For instance, the increasing number of AVs may suggest the need for a different development paradigm [2, 8]. In this chapter, we will address these challenges related to the need for complexity reduction and modularization of AV and DT architectures.

2.3 Awkward Π-nguin Orchestration Infrastructure

In [58], we introduced a high-level conceptual model of our microservice orchestration platform, depicted in Figure 2.1. We have partially implemented this platform as Awkward Π-nguin (APi), an open-source, declarative agent-based programming language based on Π calculus [40] (a process calculus) that allows us to model communication flows between microservices.[1]

APi is agent-based, meaning microservices are represented as autonomous agents, and is also holonic [53], meaning agents can be organized hierarchically as holons. The APi platform features a declarative engine implemented in Python, particularly using Smart Python Agent Development Environment (SPADE) [46], with parts implemented in ANTLR (syntax parser) [48] and BASH shell scripts (for inter-process communication). It is programming language agnostic, allowing microservices to be implemented in any language that can be executed in UNIX-friendly environments, and communication protocol agnostic, enabling services to communicate using stdin/stdout process communication, files, HTTP, TCP, UDP, or WebSockets, with additional protocols addable through a partially implemented plug-in system. This is achieved by creating wrappers around each microservice to be used in an ensemble. These wrappers then behave as agents within a holonic multi agent system (HMAS) and can serve as building blocks for creating complex architectures. Essentially, APi is a high-level network implementation of the UNIX inter-process communication system, based on input/output redirection where special programs (filters) create complex processing chains. Each process reads the standard output of the previous

[1] APi is open source and available at `https://github.com/AILab-FOI/APi`

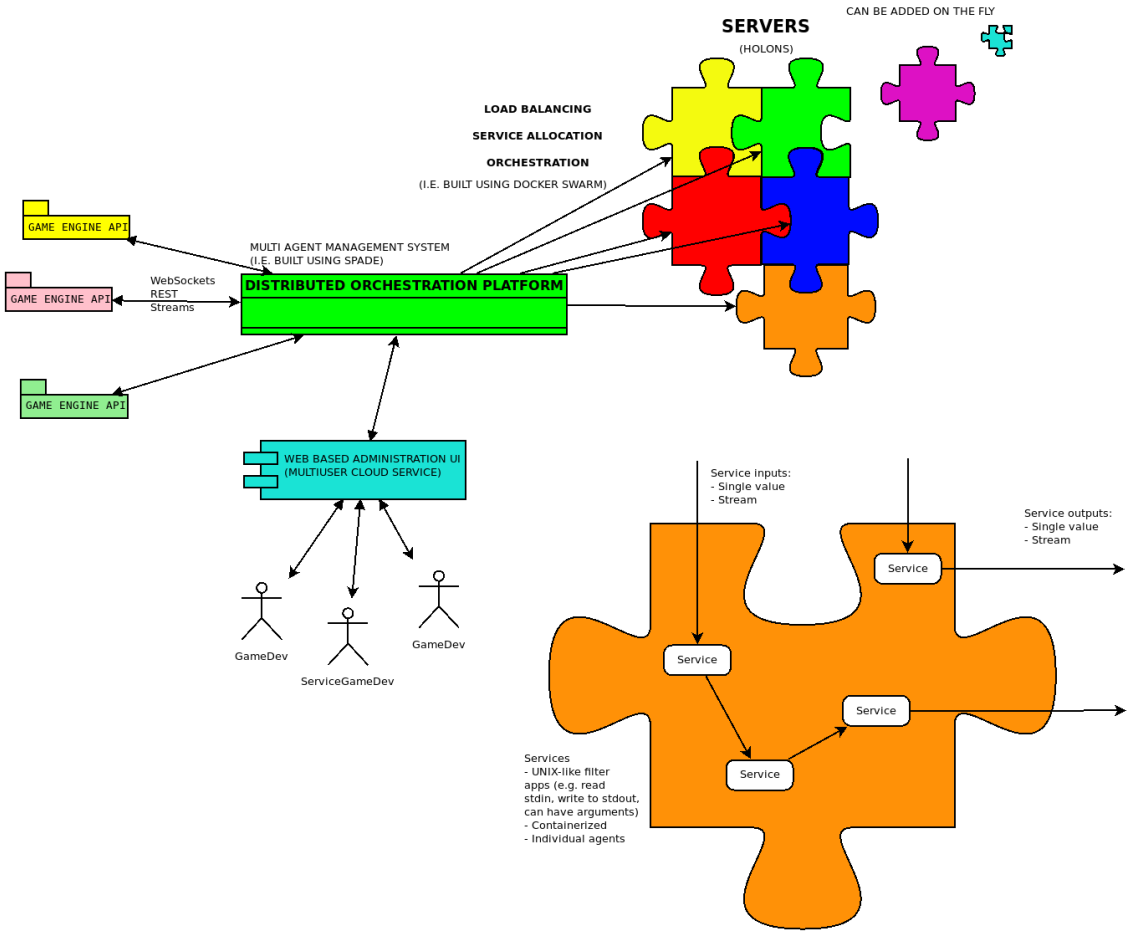


Figure 2.1: High-level Conceptual Model [58]

process as its input and "filters" it to produce new output for the next process. The same basic idea applies to APi, but with the added capability of distributing these processes (microservices or agents) across various servers and enabling them to communicate over a network regardless of their implementation.

The syntax of the APi programming language is influenced by Python and Π-calculus. The following excerpt (Listing 2.1) demonstrates some of the main features of the language.

Listing 2.1: Example APi syntax

```
environment :
 input1 => { 'val1': ?x, 'val2': ?y }
 output1 <= { 'action': ?act }

channel c :
 { 'data'->?x } -> { 'payload' -> ?x }

agent a ( x ) :
 input1 -> self
 self -> ?x

agent b :
 c -> self
```

```
 self -> output1

start a( c ) & b
```

In this excerpt, a holon is defined, which communicates with the environment through channels `input1` and `output1`. A local channel (`c`) is defined to transform messages of the form `{ 'data'->?x }` (JSON format with logic variables) on the left-hand side and emit messages of the format `{ 'payload' -> ?x }` on the right-hand side. Additionally, two agents (microservice wrappers) are defined (`a` and `b`). Agents' definitions correspond to service descriptors that describe how to start the service, how it communicates, what type of service it is, etc., and can take arguments. In each agent definition, there are one or more channel mapping definitions. For example, agent `a` will direct messages from `input1` to its own input (keyword `self`) and redirect its outputs to the channel held in the variable `?x` (an instantiation argument). Both agents are started consecutively (when agent `a` finishes successfully, agent `b` will be started). The defined holon can then be imported into other holons to create complex orchestration architectures. For example, if the above holon had been defined in a file called `holon1.api`, we could import and use its output as shown in Listing 2.2.

Listing 2.2: Example APi syntax

```
from holon1 import output1 as h1

agent a ( ... ) :
     h1 -> self
```

The platform is designed for the orchestration of AI and computer game-related microservices as outlined in [57] and [63]. In this regard, it enables the connection and assembly of complex data streaming applications, which is essential for the implementation of AV systems and consequently DTs.

2.4 Digital Twins as Game Actors

A common software development pattern in computer game design is the actor model [34, 70], which corresponds to the multiagent systems (MASs) model or more precisely to intelligent virtual environments (IVEs) [42, 51]. In such environments, actors (agents) interact with their environment, which consists of other (potentially intelligent and/or autonomous) agents as well as dynamic or inanimate, static objects.

It has been argued that IVEs provide the necessary tools to develop, test, and monitor AVs architectures [19, 57, 83]. In this chapter, we further argue that a serious game development environment with an orchestration platform, as described in Section 2.3, can offer additional benefits. Firstly, game development environments usually feature an integrated game engine that provides essential components for developing IVEs, including but not limited to 3D physics, various sensors, a simulation engine, AI methods, actor templates, cameras, logging facilities, etc. Additionally, these environments offer instant visual monitoring of the environment. Secondly, an orchestration platform allows us to implement complex data streaming ensembles that might consist of various ML cloud services, which are usually too resource-intensive to be executed on a single computer during a simulation and/or real-time monitoring.

To implement a DT of an AV, real-time monitoring of the vehicle in terms of sensor inputs, actuator outputs, and internal decision-making mechanisms is crucial. With an orchestration platform where multiple cloud-based servers can handle various complex tasks like pattern recognition, automated planning, or reasoning, such a workload can be distributed and managed. In the following,

we will demonstrate a simple proof-of-concept implementation of a DT of an AV using the APi language.

An implementation of an AV system typically consists of several sensors (which might include LiDAR, radar, cameras, etc.), several decision-making services (e.g., ML models, complex AI algorithms including but not limited to search and planning techniques), several communication services (interacting with relevant online services like traffic congestion information, routing services, etc.), and several actuators (e.g., steering, car safety monitoring like tire pressure, fuel or battery life, infotainment systems, etc.). All these subsystems can be wrapped into mutually communicating agents inside a holon, as described in Section 2.3.

For simplicity, let us assume that the AV is a holon (i.e., a multi-agent system of various components) that communicates by reading its sensors and relevant external services and writing commands to appropriate actuators. An implementation of such a holon in APi would look similar to the code in Listing 2.3.

Listing 2.3: AV holon implementation - `AVHolon.api`

```
environment :
     sensor_1 => ...
     ...
     sensor_n => ...
     external_service_1 => ...
     ...
     external_service_k => ...
     actuator_1 <= ...
     ...
     actuator_m <= ...

agent AV :
     sensor_1 -> self
     ...
     sensor_n -> self
     external_service_1 -> self
     ...
     external_service_k -> self
     self -> actuator_1
     ...
     self -> actuator_m
```

By defining the AV system in this way, we abstract away all internal communication mechanisms, meaning that we now have standard communication interfaces (channels) for all included components, and these interfaces can be reused. We can redirect them at will to other agents or holons. This redirection capability is a special feature of the declarative engine of APi—it allows us to listen to or write to any channel within the given scope (i.e., the current holon, which in this case acts as a namespace). This feature is crucial for implementing a DT since it enables us to "tap into" communication flows and process real-time data.

To gain the benefits of a DT, we should additionally add a corrective input, i.e., an additional communication channel to allow for intervention if the AV system does not function as intended. The environment would then look as shown in Listing 2.4.

Listing 2.4: Adding a corrective to the AV holon

```
environment :
```

```
     corrective => ...
     sensor_1 => ...
     ...
...
```

With the prerequisites defined, implementing a DT using APi is now straightforward. We first import all communication channels and redirect them to a new agent representing the DT. Finally, we start the DT in parallel with the AV holon defined above, as shown in Listing 2.5.[2]

Listing 2.5: Digital twin implementation

```
from holonAV import sensor_1 as s1
...
from holonAV import sensor_n as sn

from holonAV import external_service_1 as e1
...
from holonAV import external_service_k as ek

from holonAV import actuator_1 as a1
...
from holonAV import actuator_m as am

from holonAV import corrective as cr

agent AVtwin :
     s1 -> self
     ...
     sn -> self
     e1 -> self
     ...
     ek -> self
     a1 -> self
     ...
     am -> self
     self -> cr

start holonAV | AVtwin
```

In this way, the DT can monitor all inputs and outputs of the original AV system and use this data to update its IVE, simulate and analyze its decision-making process, and send corrective information if needed.

2.5 Discussion

The primary advantage of the orchestrated implementation described in Section 2.4 is its modular approach, which significantly reduces code duplication, especially in "boilerplate" communication code. In a non-orchestrated implementation, each service instance from the original AV system that communicates in any way would need to be either rewritten or extended to forward data to the DT. Additionally, for the implementation of the DT, each component simulation of the original system

[2]The | character between two agents denotes parallel execution.

would have to be recreated in an IVE, game engine, or other simulation environment using native development tools and languages. By employing agent wrappers around these components, most of the non-hardware-specific code can be reused, functioning similarly to the original setting.

However, there are some caveats. Wrapping the original AV system components as agents introduces additional overhead code, which might require extra computing resources to operate and communicate with the orchestration platform. These additional resources might not be available or desirable in real-time and mission-critical settings. Another issue that remains unresolved with this approach is the need to implement the update process of the IVE using real-time data from the original AV system. These update components need to be implemented as additional services to update the simulated environment based on sensory data and integrate with the other (potentially reused) services. On the other hand, the integration process is relatively straightforward due to the modularity of APi, allowing for the introduction of new components without altering the original services' code, which might not be feasible in a non-orchestrated setting.

2.6 Conclusion

In this chapter, we have discussed the benefits and challenges of using an orchestrated game development platform for implementing DTs for AVs. Using the APi platform we are actively developing, we demonstrated how to implement a basic DT ensemble. The key advantages of using an orchestrated game development platform, besides the various building blocks provided by game development software (especially game engines), include the elimination of duplicated and "boilerplate" code, as well as the reuse potential of the original AV system code due to the introduced modularity and process distribution. These benefits come with the cost of requiring additional computing resources on the AV system side.

Our future research aims to implement a proof-of-concept digital twinning environment for AVs, addressing some of the issues outlined above.

We have highlighted the potential of DTs in enhancing the functionality and safety of AVs. The orchestration platform for HAI methods is crucial in managing complexity and ensuring the accuracy of DTs. Moving forward, we will explore how similar principles apply to developing cognitive agents in smart mobility, showcasing another facet of HAI application.

3. Cognitive Agents & Smart Mobility

Cognitive agents are artificial intelligence systems capable of communicating in ways that are highly acceptable to humans. Technologies such as natural language processing (NLP), text to speech (TTS), speech to text (STT), and motion capture (MoCap) are typically employed to provide such interfaces. In this work-in-progress chapter, we present the current state of the B.A.R.I.C.A. infrastructure that we have developed. This infrastructure facilitates the implementation of open-source cognitive agents across a wide range of domains, capable of communicating in the Croatian language. The aim of this study is to analyze potential applications of B.A.R.I.C.A. in smart mobility, identify gaps to be addressed, explore possible applications, and provide implementation guidelines.

3.1 Introduction

A cognitive agent is a specific class of intelligent agents employing numerous AI methods, including but not limited to, ML and deep learning (DL) models, STT and TTS technologies, NLP, knowledge bases (KBs), BDI models, as well as system automation for interacting with and learning from humans [32]. Cognitive agents have gained significant popularity through various systems developed by major software vendors such as Google (*Assistant*), Microsoft (*Cortana*), Apple (*Siri*), and Amazon (*Alexa*). Their application domains include Internet of things (IoT) and fog computing [20], education [7], home service robots [75], mental health therapy [72], cognitive radio [41], and many more.

Smart mobility, representing a network of intelligent services, systems, processes, models, and people aimed at making transportation easier, combines various technologies and elements of mobility, enabling us to rethink the transportation infrastructure used in daily life and business.

In this chapter, we will present the current state of the B.A.R.I.C.A. cognitive agent infrastructure that we are actively developing [55] and demonstrate how it can be used to implement smart mobility applications. Additionally, this study will use the Design Science Research Methodology (DSRM), which provides specific guidelines for evaluation and iteration within research projects in information technologies [47]. This chapter focuses on the first three activities of the design science research process: (1) Problem identification and motivation (providing a selection of application

areas for cognitive agents in smart mobility), (2) Objectives of a solution (describing problems that can be solved using our infrastructure), and partly (3) Design and development (outlining the implementation steps needed to accomplish a solution).

3.2 B.A.R.I.C.A. Infrastructure

The primary motivation behind the B.A.R.I.C.A. system was to create an open-source cognitive infrastructure for developing cognitive agents using the Croatian language. So far, we have been able to implement the basic infrastructure as well as two use-cases (one as a presentation assistant agent and another for university student support [55, 71]). The B.A.R.I.C.A. cognitive agent system's software architecture is shown in Fig. 3.1. The two main components are a cloud-based back-end and an on-site front-end.

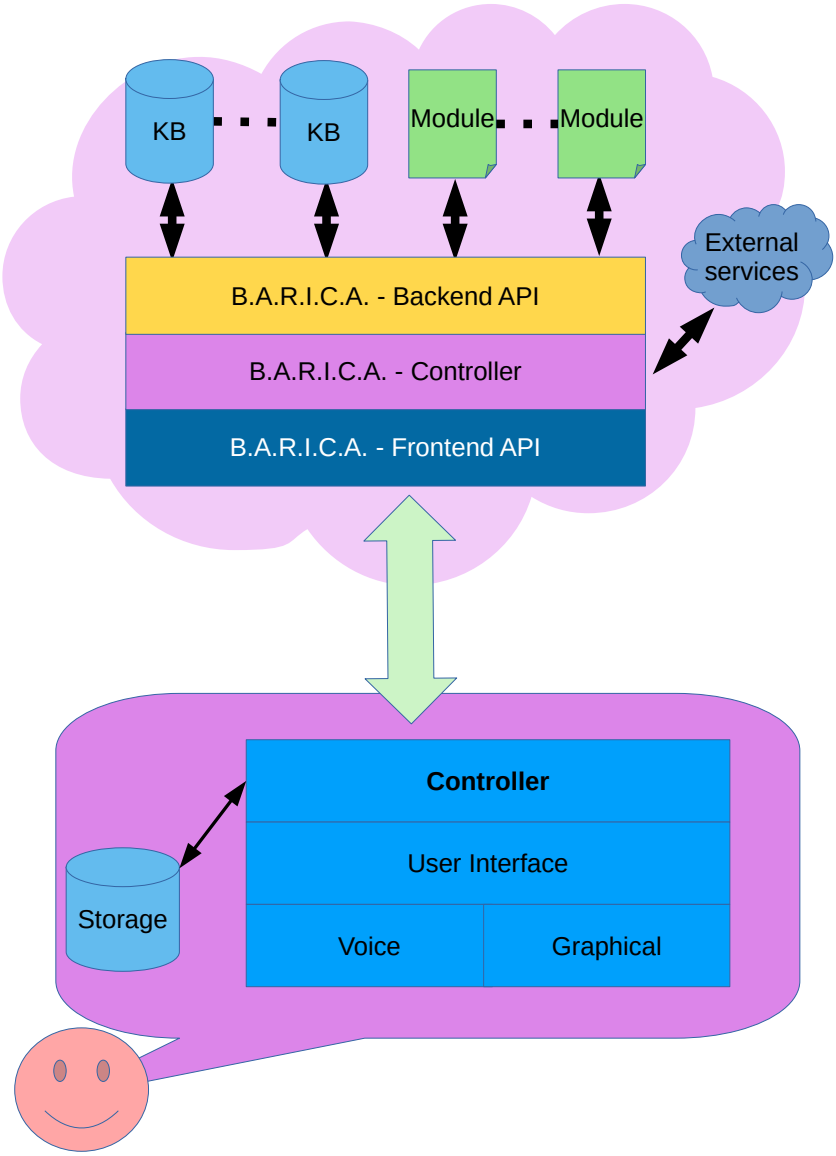


Figure 3.1: B.A.R.I.C.A. software architecture [55]

The cloud-based back-end is part of a larger framework that encompasses a microservice

orchestration platform embodying concepts of HMASs [53] and large-scale multiagent systems (LSMASs) [60, 61] through its main components: a back-end application programming interface (API), a controller, and a front-end API. This framework is being developed by the Orchestration of Hybrid Artificial Intelligence Methods for Computer Games project (O_HAI (4) Games) [58, 63]. Each component serves a specific purpose:

1. The back-end API provides access to various microservices, including but not limited to, KBs, AI modules, and external services. Thus, the back-end API can be considered the main source of intelligent actions that the overall system is capable of.
2. The controller, which serves as a middle layer, acts as a microservice orchestration system aimed at combining various back-end microservices with front-end API functions to provide a coherent system.
3. The front-end API offers front-end applications easy-to-use functionality, abstracting away all logic related to microservice orchestration.

The technologies used to build the cloud-based back-end are primarily various Python-related technologies. The cognitive features of B.A.R.I.C.A. are implemented using the NLTK and Chatterbot Python modules, providing chatbot and NLP capabilities. To transform text into speech, the external service VoiceNotePad has been utilized.

The main front-end application is web-based, relying on JavaScript technologies such as jQuery and Selenium for browser automation. The Python library Hovercraft has been used to provide the graphical user interface (GUI). To support MoCap animations for speech sequences, CrazyTalk has been used. B.A.R.I.C.A. is open source and available for free to anyone who wishes to implement cognitive agents.

3.3 Possible Applications to Smart Mobility

There have been numerous applications of smart mobility in both industry and academia. Here, we provide a brief selection of applications that might benefit from cognitive agents based on a literature review [18, 23, 76]. The applications have been chosen based on two criteria: (1) ease of implementation using the B.A.R.I.C.A. infrastructure, and (2) potential benefits for end-users if such services are implemented. This list is not exhaustive, and other application areas could also benefit from cognitive agents.

Additionally, we provide guidelines on how such services might be implemented using the B.A.R.I.C.A. infrastructure, focusing on components that might be missing and need to be integrated or developed from scratch. Fig. 3.2 provides a conceptual overview of the various components that need to be employed.

Electronic toll systems are systems that allow for the automation of toll payments, usually using an embedded computing system connected to various traffic equipment like ramps or card recognition systems. Cognitive agents can provide a natural language interface, automated detection of vehicle types and/or passengers, automated payment, etc. To use the B.A.R.I.C.A. infrastructure in electronic toll systems, as well as in any other embedded hardware, a new embedded interface might need to be developed depending on the type of embedded software and operating system used. In the case of an advanced microcontroller like the RaspberryPi, which runs a full Linux operating system, the standard interface can be used since it relies only on standard operating system software. In other cases, where no browser is available, a new interface controller and likely a new user interface would need to be developed. Additionally, a new module for electronic toll systems would need to be developed and connected to the back-end API to allow the cognitive agent to understand the domain and be used effectively. Other services, such as a payment system, can be developed as part of the electronic toll system or as an additional module. From a hardware perspective, a

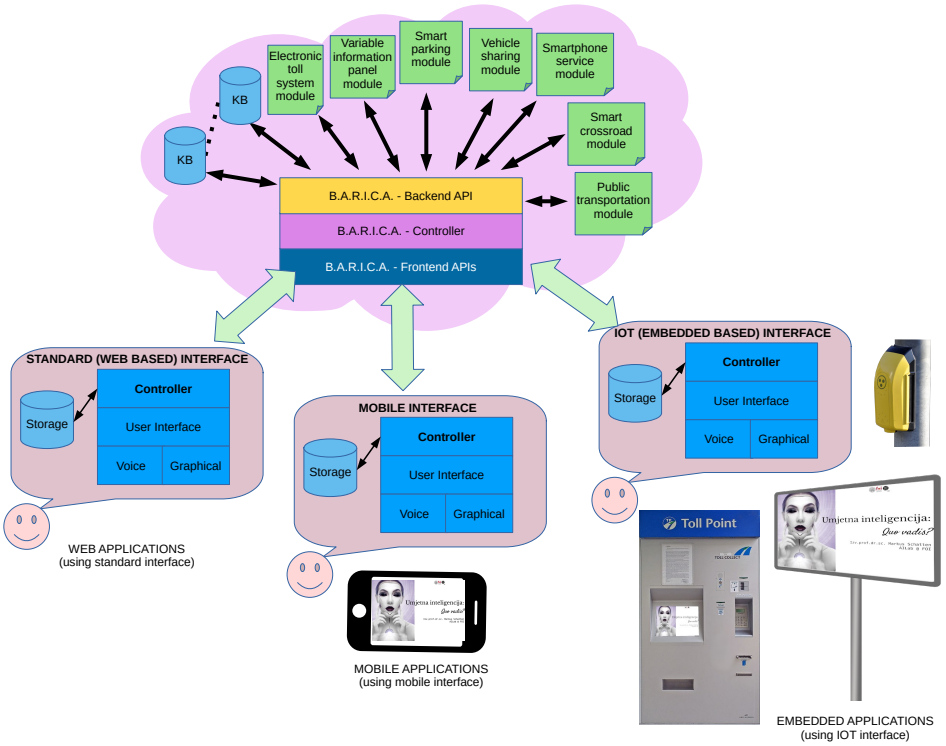


Figure 3.2: B.A.R.I.C.A. software architecture for Smart Mobility

microphone and a loudspeaker need to be added to the installation if none are available.

Variable information panels can be enhanced with cognitive agents to provide a natural language interface and two-way communication. To use the B.A.R.I.C.A. infrastructure in variable information panels, assuming a PC or more advanced microcontroller is driving the panel, no changes need to be made to the client. An additional module for the information domain of the panel must be implemented and connected to the back-end API. If the panel lacks a microphone and loudspeakers, these need to be installed to enable two-sided communication. Since variable information panels are primarily designed to provide visual information, introducing a cognitive agent would allow the panels to also provide information via audio, with the possibility of answering individuals' questions. Thus, expanding the pool of information that may be provided through the panel.

Smart crossroads To use the B.A.R.I.C.A. infrastructure in smart crossroads, a new embedded interface might need to be developed depending on the type of device that the crossroad is intended to cover. A smart crossroads module would need to be developed and implemented into the back-end API to cover features for the said domain. Depending on the use case, various sensors might be needed to track events at a crossroad, which the module would need to account for. In this context, a cognitive agent could assist people with visual disabilities by informing them when it is safe to cross the road.

Smart parking lots To use the B.A.R.I.C.A. infrastructure in smart parking lots, they should be equipped with sensors to monitor available spots, and a module that accesses that information. The cognitive agent could help navigate individuals to an empty parking slot and assist with ticket payments. These modules would then be integrated with the back-end API to enable the cognitive agent to access parking lot availability information. This implies that a parking lot would also need a device for user interaction. Depending on the types of devices available,

the client may or may not need to change.

Vehicle sharing To use the B.A.R.I.C.A. infrastructure in vehicle sharing, the vehicle should have a built-in smart device. Most new vehicles already include such devices, though their software is usually not flexible enough to add additional applications. Therefore, a new client may need to be developed to work with the available software. Cognitive agents can enhance the vehicle-sharing experience by assisting multiple individuals who share the vehicle with specifying destinations, answering questions about arrival times, helping with payments, etc. To support two-sided communication, a microphone should be included if none is present, though most newer vehicles already have them. For this use case, a separate module needs to be developed and integrated with the back-end API, providing support for vehicle-sharing features such as navigation and payment handling.

Public transport To use the B.A.R.I.C.A. infrastructure in public transport, assuming there are smart devices (monitors or TVs) present, no changes are required on the client side since such smart devices already support Internet browsers. However, additional modules for the transport domain would need to be integrated into the back-end API to support these use cases. In public transport, a cognitive agent could help users purchase and verify their tickets, provide information about possible routes, check arrival times at destinations, and similar use cases that may potentially combine information from different transport modes.

3.4 Conclusion & Future Research

In this chapter, we have provided an overview of the B.A.R.I.C.A. cognitive agent infrastructure, which is an open-source platform for developing cognitive agents in the Croatian language. We are actively developing this infrastructure as part of the O_HAI (4) Games project and a larger context dealing with smart microservice orchestration.

As part of a design science research methodology process, we have developed a conceptual overview model of various components that can be employed to implement smart mobility applications using B.A.R.I.C.A., as well as a number of possible use cases for this infrastructure and cognitive agents in general, focusing on the benefits for end-users. Additionally, we have provided guidelines on how such applications could be implemented using our infrastructure.

While B.A.R.I.C.A. is open source and free to use, it still includes several constraints for its usage in smart mobility applications, mainly concerning the current client interface, which heavily relies on a specific operating system (Linux) and browser automation (Chromium, Selenium). Thus, to fully leverage the infrastructure, new client interfaces need to be developed that are either platform-specific or, preferably, platform-independent, making them available on most platforms.

Our future research aims to build such interfaces, especially with the goal of making them platform-independent, to reach as many end-user platforms as possible. One possibility we are considering is using web assembly or pure JavaScript to embed the main functionality of the client interface into any browser, allowing usage on any device that can run a modern Internet browser.

Cognitive agents, powered by advanced AI techniques, offer transformative potential in smart mobility applications. From enhancing user interactions to automating complex processes, these agents exemplify the practical benefits of HAI orchestration. The next chapter will shift focus to game streaming and its implications for intelligent transport, demonstrating how HAI can revolutionize user experiences in gaming and beyond.

4. Game Streaming & Intelligent Transport

In this chapter, we present and analyze a game streaming engine called Lag but Good Game (laGGer), which is based on open-source technologies, standards, and protocols. We detail the system's architecture, which is grounded in multi-agent systems, and provide insights from the design and development process of this complex system. We argue that game streaming systems offer new and unprecedented opportunities in transport and mobility research, and we illustrate several use cases demonstrating how such systems can facilitate emerging applications.

4.1 Introduction

The rapid growth of cloud computing and network technologies has led to the emergence of game streaming services, enabling users to play high-quality games without requiring powerful hardware. Commercial game streaming systems such as Nvidia GeForce Now [73], Microsoft xCloud [37], the now-defunct Google Stadia [11], and others have gained significant attention and popularity in recent years. However, these platforms often rely on proprietary software and protocols, limiting the accessibility and adaptability of the technology. To date, no game streaming system has been built exclusively using open-source software, open protocols, and standards.

In this chapter, we introduce a novel game streaming engine called laGGer, which is entirely based on open-source technologies, aiming to increase the accessibility and flexibility of game streaming services. This engine is designed using a design science approach, leveraging insights from the complex process of developing multi-agent systems and containerized microservice architectures. Our system enables the creation of game instances that are streamed to a web-based client interface, allowing users to experience high-quality gaming on various devices.

Moreover, our orchestrated game streaming system facilitates user interaction by providing audio and video streaming between users. This feature enhances the overall gaming experience by enabling seamless communication and collaboration among players. By utilizing open-source software, open protocols, and standards, our system offers an alternative to commercial game streaming services, fostering innovation and promoting the development of new game streaming applications and platforms.

In the dynamic field of transport and mobility research, innovative digital technologies are continuously explored for their potential to enhance both the depth and breadth of research capabilities. Game streaming technology, a relatively recent development in the digital landscape, presents a unique set of opportunities for this domain. This technology, which allows for real-time streaming of interactive game content from a server to a client device, has the potential to revolutionize the way transport and mobility studies are conducted.

The inherent characteristics of game streaming technology, such as real-time data transmission, high accessibility, and user interaction capacity, make it a versatile tool for researchers and professionals in the transport sector. It opens up new avenues for creating immersive, realistic, and scalable simulations of various transport scenarios, facilitating advanced data collection and analysis, and fostering effective training and educational programs.

Moreover, the ability of game streaming technology to support real-time communication between users adds an additional layer of interactivity, enabling collaborative scenarios and studies on group dynamics in transport situations. This aspect is particularly relevant in the context of studying and developing solutions for connected and automated mobility systems.

In the following sections, we delve into the system's architecture, detailing the design and development process of this unique game streaming engine. Firstly, in section 4.2, we provide an overview of related work. Then, in section 4.3, we describe the architecture of the implemented system. In section 4.4, we present an overview of mobility and transport applications of game streaming technology, and in section 4.6, we draw our conclusions and provide guidelines for future research.

4.2 Related Work

Game streaming, also known as cloud gaming, is a technology that allows users to play video games remotely by streaming game content from powerful servers to their devices. This enables gamers to enjoy high-quality graphics and gameplay without needing powerful and expensive hardware on their end. The motivation behind game streaming comes from the desire to make gaming more accessible and convenient for users. By offloading the processing and rendering tasks to remote servers, game streaming allows players to enjoy resource-intensive games on less capable devices such as smartphones, tablets, or low-end computers. This approach also enables gamers to play their favorite games without worrying about hardware upgrades or compatibility issues.

Game streaming technology leverages a combination of cloud computing, high-speed internet, and video streaming protocols to deliver an optimal gaming experience. Cloud servers equipped with powerful GPUs and CPUs process and render game content, while low-latency video streaming protocols such as WebRTC, H.264, or VP9 transmit the gameplay to the user's device in real-time. Input from the user's game controller or keyboard is sent back to the server, allowing for real-time interaction with the game.

Microservice orchestration [62] using software containers, such as Docker, refers to the process of automating, managing, and coordinating the deployment, scaling, and intercommunication of microservices in a distributed system. Microservices [74] are small, independent, and loosely-coupled services that together form a larger, more complex application. Containerization, using technologies like Docker, packages each microservice with its dependencies and runtime environment, ensuring consistency and isolation across various stages of development and deployment. Examples of microservice orchestration using containers can be found in various domains, such as e-commerce platforms, content management systems [15], and even game streaming services as in our case [56, 64]. Companies like Netflix, Uber, and Spotify have adopted microservices and containerization technologies to manage their large-scale, distributed applications.

MASs are a branch of artificial intelligence that focuses on the design, modeling, and implementation of complex distributed intelligent systems composed of multiple autonomous agents. These

agents, which can be software programs or physical entities, interact and collaborate with each other to achieve common or individual goals [54, 80]. By leveraging the collective intelligence of multiple agents, MASs can address complex problems that are difficult or impossible to solve using a single-agent approach. The motivation behind multi-agent systems comes from the need to model and implement complex distributed intelligent systems that can adapt to dynamic environments, handle uncertainties, and autonomously solve problems [1, 10].

Examples of multi-agent systems can be found in various domains, including robotics (swarm robotics [17], cooperative robot teams [52]), traffic management (autonomous vehicles [26], traffic control, e.g., [81]), logistics (supply chain management [22], distributed scheduling), and even game development (NPC - Non-Playing Character - coordination [44], dynamic game environments).

4.3 System Architecture

The system's architecture is depicted in Figure 4.1. As shown, the system comprises a game streaming agent that orchestrates a pool of game agent containers. Additionally, a videoconferencing agent handles the establishment of video conference instances for each game session using Web Real-Time Communication (WebRTC) connections managed by a Janus [3] server.

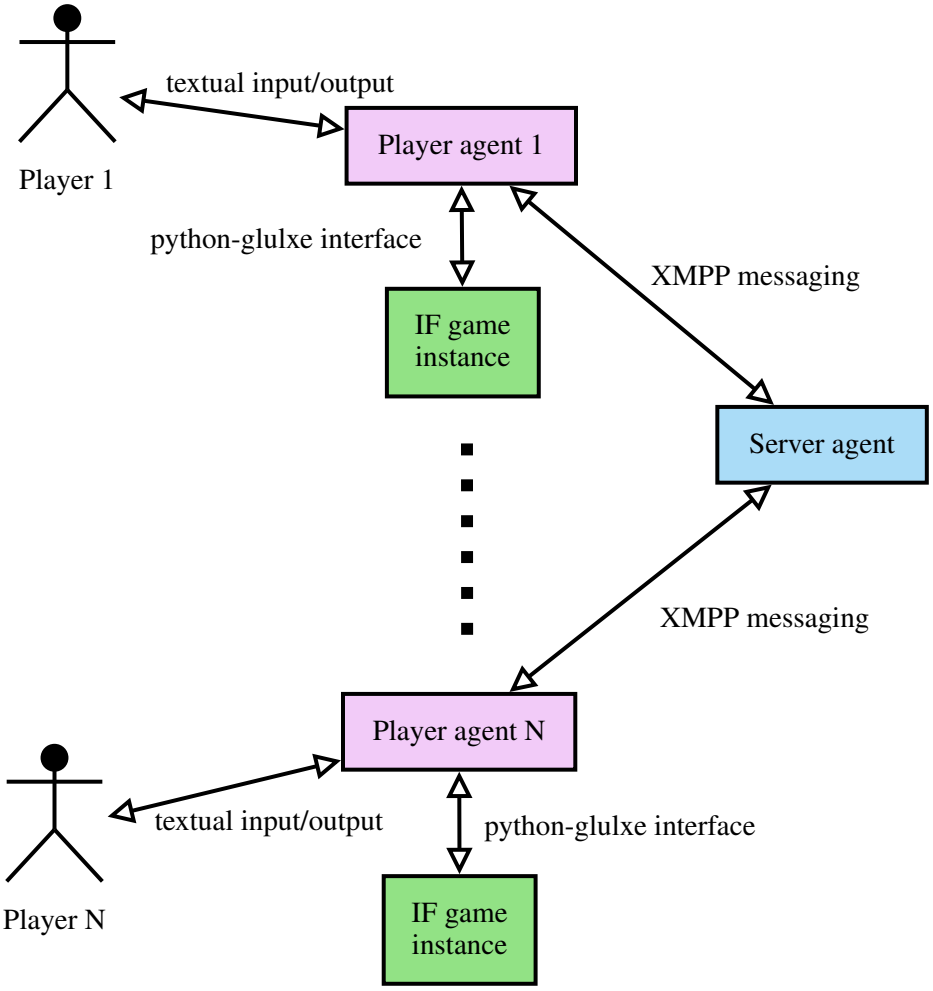


Figure 4.1: System Architecture [56]

Game agents are implemented using Docker, enabling different types of services to run in isolated environments, referred to as containers or cartridges in laGGer. Typically, a container service can be accessed, controlled, and configured through a command-line interface (CLI). However, the CLI is not ideal for services designed to provide visual data. Consequently, games are not inherently suitable to run as Docker containers without proper support.

To address this, X11 protocol support facilitates the streaming of visual data from a Docker container. X11, the X Window System [30], is a protocol that enables applications to draw visual objects. Essentially, X11 streams an application's output to the user's screen and can operate with distributed machines, where one handles processing and another handles display. X11's secure protocol is often used for remote machine access, employing a client-server model where the X Server resides on the user machine and the X Client on the remote machine, enabling bidirectional data flow [28].

X11 can also be integrated with Docker [77], allowing graphical user interfaces (GUIs) to run inside a container and stream visual output to the user display. Since X11 transports bitmaps (high-level visual data representations), it can stream visual data from any service running in a Docker container without additional adjustments. If a service in a Docker container provides a GUI and Docker is configured with X11, visual data streams to the user display seamlessly with a running X Server.

With X11 support, a game can be launched as a Docker container on a cloud server responsible for heavy processing, with the visual output streamed to the user. This means users do not need powerful hardware for processing, just sufficient resources for display.

To stream graphical data to a web browser, we utilized virtual network computing (VNC). noVNC is an HTML client and JavaScript library based on VNC concepts, enabling connection to a remote machine whose output is streamed to the user browser. This approach requires no additional software installation or setup on the user's machine, only a modern web browser.

noVNC [27] employs websockets to communicate with a VNC server. VNC [50] facilitates screen sharing and manipulation, with configurations for tunneling X11 protocol data to the web.

Integrating these technologies supports streaming a game's visual output, computed remotely, to the user display via a browser. Using a MAS approach, specifically SPADE, we integrated all described technologies into a coherent system, as shown in Figure 4.1.

Running game instances in Docker containers allowed us to build an orchestration platform supporting numerous users and game instances simultaneously. The platform scales up the number of instances on demand, accommodating new users. The system also enables real-time videoconferencing between users in a game instance using the Janus WebRTC server. Thus, we implemented a complete game streaming platform using exclusively open standards, open protocols, and open-source software[1].

4.4 Application in Transport and Mobility Research

In the following section, we explore the specific ways game streaming technology can be leveraged in transport and mobility research. This includes simulation and training, data collection and analysis, testing of automated and connected mobility solutions, gamification in transport, and the development of resilient transport systems and logistics.

Simulation and Training A city's public transport authority could use the game streaming system to create a virtual environment replicating the city's actual road and traffic conditions. Bus drivers could then use this virtual environment for training, learning to navigate the city's

[1] The current state of the system is available under an open source license at `https://github.com/AILab-FOI/laGGer`

streets, handle various traffic situations, and respond to emergencies without real-world risks, significantly improving their preparedness and response times.

Testing of Automated and Connected Mobility Solutions A company developing an autonomous vehicle could use the game streaming system to create a virtual city with various traffic scenarios, weather conditions, and unexpected events. The company could then test how well their autonomous vehicle navigates this virtual city, identifying and addressing any issues in a safe, controlled environment.

Gamification in Transport A city could use the game streaming system to create a game where citizens earn points for using public transport, cycling, or walking instead of private cars. These points could be redeemed for rewards, such as discounts on public transport tickets, encouraging more sustainable transport habits among citizens.

Resilient Transport Systems and Logistics A logistics company could use the game streaming system to simulate various disruptions to their supply chain, such as road closures or port delays. They could then use these simulations to develop and test strategies for managing these disruptions, ensuring they can deliver goods on time despite unexpected challenges.

4.5 Advantages of Game Streaming Systems

Game streaming systems offer several advantages over traditional game engines, particularly for applications in transport and mobility research:

Hardware Accessibility Game streaming systems, like laGGer, require less powerful hardware on the user's end because the heavy processing is done on the server side. This means users can access and interact with complex simulations from any device with a good internet connection, including lower-end computers, tablets, or even smartphones.

Central Deployment and Updates With a game streaming system, updates or changes to the simulation can be made centrally on the server and instantly become available to all users. This is much more efficient than updating software on each individual user's device.

Real-Time Communication Game streaming systems can facilitate real-time communication between users, which can be valuable in transport and mobility research. For example, in a training scenario, instructors could provide real-time feedback to trainees. In a research scenario, researchers could observe and communicate with participants in real time as they interact with the simulation.

Scalability Game streaming systems can typically handle a larger number of simultaneous users than traditional game engines. This is particularly useful in research studies involving large numbers of participants or training programs for large organizations.

Data Collection and Analysis Consider a research project studying driver behavior in response to different traffic signal timings. The game streaming system could simulate various traffic signal scenarios, and researchers could collect data on how drivers respond to each scenario. This data could then be analyzed to optimize traffic signal timings for improved traffic flow and reduced congestion. Because all user interactions occur on the server, game streaming systems facilitate more comprehensive data collection and analysis. Every action taken by the user in the simulation can be recorded and analyzed, providing valuable insights for research or feedback for training.

4.6 Conclusion and Future Work

Building on our work with the laGGer game streaming engine, we see a future where these systems become integral to transport and mobility research. The inherent advantages of game streaming systems, such as accessibility, ease of updates, real-time communication, scalability, and data

collection, have already proven their value. However, we believe that the true potential of these systems is yet to be fully realized.

In our future research, we aim to delve deeper into the application areas outlined in this chapter. We will focus on how game streaming technology can be leveraged in transport and mobility research, with a particular emphasis on gamification in transport and the development of resilient transport systems and logistics.

Our goal is to further explore and demonstrate how these systems can significantly contribute to these research areas, thereby advancing our understanding and application of game streaming technology in transport and mobility.

This chapter introduced the laGGer game streaming engine and explored its potential applications in transport and mobility research. By leveraging open-source technologies and HAI methods, we can create scalable, interactive simulations that provide valuable insights and training opportunities. The integration of game streaming with smart mobility solutions exemplifies the broader impact of HAI orchestration, tying together the themes and developments discussed throughout this report.

In summary, this report has demonstrated the diverse applications and benefits of HAI orchestration across multiple domains, from digital twins in autonomous vehicles to cognitive agents in smart mobility and game streaming in intelligent transport. By connecting these chapters through common methodologies and integrated conclusions, we can appreciate the comprehensive potential of HAI in transforming modern technologies and enhancing user experiences.

5. Student Projects

5.1 Introduction

In this chapter, we explore three student projects that were conducted as part of their master's theses. Each project investigates different applications of artificial intelligence and autonomous systems, contributing valuable insights and advancements in their respective areas. The projects are as follows:

Building a Self-Driving RC Car Ivan Oršolić developed a self-driving RC car, implementing various AI algorithms to achieve autonomous navigation and control. This project focuses on the practical challenges and solutions in creating a scaled-down autonomous vehicle [45].

Autonomous Vehicles as a Multi-Agent System Rene Maruševec examined the application of multi-agent systems in the context of autonomous vehicles. This study investigates how individual autonomous agents can collaborate to improve overall system performance and safety [38].

Application of Artificial Intelligence in Racing Games Luka Mandić explored the use of artificial intelligence techniques in racing games. The project aims to simulate real-world applications of AI in vehicles by creating intelligent, adaptive opponents within a game environment [35].

The following sections will delve into the details of each project, highlighting the methodologies used, challenges faced, and the results obtained.

5.2 Building a Self-Driving RC Car

The project "Building a Self-Driving RC Car" by Ivan Orsolic aims to create a scaled-down model of an autonomous vehicle using a remote-controlled (RC) car. The project encompasses the entire process from hardware selection and assembly to software development and machine learning model training.

The hardware platform includes various components such as the RC car model, electronic speed controllers (ESC), servo motors, batteries, and sensors. The choice of these components is critical for ensuring the vehicle's capability to navigate and perform autonomous tasks. The embedded

system, which serves as the brain of the RC car, includes a microcontroller that processes sensor data and controls the actuators.

Figure 5.1: The assembled car [45]

The software architecture integrates multiple machine learning techniques to enable the car to drive autonomously. This involves data acquisition, labeling, and training a neural network to recognize and respond to various driving scenarios. The project employs a convolutional neural network (CNN) to process images from the car's camera and make driving decisions.

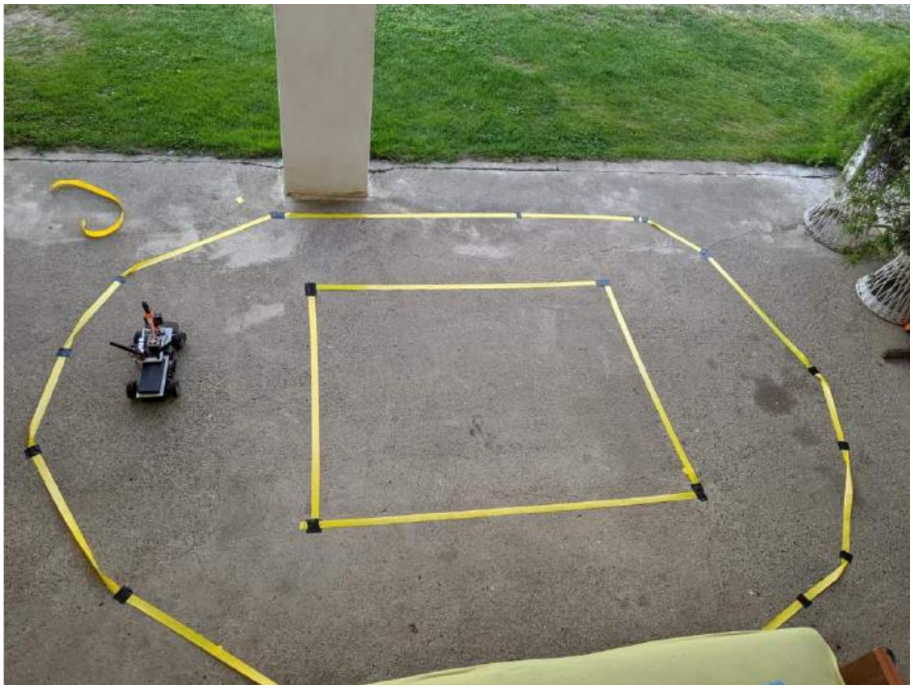


Figure 5.2: The car track [45]

The vehicle's performance is tested on a track designed to simulate real-world driving conditions, including lane changes and obstacle avoidance. This practical implementation demonstrates the feasibility and challenges of developing autonomous driving systems at a smaller scale, providing valuable insights for larger-scale applications.

This project showcases the integration of hardware and software in building an autonomous vehicle as well as highlights the potential of using scaled-down models for educational and research purposes in the field of autonomous driving.

5.3 Autonomous Vehicles as a Multi-Agent System

The project "Autonomous Vehicles as a Multi-Agent System" by Rene Maruševac explores the integration of multi-agent systems (MAS) with machine learning models to enhance the functionality and safety of autonomous vehicles (AVs). This study focuses on developing robust services for lane line detection and traffic sign classification, both critical for autonomous driving.

The lane line detection service employs a basic yet effective algorithm to identify lane markings. This approach involves parameter tuning to optimize performance for different road conditions. While the initial implementation performs well on straight roads, it struggles with curved roads. The project suggests potential improvements, such as employing a geometrical model or a deep learning approach with convolutional neural networks (CNNs) for better accuracy in real-world applications.

Figure 5.3: Lane line as detected by the algorithm [38]

For traffic sign classification, the project implements a deep learning model based on the LeNet-5 architecture, which is foundational in modern computer vision applications. Using the German Traffic Sign Recognition Benchmark (GTSRB) dataset, the model achieves a test accuracy of 94.2%. This demonstrates the effectiveness of CNNs in recognizing traffic signs, which is essential for the safe operation of AVs. The project highlights the importance of parameter tuning and suggests using more advanced architectures like VGGNet for higher accuracy.

The study also provides a comprehensive comparison of various object detection algorithms, including YOLO, SSD, and Faster R-CNN, analyzing their speed and accuracy trade-offs. For instance, YOLO offers real-time performance with acceptable accuracy, making it suitable for applications where speed is critical. In contrast, Faster R-CNN provides higher accuracy but at the cost of increased computational requirements, which may not be feasible for real-time applications on resource-constrained devices .

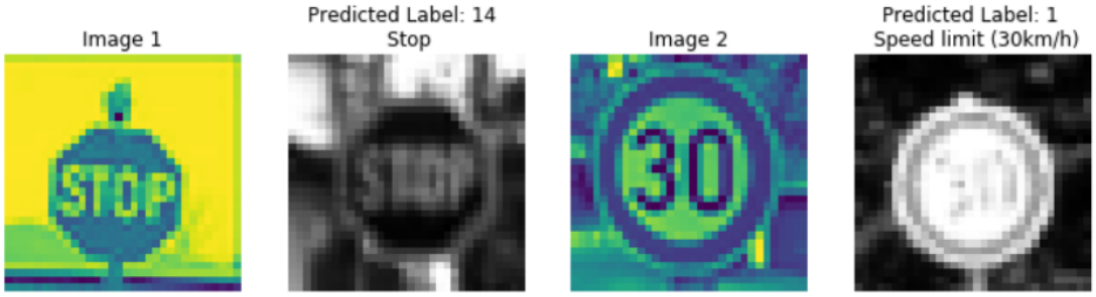


Figure 5.4: Traffic sign classification [38]

In conclusion, the project successfully integrates machine learning models into a MAS framework to enhance AV capabilities. It emphasizes the need for continuous improvement and adaptation of these models to handle the dynamic and unpredictable nature of real-world driving environments. By leveraging advanced machine learning techniques and robust MAS architectures, this study paves the way for more reliable and efficient autonomous driving systems.

5.4 Application of Artificial Intelligence in Racing Games

The third project, "Application of Artificial Intelligence in Racing Games," undertaken by Luka Mandić, explores the integration of AI methodologies within racing game environments. This project leverages advanced machine learning techniques to simulate and improve AI behavior in racing scenarios, providing insights applicable to both virtual and real-world autonomous driving systems.

The project begins with an overview of the evolution of artificial intelligence in racing games, tracing its history from early implementations to contemporary uses. It highlights key milestones such as the introduction of AI in games like "Pole Position" and "Super Mario Kart," where AI-controlled opponents provided dynamic and challenging gameplay experiences. This historical context sets the stage for understanding the advancements and current state of AI in gaming.

A significant part of the project focuses on the technical implementation of neural networks and genetic algorithms within the Unity game engine. The author develops a racing game prototype where AI agents, represented as cars, navigate a racetrack using learned behaviors. The neural networks are trained to handle steering and acceleration, enabling the cars to complete laps autonomously. The training process involves feeding the neural network various inputs related to the car's environment and desired outcomes, iteratively improving its performance.

The project also delves into the use of genetic algorithms to optimize the neural networks. Genetic algorithms mimic the process of natural selection, evolving the neural network's parameters over successive generations to enhance performance. This approach allows the AI to learn and adapt to the racing environment, improving its ability to navigate the track efficiently and effectively.

Several figures from the thesis illustrate key aspects of the implementation:

The practical section of the thesis provides detailed explanations of the code and algorithms used, including the implementation of the Unity environment, the setup of the car model, and the integration of the neural networks and genetic algorithms. The project demonstrates the potential of AI to create more realistic and challenging game environments, while also offering insights into how similar techniques can be applied to real-world autonomous vehicle systems.

By bridging the gap between gaming and real-world applications, this project not only enhances the gaming experience but also contributes to the broader field of AI research in autonomous systems. The methodologies and findings presented in this project underscore the versatility and

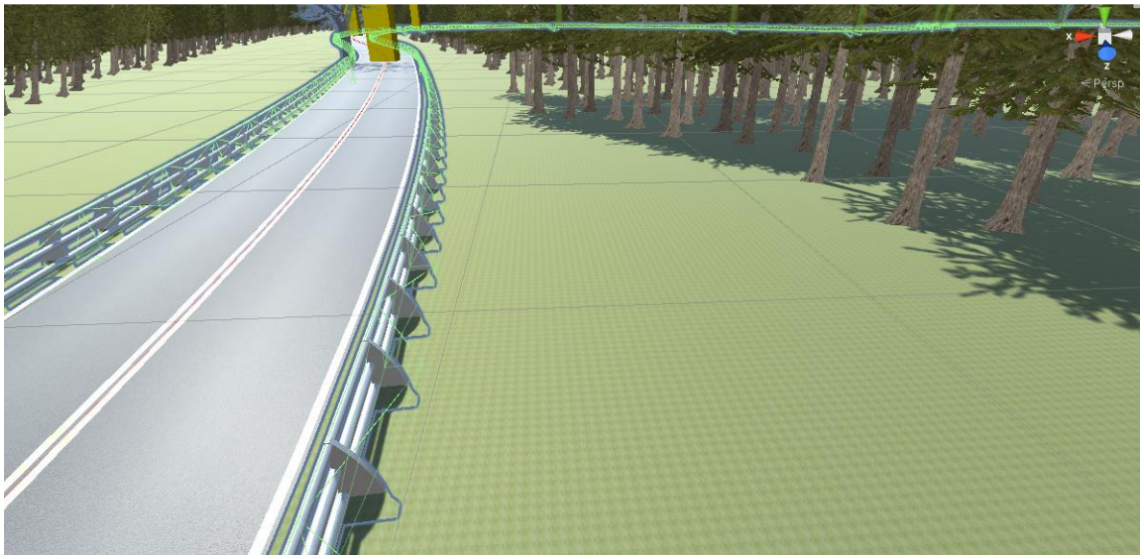


Figure 5.5: 3D road model in Unity [35]

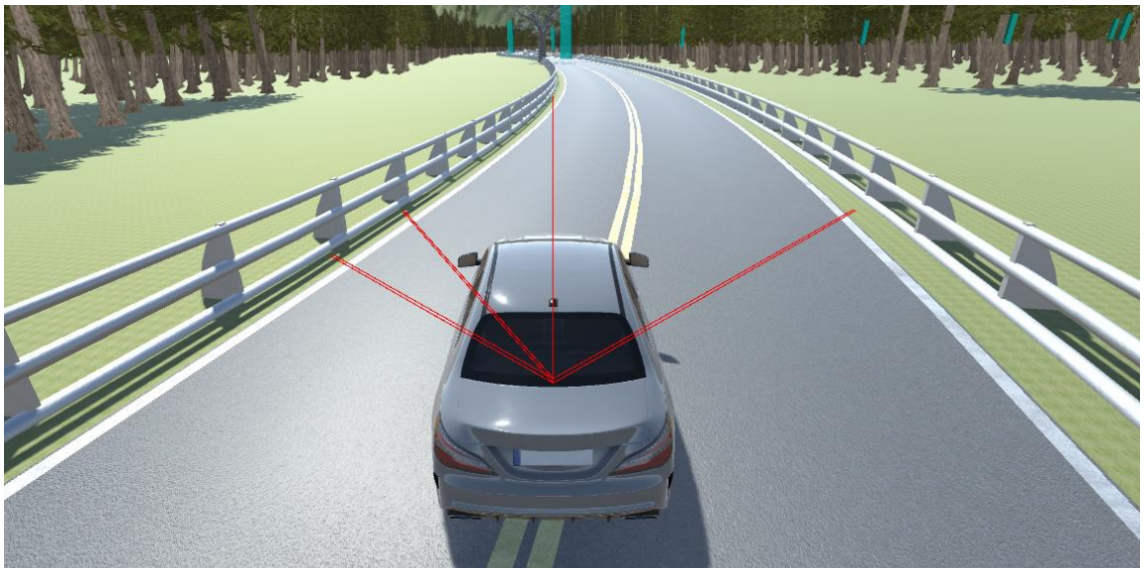


Figure 5.6: Sensors on the Car [35]

potential of AI in various domains, from entertainment to transportation.

Bibliography

[1] Hosny Ahmed Abbas, Samir Ibrahim Shaheen, and Mohammed Hussein Amin. "Organization of Multi-Agent Systems: An Overview". In: *International Journal of Intelligent Information Systems* 4.3 (2015), page 46. ISSN: 2328-7675. DOI: 10.11648/j.ijiis.20150403.11 (cited on page 21).

[2] Sadeq Almeaibed et al. "Digital twin analysis to promote safety and security in autonomous vehicles". In: *IEEE Communications Standards Magazine* 5.1 (2021), pages 40–46 (cited on page 6).

[3] A. Amirante et al. "Janus: A General Purpose WebRTC Gateway". In: *Proceedings of the Conference on Principles, Systems and Applications of IP Telecommunications*. New York, US-NY: Association for Computing Machinery, Oct. 2014, pages 1–8. ISBN: 978-1-4503-2124-2. DOI: 10.1145/2670386.2670389. (Visited on 05/21/2023) (cited on page 21).

[4] Danial Jahed Armaghani et al. "Airblast prediction through a hybrid genetic algorithm-ANN model". In: *Neural Computing and Applications* 29.9 (2018), pages 619–629 (cited on page 2).

[5] Aydin Azizi. "Hybrid artificial intelligence optimization technique". In: *Applications of Artificial Intelligence Techniques in Industry 4.0*. Springer, 2019, pages 27–47 (cited on page 2).

[6] Michael Batty. *Digital twins*. 2018 (cited on page 5).

[7] Amy Baylor. "Intelligent agents as cognitive tools for education". In: *Educational technology* (1999), pages 36–40 (cited on page 13).

[8] Ghanishtha Bhatti, Harshit Mohan, and R Raja Singh. "Towards the future of smart electric vehicles: Digital twin technology". In: *Renewable and Sustainable Energy Reviews* 141 (2021), page 110801 (cited on page 6).

[9] Kien-Trinh Thi Bui et al. “A novel hybrid artificial intelligent approach based on neural fuzzy inference model and particle swarm optimization for horizontal displacement modeling of hydropower dam”. In: *Neural Computing and Applications* 29.12 (2018), pages 1495–1506 (cited on page 2).

[10] Juan C. Burguillo. “Self-Organization”. In: *Self-Organizing Coalitions for Managing Complexity*. Emergence, Complexity and Computation 29. Cham: Springer International Publishing, 2018. Chapter 6, pages 89–100. ISBN: 978-3-319-69898-4. DOI: 10.1007/978-3-319-69898-4_6 (cited on page 21).

[11] Marc Carrascosa and Boris Bellalta. “Cloud-gaming: Analysis of google stadia traffic”. In: *Computer Communications* 188 (2022), pages 99–116 (cited on page 19).

[12] Wei Chen et al. “Novel hybrid artificial intelligence approach of bivariate statistical-methods-based kernel logistic regression classifier for landslide susceptibility modeling”. In: *Bulletin of Engineering Geology and the Environment* (2018), pages 1–23 (cited on page 2).

[13] Armando W Colombo et al. “Industrial cyberphysical systems: A backbone of the fourth industrial revolution”. In: *IEEE Industrial Electronics Magazine* 11.1 (2017), pages 6–16 (cited on page 5).

[14] Michael Cook, Simon Colton, and Azalea Raad. “Inferring Design Constraints From Game Ruleset Analysis”. In: *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE. 2018, pages 1–8 (cited on page 2).

[15] Tariq Daradkeh and Anjali Agarwal. “Modeling and Optimizing Micro-Service Based Cloud Elastic Management System”. In: *Simulation Modelling Practice and Theory* 123 (2023), page 102713. ISSN: 1569190X. DOI: 10.1016/j.simpat.2022.102713. (Visited on 05/21/2023) (cited on page 20).

[16] Dominik Dellermann et al. “Hybrid intelligence”. In: *Business & Information Systems Engineering* (2019), pages 1–7 (cited on page 1).

[17] Marco Dorigo, Guy Theraulaz, and Vito Trianni. “Swarm Robotics: Past, Present, and Future [Point of View]”. In: *Proceedings of the IEEE* 109.7 (July 2021), pages 1152–1165. ISSN: 0018-9219, 1558-2256. DOI: 10.1109/JPROC.2021.3072740. (Visited on 05/21/2023) (cited on page 21).

[18] Ricardo Faria et al. “Smart mobility: A survey”. In: *2017 International conference on internet of things for the global community (IoTGC)*. IEEE. 2017, pages 1–8 (cited on page 15).

[19] F Pereira Ferreira, Giorgenes Gelatti, and S Raupp Musse. “Intelligent virtual environment and camera control in behavioural simulation”. In: *Proceedings. XV Brazilian Symposium on Computer Graphics and Image Processing*. IEEE. 2002, pages 365–372 (cited on page 8).

[20] Fotis Foukalas. “Cognitive IoT platform for fog computing industrial applications”. In: *Computers & Electrical Engineering* 87 (2020), page 106770 (cited on page 13).

[21] Tonglin Fu and Chen Wang. “A hybrid wind speed forecasting method and wind energy resource analysis based on a swarm intelligence optimization algorithm and an artificial intelligence model”. In: *Sustainability* 10.11 (2018), page 3913 (cited on page 2).

[22] Pezhman Ghadimi et al. “Intelligent Sustainable Supplier Selection Using Multi-Agent Technology: Theory and Application for Industry 4.0 Supply Chains”. In: *Computers & Industrial Engineering* 127 (Jan. 2019), pages 588–600. ISSN: 03608352. DOI: 10.1016/j.cie.2018.10.050. (Visited on 05/21/2023) (cited on page 21).

[23] IZIX. *What Is Smart Mobility? [Online]. Accessed: 13 Jun 2022*. Feb. 23, 2021. URL: https://blog.izix.eu/en/what-is-smart-mobility (visited on 07/05/2022) (cited on page 15).

[24] Abolfazl Jaafari et al. “Hybrid artificial intelligence models based on a neuro-fuzzy system and metaheuristic optimization algorithms for spatial prediction of wildfire probability”. In: *Agricultural and Forest Meteorology* 266 (2019), pages 198–207 (cited on page 2).

[25] Yuchen Jiang et al. “Industrial applications of digital twins”. In: *Philosophical Transactions of the Royal Society A* 379.2207 (2021), page 20200360 (cited on page 5).

[26] Peng Jing et al. “Agent-Based Simulation of Autonomous Vehicles: A Systematic Literature Review”. In: *IEEE Access* 8 (2020), pages 79089–79103. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2990295. (Visited on 05/21/2023) (cited on page 21).

[27] Martin Joel et al. *noVNC - the open source VNC client*. [Online; accessed 21-May-2023]. 2023. URL: https://novnc.com (cited on page 22).

[28] Brian Joerger. *What You Need To Know About X11 Forwarding*. [Online; accessed 21-May-2023]. 2022. URL: https://goteleport.com/blog/x11-forwarding/ (cited on page 22).

[29] David Jones et al. “Characterising the Digital Twin: A systematic literature review”. In: *CIRP Journal of Manufacturing Science and Technology* 29 (2020), pages 36–52 (cited on page 5).

[30] Oliver Jones. *Introduction to the X Window System*. Prentice Hall, 1989 (cited on page 22).

[31] Mohammadreza Koopialipoor et al. “Applying various hybrid intelligent systems to evaluate and predict slope stability under static and dynamic conditions”. In: *Soft Computing* (2018), pages 1–17 (cited on page 2).

[32] In Lee. *Encyclopedia of E-business Development and Management in the Global Economy*. IGI Global, 2010 (cited on page 13).

[33] Mengshan Li et al. “Prediction of pKa values for neutral and basic drugs based on hybrid artificial intelligence methods”. In: *Scientific reports* 8.1 (2018), page 3991 (cited on page 2).

[34] Craig A Lindley. “The Gameplay Gestalt, Narrative, and Interactive Storytelling.” In: *CGDC Conf.* Citeseer. 2002 (cited on page 8).

[35] Luka Mandić. “Application of Artificial Intelligence in Racing Games”. Master’s thesis. Master’s thesis. University of Zagreb, Faculty of Organization and Informatics, 2021 (cited on pages 25, 29).

[36] Gunasekaran Manogaran, R Varatharajan, and MK Priyan. “Hybrid recommendation system for heart disease diagnosis based on multiple kernel learning with adaptive neuro-fuzzy inference system”. In: *Multimedia tools and applications* 77.4 (2018), pages 4379–4399 (cited on page 2).

[37] Xavier Marchal et al. “An Analysis of Cloud Gaming Platforms Behaviour Under Synthetic Network Constraints and Real Cellular Networks Conditions”. In: *Journal of Network and Systems Management* 31.2 (2023), pages 1–31 (cited on page 19).

[38] Rene Maruševec. “Autonomous Vehicles as a Multi-Agent System”. Master’s thesis. Master’s thesis. University of Zagreb, Faculty of Organization and Informatics, 2021 (cited on pages 25, 27, 28).

[39] Tsega Y Melesse, Valentina Di Pasquale, and Stefano Riemma. “Digital twin models in industrial operations: A systematic literature review”. In: *Procedia Manufacturing* 42 (2020), pages 267–272 (cited on page 5).

[40] Robin Milner. *Communicating and mobile systems: the pi calculus*. Cambridge university press, 1999 (cited on page 6).

[41] JI Mitola. “Cognitive radio. An integrated agent architecture for software defined radio.” PhD thesis. Kungliga Tekniska Hogskolan (Sweden), 2002 (cited on page 13).

[42] B Okreša Đurić et al. “MAMbO5: a new ontology approach for modelling and managing intelligent virtual environments based on multi-agent systems”. In: *Journal of Ambient Intelligence and Humanized Computing* 10.9 (2019), pages 3629–3641 (cited on page 8).

[43] Bogdan Okreša Đurić. “Organizational Modeling of Large-Scale Multi-Agent Systems with Application to Computer Games”. PhD thesis. Faculty of Organization and Informatics, University of Zagreb, 2018 (cited on page 2).

[44] Bogdan Okreša Đurić. “Organizational Modeling of Large-Scale Multi-Agent Systems with Application to Computer Games”. Doctoral thesis. Varaždin, HR: University of Zagreb, 2019. 236 pages. URL: https://urn.nsk.hr/urn:nbn:hr:211:783555 (cited on page 21).

[45] Ivan Oršolić. “Building a Self-Driving RC Car”. Master’s thesis. Master’s thesis. University of Zagreb, Faculty of Organization and Informatics, 2021 (cited on pages 25, 26).

[46] Javier Palanca et al. “SPADE 3: Supporting the New Generation of Multi-Agent Systems”. In: *IEEE Access* 8 (2020), pages 182537–182549 (cited on page 6).

[47] Ken Peffers et al. “Design Science Research Process: A Model for Producing and Presenting Information Systems Research”. In: (2020). DOI: 10.48550/ARXIV.2006.02763. URL: https://arxiv.org/abs/2006.02763 (cited on page 13).

[48] Hridesh Rajan. “ANTLR: A Brief Review”. In: (2022) (cited on page 6).

[49] Anton Rassõlkin et al. “Digital twin for propulsion drive of autonomous electric vehicle”. In: *2019 IEEE 60th International Scientific Conference on Power and Electrical Engineering of Riga Technical University (RTUCON)*. IEEE. 2019, pages 1–4 (cited on pages 5, 6).

[50] T. Richardson et al. “Virtual Network Computing”. In: *IEEE Internet Computing* 2.1 (1998), pages 33–38. ISSN: 10897801. DOI: 10.1109/4236.656066. (Visited on 05/21/2023) (cited on page 22).

[51] JA Rincon et al. “Developing adaptive agents situated in intelligent virtual environments”. In: *International conference on hybrid artificial intelligence systems*. Springer. 2014, pages 98–109 (cited on page 8).

[52] Yara Rizk, Mariette Awad, and Edward W. Tunstel. “Cooperative Heterogeneous Multi-Robot Systems: A Survey”. In: *ACM Computing Surveys* 52.2 (Mar. 2020), pages 1–31. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3303848. (Visited on 05/21/2023) (cited on page 21).

[53] Sebastian Rodriguez et al. “Holonic multi-agent systems”. In: *Self-organising Software*. Springer, 2011, pages 251–279 (cited on pages 6, 15).

[54] Stuart J. Russell and Peter Norvig, editors. *Artificial Intelligence: A Modern Approach*. 4th edition. Pearson Series in Artificial Intelligence. Harlow, UK: Pearson Education Limited, 2022. ISBN: 978-1-292-40113-3 (cited on page 21).

[55] Markus Schatten, Bogdan Okreša Đurić, and Tomislav Peharda. “A Cognitive Agent for University Student Support”. In: *2021 IEEE Technology & Engineering Management Conference - Europe (TEMSCON-EUR) (TEMSCON-EUR 2021)*. May 2021 (cited on pages 13, 14).

[56] Markus Schatten, Bogdan Okreša Đurić, and Tomislav Peharda. “Towards a Streamed Holographic 3D Game Engine”. In: *Proceedings of the Central European Conference on Information and Intelligent Systems*. Edited by Neven Vrček, Lourdes Guàrdia, and Petra Grd. Varaždin, HR: Faculty of Organization and Informatics, Sept. 2022, pages 17–22. (Visited on 12/11/2021) (cited on pages 20, 21).

[57] Markus Schatten, Bogdan Okreša Đurić, and Igor Tomičić. “Towards Simulation of Ambient Intelligence in Autonomous Vehicles using Car Racing Games”. In: *Central European Conference on Information and Intelligent Systems*. Faculty of Organization and Informatics Varazdin. 2019, pages 3–6 (cited on page 8).

[58] Markus Schatten, Bogdan Okreša Đurić, and Igor Tomičić. “Orchestration Platforms for Hybrid Artificial Intelligence in Computer Games-A Conceptual Model”. In: *2020 Central European Conference on Information and Intelligent Systems*. Varaždin, Croatia: Faculty of Organization and Informatics, 2020, pages 3–8 (cited on pages 6, 7, 15).

[59] Markus Schatten, Tomislav Peharda, and Igor Tomicic. “Towards an Orchestrated Game Development Approach to Digital Twinning in Autonomous Vehicles”. In: *Central European Conference on Information and Intelligent Systems*. Faculty of Organization and Informatics Varazdin. 2022, pages 3–8 (cited on page 2).

[60] Markus Schatten, Jurica Ševa, and Igor Tomičić. “A roadmap for scalable agent organizations in the internet of everything”. In: *Journal of Systems and Software* 115 (2016), pages 31–41 (cited on pages 2, 15).

[61] Markus Schatten, Igor Tomičić, and Bogdan Okreša Đurić. “A review on application domains of large-scale multiagent systems”. In: *Central european conference on information and intelligent systems*. 2017 (cited on pages 2, 15).

[62] Markus Schatten, Igor Tomičić, and Bogdan Okreša Đurić. “Orchestration Platforms for Hybrid Artificial Intelligence in Computer Games – A Conceptual Model”. In: *Central European Conference on Information and Intelligent Systems*. Central European Conference on Information and Intelligent Systems. Edited by Vjeran Strahonja, William Steingartner, and Valentina Kirinić. Varaždin, HR: Faculty of Organization and Informatics, University of Zagreb, 2020, pages 3–8 (cited on page 20).

[63] Markus Schatten, Igor Tomičić, and Bogdan Okreša Đurić. “Towards Application Programming Interfaces for Cloud Services Orchestration Platforms in Computer Games”. In: *2020 Central European Conference on Information and Intelligent Systems*. Varaždin, Croatia: Faculty of Organization and Informatics, 2020, pages 9–14 (cited on pages 8, 15).

[64] Markus Schatten, Igor Tomičić, and Bogdan Okreša Đurić. “Towards Application Programming Interfaces for Cloud Services Orchestration Platforms in Computer Games”. In: *Central European Conference on Information and Intelligent Systems*. Central European Conference on Information and Intelligent Systems. Edited by Vjeran Strahonja, William Steingartner, and Valentina Kirinić. Varaždin, HR: Faculty of Organization and Informatics, University of Zagreb, Oct. 7–9, 2020, pages 9–14 (cited on page 20).

[65] Markus Schatten et al. “Agents as bots–an initial attempt towards model-driven mmorpg gameplay”. In: *International conference on practical applications of agents and multi-agent systems*. Springer. 2017, pages 246–258 (cited on page 2).

[66] Markus Schatten et al. “Automated MMORPG Testing–An Agent-Based Approach”. In: *International conference on practical applications of agents and multi-agent systems*. Springer. 2017, pages 359–363 (cited on page 2).

[67] Markus Schatten et al. “Large-Scale Multi-Agent Modelling of Massively Multi-Player On-Line Role-Playing Games–A Summary”. In: *Central European Conference on Information and Intelligent Systems*. 2017 (cited on page 2).

[68] Markus Schatten et al. “A cognitive agent’s infrastructure for smart mobility”. In: *Transportation Research Procedia* 64 (2022), pages 199–204 (cited on page 2).

[69] Markus Schatten et al. “An Orchestrated Game Streaming System for Transport and Mobility Research”. In: *Transportation Research Procedia* 73 (2023) (cited on page 2).

[70] Danielle R Silva et al. “A synthetic actor model for long-term computer games”. In: *Virtual Reality* 5.2 (2000), pages 107–116 (cited on page 8).

[71] Tajana Šokec. *Modeliranje kontekstualno svjesnog agenta za razgovor na hrvatskom jeziku uz pomoć konačnog automata i strojnog učenja*. Rektorova nagrada Sveučilišta u Zagrebu. Mentor: Markus Schatten. 2019 (cited on page 14).

[72] Shinichiro Suganuma, Daisuke Sakamoto, and Haruhiko Shimoyama. “An embodied conversational agent for unguided internet-based cognitive behavior therapy in preventative mental health: feasibility and acceptability pilot trial”. In: *JMIR mental health* 5.3 (2018), e10454 (cited on page 13).

[73] Mirko Suznjevic, Ivan Slivar, and Lea Skorin-Kapov. “Analysis and QoE evaluation of cloud gaming service adaptation under different network conditions: The case of NVIDIA GeForce NOW”. In: *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE. 2016, pages 1–6 (cited on page 19).

[74] Johannes Thönes. “Microservices”. In: *IEEE Software* 32.1 (2015), pages 116–116. ISSN: 0740-7459, 1937-4194. DOI: 10.1109/MS.2015.11. (Visited on 05/21/2023) (cited on page 20).

[75] Chien Van Dang et al. “Application of soar cognitive agent based on utilitarian ethics theory for home service robots”. In: *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE. 2017, pages 155–158 (cited on page 13).

[76] Peter Viechnicki et al. “Smart mobility – Reducing congestion and fostering faster, greener, and cheaper transportation options”. In: (2015) (cited on page 15).

[77] Martin Viereck. *X11docker*. [Online; accessed 21-May-2023]. 2023. URL: https://github.com/mviereck/x11docker (cited on page 22).

[78] Minggang Wang et al. “A novel hybrid method of forecasting crude oil prices using complex network science and artificial intelligence algorithms”. In: *Applied energy* 220 (2018), pages 480–495 (cited on page 2).

[79] Shao-Hua Wang, Chia-Heng Tu, and Jyh-Ching Juang. “Automatic traffic modelling for creating digital twins to facilitate autonomous vehicle development”. In: *Connection Science* 34.1 (2022), pages 1018–1037 (cited on pages 5, 6).

[80] Michael Wooldridge. *An Introduction to MultiAgent Systems*. 2nd edition. Glasgow, UK: John Wiley & Sons Ltd., 2009. 484 pages. ISBN: 978-0-470-51946-2. URL: http://www.cs.ox.ac.uk/people/michael.wooldridge/pubs/imas/IMAS2e.html (cited on page 21).

[81] Jiachen Yang, Jipeng Zhang, and Huihui Wang. “Urban Traffic Control in Software Defined Internet of Things via a Multi-Agent Deep Reinforcement Learning Approach”. In: *IEEE Transactions on Intelligent Transportation Systems* 22.6 (June 2021), pages 3742–3754. ISSN: 1524-9050, 1558-0016. DOI: 10.1109/TITS.2020.3023788. (Visited on 05/21/2023) (cited on page 21).

[82] Georgios N. Yannakakis and Julian Togelius. *Artificial Intelligence and Games*. Cham: Springer International Publishing, 2018. ISBN: 978-3-319-63519-4. DOI: 10.1007/978-3-319-63519-4 (cited on page 2).

[83] Tao Zhang et al. "A novel platform for simulation and evaluation of intelligent behavior of driverless vehicle". In: *2008 IEEE International Conference on Vehicular Electronics and Safety*. IEEE. 2008, pages 237–240 (cited on page 8).