Umjetni inteligentni agent kao osnovna jedinica višeagentnih sustava i modeliranja temeljenog na agentima

Jocković, Denis

Undergraduate thesis / Završni rad

2020

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

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:572603

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Denis Jocković

UMJETNI INTELIGENTNI AGENT KAO OSNOVNA JEDINICA VIŠEAGENTNIH SUSTAVA I MODELIRANJA TEMELJENOG NA AGENTIMA

ZAVRŠNI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Denis Jocković

Matični broj: 45938/17–R

Studij: Informacijski sustavi

UMJETNI INTELIGENTNI AGENT KAO OSNOVNA JEDINICA VIŠEAGENTNIH SUSTAVA I MODELIRANJA TEMELJENOG NA AGENTIMA

ZAVRŠNI RAD

Mentor :

Dr. sc. Bogdan Okreša Đurić

Varaždin, rujan 2020.

Denis Jocković

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Ovaj rad bavi se inteligentnim agentima. Konkretno, bavi se svojstvima i tipovima inteligentnih agenata te će ukratko biti prikazane neke od mogućih primjena. Nadalje, rad se bavi višeagentnim sustavima koji su usko vezani za agente iz razloga što su to zapravo sustavi povezanih inteligentnih agenata. Rad se bavi i s modeliranjem temeljenom na agentima koje je zapravo vrsta višeagentnog sustava. Navedene su prednosti i nedostatci modeliranja kako bi se vidjelo u kojim situacijama se ono koristi. U radu će se usporediti modeliranje s višeagentnim sustavima i na taj način će se vidjeti koje su njihove sličnosti i razlike te kada se upotrebljava jedno a kada drugo.

Ključne riječi: inteligentni agenti; višeagentni sustavi; modeliranje temeljeno na agentima; svojstva; autonomnost; fleksibilnost; mobilnost; primjena; uspoređivanje;

Sadržaj

1. Uvod 1

2. Metode i tehnike rada 2

3. Definiranje inteligentnog agenta 3

3.1. Karakteristike inteligentnih agenata 3

3.1.1. Autonomnost 4

3.1.1.1. Fleksibilnost 5

3.1.1.2. Adaptabilnost 6

3.1.2. Racionalnost 7

3.1.3. Učenje 7

3.1.4. Mobilnost 8

3.2. Tipovi inteligentnih agenata 8

3.2.1. Jednostavni refleksni agenti 9

3.2.2. Agenti s bilježenjem promjena u okolini 9

3.2.3. Agenti temeljeni na cilju 11

3.2.4. Agenti temeljeni na korisnosti 12

3.3. Primjena inteligentnih agenata 13

4. Višeagentni sustavi 15

4.1. Komunikacija agenata 15

4.2. Svojstva agenata 15

4.2.1. Rukovodstvo 16

4.2.2. Donošenje odluka 17

4.2.3. Heterogenost 17

4.2.4. Kašnjenje 17

4.2.5. Metrike 18

4.2.6. Kretanje agenata 18

4.3. Primjene višeagentnih sustava 19

4.3.1. Internet 19

4.3.2. Robotika 20

4.3.3. Proizvodni pogon 21

4.3.4. Kontrolna procesa 21

4.3.5. Financijske djelatnosti 21

4.4. Modeliranje temeljeno na agentima 22

4.4.1. Prednosti modeliranja temeljenog na agentima 23

4.4.2. Nedostatci modeliranja temeljenog na agentima 24

4.4.3. Primjeri modeliranja temeljenog na agentima 25
4.5. Usporedba višeagentnih sustava i modeliranja temeljenog na agentima 26

5. Križić-kružić protiv računala 28
5.1. Programski kod 28
5.2. Rad s programom 31

6. Zaključak 33

Popis literature 37

Popis slika 38

1. Prilog 1 39

1. Uvod

Današnji svijet prepun je strojeva i delikatnih sustava koje olakšavaju život čovjeka. Od satelita u orbiti koji ispunjavaju razne čovjekove potrebe, preko interneta koji je na mnoge načine obogatio život čovječanstva, pa sve do modernih proizvodnih postrojenja. Sve te tvorevine potrebno je kontrolirati, unaprjeđivati i dorađivati. U početku je posao održavanja padao isključivo na čovjeka, međutim kako se tehnologija razvijala i širila održavanje je postajalo sve teže. Ljudima nije lako u ograničenom vremenu kontrolirati brojne aspekte kojima je kontrola potrebna. Kako bi se poboljšao i olakšao svakodnevni život i rad osmišljeni su inteligentni agenti.

Inteligentni agenti danas se koriste u mnogim poljima ljudskog djelovanja kako bi unaprijedili i ubrzali rad ljudi. Primjerice, koriste se za nadzor rada raznih sustava i za ispravljanje kvarova. Na internetu su inteligentni agenti također široko rasprostranjeni. Koriste se za osiguravanje sigurnosti, personaliziranje sadržaja, razgovor s „online asistentima“ i mnoge druge stvari. U ovome radu navedeni su još neki od načina korištenja inteligentnih agenata.

Inteligentni agenti su usko povezani s umjetnom inteligencijom koja je trenutno vrlo popularna što je još jedan od razloga za njezino proučavanje. Zbog svega navedenog dobra je ideja biti upoznat sa konceptom inteligentnih agenata i srodnim konceptima poput višeagentnih sustava i sustava temeljenih na modeliranju agenata.

2. Metode i tehnike rada

Za provođenje istraživačkih aktivnosti uglavnom je korišten Internet. Osim znanstvenih radova, na internetu su pronađene i knjige i isječci iz časopisa koji su korišteni u izradi. Za pronalazak nekih izvora korišten je google scholar. Google scholar je pretraživač od googlea koji služi za pretraživanje znanstvenih radova.

Za pomoć u citiranju korišten je Mendeley koji služi za pomoć u rukovanju s literaturom. Znanstveni rad je pisan u programima Microsoft Word i Overleaf. Overleaf je online LaTeX editor koji je besplatan. Za pisanje programa korišten je Visual Studio 2019. Projekt korišten za izradu programa je Windows Forms App u .NET Frameworku, a programski jezik korišten je C#. U pisanju programa korištene su i službene stranice Microsofta za C# u Visual Studiju.

3. Definiranje inteligentnog agenta

U ovom poglavlju opisano je što je to inteligentni agent, koja je razlika između običnog agenta i inteligentnog agenta i koja svojstva inteligentni agent posjeduje. Također, navedene su i opisane različite vrste inteligentnih agenata. Završno, opisano je kako se inteligentni agenti mogu koristiti.

Za početak je potrebno definirati agenta. Agent je nešto što može djelovati prema primljenim vanjskim informacijama.[1] Inteligentni agent je agent koji donosi svoje odluke uzimajući u obzir svoje prijašnje iskustvo.[2] Iz navedenih definicija može se zaključiti da je razlika između agenta i inteligentnog agenta velika. Primjer agenta je termostat, termostat prima podatke o temperaturi u sobi i ovisno o temperaturi on će ili isključiti ili uključiti grijanje – za njegovo djelovanje nije bitno kolika je ranije bila temperatura. Ako se agent želi izraditi u nekom programskom jeziku, može se vrlo lako programirati. Inteligentni agenti su teži za izradu te se za njih zna uzimati u obzir i prijašnje iskustvo.

Agent se može definirati i pomoću formule: Agent = Arhitektura + Program.[1], [3], [4] U tom slučaju pod arhitekturom se misli na uređaj koji izvršava agenta, a program predstavlja funkciju koju agent treba izvršiti. Arhitektura koja izvršava agenta uvijek mora imati senzore pomoću kojih može primati informacije iz okoliša. Senzor, na primjer, može biti neka kamera ili mikrofon. Također arhitektura mora imati efektore, a oni omogućuju agentu utjecaj na okoliš. Efektor mora agentu omogućiti djelovanje na okoliš.

3.1. Karakteristike inteligentnih agenata

Svaki inteligentni agent ima određene karakteristike koje bi trebao posjedovati kako bi mogao obavljati posao ispravno. U ovome poglavlju te su karakteristike definirane i opisane pomoću primjera. Valja napomenuti da agent posjeduje navedene karakteristike bez obzira nalazi li se on u višeagentnom sustavu ili djeluje sam.

Jedno od najvažnijih svojstava za inteligentnog agenta je autonomnost. Agent je autonoman ako može sam donositi odluke.[1], [4] Autonomne agente koriste, primjerice, mrežne usluge za razmjenu videozapisa kako bi svojim korisnicima predlagale videozapise koji bi ih mogli interesirati. Takvi agenti se mogu prilagoditi bilo kojem korisniku te im nije potrebna pomoć za predlaganje videozapisa. Kako bi bio autonoman, agent treba biti fleksibilan i adaptabilan.[2]

U nastavku su opisane sve bitne karakteristike koje agenti mogu posjedovati. Uz opis karakteristike navedeni su i primjeri koji pomažu u njihovom boljem shvaćanju. Svojstva koja su opisana u radu su:

- Autonomnost
  - Fleksibilnost
  - Adaptabilnost
- Racionalnost

- Učenje
- Mobilnost

3.1.1. Autonomnost

Autonomnost je jedno od svojstava po kojima se inteligentni agenti razlikuju od običnih programa. Autonomni agenti su sposobni raditi prema ostvarivanju cilja bez uplitanja korisnika. Sve što korisnik mora napraviti jest odrediti koji je agentov krajnji cilj.[5] Agenti su autonomni onoliko koliko je njihovo ponašanje određeno njihovom iskustvu.[1]

Kao što je rečeno ranije, da bi agent bio autonoman on mora biti fleksibilan odnosno mora moći sam kontrolirati svoje postupke. Također agent mora biti adaptabilan odnosno mora se moći prilagoditi bilo kakvoj promjeni stanja okoliša[1] i mora imati pristup informacijama koje mu omogućavaju djelovanje.[5] Dakako, autonomni agenti bi trebali biti i sposobni za učenje i imati dovoljno znanja da obave potreban posao.[1], [6]

Prije nego što autonomni inteligentni agenti krenu obavljati neku djelatnost prvo će pomoću svojih senzora prikupiti informacije o okolišu. Nakon toga će obaviti neki posao uzimajući u obzir trenutno stanje okoliša i koristeći svoje znanje.[2] Naravno, agenti su sposobni sve to obaviti samostalno, bez ikakve pomoći korisnika.

Za primjer se može uzeti samovozeći automobil koji nema potrebu za vozačem. Takav stroj je i fleksibilan i adaptabilan. Samovozeći automobili uvijek koriste razne senzore kako bi prikupljali informacije o okolišu, prema toga oni stalno imaju potrebne informacije za djelovanje. Iz tog se može zaključiti da su sustavi inteligentnih agenata koje koriste takvi samovozeći auti autonomni. Takvi automobili su se već i testirali na javnim autocestama te se moglo vidjeti da u autima nema nikoga, što je dokaz da su oni ispunjavali svoj zadatak bez uplitanja korisnika.

Valja napomenuti da se samovozeće automobile koji imaju vozača ne smatra autonomnima nego automatskim. Naime, kako bi automobil bio autonoman, on mora biti sposoban voziti se na cesti sam bez ikakve potrebe za vozačem. Automatski automobili mogu imaju određenu kontrolu nad svojim postupcima međutim za njihovu vožnju je potreban vozač.[7]

Rad s autonomnim inteligentnim agentima je donio i razne pravne i etičke probleme. S pravnog gledišta, tko je odgovoran ako autonomni agent napravi grešku, vlasnik ili kreator agenta? A s etičkog gledišta, koliko slobode odlučivanja se može dati autonomnim agentima u određenim situacijama i što da agenti naprave ako moraju izabrati između dvije loše akcije?[5]

Konkretni pravni i etički problemi se mogu vidjeti na primjeru samovozećih automobila. Ako se dogodi da prilikom takav automobil prilikom vožnje prekrši neki zakon, je li kriv kreator automobila ili je kriv vlasnik automobila. Pritom se mora uzeti u obzira da vlasnik nije morao biti ni u automobilu, ali je ipak njegov vlasnik. Isto tako, kreator automobila više nije njegov vlasnik, ali je on programirao sustav agenata čijim je akcijama automobil prekršio zakon. Drugi slučaj je da iznenada pred automobilom naiđe netko i u toj situaciji automobil ima izbor: ili će pogoditi osobu koja je iskočila pred njega ili će se slupati. Što bi u takvoj situaciji automobil trebao napraviti? Upravo iz tih razloga je danas korištenje samovozećih automobila vrlo ograničeno i

dozvoljena im je samo vožnja po autocestama.

Istraživanje je pokazalo da je 2016. godine 90% sudara na cesti uzrokovano ljudskom greškom.[8], [9] Samim time jedan od razloga za razvoj samovozećih automobila jest bilo smanjivanje nesreća. Istraživanja su zaista pokazala da vozila sposobna za takvu vožnju doživljavaju manje nesreća kada je vožnja automatizirana. Međutim, istraživanje je također pokazalo da je postotak teških nesreća (smrtni slučajevi i teške ozlijede) veći kada je uključena automatizirana vožnja.[9] Iz toga se može zaključiti da je učestalost sudara manja, međutim sudari su smrtonosniji kada je uključena automatizirana vožnja. U istraživanju je istaknut slučaj kada je samovozeći automobil ubio pješaka jer ga je prekasno registrirao, a „emergency breaking“ (eng. „panično“ kočenje) nije bilo omogućeno.[9] To je jedan od slučajeva kada je samovozeći automobil zaista bio kriv za nesreću. Pošto je broj smrtnih slučajeva veći kada su u pitanju samovozeći automobili može se zaključiti kako tehnologija i dalje nije spremna za raširenu upotrebu te da ju je i dalje potrebno usavršavati.

3.1.1.1. Fleksibilnost

Fleksibilnost je karakteristika agenata koji mogu procijeniti kako će određene akcije utjecati na okoliš. Zato agenti koji nisu fleksibilni ne mogu praviti odluke već samo pratiti već zacrtano ponašanje.[1], [2] Većina inteligentnih agenata iz video igri su fleksibilni zato što se prilagođavaju potezima igrača i djeluju prema njima.

U obzir se može uzeti partija šaha protiv računala, prilikom takve igre dobar program će provjeriti koje poteze je igrač napravio i koje će poteze vjerojatno napraviti te će prema tome odigrati svoj potez. Kada agent ne bi bio fleksibilan, program ne bi mogao koristiti poteze igrača da promijeni svoju taktiku i tada igra protiv računala uopće ne bi bila zanimljiva.

Fleksibilni inteligentni agenti mogu biti reaktivni, proaktivni i društveno fleksibilni.[5] Te tri vrste fleksibilnosti se dijele po kompleksnosti implementacije i mogućnostima primjene. Najbolji inteligentni agenti bi trebali biti fleksibilni na sva tri načina jer svaki od njih ima svoju svrhu i korist.

Reaktivni inteligentni agenti su povezani s okolišem te mogu odrađivati akcije nakon što se u okolišu dogodi neka promjena.[5] To znači da se reaktivni agent „ne brinu“ što će se koji će biti utjecaj te akcije na okoliš.(Russell and Norvig 1995) Reaktivnost je, od spomenute tri vrste fleksibilnosti, najlakša za implementaciju. Naime, kako bi inteligentni agent bio reaktivan on ne mora moći komunicirati s korisnicima niti mora moći predvidjeti što će se u budućnosti dogoditi. Prilikom implementacije reaktivnosti programer se samo mora pobrinuti da agent može odgovoriti na registrirane izmjene u okolišu.

Reaktivni inteligentni agent bi se mogao pronaći u nekom poduzeću s proizvodnim pogonom. U tom slučaju agent bi pratio rad pogona te kada bi podaci od nekog senzora pokazali pad performansi agent bi mogao pokušati popraviti kvar. Takav agent je reaktivan zato što bi mogao reagirati na izmjenu okoliša (u ovom slučaju promjenu podataka sa senzora), ali nije proaktivan jer ne može upozoriti na kvar prije nego što se kvar dogodi i nije društveno fleksibilan zato što u njegov rad nije uključena komunikacija s korisnicima.

U većini situacija reaktivnost nije dovoljna kako bi agent mogao ispunjavati sve svoje obaveze. U takvim situacijama potrebno je dodati i proaktivnost prilikom implementacije agenta. Proaktivnost je vrsta fleksibilnosti u kojoj agent pokušava postići neki cilj. To znači da agent neće samo reagirati na određene promjene u okolišu već će samoinicijativno raditi promjene kako bi postigao zadani cilj.[5]

Proaktivni agenti su teži za implementiranje jer osim reagiranja na promjene u okolišu oni moraju znati kako djelovati na okoliš da postignu ono što žele postići, odnosno moraju znati kako će njihove akcije djelovati na okoliš. Osim što su teži za implementiranje proaktivni agenti su i cjenjeniji. Naime, takvi agenti su dobri za smanjenje opterećenja korisnika jer mogu odraditi poslove koje bi inače korisnici morali odrađivati sami.

Za primjer proaktivnog agenta također se može uzeti proizvodno poduzeće. Proaktivni agent koji se brine za pogon trebao bi moći pozitivno utjecati na pogon svojim djelovanjem. Konkretno, proaktivni agent bi u nekim slučajevima mogao vidjeti na koji način može utjecati na pogon da se učinkovitost proizvodnje poveća. Uspoređujući ga s agentom koji je samo reaktivan, može se reći da bi se proaktivni agent u većini slučajeva bolje brinuo za pogon zato što bi se mogao brinuti za pogon cijelo vrijeme, a ne samo reagirati u slučaju neke promjene.

Treći aspekt fleksibilnosti je društvenost. Društvenost je zapravo sposobnost inteligentnog agenta za komuniciranje s drugim agentima koji su često ljudi. Društvenost je potrebna zato što vrlo često u stvarnom svijetu nije moguće riješiti neki problem bez komuniciranja s drugima.[5]

Agenti najčešće koriste neki jezik za komuniciranje s drugim agentima. Korišteni jezici imaju svoje protokole koji se koriste širom svijeta. Društvenost ima svoje teškoće u implementiranju. Prilikom međusobne komunikacije agenti često koriste autentikaciju kako bi drugi agenti znali s kim točno komuniciraju. Agentima je potrebno omogućiti što lakši pronalazak drugih agenata koji su im u trenutku potrebni te je potrebno implementirati agenta na način da može prepoznati je li drugi agent dovoljno pouzdan za suradnju.[5]

Inteligentni agenti koji su sposobni za komunikaciju mogu se pronaći u višeagentnim sustavima. Dapače, agenti koji su dio višeagentnih sustava moraju biti sposobni za komunikaciju jer inače ne bi mogli surađivati s drugim agentima. Više o tome u dijelu s višeagentnim sustavima.

3.1.1.2. Adaptabilnost

Adaptabilnost je svojstvo agenta da se prilagodi okolišu koji se mijenja.[4] Kako bi se mogli prilagoditi okolišu agenti moraju evoluirati, odnosno moraju razvijati svoje mogućnosti komunikacije, načine rješavanja problema, svoje znanje i sposobnosti itd. Adaptivni inteligentni agenti moraju biti sposobni u potpunosti promijeniti svoje ponašanje ako se okoliš dovoljno promijeni. Krajnji cilj agenata je ispunjavanje svojeg zadatka te oni moraju mijenjati svoje ponašanje kako bi taj zadatak postigli.[10], [11] Adaptabilan agent može raditi u puno različitih okoliša, autonomni agenti bi trebali biti adaptabilni.[1]

Razlika između fleksibilnosti i adaptabilnosti je činjenica da agenti koji su fleksibilni ali

nisu adaptabilni ne moraju u obzir uzimati mijenjanje okoliša. Za primjer se mogu uzeti samovozeći automobili. Takvim automobilima upravlja sustav inteligentnih agenata koji je i fleksibilan i adaptabilan. Fleksibilnost sustava omogućuje dolazak iz točke polaska u točku odredišta te se brine da, primjerice, automobil ima dovoljno goriva za cijeli put i da auto poštuje prometna pravila. Adaptabilnost sustava se brine da automobil zna što treba napraviti u slučaju neke izvanredne situacije poput pljuska ili mećave ili nečeg sličnog. Dakle u slučaju fleksibilnosti okoliš je nepromijenjen odnosno on je u svom očekivanom obliku, a adaptivnost se bavi situacijama u kojima se okoliš izmijenio na nepredviđeni način.

3.1.2. Racionalnost

Racionalni agent je agent koji svojim djelovanjem pokušava postići zadani cilj što efikasnije može.[1] To znači da takav agent u većini slučajeva ne bi trebao izvršavati akcije koje ne pridonose njegovom cilju.[5] U slučaju da je cilj mjerljiv, racionalni agent mora djelovati tako da maksimizira vrijednost cilja.[6]

Racionalni agent treba prikupiti sve moguće podatke iz okoliša prije djelovanja kako bi znao kakvo je djelovanje najefikasnije. Nakon djelovanja, agent ima načine provjeravanja uspješnosti izvršenog zadatka. Uspješnost se provjerava zato što se može dogoditi da zadatak nije izvršen na optimalan način. U tom slučaju agent može naučiti iz tog neuspjeha ako je na taj način implementiran.[6]

Dobri racionalni agenti bi trebali moći koristiti prijašnje iskustvo i znanje prilikom svojeg djelovanja.[5] U pravilu će agenti s više znanja moći efikasnije djelovati. Samim time, što se agent više koristi to će biti bolji.

To što je agent racionalan ne znači da će uvijek izabrati točnu akciju. Agent može djelovati samo na osnovu onoga što zna, a zna ono onoliko koliko mu omogućuju njegovi senzori. Ako jedan agent odabere bolju akciju u određenoj situaciji od drugoga, to ne znači nužno da je racionalniji, već je možda dobio više podataka iz okoliša.[1]

Racionalne agente se može pronaći nekim igrama koje se igraju protiv računala poput igrice križić-kružić. Računalo će uvijek pokušati pobijediti u igrici na što brži i efikasniji način. Također ako se implementira na dovoljno dobar način, program može učiti iz svake odigrane partije te nakon vrlo velikog broja igri može biti izrazito težak protivnik. Igrica se može implementirati na način da prvo odigra veliki broj partija sama sa sobom te nakon toga krene igrati sa korisnikom. Na taj način se može vidjeti da je korisniku teže pobijediti ako je računalo odigralo više partija samo sa sobom.

3.1.3. Učenje

Učenje inteligentnih agenata je poboljšavanje njihovih performansi tijekom vremena.[5] Ta karakteristika je vrlo bitna za inteligentne agente te je i danas predmet proučavanja. Agenti koji su sposobni učiti su u dosta slučajeva napredniji od drugih.

Takvi agenti obično prave manje grešaka jer nakon što jednom naprave grešku, istu

više ne bi trebali ponoviti. Osim toga, rješenja mogu pronaći brže i bolje. U slučaju da agenti rade s korisnicima mogu im se bolje prilagoditi. U tome im pomaže upravo sposobnost učenja koja omogućuje bolje buduće djelovanje.[5] Učenje može pomoći agentu u shvaćanju da su neke njegove akcije krive. Osim toga učenje može olakšati adaptaciju agenta na promjenu u okolišu[1]

Umjetne neuronske mreže omogućuju učenje pomoću analiziranja primjera.[12] Tesla, američki proizvođač automobila, koristi neuronske mreže kako bi poboljšao autopilota kojeg koriste samovozeći automobili. Konkretno, Tesla prikuplja podatke o situacijama u kojima se nalaze njegovi automobili u stvarnom svijetu i koristi te podatke kako bi njegov „autopilot“ dobio bolju percepciju stvarnosti i kako bi mogao bolje funkcionirati u vožnji.[13]

Nije lako implementirati agenta s mogućnosti učenja, međutim agenti znaju biti cijenjeni. Sposobnost učenja može agentu dati mogućnost rješavanja većeg broja problema te njihovo rješavanje može biti brže i bolje. Iz tih je razloga učenje veliko područje proučavanja umjetne inteligencije.[5]

3.1.4. Mobilnost

Mobilni agenti su oni koji se mogu kretati kroz mrežu i preseliti iz jednog računala u drugo.[5] Kako bi se agenti mogli seliti računala moraju imati potrebnu arhitekturu. Pod tim se podrazumijevaju serveri za agente, a to su zapravo programi koji omogućuju rad agenata na određenom računalu.[4], [14]

Valja napomenuti da mobilnost ima svoje probleme. Kada agent „otputuje“ od jednog računala u drugo, novo računalo ne zna treba li dopustiti agentu djelovanje što može stvoriti probleme. Razlog tomu je što računalo ne zna je li novi agent virus ili će se njegovo djelovanje odraziti loše na računalu. Zato je prilikom izrade mobilnog agenta bitno implementirati ga na način da se on i njegovo djelovanje može lako identificirati.[4]

Za primjer se može uzeti neki agent koji mora dobaviti određene podatke iz udaljene baze podataka. Postoje dva načina na koja se podaci mogu nabaviti. Prvi je da agent udaljeno zatraži od baze podataka tražene podatke, ali takav pristup zahtjeva konstantno održavanje konekcije što može usporiti ostatak mrežnog prometa. Drugi način je da agent ode do računala na kojem se nalazi baza podataka, pregleda bazu za traženim podacima i vrati se nazad s njima.[5] Drugi pristup manje zauzima manje prometa na mreži te rasterećuje računalo s kojeg su podaci zatraženi te je zato dobar za korištenje. Naravno, kako bi se drugi pristup mogao iskoristiti potrebno je imati mobilnog agenta. Zato je takve agente dobro imati u slučaju da je potrebno puno raditi s drugim udaljenim računalima.

3.2. Tipovi inteligentnih agenata

U prošlom poglavlju su navedena svojstva koja inteligentni agenti mogu posjedovati. Agenti većinom ne posjeduju sva svojstva već samo ona koja su im potrebna. Svojstva agenata određuju njihov način djelovanja i učinkovitost. Što manje svojstava agent posjeduje to će

biti lakši za implementirati, ali će biti manje učinkovit. Prema tome, prilikom izrade agenata programeri se moraju odlučiti kakav im je agent potreban za posao koji treba obaviti.

Razlike između pojedinih tipova će se vidjeti i na primjerima. Za primjere će se koristiti inteligentni agenti koji se koriste za pomoć u poljoprivrednoj proizvodnji. Na taj način bit će prikazano kako različiti tipovi agenata mogu djelovati u istoj situaciji.

Postoji puno podjela agenata na različite tipove, jednu od poznatijih su definirali S. Russel i P.Norvig. Prema njima, inteligentni agenti se mogu podijeliti na četiri tipa:

- Jednostavni refleksni agenti
- Agenti s bilježenjem promjena u okolini
- Agenti temeljeni na cilju
- Agenti temeljeni na korisnosti[1]

3.2.1. Jednostavni refleksni agenti

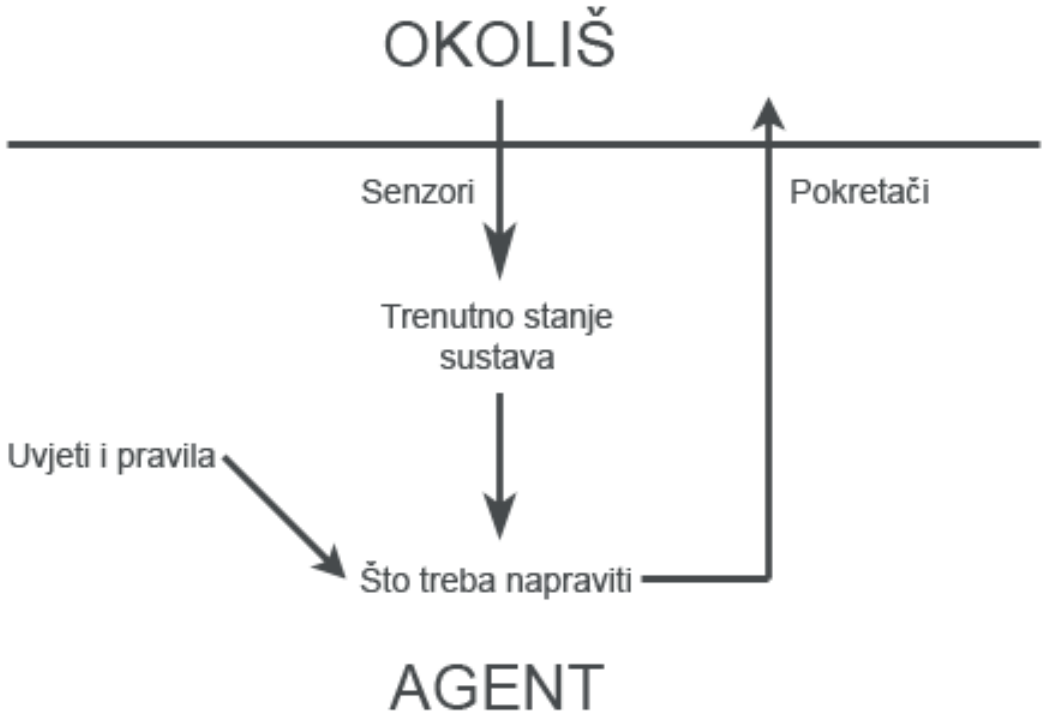
Najjednostavniji tip agenta su jednostavni refleksni agenti. Oni primaju podatke iz svog okoliša te na temelju primljenih podataka odluče što trebaju napraviti. Ne uzimaju u obzir ni prošla niti buduća stanja okoliša što ih čini najlakšima za implementaciju. Jednostavni refleksni agenti nemaju zadani cilj što znači da uopće nisu proaktivni, osim toga nisu niti previše adaptabilni jer neće znati što trebaju napraviti ako se okoliš previše promijeni.

Na slici dolje može se vidjeti da agent pomoću svojih senzora prima podatke o okolišu. Na osnovu podataka zaključuje kakav je okoliš te uz pomoć pravila odlučuje kakvu akciju treba poduzeti. Nakon toga djeluje na okoliš. Kako bi ovaj agent bio upotrebljiv okoliš mora biti u potpunosti poznat što predstavlja veliki problem jer su okoliši u većini slučaja barem djelomično nepoznati.[1]

Jednostavni refleksni agent bi u poljoprivredi mogao služiti za analizu zemlje i predlaganje boljeg načina obrade za dobivanje boljih uroda. Čak i tada, ne bi bio vrlo koristan jer ne bi znao koji je usjev posađen. Pošto različiti usjevi zahtijevaju različite uvjete zemlje kako bi rasli, jednostavni refleksni agent bi se samo mogao brinuti da su ispunjeni osnovni uvjeti za rast. Postoji još jedan problem. U slučaju da agent pronađe štetočine u usjevu, njegova akcija će vjerojatno biti uklanjanje štetočina pomoću nekog pesticida. Međutim, pesticidi ne djeluju odmah, a jednostavni refleksni agenti nisu svjesni prošlih stanja već su samo svjesni sadašnjeg. Postoji šansa da bi prilikom idućeg dobavljanja informacija o okolišu agent ponovo primijetio štetočine te bi ponovo naredio špricanje usjeva s pesticidom, ali pretjerano špricanje je također štetno te bi time agent naškodio usjevu.

3.2.2. Agenti s bilježenjem promjena u okolini

Agenti koji bilježe promjene su vrlo slični agentima spomenutim u prošlom poglavlju. Razlika je u tome da ovaj tip agenta bilježi prošlo stanje okoline što znači da prilikom odlučivanja

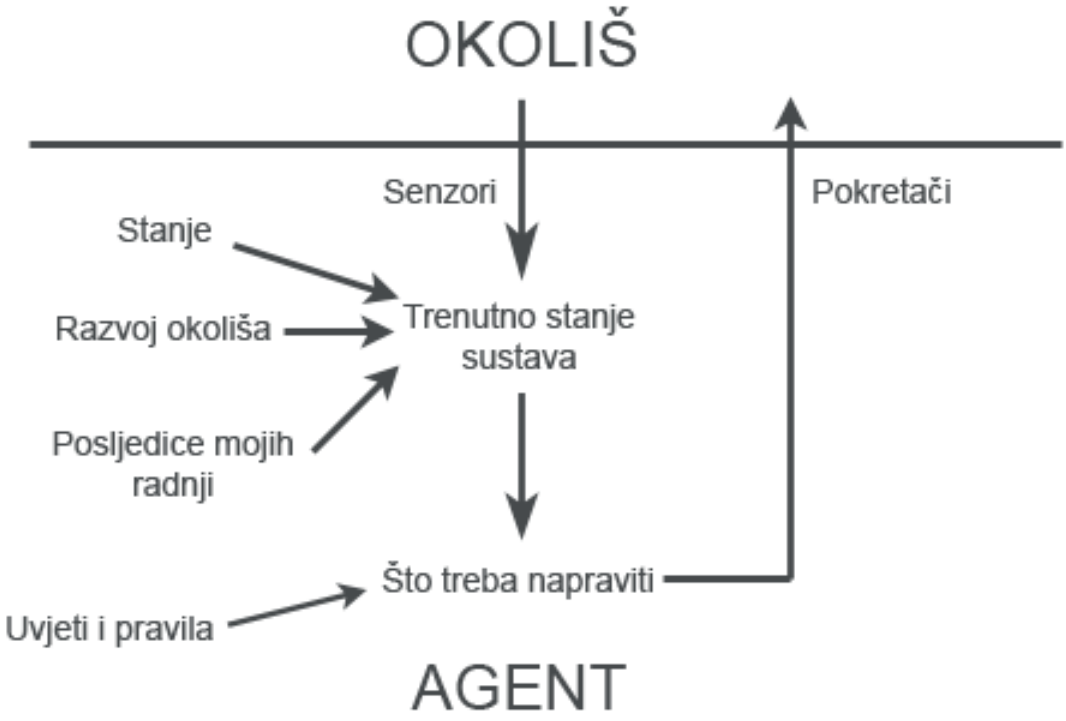


Slika 1: Slikoviti prikaz jednostavnih refleksnih agenata prema [1]

o akciji koju će poduzeti ima potpunije informacije. Takav tip agenta ne mora imati u potpunosti poznati okoliš kako bi bio funkcionalan. Iako napredniji, agenti s bilježenjem promjena u okolini su i dalje vrlo jednostavni te nisu uopće proaktivni jer ni oni nemaju ciljeve. Adaptabilniji su od jednostavnih refleksnih agenata pošto imaju barem neko znanje o prošlim stanjima okoliša.

Na slici se može vidjeti kako djeluje agenti. Pomoću senzora primaju podatke o okolišu. Kako bi odlučili što će napraviti sljedeće oni gledaju u kakvom bi stanju okoliš bio bez njih, koje je bilo prethodno stanje okoliša, kako će njegove akcije utjecati na okoliš i koje je trenutno stanje okoliša. Agent napravi neki posao tek nakon što je razmotrio sve te podatke.[1]

Iz navedenog se može zaključiti da su jednostavni refleksni agenti puno brži jer moraju obraditi manje podataka, međutim agenti koji bilježe promjene u okolini su bolji u većini slučajeva jer su prilagodljiviji na promjene u okolini. Agenti koji bilježe promjene okoliša bi bili bolji izbor za korištenje u poljoprivredi upravo iz tog razloga. U prošlom poglavlju su navedeni problemi koje bi agent imao zato što zna samo za sadašnje stanje. Agenti koji bilježe promjene okoliša ne bi imali te probleme jer su oni svjesni onoga što se dogodilo prije. Konkretno, bili su svjesni da su već špricali usjev s pesticidima pa to ne bi učinili ponovno. Međutim, ostali problemi su i dalje prisutni. Isto kao i jednostavni refleksni agenti, agenti koji bilježe promjene bi se mogli koristiti samo da se kontroliraju osnovne potrebe usjeva jer oni također nemaju cilj, odnosno ne mogu odrediti za koji usjev pripremiti zemlju.



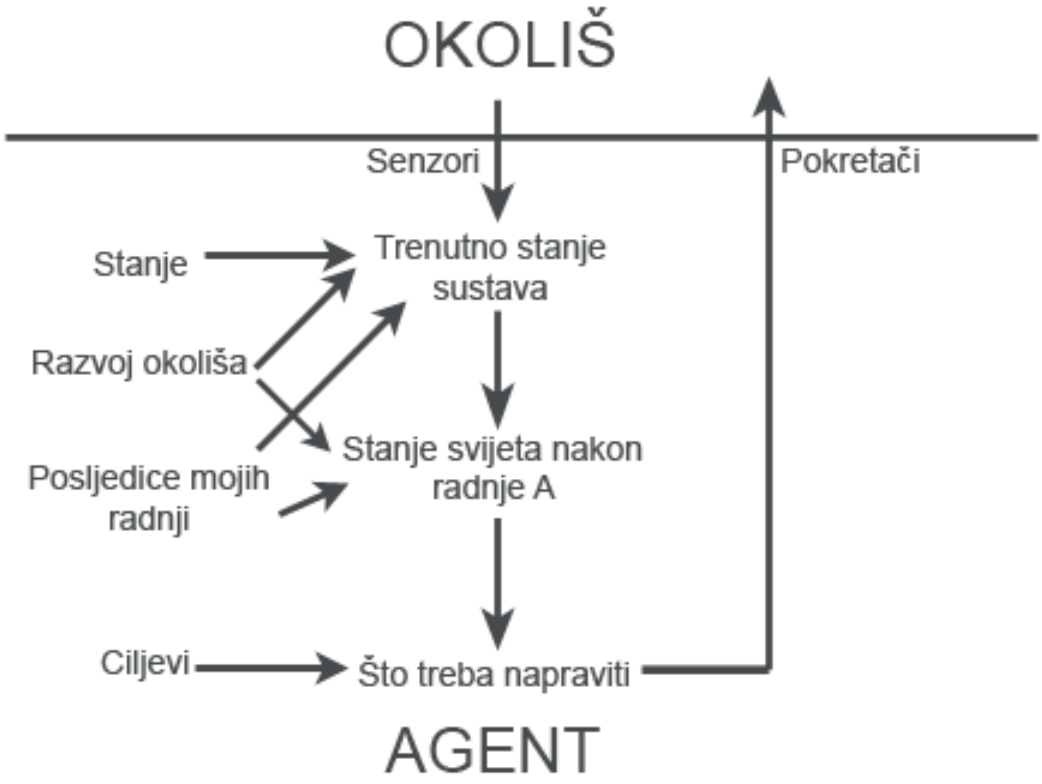
Slika 2: Slikoviti prikaz agenata s bilježenjem promjena u okolini prema [1]

3.2.3. Agenti temeljeni na cilju

Uspoređujući agente temeljene na cilju s već spomenutim tipovima agenata može se primijeniti velika razlika. Prethodno spomenuti tipovi agenata su mogli samo reagirati na neku promjenu te nisu bili sposobni za bilo kakvo odlučivanje o budućnosti, odnosno nisu uopće bili proaktivni. Agenti temeljeni na cilju djeluju na način da pokušavaju ostvariti zadani cilj. To im omogućuje puno veću fleksibilnost prilikom rada jer imaju sposobnost odlučiti koju akciju mogu poduzeti da se dovedu bliže ostvarenja cilja. Samim time, takvi agenti jesu proaktivni.[1]

Kao što se može vidjeti na slici ispod, agenti temeljeni na cilju uzimaju puno stvari u obzir prilikom odabira akcije. Uzimaju podatke o okolišu te na temelju podataka, prošlog stanja okoliša i načina na koji okoliš funkcionira bez njega stvaraju informaciju o tome kakvo je trenutno stanje okoliša. Uz pomoć trenutnog stanja okoliša, znanja o tome kako okoliš funkcionira bez agenta i znanja o tome kako njegovo djelovanje utječe na okoliš agent stvara pretpostavku o tome kako će okoliš izgledati ako poduzme određenu akciju. Ako će poduzimanjem te akcije agent doći bliže ostvarenju svog cilja onda ju i poduzima.

Za razliku od prethodnih tipova, agenti temeljeni na cilju bi bili vrlo korisni za poljoprivrednu proizvodnju. Agentu se može zadati željeni usjev te se agent može u potpunosti fokusirati na ispunjavanje uvjeta potrebnih za uspjeh tog usjeva. Osim toga, prijašnji tipovi su samo mogli reagirati na neke probleme kod razvoja usjeva poput pojave štetočina, međutim agenti temeljeni na cilju mogu djelovati proaktivno i osigurati bolju podlogu za rast usjeva prije nego što je usjev uopće posađen te se mogu prilagoditi promjenama poput nepovoljnih vremenskih



Slika 3: Slikoviti prikaz agenata temeljenih na cilju prema [1]

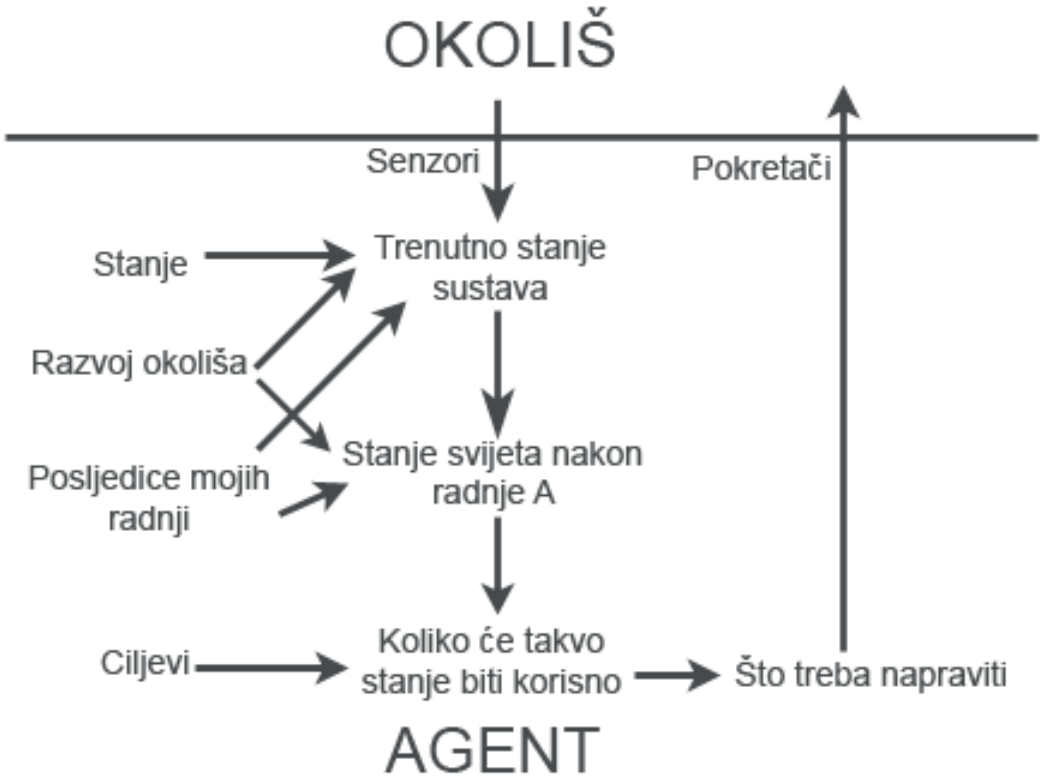
uvjeta ili štetočinama. Samim time, agent temeljen na cilju može kontrolirati čitav poljoprivredni proces od pripreme tla pa sve do žetve.

3.2.4. Agenti temeljeni na korisnosti

Agenti temeljeni na korisnosti slični su agentima iz prošlog poglavlja, ali ovi agenti donose odluke po tome koliko će određena akcija biti korisna prilikom za određeni cilj. Primjerice, ako agent može birati između dvije akcije koje će ga dovesti do cilja, izabrat će onu koja je po nekim kriterijima bolja. Kriteriji koje će agent koristiti mogu biti razni te su određeni prilikom implementacije. Dakle, agenti temeljeni na korisnosti mogu kvantificirati potencijalno buduće stanje. Agenti će napraviti akcije koje vode prema onom stanju koje je „ocjenjeno“ kao najbolje.[1] Dakle, agenti iz prošlog poglavlja ne bi mogli vidjeti razliku između različitih akcija koje vode ka istom cilju, makar jedna akcija bila bolja od druge, dok agenti temeljeni na korisnisti mogu.

Kao i svi ostali, agenti temeljeni na korisnosti uzimaju podatke o okolišu pomoću senzora. Nakon toga stvaraju pretpostavku o tome kakav će okoliš biti ako poduzmu određenu akciju na isti način kao i agenti temeljeni na cilju. Agent određuje koje mu buduće stanje više odgovara pomoću određenih karakteristika koje su mu bitne. Kada odredi koje je to stanje, agent poduzima akciju potrebnu kako bi se ono ispunilo.

Agenti temeljeni na korisnosti bi mogli u potpunosti ispunjavati zadatak pomaganja prilikom poljoprivredne proizvodnje. Ne samo da bi mogli kontrolirati čitav proces kao i agenti



Slika 4: Slikoviti prikaz agenata temeljenih na korisnosti prema [1]

temeljeni na cilju, već bi ga mogli kontrolirati na najoptimalniji mogući način. Također, mogli bi pomagati korisnicima u odlučivanju. Primjerice, možda bi poljoprivrednik želio uzgajati dva različita usjeva, u takvoj situaciji agent temeljen na korisnosti može odrediti koji je usjev bolje uzgajati. Također, ako postoji više ciljeva a nijedan se ne može ispuniti sa sigurnošću, agent može odrediti koje je ciljeve najbolje ispunjavati s obzirom na to koliki je rizik i korist.[1]

Kada se usporede svi tipovi agenata, može se primijetiti da su agenti temeljeni na korisnosti najbolji za korištenje u poljoprivrednoj proizvodnji ako poljoprivrednik želi znati koju od poljoprivrednih kultura treba posaditi te koji je najbolji način za njezino održavanje. Naravno, ta vrsta agenata je i najkompleksnija što znači da ih je najteže implementirati. Alternativa je da se koriste agenti temeljeni na cilju. Za razliku od agenata temeljenih na korisnosti oni ne bi mogli pomagati korisnicima u izabiru različitih ciljeva i ne bi mogli birati najunosnije akcije za dolazak do cilja, ali bi inače mogli pomagati u proizvodnji od početka do kraja. Jednostavni refleksni agenti i agenti s bilježenjem promjena u okolini nisu dovoljno napredni kako bi se koristili za takav zadatak.

3.3. Primjena inteligentnih agenata

Inteligentni agenti se koriste u puno različitih svrha od poljoprivrede pa sve do medicine. U poljoprivredi se, primjerice, agenti mogu koristiti za praćenje usjeva, uklanjanje smetnji pri rastu i djelovanje u svrhu pospješivanja rasta usjeva.[15] U industrijskim postrojenjima agenti se koriste za praćenje rada pogona što uključuje upravljanje radom pogona.[16], [17] Inteli-

gentni agenti se mogu koristiti za stvaranje raznih uradaka od kojih su najpoznatije glazbene skladbe.[18] Pritom valja napomenuti da će takav program koristiti već postojeće skladbe te ih kombinirati da napravi svoju „skladbu“.

U informatici, inteligentni agenti se koriste prilikom izrade programa koji služe za personaliziranje korisničkog iskustva. Primjerice, razne banke koriste inteligentne agente koji komuniciraju s klijentima. Umjetni inteligentni agenti se mogu koristiti i za analiziranje podataka raznih vrsti. Google koristi inteligentne agente sposobne za učenje kako bi poboljšao rad svojeg prevoditelja.[12]

To su samo neke od načina primjene inteligentnih agenata. Umjetni inteligentni agenti su i dalje predmet istraživanja. Štoviše, umjetna inteligencija, u čijem se sklopu proučavaju inteligentni agenti, jedna je od najperspektivnijih disciplina današnjice. Prema tome, može se očekivati da će se u budućnosti naći još mnogo primjena za inteligentne agente i da će se današnji inteligentni agenti usavršiti i raditi svoj posao još bolje.

4. Višeagentni sustavi

Neki su problemi suviše kompleksni kako bi ih rješavali pojedinačni inteligentni agenti. Stvaranjem višeagentnih sustava riješen je taj problem. Višeagentni sustavi su grupe agenata koje surađuju kako bi riješile neki zadatak.[19]

Višeagentni sustavi se uobičajeno koriste za kompleksne zadatke koji se dijele na manje cjeline te svaku cjelinu preuzima jedan inteligentni agent.[19], [20] Na taj način se omogućava rješavanje problema koje bi jedan agent teško riješio zbog prevelike kompleksnosti. Osim što pojednostavljuje zadatke za pojedine agente implementacija višeagentnih sustava je i pouzdanija zato što u slučaju kvara nekog agenta drugi agent iz sustava može preuzeti njegov posao.[19]

4.1. Komunikacija agenata

Kako bi višeagentni sustav radio, agenti moraju međusobno komunicirati. Komunikacija agenata u višeagentnom sustavu je osmišljena na način da agenti mogu komunicirati samo s neposrednim susjedima s kojima su povezani. Na taj način se smanjuje opterećenost agenata koji ne moraju konstantno prosljeđivati poruke već to vrijeme mogu potrošiti obavljajući koristan posao. Samim time sustavi su skalabilniji zato što mogu podnijeti veći broj članova bez da postanu opterećeni porukama.[19]

Ako agenti mogu komunicirati samo sa susjedima, postavlja se pitanje kako da agent dobavi informaciju od nekog udaljenog člana sustava. U višeagentnom sustavu postoje agenti koji sadrže popis poslova koje pojedini agent u sustavu radi. Kada agent treba podatke koje ne može dobiti od susjeda, on komunicira s agentom koji ima popis svih agenata.[19], [21] Na taj način agent može saznati od kojeg člana u sistemu može dobiti potrebne podatke. Nakon toga preostaje samo kontaktiranje tog člana.

Postoje dva načina na koja agent može kontaktirati udaljenog člana. Prvi način je da agenti komuniciraju preko agenta zaduženog za takvo komuniciranje.[19] Taj način je problematičan zato što bi sve udaljene komunikacije morale prolaziti preko tog agenta. Osim toga, kada bi se taj agent pokvario udaljena komunikacija ne bi više bila moguća. Drugi način je da agenti komuniciraju direktno što rješava probleme prve implementacije.[19]

4.2. Svojstva agenata

Kao što svaki inteligentni agent mora posjedovati određena svojstva, tako su i višeagentni sustavi određeni nekim svojim svojstvima. Upoznavanje sa svojstvima je bitno jer se na taj način upoznaje i s načinom rada višeagentnog sustava. Svojstva koja su opisana u idućim odlomcima su:

- Rukovodstvo

- Donošenje odluka
- Heterogenost
- Kašnjenje
- Metrike
- Kretanje agenata

4.2.1. Rukovodstvo

Prvo svojstvo po kojem se višeagentni sustavi mogu razlikovati je posjedovanje ili nedostatak vođe. U višeagentnom sustavu koji nema vođu, agenti sami definiraju svoje zadatke te djeluju kako bi ih postigli. U određenim slučajevima će više agenata biti zaduženo za izvršavanje istog zadatka. U takvim slučajevima djelovanje agenata će biti određeno njihovim dogovorom.[19]

Ako višeagentni sustav ima vođu, on će određivati ciljeve i davati zadatke drugim agentima, a ostali agenti će svoje zadatke izvršavati. Vođa je agent koji je ili predefiniran za tu poziciju ili su ga ostali agenti odredili. Vođe se mogu mijenjati tijekom rada, a osim toga vođa može biti i grupa agenata.[19], [22]

Višeagentni sustavi koji nemaju vođu mogu biti uređeni na više načina. Sustav može biti organiziran na način da je svaki agent specijaliziran za određeno područje. U tom slučaju agenti komuniciraju s drugim agentima te usklade svoja rješenja s drugima.[17], [23], [24]

Drugi način je da se agenti natječu oko toga tko će dobiti resurse kako bi mogao ispuniti svoj zadatak. Naime, svakoj akciji koju pojedini agenti žele odraditi se dodjeljuje „cijena“ koja označava vrijednost usluge. Agenti čije su usluge najvrijednije dobivaju resurse. U ovom načinu agenti također moraju komunicirati kako bi se mogli složiti oko cijena usluga.[17], [25]–[27]

Postoji još jedan način koji potiče suradnju među agentima. Po tom načinu pojedini agenti mogu stvarati rješenja za određene probleme. Nakon što su rješenja stvorena kontaktiraju se drugi agenti koji mogu rješenja testirati i poboljšavati.[17], [28] Na taj bi način konačno rješenje trebalo biti vrlo kvalitetno jer je puno agenata sudjelovalo u njegovoj izradi.

I sustavi s vođom i sustavi bez vođe imaju svoje prednosti i nedostatke. Nedostatak decentraliziranih sustava, odnosno sustava bez vođe je da donošenje odluka zna trajati jako dugo vremena zato što se svi agenti koji su dio sustava moraju dogovoriti oko zadataka. To znači da puno vremena koje bi agenti mogli koristiti za ispunjavanje cilja, oni koriste za donošenje odluka. Nedostatak sustava koji imaju vođe jest da agenti moraju stalno znati koji je član sustava vođa. To znači da u tom slučaju agenti također gube vrijeme na praćenje trenutnog vođe. Trenutno se vode istraživanja vezana za poboljšavanje sustava bez vođa.[29]

4.2.2. Donošenje odluka

U višeagentnim sustavima odluke se donose na temelju podataka koje skupljaju senzori, isto kao i kod pojedinačnih inteligentnih agenata. Također, u većini slučajeva višeagentni sustavi neće donositi svoje odluke samo na temelju dobivenih podataka. Višeagentni sustavi se dijele na linearne i nelinearne prema tome koliko na njih utječu ulazni podaci iz senzora.[19]

Za linearne višeagentne sustave je karakteristično velika veza između ulaznih podataka i donošenja odluke.[19] Linearnost je lakša za implementirati zato što postoji manje varijabli koje se moraju uzeti u obzir kod donošenja odluka. Međutim, višeagentni sustavi nisu uvijek linearni zato što je često potrebno uzeti u obzir i mnoge druge faktore osim podataka sa senzora.

Nelinearni sustavi su oni koji uz ulazne podatke uzimaju i puno drugih faktora u obzir prilikom donošenja odluke.[19], [30] Takvi sustavi su teži za implementaciju ali se i oni koriste jer je u nekim situacijama potrebno uzeti u obzir puno varijabli. Primjerice, višeagentni sustav koji upravlja samovozećim autom mora za svaku svoju odluku uzeti u obzir puno različitih faktora.

Linearni sustavi često ne uzimaju u obzir posljedice svojih akcija. To može biti izrazito loše jer njihove akcije mogu prouzrokovati lošije stanje u budućnosti. Linearno djelovanje jednog člana sustava može drugog člana dovesti u lošiju situaciju. Jedina pozitivna stvar kod linearnog djelovanja je brzina.[17], [31], [32]

Pošto i linearni i nelinearni sustavi imaju svoje pozitivne strane, arhitektura višeagentnih sustava često objedinjuje i jedno i drugo. Naime, sustavi se nerijetko sastoje od tri „sloja“. Najniži „reaktivni“ sloj određuje kako bi se trebalo reagirati na osnovi podataka iz okoliša. Srednji sloj u svoje djelovanje uključuje i znanje koje posjeduju agenti. Gornji sloj arhitekture se bavi društvenim učinkom akcije te upravlja odnosima između slojeva.[17]

4.2.3. Heterogenost

Nisu svi agenti isti, svakog agenta se može razlikovati na više načina.[33] Gledajući raznolikosti agenata, višeagentni sustavi mogu biti homogeni ili heterogeni. Heterogeni sustavi su oni čiji se agenti razlikuju na više načina. Inteligentni agenti u homogenim sustavima imaju ista obilježja i funkcije.[19]

Homogeni višeagentni sustavi mogu biti samo sustavi čiji agenti dobavljaju slične podatke i obavljaju sličan posao.[19] U slučaju da članovi višeagentnog sustava moraju obavljati vrlo različite poslove, sustav će vjerojatno biti heterogen. Višeagentni sustavi su vrlo često heterogeni upravo zato što nije rijetkost da agenti moraju obavljati različite poslove.

4.2.4. Kašnjenje

Agenti koji su dio višeagentnog sustava nikada ne mogu izvršiti dobiveni zadatak u istom trenutku u kojem su ga i dobili. Uvijek se dogodi neko kašnjenje koje može biti veće ili manje, ali bez obzira utječe na potrebno vrijeme izvršavanja zadatka. Vrlo mala kašnjenja nisu problem, ali dosta često su kašnjenja nezanemariva.[19]

Kašnjenja mogu biti izazvana mnogim čimbenicima na koje agenti sustava nemaju utjecaj. Problem s internetskom konekcijom može biti izvor velikog kašnjenja zato što agenti ne mogu niti primati niti slati podatke. Također, činjenica je da višeagentni sustavi imaju ograničene resurse koje koriste svi članovi sustava. Može se dogoditi da jedan agent ne može izvršiti svoj zadatak odmah zato što mu potrebni resursi nisu na raspolaganju pa mora čekati da ih neki drugi agent oslobodi.[19]

Kašnjenje u višeagentnim sustavima može biti prouzrokovano predugim donošenjem odluka. Kao što je rečeno u prijašnjim poglavljima, u sustavima bez vođe svi agenti zajedno donose odluke o akcijama koje će se poduzeti. To znači da nakon što agent donese odluku, on mora čekati na odobrenje ostalih agenata kako bi mogao krenuti s izvršavanjem akcije. Može se dogoditi da agent mora čekati na izvršavanje svoje zadaće jer njegovom zadatku nije pridodana dovoljna važnost, a može se dogoditi i da mora mijenjati svoju planiranu akciju zato što bi smetala drugim agentima.[17], [34]

U slučaju da planirane akcije agenta ne odgovaraju drugim agentima, mora doći do dogovora oko toga kako će se postupati dalje. To vodi do daljnjih kašnjenja zato što su inicijalno agenti napravljeni da izvršavaju zadatke najbolje što znaju. Međutim, pošto je teško postići konsenzus s takvim „razmišljanjem“, postoje prijedlozi da agenti u sustavu prihvate izvršiti zadatak na manje efikasan način, ako će na taj način ranije postići konsenzus.[17], [35], [36] Smatra se da bi taj način bio bolji za višeagentni sustav kao cjelinu, jer bi uštađeno vrijeme koje bi se inače potratilo na pregovaranje moglo pretvoriti u koristan rad, što bi u konačnici smanjilo kašnjenje i povećalo efikasnost čitavog sustava.

4.2.5. Metrike

U većini višeagentnih sustava agenti se moraju dogovarati kako bi sustav funkcionirao. Sustavi u kojima se agenti dogovaraju o vrijednostima parametara mogu se prema tome podijeliti. Prema toj podjeli postoje višeagentni sustavi prvog reda (first order), drugog reda (second order) i trećeg reda (higher order).[19]

Agenti iz sustava prvog reda moraju se dogovarati oko vrijednosti samo jednog parametra. Sljedeći istu logiku, sustavi drugog reda se dogovaraju za vrijednosti dva parametra.[19], [37]Sustavi višeg reda su kompleksniji od prethodna dva.

Sustavi trećeg reda su oni koji koriste jednu ili dvije metrike i treću vrijednost povezanu s tim metrikama kako bi dobili potreban rezultat.[19], [38] Primjerice, samovozeći automobil mora biti svjestan udaljenosti od drugih automobila u prometu i svoje brzine, pomoću tih vrijednosti može izračunati akceleraciju i brzinu drugih automobila. Nakon što su mu poznate te varijable samovozeći automobil može postaviti svoju brzinu na odgovarajuću vrijednost.

4.2.6. Kretanje agenata

Neki višeagentni sustavi dozvoljavaju agentima ulazak i izlazak iz sustava, a drugi ne. Prema tome se sustavi mogu podijeliti na statične i dinamične. Dinamični sustavi su oni u

kojima agenti mogu slobodno ulaziti i izlaziti dok iz statičnih ne mogu.[19]

Članovi statičnih višeagentnih sustava se ne mogu mijenjati, što znači da se u svakom trenutku zna koji su agenti u sustavu i koja je njihova svrha. Za razliku od statičnih, agenti u dinamičnim sustavima moraju pratiti koji se agenti trenutno nalaze u sustavu i koja je njihova svrha.[19] Ne moraju svi agenti pratiti ulaske i izlaske članova sustava već samo neki, primjerice agenti koji su zaduženi za komunikaciju mogu biti zaduženi i za praćenje ulazaka i izlazaka iz sustava.

Dinamički sustavi su teži za implementaciju zato što agenti moraju moći detektirati ulaske i izlaske, također agenti dinamičkih sustava moraju stalno ažurirati podatke o trenutnim agentima u sustavu. S druge strane, agenti u dinamičkim sustavima mogu uvijek biti oni koji su specijalizirani za traženi posao, dok u statičnim sustavima moraju uvijek biti svi agenti koji će biti potrebni za obavljanje posla. Također, ako se dogode neke nepredviđene okolnosti, u sustav može ući agent koji je najsposobniji za rješavanje situacije, dok u statičnim sustavima situaciju moraju rješavati agenti koji možda nisu napravljeni za to.

4.3. Primjene višeagentnih sustava

Od druge polovice prošlog stoljeća, kada su prvi put korišteni, pronađene su brojne namjene za višeagentne sustave. Od kontroliranja nuklearnih reaktora do financijskih poslova, višeagentni sustavi se mogu pronaći velikom broju ljudskih djelatnosti. Ovaj dio će predstavljati osvrt na brojne mogućnosti primjene sustava.

4.3.1. Internet

Od interneta u oblacima do društvenih mreža, internet je mreža na kojoj su višeagentni sustavi široko rasprostranjeni. Sustavi su potrebni zato što je internetska mreža vrlo velika i rasprostranjena te ima puno korisnika. Iz tih razloga bilo bi ju teško održavati bez upotrebe agenata.

Internet u oblaku (cloud computing) je sistem koji omogućuje povezivanje različitih stvarnih ili virtualnih računala u svrhu pružanja usluga. Na računala koja su dio oblaka (cloud) se stavljaju resursi koji će korisnicima koristiti.[39], [40] Korisnici mogu koristiti resurse s računala u oblaku pomoću virtualizacije koja omogućuje stvaranje virtualnih računala.[19]

Resursi dostupni na računalima u oblaku se koriste od strane mnogih različitih korisnika. Kako bi se resursi mogli efikasno distribuirati korisnicima potrebno je upravljati njihovom količinom i dijeljenjem. Višeagentni sustavi nadgledaju zahtjeve za korištenjem resursa te prema određenim parametrima dodjeljuju korisnicima resurse. Sustavi su zaslužni i za odnose sa korisnicima te određuju koliko su pojedini resursi vrijedni u odnosu na njihovu potražnju.[39], [40]

Kako su resursi na računalima u oblaku konačni, višeagentni sustavi se na različite načine brinu da se ne potroši previše resursa. Povećavanje cijena je jedan od načina za smanjenje potrošnje resursa.[19], [41], [42] U krajnjem slučaju sustavi mogu i odgoditi korisnikov zahtjev

za korištenjem resursa te na taj način osigurati potrošnju resursa u normalnim granicama.[39], [40]

Kako bi svojim korisnicima pružili što bolje sadržaj, društvene mreže koriste višeagentne sustave za personaliziranje iskustva.[19] Personaliziranje funkcionira na način da agenti uzmu u obzir stranice koje korisnik prati i teme o kojima raspravlja pa na temelju toga korisniku ponude druge stranice koje bi ga mogle interesirati. Agenti koji se koriste u takvim situacijama su obično sposobni za učenje zato što postaju sve točniji što više korisnik koristi društvenu mrežu.

Višeagenti sustavi se mogu koristiti i za detektiranje prijetnji s interneta. Takvi agent moraju biti sposobni za učenje jer se moraju konstantno upoznavati s novim sigurnosnim prijetnjama. U takvim sustavima neki agenti su zaduženi za preuzimanje podataka koji dolaze s interneta na računalo. Drugi agenti služe za pregledavanje podataka za maliciozan sadržaj. U slučaju da je pronađen zlonamjeran kod, agenti koji su pregledavali sadržaj javljaju agentima koji će prenijeti poruku. Poruka će se prenijeti agentima koji su zaduženi za odlučivanje o akciji koja se treba poduzeti kao odgovor na prijetnju.[19], [43]

Višeagentni sustavi korišteni za sigurnost su dobar primjer heterogenih sustava. U njemu se neki agenti specijaliziraju za dobavljanje podataka, drugi za pregledavanje, treću za prenošenje poruka i četvrti za odlučivanje o daljnjem postupku. U navedenom primjeru se vidi da agenti u sustavu moraju surađivati i komunicirati kako bi mogli uspješno obavljati svoj posao.[19], [43]

Korisnici interneta šalju i primaju podatke. Podaci putuju internetom u paketima koji se preusmjeravaju puno puta od izvora do odredišta. Preusmjeravanje paketa je jedna od mogućih korisnosti višeagentnih sustava. Za takvo što su potrebni mobilni agenti koji putuju do svakog čvora. Takvim postupkom višeagentni sustav stvara mapu čvorova koju može koristiti za efikasno preusmjeravanje paketa.[19], [44] Na taj način se mogu pronaći bolji putovi za paket što može smanjiti vrijeme potrebno da paket stigne na odredište.

4.3.2. Robotika

Korištenje inteligentnih agenata za upravljanje robotima jedan je od najpoznatijih načina njihova korištenja, sudeći prema učestalosti pojavljivanja u popularnoj kulturi. Za rad robota potrebni su vrlo kompleksni višeagentni sustavi koji moraju obavljati razne akcije kako bi robot funkcionirao. Iz tih razloga znanstvenici još uvijek istražuju kako poboljšati rad robota.

Smatra se da je prilikom implementacije višeagentnog sustava za upravljanje robotima najbitnije postići koordinaciju među robotima i omogućiti im planiranje puta. Kako bi se to omogućilo smišljen je višeagentni sustav koji povezuje sklopovlje robota s programskom podrškom robota. Sklopovlje robota se sastoji od agenata koji koriste senzore pokreta, kamere i slične naprave kako bi dobili informacije o fizičkom okolišu u kojem se robot nalazi. Podaci koje su ti agenti primili se šalju agentima koji su dio programske podrške. Agenti softwarea koriste dobivene podatke kako bi analizirali okoliš i na taj način odlučili gdje će robot ići.[19], [45]

Može se primijetiti da je višeagentni sustav zadužen za upravljanje robotom također sastavljen od više drugačijih agenata, odnosno sustav je heterogen. To nije ništa čudno za

takav kompleksan proces kao što je upravljanje robotom. Iz ovog primjera se također može vidjeti da okoliš iz kojeg agenti primaju podatke može i ne mora biti fizički. Agenti sklopovlja su primali podatke iz fizičkog okoliša te su za to koristili odgovarajuće alate. Agenti programske podrške primali su podatke nisu imali kamere i senzore jer oni nisu primali podatke iz fizičkog okoliša.[19]

4.3.3. Proizvodni pogon

Proizvodni pogon se sastoji od puno različitih dijelova koji moraju raditi usklađeno kako bi proizveli neki proizvod. Višeagentni sustavi se mogu koristiti kako bi uskladili rad svih dijelova pogona. U takvom slučaju, agenti predstavljaju dijelove tvornice.[16], [17] Svaki agent ima svoj zadatak koji mora odraditi, a kako bi odradili svoje zadatke točno, agenti moraju komunicirati međusobno.

Gotovo svaki pogon ima svog poslovođu koji govori ljudima što bi i kako trebali napraviti. Višeagentni sustavi zaduženi za proizvodnju također mogu imati vođu koji će svakom agentu dati određeni posao u proizvodnji. Sustavi koji su zaduženi za veće tvornice mogu imati izrađenu hijerarhiju gdje će neki agent biti vođa te će druge agente rasporediti na različite pogone. Svaki pogon također može imati vođu koji će agente rasporediti na proizvodne procese, a svaki proces će imati vođu koji će rasporediti agente na određenu komponentu.[16], [17] Takva struktura je dobra jer ne zahtjeva puno surađivanja i pregovaranja među agentima već svaki agent ima određeni posao koji treba uraditi.

4.3.4. Kontrolna procesa

Mnoge djelatnosti, poput nuklearnih elektrani, bitno je izrazito dobro kontrolirati kako se ne bi dogodila katastrofa. Iz tog razloga se u određenim procesima koriste agenti koji se brinu da sve funkcionira kako bi trebalo. Agenti koji su dio sustava za nadgledanje su zaduženi za određeni dio procesa. Svaki agent može pomoću raznih senzora pregledavati svoj dio procesa. Nakon što su prikupili podatke agent ih može evaluirati ili pomoću komunikacijskog agenta poslati drugom agentu koji će provjeriti podatke te otkriti postoji li kakva greška u radu.

Djelatnosti kojima je potrebna kontrola uključuju korištenje nuklearne energije za proizvodnju struje, sudaranje čestica, prenošenje električne energije i druge. Svi navedeni procesi uključuju korištenje velikih količina energije pa su samim time vrlo opasni. Kontrola procesa se može koristiti i za druge manje opasne radnje poput kontroliranja svemirskih letjelica i praćenja klimatskih promjena.[17], [46]

4.3.5. Financijske djelatnosti

U kontekstu financija, višeagentni sustavi se mogu koristiti na mnogo različitih načina. Primjerice, korisnici mogu koristiti sustave kako bi analizirali poslovanje tvrtki ili kako bi pratili kretnje na tržištu. Osim toga, agenti mogu analizirati poteze korisnika s obzirom na tržište i savjetovati korisnika o boljem načinu ulaganja novca.

U višeagentnim sustavima koji se bave financijama postoje agenti koji osluškuju financijske vijesti i skupljaju podatke u tržištu i tržišnim kretanjima. Ti agenti će komunicirati s drugim agentima koji će koristiti podatke kako bi obavili određeni zadatak.[17], [47] Kao što je opisano u prošlom poglavlju zadaci agenata mogu biti različiti pa će agent koji analizira podatke i šalje odgovor korisniku morati dobiti odgovarajuće informacije od drugih agenata. Primjerice, agenti koji primi podatke može filtrirati podatke koji su korisni za korisnika.

U ovom primjeru, kao i u svim ostalim, vidi se važnost komuniciranja među agentima. Neki agenti u sustavu su zaduženi za prikupljanje podataka, drugi za analiziranje podataka, a mogu postojati i komunikacijski agenti pomoću kojih ostali komuniciraju. Kako bi sustav funkcionirao pravilo izrazito je važno da agenti komuniciraju kvalitetno. Primjerice, agent zadužen za analiziranje mora komunicirati s agentom koji skuplja podatke jer inače taj agent neće znati koje podatke treba skupiti. Isto tako, nakon što agent skupi podatke, podaci se moraju što brže prenijeti drugom agentu na analizu kako bi korisnik što brže dobio odgovor. Sve to zahtijeva vrlo razvijen sustav komunikacije.

4.4. Modeliranje temeljeno na agentima

Simulacije se koriste kako bi se promatrane situacije mogle proučavati u kontroliranim uvjetima. Uspješna simulacija može značiti puno za rješavanje nekog problema. Modeliranje temeljeno na agentima se koristi za stvaranje realističnih simulacija pri čemu svaki agent simulira jedan entitet u stvarnom svijetu.

Modeliranje temeljeno na agentima je implementacija višeagentnog sustava u svrhu stvaranja modela odnosno simulacije u kojem svaki agent sustava predstavlja jednog stvarnog sudionika procesa.[48] Dobiveni model će simulirati odnose između pojedinih agenata.[49] Ako je model dobro napravljen prikazivat će situaciju vrlo sličnu stvarnoj.

Potrebno je obratiti pažnju na neke stvari kako bi modeliranje bilo uspješno. Vrlo je bitno da agenti koji se koriste za stvaranje modela budu autonomni, odnosno da nisu ovisni o ponašanju bilo kojeg drugog dijela simulacije.[48] To znači da agenti moraju biti sposobni prilagođavati se okolišu i u situaciji reagirati onako kako bi reagirali akteri u stvarnom svijetu. Također, potrebno je odrediti međusobne odnose agenata i odnos između agenata i okoliša.[48] Primjerice, kod modeliranja supermarketa je potrebno odrediti koji su agenti kupci a koji trgovci i potrebno je odrediti kako se kupci ponašaju kad su u supermarketu. Osim na agente i interakciju, potrebno je paziti i na izradu okoliša za model. Okoliš bi trebao biti dovoljno detaljan da model može biti vjeran stvarnom svijetu, a dovoljno poopćen da ga nije preteško izraditi.[48]

Modeli predstavljaju aproksimativan prikaz stvarnog stanja, pa se postavlja pitanje zašto se ne koristi drugi način za dobivanje potrebnih informacija. Modeliranje se koristi, između ostalog, zato što daje informacije o stanju svakog pojedinog sudionika simulacije kao i o stanju čitavog modela.[48] Većina drugih metoda ne bi davala informacije o ponašanjima pojedinih sudionika. Osim tog, postoje specifične situacije u kojima se ne isplati koristiti bilo koju drugu metodu osim modeliranja temeljnog na agentima.

Modeliranje se uvijek koristi kada su interakcije među agentima kompleksne.[50] U tak-

vim situacijama je teško napraviti matematički model jer postoji puno mogućih scenarija što izrazito komplicira izradu takvog modela. Modeliranje temeljeno na agentima je u tom slučaju idealno zato što agenti u modelu znaju kako se trebaju ponašati te će oni sami stvarati interakcije.

U situacijama čiji se sudionici razlikuju također je potrebno koristiti modeliranje.[50] Za primjer se mogu uzeti modeli prometa. U prometu su svi sudionici druge osobe koje se razlikuju po svom ponašanju. Modeliranje se koristi i za situacije u kojima sudionici mogu mijenjati svoj položaj.[50] Recimo, sudionici prometa će vrlo vjerojatno poći drukčijom rutom ako vide da se negdje dogodilo zagušenje.

4.4.1. Prednosti modeliranja temeljenog na agentima

U prošlim odlomcima objašnjeno je u kojim se slučajevima treba koristiti modeliranje. U ovom poglavlju bit će objašnjeno iz kojih razloga je dobro koristiti modeliranje i zašto je modeliranje u određenim slučajevima bolje od drugih načina. Za početak je dobro opisati tri glavne prednosti koje donosi stvaranje modela s višeagentnim sustavima.

U stvarnom svijetu se često događaju stvari koje mnogi ljudi ne bi očekivali, stvari koje su prije iznimka nego pravilo. Kada se rade bilo kakvi izračuni ili simulacije s ciljem dobivanja podataka o svijetu, bitno je uključiti u obzir i te „rubne“ situacije. Međutim, većina metoda ima velikih problema sa uvrštavanjem takvih situacija u svoj izračun. Način na koji modeliranje funkcionira omogućuje stvaranje „rubnih“ situacija što znači da su takvi slučajevi uzeti u obzir u konačnom rezultatu.

Za primjer „rubnih“ situacija mogu se uzeti gužve u prometu.[50] Kada se krenu stvarati gužve u prometu normalno bi bilo za očekivati da će vozači reagirati na način da se gužve smanje. Međutim vrlo često se događa upravo suprotno, reakcije vozača dodatno doprinose stvaranju gužvi. Modeliranje temeljeno na agentu bi moglo reproducirati takav ishod zato što su agenti programirani da reagiraju na način kako bi reagirali i vozači, a neke druge metode ne bi uzele u obzir takav ishod jer takva reakcija vozača nije intuitivna pa ne bi bila ni razmatrana.

Modeliranje se koristi i kada se želi opisati ponašanje pojedinaca u nekoj situaciji, primjerice kupaca u supermarketu. Modeli supermarketa se mogu praviti zato što trgovci žele vidjeti kako ubrzati kupovinu da kupci imaju osjećaj kao da nisu puno vremena proveli u trgovini ili se prave da trgovci vide kako police u supermarketu trebaju biti postavljane da bi se maksimalno povećala kupovina ljudi.[50], [51] Modeliranjem supermarketa može se vidjeti kako različit raspored polica ili različita brzina na blagajni utječu na ponašanje kupaca. Na taj način trgovci mogu vidjeti koji je optimalan način rada supermarketa.

Treći razlog zašto je modeliranje temeljeno na agentima dobro jest njegova fleksibilnost koja se očituje na više različitih načina. Naime, agenti koji stvaraju model se uvijek mogu prilagođavati stvarnom svijetu mijenjajući neke svoje karakteristike.[50] Primjerice, u slučaju da je krivo pretpostavljeno kako se sudionici događaja ponašaju, ponašanje agenta koji je dio modela je vrlo lako izmijeniti. Takva fleksibilnost ne postoji kod velikog broja drugih metoda. Modeliranje je fleksibilno i zato što je moguće vrlo lako izmjenjivati broj agenata koji sudjeluju

u simulaciji.[50] Kada se modelira supermarket moguće je mijenjati broj blagajnika u simulaciji kako bi se pronašao njihov optimalan broj u određenoj situaciji.

4.4.2. Nedostatci modeliranja temeljenog na agentima

Kako bi se mogla stvoriti potpuna slika o modeliranje, nije dovoljno raspravljati samo o prednostima. Treba imati na umu da modeliranje ima i svoje nedostatke koje u nekim slučajevima otežavaju izradu korektnog modela. Bitno je da se željeni model može izraditi bez ikakvih nedostataka pa je poželjno prije izrade provjeriti hoće li nedostatci utjecati na kvalitetu modela.

Jedan od poteškoća u izradu modela je postizanje željene razine detaljnosti.[48] Manjak detaljnosti znači da model ne predstavlja stvarni svijet, a ako je to slučaj onda se iz modela ništa korisno ne može zaključiti. To znači da stvoreni model uopće neće biti koristan ako nije detaljan. Tehnički gledano, model ne može biti previše detaljan zato što detaljnost znači vjerniji prikaz stvarnog svijeta. Međutim, što je model detaljniji to ga je teže implementirati jer se mora prikupiti više podataka o predmetu modeliranja. Zato je prije izrade modela potrebno provjeriti koliko detaljan treba biti i prema tome skupljati podatke potrebne za njegovu izradu.

Svi agenti koji sudjeluju u modeliranju su povezani, samim time izmjena jednog agenta utječe na druge. Zato potreba za izmjenom jednog agenta može voditi prema potpuno drugom modelu.[48] Prilikom izrade simulacije prometa u nekom gradu agentima se moraju dodati određena svojstva koja će simulirati ponašanje vozača, ako vozači nisu vrlo dobro okarakterizirani dana svojstva se neće poklapati sa stvarnom situacijom. To će značiti da model nije dobro napravljen. U ovom kontekstu se može spomenuti i problem replikacije ljudskog ponašanja. Naime, modeliranje temeljeno na agentima vrlo često zahtijeva simuliranje ljudskog ponašanja od strane agenata. Problem u tome je što su ljudi vrlo kompleksna bića te je jako teško predvidjeti kako će se neki ljudi ponašati.[50] Iz razloga što nikad nije moguće u potpuno točno simulirati ponašanja stvarnog svijeta simulacije nikada nisu jednake događajima u stvarnom svijetu, međutim simulacije mogu biti vjeran prikaz svijeta ako im se pristupi ozbiljno.

Ponekad je model pretežak i prevelik za implementaciju. To se može dogoditi iz više razloga. U prošlom odlomku je spomenuta kompleksnost implementiranja ljudi od strane agenta. U slučaju da je stvar koja se želi modelirati previše kompleksna za vjeran prikaz od strane agenata, model nije moguće stvoriti.[48] Naime, u tom slučaju agenti neće vjerno imitirati ono što bi trebali pa će samim time čitava simulacija biti daleko od točne.

Neke modele se ne može napraviti zato što još uvijek ne postoje kapaciteti za to. Ponekad je za izradu modela potrebno uključiti jako puno agenata ili je okoliš u kojem se model nalazi jako velik i kompleksan.[48] Za takve slučajeve su potrebna snažna računala. Iako su danas računala vrlo jaka, neke simulacije su i dalje vrlo problematične za izvođenje.[50]

Nedostatak modeliranja temeljenog na agentima je i činjenica da je za svaku situaciju potrebno napraviti drugi model. Razlog tomu je činjenica da se svaka situacija razlikuje ili po okolišu ili sudionicima. Primjerice, kada se radi model za regulaciju prometa, on se može uzimati u obzir samo u kontekstu mjesta u kojem je napravljen, a za neko drugo mjesto mora se raditi drugi model koji bi tome bio prikladniji.

4.4.3. Primjeri modeliranja temeljenog na agentima

Modeliranje se može iskoristiti za bilo koju situaciju koja se želi analizirati, a previše je kompleksna za druge načine analize. Kontrola prometa je jedan od najpoznatijih slučajeva korištenja modeliranja. Osim toga, modeliranje se koristi za optimiziranje poslovnih procesa, analizu tržišta i mnoge druge stvari.

Mnogi gradovi imaju loš prometni sustav koji u određenom dijelu dana postane zagušen. Pošto su gužve u prometu izvor iritacije za mnogobrojne vozače, često se postavlja pitanje kako ih smanjiti. Neki gradovi su uveli naplaćivanje vožnje kroz svoj određena područja, međutim to nije trajno rješenje i obično nikada ne pomogne. Kako bi se gužve smanjile obično je potrebno na neki način izmijeniti prometnu infrastrukturu. Kao pomoć u analizi trenutne i mogućih budućih infrastruktura dobro je koristiti modeliranje.

Promet je povoljan za izradu simulacija zato što je većina podataka o njemu poznata. Prva stvar kod izrade simulacije prometa nekog grada je prikupljanje podataka. Pritom je naravno potrebno izraditi okoliš, u ovom slučaju je to prometna mreža. Nakon toga je potrebno dodati sudionike u prometu koje će simulirati agenti. Kod izrade sudionika u prometu bitno je znati kako se ponašaju vozači, pješaci, biciklisti i ostali. Također, bitno je znati koliko ima određenih sudionika u određenom trenutku. Na kraju potrebno je znati otprilike kuda sudionici prometa idu i zašto.[50] Imajući sve te informacije može se stvoriti simulacija prometa nekog grada te na temelju te simulacije se može vidjeti kako bi se situacija mogla poboljšati.

U prijašnjim poglavljima je spominjan primjer supermarketa. Modeli supermarketa se izrađuju kako bi se moglo vidjeti, između ostalog, na koji način se police mogu optimalno postaviti, koliko će kupci čekati na blagajni bez iritacije i slično. Simulacija se izrađuje na isti način kao i model prometa. Potrebno je prikupiti sve podatke o uređenju supermarketa, podatke o ponašanju i broju kupaca i podatke o zaposlenicima. Nakon izrade modela može se vidjeti kako optimalno urediti supermarket da ljudi kupe što više stvari bez da provedu previše vremena unutar njega, koliko vremena mogu provesti na blagajni bez da postanu nestrpljivi i koliki je optimalan broj ljudi u supermarketu.[50], [51]

Financijsko tržište karakteriziraju periodi rasta i pada. Postoje teorije da je pomoću modeliranja moguće ublažiti periode pada.[49] Smatra se da u trenutnom načinu funkcioniranja tržišta postoje greške te da je potrebno uvesti određene izmijene. Osim toga modeli financijskog tržišta pomažu i ulagačima prilikom ulaganja, a i zaposlenima u financijskoj industriji.

Kada se priča o financijskom tržištu ponajprije se misli na tržište dionica i obveznica. Takvo tržište moguće je replicirati modeliranjem temeljenom na agentima jer je poznat okoliš i poznato je ponašanje dioničara. Modeli se obično rade za jedan burzovni indeks (primjerice CROBEX). Model nekog indeksa može se upotrijebiti na više načina. Ljudi koji rade na burzi mogu koristiti model da vide kako će različite promjene u poslovanju utjecati na vrijednosti indeksa. Modeli se mogu koristiti za pronalazak najbolje minimalne promjene cijene dionice koja će se prikazati.[50] Dioničari mogu koristiti modele kako bi predvidjeli kretanja dionaca s nekog indeksa na burzi. Za takve modele je prilikom modeliranja potrebno uzeti u obzir i vijesti izvana na koje utječu na kretanje vrijednosti dionica.

4.5. Usporedba višeagentnih sustava i modeliranja temeljenog na agentima

U prethodnim poglavljima opisani su i višeagentni sustavi i modeliranje temeljeno na agentima. Iz njihovih opisa se može vidjeti da između ta dva pojma postoji puno sličnosti, ali oni nisu isti. U ovom poglavlju će se opisati sličnosti i razlike ta dva pojma. Za lakše razumijevanje njihovih razlika bit će navedeni primjeri njihove uporabe u istom okolišu.

U poglavlju o modeliranju je navedeno da su modeli temeljeni na agentima zapravo višeagentni sustavi. Prema tome, postavlja se pitanje kako je moguće da postoji razlika između ta dva pojma. Višeagentni sustavi koji se koriste za modeliranje su zapravo posebna vrsta višeagentnih sustava koji služe drukčijoj svrsi od ostalih sustava. Naime, većina višeagentnih sustava služi za pomoć pri rješavanju kompleksnih zadataka.[19] Za razliku od toga, modeliranje temeljeno na agentima služi za objašnjavanje i analiziranje određenog sustava.[50] S obzirom na njihove različite namjene, može se zaključiti da se sustavi i modeliranje u nekim aspektima i razlikuju.

Modeliranje i višeagentni sustavi su povezani pojmovi stoga je logično da posjeduju određene sličnosti. Primjerice, oba pojma zahtijevaju veći broj agenata kako bi mogli funkcionirati. Također, komunikacija im je oboma nužna za uspješan rad. I agenti koji služe za modeliranje i agenti unutar sustava mogu posjedovati jednaka svojstva poput autonomnosti, racionalnosti i drugih. Za kraj, i jedno i drugo pomažu ljudima u obavljanju određenih poslova. Opisane sličnosti su većinom poopćena svojstva višeagentnih sustava tako da nije ni čudno da im je to zajedničko.

Kada se pojmovi pogledaju bolje mogu se vidjeti razlike. Višeagentni sustavi služe za obavljanje konkretnih poslova poput kontroliranja ispravnosti nuklearnih reaktora. Samim time, agenti koji su dio takvih sustava su napravljeni kako bi obavljali posao koji im je dodijeljen. Međutim, agenti koji su služe za modeliranje su napravljeni kako bi oponašali stvarne čimbenike određene situacije. Osim toga, agenti u modeliranju pomažu za analizu čimbenika koje trebaju oponašati te svi kolektivno pomažu u analizi čitavog modela. Dakle, ti agenti nisu napravljeni da bi obavljali nikakav specifičan posao.

Uloga komunikacije u modeliranju i višeagentnim sustavima je također drukčija. U višeagentnim sustavima agenti moraju komunicirati kako bi mogli uspješno obavljati svoj posao. Naime, bez komunikacije ne bi mogli surađivati i ne bi mogli dobiti potrebne podatke za svoj rad. Za razliku od toga, agenti u modeliranju oponašaju sudionike nekog stvarnog okoliša, a sudionici stvarnim okoliša komuniciraju. Samim time, agenti modela moraju komunicirati. Štoviše, komunikacija među agentima modela se obično analizira zato što može pomoći u shvaćanju stvarnosti.

Za konkretan primjer razlika između dva pojma može se uzeti financijsko tržište. Korisnik može koristiti višeagentni sustav za izdvajanje sebi važnih vijesti ili za kontroliranje svojeg rada. Modeliranje temeljno na agentima se može koristiti za analiziranje tržišta, pretpostavljanje kretanja dionica ili analiziranje rada neke institucije. Dakle, višeagentni sustavi se koriste za obavljanje nekog konkretnog posla koje korisniku mogu olakšati rad, dok se modeliranje koristi

za izradu analiza i predviđanje koje korisniku omogućuju bolja ulaganja u budućnosti.

Na primjeru proizvodnog poduzeća razlika se može vidjeti i bolje. Višeagentni sustavi služe za kontroliranje pojedinih strojeva u pogonu. Odnosno, višeagentni sustavi su zaduženi za proizvodnju i inspekciju strojeva za bilo kakvim greškama. Modeliranje temeljeno na agentima može služiti za analiziranje efikasnosti proizvodnje: modeliranjem se može provjeriti koji je omjer najoptimalniji za izradu proizvoda ili postoji li neki drugi način proizvodnje koji bi ju usavršio. Dakle, modeliranje služi za analizu, a višeagentni sustavi za obavljanje poslova korisnika.

5. Križić-kružić protiv računala

Praktični dio ovog rada koji se tiče izrade jednostavnog inteligentnog agenta sastoji se od izrade programa za igru križić-kružić u kojoj korisnik igra protiv računala. Za pobjedu u igri korisnik ili računalo mora imati tri usporedna križića ili kružića. To je moguće ostvariti na osam načina. Pobijediti se može ako se u bilo koja tri horizontalna reda postave odgovarajući znakovi. Također, isto vrijedi i za postavljanje znakova u vertikalne redove. Zaključno, igrač može pobijediti ako postavi tri svoja znaka na bilo koju dijagonalu.

Igra je zamišljena na način da računalo ima prvi potez. Prva stvar koju računalo mora napraviti jest donijeti odluku o željenom načinu pobjede. To znači da računalo mora odlučiti kojim se od osam gore navedenih načina želi služiti kako bi došlo do pobjede. Nakon donošenja odluke, računalo će povući potez koji će voditi prema pobjedi za koju se odlučilo.

Nakon što je računalo odigralo svoj potez, na redu je korisnik. Može se dogoditi da korisnikov potez onemogući zamišljenu pobjedu računala. Primjerice, računalo je htjelo pobijediti stavljajući križiće na prvi horizontalni red, međutim korisnik je stavio kružić u jedno od polja u tom redu. U tom slučaju računalo mora izabrati novi način pobjede te odigrati odgovarajući potez.

Računalo se mora pobrinuti i da ga korisnik ne pobijedi. U tu svrhu računalo prije svakog svog poteza mora provjeriti može li ga korisnik u idućem potezu pobijediti. Ako može onda to treba spriječiti, inače računalo igra kako bi i inače igralo.

5.1. Programski kod

U prošlom poglavlju rečeno na kojih osam načina se u igri može ostvariti pobjeda. Prema tome, u programskom kodu se koristi enumeracija u kojoj se nabraja svih osam načina. Pomoću te enumeracije računalo će kasnije odabrati kojim načinom želi pobijediti.

```
enum enumPobjeda
{
    prviVodoravno,
    drugiVodoravno,
    treciVodoravno,
    prviOkomito,
    drugiOkomito,
    treciOkomito,
    goreLijevo,
    goreDesno
};
```

Svaka gore navedena enumeracija ima svoju brojevnu vrijednost. Primjerice, brojevna vrijednost od prviVodoravno je 0, a brojevna vrijednost od drugiVodoravno je 1. Na osnovu toga je napravljena funkcija kojom se nasumično odabire jedna od vrijednosti u enumeraciji.

```
public int biranjePoteza()
```

```
{
    Random rnd = new Random();
    int broj = rnd.Next();
    broj = broj % 8;
    return broj;
};
```

U funkciji se deklarira generator nasumičnih brojeva te se generira jedan takav broj. Nakon toga, pošto u enumeraciji postoji samo 8 vrijednosti, traži se ostatak dijeljenja nasumičnog broja s 8. Dobiveni broj će sigurno biti jednak nekoj od brojevnih vrijednosti u enumeraciji.

```
if (brojNiza == (int)enumPobjeda.prviVodoravno)
{
    prviVodoravno();
}
if (brojNiza == (int)enumPobjeda.drugiVodoravno)
{
    drugiVodoravno();
}
```

Gore navedeni kod je dio funkcije OdigravanjePoteza(). U varijablu brojNiza sprema se vrijednost koju je vratila funkcija biranjePoteza(). Nakon toga se u jednoj od funkcija uspoređuje vrijednost varijable brojNiza sa brojevnim vrijednosti iz enumPobjeda te se izvršava odgovarajuća funkcija. Gore je napisan dio funkcije za prva dva načina pobjede, a u kodu postoje funkcije i za ostale načine. U nastavku su prikazani dijelovi funkcije prviVodoravno() pomoću kojih će biti pojašnjeno čemu ta funkcija služi.

```
if (btn1.Text == "O")
{
    brojNiza = biranje.biranjePoteza();
    return;
}
```

U navedenom grananju provjerava se je li protivnik označio polje koje bi onemogućilo pobjedu računalu. U slučaju da je polje označeno računalu u funkciji biranjePoteza() bira novi način pobjede. Te ponovno ulazi funkciju OdigravanjePoteza() te iz nje ulazi u funkciju za odgovarajući način pobjede. To se ponavlja sve dok se ne pronađe odgovarajući način pobjede.

```
if (!provjeriDostupnost())
{
    odigrajPotez();
    return;
}
```

Funkcija prviVodoravno() i sve srodne funkcije sadrže gore napisani dio koda. U funkciji provjeriDostupnost() provjerava se može li uopće računalo pobijediti ili se nalazi u situaciji u kojoj niti jedan potez neće pomoći u pobjedi. Ukoliko se nalazi u situaciji u kojoj pobjeda nije moguća izvršava se kod funkcije odigrajPotez() koji omogućuje popunjavanje nasumičnog polja.

```
int brojGumba = biranje.odrediGumb();
switch (brojGumba)
{
    case 0:
        if (btn4.Text == "␣")
        {
            btn4.Text = "X";
            btn4.Enabled = false;
            odigrano = true;
        }
        break;
```

Kada pobjeda jeste moguća na način koji je odabran, računalo mora izabrati koje će slobodno polje popuniti. To radi na način da pomoću funkcije odrediGumb() odredi broj od 0 do 2. Nakon toga, računalo upisuje odgovarajući znak u odabrano polje ako je ono slobodno te onemogućuje daljnje unose u to polje.

Prije nego što računalo napravi svoj potez ono mora provjeriti dvije stvari. Računalo mora provjeriti je li netko već pobijedio i mora provjeriti je li korisnik u mogućnosti pobjede. To radi na način da u funkciji OdigravanjePoteza() poziva funkcije Pobjeda() i obrana().

```
if (btn1.Text == "X" && btn2.Text == "X" && btn3.Text == "X")
{
    txtPobjeda.Text = "Pobijedilo␣je␣čraunalo!";
    btn1.BackColor = Color.Red;
    btn2.BackColor = Color.Red;
    btn3.BackColor = Color.Red;
    disableButton();
}
```

U isječku koda funkcije Pobjeda() može se vidjeti da računalo provjerava je li netko pobijedio provjeravajući pomoću selekcije jesu li polja ispunjena na pobjednički način. Ako jesu, onda računalo ispiše tko je pobijedio te oboja odgovarajuća polja, također korisnik nakon toga dobije mogućnost za pokretanje nove igre. Gore je naveden kod za jednu od situacija, a u kodu su pokrivene i sve ostale.

```
if(btn1.Text == "O" && btn2.Text == "O" && btn3.Text == "␣")
{
    btn3.Text = "X";
    btn3.Enabled = false;
    odigrano = true;
    return true;
}
```

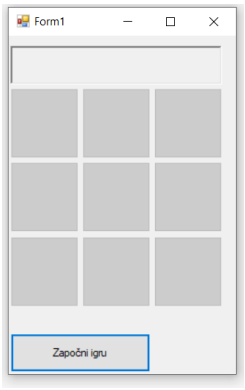
Gore se može vidjeti jedna od selekcija koja pregledava može li korisnik pobijediti. Ako korisnik može pobijediti onda računalo stavlja svoj znak na to mjesto te na taj način korisnik više ne može pobijediti tim načinom. Kao što računalo provjerava situaciju na ovom isječku, ono provjerava i sve ostale situacije u kojima bi korisnik mogao u idućem potezu pobijediti.Ako

se umjesto znaka „O“ provjerava znak „X“ taj se kod može koristiti kako bi računalo provjerilo može li pobijediti u svom trenutnom potezu. Zapravo, prva stvar koju će računalo provjeriti jest može li pobijediti u trenutnom potezu i ako može igra će se završiti.

U programu se može naći i kod koji kontrolira izgled i mogućnost pritiska na gumb. Također, svaki gumb ima svoj rukovatelj događaja. U slučaju gumbova koji predstavljaju polja, rukovatelj događaja postavlja polje na korisnikov znak, onemogućuje daljnji unos u to polje i daje potez računalu. U slučaju gumba za početak računalo bira početni način pobjede i odrađuje svoj potez.

5.2. Rad s programom

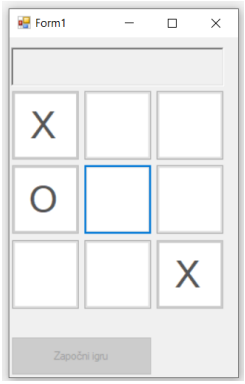
Kada korisnik uključi program pojavi se forma na kojoj se nalazi deset gumba i jedan textbox. Devet gumba služe kao polja za upis „križića“ i „kružića“, deseti gumb služi za pokretanje programa. Textbox ispisuje rezultat na kraju igre.



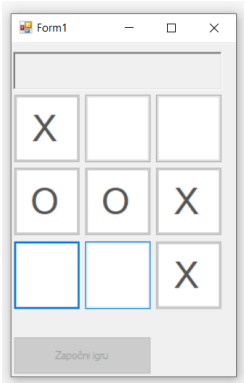
Slika 5: Početni izgled forme

Slika 6: Izgled forme nakon početka igranja

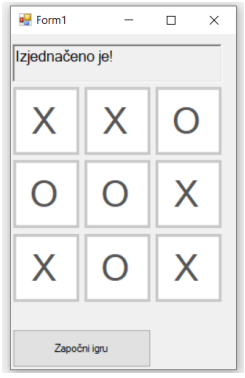
Jedini gumb koji se može pritisnuti pri pokretanju igre je gumb „Započni igru“. Pritiskom na taj gumb računalo će odigrati svoj prvi potez te će omogućiti pritisak na ostale gumbe. Također, gumb „Započni igru“ će tada biti onemogućen.Kao što se vidi na slici 7 računalo će odigravati svoje poteze na način da postigne neku od mogućih pobjeda. Na slici 8 se može vidjeti da računalo onemogućuje pobjedu korisnika nakon što mu preostane jedan potez do pobjede.



Slika 7: Računalo pokušava ostvariti pobjedu



Slika 8: Računalo ne dozvoljava pobjedu korisnika



Slika 9: Izgled programa prilikom završetka igre

Korisnik i računalo će odigravati poteze dok jedno od njih ne pobijedi ili dok ne ponestane poteza. U slučaju da pobijedi računalo u textboxu će se upisati prikladna poruka i pobjednički niz će biti označen crvenom bojom. U slučaju da pobijedi korisnik niz će biti obojan u zelenu boju. Na slici 9 se vidi kako izgleda forma nakon što je rezultat izjednačen. U svakom slučaju, nakon što igra završi gumbovi za unos „križića“ i „kružića“ bit će onemogućeni, a gumb za pokretanje igre omogućen.

6. Zaključak

Inteligentni agenti su predmet istraživanja već duže vremena. Iako su neki aspekti inteligentnih agenata već poznati, poput autonomnosti, sposobnosti učenja i sličnih, mnogi od njih i dalje nisu u potpunosti istraženi. Neki od njih, poput učenja, i dalje su vrlo kompleksni i teški za implementaciju te je potrebno još rada kako bi se s njima mogli postići bolji rezultati na lakši način.

U mnogim slučajevima inteligentni agenti se pojavljuju zajedno u obliku višeagentnih sustava. Višeagentni sustavi su bitni zato što omogućuju korištenje više jednostavnijih agenata u suprotnosti s jednim kompleksnim. Pošto je izrađivanje jednostavnih agenata lakše od izrađivanja kompleksnih višeagentni sustavi mogu pronaći brojne primjene u stvarnom svijetu.

Modeliranje temeljeno na agentima omogućuje korištenje višeagentnih sustava za izrađivanje simulacija i analiziranje podataka dobivenih iz modela. Kompleksni sustavi koji se ne mogu objasniti jednadžbama i funkcijama mogu se modelirati pa se zato modeliranje koristi za analizu mnogobrojnih sustava koje bi bilo vrlo teško analizirati drukčije. Naravno, modeliranje ima svoje granice i ne može raditi s previše kompleksnim sustavima, ali takvi sustavi su vrlo rijetki u današnje vrijeme i obično su računala dovoljno snažna za izradu modela.

Svi navedeni koncepti su vrlo važni te se danas koriste u mnogim poljima ljudske djelatnosti. Neke aplikacije i koncepti inteligentnih agenata nisu do kraja istraženi, primjerice učenje, a iz njih proizlaze pozitivne posljedice. Stoga se može zaključiti da u njihovo istraživanje vrijedi ulagati.

Popis literature

[1] S. Russell i P. Norvig, *Artificial Intelligence: A Modern Approach (2nd Edition)*. 1995, ISBN: 0131038052. DOI: 10.1017/S0269888900007724. arXiv: arXiv:1011.1669v3.

[2] F. Mills i R. Stufflebeam, *Introduction to Intelligent Agents - The Mind Project*, 2005. adresa: http://www.mind.ilstu.edu/curriculum/ants%7B%5C_%7Dnasa/intelligent%7B%5C_%7Dagents.php (pogledano 28. 7. 2020).

[3] C. G. Jung i K. Fischer, „Logic-Based Hybrid Agents", *Computational Logic: Logic Programming and Beyond: Essays in Honour of Robert A. Kowalski Part I*, A. C. Kakas i F. Sadri, ur. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, str. 626–654, ISBN: 978-3-540-45628-5. DOI: 10.1007/3-540-45628-7_23. adresa: https://doi.org/10.1007/3-540-45628-7%7B%5C_%7D23.

[4] K. Rabuzin, M. Maleković i M. Bača, „1 . AGENT DEFINITIONS", sv. 30, str. 155–170, 2006.

[5] A. Moreno, *Agent properties*, 2010. adresa: https://www.slideshare.net/ToniMorenoURV/agent-properties (pogledano 1. 8. 2020).

[6] P. Koehn, „Intelligent agents", 2020, ISSN: 15577317. DOI: 10.1145/176789.176790.

[7] J. Chang, T. Healy i J. Wood, „The Potential Regulatory Challenges of Increasingly Autonomous Motor Vehicles", *Santa Clara Law Review*, 2012, ISSN: 0146-0315. DOI: 10.1525/sp.2007.54.1.23.. arXiv: arXiv:1011.1669v3.

[8] National Center for Statistics and Analysis, „2016 fatal motor vehicle crashes: Overview", *Traffic Safety Facts*, br. October, str. 1–9, 2017, ISSN: 0970-1591.

[9] S. Wang i Z. Li, „Exploring the mechanism of crashes with automated vehicles using statistical modeling approaches", eng, *PloS one*, sv. 14, br. 3, e0214550–e0214550, ožujak 2019, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0214550. adresa: https://pubmed.ncbi.nlm.nih.gov/30921396%20https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6438496/.

[10] I. Imam i L. Kerschberg, „Adaptive intelligent agents", *Journal of Intelligent Information Systems*, sv. 9, br. 3, str. 211–213, 1997, ISSN: 09259902. DOI: 10.1023/A:1008672326807.

[11] I. F. Imam i Y. Kodratoff, „Intelligent adaptive agents. A highlight of the field and the AAAI-96 workshop", *AI Magazine*, 1997, ISSN: 07384602.

[12] L. Hardesty, *Explained: Neural networks | MIT News | Massachusetts Institute of Technology*, 2017. adresa: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414 (pogledano 25. 8. 2020).

[13] *Autopilot AI | Tesla*, 2020. adresa: https://www.tesla.com/autopilotAI (pogledano 25. 8. 2020).

[14] A. Lingnau, O. Drobnik i P. Dömel, „An HTTP-based infrastructure for mobile agents", *Fourth International World Wide Web . . .*, 1995.

[15] F. Xiong i K. Qiao, „Intelligent system and its application in agriculture", str. 5597–5602, 1999.

[16] H. Van Dyke Parunak, „Manufacturing Experience with the Contract Net", *Distributed Artificial Intelligence*, 1987. DOI: 10.1016/b978-0-934613-38-5.50013-1.

[17] K. P. Sycara, „Multiagent Systems", sv. 19, br. 2, str. 79–92, 1998.

[18] M. Navarro, J. Corchado Rodríguez i Y. Demazeau, „A Musical Composition Application Based on a Multiagent System to Assist Novel Composers", 2014.

[19] A. Dorri, S. Kanhere i R. Jurdak, „Multi-Agent Systems: A survey", *IEEE Access*, str. 1, 2018. DOI: 10.1109/ACCESS.2018.2831228.

[20] H. Rezaee i F. Abdollahi, „Average consensus over high-order multiagent systems", *IEEE Transactions on Automatic Control*, 2015, ISSN: 00189286. DOI: 10.1109/TAC.2015.2408576.

[21] H. F. Ahmad, „Multi-agent systems: Overview of a new paradigm for distributed systems", *Proceedings of IEEE International Symposium on High Assurance Systems Engineering*, 2002, ISBN: 0769517692. DOI: 10.1109/HASE.2002.1173110.

[22] Q. Song, F. Liu, H. Su i A. V. Vasilakos, „Semi-global and global containment control of multi-agent systems with second-order dynamics and input saturation", *International Journal of Robust and Nonlinear Control*, 2016, ISSN: 10991239. DOI: 10.1002/rnc.3515.

[23] S. E. Lander, V. R. Lesser i M. E. Connell, „Knowledge-based conflict resolution for cooperation among expert agents", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1991, ISBN: 9783540540083. DOI: 10.1007/BFb0014282.

[24] J. S. Liu i K. P. Sycara, „Coordination of multiple agents for production management", *Annals of Operations Research*, 1997, ISSN: 15729338. DOI: 10.1023/a:1018911613698.

[25] R. Davis i R. G. Smith, „Negotiation as a metaphor for distributed problem solving", *Artificial Intelligence*, 1983, ISSN: 00043702. DOI: 10.1016/0004-3702(83)90015-2.

[26] T. Mullen i M. E. Wellman, „Some issues in the design of market-oriented agents", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1996, ISBN: 3540608052. DOI: 10.1007/3540608052_73.

[27] T. Sandholm, „Implementation of the contract net protocol based on marginal cost calculations”, *Proceedings of the National Conference on Artificial Intelligence*, 1993, ISBN: 0262510715.

[28] V. R. Lesser, „A retrospective view of FA/C distributed problem solving”, *IEEE Transactions on Systems, Man and Cybernetics*, 1991, ISSN: 21682909. DOI: 10.1109/21.135681.

[29] W. V. D. Hoek, G. A. Kaminka i J. Werfel, „Collective Decision-Making in Multi- Agent Systems by Implicit Leadership Terms of Use”, 2010.

[30] H. Yang, B. Jiang, V. Cocquempot i H. Zhang, „Stabilization of Switched Nonlinear Systems With All Unstable Modes: Application to Multi-Agent Systems”, *IEEE Transactions on Automatic Control*, 2011, ISSN: 0018-9286. DOI: 10.1109/tac.2011.2157413.

[31] B. A. Huberman i T. Hogg, „The behavior of computational ecologies”, *The Ecology of Computation*, North-Holland, 1988, str. 77–115.

[32] J. D. Thomas i K. Sycara, „Heterogeneity, stability, and efficiency in distributed systems”, *Proceedings - International Conference on Multi Agent Systems, ICMAS 1998*, 1998, ISBN: 081868500X. DOI: 10.1109/ICMAS.1998.699069.

[33] S. Adiloğlu, S. Fajardo, García-Galvan, F. R., V. Barranco, J. C. Galvan i S. F. Batlle, „We are IntechOpen , the world ’ s leading publisher of Open Access books Built by scientists , for scientists TOP 1 %”, *Intech*, sv. i, br. tourism, str. 13, 2016. DOI: http://dx.doi.org/10.5772/57353. adresa: https://www.intechopen.com/books/advanced-biometric-technologies/liveness-detection-in-biometrics.

[34] S. Kraus, J. Wilkenfeld i G. Zlotkin, „Multiagent negotiation under time constraints”, *Artificial Intelligence*, 1995, ISSN: 00043702. DOI: 10.1016/0004-3702(94)00021-R.

[35] L. Garrido i K. Sycara, „Multi-Agent Meeting Scheduling: Preliminary Experimental Results”, *Proceedings of the Second International Conference on Multiagent Systems*, 1996.

[36] J. Liu i Sycara, „Distributed Problem Solving through Coordination in a Society of Agents”, *Proceedings of the 13th International Workshop on Distributed Artificial Intelligence*, 1994.

[37] W. Zhang, Y. Liu, J. Lu i J. Cao, „A novel consensus algorithm for second-order multi-agent systems without velocity measurements”, *International Journal of Robust and Nonlinear Control*, 2017, ISSN: 10991239. DOI: 10.1002/rnc.3694.

[38] W. Ren, K. Moore i Y. Q. Chen, „High-order consensus algorithms in cooperative vehicle systems”, *Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control, ICNSC’06*, 2006, ISBN: 1424400651. DOI: 10.1109/icnsc.2006.1673189.

[39] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg i I. Brandic, „Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility”, *Future Generation Computer Systems*, 2009, ISSN: 0167739X. DOI: 10.1016/j.future.2008.12.001.

[40] K. M. Sim, „Kent Academic Repository Agent-Based Cloud Computing”, sv. 5, str. 564–577, 2012.

[41] J. Bajo, F. De la Prieta, J. M. Corchado i S. Rodríguez, „A low-level resource allocation in an agent-based Cloud Computing platform”, *Applied Soft Computing Journal*, 2016, ISSN: 15684946. DOI: 10.1016/j.asoc.2016.05.056.

[42] J. Fiosina i M. Fiosins, „Density-based clustering in cloud-oriented collaborative multi-agent systems”, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, ISBN: 9783642408458. DOI: 10.1007/978-3-642-40846-5_64.

[43] L. Mechtri, F. D. Tolba i S. Ghanemi, „MASID: Multi-agent system for intrusion detection in MANET”, *Proceedings of the 9th International Conference on Information Technology, ITNG 2012*, 2012, ISBN: 9780769546544. DOI: 10.1109/ITNG.2012.18.

[44] A. Dorri, „An EDRI-based approach for detecting and eliminating cooperative black hole nodes in MANET”, *Wireless Networks*, 2017, ISSN: 15728196. DOI: 10.1007/s11276-016-1251-x.

[45] C. G. Cena, P. F. Cardenas, R. S. Pazmino, L. Puglisi i R. A. Santonja, „A cooperative multi-agent robotics system:Design and modelling”, *Expert Systems with Applications*, 2013, ISSN: 09574174. DOI: 10.1016/j.eswa.2013.01.048.

[46] H. Wang i C. Wang, „Intelligent agents in the nuclear industry”, *Computer*, 1997, ISSN: 00189162. DOI: 10.1109/2.634838.

[47] K. Sycara, A. Pannu, M. Williamson, D. Zeng i K. Decker, „Distributed intelligent agents”, *IEEE Expert-Intelligent Systems and their Applications*, 1996, ISSN: 08859000. DOI: 10.1109/64.546581.

[48] F. Klügl i A. Bazzan, „Agent-Based Modeling and Simulation”, *AI Magazine*, sv. 33, str. 29–40, 2012. DOI: 10.1609/aimag.v33i3.2425.

[49] A. B. Shiflet i G. W. Shiflet, „An Introduction to Agent-based Modeling for Undergraduates”, *Procedia Computer Science*, sv. 29, str. 1392–1402, 2014, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2014.05.126. adresa: http://www.sciencedirect.com/science/article/pii/S1877050914003032.

[50] E. Bonabeau, „Agent-based modeling: Methods and techniques for simulating human systems”, *Proceedings of the National Academy of Sciences*, sv. 99, br. suppl 3, str. 7280–7287, 2002, ISSN: 0027-8424. DOI: 10.1073/pnas.082080899. adresa: https://www.pnas.org/content/99/suppl%7B%5C_%7D3/7280.

[51] J. L. Casti, „Would-be worlds: how simulation is changing the frontiers of science”, *Choice Reviews Online*, 1997, ISSN: 0009-4978. DOI: 10.5860/choice.34-3354.

Popis slika

1. Slikoviti prikaz jednostavnih refleksnih agenata prema [1] 10
2. Slikoviti prikaz agenata s bilježenjem promjena u okolini prema [1] 11
3. Slikoviti prikaz agenata temeljenih na cilju prema [1] 12
4. Slikoviti prikaz agenata temeljenih na korisnosti prema [1] 13
5. Početni izgled forme 31
6. Izgled forme nakon početka igranja 31
7. Računalo pokušava ostvariti pobjedu 32
8. Računalo ne dozvoljava pobjedu korisnika 32
9. Izgled programa prilikom završetka igre 32

1. Prilog 1

U nastavku slijedi programski kod za program.

```
biranjeNiza biranje = new biranjeNiza();
int brojNiza;
bool odigrano = false;

private void button10_Click(object sender, EventArgs e)
{
    omoguciPoteze();
    brojNiza = biranje.biranjePoteza();
    OdigravanjePoteza();
}

private void btn1_Click(object sender, EventArgs e)
{
    btn1.Text = "O";
    btn1.Enabled = false;
    OdigravanjePoteza();
}

private void btn2_Click(object sender, EventArgs e)
{
    btn2.Text = "O";
    btn2.Enabled = false;
    OdigravanjePoteza();
}

private void btn3_Click(object sender, EventArgs e)
{
    btn3.Text = "O";
    btn3.Enabled = false;
    OdigravanjePoteza();
}

private void btn4_Click(object sender, EventArgs e)
{
    btn4.Text = "O";
    btn4.Enabled = false;
    OdigravanjePoteza();
}

private void btn5_Click(object sender, EventArgs e)
{
    btn5.Text = "O";
    btn5.Enabled = false;
    OdigravanjePoteza();
}

private void btn6_Click(object sender, EventArgs e)
{
```

```
    btn6.Text = "O";
    btn6.Enabled = false;
    OdigravanjePoteza();
}

private void btn7_Click(object sender, EventArgs e)
{
    btn7.Text = "O";
    btn7.Enabled = false;
    OdigravanjePoteza();
}

private void btn8_Click(object sender, EventArgs e)
{
    btn8.Text = "O";
    btn8.Enabled = false;
    OdigravanjePoteza();
}

private void btn9_Click(object sender, EventArgs e)
{
    btn9.Text = "O";
    btn9.Enabled = false;
    OdigravanjePoteza();
}

//Funkcija u kojoj se nalaze stvari koje racunalo treba napravit pri igranju
private void OdigravanjePoteza()
{
    Pobjeda();
    napad();
    obrana();
    while (odigrano == false)
    {
        if (brojNiza == (int)enumPobjeda.prviVodoravno)
        {
            prviVodoravno();
        }
        if (brojNiza == (int)enumPobjeda.drugiVodoravno)
        {
            drugiVodoravno();
        }
        if (brojNiza == (int)enumPobjeda.treciVodoravno)
        {
            treciVodoravno();
        }
        if (brojNiza == (int)enumPobjeda.prviOkomito)
        {
            prviOkomito();
        }
        if (brojNiza == (int)enumPobjeda.drugiOkomito)
        {
            drugiOkomito();
```

```
        }
        if (brojNiza == (int)enumPobjeda.treciOkomito)
        {
            treciOkomito();
        }
        if (brojNiza == (int)enumPobjeda.goreLijevo)
        {
            goreLijevo();
        }
        if (brojNiza == (int)enumPobjeda.goreDesno)
        {
            goreDesno();
        }
    }
    odigrano = false;
    Pobjeda();
}

//Provjera je li se dogodio zavrsetak igre i poduzimanje odgovarajucih mjera
private void Pobjeda()
{
    if (btn1.Text != "␣" && btn2.Text != "␣" && btn3.Text != "␣" &&
        btn4.Text != "␣" && btn5.Text != "␣" && btn6.Text != "␣" &&
        btn7.Text != "␣" && btn8.Text != "␣" && btn9.Text != "␣")
    {
        txtPobjeda.Text = "Izjednaceno␣je!";
        disableButton();
    }
    if (btn1.Text == "X" && btn2.Text == "X" && btn3.Text == "X")
    {
        txtPobjeda.Text = "Pobijedilo␣je␣racunalo!";
        btn1.BackColor = Color.Red;
        btn2.BackColor = Color.Red;
        btn3.BackColor = Color.Red;
        disableButton();
    }
    if (btn1.Text == "X" && btn2.Text == "X" && btn3.Text == "X")
    {
        txtPobjeda.Text = "Pobijedilo␣je␣racunalo!";
        btn1.BackColor = Color.Red;
        btn2.BackColor = Color.Red;
        btn3.BackColor = Color.Red;
        disableButton();
    }
    if (btn4.Text == "X" && btn5.Text == "X" && btn6.Text == "X")
    {
        txtPobjeda.Text = "Pobijedilo␣je␣racunalo!";
        btn4.BackColor = Color.Red;
        btn5.BackColor = Color.Red;
        btn6.BackColor = Color.Red;
        disableButton();
    }
    if (btn7.Text == "X" && btn8.Text == "X" && btn9.Text == "X")
```

```
{
    txtPobjeda.Text = "Pobijedilo je racunalo!";
    btn7.BackColor = Color.Red;
    btn8.BackColor = Color.Red;
    btn9.BackColor = Color.Red;
    disableButton();
}
if (btn1.Text == "X" && btn4.Text == "X" && btn7.Text == "X")
{
    txtPobjeda.Text = "Pobijedilo je racunalo!";
    btn1.BackColor = Color.Red;
    btn4.BackColor = Color.Red;
    btn7.BackColor = Color.Red;
    disableButton();
}
if (btn2.Text == "X" && btn5.Text == "X" && btn8.Text == "X")
{
    txtPobjeda.Text = "Pobijedilo je racunalo!";
    btn2.BackColor = Color.Red;
    btn5.BackColor = Color.Red;
    btn8.BackColor = Color.Red;
    disableButton();
}
if (btn3.Text == "X" && btn6.Text == "X" && btn9.Text == "X")
{
    txtPobjeda.Text = "Pobijedilo je racunalo!";
    btn3.BackColor = Color.Red;
    btn6.BackColor = Color.Red;
    btn9.BackColor = Color.Red;
    disableButton();
}
if (btn1.Text == "X" && btn5.Text == "X" && btn9.Text == "X")
{
    txtPobjeda.Text = "Pobijedilo je racunalo!";
    btn1.BackColor = Color.Red;
    btn5.BackColor = Color.Red;
    btn9.BackColor = Color.Red;
    disableButton();
}
if (btn3.Text == "X" && btn5.Text == "X" && btn7.Text == "X")
{
    txtPobjeda.Text = "Pobijedilo je racunalo!";
    btn3.BackColor = Color.Red;
    btn5.BackColor = Color.Red;
    btn7.BackColor = Color.Red;
    disableButton();
}
if (btn1.Text == "O" && btn2.Text == "O" && btn3.Text == "O")
{
    txtPobjeda.Text = "Pobijedili ste!";
    btn1.BackColor = Color.Green;
    btn2.BackColor = Color.Green;
    btn3.BackColor = Color.Green;
```

```
    disableButton();
}
if (btn4.Text == "O" && btn5.Text == "O" && btn6.Text == "O")
{
    txtPobjeda.Text = "Pobijedili␣ste!";
    btn4.BackColor = Color.Green;
    btn5.BackColor = Color.Green;
    btn6.BackColor = Color.Green;
    disableButton();
}
if (btn7.Text == "O" && btn8.Text == "O" && btn9.Text == "O")
{
    txtPobjeda.Text = "Pobijedili␣ste!";
    btn7.BackColor = Color.Green;
    btn8.BackColor = Color.Green;
    btn9.BackColor = Color.Green;
    disableButton();
}
if (btn1.Text == "O" && btn4.Text == "O" && btn7.Text == "O")
{
    txtPobjeda.Text = "Pobijedili␣ste!";
    btn1.BackColor = Color.Green;
    btn4.BackColor = Color.Green;
    btn7.BackColor = Color.Green;
    disableButton();
}
if (btn2.Text == "O" && btn5.Text == "O" && btn8.Text == "O")
{
    txtPobjeda.Text = "Pobijedili␣ste!";
    btn2.BackColor = Color.Green;
    btn5.BackColor = Color.Green;
    btn8.BackColor = Color.Green;
    disableButton();
}
if (btn3.Text == "O" && btn6.Text == "O" && btn9.Text == "O")
{
    txtPobjeda.Text = "Pobijedili␣ste!";
    btn3.BackColor = Color.Green;
    btn6.BackColor = Color.Green;
    btn9.BackColor = Color.Green;
    disableButton();
}
if (btn1.Text == "O" && btn5.Text == "O" && btn9.Text == "O")
{
    txtPobjeda.Text = "Pobijedili␣ste!";
    btn1.BackColor = Color.Green;
    btn5.BackColor = Color.Green;
    btn9.BackColor = Color.Green;
    disableButton();
}
if (btn3.Text == "O" && btn5.Text == "O" && btn7.Text == "O")
{
    txtPobjeda.Text = "Pobijedili␣ste!";
```

```
            btn3.BackColor = Color.Green;
            btn5.BackColor = Color.Green;
            btn7.BackColor = Color.Green;
            disableButton();
        }
    }

    //Funkcija za postavljanje gumbova prilikom zavrsetka igre
    private void disableButton()
    {
        btn1.Enabled = false;
        btn2.Enabled = false;
        btn3.Enabled = false;
        btn4.Enabled = false;
        btn5.Enabled = false;
        btn6.Enabled = false;
        btn7.Enabled = false;
        btn8.Enabled = false;
        btn9.Enabled = false;
        btnStart.Enabled = true;
    }

    //Funkcija za pozivanje ostalih funkcija za provjeravanje provjeravanje moguce
        pobjede korisnika
    private bool obrana()
    {
        if (prviVodoravnoObrana())
        {
            return true;
        }
        if (drugiVodoravnoObrana())
        {
            return true;
        }
        if (treciVodoravnoObrana())
        {
            return true;
        }
        if (prviOkomitoObrana())
        {
            return true;
        }
        if (drugiOkomitoObrana())
        {
            return true;
        }
        if (treciOkomitoObrana())
        {
            return true;
        }
        if (goreLijevoObrana())
        {
            return true;
```

```
    }
    if (goreDesnoObrana())
    {
        return true;
    }
    return false;
}

//Funkcija za pozivanje ostalih funkcija za provjeravanje provjeravanje moguce
   pobjede racunala
private bool napad()
{
    if (prviVodoravnoNapad())
    {
        return true;
    }
    if (drugiVodoravnoNapad())
    {
        return true;
    }
    if (treciVodoravnoNapad())
    {
        return true;
    }
    if (prviOkomitoNapad())
    {
    return true;
    }
    if (drugiOkomitoNapad())
    {
        return true;
    }
    if (treciOkomitoNapad())
    {
        return true;
    }
    if (goreLijevoNapad())
    {
        return true;
    }
    if (goreDesnoNapad())
    {
        return true;
    }
    return false;
}

//Funkcija za provjeravanje moze li korisnik pobijediti s tim nacinom i ako moze
   sprijeciti ga
private bool prviVodoravnoObrana()
{
    if(btn1.Text == "O" && btn2.Text == "O" && btn3.Text == "␣")
    {
```

```
            btn3.Text = "X";
            btn3.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn1.Text == "O" && btn3.Text == "O" && btn2.Text == "␣")
        {
            btn2.Text = "X";
            btn2.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn3.Text == "O" && btn2.Text == "O" && btn1.Text == "␣")
        {
            btn1.Text = "X";
            btn1.Enabled = false;
            odigrano = true;
            return true;
        }
        return false;
    }

    private bool drugiVodoravnoObrana()
    {
        if (btn4.Text == "O" && btn5.Text == "O" && btn6.Text == "␣")
        {
            btn6.Text = "X";
            btn6.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn4.Text == "O" && btn6.Text == "O" && btn5.Text == "␣")
        {
            btn5.Text = "X";
            btn5.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn6.Text == "O" && btn5.Text == "O" && btn4.Text == "␣")
        {
            btn4.Text = "X";
            btn4.Enabled = false;
            odigrano = true;
            return true;
        }
        return false;
    }

    private bool treciVodoravnoObrana()
    {
        if (btn7.Text == "O" && btn8.Text == "O" && btn9.Text == "␣")
        {
            btn9.Text = "X";
```

```
            btn9.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn7.Text == "O" && btn9.Text == "O" && btn8.Text == "␣")
        {
            btn8.Text = "X";
            btn8.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn9.Text == "O" && btn8.Text == "O" && btn7.Text == "␣")
        {
            btn7.Text = "X";
            btn7.Enabled = false;
            odigrano = true;
            return true;
        }
        return false;
    }

    private bool prviOkomitoObrana()
    {
        if (btn1.Text == "O" && btn4.Text == "O" && btn7.Text == "␣")
        {
            btn7.Text = "X";
            btn7.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn7.Text == "O" && btn4.Text == "O" && btn1.Text == "␣")
        {
            btn1.Text = "X";
            btn1.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn1.Text == "O" && btn7.Text == "O" && btn4.Text == "␣")
        {
            btn4.Text = "X";
            btn4.Enabled = false;
            odigrano = true;
            return true;
        }
        return false;
    }

    private bool drugiOkomitoObrana()
    {
        if (btn2.Text == "O" && btn5.Text == "O" && btn8.Text == "␣")
        {
            btn8.Text = "X";
            btn8.Enabled = false;
```

```
            odigrano = true;
            return true;
        }
        if (btn8.Text == "O" && btn5.Text == "O" && btn2.Text == "␣")
        {
            btn2.Text = "X";
            btn2.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn2.Text == "O" && btn8.Text == "O" && btn5.Text == "␣")
        {
            btn5.Text = "X";
            btn5.Enabled = false;
            odigrano = true;
            return true;
        }
        return false;
    }

    private bool treciOkomitoObrana()
    {
        if (btn3.Text == "O" && btn6.Text == "O" && btn9.Text == "␣")
        {
            btn9.Text = "X";
            btn9.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn9.Text == "O" && btn6.Text == "O" && btn3.Text == "␣")
        {
            btn3.Text = "X";
            btn3.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn3.Text == "O" && btn9.Text == "O" && btn6.Text == "␣")
        {
            btn6.Text = "X";
            btn6.Enabled = false;
            odigrano = true;
            return true;
        }
        return false;
    }

    private bool goreLijevoObrana()
    {
        if (btn1.Text == "O" && btn5.Text == "O" && btn9.Text == "␣")
        {
            btn9.Text = "X";
            btn9.Enabled = false;
            odigrano = true;
```

```
            return true;
        }
        if (btn9.Text == "O" && btn5.Text == "O" && btn1.Text == "␣")
        {
            btn1.Text = "X";
            btn1.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn1.Text == "O" && btn9.Text == "O" && btn5.Text == "␣")
        {
            btn5.Text = "X";
            btn5.Enabled = false;
            odigrano = true;
            return true;
        }
        return false;
    }

    private bool goreDesnoObrana()
    {
        if (btn3.Text == "O" && btn5.Text == "O" && btn7.Text == "␣")
        {
            btn7.Text = "X";
            btn7.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn7.Text == "O" && btn5.Text == "O" && btn3.Text == "␣")
        {
            btn3.Text = "X";
            btn3.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn3.Text == "O" && btn7.Text == "O" && btn5.Text == "␣")
        {
            btn5.Text = "X";
            btn5.Enabled = false;
            odigrano = true;
            return true;
        }
        return false;
    }

    //Funkcija koja provjerava moze li čraunalo pobijediti u iducem potezu

    private bool prviVodoravnoObrana()
    {
        if(btn1.Text == "X" && btn2.Text == "X" && btn3.Text == "␣")
        {
            btn3.Text = "X";
            btn3.Enabled = false;
```

```
            odigrano = true;
            return true;
        }
        if (btn1.Text == "X" && btn3.Text == "X" && btn2.Text == "␣")
        {
            btn2.Text = "X";
            btn2.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn3.Text == "X" && btn2.Text == "X" && btn1.Text == "␣")
        {
            btn1.Text = "X";
            btn1.Enabled = false;
            odigrano = true;
            return true;
        }
        return false;
    }

    private bool drugiVodoravnoObrana()
    {
        if (btn4.Text == "X" && btn5.Text == "X" && btn6.Text == "␣")
        {
            btn6.Text = "X";
            btn6.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn4.Text == "X" && btn6.Text == "X" && btn5.Text == "␣")
        {
            btn5.Text = "X";
            btn5.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn6.Text == "X" && btn5.Text == "X" && btn4.Text == "␣")
        {
            btn4.Text = "X";
            btn4.Enabled = false;
            odigrano = true;
            return true;
        }
        return false;
    }

    private bool treciVodoravnoObrana()
    {
        if (btn7.Text == "X" && btn8.Text == "X" && btn9.Text == "␣")
        {
            btn9.Text = "X";
            btn9.Enabled = false;
            odigrano = true;
```

```
            return true;
        }
        if (btn7.Text == "X" && btn9.Text == "X" && btn8.Text == "␣")
        {
            btn8.Text = "X";
            btn8.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn9.Text == "X" && btn8.Text == "X" && btn7.Text == "␣")
        {
            btn7.Text = "X";
            btn7.Enabled = false;
            odigrano = true;
            return true;
        }
        return false;
    }

    private bool prviOkomitoObrana()
    {
        if (btn1.Text == "X" && btn4.Text == "X" && btn7.Text == "␣")
        {
            btn7.Text = "X";
            btn7.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn7.Text == "X" && btn4.Text == "X" && btn1.Text == "␣")
        {
            btn1.Text = "X";
            btn1.Enabled = false;
            odigrano = true;
            return true;
        }
        if (btn1.Text == "X" && btn7.Text == "X" && btn4.Text == "␣")
        {
            btn4.Text = "X";
            btn4.Enabled = false;
            odigrano = true;
            return true;
        }
        return false;
    }

    private bool drugiOkomitoObrana()
    {
        if (btn2.Text == "X" && btn5.Text == "X" && btn8.Text == "␣")
        {
            btn8.Text = "X";
            btn8.Enabled = false;
            odigrano = true;
            return true;
```

```
    }
    if (btn8.Text == "X" && btn5.Text == "X" && btn2.Text == "␣")
    {
        btn2.Text = "X";
        btn2.Enabled = false;
        odigrano = true;
        return true;
    }
    if (btn2.Text == "X" && btn8.Text == "X" && btn5.Text == "␣")
    {
        btn5.Text = "X";
        btn5.Enabled = false;
        odigrano = true;
        return true;
    }
    return false;
}

private bool treciOkomitoObrana()
{
    if (btn3.Text == "X" && btn6.Text == "X" && btn9.Text == "␣")
    {
        btn9.Text = "X";
        btn9.Enabled = false;
        odigrano = true;
        return true;
    }
    if (btn9.Text == "X" && btn6.Text == "X" && btn3.Text == "␣")
    {
        btn3.Text = "X";
        btn3.Enabled = false;
        odigrano = true;
        return true;
    }
    if (btn3.Text == "X" && btn9.Text == "X" && btn6.Text == "␣")
    {
        btn6.Text = "X";
        btn6.Enabled = false;
        odigrano = true;
        return true;
    }
    return false;
}

private bool goreLijevoObrana()
{
    if (btn1.Text == "X" && btn5.Text == "X" && btn9.Text == "␣")
    {
        btn9.Text = "X";
        btn9.Enabled = false;
        odigrano = true;
        return true;
    }
```

```
    if (btn9.Text == "X" && btn5.Text == "X" && btn1.Text == "␣")
    {
        btn1.Text = "X";
        btn1.Enabled = false;
        odigrano = true;
        return true;
    }
    if (btn1.Text == "X" && btn9.Text == "X" && btn5.Text == "␣")
    {
        btn5.Text = "X";
        btn5.Enabled = false;
        odigrano = true;
        return true;
    }
    return false;
}

private bool goreDesnoObrana()
{
    if (btn3.Text == "X" && btn5.Text == "X" && btn7.Text == "␣")
    {
        btn7.Text = "X";
        btn7.Enabled = false;
        odigrano = true;
        return true;
    }
    if (btn7.Text == "X" && btn5.Text == "X" && btn3.Text == "␣")
    {
        btn3.Text = "X";
        btn3.Enabled = false;
        odigrano = true;
        return true;
    }
    if (btn3.Text == "X" && btn7.Text == "X" && btn5.Text == "␣")
    {
        btn5.Text = "X";
        btn5.Enabled = false;
        odigrano = true;
        return true;
    }
    return false;
}

private void prviVodoravno()
{
    //provjeri je li moguce pobijediti i ako nije odigraj bilo koji potez
    if (!provjeriDostupnost())
    {
        odigrajPotez();
        return;
    }
    //provjera je li protivnik onemogucio pobijedivanje ovim nacinom i biranje
        drugog nacina ako je
```

```
    if (btn1.Text == "O")
    {
        brojNiza = biranje.biranjePoteza();
        return;
    }
    if (btn2.Text == "O")
    {
        brojNiza = biranje.biranjePoteza();
        return;
    }
    if (btn3.Text == "O")
    {
        brojNiza = biranje.biranjePoteza();
        return;
    }
    //odabiranje moguceg polja i upisivanje iksica u njega
    int brojGumba = biranje.odrediGumb();
    switch (brojGumba)
    {
        case 0:
            if (btn1.Text == "␣")
            {
                btn1.Text = "X";
                btn1.Enabled = false;
                odigrano = true;
            }
            break;
        case 1:
            if(btn2.Text == "␣")
            {
                btn2.Text = "X";
                btn2.Enabled = false;
                odigrano = true;
            }
            break;
        case 2:
            if (btn3.Text == "␣")
            {
                btn3.Text = "X";
                btn3.Enabled = false;
                odigrano = true;
            }
            break;
    }
}

private void drugiVodoravno()
{
    if (!provjeriDostupnost())
    {
        odigrajPotez();
        return;
    }
```

```
    if (btn4.Text == "O")
    {
        brojNiza = biranje.biranjePoteza();
        return;
    }
    if (btn5.Text == "O")
    {
        brojNiza = biranje.biranjePoteza();
        return;
    }
    if (btn6.Text == "O")
    {
        brojNiza = biranje.biranjePoteza();
        return;
    }
    int brojGumba = biranje.odrediGumb();
    switch (brojGumba)
    {
        case 0:
            if (btn4.Text == "␣")
            {
                btn4.Text = "X";
                btn4.Enabled = false;
                odigrano = true;
            }
            break;
        case 1:
            if (btn5.Text == "␣")
            {
                btn5.Text = "X";
                btn5.Enabled = false;
                odigrano = true;
            }
            break;
        case 2:
            if (btn6.Text == "␣")
            {
                btn6.Text = "X";
                btn6.Enabled = false;
                odigrano = true;
            }
            break;
    }
}

private void treciVodoravno()
{
    if (!provjeriDostupnost())
    {
        odigrajPotez();
        return;
    }
    if (btn7.Text == "O")
```

```
    {
        brojNiza = biranje.biranjePoteza();
        return;
    }
    if (btn8.Text == "O")
    {
        brojNiza = biranje.biranjePoteza();
        return;
    }
    if (btn9.Text == "O")
    {
        brojNiza = biranje.biranjePoteza();
        return;
    }
    int brojGumba = biranje.odrediGumb();
    switch (brojGumba)
    {
        case 0:
            if (btn7.Text == "␣")
            {
                btn7.Text = "X";
                btn7.Enabled = false;
                odigrano = true;
            }
            break;
        case 1:
            if (btn8.Text == "␣")
            {
                btn8.Text = "X";
                btn8.Enabled = false;
                odigrano = true;
            }
            break;
        case 2:
            if (btn9.Text == "␣")
            {
                btn9.Text = "X";
                btn9.Enabled = false;
                odigrano = true;
            }
            break;
    }
}

private void prviOkomito()
{
    if (!provjeriDostupnost())
    {
        odigrajPotez();
        return;
    }
    if (btn1.Text == "O")
    {
```

```
            brojNiza = biranje.biranjePoteza();
            return;
        }
        if (btn4.Text == "O")
        {
            brojNiza = biranje.biranjePoteza();
            return;
        }
        if (btn7.Text == "O")
        {
            brojNiza = biranje.biranjePoteza();
            return;
        }
        int brojGumba = biranje.odrediGumb();
        switch (brojGumba)
        {
            case 0:
                if (btn1.Text == "␣")
                {
                    btn1.Text = "X";
                    btn1.Enabled = false;
                    odigrano = true;
                }
                break;
            case 1:
                if (btn4.Text == "␣")
                {
                    btn4.Text = "X";
                    btn4.Enabled = false;
                    odigrano = true;
                }
                break;
            case 2:
                if (btn7.Text ==  "␣")
                {
                    btn7.Text = "X";
                    btn7.Enabled = false;
                    odigrano = true;
                }
                break;
        }
    }

    private void drugiOkomito()
    {
        if (!provjeriDostupnost())
        {
            odigrajPotez();
            return;
        }
        if (btn2.Text == "O")
        {
            brojNiza = biranje.biranjePoteza();
```

```
            return;
        }
        if (btn5.Text == "O")
        {
            brojNiza = biranje.biranjePoteza();
            return;
        }
        if (btn8.Text == "O")
        {
            brojNiza = biranje.biranjePoteza();
            return;
        }
        int brojGumba = biranje.odrediGumb();
        switch (brojGumba)
        {
            case 0:
                if (btn2.Text == "␣")
                {
                    btn2.Text = "X";
                    btn2.Enabled = false;
                    odigrano = true;
                }
                break;
            case 1:
                if (btn5.Text == "␣")
                {
                    btn5.Text = "X";
                    btn5.Enabled = false;
                    odigrano = true;
                }
                break;
            case 2:
                if (btn8.Text == "␣")
                {
                    btn8.Text = "X";
                    btn8.Enabled = false;
                    odigrano = true;
                }
                break;
        }
    }

    private void treciOkomito()
    {
        if (!provjeriDostupnost())
        {
            odigrajPotez();
            return;
        }
        if (btn3.Text == "O")
        {
            brojNiza = biranje.biranjePoteza();
            return;
```

```
    }
    if (btn6.Text == "O")
    {
        brojNiza = biranje.biranjePoteza();
        return;
    }
    if (btn9.Text == "O")
    {
        brojNiza = biranje.biranjePoteza();
        return;
    }
    int brojGumba = biranje.odrediGumb();
    switch (brojGumba)
    {
        case 0:
            if (btn3.Text == "␣")
            {
                btn3.Text = "X";
                btn3.Enabled = false;
                odigrano = true;
            }
            break;
        case 1:
            if (btn6.Text == "␣")
            {
                btn6.Text = "X";
                btn6.Enabled = false;
                odigrano = true;
            }
            break;
        case 2:
            if (btn9.Text == "␣")
            {
                btn9.Text = "X";
                btn9.Enabled = false;
                odigrano = true;
            }
            break;
    }
}

private void goreLijevo()
{
    if (!provjeriDostupnost())
    {
        odigrajPotez();
        return;
    }
    if (btn1.Text == "O")
    {
        brojNiza = biranje.biranjePoteza();
        return;
    }
```

```
    if (btn5.Text == "O")
    {
        brojNiza = biranje.biranjePoteza();
        return;
    }
    if (btn9.Text == "O")
    {
        brojNiza = biranje.biranjePoteza();
        return;
    }
    int brojGumba = biranje.odrediGumb();
    switch (brojGumba)
    {
        case 0:
            if (btn1.Text == "␣")
            {
                btn1.Text = "X";
                btn1.Enabled = false;
                odigrano = true;
            }
            break;
        case 1:
            if (btn5.Text == "␣")
            {
                btn5.Text = "X";
                btn5.Enabled = false;
                odigrano = true;
            }
            break;
        case 2:
            if (btn9.Text == "␣")
            {
                btn9.Text = "X";
                btn9.Enabled = false;
                odigrano = true;
            }
            break;
    }
}

private void goreDesno()
{
    if (!provjeriDostupnost())
    {
        odigrajPotez();
        return;
    }
    if (btn3.Text == "O")
    {
        brojNiza = biranje.biranjePoteza();
        return;
    }
    if (btn5.Text == "O")
```

```
    {
        brojNiza = biranje.biranjePoteza();
        return;
    }
    if (btn7.Text == "O")
    {
        brojNiza = biranje.biranjePoteza();
        return;
    }
    int brojGumba = biranje.odrediGumb();
    switch (brojGumba)
    {
        case 0:
            if (btn3.Text == "␣")
            {
                btn3.Text = "X";
                btn3.Enabled = false;
                odigrano = true;
            }
            break;
        case 1:
            if (btn5.Text == "␣")
            {
                btn5.Text = "X";
                btn5.Enabled = false;
                odigrano = true;
            }
            break;
        case 2:
            if (btn7.Text == "␣")
            {
                btn7.Text = "X";
                btn7.Enabled = false;
                odigrano = true;
            }
            break;
    }
}

//Funkcija za postavljanje forme na pocetan polozaj za igru
private void omoguciPoteze()
{
    btn1.BackColor = Color.White;
    btn2.BackColor = Color.White;
    btn3.BackColor = Color.White;
    btn4.BackColor = Color.White;
    btn5.BackColor = Color.White;
    btn6.BackColor = Color.White;
    btn7.BackColor = Color.White;
    btn8.BackColor = Color.White;
    btn9.BackColor = Color.White;
    btn1.Text = "␣";
    btn2.Text = "␣";
```

```
    btn3.Text = "␣";
    btn4.Text = "␣";
    btn5.Text = "␣";
    btn6.Text = "␣";
    btn7.Text = "␣";
    btn8.Text = "␣";
    btn9.Text = "␣";
    txtPobjeda.Text = "␣";
    btn1.Enabled = true;
    btn2.Enabled = true;
    btn3.Enabled = true;
    btn4.Enabled = true;
    btn5.Enabled = true;
    btn6.Enabled = true;
    btn7.Enabled = true;
    btn8.Enabled = true;
    btn9.Enabled = true;
    btnStart.Enabled = false;
}

// Funkcija za provjeravanje mogucnosti pobjede racunala
private bool provjeriDostupnost()
{
    if((btn1.Text == "O" || btn2.Text == "O" || btn3.Text == "O") &&
        (btn4.Text == "O" || btn5.Text == "O" || btn6.Text == "O") &&
        (btn7.Text == "O" || btn8.Text == "O" || btn9.Text == "O") &&
        (btn1.Text == "O" || btn4.Text == "O" || btn7.Text == "O") &&
        (btn2.Text == "O" || btn5.Text == "O" || btn8.Text == "O") &&
        (btn3.Text == "O" || btn6.Text == "O" || btn9.Text == "O") &&
        (btn1.Text == "O" || btn5.Text == "O" || btn9.Text == "O") &&
        (btn3.Text == "O" || btn5.Text == "O" || btn7.Text == "O"))
    {
        return false;
    }
    return true;
}

private void odigrajPotez()
{
    if (btn1.Text == "␣")
    {
        btn1.Text = "X";
        btn1.Enabled = false;
        odigrano = true;
        return;
    }
    if (btn2.Text == "␣")
    {
        btn2.Text = "X";
        btn2.Enabled = false;
        odigrano = true;
        return;
    }
```

```
    if (btn3.Text == "␣")
    {
        btn3.Text = "X";
        btn3.Enabled = false;
        odigrano = true;
        return;
    }
    if (btn4.Text == "␣")
    {
        btn4.Text = "X";
        btn4.Enabled = false;
        odigrano = true;
        return;
    }
    if (btn5.Text == "␣")
    {
        btn5.Text = "X";
        btn5.Enabled = false;
        odigrano = true;
        return;
    }
    if (btn6.Text == "␣")
    {
        btn6.Text = "X";
        btn6.Enabled = false;
        odigrano = true;
        return;
    }
    if (btn7.Text == "␣")
    {
        btn7.Text = "X";
        btn7.Enabled = false;
        odigrano = true;
        return;
    }
    if (btn8.Text == "␣")
    {
        btn8.Text = "X";
        btn8.Enabled = false;
        odigrano = true;
        return;
    }
    if (btn9.Text == "␣")
    {
        btn9.Text = "X";
        btn9.Enabled = false;
        odigrano = true;
        return;
    }
}

enum enumPobjeda
{
```

```
    prviVodoravno,
    drugiVodoravno,
    treciVodoravno,
    prviOkomito,
    drugiOkomito,
    treciOkomito,
    goreLijevo,
    goreDesno
};

public int biranjePoteza()
{
    Random rnd = new Random();
    int broj = rnd.Next();
    broj = broj % 8;
    return broj;
}

public int odrediGumb()
{
    Random rnd = new Random();
    int brojGumba = rnd.Next();
    brojGumba = brojGumba % 3;
    return brojGumba;
}
```