KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Algirdas Dobrovolskis

Daugelio agentų sistemos

Magistro darbas

Darbo vadovas
Prof. habil. dr. E. Kazanavičius

Kaunas, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Algirdas Dobrovolskis

Daugelio agentų sistemos

Magistro darbas

Recenzentas
dr. J. Čeponis

2007-05-28

Vadovas
Prof.habil dr. E. Kazanavičius

2007-05-24

Atliko
IFM-1/1 gr. stud.
Algirdas Dobrovolskis

2007-05-28

Kaunas, 2007

TURINYS

Lentelių sąrašas

Paveikslų sąrašas

SUMMARY

Multi-agent systems are a rapidly developing area of research. Languages are used for understandability of agent analysis, modeling and computerization processes for analysts as soon as system developers. The advantages of visual modeling are standard notation, unified concepts, and intuitive use. There are no single language suitable multi-agent systems modeling. Several languages were analyzed: AUML, objecticly oriented Petri networks, Agent Markup Language (AML), Agent Description Language (ADL), Q scenario description language, Multi-Agent Modeling Language (MAML), SCR++ - A Language Design for Real - Time Multi-Agent Systems), SDLM, with regards to their possibilities to represent and execute multi-agent system processes.

It is proposed to extend UML functional, static and communication models with AUML stereotypes in multi-agents process modeling with succeeding transformation to execution language. Methodology was created describing a way to design multi-agent systems with UML models extended with AUML. The exclusive feature in proposed way of modeling lies in joining created models and making possible to analyst prepare program code for system developer. In order to test the proposed methodology scenario for modeling was prepared and models were designed: three models using UML notation, and one model using AUML extension.

For implementation of proposed method, UML CASE tool Magic Draw was extended with stereotypes required for the proposed method of modeling. User templates were created to ease modeling based by proposed scenario.

Key words: UML, multi-agent systems, AUML, methodology.

ĮVADAS

Plačiai naudojamos paskirstyto dirbtinio intelekto sistemos vis dažniau analizuojamos kaip daugelio agentų sistemos. Patogu tirti agentų sistemas, nes agentai būdami įvairių tipų lanksčiai prisitaiko prie paskirstytos sistemos poreikių. Jau dabar agentų sistemos naudojamos mažų įmonių, universitetuose, bei tarptautinėse kompanijose (Alcatel, Apple, AT&T, BT, Daimler-Benz, DEC, HP, IBM, Lotus, Microsoft, Oracle, Sharp). Teigiama, kad per keletą ateinančių metų daugelis naujų informacinių technologijų projektų naudos įdiegtas agentais pagrįstas sistemas[16].

Agentas - tai abstraktus arba fizinis nepriklausomas objektas, vykdantis gautas užduotis pagal surinktus iš aplinkos duomenis ir reaguodamas į aplinkos pokyčius sėkmingai atlieka jam skirtą darbą. Struktūriškai jis yra sensorių, pavarų ir sprendimo algoritmų rinkinys. Pagal elgseną agentas tai žymėjimas nuo vidinės erdvės (į ką agentas reaguoja) iki išorinės erdvės (ką agentas gali paveikti)[2].

Daugelio agentų sistema - tai autonominių bendradarbiaujančių agentų rinkinys siekiantis bendro tikslo, bet lygiagrečiai kiekvienas agentas siekia savo individualaus tikslo. Daugelio agentų sistema skirta pasiekti tikslams, sunkiai įgyvendinamiems pavieniams agentams ar vientisoms sistemoms[1].

Darbo tyrimo sritis – daugelio agentų sistemos modeliavimas, kur šiuo metu atliekama nemažai tyrimų. norint išspręsti iškilusias problemas: didelė modeliavimo kalbų aibė - skirtingos kalbos akcentuoja skirtingas agentų savybes ir nėra nė vienos, kuri galėtų detaliai aprašyti agentų sistemas. Daugelis modeliavimo kalbų yra tekstinio pavidalo arba joms dar nesukurti grafiniai modeliavimo įrankiai, renkantis modeliavimo kalbą yra daug kriterijų. Tam turėtų būti naudojama praktiniam modeliui sukurti vaizdinė kalba, leidžianti aprašyti agentų veikimo procesus, tarpusavio sąveikas, vidinius procesus, pranešimus, susiejant juos su agentų būsenų pokyčiais. Ši kalba turėtų būti realizuota grafiniame modeliavimo įrankyje, kuris užtikrintų agentų sistemos modelių kūrimą, išsaugojimą ir keitimą į programos aprašą ar perdavimą projektavimo įrankiams. Yra daug agentų modeliavimo kalbų, kurios gerai įgyvendina kai kuriuos iš šių reikalavimų, tačiau nėra tokios kalbos, kuri tenkintų visus išvardintus reikalavimus [3].

Šio darbo tikslas - atlikti daugelio agentų sistemų modeliavimo kalbų analizę ir parinkus modeliavimo kalbą aprašyti modeliavimo eigą bei sukurti daugelio agentų sistemos modelį pagal sudarytą scenarijų. Darbe sprendžiami uždaviniai:

- ištirti daugelio agentų sistemų charakteristikas, kategorijas bei komunikavimo metodus tarp agentų ir su išorine aplinka;
- atlikti modeliavimo kalbų palyginimą, atskleidžiant kiekvienos privalumus ir trūkumus;

- daugelio agentų sistemos modeliavimui pritaikyti daugelio agentų sistemų modeliavimo kalbų analizėje parinktą modeliavimo kalbą;
- vizualiai paaiškinti agentų sistemos veikimo principus naudojant agentams būdingą sąveikos modelį;
- pasiūlyti metodiką ir išbandyti ją eksperimentu sukuriant daugelio agentų sistemos modelį pagal aprašytą aprašyti scenarijų.

Darbas sudarytas iš 3 pagrindinių skyrių. Taip pat yra darbo santrauka anglų kalba (summary), darbo išvados, literatūros sąrašas, santrumpų žodynas, turinys, lentelių ir paveikslų sąrašai. Darbo struktūra:

1 skyriuje nagrinėjamos daugelio agentų sistemų kategorijos, charakteristikos. Pateikti agentų sistemų klasifikacijos ir komunikavimo metodai tarp agentų. Komunikavimo metodai palyginti tarpusavyje nustatant jų privalumus ir trūkumus. Atlikta vaizdinio modeliavimo kalbų, skirtų agentų sistemoms modeliuoti, analizė. Analizuojamos unifikuota modeliavimo kalba (UML) su agentų sistemų išplėtimu (AUML), architektūra aprašyta objektiškai orientuotais Petri tinklais, AML (Agent Markup Language) agentų sužymėjimo kalba, ADL (Agent Description Language) agento aprašymo kalba, Q agentų scenarijaus aprašymo kalba, MAML (Multi-Agent Modeling Language) – daugelio agentų modeliavimo kalba, SCR++ (A Language Design for Real - Time Multi-Agent Systems) - išplečiama kalba daugelio agentų sistemoms, SDLM – daugelio agentų organizacijos modeliavimo kalba. Atliktas modeliavimo kalbų palyginimas, atskleidžiant kiekvienos privalumus ir trūkumus, taip parodant, kad universalios modeliavimo metodikos, kuri tenkintų daugelį reikalavimų, nėra.

2 skyriuje pateikiama daugelio agentų sistemos modeliavimo metodika UML funkciniams modeliams, statiniams modeliams ir komunikavimo modeliams juos papildant agentų išplėtimu - AUML sąveikos diagrama.

3 skyriuje atliktas eksperimentas. Pagal Microsoft PH2007[16] paketo scenarijų sudarytas agentų sistemos aprašymas ir pagal pasiūlytą 2 skyriuje modeliavimo metodiką sudarytas eksperimentinis agentų sistemos modelis.

1. DAUGELIO AGENTŲ SISTEMŲ IR JŲ MODELIAVIMO KALBŲ ANALIZĖ

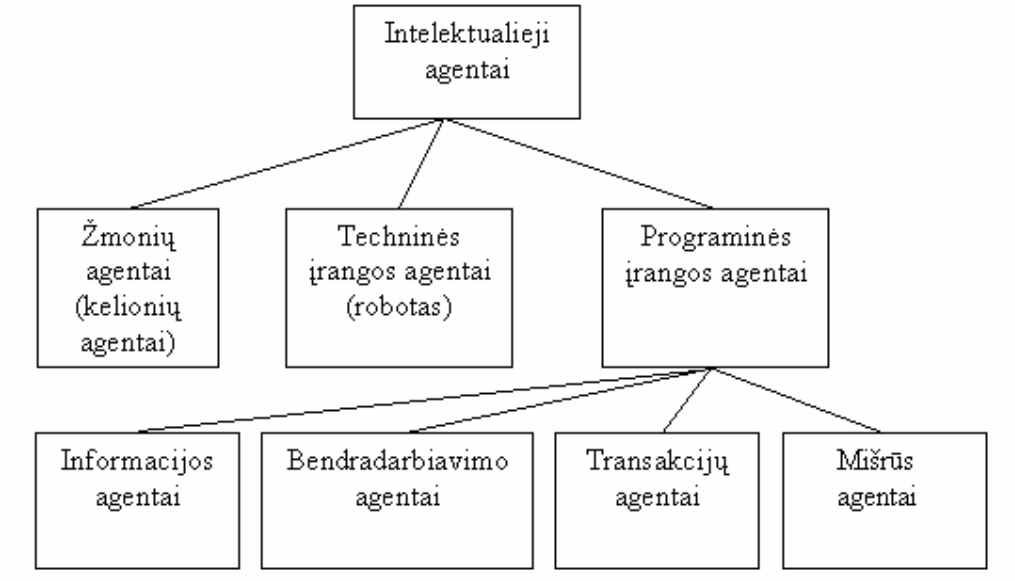
Struktūriškai, agentas - tai sensorių, sprendimo parinkėjų ir pavarų rinkinys[2]. Pagal elgseną - tai žymėjimas nuo vidinės erdvės (į ką agentas reaguoja) iki išorinės erdvės (ką agentas gali paveikti). Agentų pavyzdžiai: ląstelės, skruzdėlės, kompiuterių programos robotai ir žmonės.

Didesni agentai (daugelio agentų sistemos) - tai mažesnių agentų organizacija. Jų pavyzdžiai: imuninė sistema, nervų sistema, daugialąsteliniai organizmai, ekologinės sistemos, vabalų bendruomenės, paskirstyti skaičiavimai, komunikacijų tinklai, Petri tinklai, evoliucijos algoritmai, dirbtinė gyvybė, ekonominės sistemos, korporacijos, Internetas, elektros grandinių valdymo sistemos.

1.1. Agentų sistemų tipai

Pagrindiniai agentų tipai[1]:

- Žmonių agentai (*Human agents*);
- Techninės įrangos agentai (*Hardware agents*);
- Programinės įrangos agentai (*Software agents*).



1.1. pav. Agentų tipai

1.1. paveiksle *intelektualūs agentai* yra apibrėžiami, kaip programinės įrangos komponentai, kurie gali vykdyti specifines užduotis ir valdyti intelekto laipsnį, kuris leidžia bendrauti su aplinka ir atlikti užduočių dalis automatiškai. *Informacijos agentų* paskirtis- informacijos paieška paskirstytose sistemose arba tinkluose. *Bendradarbiavimo agentų* paskirtis - išspręsti sudėtingas

problemas naudojant komunikavimo ir bendradarbiavimo mechanizmus. *Transakcijų agentų* paskirtis - vykdyti ir tikrinti transakcijas. *Mišrūs agentai* sujungia kelių rūšių agentų savybes.

Agentai taikomi:

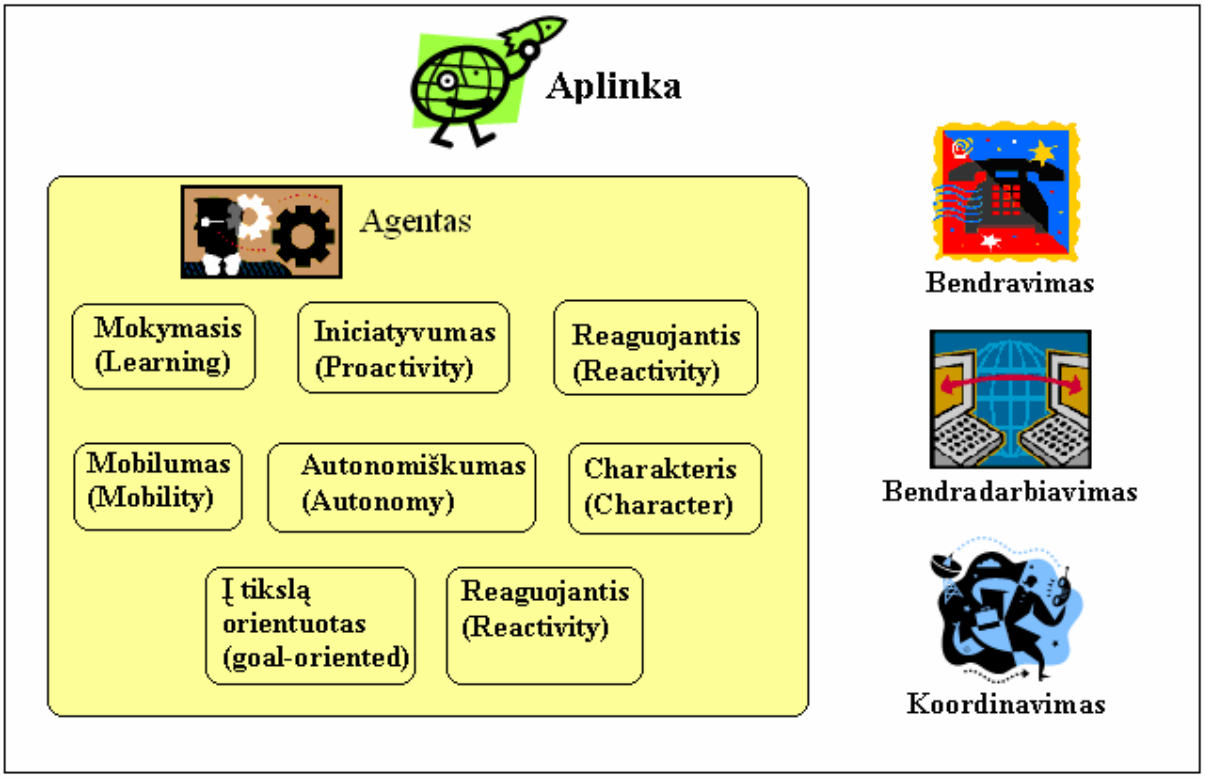
- Paprasti el. pašto filtrai
- Mobilūs taikymai
- Intelektuali pagalba (asistavimas)
- Sudėtingų kritinių sistemų valdymas (oro erdvės valdymas)

1.2. Intelektualiųjų agentų charakteristikos

Intelektualiųjų agentų charakteristikos 1.2. paveiksle grupuojamos į dvi dideles grupes[6]:

- *Vidinės savybės* formuoja agento "vidų", tai yra apibrėžia veiksmą agento viduje (galimybė mokytis, reaguojamumas, autonomiškumas ir tikslingumas (goal-orientedness));
- Išorinėms savybėms priklauso visos charakteristikos, kurios įtakoja kelių agentų bendravimą (tos savybės kartais vadinamas *socialumu*);

- Reaguojamumas - agentas privalo atitinkamai reaguoti į daromą poveikį arba informaciją iš savo aplinkos;
- Iniciatyvumas/tikslingumas- jeigu intelektualusis agentas nereaguoja į aplinkos pasikeitimus, bet imasi iniciatyvos atsižvelgiant į aplinkybes, tai yra vadinama iniciatyviu elgesiu;
- Samprotavimas/Mokymas - agentas elgiasi racionaliai (protingai), kada jis dirba arti savo tikslų pasiekimo arba ties vienu iš dalinių tikslų. Be to galimybė išmokti iš ankstesnės patirties ir sėkmingai pritaikyti savo elgesį aplinkoje yra svarbi intelektualiam agentų elgesiui;
- Autonomiškumas - Vienas iš svarbiausių skirtumų tarp agentų ir tradicinių programinės įrangos programų yra agentų galimybė *siekti savo tikslų autonomiškai*, be bendravimo arba komandų iš aplinkos;
- Mobilumas - apibrėžia agento galimybę judėti (*navigate*) elektroniniuose bendravimo (*communication*) tinkluose;
- Bendravimas/Bendradarbiavimas **-** agentas dažnai reikalauja bendravimo su savo aplinka, kad pilnai atliktų savo užduotis. Kai bendradarbiauja keli agentai, sudėtingos užduotys spendžiamos greičiau ir geriau.
- Charakteris - jeigu agentas atrodo savo vartotojui, kaip *virtuali asmenybė*, tada jis turi turėti tam tikras žmogaus charakteristikas, kad pateisintų šią rolę. Agento svarbiausios charakteristikos yra *sąžiningumas, pasikliaujamumas, patikimumas.*



1.2. pav. Agentų charakteristikos

1.3. Agentų sistemos klasifikacija

Agentų sistemos klasifikuojamos pagal tris kriterijus[8]:

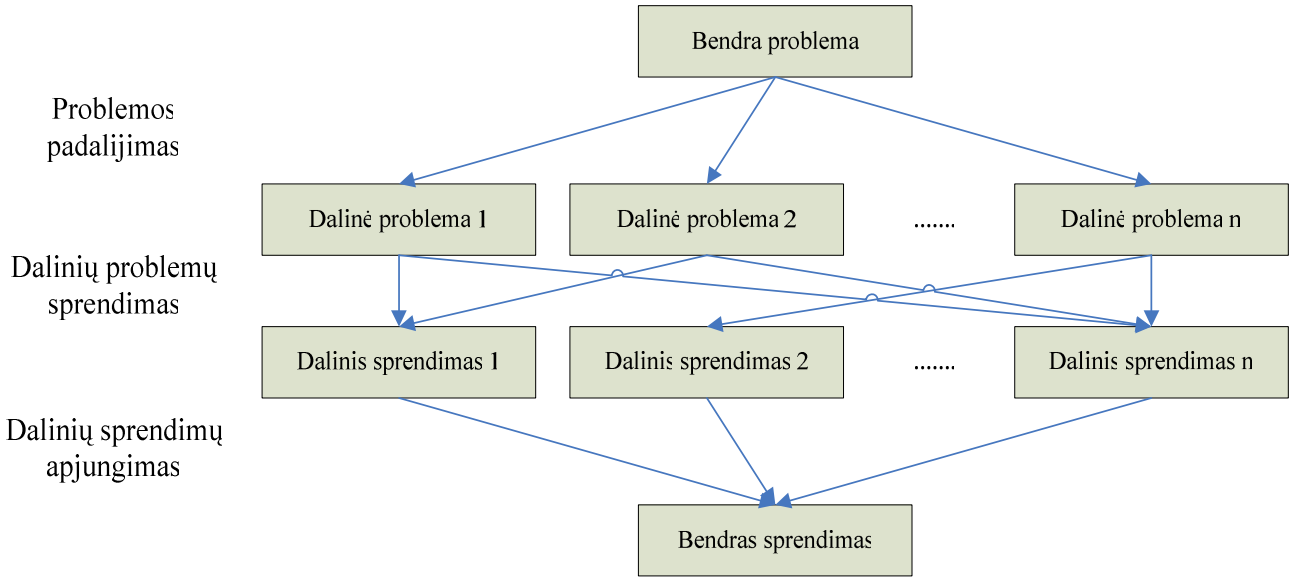
- Intelektas:
  - Paprastas agentas turi ribotą intelekto laipsnį;
  - Sudėtingas agentas turi didesnį intelekto laipsnį;
- Mobilumas:
  - Stacionarūs;
  - Mobilūs;
- Agentų kiekis:
  - Pavienis (*single*) agentas keliauja į aplinką, kuri neturi kitų agentų;
  - Daugelio agentų (*multi-agent*) sistemos susideda iš kelių agentų, kurie gali komunikuoti arba netgi bendradarbiauti vienas su kitu.

Agentas autonominis tada, kai jis neprižiūrimas. Neautonominis vienetas neturi sprendimo laisvės ir vykdo tik prižiūrėtojų nurodymus. Visiškai nepriklausomas agentas neturi prižiūrėtojų ir visiškai nepriklausomai nusprendžia, ką darys ir kada.

Standartinėje daugelio -agentų sistemoje sprendimą priima didelė grupė agentų su tam tikru autonomijos laipsniu. Tokios sistemos elgesys vyksta realiu laiku todėl jo dinamika sunkiai analizuojama ir dažnai prieštarauja intuicijai. Pavyzdžiui elektros tinklo valdymo sistemą sudaro

keletas žmonių ir daugybė mechaninių agentų, įskaitant dešimtis tūkstančių relių. Mechaniniai agentai santykinai paprasti, bet jų kolektyvinė dinamika yra paini dėl kaskadinių sutrikimų (priklausomų sutrikimų sekų) ir fazinių perėjimų (staigūs ir drastiški elgsenos pasikeitimai).

1.3. paveiksle parodytas sprendimo paskirstymas tarp didelio skaičiaus autonominių agentų turi ir privalumų, ir trūkumų. Dauguma trūkumų atsiranda dėl staigios elgsenos, kurią sunku nuspėti, gali būti sunku valdyti ir gali būti nemalonių siurprizų.



1.3. pav. Problemų padalijimas

Jų privalumai:

- Lygiagretus vykdymas: Daugelio uždavinių sunkumas auga eksponentiškai didėjant jų dydžiui. Vienas iš sprendimo būdų - išskaidyti į mažesnes užduotis ir spręsti lygiagrečiai.
- Didelis greitis: Sistemos refleksai pagreitinami naudojant agentą reikiamoje vietoje, be uždelsimų įtraukiant ilgą komandų grandinę.
- Padidėjęs patikimumas: Kai autonominis agentas sugenda, jo kaimynai gali visiškai arba bent dalinai kompensuoti jo praradimą.
- Agentų taikymo įvairovė

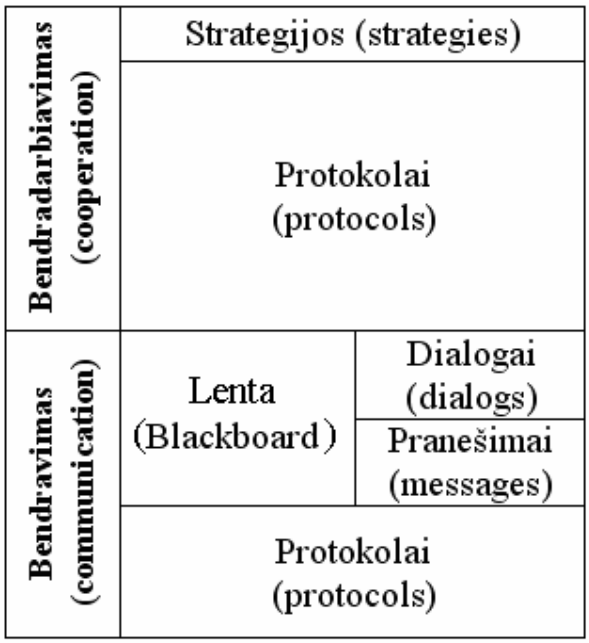
Modeliuojant agentų sistemą, sprendžiamai problemai, reikia pasirinkti:

- Agentus
- Problemos dekompoziciją, skaidant į sub problemas agentams
- Kiekvieno agento autonomijos laipsnį
- Surenkamus agento duomenis(jo vidinės erdvės apimtį)
- Mechanizmą pagal kurį agentai koordinuos savo veiksmus

Nėra metodikos, pasirenkant racionaliausius sprendimus ir visi jie yra svarbūs, bet paskutiniajam sprendimui yra literatūros į kurią galima atsižvelgti.

1.4. Bendravimas ir bendradarbiavimas daugelio-agentų sistemose

1.4. paveiksle agentų bendradarbiavimas sudarytas iš strategijų ir protokolų. Bendravimas tarp agentų naudoja skelbimų lentą, dialogus, pranešimus ir protokolus dialogų bei pranešimų siuntimui.



1.4. pav. Agentų bendravimas ir bendradarbiavimas[10]

Taigi protokolai daugelio agentų sistemose naudojami tiek bendraujant, tiek bendradarbiaujant tarpusavyje.

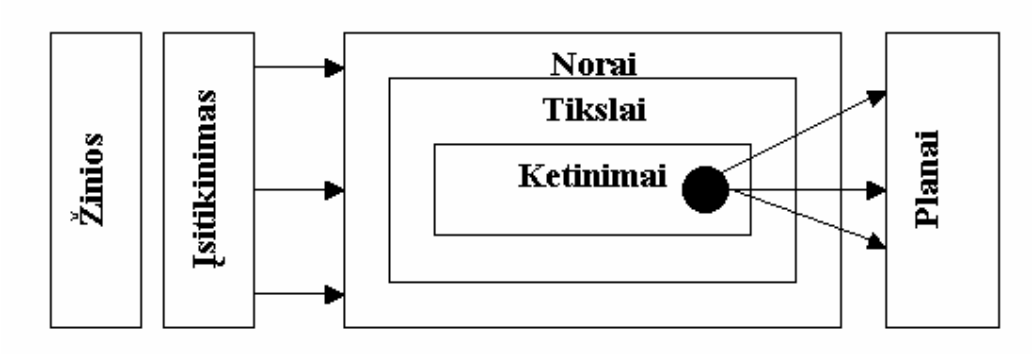
1.5. Kita klasifikacija: Patariamieji agentai

Patariamieji agentai[3] dažnai vadinami *BDI agentais*: (*Belief, Desire, Intention* - įsitikinimas, noras, ketinimas):

- *Įsitikinimai* išreiškia agento fundamentalų požiūrį į jo aplinką. Agentas naudoja juos, kad išreikštų savo lūkesčius dėl galimos būsimos būsenos;
- *Norai* gaunami tiesiogiai iš įsitikinimų. Jie turi agento nuomonę apie būsimą situaciją;
- *Tikslai* - priešingai nei norai, privalo būti realistiški ir neturi konfliktuoti vienas su kitu;
- *Ketinimai* yra *tikslų poaibis*. Jeigu agentas nusprendžia siekti specifinio tikslo, šis tikslas tampa ketinimu;
- *Planai* apjungia agento ketinimus į atitinkamus vienetus. Čia yra artimas sujungimas tarp ketinimų ir planų.

1.6. Bendradarbiavimo agentų struktūra

1.5. paveiksle bendradarbiaujantys agentai sudaryti iš ketinimų, tikslų ir norų. Įtakojant vieną agentą kitų agentų žiniomis ir įsitikinimais galima keisti bendradarbiaujančių agentų norus bei tikslus ir taip įtakoti agentų ketinimus.

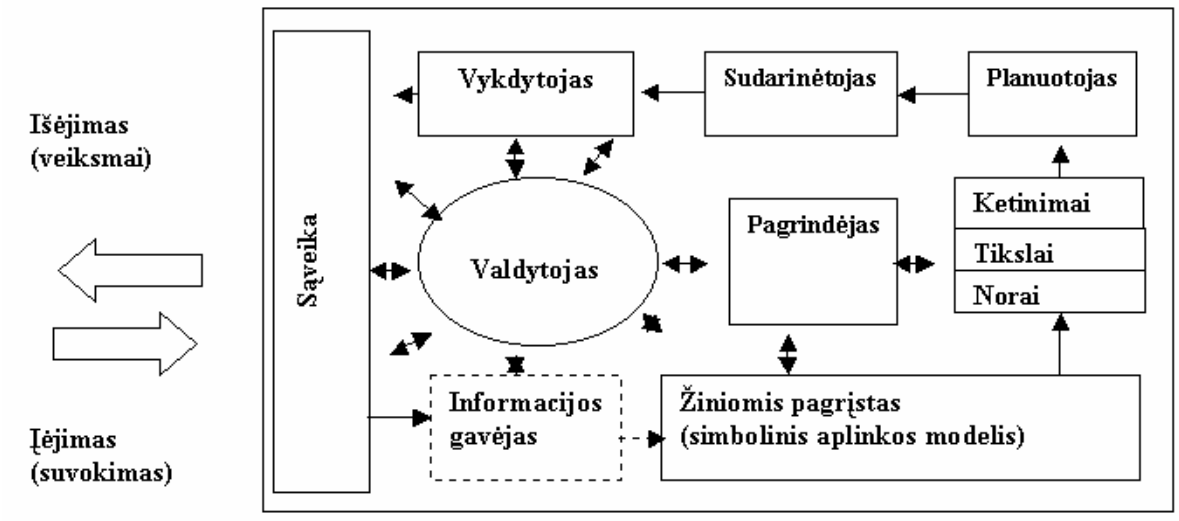


1.5. pav. BDI agento struktūra

1.6. paveiksle patariamųjų agentų ketinimus, tikslus ir norus įtakojantys agentai yra keleto rūšių:

- Vykdytojas;
- Sudarinėtojas;
- Planuotojas;
- Valdytojas;
- Pagrindėjas;
- Informacijos gavėjas.

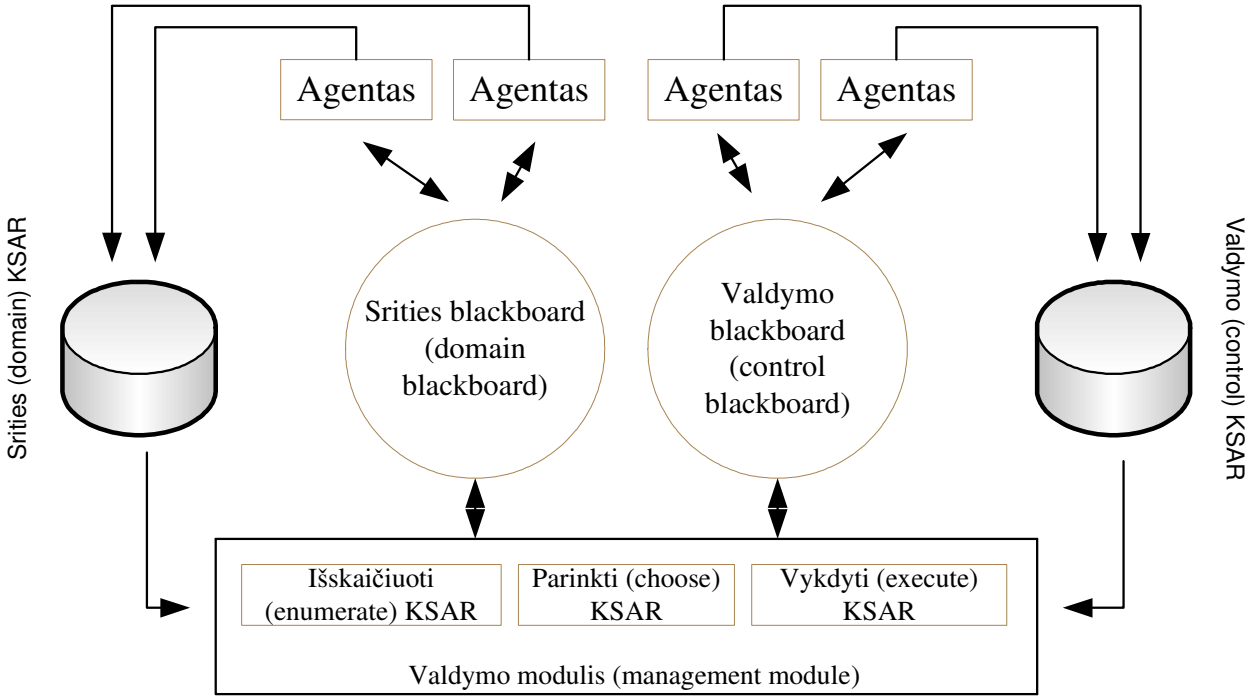
Kiekvienas agentas turi priskirtą rolę, todėl netrukdomas kitų užduočių atlieka jam skirtą užduoties dalį ir perduoda ją kitam agentui.



1.6. pav. Patariamojo agento struktūra[2]

1.7. Bendradarbiavimo agentų modelio architektūra

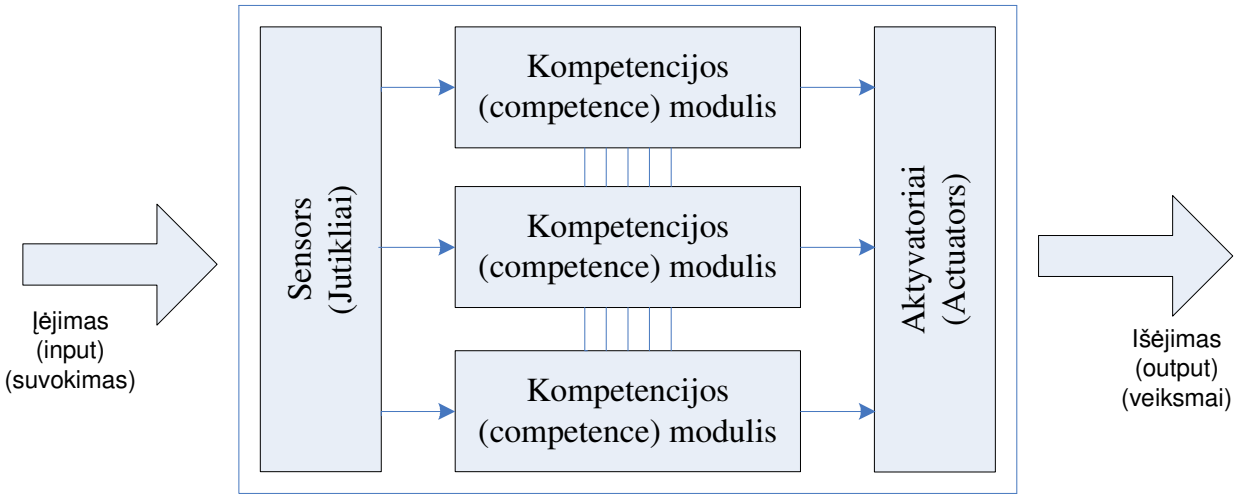
1.7. paveiksle bendradarbiaujančius agentus sudaro srities ir valdymo moduliai. Valdymo modulio veiksmai išskaičiuoti, parinkti ir vykdyti. Naudojamos 2 skelbimų lentos: srities ir valdymo.



1.7. pav. BDI modelio architektūra[2]

1.8. Reaguojančių agentų architektūra

Reaguojančio agento architektūra 1.8. paveiksle sudaryta iš jutiklių, kompetencijos modulių ir aktyvatorių. Jutikliai įrašo informaciją, gautą iš aplinkos. Toliau ji siunčiama į tam tikras užduotis atliekančius modulius. Šie apdoroja informaciją ir suformuoja atitinkamą reakciją, kurią aktyvatoriai (*actuators*) perduoda į aplinką.



1.8. pav. Reaguojančių agentų architektūra[2]

1.9. Stacionarūs ir mobilūs agentai

Agentai gali būti skirstomi į stacionarius ir mobilius.

Stacionarių agentų savybės:

- Galimybė siųsti pranešimus nutolusiems objektams.
- Negali patekti į kitą aplinką, t.y. kompiuterį.
- Principas – nutolęs procedūros iššaukimas - RPC (Remote Procedure Call)

Mobilių agentų savybės:

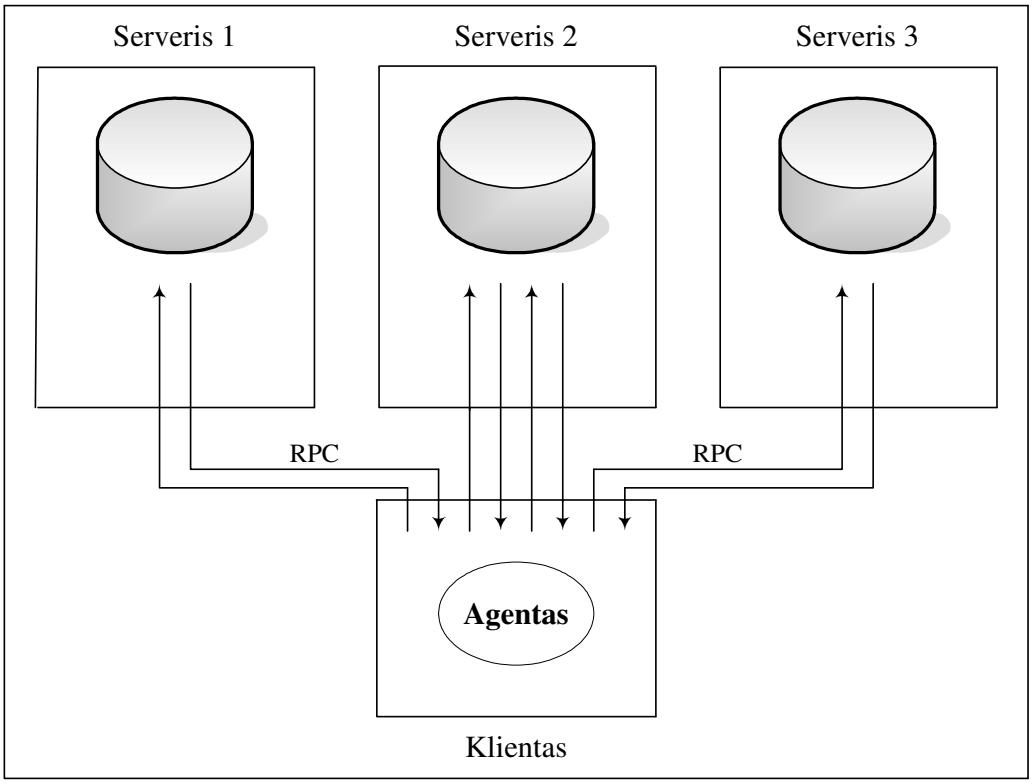
- Galimybė laisvai judėti – migruoti – elektroniniu tinklu ir kompiuteriais.
- Galimybė komunikuoti su aplinkos objektais, pvz., informacijos šaltiniais arba kitais agentais.
- Principas – nutolęs programavimas – RP (Remote Programming).

Tolesniuose skyriuose tirsime stacionarių ir mobilių agentų komunikavimą ir jų skirtumus.

1.9.1. Stacionaraus agento komunikavimas

Komunikavimas 1.9. paveiksle, naudojant RPC, yra atliekamas, remiantis klientas-serveris principu. Visi tinklu siunčiami pranešimai yra arba užklausos (*request*), arba atsakymai (*reply*). Jie keliauja pagal standartinius protokolus, kurie iš anksto yra žinomi.

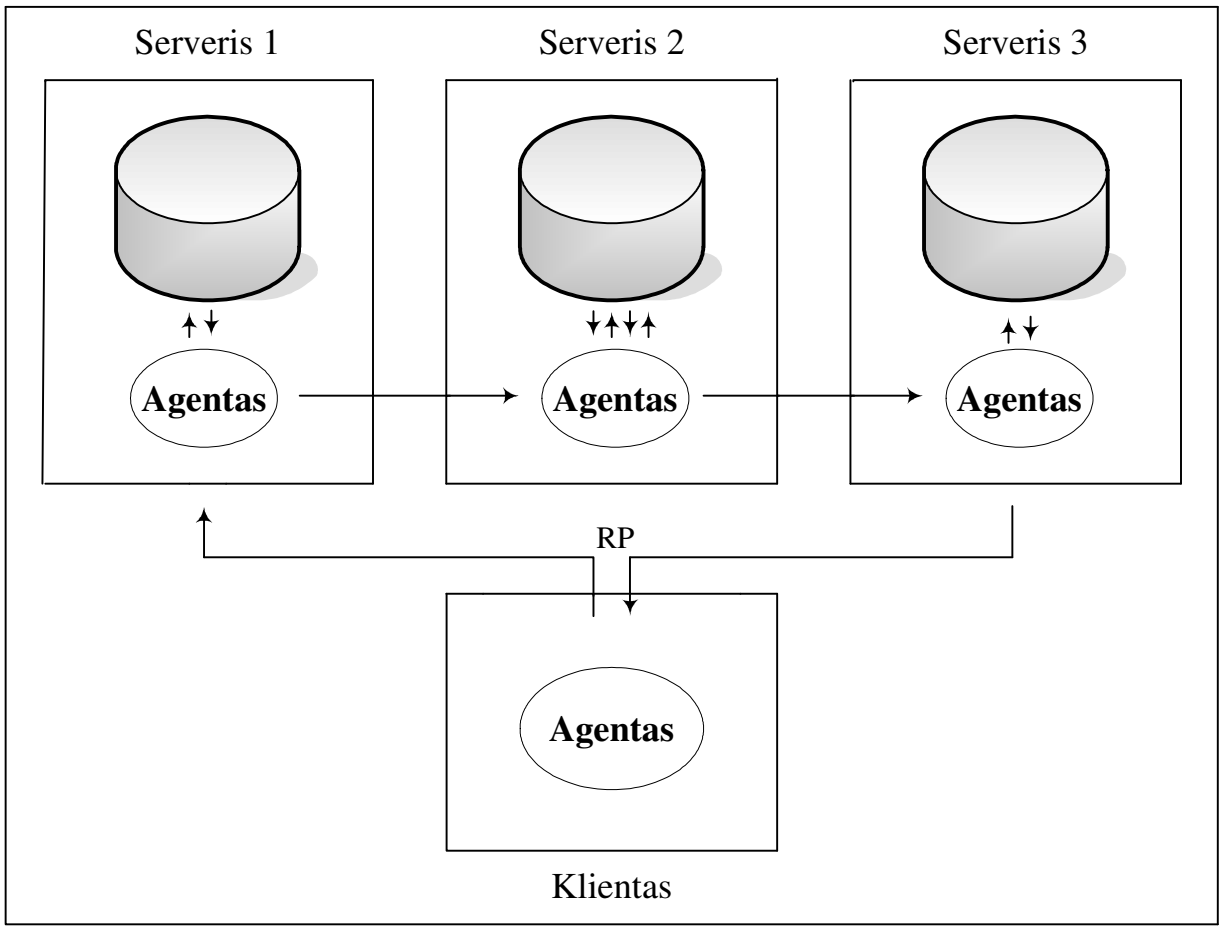
RPC naudingumą riboja apkrautas tinklas ir griežta protokolo struktūra.



1.9. pav. Stacionaraus agento komunikavimas[2]

1.9.2. Mobilių agentų komunikavimas

1.10. paveiksle mobilių agentų RP nuo RPC skiriasi tuo, kad čia nevyksta užklausų ir atsakymų mainai. Vietoj to, klientas iškviečia procedūrą, kuri įvykdoma serveryje. Vietoj užklausų ir atsakymų RP sudaro pranešimai, talpinantys procedūrą ir su ja susijusias duomenų struktūras. Tokie pranešimai atitinka mobilių agentų elgseną. RP pasižymi didesniu lankstumu ir mažiau apkrauna tinklą.



1.10. pav. Mobilių agentų komunikavimas[3]

1.9.3. Mobilių ir stacionarių agentų komunikavimo palyginimas

Palyginkime mobilių ir stacionarių agentų komunikavimą 1 lentelėje pateikiant jų savybes.

1 lentelė. Mobilių ir stacionarių agentų komunikavimo palyginimas

Komunikacija	Savybės	Agentai
Nutolęs programavimas (RP)	- Aukštas intelekto lygis - Lankstumas	Mobilus
Nutolęs procedūros iškvietimas (RPC) SQL paraiškos (queries)	- Žemas intelekto lygis - Nuosavas protokolas - Uždara aplinka	Stacionarus

Mobilių agentų privalumai:

- Sumažinta tinklo apkrova.
- Sumažintas kliento resursų panaudojimas.
- Asinchroninis veikimas.
- Perkonfigūravimo galimybės.
- Aktyvus funkcionavimas.
- Decentralizuota struktūra.

Mobilių agentų trūkumai:

Techninės kliūtys, neleidžiančios pašalinti saugumo problemos:

- Transportavimas/migravimas (*transport/migration*).
- Efektyvumas (*efficiency*).
- Standartai/suderinamumas (s*tandarts/interoperability*).
- Ataskaitų ruošimo sistemos (billing systems).

2 lentelė. Mobilių agentų programinės įrangos lygmenys

Agento lygis	Bazinis funkcionalumas (API): - Mobilumas. - Komunikacija. - Identifikavimas. - Perdavimai (negotiations). - Filtravimas.
Saugumo lygis	Saugumo protokolai. Duomenų šifravimas (encryption). Skaitmeninis parašas. Ugniasienės (firewalls).
Komunikacijos lygis	Protokolai, dokumento formatas, RPC, RP, Nuotolinis metodo iškvietimas (RMI), Objekto platinimas (serialization)

Agento lygmens nurodyto 2 lentelėje funkcijos:

- Stebėti aktyvius agentus kompiuteryje.
- Priversti agentus funkcionuoti.

Saugumo lygmens funkcijos:

- Tinkle pernešti pranešimus ir objektus.
- Užtikrinti, kad objektas nebūtų skaitomas pašalinių.
- Užtikrinti, kad į sistemą nepatektų nežinomas objektas.

- Gali naudoti skaitmeninius parašus ir sertifikatus.

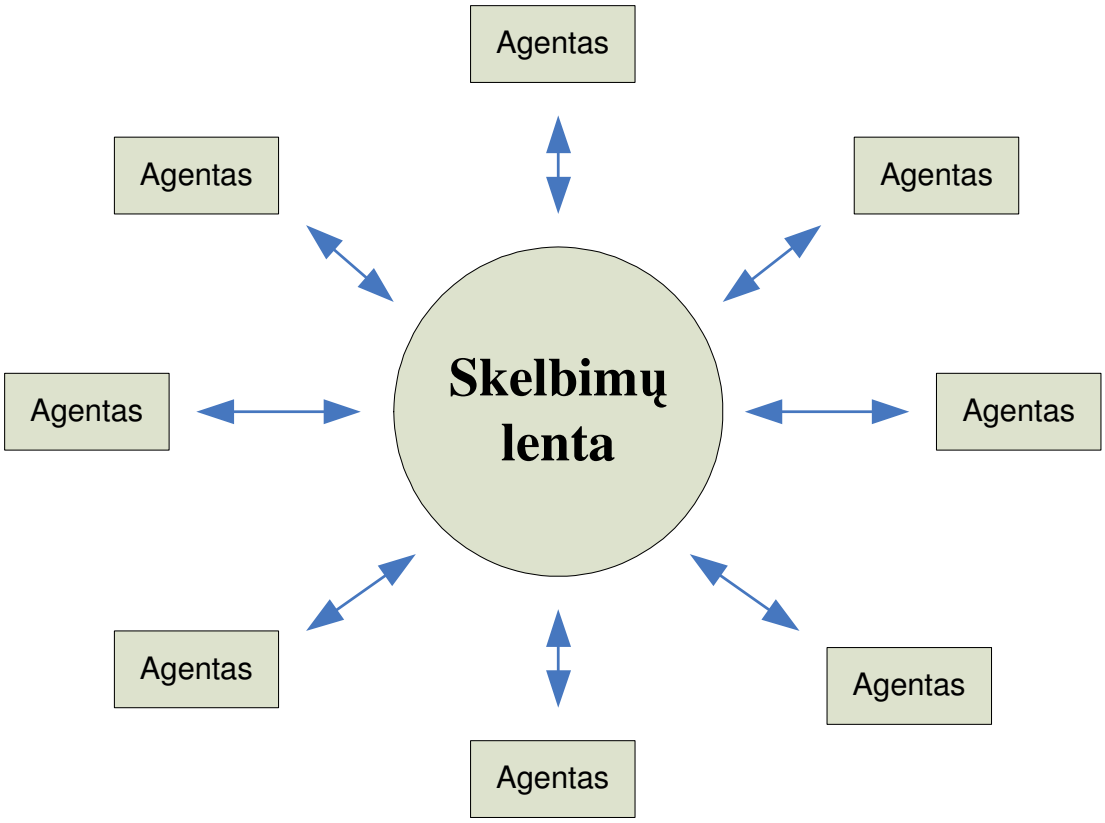
Komunikacijos lygmens funkcijos:

- Saugo siuntimo protokolus, objektų formatus.
- Turi būti realizuota RPC (Remote Procedure Call), RP (Remote Programming) ir objekto serializacija (Object Serialization).

Iš palyginimo matyti, jog ir mobilūs ir stacionarūs agentai turi savo privalumų, todėl gali būti panaudoti skirtinguose taikymuose.

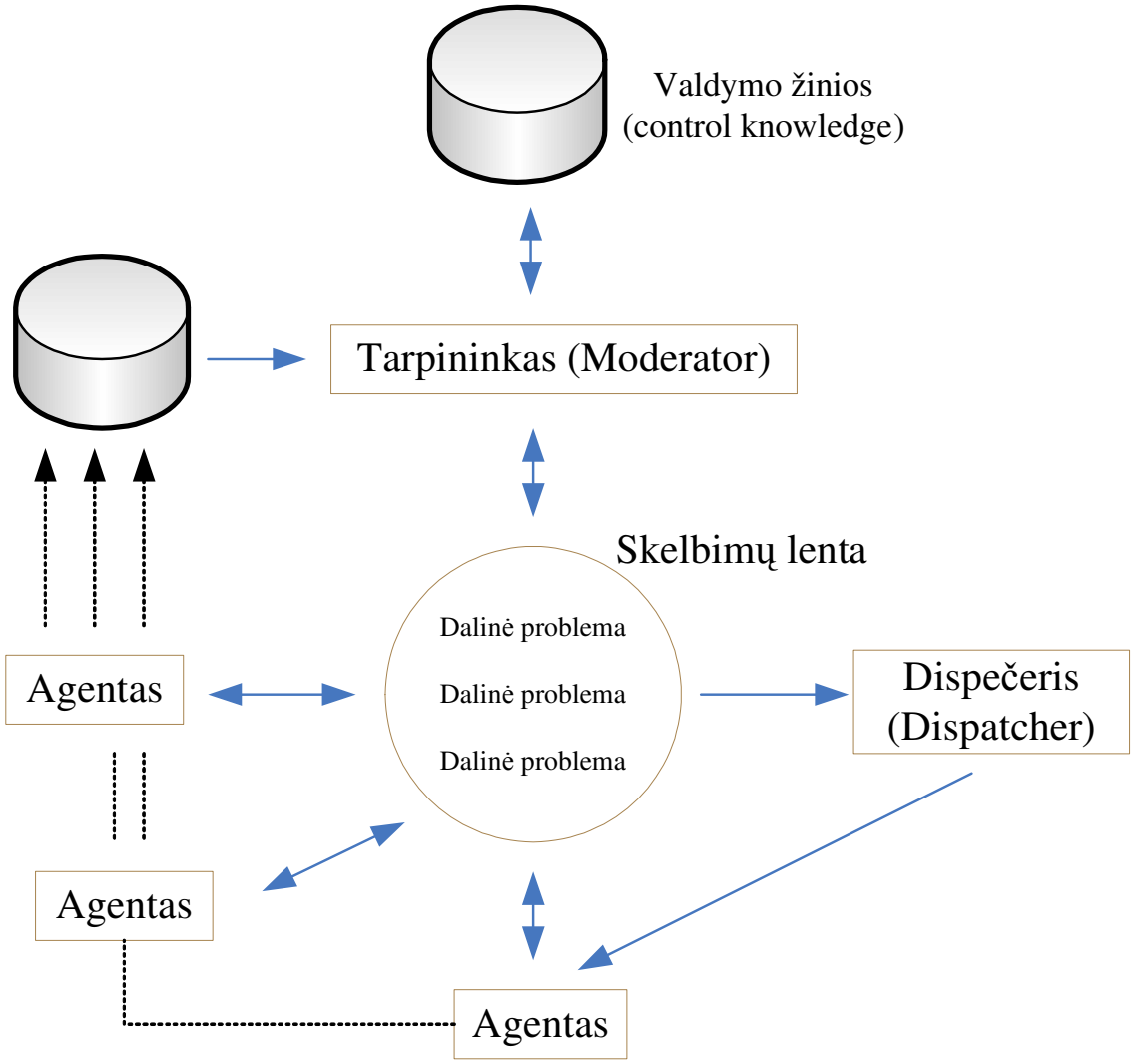
1.10. Agentų bendravimo metodai

Skelbimų lentos (blackboard) metodas 1.11. paveiksle leidžia visiems agentams, kurie sudaro daugelio agentų sistemą keistis informacija ir žiniomis.



1.11. pav. Agentų „skelbimų lenta“

Išplėsta skelbimų lentos (blackboard) struktūra parodyta 1.12 paveiksle. Čia agentai dalines problemas į skelbimų lentą deda ir per tarpininkus, o užduotis gauna iš dispečerio. Taip pagreitinamas problemos sprendimas ir tolygiau padalinamos problemos.



1.12. pav. išplėsta skelbimų lentos struktūra[1]

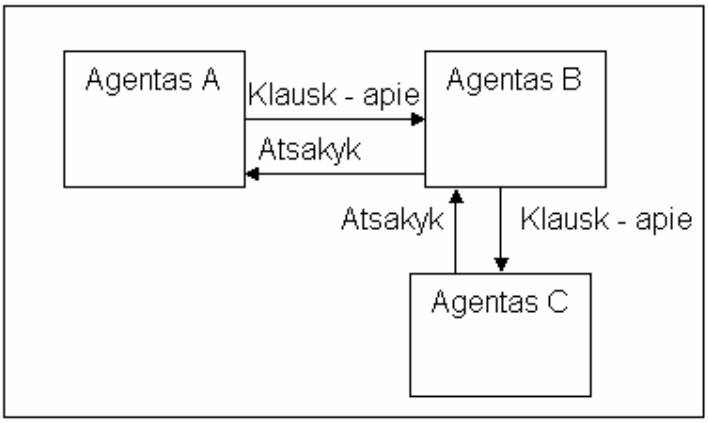
1.13. paveikslas demonstruoja pagrindinį į pranešimus orientuotą agentų sistemos principą. Agentas siuntėjas siunčia pranešimą į agentą gavėją.

1.13. pav. Pranešimas tarp agentų

Nenaudojamas joks duomenų tarpinis saugojimas (buffering) ir kiti agentai negali perskaityti ne jiems adresuotą pranešimą. Taip vadinamas transliavimas (broadcasting) yra išimtis, nes pranešimas siunčiamas visiem sistemos agentams ar specialiai grupei. Tačiau įprastais atvejais siuntėjas prideda unikalų adresą prie pranešimo ir tada pranešimą gali perskaityti tik tas agentas, kuris turi tą adresą. Taigi, komunikavimo protokolas turi būti apibrėžtas taip, kad atitiktų ir

pranešimo formatą, ir komunikavimo kalbą. O svarbiausia, visi susiję agentai turi žinoti komunikavimo kalbos semantiką.

1.14. paveiksle agentas A siunčia pranešimą agentui B, į kurį atsakoma specialia informacija. Agentas B pats reikalauja informacijos iš agento C, nes jis pats nesiunčia reikalingos informacijos. Jis siunčia atitinkamą užklausą agentui C, iš kurio jis gauna atsakymą. Jei atsakyme yra reikalinga informacija, agentas B gali siųsti šį pranešimą agentui A. Taip pat įmanomas kelių nuoseklių ir susijusių pranešimų siuntimas. Sudėtingi dialogai gali turėti neigiamos įtakos interpretuojant pranešimą, nes susietų agentų supratimo lygis keičiasi nuolat. Užklausų klasifikavimas į tipus neišsprendžia problemos, nes neleidžiama keisti dialogo prasmės.



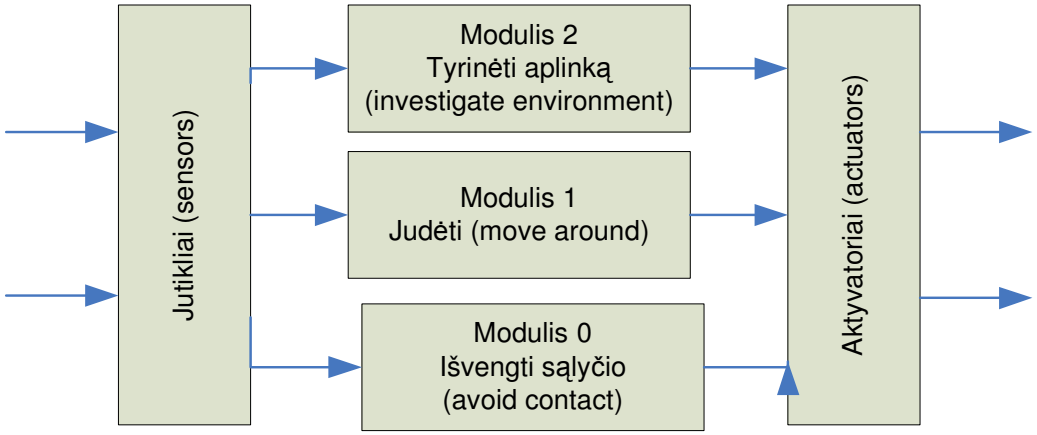
1.14. pav. Dialogo struktūra

1.11. Agentų architektūros

Yra dvi pagrindinės agentų architektūros: Brooks architektūra ir Interrap architektūra. Brooks architektūrą sudaro jutikliai, sprendimo moduliai ir aktyvatoriai. Interrap architektūrą sudaro Žinių bazė, valdymo blokas ir sąsaja. Panagrinėsime abi architektūras detaliau.

1.15. paveiksle parodytas Brooks architektūrą atspindintis roboto pavyzdys:

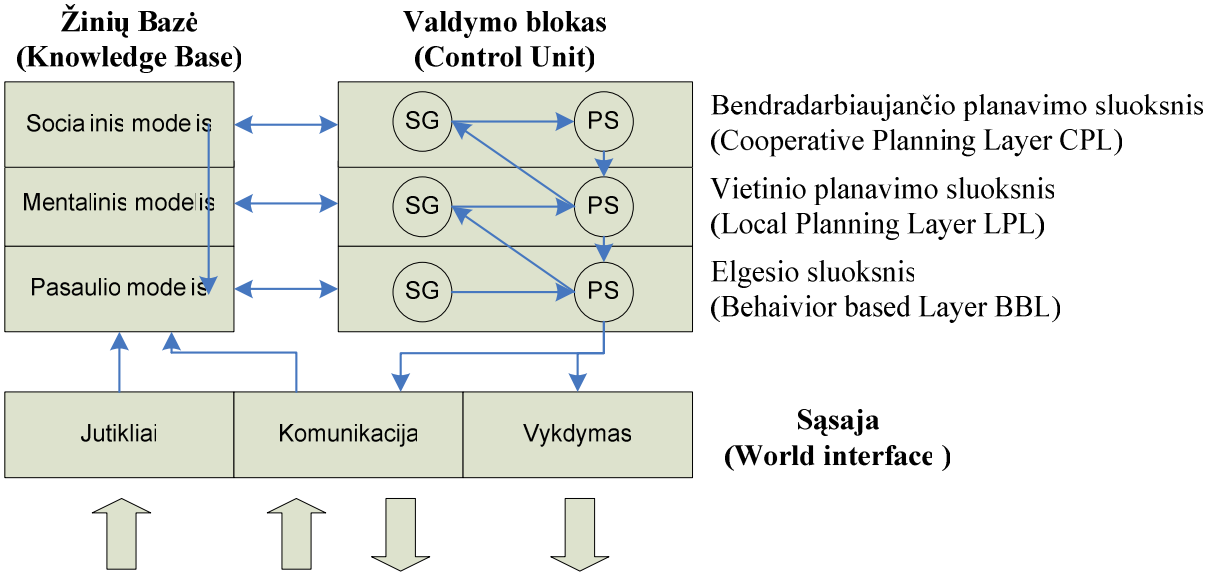
- Nulinis modulis rodo žemiausią lygmenį, kurio paskirtis yra atpažinti kliūtis ir atitinkamai į jas reaguoti. Šis modulis neturi sąryšio su kitais moduliais.
- Pirmasis modulis priverčia robotą judėti po aplinką. Stebint ir įvertinant nulinio modulio įėjimus ir išėjimus, robotui suteikiama galimybė apeiti kliūtis.
- Antrasis modulis, pasinaudodamas pirmuoju, priverčia robotą judėti prie tam tikrų objektų.



1.15. pav. Brooks architektūra[2]

Interrap architektūros 1.16 paveiksle pagrindiniai komponentai:

- Žinių bazė:
  - Socialinis modelis.
  - Mentalinis modelis.
  - Pasaulio modelis.
- Valdymo blokas:
  - Bendradarbiaujančio planavimo sluoksnis (CPL).
  - Vietinio planavimo sluoksnis (LPL).
  - Elgesio planavimo sluoksnis (BBL).
  - Atskirai kiekvieną valdymo bloko lygį sudaro uždavinio aktyvavimo (SG) ir planavimo (PS) moduliai.
- Sąsaja su aplinka.



1.16. pav. Interrap architektūra[2]

Taigi Brooks architektūra daugiau skirta mechaniniams, nesudėtingiems agentams, o Interrap architektūra skirta sudėtingiems, informacijos agentams, bei sistemoms kurios reikalauja turėti žinių bazę.

1.12. Daugelio agentų sistemų modeliavimo kalbų analizė

Modeliuojant agentus galima pasirinkti įvairias kūrimo sistemas, kadangi nėra sukurto standarto:

- Agentų unifikuota modeliavimo kalba AUML;
- Architektūra aprašyta objektiškai orientuotais Petri tinklais;
- Agentų sužymėjimo kalba AML (Agent Markup Language);
- Agento aprašymo kalba ADL (Agent Description Language);
- Agentų scenarijaus aprašymo kalba Q;
- Daugelio agentų modeliavimo kalba MAML (Multi-Agent Modeling Language);
- Išplečiama kalba daugelio agentų sistemoms SCR++ (A Language Design for Real-Time Multi-Agent Systems);
- Daugelio agentų organizacijos modeliavimo kalba SDLM.

Panagrinėsime kiekvieną modeliavimo kalbą atskirai, parodant jų privalumus ir trūkumus. Atliksime jų palyginimą pagal pateiktus reikalavimus ir parinksime modeliavimo kalbą.

1.12.1. Agentų unifikuota modeliavimo kalba AUML

Šis UML kalbos išplėtimas modeliuoja agentus šiose srityse[5]:

- Agento struktūra (klasių diagramos);
- Sistemos tikslai remiantis notacijomis;
- Panaudojimo atvejai;
- Architektūros diagrama;
- Socialiniai aspektai;
- Aplinka;
- Darbo eiga/planavimas;
- Paslaugos gautos sudėjus aplinką ir agento klases;
- Abstrakcijos lygiai;
- Laikinos savybės – gyvos sekos schemos ir laiko sritys;
- Strategijos;
- Įrengimas ir mobilumas;

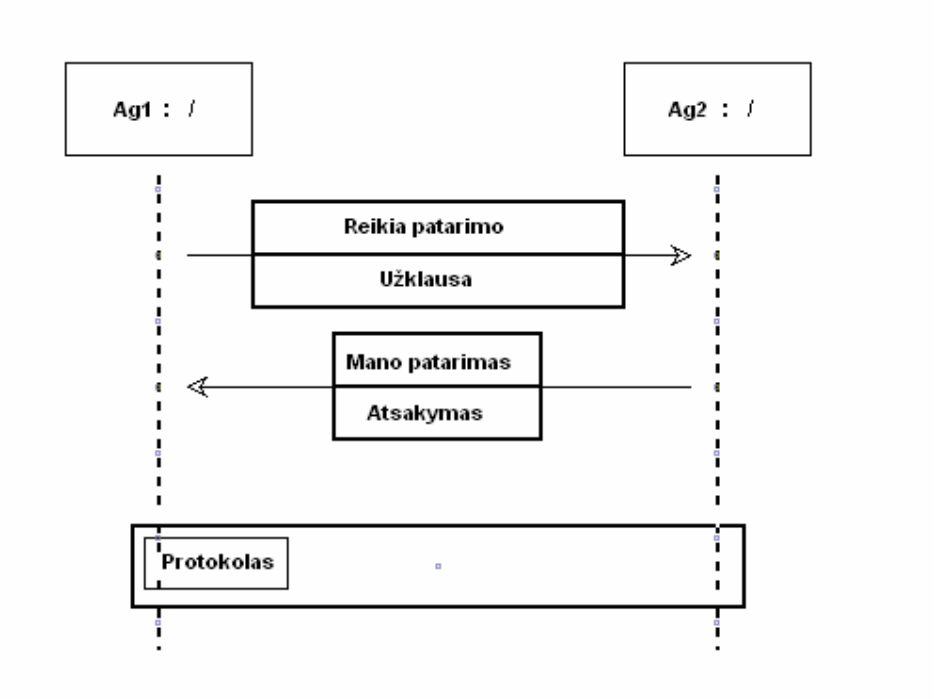
Pagrindiniai aprašymo metodai yra:

Klasių diagramos – nustato agentų vidaus elgseną, elgseną su išore naudojant standartines ir išplėstas UML klasių diagramas.

Sąveikos diagramos – apibūdina objekto sąveikas su kitais objektais. Jose įtrauktos bendradarbiavimo, sekų ir sąveikos apibendrinimo diagramos.

Nors AUML panaudoja UML sukurtus komponentus ir diagramas visur kur įmanoma, tačiau juo neatsiriboja ir esant būtinybei naudoja kitas sistemas arba išplečia UML komponentus ar diagramas.

UML diagramų išplėtimas - sąveikos diagrama, apjungianti komunikacijų diagramą su protokolo diagrama parodyta 1.17 paveiksle.



1.17. pav. Agentų unifikuotos modeliavimo kalbos AUML modelis

1.12.2. Objektiniai Petri tinklai

Taip galima aprašyti statinius ir dinaminius agentus, įvertinti kiekvieno agento savybes individualiai ir bendru sistemos mastu. Agentas nagrinėjamas kaip atskiras komponentas ir agentų bendruomenės dalis(agento ir bendruomenės lygiai). Iš agento lygmens galima analizuoti kiekvieną agentą, iš bendruomenės lygmens galima kurti sistemos dizainą bendravimą tarp agentų. Vizualiai nagrinėjant architektūrą galima paaiškinti agentų sistemą nuo modeliavimo iki sistemos realizavimo, kadangi galutinis produktas yra diagramų kodo generavimas.

Architektūros komponentai:

- Skaičiavimo agentai – bendrauja tarpusavyje, su aplinka ir vartotojais. Juose yra pasirinkimų nustatymai, elgesio aprašymas ir sąsajos;

- Sujungimo agentai – aprašo agentų bendravimą ir nustato taisykles, valdančias agentų bendravimą;
- Apribojimai – daugelio-agentų sistemos taisyklės, kurios nurodo kad pažeidus tam tikrą parametrą sistemai rezultatas tampa mažiau priimtinas arba iš viso neleistinas.

1.12.3. Agentų sužymėjimo kalba AML

Kalba paremta XML kalba. Aprašant agentus elementai aprašo agentą turinyje, o atributai suteikia valdymo duomenis apie agentą[4]. Agentai aprašomi hierarchiniais, susietais arba individualiais failais. Pagrindiniai komponentai:

- <channel> - srities aprašymas;
- <category> - kategorijos nurodymas;
- <agent> - agento aprašymas;
- <source> - agento šaltinis;
- <query> - užklausos aprašymas;
- <result> - užklausos rezultato aprašymas.

Pavyzdžiui:

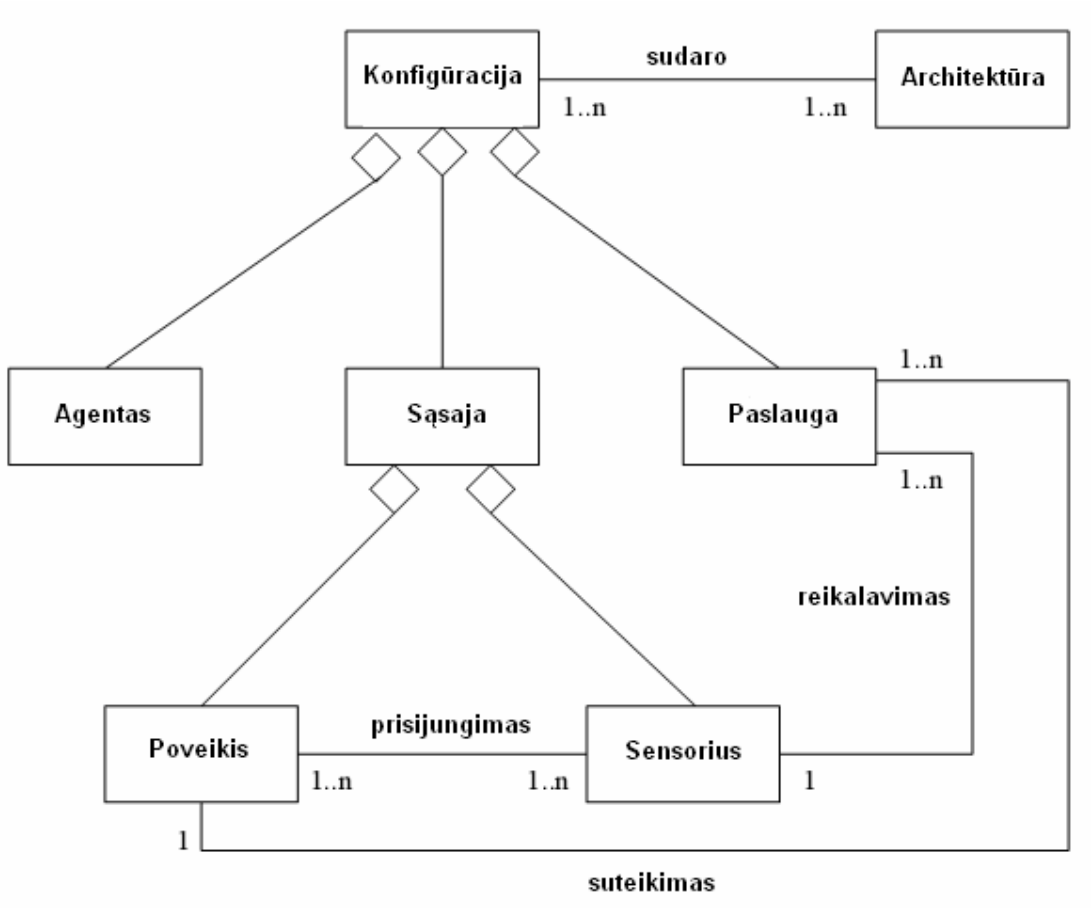
```
<aml:Class ID="Availability">
  <aml:oneOf parseType="daml:collection">
    <aml:Thing rdf:ID="InStock">
      <rdfs:label>In stock</rdfs:label>
    </aml:Thing>
    <aml:Thing rdf:ID="BackOrdered">
      <rdfs:label>Back ordered</rdfs:label>
    </aml:Thing>
    <aml:Thing rdf:ID="SpecialOrder">
      <rdfs:label>Special order</rdfs:label>
    </aml:Thing>
  </aml:oneOf>
</aml:Class>
```

1.12.4. Agentų aprašymo kalba ADL

Kalba skirta agentų architektūros aprašymui. Jos pagrindiniai komponentai[7]:

- Komponentai – skaičiavimų arba duomenų dalys;
- Sąsajos – skirtos bendravimui su išoriniu pasauliu;
- Sujungimai – skirti bendravimui tarp komponentų;
- Konfigūracijos – tai sujungimų ir komponentų grafai apibūdinantys agentų sistemos architektūros struktūrą;
- Teiginiai – sistemos savybės apibūdinančios sistemos dalis, taip kad pažeidus sąlygas būtų atmetamas priešingai teigiančių agentų variantas;
- Hierarchinės kompozicijos – rodo komponentų architektūras kaip vieną komponentą didesnėse sistemų architektūrose.

Agento modelis aprašytas su ADL parodytas 1.18. paveiksle.

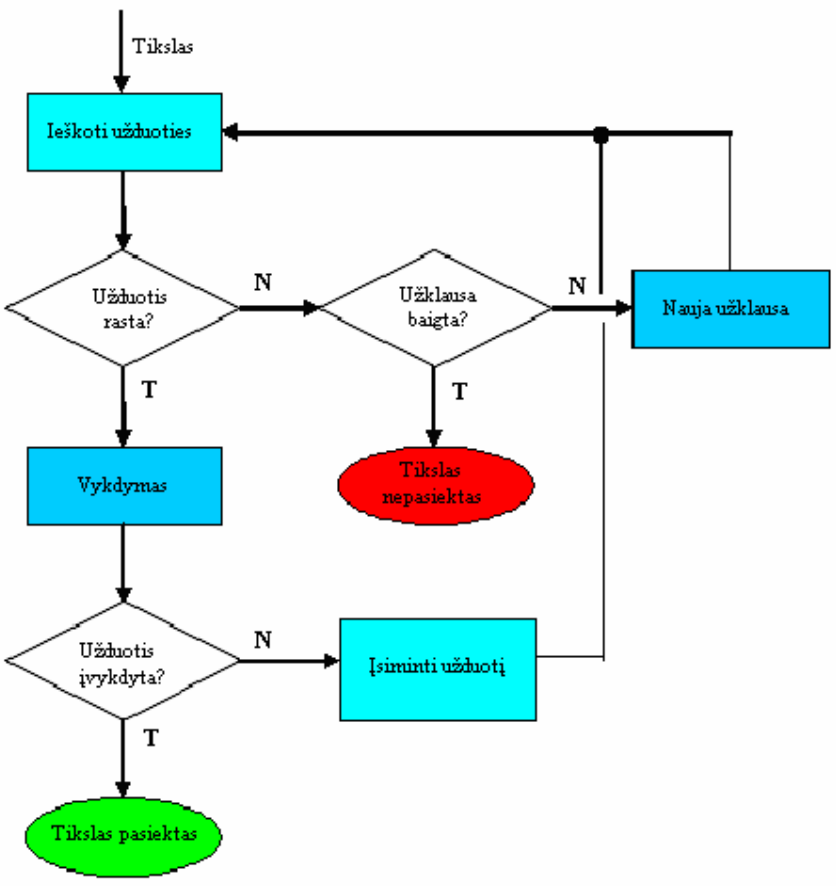


1.18. pav. ADL modelis

JADL tai pagal ADL parengta kalba, skirta protingų agentų architektūros aprašymui. Pagrindiniai jos elementai:

- Plano elementai;
- Taisyklės;
- Ontologijos – įvykių kategorijos į kurias agentai reaguoja;
- Paslaugos – tai agentų veiksmų rinkiniai.

Architektūra apima veiklos planavimą, tvarkaraščio sudarymą ir klaidų apdorojimą. Veiklos planavimas su JADL parodytas 1.19. paveiksle.



1.19. pav. JADL modelis

1.12.5. Agentų scenarijaus aprašymo kalba Q

Tai išplėsta lisp kalba skirta bendraujančių agentų aprašymui[12]. Q pritaikyta aprašyti agentus pagal žmogaus agentų elgesio modelius. Sukurta taip, kad tiktų dviems sistemoms: Microsoft agent ir FreeWalk agents. Kalba leidžia aprašyti visiškai kontroliuojamus agentus. Pavyzdžiui:

```
(defscenario card14()
    (scene1
        (otherwise
        (!speak „text1“)
        (!speak „text2“)
        (!display :url“http://webpage/index.htm“)
        (go scene2)
    (scene2
      ((?watch_web :url“http://webpage/index.htm“)
        (!speak „text3“)
        (!speak „text4“)
        (go scene2))
      ((?watch_web :url“http://webpage/page1.htm“)
        (!gesture :animation „gesture“)
        (!speak „text5“)
        (!speak „text6“)
        (go scene2))
      ((?watch_web :url“http://webpage/page2.htm“)
        (card42 self)
        (go scene2))
       ((?timeout:time 20)
```

```
        (go scene3)))
    (scene3
      (otherwise
        (!speak „text7“)
        (!speak „text8“)
```

Sunku aprašyti ir analizuoti sudėtingus agentus, tinka tik vartotojų pagalbos agentams.

1.12.6. Daugelio agentų modeliavimo kalba MAML

Tai išplėsta Objective-C (naudojant swarm bibliotekas). Sukurtas MAML kompiliatorius, kuris swarm kodą paruošia gcc kompiliatoriui[15]. Kalba yra pradinėje kūrimo stadijoje, tačiau jau galima sukurti agento struktūrą naudojant papildomus programavimo įrankius, palengvinant programavimo dalį aprašomą su MAML. Pavyzdžiui:

```
model EdgeCity {

@var: int NumOfPlaces;
@var: int NumOfFirms;
@var: double A, B, r1, r2, mobility;
@var: double initialFlatness;
@var: int randomMoveFactor;
@var: [] int places;
@var: [] int places;
@schedule cyclic(1) {
   0: @planDef seq {
         @forEach groupOfFirm determineMove;
         @to infoBank updateInformation;
   }}
```

Kompiliatorius kol kas tik kūrimo stadijoje, nėra vizualinio modeliavimo galimybės, reikalingi papildomi programavimo įrankiai.

1.12.7. Išplečiama kalba daugelio agentų sistemoms SCR++

Dar tik kuriama daugelio agentų modeliavimo kalba. Joje planuojama realizuoti[9]:

- Agentų mokymąsi;
- Komunikavimą;
- Loginę sąsają;
- Programos verifikavimą.

Visus šiuos parametrus vertinant realiame laike.

Privalumai: aprašomas C++ kalbos sintakse, todėl greitai kuriamas agentų sistemos kodas.

Trūkumai: darbas dar nėra baigtas, viešai prieinamas taps tik po keleto metų.

1.12.8. Daugelio agentų organizacijos modeliavimo kalba SDLM

Programavimo kalba optimizuota daugelio agentų bendradarbiavimo socialinėje struktūroje modeliavimui[13]. Joje griežtai apibrėžiamos objektiškai orientuotos savybės surištos su autopažintine logika. Kalbos privalumai:

- Sudėtingų modelių modeliavimo paprastumas;

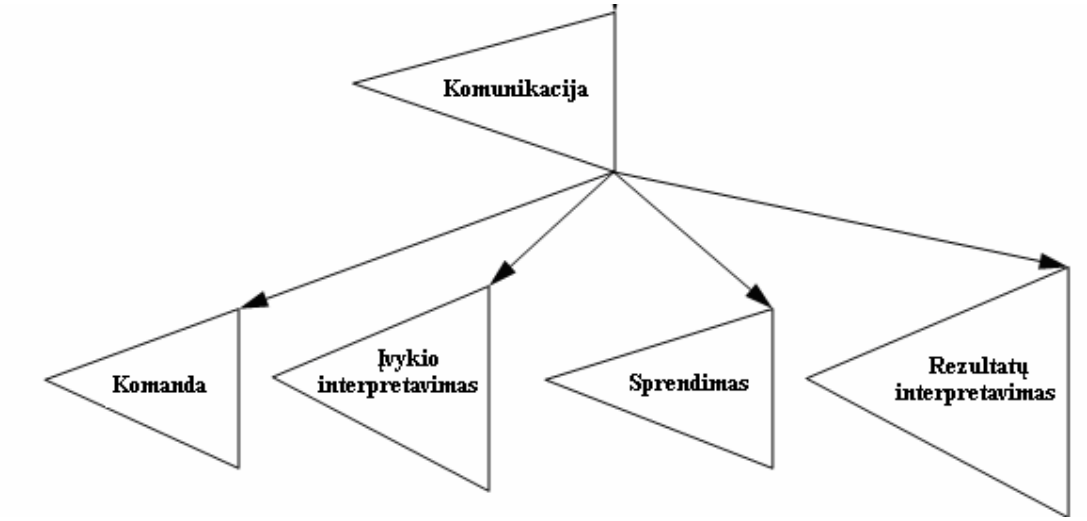
- Lankstus agentų modelių atvaizdavimas;
- Kodo pakartotinis panaudojimas (reuse).

SDLM bendradarbiavimo taisyklių pavyzdys:

```
time negotiatingRound ?round\
greater ?round 0\
notInferred
 last negotiatingRound (and
       informalActivitiesCache ?cache\
       at ?cache (activityComposition ?cActivity ?cooperators)\
       includes ?cooperators self)\
endorsementScheme ?endScheme ?endorsementSet ?base\
maxValue ?bestEndValue ?endValue
 (and
       endorsementFor ?activity possibleCollaborator ?arg\
       endorsementList ?activity ?endList\
       endorsementListValue ?endValue ?endList ?endScheme)\
time period ? period\
randomChoice ?choice ?activity [?period ?round]
 (and
       endorsementFor ?activity possibleCollaborator ?arg\
       endorsementList ?activity ?endList\
       endorsementListValue ?endValue ?endList ?endScheme)\
and
       cooperationOfferedTo ?choice\
       at ?choice (cooperationOfferedBy self)
```

SDLM komunikavimas 1.20 paveiksle susideda tokių veiksmų:

- komanda – siunčiama kitam agentui;
- įvykio interpretavimas – interpretuojamas gautas įvykis;
- sprendimas – priimamas sprendimas;
- rezultatų interpretavimas- įvykio prieš sprendimą palyginimas su gautu nusiuntus sprendimą.



1.20. pav. SDLM komunikavimo struktūra

1.13. Daugelio agentų sistemų modeliavimo kalbų palyginimas

Analizuotas modeliavimo kalbas palyginsime pagal tokius reikalavimus:

- vizualumas;
- standartas agentų modeliavimui;
- modeliavimo įrankių ir priemonių buvimas/nebuvimas;
- galimybė aprašyti agentų procesus, pranešimus, būsenų pokyčius;
- kodo generavimas į vizualius modelius ir modelių generavimas į programos aprašą;

Kiekvienas reikalavimas 3 lentelėje vertinimas 10 balų sistemoje, daugiausiai balų surinkusi modeliavimo kalba parinkta tolimesnei darbo eigai. Jei požymis visai netinkamas, rašomas 0, jei požymis yra vertinama nuo geriausio – 10, naudojant kiekvieną balą tik vienai modeliavimo kalbai toje kategorijoje.

3 lentelė. Agentų modeliavimo sistemų palyginimas

Pavadinimas	Vizuali	FIPA patvirtinta/pateikta patvirtinimui	Turi modeliavimo įrankius	Galimybė aprašyti procesus	Generavi-mas	Viso balų
AUML	10	10	8	10	10	58
Petri tinklai	8	0	10	8	8	34
AML	0	0	0	5	0	5
ADL	9	0	9	9	9	36
Q	0	0	0	3	0	3
MAML	0	0	0	4	0	4
SRC++	0	0	0	7	0	7
SDLM	0	0	0	6	0	6

1.14. Daugelio agentų sistemų analizės išvados

Pagal 3 lentelės parametrus buvo pasirinkta agentų unifikuota modeliavimo AUML kalba, jos privalumai modeliuojant daugelio agentų sistemas:

- Vizuali;
- Patogi ir lengvai suprantama;
- Iš vizualių modelių galima generuoti programos kodą;
- Taps agentų modeliavimo standartu, patvirtintu FIPA (Foundation for Intelligent Physical Agents)

Trūkumai:

- Nėra pilnai specifikuoti architektūros, agentų komunikacijos modeliai;
- Nėra įrankių kurie palaiko AUML, reikia išplėsti UML įrankius (MagicDraw, Visio).

2. AGENTŲ SISTEMŲ MODELIAVIMO METODIKA AGENTŲ UNIFIKUOTOS MODELIAVIMO KALBOS (AUML) PAGRINDU

Pagrindinė modeliavimo problema yra ta, kad yra daug modeliavimo kalbų, kurios akcentuoja skirtingas daugelio agentų sistemos modelio savybes ir nėra nė vienos, kuri tenkintų visas išvardintas savybes. Daugelis kalbų yra tekstinio pavidalo. Sunku pasirinkti tinkamą modeliavimo kalbą, nes trūksta reikiamos modeliavimo medžiagos bei modeliavimo metodikos.

2.1. Agentų sistemų modeliavimo metodikos tikslai ir uždaviniai

Modeliavimo metodikos tikslas –sudaryti analitikui pritaikytą agentų procesų specifikavimo metodiką, pagal kurią aprašytą agentų sistemą būtų galima transformuoti į programos aprašą.
Agentų sistemoms modeliuoti buvo pasirinkta unifikuota modeliavimo kalba (UML) su agentų išplėtimu (AUML) kuri palengvina kelią nuo modeliavimo iki realizavimo, ir modeliavimo procesą padaro prieinamą ir aiškų visiems agentų kūrimo dalyviams.

Metodikos uždaviniai:

- Sudaryti šablonus agentų sistemoms modeliuoti unifikuota modeliavimo kalba (UML) su agentų praplėtimu (AUML).
- Detaliai aprašyti modeliavimo metodiką pagal sukurtus šablonus;
- Pateikti modeliavimo pavyzdį pagal pateiktą metodiką ir šablonus.

Eksperimentiniam modeliavimui pasirinkta nano robotų sistema, nagrinėjama kaip daugelio agentų sistema, kurios agentai bendradarbiauja tarpusavyje ir bendrauja su aplinka. Pagal metodiką sukurtame eksperimentiniame modelyje panaudoti ir unifikuotos modeliavimo kalbos išplėtimo agentams schemos.

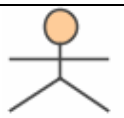
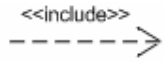
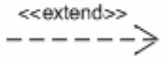
2.2. Šablonai vartotojui

Ankstesniuose skyriuose pateikti analizuoti agentų sistemų modeliavimo sistemos, pasiūlyta unifikuota modeliavimo kalba papildant agentų išplėtimu (AUML). 2.2 skyriuje remiantis įgyta patirtimi, sudaryti ir pateikti specifikacijų šablonai vartotojui, padėsiantys sukurti agentų sistemos modelius skirtus analitikui transformuoti į programos aprašą.

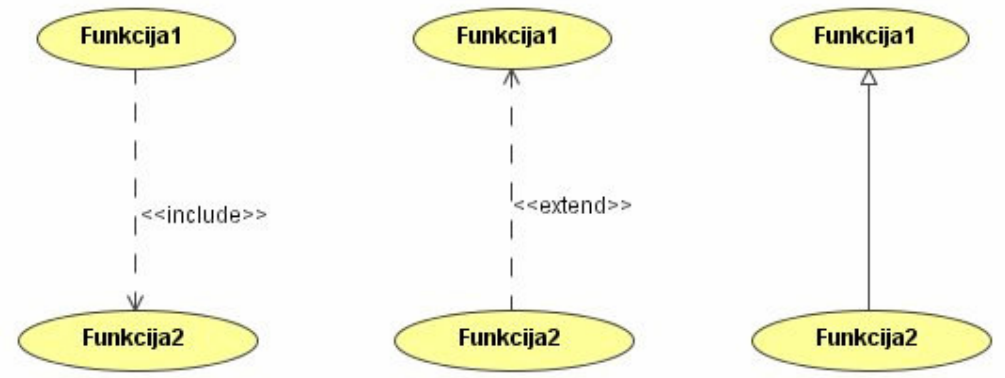
2.2.1. Funkcinis modelis.

Unifikuotos agentų modeliavimo kalbos funkcinis modelis kuriamas pagal hierarchiją kur aukščiausiai esantis agentas yra sprendžiantysis, o kiti agentai gali atlikti tik dalį operacijų, kurias atlieka sprendžiantysis.

4 lentelė. Funkcinio modelio elementai

Pavadinimas	Kaip braižomas	Ką reiškia
Aktorius		Kiekvienas aktorius aprašo skirtingą agento rolę
Vartojimo atvejis		Aprašo vieną agento funkciją
Ryšiai		
<<include>>		Aprašo funkciją kuri įtraukta į nurodytą vartojimo atvejį
<<extend>>		Aprašo funkciją, kuri išplečia nurodytą vartojimo atvejį
Apibendrinimas (generalization)		Apibendrina vieną vartojimo atvejį kitu, bendresniu

Aprašant didelio funkcionalumo agentų sistemas, funkcijas turinčias panašių požymių galima grupuoti į pakuotes (package), jei keli aktoriai naudoja tas pačias funkcijas, jų funkcinius modelius galima sujungti apibendrinimo ryšiu. Funkcinio modelio ryšių pavyzdys panaudojant lentelėje nurodytus elementus yra 2.1. paveiksle.

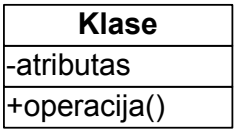


2.1 pav. funkcinio modelio ryšių pavyzdys

2.2.2. Statinis modelis.

Naudojant statinį modelį nurodomos modeliuojamų agentų klasės[18]. Objektinėje programoje klasės turi atributus (klasės narių kintamuosius), operacijas (klasės narių funkcijas) ir ryšius su kitomis klasėmis.

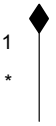
Klasės, turinčios vieną atributą ir vieną operaciją pavyzdys yra 2.2. paveiksle.



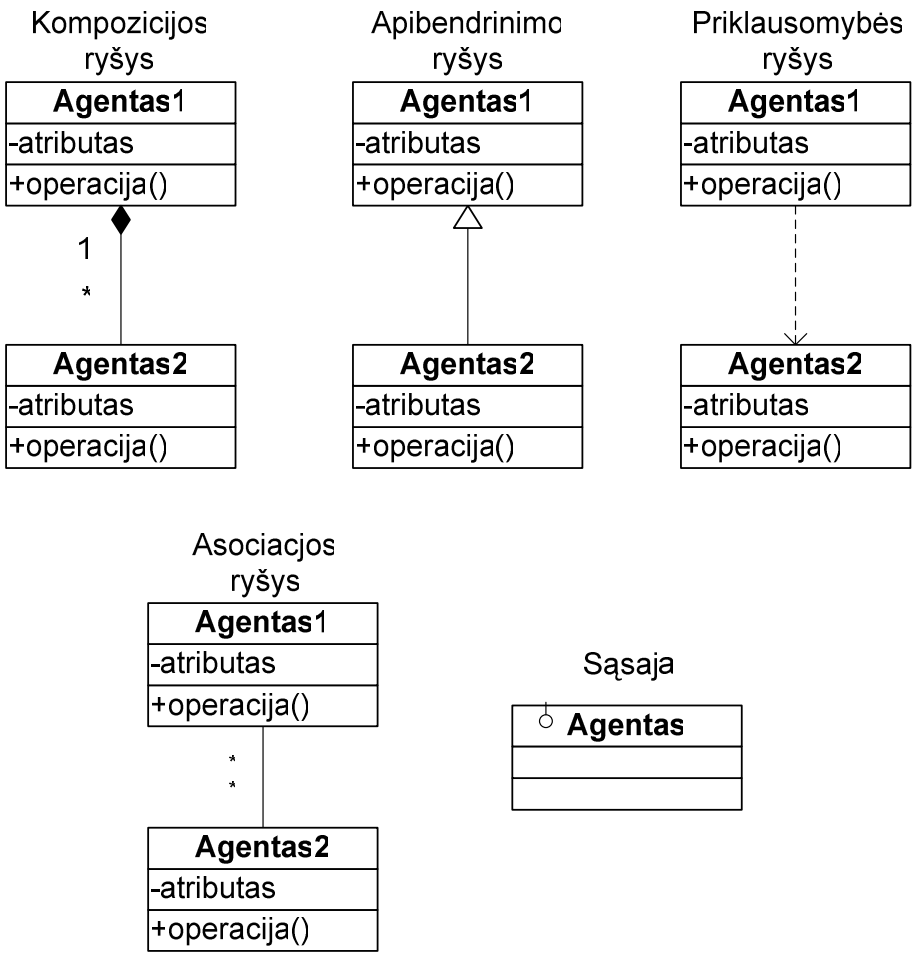
2.2 pav. Klasės pavyzdys

Unifikuotos agentų modeliavimo kalbos naudojami ryšiai aprašyti 5 lentelėje. Nurodomas ryšio pavadinimas, kaip jis braižomas ir ką reiškia statiniame agentų modelyje.

5 lentelė. Tarp klasių naudojami ryšiai

Pavadinimas	Kaip braižomas	Ką reiškia
Priklausomybė		Nurodo silpną ryšį tarp klasių
Kompozicija		Nurodo stiprią agregaciją ar ryšį
Paveldėjimas		Apibendrina kelias klases į vieną bendrą kuriai visos priklauso
Agregacija/asociacija		Nurodo silpną agregaciją tarp klasių
Sąsaja		Nurodo sąsają

2.3 paveiksle pateikti ryšių tarp klasių iš aukščiau esančios lentelės pavyzdžiai.



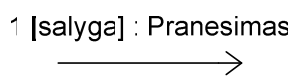
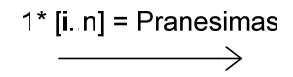
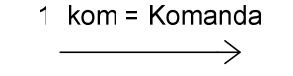
2.3 pav. Ryšių tarp klasių pavyzdžiai

2.2.3. Komunikavimo modelis

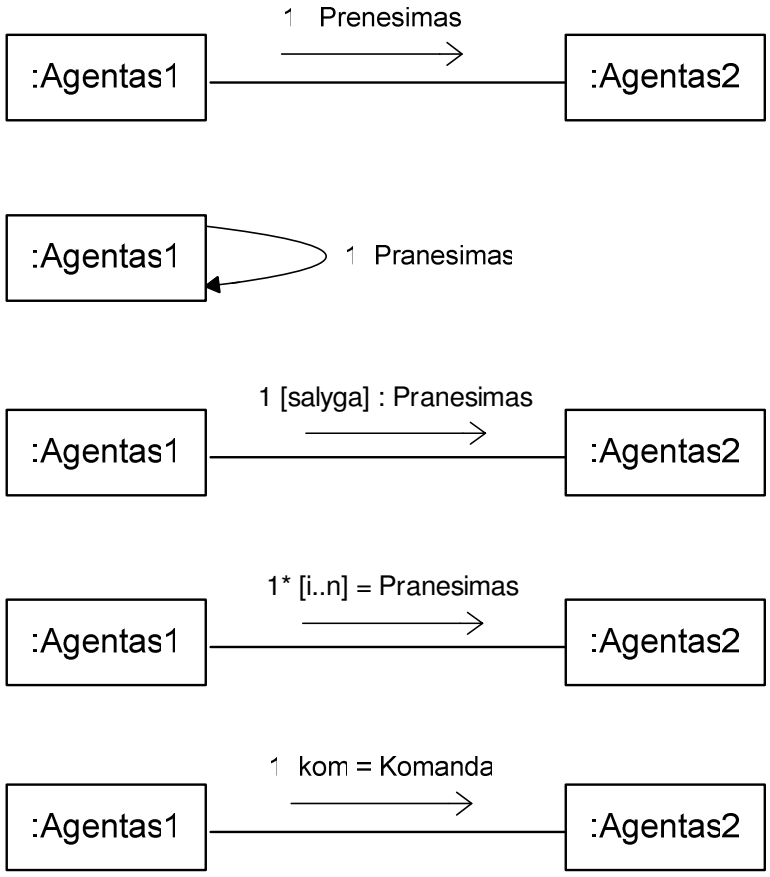
Komunikavimo modelis[19] yra dinaminis objektų modeliavimas, skirtas iliustruoti kaip objektai sąveikauja. Unifikuotos agentų modeliavimo kalbos komunikavimo modelio elementai aprašyti 6 lentelėje.

6 lentelė. Komunikavimo modelio elementai

Pavadinimas	Kaip braižomas	Ką reiškia
Objektas	:Objektas	Reiškia vieną iš objektų tarp kurių vyksta komunikacija
Grandis (link)		Sujungimas tarp dviejų objektų. Reiškia, kad objektai yra tarpusavyje susiję arba matomi
Pranešimas	1: Pranesimas	Pranešimas iš vieno objekto kitam

Pranešimas sau		Objekto pranešimas sau pačiam
Sąlyginis pranešimas		Pranešimas siunčiamas tik jei tenkinama nurodyta sąlyga
Iteracija/ciklas		Pranešimas siunčiamas visiems objekto nariams
Pranešimas su statiniu metodu		Pranešimas įvykdo statinio metodą

Pastaba. Per vieną grandį gali eiti keli pranešimai ir net abipusiai pranešimai, nes visi pranešimai tarp dviejų objektų siunčiami ta pačia linija kaip dvikrypčiu greitkeliu. 2.4. paveiksle pateikti pranešimų pavyzdžiai.

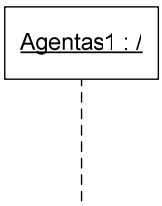
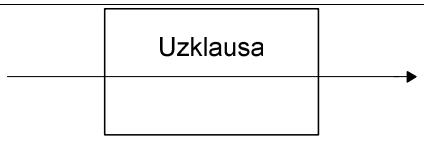
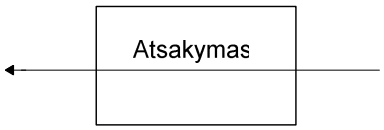
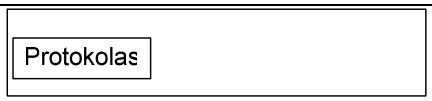


2.4 pav. Komunikacijos modelio pranešimų pavyzdys

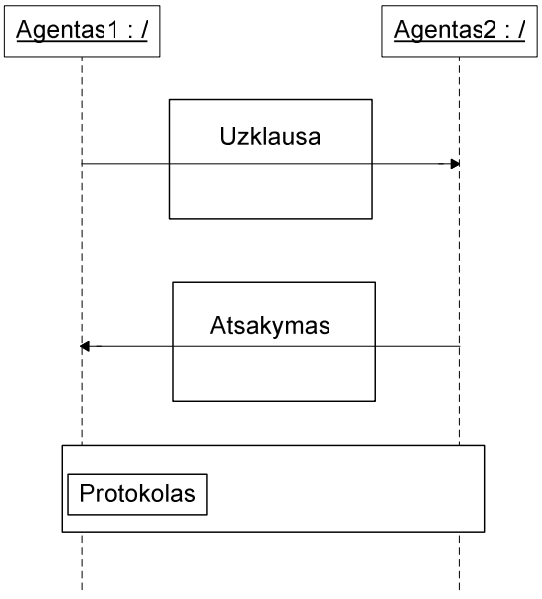
2.2.4. Sąveikos modelis

Sąveikos diagrama yra agentų išplėtimas unifikuotai modeliavimo kalbai parodantis sąveiką taip skirtingų vaidmenų agentų jiems „susikalbėti“ skirtu protokolu. Sąveikos modelio elementai aprašyti 7 lentelėje.

7 lentelė. Sąveikos modelio elementai

Pavadinimas	Kaip braižomas	Ką reiškia
Agentas		Reiškia vieną iš agentų tarp kurių vyksta sąveika
Užklausa		Vieno agento užklausa kitam
Atsakymas		Agento atsakymas į pirmojo nusiųstą užklausą
Protokolas		Nurodo kokiu protokolu sąveikauja agentai

2.5. paveiksle pateikiamas sąveikos modelio pavyzdys.



2.5 pav. Sąveikos modelio pavyzdys

2.3. Išvados

Šiame skyriuje aprašyti ir paaiškinti daugelio agentų sistemų modelio šablonai:

- Funkciniam modeliui;
- Statiniam modeliui;
- Komunikavimo modeliui;
- Sąveikos modeliui;

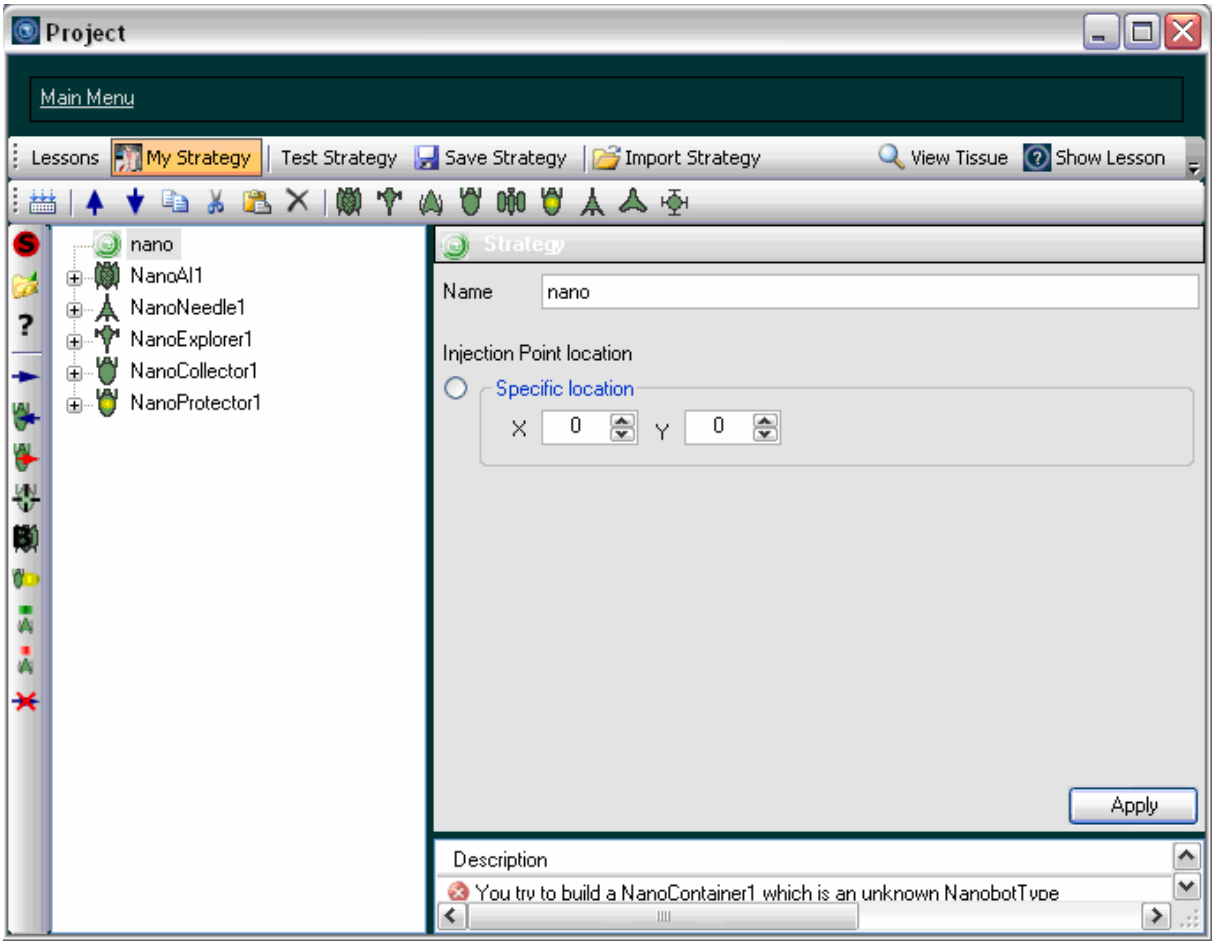
Remdamasis aprašyta metodika vartotojas galės grafiškai aprašyti daugelio agentų sistemą būdingais modeliais taip palengvindamas modelio suvokimą analitikui, bei tolimesnio daugelio agentų sistemos kūrimo eigą programuotojui.

3. EKSPERIMENTAS

Eksperimento dalyje pasiūlyta modeliavimo metodika pritaikoma UML įrankiui. Darbe analizuoti modeliai kuriami su MagicDraw įrankiu įvedant papildomus stereotipus. Unifikuotos modeliavimo kalbos modelius pasiūlyta papildyti agentų išplėtimu AUML. Tokiu būdu sudarytas modelis detaliau paaiškinamas, jį lengviau galima konvertuoti į programos aprašą.

3.1. Daugelio agentų sistemos modelis naudojant siūlomą metodiką

Daugelio agentų sistemai modeliuoti pasirinkta nano robotų sistema, nagrinėjama su programiniu paketu PH2007[16], kurios pavyzdį matome 3.1. paveiksle.



3.1. pav. Programinio paketo PH2007 pavyzdys

Sistema turi kelis nano robotų tipus, galinčius bendradarbiauti tarpusavyje. Nano robotai patekę į žmogaus organizmą suranda ligos židinį jį sunaikina bei atlikę užduotį pasišalina iš organizmo. Jų tipai aprašyti 8, o funkcijos 9 lentelėje. Naudojami angliški pavadinimai, kad aprašius sistemą grafiniais modeliais jų programos aprašas atitiktų paketo PH2007 aprašus.

8 lentelė. Nano robotų tipai

Pavadinimas	Tipas	Funkcijos
NanoAI	Sprendimo agentas	„Gamina“ kitus nano robotus, valdo jų veiksmus ir atlieka sprendimus
NanoExplorer	Mobilus agentas	Skirtas ligos židinio ir antikūnių paieškai
NanoProtector	Mobilus agentas	Skirtas kitų nano robotų apsaugai nuo antikūnių
NanoColector	Mobilus agentas	Skirtas medžiagų pergabenimui iš „gydomųjų taškų“ į nano adatas
NanoNeedle	Stacionarus agentas	Skirtas gydomųjų medžiagų suleidimui į ligos židinius

9 lentelė. Pagrindinės nano robotų funkcijos

Funkcija	Ką ji atlieka	Nano robotai turinys šią funkciją
Build	Nano robotų „gamyba“	NanoAI
Scan	Aplinkos žvalgyba	NanoAI, NanoCollector, NanoProtector, NanoExplorer, NanoNeedle
MoveTo	Judėjimas į nurodytas koordinates	NanoAI, NanoCollector, NanoProtector, NanoExplorer.
CollectFrom	Medžiagos pakrovimas į nano robotą	NanoCollector
TransferTo	Medžiagos iškrovimas į nano adatą	NanoCollector
DefendTo	Gynyba nuo antikūnių	NanoProtector, NanoNeedle
StopMoving	Sustoti nepasiekus nurodyto tikslo	NanoAI, NanoCollector, NanoProtector, NanoExplorer
ForceAutoDestruction	Nano roboto susinaikinimas	NanoAI, NanoCollector, NanoProtector, NanoExplorer, NanoNeedle

Pastaba. Sunaikinus NanoAI robotą kiti robotai tampa nevaldomi ir nebevykdo jokių veiksmų.

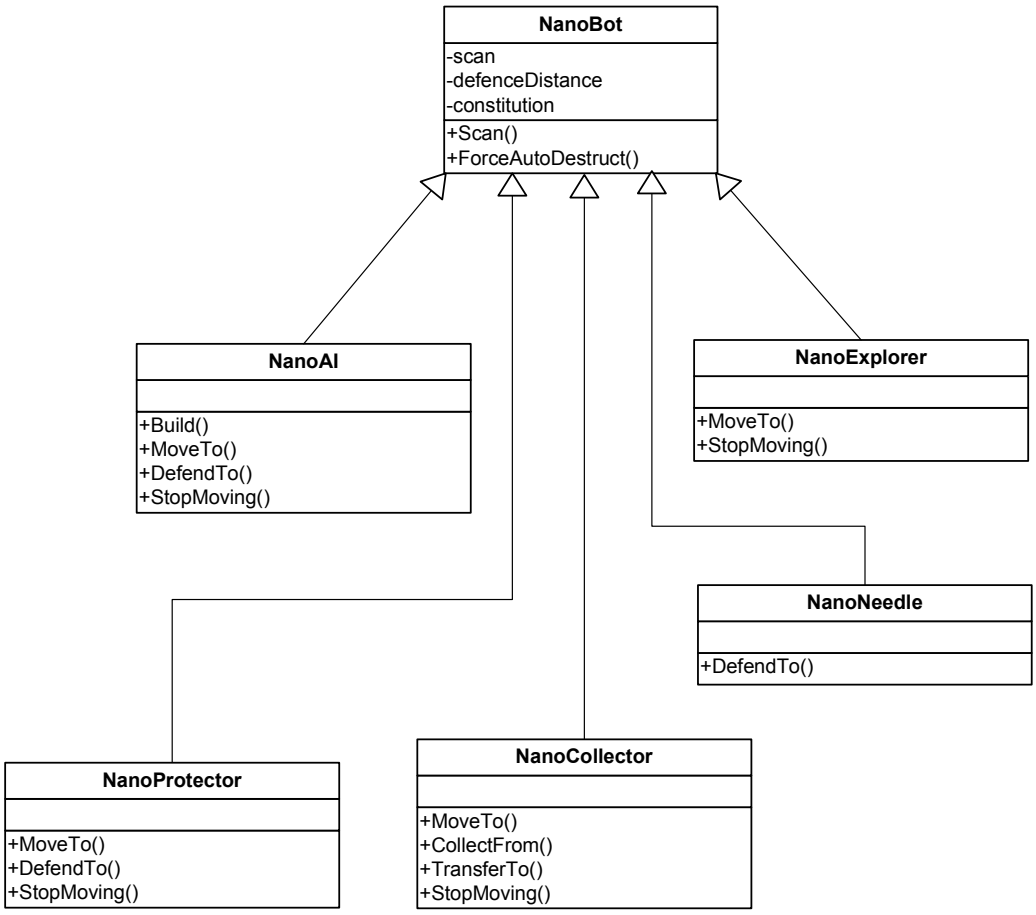
3.2. Daugelio agentų sistemos funkcinis modelis

Naudojantis lentelėse pateiktais parametrais, sudaromas funkcinis modelis, kurio aktoriai yra NanoAI, NanoExplorer, NanoDefender, NanoCollector ir NanoNeedle. 3.1. paveiksle pateiktas funkcinis modelis, tačiau jei agentų sistema sudėtinga, patartina išskaidyti aktorius į atskirus funkcinius modelius, kad modelis būtų lengviau suprantamas pvz. 3.2. paveiksle.

3.2. pav. Eksperimentinis daugelio agentų sistemos funkcinis modelis

3.3. Daugelio agentų sistemos statinis modelis

Pagrindinė klasė NanoBot, kuriai priklauso visų tipų agentai. Čia saugomi kintamieji apie nano roboto apžvalgos atstumą (scan), gynybos atstumą (defenceDistance), gyvybingumą (constitution). Pagrindinės funkcijos kurios yra pas kiekvieną nano robotą: Scan() ir ForceAutoDestruct(). Kitų klasių pavadinimai atitinka nano robotų pavadinimus: NanoAI, NanoExplorer, NanoDefender, NanoCollector ir NanoNeedle.

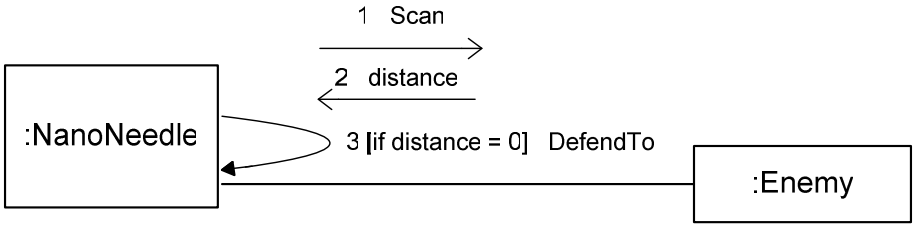


3.3. Daugelio agentų sistemos statinis modelis

Nano agentų klasės 3.3. paveiksle sujungtos su NanoBot klase apibendrinančiu ryšiu, kadangi NanoBot klasė sujungia bendras nano robotų savybes, juos apibendrindama. Atributai šiuo atveju nenurodomi, kadangi jie agentų sistemai nėra reikšmingi.

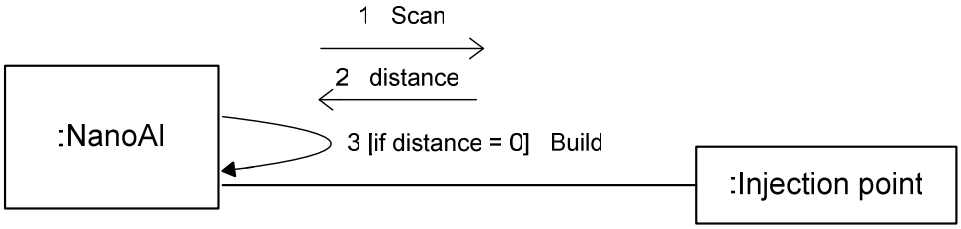
3.4. Daugelio agentų sistemos komunikavimo modelis

Šioje daugelio agentų sistemoje komunikacija vyksta tarp nano robotų ir antikūnių. Jei į nano roboto matymo lauką Scan pakliūna antikūnis nano robotas ginasi nuo jo su DefendTo funkcija. Kadangi gynybos funkciją turi keletas nano robotų tipų bus nagrinėjamas pavyzdys tik su NanoNeedle roboto tipu.



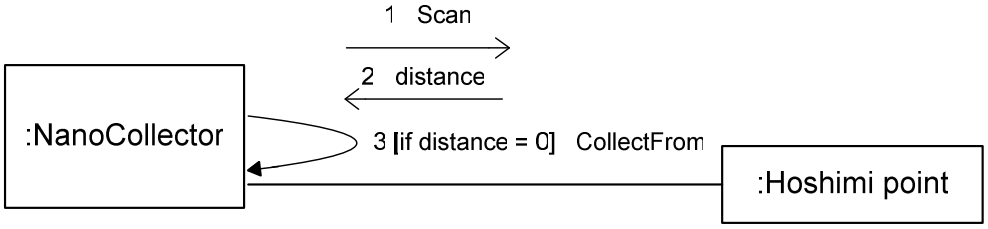
3.4. pav. Nano roboto komunikacijos modelis su antikūniu

Jei aptinkamas ligos židinys, nano robotas NanoAI jame „pagamina“ nano robotą NanoNeedle. Ligos židinys vadinamas injekcijos tašku (injection point).



3.5. pav. Nano roboto komunikacijos modelis su injekcijos tašku

Jei aptinkamas medžiagos taškas (hoshimi point), nano robotas NanoClollector pasikrauna medžiagą ir gabena į nano adatą NanoNeedle.

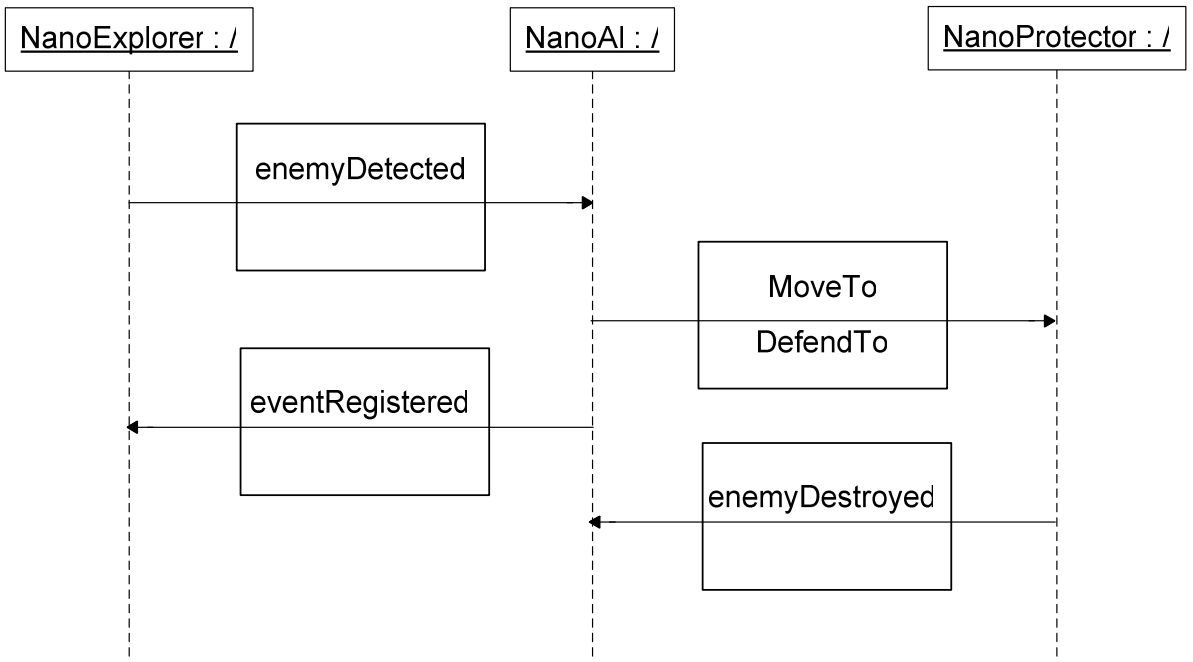


3.6. pav. Nano roboto komunikacijos modelis su medžiagos tašku

3.5. Daugelio agentų sistemos sąveikos modelis

Agentų sąveika vyksta per sprendimų agentą NanoAI. NanoAI gauna pranešimus iš kitų nano robotų, nusprendžia ką daryti ir siunčia komandą nano robotui, kuris tą komandą įvykdo. Kadangi visi agentai yra tos pačios klasės NanoBot apibendrinimas, todėl sąveikos diagramoje jų protokolas nebus nurodomas.

3.7. paveiksle nano robotas žvalgas NanoExplorer pastebėjęs antikūnį, praneša sprendimų agentui NanoAI ir šis duoda komandą nano robotui NanoDefender apsiginti.

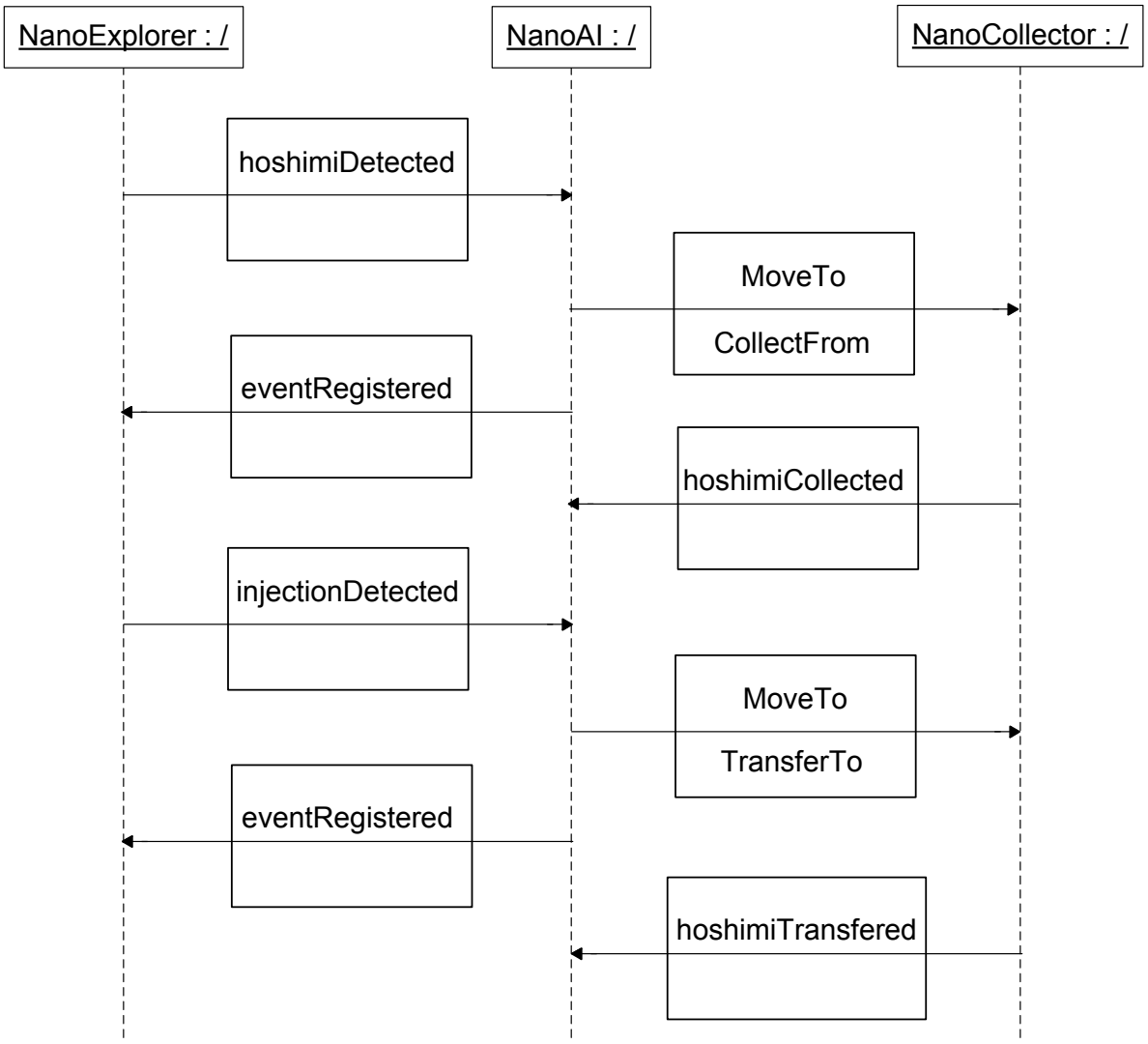


3

3.7. pav. Nano robotų sąveikos modelis ginantis

Šiuo atveju įvykiai (events) yra enemyDetected aptikus priešą, eventRegistered sprendimų agentui priėmus pranešimą ir enemyDestroyed gynybiniam agentui NanoDefender įvykdžius komandą DefendTo.

3.8. paveiksle nano robotas žvalgas NanoExplorer pastebėjęs medžiagos tašką praneša sprendimų agentui NanoAI, kuris duoda komandą pasikrauti medžiagos nano robotui NanoCollector. Nano žvalgui aptikus injekcijos tašką ir apie tai pranešus nano robotui NanoAI, šis siunčia komandą nano robotui NanoCollector iškrauti medžiagą į nano adatą NanoNeedle.



3.8. pav. Nano robotų sąveikos modelis transportuojant

Šiuo atveju įvykiai (events) yra hoshimiDetected aptikus medžiagą, eventRegistered sprendimų agentui priėmus pranešimą, hoshimiCollected nano robotui NanoCollector įvykdžius komandą CollectForm (paėmus medžiagą) ir hoshimiTransfered NanoCollector įvykdžius komandą TransferTo.

3.6. Eksperimento išvados

Eksperimentui pasirinktas nano robotų daugelio agentų sistemą aprašantis PH2007 SDK modulio veikimo scenarijus. Pagal sudarytą agentų sistemos aprašymą, panaudojant 2 skyriuje pasiūlytą metodiką, sukurtas vizualus daugelio agentų sistemos aprašymas analitikui, kuris jį galės analizuoti ir sugeneruoti programos kodo aprašą.

IŠVADOS

1. Šiame darbe buvo apibrėžti pagrindiniai daugelio agentų sistemų tipai, nurodytos intelektualių agentų charakteristikos, analizuota daugelio agentų sistemų klasifikacija, palyginti stacionarūs bei mobilieji agentai, jų komunikavimo skirtumai.

2. Atliktas modeliavimo kalbų palyginimas, atskleidžiant kiekvienos privalumus ir trūkumus, bei parodant, kad universalios modeliavimo metodikos, kuri tenkintų daugelį reikalavimų, nėra. Kaip pagal daugiausiai parametrų tinkanti modeliavimo sistema parinkta unifikuota modeliavimo kalba UML su agentų išplėtimu AUML.

3. Pasirinktąja unifikuota modeliavimo kalba UML su agentų išplėtimu AUML, aprašyta funkcinių modelių, statinių modelių, komunikavimo modelių ir sąveikos modelių metodika daugelio agentų sistemų modeliavimui. Modeliavimo įrankis MagicDraw pritaikytas agentų išplėtimo stereotipams, sukurti unifikuotos modeliavimo kalbos UML su agentų išplėtimu AUML šablonai vartotojui.

4. Eksperimentui pasirinktas nano robotų daugelio agentų sistemą aprašantis PH2007 SDK modulio veikimo scenarijus. Pagal sudarytą agentų sistemos aprašymą, panaudojant 2 skyriuje pasiūlytą metodiką, sukurtas vizualus daugelio agentų sistemos aprašymas analitikui, kuris jį galės analizuoti ir sugeneruoti programos kodo aprašą.

5. Kadangi daugelio agentų išplėtimas AUML unifikuotai modeliavimo kalbai UML nėra baigtas specifikuoti, tolimesnio darbo tikslas išplėsti metodiką papildomai atsiradusiais daugelio agentų sistemų modeliais ir pritaikyti juos eksperimentiniame modeliavime.

TERMINŲ IR SANTRUMPŲ ŽODYNAS

UML (*Unified Modeling Language*)	Unifikuota modeliavimo kalba.
AUML (*Agent Unified Modeling Language*)	Unifikuota modeliavimo kalba su agentų išplėtimu.
CASE (*Computer Aided Software Engineering*)	Kompiuterizuotas programinis įrankis
RPC *(Remote Procedure Call)*	Nutolęs procedūros iškvietimas
RP (Remote Programming)	Nutolęs programavimas
Blackboard	Skelbimų lenta
CPL *(Cooperative Planning Layer)*	Bendradarbiaujančio planavimo sluoksnis
LPL *(Local Planning Layer)*	Vietinio planavimo sluoksnis
BBL *(Behavior Based Layer)*	Elgesio planavimo sluoksnis
AML *(Agent Markup Language)*	Agentų sužymėjimo kalba.
ADL *(Agent Description Language)*	Agento aprašymo kalba.
MAML *(Multi-Agent Modeling Language)*	Daugelio agentų modeliavimo kalba.
SCR++ *(A Language Design for Real-Time Multi-Agent Systems)*	Išplečiama kalba daugelio agentų sistemoms
SDLM (Multi Agent Organization Modeling Language)	Daugelio agentų organizacijos modeliavimo kalba
API (*Application Program Interface*)	Plečiama (lengvai pritaikoma) programos sąsaja.

LITERATŪRA

[1] ASAP theme: Multi agents [interaktyvus], [žiūrėta 2007 03 21], prieiga per internetą: http://www.asap.cs.nott.ac.uk/themes/ma.shtml

[2] Intelligent software agents: foundations and applications, Walter Brenner, Rudiger Zarnekow, Hartmut Witting in co-operation with Claudia Schubert,1998 Berlin;

[3] Agent modeling, Gal A. Kaminka [interaktyvus], [žiūrėta 2007 03 21], prieiga per internetą: http://www.cs.biu.ac.il/~galk/Research/agentmod.html

[4] AML Syntax description, Dr. Wolf Garbe, [interaktyvus], [žiūrėta 2006 06 19], www.powermarkets.com/;

[5] Exploiting UML in the Design of Multi-Agent Systems, Federico Bergenti and Agostino Poggi, [interaktyvus], [žiūrėta 2006 06 19], www.auml.org;

[6] Intelligent software agents: foundations and applications, Walter Brenner, Rudiger Zarnekow, Hartmut Witting in co-operation with Claudia Schubert,1998 Berlin;

[7] JADL-an Agent Description Language for Smart Agents, Thomas Konnerth, Benjamin Hirsch, and Sahin Albayrak, [interaktyvus], [žiūrėta 2006 06 19], http://staff.science.uva.nl/~ulle/DALT-2006//;

[8] Multi-agent systems, Talukdar, S., [interaktyvus], [žiūrėta 2006 06 19], www.ieeexplore.ieee.org;

[9] Referance to Vijay Halaharvi and Gopal Gupta, "SCR++: An Extensible Language for Multi-Agent Systems";

[10] Towards an agent architectural description language for information systems, Stéphane Faulkner, Manuel Kolp, [interaktyvus], [žiūrėta 2006 06 19], www.**ieee**xplore.**ieee**.org;

[11] Towards a Standardization of Multi-Agent System, Roberto A. Flores-Mendez, [inyteraktyvus], [žiūrėta 2006 06 19], http://www.acm.org/crossroads/;

[12] Q: A Scenario Description Language for Interactive Agents, Toru Ishida, [interaktyvus], [žiūrėta 2006 06 19], www.ieeexplore.ieee.org;

[13] SDML: A Multi-Agent Language for Organizational Modelling, Scott Moss, Helen Gaylard, Steve Wallis and Bruce Edmonds, [interaktyvus], [žiūrėta 2006 06 19], http://cfpm.org/cpmrep16.html;

[14] Multi agent systems, Katia Sycara, [interaktyvus], [žiūrėta 2007 03 28], http://www.aaai.org/AITopics/html/multi.html;

[15] Multi-Agent Modeling Language, László Gulyás, Tamás Kozsik, Sándor Fazekas, [interaktyvus], [žiūrėta 2007 04 22], http://www.maml.hu/maml/initiative/index.html;

[16] PH2007 SDK Reference, Richard Clark, [inyteraktyvus], [žiūrėta 2007 04 22], http://www.project-hoshimi.com/Reference/index.html;

[17] Software Agents: An Overview, Hyacinth S. Nwana [interaktyvus], [žiūrėta 2006 01 07]. Prieiga per internetą:

[18] Unified Modeling Language: Superstructure version 2.1.1. Object Management Group. 2007-02-03, [interaktyvus], [žiūrėta 2007 03 27]. Prieiga per internetą: http://www.omg.org/cgi-bin/apps/doc?formal/07-02-03.pdf.

[19] UML 2 Communication Diagrams, Scott W. Ambler, 2006, [interaktyvus] Object Management Group. 2007-02-03, [žiūrėta 2007 05 12]. Prieiga per internetą: http://www.agilemodeling.com/artifacts/communicationDiagram.htm