VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Mobiliosios komercijos agentai WAP paslaugose

Mobile commerce agents in WAP services

Magistro baigiamasis darbas

Atliko: Vytautas Gaidys
(parašas)

Darbo vadovas: Doc. dr. Saulius Minkevičius
(parašas)

Recenzentas: Doc. dr. Valdas Undzėnas
(parašas)

Vilnius
2011

Santrauka

Problema – daug agentų technologiją nagrinėjančių straipsnių ir kitų rašto darbų, tačiau mažai sukurtų realių programų, naudojančių ją. Neaiškus agentų technologijos panaudojimas ir pritaikymas mobiliojoje komercijoje, nedidelis susidomėjimas iš verslo pusės. Neaiški padėtis agentų platformos rinkoje.

Darbo tikslas – agentų technologijos panaudojimo mobiliosios komercijos kontekste analizė.

Spręsti uždaviniai:

- agentų platformų kokybinę ir kiekybinę analizė;
- mobiliosios komercijos sistemos modelio sukūrimas;
- modelio realizavimas suprogramuojant sistemą.

Darbo metodai – agentų platformų kokybinė (literatūros analizė) ir kiekybinė analizė (atlikti programiniai testai), prototipo realizavimas.

Išvados. Mobiliosios komercijos sistemos prototipu, sukurtu naudojant agentų technologiją, parodyta, kad pati agentų technologija yra gyvybinga ir jos atsisakyta per anksti. Sukurtas sistemos prototipas gali būti panaudotas praktikoje ir atnešti naudą.

Raktiniai žodžiai: agentų technologija, mobilioji komercija, WAP paslaugos.

Summary

Main problem – there is a lot of work done which studies agent technology. But taken in comparison, there is much less real software created using this technology. Not obvious use of agent technology leads to a little or even no attention from business. Furthermore, not clear situation in agent platforms market reduces it drastically.

Work objective – analysis of agent technology use in mobile commerce context.

Solved problems:

- Qualitative and quantative analysis of agent platforms.
- Creation of mobile commerce system model.
- Model implementation.

Conclusion. With creation of mobile commerce system prototype, which uses agent technology, it is shown that agent technology is viable and this technology should not be rejected to early. Created system prototype can be used in practise. One can benefit from using it.

Keywords: agent technology, mobile commerce, WAP service.

Turinys

ĮVADAS

Tobulėjant mobiliosioms technologijoms, atsiranda nauji mobiliųjų prietaisų pritaikymo būdai, kurie gali palengvinti mūsų užduočių atlikimą. Savo nišą komercijoje turi ir mobiliosios technologijos. Komercijos rūšis, kurioje parduodant ar teikiant paslaugas naudojamos mobiliosios technologijos, vadinama mobiliąja komercija (kai kur toliau naudojama santrumpa m-komercija). Mobilioji komercija apima įvairias mokslo sritis, kaip duomenų gavyba, saugumas ir kt.

Su naujų mobiliųjų technologijų atsiradimu ir vystymusi pradedamos teikti ir naujos mobiliosios komercijos paslaugos. Jos teikiamos ne tik pokalbių ar trumpųjų žinučių pagalba, bet ir panaudojant bevielį ryšį ar Bluetooth. Naudojant naujausias technologijas mobilios komercijos sistemose, reikia ir sudėtingesnės programinės įrangos, kuri gebėtų išnaudoti technologijos teikiamus privalumus ir tinkamai vykdyti užduotis. Kyla tokie reikalavimai programinei įrangai, kad ji mokėtų prisitaikyti prie aplinkos ar vartotojų veiksmų, turėtų dirbtinio intelekto elementų.

Mobilieji agentai – programinės įrangos vienetas, veikiantis mobilioje aplinkoje ir reaguojantis į ją ir/arba vartotojo veiksmus. Mobilieji agentai geba veikti autonomiškai, turi galimybę migruoti iš vieno mobiliojo įrenginio į kitą, dinamiškai prisitaikyti prie aplinkos ir vartotojo. Šios mobiliųjų agentų savybės „padengia“ mobiliosios komercijos reikalavimus programinei įrangai, todėl mobiliosios komercijos agentai tinka mobiliajai komercijai įgyvendinti. Mobilieji agentai yra primiršta, tačiau vis dar plėtojama technologija, glaudžiai susijusi su paskirstytomis sistemomis, informacijos gavyba, žiniatinkliu, elektronine komercija ir dirbtiniu intelektu. Mobilusis agentas yra autonominė esybė, kuri gali heterogeniniuose tinkluose migruoti iš vieno įrenginio į kitą. Agentas gali sustabdyti savo veiklą bet kuriame taške, migruoti į kitą įrenginį ir ten pratęsti savo darbą. Mobiliųjų agentų metodologija nukrypsta nuo įprasto serverio/kliento architektūros ir iškelia naują paradigmą paskirstytose sistemose, kurioje agentai yra autonominiai.

Aktualumas ir naujumas. Šių dienų realybė – tai masinis mobiliųjų telefonų naudojimas, kasdienių užduočių atlikimas mobilaus telefono ar delninuko pagalba, agentų technologijos populiarumo didėjimas. Todėl agentų technologija yra viena iš labiausiai dinamiškų ir perspektyvių sričių moksliniams tyrimams ir plėtrai informacinėse technologijose. Multiagentų ir mobiliųjų agentų technologijos daro didelį poveikį

programinės įrangos kūrime beveik visais aspektais ne su skaičiavimais susijusiose disciplinose. Ši technologija lemia didelius informacinės visuomenės pokyčius. Tai yra viena iš sėkmingiausių verslo pasaulio programų, tiesiogiai sprendžianti pastarojo problemas. [LK02]

Virtualios organizacijos ir mobilioji komercija tapo „madinga“ praktika šiandienos pasaulio įmonėse. Programiniai agentai veikia kaip esminis interneto, kaip plataus informacijos šaltinio, komercijos sistemų, kurios šiuo metu kuriamos visame pasaulyje, multiagentų ir mobiliųjų agentų tarpusavio sąveikos komponentas. Jie suteikia tinkamą ir efektyvų karkasą šiuolaikiniam verslui. Bevielio ryšio ir mobiliojo ryšio technologija sulaukė daug dėmesio per pastarąjį dešimtmetį. Visų pirma, belaidžiai tinklai, Bluetooth tinklais, ir bevieliai mišrieji tinklai numanomi kaip naujos kartos mobiliojo ryšio sistemos. Praeitame dešimtmetyje išaugo interneto paslaugų tyrimai [Hun08]. Sistemų tyrimai yra atliekami ne tik akademinėse ir mokslinių tyrimų bendruomenėse, tačiau ir pramonės srityje.

Kai vartotojas nori pasinaudoti mobiliosios komercijos paslaugomis, mobiliojo mokėjimo procesas turi būti patikimas, saugus ir greitas. Mobilieji mokėjimai yra greitas ir patogus būdas vartotojui apsimokėti sąskaitas ar atsiskaityti už paslaugas, todėl m-komercija tampa vis populiaresnė [Che07]. Perėjimas nuo fizinio atsiskaitymo metodo, kai atsiskaitoma grynaisiais, prie virtualaus, kai atsiskaitoma pervedimais, davė daug naudos vartotojams ir rinkos dalyviams [CA04]. Mobiliųjų mokėjimų metodai šiuo metu yra labai įvairūs ir šioje srityje jaučiamas standartinių paslaugų trūkumas visame bankininkystės mokėjimų sektoriuje.

Nevienalyčių m-komercijos mokėjimo sistemų vienas iš trūkumų yra bendradarbiavimo stoka tarp įvairių dalyvių, įskaitant bankus, kredito kortelių įmones, mobiliojo ryšio operatorius ir nepriklausomus dalyvius [Li05]. Vis daugiau mobiliųjų telefonų naudotojų tikisi pasinaudoti įvairiomis paslaugos m-komercijos mokėjimo sistemomis.

Problema. Yra daug agentų technologiją nagrinėjančių straipsnių ir kitų rašto darbų, tačiau mažai yra sukurtų realių programų, naudojančių ją. Neaiškus agentų technologijos panaudojimas ir pritaikymas mobiliojoje komercijoje, nedidelis susidomėjimas ja iš verslo pusės. Neaiški padėtis agentų platformų rinkoje.

Darbo objektas – agentų platformos.

Tam, kad agentas galėtų egzistuoti ir vykdyti savo darbą, jam reikalinga tinkama programinė aplinka, kurioje agentas galėtų veikti ir įgyvendinti jam priskirtas užduotis. Agentų platformos pateikia tokią programinę aplinką, kurioje agentai yra sukuriami ir gali vykdyti savo užduotis. Mobiliojo agento programinė aplinka yra paskirstyta sistema, veikianti tinkle heterogeninio tinklo kompiuteriuose. Pradinis reikalavimas mobiliojo agento aplinkai

yra pateikti priemonių rinkinį, kuriuo naudojantis mobilieji agentai gali atlikti reikiamą darbą, migruoti. Taip pat programinė aplinka gali pateikti pagalbinius servisus, kurių pagalba galima pasiekti kitas mobiliųjų agentų sistemas ir jų teikiamus servisus. [Pet05]

Agentų technologija suprantama kaip sistemų struktūrizavimo, projektavimo ir kūrimo paradigma, kuri reikalauja sudėtingos sąveikos tarp autonominių paskirstytų komponentų. The Foundation for Intelligent Physical Agents, sutrumpintai - FIPA fondas (www.fipa.org), yra bene didžiausia tarptautinė organizacija, kuri nagrinėja ir kuria standartus interaktyvioms programinės įrangos sistemoms, paremtoms agentų paradigma [KUU02].

Hipotezė – agentų technologija yra gaji, bet be reikalo pamiršta.

Darbo tikslas – agentų technologijos panaudojimo galimybių mobiliojo komerjecijoje analizė ir tos technologijos gyvybingumo parodymas.

Uždaviniai:

- atlikti agentų platformų kokybinę ir kiekybinę analizę;
- sukurti mobiliosios komercijos sistemos modelį;
- realizuoti sistemos prototipą naudojant agentų technologijas ir parodyti, kad tai yra patogu, šios technologijos yra be reikalo pamirštos, jos turi savo nišą mobiliojoje komercijoje.

Darbo metodai – agentų platformų kokybinė (literatūros analizė) ir kiekybinė analizė (atlikti programiniai testai), sukurto prototipo realizavimas.

Darbo dalys – agentų analizė, agentų platformų analizė, agentus naudojančios sistemos prototipo sukūrimas.

Agentų tarpusavio komunikacija yra esminis raktas, realizuojantis agentų paradigmą, kaip kad žmonių kalba yra tarpusavio komunikavimo įrankis, greito vystymosi priežastis. Gerai ir tiksliai apibrėžta komunikavimo architektūra yra privalomas sėkmingo ir plataus mobiliųjų agentų technologijos panaudojimo pagrindas. Mobiliųjų agentų technologijos tyrimai iškėlė naujų komunikavimo protokolų kūrimos ar dialogo formų poreikį. Taip pat buvo pateiktos mobiliųjų agentų aplinkų tarpusavio komunikavimo koncepcijos ir atviri komunikavimo karkasai mobiliesiems programiniams agentams. Esamos mobiliųjų agentų sistemos naudoja komunikavimo mechanizmus, tokius kaip pranešimai, vietiniai ar išoriniai procedūrų kvietimai, tačiau negalime žinoti kiekvieno karkaso komunikavimo tipų, kad tinkamai vyktų bendravimas. Tačiau agentų gebėjimas migruoti, nenutraukiant aktyvaus komunikavimo, nėra tinkamai apibrėžtas visuose komunikavimo mechanizmuose. [SC98].

Šiame baigiamajame magistro darbe aprašomos ištirtos agentų platformos ir pateikiama apibendrinta platformų savybių lentelė. Pasirinktoms dviems agentų platformoms yra

suprogramuoti testai ir įvertinta platformų sparta. Darbe pateikiamas kuriamos mobiliosios komercijos agentų sistemos modelis, vertinami alternatyvūs modeliai ir pagrindžiamas pasirinktas modelis. Aprašoma ir vertinama mobiliosios komercijos agentų sistema sukurta, remiantis pateiktu modeliu.

Praktinis rezultatų reikšmingumas. Atlikta agentų platformų analize parodyta, kurios platformos yra tinkamiausios mobiliosios komercijos sistemų, naudojančių agentų technologiją ir teikiančių WAP paslaugą, kūrimui. Sukurtu mobiliosios komercijos sistemos prototipu parodytas agentų technologijos gyvybingumas.

1. Susiję darbai, jų apžvalga

Apie du tūkstantuosius metus mobiliųjų agentų technologijai itin didelį dėmesį skyrė mokslo visuomenė. Pastaraisiais metais susidomėjimas šia jau lyg ir primiršta technologija, remiantis naujomis publikacijomis, vėl išaugo. Tačiau, nepaisant technologijos privalumų darbų kiekis taip ir „neperaugo“ į platų kiekį realių programų, kuriomis naudotųsi vartotojai. Viena iš to priežasčių gali būti tradicinio ir agentų mobiliosios komercijos modelių kiekybinės analizės trūkumas. Kadangi agentų ir tradicinės komercijos priemonių analizė yra paprasčiausiai pasenusi arba analizuojamų sistemų ar įrankių analizė netenkina verslo ir neįtikina jo, kad skirtų daugiau dėmesio agentų technologijai. Šis darbas skirtas išnagrinėti kelias populiariausias agentų kūrimo sistemas, jas aprašyti, atlikti šių sistemų kiekybinę ir kokybinę analizę bei sukurti realiai veikiančią mobiliosios komercijos sistemą, naudojančią agentų technologiją. Taip pat pateikiamos mobiliosios komercijos sritys, kur gali būti pritaikyta agentų technologija ir kaip jose gali būti pritaikyti mobilumo šablonai, galintys panaudoti WAP protokolo galimybes.

Agentų technologija nagrinėjama jau daugiau nei dešimtį metų, tačiau indėlis į realias programas nėra didelis. Kai kuriuose darbuose yra atliekama skirtingų platformų analizė [PCVM99], [NM09], [BPL05] ir kt., tačiau juose sistemos palyginamos kokybiniu būdu ir dauguma lyginamų sistemų šiuo metu yra nebepalaikomos ir/ar nepasiekiamos, ar neatsižvelgiama į agentų technologijos mobilumo principus, lyginami kokybiniai parametrai per daug siaurai, kad galima būtų įvertinti agentų sistemą. Seniau buvusios populiarios agentų kūrimo platformos, tokios kaip Grasshopper ar Concordia, yra seniai nebetobulinamos. Aglets agentų platformos kūrimas yra atnaujintas – jos negalutinė „Alpha“ versija buvo išleista 2010 metais, kuomet paskutinis atnaujinimas buvo išleistas 2002 metais. Šios analizės nėra tinkamos agentų technologijoms mobiliosios komercijos kontekste nagrinėti, todėl būtina atlikti šiuo metu egzistuojančių sistemų lyginamąją analizę ir pateikti atnaujintą prieinamą analizę, kuria galima būtų pasinaudoti kuriant mobiliųjų agentų programas. Juolab, kad bus atsiradusios ir šiuo metu egzistuojančių, seniau lygintų agentų sistemų versijos, kuriose atnaujinimai nebus vien tik susieti su klaidų pataisymais, o per ne vienerius metus pačios sistemos gali pasikeisti iš esmės: pakisti naudojama agentų komunikavimo kalba, tapti komercinėmis ir pan. Šiame darbe kiekybiškai palyginamos Aglets, JADE agentų platformos,

kurių naujausios versijos buvo išleistos per pastaruosius kelerius metus, ir Jack, ADK (Tryllian), MadKit, Zeus, Grasshopper. Parinktoms agentų sistemoms buvo keliami reikalavimai, kad naudojant jas būtų galima sukurti programas, naudojančias mobiliosios komercijos agentus.

1.1. Naudojamos technologijos

Pastaraisiais metais mobilieji įrenginiai, ypač mobilieji telefonai, tapo neatsiejama kiekvieno žmogaus dalimi, pradedant nuo pačių mažiausiųjų ir tęsiant iki garbaus amžiaus žmonių. Dėl savo įgimtinių trūkumų lyginant su laidiniu ryšiu, bevielėms operacijoms atlikti turi būti panaudotos technologijos, leidžiančios sumažinti šiuos trūkumus. Darbe naudojamos technologijos, kurios leidžia sumažinti bevielių technologijų trūkumus, tokius kaip ilgas atsako signalas, nedidelis greitis, didesnė klaidų tikimybė – tai WAP, kuris ir buvo sukurtas ir tobulinamas specialiai mobiliesiems įrenginiams, ir agentų technologijos, leidžiančios sumažinti trikio pasekmes.

Papildomos problemos siejamos su mobiliosios komercijos naudojimu yra pačios bevielės technologijos brangumas lyginant su laidiniu ryšiu, kuomet duomenų kiekiai perduodami laidais yra jau, bent Lietuvoje, retai kur skaičiuojami, o didesnis, kad ir kelių megabaitų duomenų perdavimas naudojant bevieles komunikacijas yra pakankamai brangus, jei ne už patį perduotą duomenų kiekį, tai sudaromi papildomi įsipareigojimai su vartotoju. Taip pat verta nepamiršti, kad dauguma mobiliųjų įrenginių vartotojų neturi pakankamai patirties dirbant su internetu ar asmeniniu kompiuteriu, todėl interfeisas, kuriuo gali būti atliekamos mobiliosios komercijos transakcijos, turi būti intuityvus, paprastas ir neapkrautas. [MT00]

1.1.1. Agentų technologija

Pati agentų technologija pasaulyje yra tirinėjama jau daugiau nei dešimtį metų. Pradžioje kildavę diskusijų, ką laikyti programiniu agentu ir kokios agento savybės yra pagrindinės, tačiau ilgainiui nusistovėjo ir buvo sutarta, kad šios yra esminės agento savybės:

- autonomiškumas;
- socialinis gebėjimas;
- jautrumas;
- iniciatyvumas;

- dažniausiai būna statiški.

Nwanas [Nwa96] pasiūlė agentus klasifikuoti pagal išskirtas savybes ir funkcijas:

- mobilumą. Agentas statinis arba mobilus;
- simbolinio samprotavimo modelio buvimą: svarstantis (angl. deliberative) arba reaktyvus (angl. reactive).
- idealių ir svarbiausių atributų parodymą: tokių kaip autonomiškumas, kooperacija, mokymasis.
- vaidmenis: informacijos arba Interneto;
- hibridines filosofijas, kurios viename agente apjungia du ar daugiau principus;
- antrinius požymius, tokius kaip universalumas (angl. versatility), geranoriškumas (angl. benevolence), teisingumas (angl. veracity), tikrumas (angl. trustworthiness), laikinas nepertraukiamumas (angl. temporal continuity), galimybė grakščiai „nulūžti“ (angl. fail gracefully), ir protinės bei emocinės savybės.

Iš šių charakteristikų Nwana išvedė keturis agentų tipus:

- bendradarbiaujantieji agentai (angl. collaborative agents);
- bendradarbiaujantieji besimokantys agentai (angl. collaborative learning agents);
- sąsajos agentai (angl. interface agents);
- sumanūs agentai (angl. smart agents);

Pagal šias savybes Nwanas suskirstė agentus į 7 kategorijas:

1. Bendradarbiaujantieji agentai (angl. collaborative agents).
2. Sąsajos agentai (angl. interface agents).
3. Mobilieji agentai (angl. mobile agents).
4. Informaciniai/Interneto agentai (angl. information/Internet agents).
5. Reaktyvieji agentai (angl. reactive agents).
6. Hibridiniai agentai (angl. hybrid agents).
7. Sumanūs agentai (angl. smart agents).

Be šių agentų galima dar išskirti ir agentą tarpininką (angl. mediator agent), skirtą informacijai integruoti.

Bendradarbiaujantieji agentai. Šie agentai pasižymi autonomiškumu ir gebėjimu bendrauti su kitais agentais tam, kad galėtų atlikti savo užduotį. Jie gali mokytis, tačiau šia savybe retai naudojasi. Kartais šiems agentams tenka derėtis tam, kad prieiti kompromisą su kitais agentais dėl atliekamų užduočių. Vienas svarbiausių bendradarbiaujančios agentų sistemos uždavinių – subalansuoti duomenų bazių ir senstelėjusios įrangos sistemų

autonomiškumą su potencialiomis galimybėmis jas patobulinti panaudojant bendradarbiaujančius informacijos agentus. Iš kitos pusės, agentai nors ir sąveikauja, tačiau vykdydami užduotis nepraranda savo individualaus autonomiškumo laipsnio.

Sąsajos agentai. Agento pagrindinės savybės:

- autonomiškumas
- asistavimas vartotojui
- gebėjimas mokytis. Agentas mokosi šiais būdais:
    - mokymasis iš vartotojo. Agentas stebi ir imituoja vartotoją;
    - mokymasis iš kritikos. Gauna iš vartotojo teigiamus ir neigiamus atsiliepimus;
    - mokymasis pagal instrukciją. Agentas gauna tikslią instrukciją iš vartotojo kaip veikti;
    - mokymasis iš kitų agentų. Agentas klausia kitų agentų patarimo.

Mobilusis agentas. Pagrindinės šio agento savybės:

- autonomiškumas;
- bendradarbiavimas.

Agentas sugeba klajoti kompiuteriniuose tinkluose, atlikti reikalingą užduotį, veikdami savo vartotojo vardu. Svarbiausia sąlyga – atlikęs užduotį, jis privalo grįžti „namo“.

Informaciniai agentai. Pagrindinės savybės:

- autonomiškumas;
- gebėjimas mokytis.

Informacinis agentas savo apibrėžimu panašus į bendradarbiaujantį agentą. Vienas pagrindinių skirtumų – bendradarbiaujantys agentai apibūdinami pagal tai, kas jie yra, o informaciniai pagal tai, ką jie daro.

Informacinis agentas atlieka tokias:

- kaupia ir tvarko informaciją;
- sintezuoja ir pateikia informaciją;
- asistuoja vartotojui, t.y., padeda atlikti internete su informacijos apdorojimu susijusias užduotis.

Informaciniai agentai dar gali būti klasifikuojami:

- komunikacijos agentas;
- žinių agentas;
- bendradarbiavimo agentas;
- užduočių agentas.

Reaktyvusis agentas. Pagrindinė savybė – paprastas sąveikavimas su kitais agentais. Reaktyvieji agentai priklauso specifinei agentų grupei, kuri neturi vidinio simbolinio aplinkos modelio, vietoj to jie veikiai/atsako pagal veiksmas-atoveiksmis būdų. Tačiau jei į agentų grupę pažvelgsime globaliai, tai galime pastebėti sudėtingų elgesio modelių.

Hibridiniai agentai. Iš principo hibridiniai agentai savyje apjungia dviejų ar daugiau aukščiau minėtų agentų rūšių savybes. Pagrindinė priežastis dėl ko atsirado hibridiniai agentai yra ta, kad skirtingais atvejais geriau turėti skirtingų agentų savybes viename agente.

Sumanūs agentai. Tikslaus šių apibūdinimo nėra, nėra tikslaus agento sumanumo apibūdinimo. Apskritai sumanūs agentai šiuo metų yra labiau mokslininkų siekimas negu realybė.

Agentas tarpininkas. Pagrindinis šio tipo specializuoto agento uždavinys – palaikyti heterogeninių informacijos sistemų suderinamumą. Tarpininkas buvo apibrėžtas kaip „programinis modulis, kuris naudoja užkoduotas žinias apie duomenų aibes ar poaibius, tam kad sukurtų informaciją aukštesnio lygio programoms. Tarpininkas yra palaikomas kitų agentų, vadinamų „aplankais“ (angl. „wrapper“), aibės. Kiekvienas toks agentas suteikia priėjimą prie lokalaus informacijos šaltinio ir pateikia to šaltinio turinį bei konvertuoja reikiamus duomenis.

Pagrindinės agento tarpininko užduotys:

- kitos srities specifinės ontologijos interpretacija, pagal savo srities ontologijas;
- užklausos apdorojimas ir skaidymas;
- Agentų užklausų rezultatų apjungimas į vieną atsakymą, pateikiamą vartotojui.

1.1.2. Mobilioji komercija

Prekių ir paslaugų pirkimas ir pardavimas, panaudojant mobiliuosius įrenginius, tokius kaip mobilusis telefonas ar delninis kompiuteris, ir yra Mobilioji komercija. Mobiliosios komercijos užnugaryje esančios technologijos, pagrįstos WAP protokolu, yra toli pažengusios Europoje, o telefonuose jau yra įdiegtos WAP naršyklės. 2010 metais Europoje buvo apie 40% sumaniųjų telefonų visų telefonų rinkos, o Didžiojoje Britanijoje net 70%, kas mobiliosios paplitimą ir panaudojimą dar labiau paspartins. Mobiliųjų telefonų kompanijos bendradarbiauja su ryšio paslaugų tiekėjais ir stambiosiomis informacinių technologijų kompanijomis kurdamos WAP protokolą palaikančius sumaniuosius telefonus, bei paslaugas leidžiančias gauti vartotojams reikiamą informaciją. Panaudojant Bluetooth ar bevielį ryšį ir mobiliuosius ar sumaniuosius telefonus galime jau dabar mokėti už automobilių stovėjimą,

komunalinius mokesčius, naudotis banko paslaugomis, gauti el. paštą ir kt. Mobilioji komercija ir naujosios technologijos gali padėti nutiesti naujus kelius tradiciniams paslaugų teikimo būdams, taip išplėsdamos mūsų darbo vietą už kambario ribų. Be to, didėjantis paslaugų greitis, saugumas ir interaktyvumas gali pabandyti konkuruoti su laidine komercija ir tapti pagrindiniu būdu skaitmeninės komercijos operacijoms vykdyti.

1.1.3. WAP protokolas

Pagrindinė WAP kaip standarto bevielei komunikacijai idėja buvo išplėsti mobiliosios telefonijos komunikavimo galimybes, t.y. mobilųjį telefoną paversti įrenginiu, skirtu ne tik pokalbiams, bet skirti bendrauti tiek su kitais įrenginiais ar programomis bevieliame tinkle. WAP standarte atsižvelgta į bevielės duomenų perdavimo technologijos apribojimus lyginant su laidine ir į mobiliųjų įrenginių apribojimus lyginant su personaliniais kompiuteriais.

WAP standartas apibrėžia „end-to-end“[1] protokolą, kuriuo pabrėžiama, kad operacijos turi būti atliekamos komunikavimo mazguose, arba kaip įmanoma arčiau jų. Protokolas kartu su naršykle, kaip programa pateikiančia interfeisą skirtą naršyti turinį, yra du esminiai WAP standarto komunikacijos elementai. Programos, skirtos turiniui naršyti, protokolas yra daugiasluoksnis komunikavimo protokolas, kuris yra kiekvieno su WAP standartu susietu vartotojo komponento dalis. Kitoje tinklo pusėje yra serverio komponentas, kuris įgyvendina kitą „end-to-end“ principo protokolo dalį. Tinklo pusėje, serverio dalis, dažnai suprantama kaip vartai, skirta nukreipti vartotojo užklausoms į programų serverius. Šis serveris gali fiziškai priklausyti operatoriaus tinklui, veikdamas kaip tiltas tarp bevielių ir laidinių kompiuterio tinklų.

Integruota į mobilųjį įrenginį naršyklė geba atvaizduoti informaciją vartotojui ir priimti jo užklausas. Pagrindinis būdas informacijos atvaizdavimui yra WML kalba, paremta duomenų apibrėžimais DTD skirtais XML kalbai. Pagrindinis WML vienetas yra „korta“[2], kuri apibrėžia vieną vartotojo matomą langą. Navigacija yra atliekama tarp „kortų“ , kurios yra priskirtos tai pačiai „kaladei“[3]. „Korta“ yra hierarchiškai žemiausias WML dokumento elementas, o „kaladė“ aukščiausio lygio WML dokumento elementas, pateikiantis galimas alternatyvas vartotojo veiksmams. Teoriškai, manoma, kad vartotojas prisijungdamas prie WAP vartų, pateikia pradinį puslapį (pasirinktą WML „kaladę“), o tuomet atlieka veiksmus rinkdamasis iš skirtingų galimų kaladžių. Patartina, kad informacija pateikiama pradinėje

[1] „End-to-end“ - http://en.wikipedia.org/wiki/End-to-end_principle
[2] (angl. card)
[3] (angl. deck)

„kaladėje“ būtų sistematizuota, vartotojas galėtų rasti reikiamą informacijąar pasiekti jam aktualiausias, mėgstamas ar dažniausiai lankytas vietas nuorodų pagalba ar su kuo mažiau pasirinkimų. [MT00]

2. Agentai ir jų sistemos

2.1. Programinio agento paradigma

Agentas yra vienas programinis objektas, turintis apibrėžtą tikslą. Jis siekia šio tikslo nepriklausomai funkcionuodamas, ir tuo pačiu sąveikauja su savo aplinka ir/arba kitais agentais. Pati agento sąvoka yra plačiai naudojama mokslininkų ir tyrėjų diskurse ir dažnai bendrąja prasme įvardijama kaip sistemų analizavimo įrankis. Plačiau informacijos agentas apibrėžiamas kaip autonominis programinis objektas, turintis prieigą prie vieno ar daugelio heterogeninių, geografiškai paskirstytų informacijos šaltinių ir gebantis specializuotai tarpininkauti įgyjant ir apdorojant svarbią informaciją. Agentas veikia vartotojų ar kitų agentų vardu ir, dažniausiai, realiu laiku.

Daugeliui agentų technologijų mokslininkų, vienas iš labiausiai priimtinų apibrėžimų yra šis: programinis agentas – tai programinės įrangos objektas, kuris nepertraukiamai ir autonomiškai funkcionuoja tam tikroje aplinkoje ir gali bendrauti su kitais agentais ar procesais. [Sho97]

2.2. Agento savybės ir funkcijos

Agentas, kaip kompiuterinė sistema, turi turėti žmonėms priskiriamas savybes:

- mentalinės sąvokos: žinios, tikėjimas, intencija, ir t.t.;
- emocinės sąvokos;
- vizualinis vaizdavimas (pvz. animuotas veidas, ir t.t.);
- kiti agento atributai;
- mobilumas;
- teisingumas: agentai valingai neskleidžia neteisingos informacijos;
- palankumas: nėra konfliktuojančių tikslų tarp agentų;
- racionalumas: agentai neveiks prieš savo tikslus.

Agentų technologijų kūrėjų ir mokslininkų priimtu bendru sutarimu, kiekvienas agentas gali turėti tokias savybes:

- reaktyvumas (angl. reactivity): galimybė atrankos būdu jausti ir veikti. Agentai suvokia juos supančią aplinka, (kuri gali būti fizinis pasaulis, vartotojas

panaudojus vartotojo sąsają, kitų agentų grupė, Internetas, ar galbūt kombinaciją viso tai kas išvardinta), ir laikui bėgant reaguoti į pokyčius atsirandančius toje aplinkoje;

- autonomiškumas (angl. autonomy): iniciatyvus ir savarankiškas funkcionavimas. agentai funkcionuoja be tiesioginio žmogaus įsikišimo ir gali kontroliuoti savo veiksmus ir vidines būsenas;
- bendradarbiavimas (angl. collaborative behavior): galimybė dirbti išvien su kitais agentais siekiant bendro tikslo. agentai bendrauja su kitais agentais (ir galbūt žmonėmis) tam tikros agentų bendravimo kalbos pagalba;
- „žinių lygio“ bendravimas (angl. „Knowledge-level“ communication): galimybė bendrauti su asmenimis ir kitais agentais, kalba, kuri panaši į žmonių kalbą, o ne įprastiniais simbolinio lygio „programa-su-programa“ protokolais;
- dedukciniai gabumai (angl. inferential capability): pasinaudojus sukauptomis žiniomis apie bendruosius tikslus ir pageidaujamus metodus, agentas gali veikti su abstrakčiom užduotim. Agentas gali turėti tikslius savo, vartotojo, situacijos ir kitų agentų modelius;
- laikinas nepertraukiamumas (angl. temporal continuity): tapatybės ir būsenos išlaikymas ilgais laiko tarpais;
- individualybė (angl. personality): galimybė išreikšti charakterio bruožus tokius kaip emocijos ir pan.;
- prisitaikomumas (angl. adaptivity): galimybė mokytis ir tobulėti su patirtimi;
- mobilumas (angl. mobility): galimybė savarankiškai migruoti numatyta kryptimi iš vienos platformos į kitą.

Nors pirmieji darbai agentų srityje buvo pradėti siekiant išstudijuoti paskirstytojo intelektualumo (angl. distributed intelligence) skaičiuojamuosius modelius (angl. computational models), tačiau pagrindinė susidomėjimo banga kilo atsiradus dviem papildomiems praktinės kilmės tikslams:

1. Paskirstytų skaičiavimų sudėtingumo supaprastinimas.
2. Esančių vartotojo sąsajų trūkumų įveikimas.

Abu šie tikslai turi bendrą tikslą – pagerinti sąsają tarp žmogaus ir programos. Bendras agentų privalumas yra tas, kad jie visas problemas gali spręsti tuo pat metu.

Dauguma informacinių agentų yra kuriami akademinėse ir verslo tyrimų laboratorijose. Manoma, kad ilgainiui jų funkcijos stipriai plėsis ir bus naudojamos daugelyje su informacija susijusių gyvenimo sričių. Kiekvieno agento funkcijos priklauso nuo aplinkos, kurioje jis turi

veikti ir kokias funkcijas atlikti. Pagrindinis reikalavimas agentui – laiku ir vietoje priimtas teisingas sprendimas, kuris teikia naudą.

Šiuo metu galima apibrėžti kelias pagrindines funkcijas, kurias turi atlikti agentas:

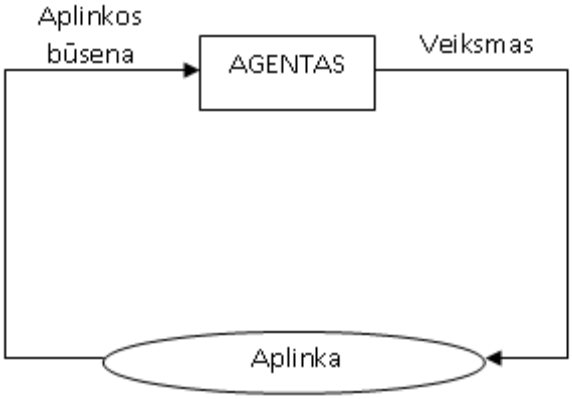
1. Informacijos kaupimas ir tvarkymas.
2. Informacijos sintezė ir prezentacija.
3. Intelektualus vartotojo asistavimas.

2.3. Agento veikimas

2.3.1. Aplinkos savybės

Agentas egzistuoja aplinkoje, kuri sudaryta iš tam tikrų būsenų. Jis sąveikauja su aplinka konkrečiais veiksmais. Agentas gali veikti įvairių tipų aplinkose, todėl aplinka turi didelę įtaką agentų veiksmams.

1 pav. pavaizduotas abstraktus agento veikimas, įtakojamas aplinkos.



1 pav. Abstraktus agento funkcionavimas [Vys08]

Aplinkos savybės:

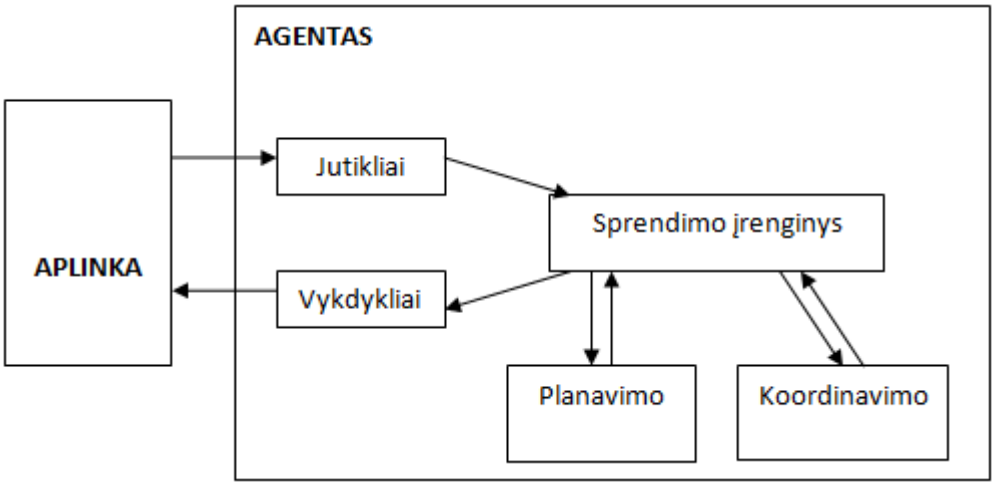
1. Prieinama ir neprieinama. Tai tokia aplinka, kai agentas gauna visą informaciją apie aplinkos būsenas. Dažniausiai visos informacijos jis negauna.
2. Determinuota ir nedeterminuota. Aplinka yra determinuota, kai žinomas veiksmo vykdymo rezultatas. Tačiau dažnai realūs uždaviniai yra nedeterminuoti.
3. Epizodinė ir neepizodinė. Epizodinėje aplinkoje agento veiksmai priklauso nuo diskrečių epizodų skaičiaus.

4. Statinė ir dinaminė. Dažniausiai susiduriame su dinamine aplinka, kai kinta ne tik aplinka, bet ir joje dalyvaujantys agentai.
5. Diskreti ir tęstinė. Aplinka yra diskreti, kai agentas turi atlikti baigtinį skaičių veiksmų.

2.3.2. Agento veikimas

Agentai laikosi tikėjimų, noro ir ketinimų principo (Belief, Desire and Intention). Įvykus kokiam nors įvykiui, agentas į jį reaguoja remdamasis savo žiniomis (tikėjimai) ir tikslais, kuriuos nori pasiekti šiuo įvykiu (troškimai). Tikslai įgyvendinami tam tikrais, iš anksto numatytais planais (ketinimai). Vadinasi, programinis agentas gali pakeisti žmogų, nes turi galimybę išsirinkti ir priimti geriausią sprendimą. Toks agentas gali veikti savarankiškai, jis gali būti naudojamas kaip tam tikras „protingas“ asistentas, laborantas atliekant pasirinktas užduotis. Be to, agentas gali būti tam tikros sistemos valdytojas, kuris atlieka tarpininko tarp tos sistemos posistemių vaidmenį.

Klasikinė agento architektūra, kai agentas priima informaciją iš aplinkos per sensorius ir atlieka veiksmus, kurie įtakoją aplinką panaudojant efektorius, pavaizduota 2 paveiksle.



2 pav. Agento architektūra [Vys08]

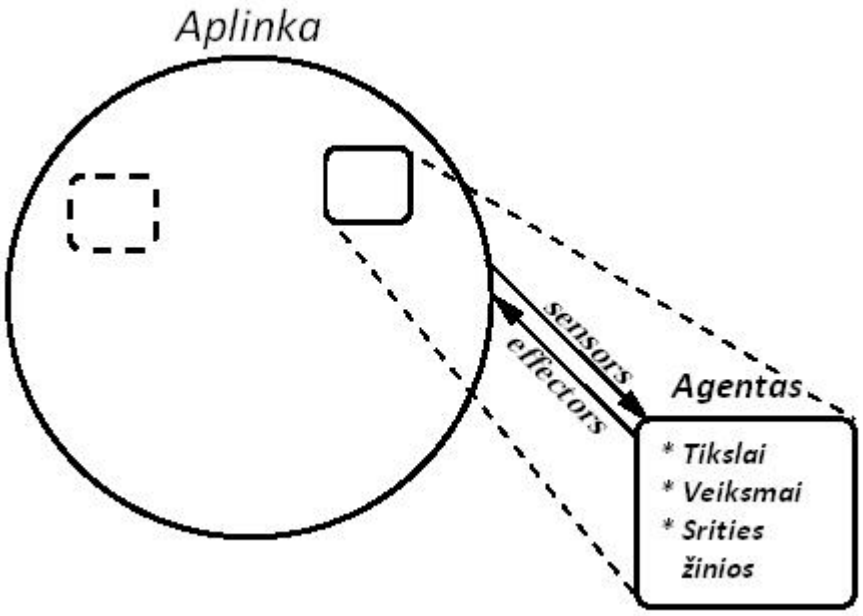
Veiksmų pasirinkimas yra atliekamas uždavinių sprendimų įrenginyje (angl. problem solver), kuris jei reikia gali pasitelkti planavimo (angl. scheduling) ir koordinavimo (angl. coordination) modulius. Planavimo veiklos metu yra nustatoma veiksmų seka užduotam tikslui pasiekti; koordinavimo veiklos metu, tam, kad būtų pasiekti aukštesnio lygio tikslai,

yra bendradarbiaujama ir su kitais agentais. Idealiu atveju, agentas, nepertraukiamai funkcionuojantis aplinkoje, turėtų sugebėti mokytis iš savo patirties. Be to, tikėtina, kad agentas egzistuos aplinkoje kartu su kitais agentais ir procesais, todėl agentas turi sugebėti bendrauti ir bendradarbiauti su jais, ir galbūt net keisti savo vietą toje aplinkoje.

Objektas, funkcionuojantis dinamiškoje aplinkoje arba aplinkoje, kur galimi tam tikri pokyčiai, naujos situacijos, turi turėti mechanizmą, kuris jam padėtų prie jų prisitaikyti, sistemingai apdorotų ir sujungtų naują informaciją apie aplinką. Tarkime, turime į tikslą orientuotų agentų sistemą, kur kiekvienas agentas turi būti atsakingas už vienos ar kelių funkcijų vykdymą tuo pat metu. Funkcionalumu paremtos sistemos efektyvumą didina galimybė saugoti vidines būsenas ir mokymosi galimybė, kai kiekvienas režimo modulis gali būti pritaikytas nepriklausomai, dirbant su atskira terpės reprezentacija. Suvokimo komponentas apdoroja jutiklių duomenis ir sukuria tipinių situacijų, kylančių iš objekto aplinkos sąveikų, koncepcijas.

2.4. Vieno agento sistema

Agentas vieno agento sistemoje modeliuoja save patį, aplinką ir tarpusavio sąveiką. Kitaip tariant, tokia sistema veikia centralizuotai. Agentas yra ir dalis aplinkos, tuo pačiu, jis turi aplinkoje dar ir papildomus komponentus. Jie yra nepriklausomi objektai, turintys savas žinias ir tikslus. Vieno agento sistemoje agentas atpažįsta tokius komponentus. Jei egzistuoja kiti agentai, jie yra traktuojami kaip aplinkos dalis. [Vys08]



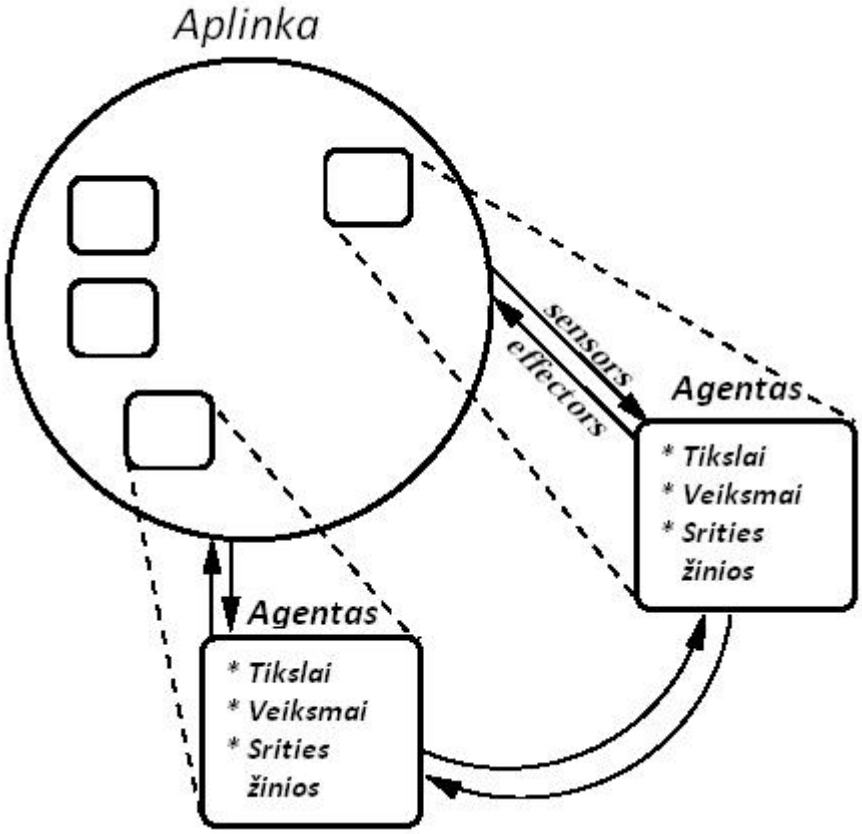
3 pav.Vieno agento sistema [Vys08]

2.5. Multiagentai

Multiagentų sistemos skiriasi nuo vieno agento sistemų tuo, kad egzistuoja keletas agentų, kurie modeliuoja vienas kito tikslus ir veiksmus.

Multiagentų sistemos susideda iš aibės agentų, kurie bendradarbiauja tarpusavyje, kad įvykdytų konkrečius uždavinius. Tokios sistemos išsiskiria moduliškumu ir suteikia efektyvų metodą projektuoti sistemas, kurios naudoja paskirstytus, įvairiarūšius informacijos šaltinius sprendimams priimti. Sistemos su adaptyvia agentų mokymosi strategija geriausiai veikia didelėse, atvirose, dinamiškose ir neprognozuojamose terpėse . Jos savarankiškai parenka ir įvykdo vartotojų užduotis. Individualaus agento požiūriu multiagentų sistemos skiriasi nuo vieno agento sistemų tuo, kad aplinkos dinamika gali būti nustatoma kitų agentų pagalba. Be to, kiti agentai apgalvotai veikia aplinką nenuspėjamais būdais. Taigi, visos multiagentų sistemos gali būti laikomos kaip turinčios dinamines aplinkas. [Vys08]

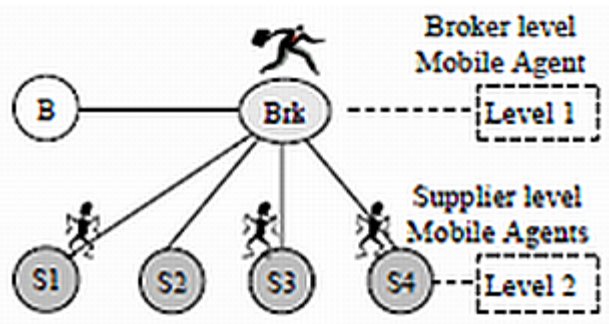
Multiagentų sistemų valdymą sudaro tokios funkcijos: agentų koordinavimas, suderinamumas, agentų ar jų grupių aktyvavimas ir deaktyvavimas, agentų parinkimas, naujų agentų sukūrimas, nereikalingų agentų sunaikinimas, individualių agentų ir jų grupių pritaikymas prie pakitusios aplinkos ir mokymas. [Sam02]



4 pav. Multiagentinė sistema [Vys08]

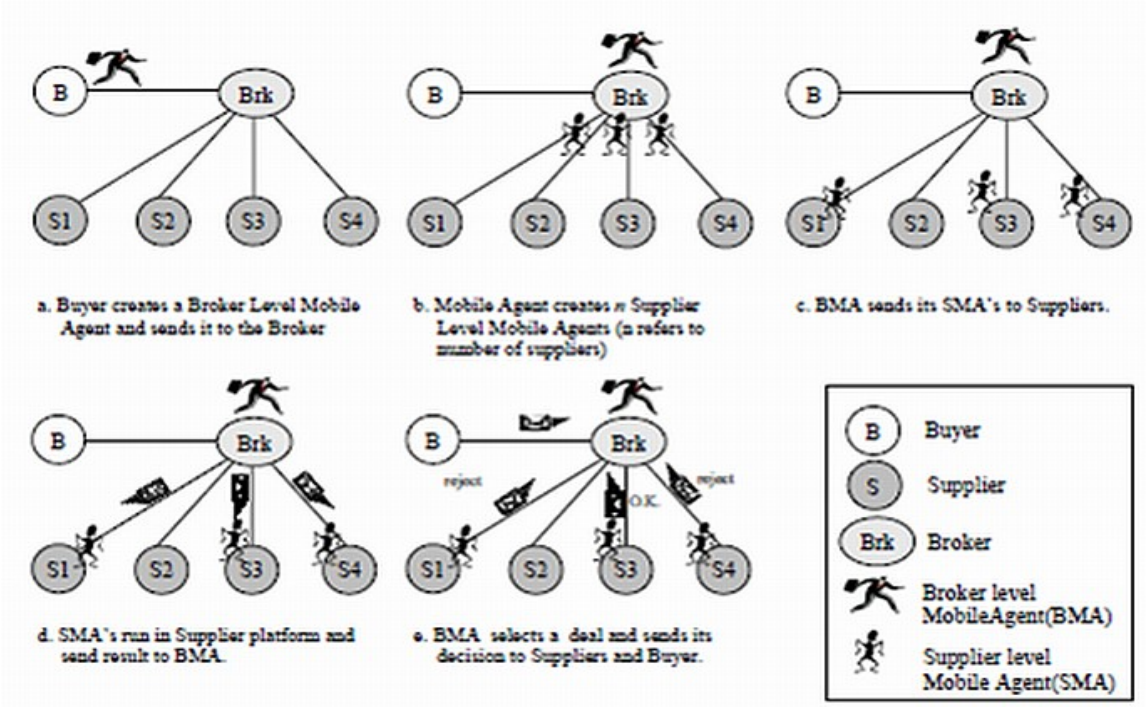
2.6. Dviejų lygių mobiliųjų agentų sistema

Sistema yra sudaryta iš mobiliųjų agentų, kurie priklauso dviems skirtingiems lygiams savo vykdymu, atsakomybėmis ir užduotimis, kaip parodyta 5 paveiksle, brokerio mobiliojo agento lygis (BMA – Broker Mobile Agent) ir tiekėjo mobilus agento lygis (SMA – Supplier Mobile Agent). BMA yra sukuriamas pirkėjo agento ir siunčiamas brokerio agentui.



5 pav. Dviejų lygių mobiliųjų agentų sistema [SE03]

BMA sukuria SMA agentus ir siunčia juos tiekėjams, siekiant atlikti paiešką jų duomenų bazėse ir parinkti vieną iš produktų, ir derėtis su tiekėju (jei būtina). Sistema neapima tik vieną mobilųjį agentas, kuris aplanko kiekvieną tiekėją. Vietoj to, yra siunčiamas mobiliojo agentas klonas kiekvienam iš tiekėjų, tuo pačiu metu, ir tokiu būdu panaudojant lygiagretų apdorojimą. Šis lygiagrečių skaičiavimų modelis yra ypač svarbus, nes daugiau tiekėjų, gali būti apieškoti per trumpesnį laiką ir pirkėjui greičiau pateikiami rezultatai, padėsiantys geriau išrinkti savo sprendimą, kurį tiekėją pasirinkti.



6 pav. Produkto pirkimas iš tiekėjo dviejų lygių mobiliųjų agentų sistemoje

Daugelyje mobiliosios komercijos sistemų, pirkėjas turi fiksuotą tiekėjų skaičių, kurie yra inicializuoti su startuojant sistemai. Kai naujas tiekėjas turi būti pridėtas, tai turi būti atliekama rankiniu siunčiant jo adresą ir reikiamus parametrus. Didelėse ir dinamiškose aplinkose gali būti kintantis pirkėjų ir tiekėjų skaičius bet kuriuo metu. Dinamiškai besikeičiančiose mobiliosios komercijos rinkose, sistema turėtų turėti galimybę prisitaikyti prie šiame kintančių aplinkybių. Kad įvykdyti šį reikalavimą, turi būti sistemą, kuri naudoja registravimo/paskelbimo mechanizmą, užtikrinantį šias registravimo ir naujų tiekėjų paskelbimo operacijas, siekiant padidinti našumą ir veiksmingumą viešųjų pirkimų procesui, taip pat atsižvelgiant į sąnaudas, kokybę, efektyvumą ir laiką, tiek pirkėjų, tiek ir tiekėjų. Vartotojo, kuris nori pirkti ar parduoti produktą sukuria agentą, suteikia jam tam strateginę

kryptį, ir siunčia jį į centralizuotą agentų rinką. Sistemos mobilus agentai aktyviai aplanko tiekėjus ir komunikuoja su jais. Kiekvieno agento tikslas yra atlikti priimtiną sandėrį, atsižvelgiant į vartotojo nustatytus apribojimus, pavyzdžiui, pageidaujamą kainą, mažiausią priimtiną kainą, datą, iki kurios užbaigti sandorį ar pageidaujamą kiekį. Pirkimo procesas vykdymas atliekamas tokia tvarka:

1. Yra keletas pirkėjų ir tiekėjų, kurie prisijungė prie sistemos su savo produkcija ir paslaugomis. Pirkėjų ir tiekėjų sistema, skaičių gali padidinti arba sumažinti bet kuriuo metu.
2. Kai vartotojas nori pirkti produktą, jis turi pateikti prašymą iš pirkėjo posistemio. Pirkėjas agentas gauna prašymą, sukuria mobilųjį agentą, nustato reikiamus kintamuosius ir siunčia jį brokeriui, kaip parodyta paveiksle 6.a.
3. Kai šis mobilusis agentas (vadinamas brokeris lygio mobiliuoju agentu (BMA)) atvyksta pas brokerį, jis patikrina brokerio biblioteką, atrenka tiekėjus, kurie teikia reikalaujamą produktą ir sukuria naują tiekėjų lygio mobilųjį agentą (SMA) kiekvienam iš pasirinktų tiekėjų. Vėliau, BMA siunčia kiekvienai iš šių ką tik sukurtų agentų tiekėjams, kaip parodyta paveiksle 6.b ir paveikslas 6.c.
4. Kiekvienas SMA ieško produktų tiekėjo duomenų bazėje, derasi su tiekėjo agentas ir siunčia rezultatus atgal į BMA, kaip parodyta paveiksle 6.d.
5. Po to, kai visi rezultatai yra surinkti iš SMA mobiliųjų agentų (arba kai nurodytas užklausos laikas baigėsi), BMA parenka geriausią surastą sprendimą ir siunčia patvirtinimo laišką šiam SMA ir daro užklausimą produkto įsigijimui ir paties BMA mobiliojo agento susinaikinimui. BMA mobilusis agentas išsiunčia atmetimo pranešimus visiems kitiems SMA ir jie visi susinaikina patys. BMA taip pat siunčia pranešimą apie derybas ir ataskaitą pirkėjo agentui, kaip parodyta paveiksle 6.e. Ir tuomet vartotojui belieka patvirtinti agento pateiktą siūlymą. [SE03]

2.7. Mobiliųjų agentų projektavimo šablonai

Mobilūs agentai yra autonominės kompiuterinės programos, kurios gali migruoti tarp skirtingų tinklų savarankiškai, tęsiant jų vykdomas užduotis nuo taško, kur jie sustabdė savo užduotis prieš migraciją. Programų, pagristų mobiliaisiais agentais, kūrimą galima supaprastinti ir pagerinti panaudojant projektavimo šablonus. Tačiau, nepaisant jų potencialo,

šablonai retai buvo naudojami kuriant taikomąsias programas besiremiančias mobiliųjų agentų paradigma. To priežastys gali būti informacijos apie mobiliųjų agentų šablonų trūkumas arba tai, kaip sudėtinga būtų jas įgyvendinti atsižvelgiant į konkretų karkasą.

Yra skiriami šie mobiliųjų agentų projektavimo šablonai:

- „Itinerary „ - „Keliaujantis“. Mobilusis agentas migruoja tarp skirtingų įrenginių iš vieno į kitą tam, kad atliktų užduotis, prieš tai gavęs sąrašą šaltinių;
- žvaigždės formos. Mobilusis agentas prieš pradėdamas vykdyti užduotį gauna sąrašą šaltinių, į kuriuos turi migruoti. Tuomet migruoja į pirmą šaltinį, atlieka savo darbą ir grįžtą į pagrindinį šaltinį ir taip ciklišKai pereina visą sąrašą;
- besišakojantis. Mobilusis agentas prieš pradėdamas vykdyti užduotį gauna sąrašą šaltinių, į kuriuos turi migruoti, ir sugeneruoja savo kloną ir pasiunčia į visus šaltinius, kur klonai atlieką savo užduotis. Šis šablonas svarbus tuom, kad mobiliųjų agentų užduočių įvykdymui panaudojamas lygiagretus vykdymas;
- šeimininko-vergo;
- MoProxy;
- susitikimo (angl. Meeting);
- „Facilitator“;
- „Mutual Itinerary Recording“ – „Tarpusavio Kelionės registravimas“.

Mobilus agentai gali padėti įgyvendinti tam tikros esybės tikslus tikslus ir interesus, migruojant tarp skirtingų tinklo įrenginių, ar tarp tinklų, vykdant užduotis lokaliai ir tęsti jų vykdymą kitoje vietoje, nuo to taško, kur jie sustojo veikti prieš migruojant. Mobilūs agentai taip pat gali turėti, pavyzdžiui, dirbtinio intelekto ir gebėjimo bendradarbiauti funkcijas.

Mobiliųjų agentų naudojimas teikia keletą privalumų jų vartotojams:

- sumažinti tinklo srautą, nes sąveika gali būti atliekami vietoje, nepriklausomai nuo tinklo gaišaties;
- asinchroninis ir decentralizuotą įgyvendinimą, leidžianti vartotojui atsijungti nuo tinklo, kai agentai atlieka užduotį;
- gebėjimas nustatyti pasikeitusias aplinkos sąlygas ir reaguoti į aplinkos pokyčius ir toliau vykdyti užduotis savarankiškai, supaprastinant išskirstytų sistemų, kurios yra labiau patikimos ir atsparios klaidoms, kūrimą.

Galima teigti, kad šie privalumai gali būti gauti panaudojant kitas paradigmas. Tačiau, nepaisant to, didelis mobiliųjų agentų privalumas yra sujungti šiuos kelis privalumus į vientisą ir labiau abstrakčią paradigmą. [ZZ02]

Mobiliųjų agentų trūkumas yra tai, kad reikia įdiegti agentą palaikančią platformą į kiekvieną jį priimantį mobilųjį įrenginį, kurį agentui reikia aplankyti. Be to, šiose platformose, agentų kodas yra interpretuojamas, kas gali paveikti dėl sistemos veikimo spartą neigiamai. Šia prasme yra rekomenduojama vengti agentų migracijos be reikalo, nes tai kartu padidina ir tinklo srautą. Be to, perduodamas agentų kodas ir duomenys turi būti kuo trumpesni, jeigu norime pasiekti kuo didesnį agentų technologijos teikiamą naudą. Agento funkcionalumas gali būti skaidomas į tai, kas turi būti vykdoma tiksliai, žinant ko reikia. Informacija, kurią nesiruošiama panaudoti gali būti lengvai išmesta. Be to, svarbu paminėti, saugumą. Tai sudėtingas uždavinys, su kuriuo susiduriama visų išskirstytų sistemų kūrime. Kuo sistema saugesnė, tuo labiau tai neigiamai veikia sistemos spartą. [FDAS02]

Nors mobilieji agentai gali atnešti daug naudos ir patogumo į mobiliosios komercijos sistemas, sėkmę ar nesėkmę šiai paradigmai tiesiogiai užtikrina saugumo klausimas, ar tinkamas apsaugos mechanizmas gali būti veiksmingai įgyvendinamas ir naudojamas. Ypatingas dėmesys turi būti skirtas agentų migravimo metu, kai šie su svarbiais duomenimis gali būti perimti ir panaudoti be sutikimo kenkėjiškų programų [MSS99].

3. Agentų platformos

Vyrauja dvi agentinų sistemų projektavimo kryptys: autonominių agentų ir multiagentinių sistemų (MAS). Autonominis agentas sąveikauja tik su vartotoju ir realizuoja tam tikras funkcijas. Tuo tarpu MAS sąveikauja skirtingi agentai ir atlieka uždavinius, kurių nepajėgia išspręsti vienas agentas. MAS dar kartais vadinamos agentūromis (agency), agentų bendruomenėmis (society), komandomis (team) ir pan. Manoma, kad autonominiai agentai greitai nebegalės veikti vieni, nes plačiai paplitus agentų technologijai, jie negalės atlikti savo užduočių izoliuotai, nesusitikdami su kitais agentais.

MAS pranašumai lyginant su vieno agento programa akivaizdūs: sumažėja kolektyviai atliekamo veiksmo trukmė, pasiekiamas didesnis patikimumas, nes nesėkmė gali nutikti tik atskirame taške, sistemą lengva modifikuoti, tobulinti, naudoti daugelį kartų ir t. t. Esminis MAS elementas yra programinis agentas, sugebantis suvokti situaciją, priimti sprendimą ir komunikuoti su kitais agentais. Šiomis galimybėmis MAS kokybiškai skiriasi nuo egzistuojančių sistemų, nes pastarosios neturi savęs organizavimo ir evoliucijos savybių. MAS atveju agentai ,,gyvena" ir veikia bendruomenėje, kur neišvengiamas bendradarbiavimas siekiant tiek savo, tiek bendrųjų sistemos tikslų. Agentų bendradarbiavimas įgyvendinamas per jų kooperaciją, tačiau dažnai pasitaiko ir konkurencijos atvejų, kai agentai turi skirtingus tikslus.

Vis didesnę reikšmę MAS tyrimo ir konstravimo srityje įgauna virtualių agentų organizacijų klausimai. Organizacijos apibrėžiamos agentų struktūra ir jų funkcijomis, charakterizuojamos sprendžiamais uždaviniais, agentų vaidmenų ir teisių paskirstymu, gebėjimu bendrauti, įvertinti savo veiklą ir pan. Pirmosiose MAS agentai atstovavo asmenis, kurių vardu ir nurodymais sąveikavo tarp savęs pranešimais (pavyzdžiui, pardavėjo ir pirkėjo agentai Internete). Agentas gautą užduotį galėjo suskaidyti ir jos dalis paskirstyti tarp kitų agentų, gauti rezultatus ir priimti sprendimą. Tai buvo paprastos MAS struktūros, vadinamos grupinėmis (groups), komandinėmis (teams) ir interesų grupėmis (interest groups). Šią paradigmą pradėjo keisti paskirstytų sistemų koncepcija, kurioje žinios ir resursai paskirstomi tarp agentų, išlaikant bendrą valdymo organą, priimantį sprendimą kritinėse arba konfliktinėse situacijose. Tai hierarchinis MAS organizacijos modelis, pagrįstas ,,šeimininko/vergo" (Master/Slave) santykiais. Tokioje struktūroje galimi ir keli hierarchijos lygiai.

MAS struktūra dominuoja decentralizuotų sistemų lygmenyje, vadinama rinka (Market). Paprasčiausia rinkos tipo organizacija pagrįsta tiekėjo ir pirkėjo santykiais, o pagrindinis modelis, susijęs su tokia struktūra, yra konkurencinės (competitive) MAS. Čia dalyvaujantys agentai suinteresuoti tik savimi, kiekvienas turi savo tikslus, todėl tarp jų vyksta konkurencija arba prekių bei paslaugų tiekime, arba jų pirkime. Toks modelis gerai tinka atvirose sistemose, kuriose gali dalyvauti skirtingų asmenų suprojektuoti agentai. Siekiant suaktyvinti konkuruojančių agentų bendravimą, gali būti naudojamas ir taip vadinamas federacinės bendruomenės (federation community) modelis. Tokiose organizacijose sistemos agentai suskirstomi į grupes, kiekviena iš jų susiejama su palengvinančiu darbą įgaliotiniu (facilitator), kuris rūpinasi ,,svetimų" agentų identifikavimu bei duoda leidimą su jais komunikuoti.

MAS aplinka yra intelektuali, kurią galima taikyti įvairiose srityse, kur reikalingas situacijos stebėjimas ir įvertinimas arba kontrolė, diagnostika. Taip pat MAS tinka ištekliams planuoti, kai skirtingi agentai derina savo planus, kad išvengtų konfliktų arba juos galėtų kartu išspręsti ir pasiekti maksimalią visos sistemos naudą [PAG08].

Agentų platformos tikslas yra supaprastinti agentų programų kūrimą, pateikiant infrastruktūra, kurioje agentai galės „gyventi". Ją sudaro pagrindinės valdymo dalys, tam kad galima būtų priimti ir laikyti savyje agentus vienarūšėje aplinkoje, ir papildomai yra pateikiamos priemonės agentų tarpusavio komunikacijai. Kitaip sakant, agentų platformos yra apibūdinamos vidinę ir socialinę architektūrą. [BPL05] Vidinė architektūra apibrėžia vidines koncepcijas ir mechanizmus, kuriuos agentas naudoja, tam kad atliktų reikalingus sau veiksmus, tuo tarpu socialinė architektūra atsakinga už koordinacija tarp agentų ir komandos valdymą. Techniškai, platforma apibūdinama naudojama programavimo kalba, kurią ji pateikia agentams realizuoti, ir prieinamais įrankiais skirtais kūrimui, administravimui ir agentų derinimui. Šiame darbe nesiekiama apžvelgti visų galimų platformų Šioje dalyje aprašytos svarbiausios platformos šiai dienai, atskirus jas nuo kitų pagal jų kodo prieinamumą, vykdomą tobulinimą, dokumentacijos kiekį. Taip pat pateiktas keletas gairių, kuriomis vadotasi renkantis agentų platformą darbui atlikti.

3.1. Agentų platformos pasirinkimo kriterijai

Agentų platformos ir metodologijos pasirinkimo kriterijai įtakojami problemos rūšies. Siūlomi platformos įrankiai turi padėti kurti sistemą nuo paties modeliavimo iki sistemos

projektavimo ir sukūrimo. Kūrimo įrankiai gali būti skirstomi į kodo kūrimą orientuotus, tokius kaip integruotos programavimo platformos (IDE), įrankius derinimui, testavimui, diegimui ir administravimui. Pasirinktos agentų platformos paprastai stengiasi atitikti, kuriuos nors standartus (FIPA ar MASIF, arba abu), todėl renkantis platformą kuriamai sistemai „prisiuvamas“ ir konkretus agentų standartas. Patartina nepamiršti, kad agentų standartai nesikirstų su problemos sferos standartais. Pačios naudojamų agentų koncepcijos pasirinkimas taip pat gali nulemti projekto sėkmę, kuomet sprendžiamas kurios agentų savybės yra svarbios. Konkreti platforma nusako taip pat ir priemones, kuriomis bus realizuojama sistema, taip pat ir palaikomus standartus ir koncepcijas. Taigi, ji ir nusako agentų filosofiją, kuri bus panaudota kuriant sistemą. Jei pateiktoji filosofija neatspindi svarbiausių sprendžiamos problemos aspektų, arba kertasi su problemoje apibrėžtais standartais ir yra nesuderinama, tuomet tai komplikuos problemos sprendimą ir iššauks papildomų sąnaudų realizuojant sistemą. Visų šitų kriterijų teisingas pasirinkimas sumažins riziką, kad projektas gali būti nesėkmingas, ir padidins tikimybė, kad bus sukurta efektyvi agentų technologiją naudojanti sistema. Panašios priklausomybės tarp agentų platformos ir metodologijos taip pat egzistuoja. Metodologija turi palaikyti naudojamą agentų filosofiją, kitaip kyla nesuderinamumas tarp platformos ir agentų platformos, kuris lemia spragą tarp sistemos modeliavimo ir realizacijos. [SBPL04] Šiuo aspektu galima pamėginti paaiškinti didelį atsiskyrimą tarp agentų technologijos teorinių tyrimų ir praktinės realizacijos spragą, kuomet mokslinėje visuomenėje buvo atlikti nemažai darbo, o tas darbas taip ir nepavirto į didelį kiekį plačiai naudojamų programų. Kalbant toliau apie metodologijos ir platformos tarpusavio susiejimą, pavykus susieti metodologiją ir platformos artefaktus, įmanoma pasiekti sklandų perėjimą nuo projektavimo į realizaciją, pavyzdžiui per kodo generatorius, kuomet sukūrus sistemos modelį, panaudojus kodo generatorių yra sugeneruojamas programinis kodas, kas labai sumažina rutininio funkcijų aprašomojo darbo. Rezultate, kompromisas tarp standartų ir koncepcijų turi būti pasirinktas. Tam tikra koncepcija gali gerai padėti spręsti problemą, tačiau nepakankama parama iš metodologijos ar agentų platformos, privers apsvarstyti problemai spręsti kuriamos sistemos modelį iš naujo. Platus įrankių pasirinkimas ir jų kokybė taip pat daro įtaka metodologijos ir platformos pasirinkimui. Taip pat egzistuoja ir kiti, su problemos sritimi mažai susiję parametrai, kaip greitis, prieinamumas (nemokamas ar komercinis), panaudojamumas, dokumentacija, palaikymas. [NM09]

3.2. Agentų technologijos standartai ir modeliai

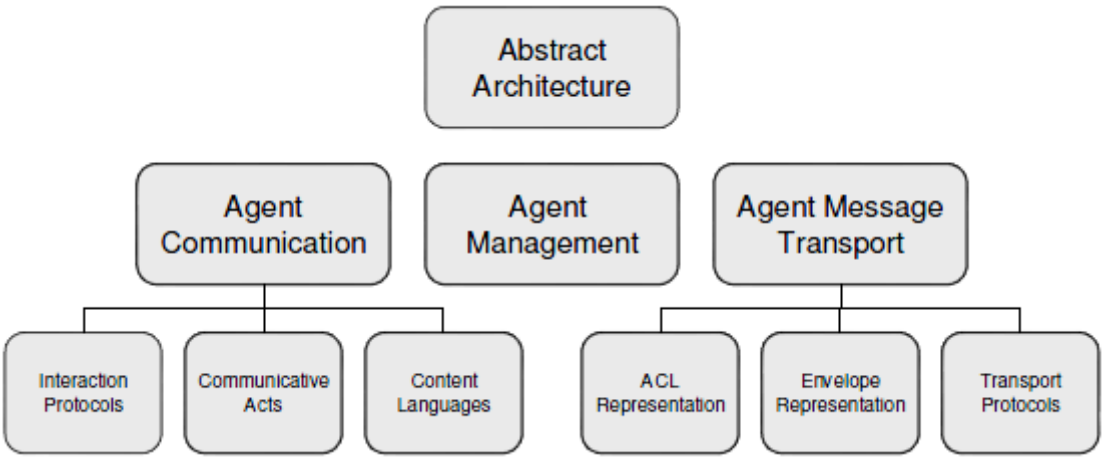
Viena pirmųjų plačiai žinoma problema agentų sistemų kūrime buvo standartų nebuvimas. Nesant vieningam standartui, kūrėjai spręsdavo tas pačias problemas skirtingais metodais, todėl agentų sistemų sudėtingumas išaugdavo, kadangi skirtingi taikomi nauji metodai reikalaudavo papildomo laiko jų perpratimu, todėl standartų būtinumas buvo akivaizdus. Šiuo metu yra du stipriai vyraujantys standartai, tai yra OMG grupės MASIF ir FIPA organizacijos standartas. Pirmasis buvo sukurtas 1997 metais, antrasis taip pat 1997 metais. FIPA pateiktas standartas apibrėžia agentų platformos architektūrą ir tarpusavio sąveikavimas tarp tarpinių[4] platformų. Papildomi MASIF standartai apibrėžia bazinius agentų mobilumo principus.

3.2.1. FIPA Standartas

Svarbus pamatas tarpinėms agentų platformoms yra specifikacijos išleistos Foundation for Intelligent Physical Agents (FIPA). FIPA yra ne pelno IEEE kompiuterijos bendruomenės organizacija, kurios tikslas yra standartų kūrimas programinių agentų tarpsąveikavimui nevienarūšiuose tinkluose. Ši organizacija buvo aktyvi nuo 1996 iki 2005 metų, veikė Šveicarijoje. FIPA organizacijos pateikiamas modelis[5] yra siūlymas kaip kurti agentų platformą, kad ji būtų suderinama su kitais su FIPA standartu suderinamomis agentų platformomis. FIPA specifikacijoje tik kalbama apie sąsajas, kuriais agentai galėtų komunikuoti. Specifikacija skaidoma į penkias kategorijas: programų dalis, abstrakti architektūra, agentų komunikavimas, agentų valdymas, agentų žinučių transportavimas.

Dirbant prie specifikacijų buvo turima omenyje ne tik pačios tarpinės agentų platformos, bet ir programos. Specifikacijos siejamos su programomis pateikia sistematišką galimų problemų sprendimo sričių pavyzdį, kuriuose naudojama agentų technologija. Specifikacijos, susijusios su tarpinėmis platformomis, detalizuoja visus kūrimo blokus, kuriuos turi turėti abstrahuota agentų platformos architektūra.

[4] (angl. middleware)
[5] (angl. Agent Framework Reference Model)



7 pav. FIPA specifikacijų bendra hierarchija (paimta iš www.fipa.org)

Pirmasis standartas išleistas 1997 metais, po metų, t.y. 1998, išleistas naujas standartas, o 2000 išleista paskutinė versija, šiuo metu jau negaliojanti. Komunikacija tarp dviejų agentų remiasi žinučių transporto komponentu, kurio užduotis yra išsiųsti žinutę panaudojanti agentų komunikavimo kalbą ACL.

FIPA 2000 standarto bruožai:

- pagrindinis tikslas – komunikacijos tarpsąveikavimas;
- apibrėžia agentų komunikavimo kalbą ir ontologijas;
- atkreipia dėmesį į agentų mobilumą panaudojant ACL žinutes;
- šiuo metu nebegaliojantis.

FIPA kurti standartai skirti skatinti agentų sistemų ir agentų programų tarpsąveikavimą. FIPA modelyje agentų platforma suprantama kaip keturių komponentų rinkinys: agentų, agentų stebėjimo sistemos[6] (AMS) agento, žinučių transporto sistemos[7] (MTS), katalogo paslaugų modulio[8] (DF) agento. DF ir AMS palaiko agentų valdymą, o MTS užtikrina žinučių perdavimo servisą. FIPA neapibrėžia agentų mobilumo kaip privalomos savybės, tačiau pateikia rekomendacijas kaip platformos kūrėjai galėtų savo modulį mobilumui užtikrinti.

3.2.2. MASIF Standartas

Mobile Agent System Interoperability Facility (MASIF) standartas yra skirtas mobiliųjų agentų sistemoms, kurios yra pasiūlytos Object Management Group (OMG) grupės. MASIF

[6] (angl. Agent Monitoring System)
[7] (angl. Message Transport System)
[8] (angl. Directory Facilitator)

standarto pagrindinis tikslas yra nustatyti pamatus, kurie leistų MASIF standartui suderinamoms agentų platformoms atlikti agentų migravimą, netgi heterogeniniuose tinkluose (darant prielaidą, kad platformose naudojama ta pati programavimo kalba). Standartas išleistas 1997 metais.

OMG MASIF (1997):

- pagrindinis tikslas – tarpsąveikavimas tarp platformų;
- apibrėžia agentų „agentūrų“ (agentų buveinė) interfeisus;
- apibrėžia tinklo perdavimo protokolą;
- susijęs su Corba, ir galbūt dėl to nebuvo plačiai priimtas.

Tam, kad pasiekti agentų mobilumą mišriose sąveikose, MASIF standarte apibrėžia šias sritis: agentų valdymas, agentų perkėlimas, agentų ir platformų vardų suteikimas, agentų sistemos tipo ir vietos sintaksė. Agentų valdymas rūpinasi agentų gyvavimo ciklo kontrole, įskaitant ir tuos agentus, kurie yra nutolusiose platformose. Agentų valdymas apibrėžia standartizuotus interfeisus, agentų kūrimui ir sunaikinimui, taip pat ir jų darbo sustabdymui ir pratęsimui. Agentų perkėlimu sprendžiamas agentų mobilumo klausimas, kuriuo siekiama užtikrinti bendrą infrastruktūrą, kuria naudojantis agentas galėtų laisvai keliauti tarp skirtingų platformų. Pagrindinis to reikalavimas yra nutolusių agentų vietos nustatymas. naudojantis platformų vardais. Todėl standartizuojama agentų ir platformų vardų sintaksė ir semantika. Pavadinimų sintaksės standartizavimas užtikrina, kad platformos galės surasti viena kitą.

MASIF standarte mobiliųjų agentų kontekste yra aptariamas saugumas, kuomet agentų sistema galėtų apsaugoti savo resursus nuo atkeliaujančių į sistemą agentų. Tokiu atveju sistema turi identifikuoti ir patikrinti atkeliaujančio agento leidimus. Tai įgalina kreipimųsi kontrolę ir agentų autentifikaciją ir autorizaciją.

3.2.3. MASIF ir FIPA standartų palyginimas

MASIF standartas yra paremtas agentų platformomis ir įgalina agentus migruoti tarp jų. FIPA specifikacija paremta nuotoliniais komunikavimo servisais. Pirmasis standartas paremtas mobiliaisiais agentais, keliaujančiais tarp agentų sistemų per Corba interfeisus ir neatkreipia dėmesio į komunikaciją tarp agentų. FIPA standartas susitelkia ties į agentų bendravimą panaudojant kalbas ar ontologijas, bet nepasako daug apie mobilumą. FIPA pasiūlo sprendimą agentų bendravimo paradigmai, kaip natūralų komunikavimo ir kooperacijos būdą, ir yra labiau tinkamas su kitomis dirbtinio intelekto technologijomis.

MASIF standartas paremia mobiliųjų agentų paradigmą, kas labiau tinkama kuomet dinamiškumas reikalingas. Buvo mėginimas apjungti šiuos du standartus. Tai buvo mėginama atlikti su FIPA 2000 specifikaciją, kurioje sprendžiamas agentų mobilumo klausimas. ACTS (Advanced Communications, Technologies and Services) Europos tyrimų programos dalis buvo išnagrinėti MASIF ir FIPA standartų integraciją. Šiuo metu, kai kurios agentų platformos, pvz. JADE, yra suderinamos su abejais standartais, kas išsprendžia mobilumo ir komunikavimo klausimus ir palengvina agentų programų projektavimą ir kūrimą.

1 lentelė. Standartų aptariamos sritys

	FIPA	MASIF
Valdymas	Taip	Taip
Komunikacija	Taip	Ne
Mobilumas	Ne	Taip
Saugumas	Ne	Taip

3.2.4. BDI – Belief-Desire-Intention modelis

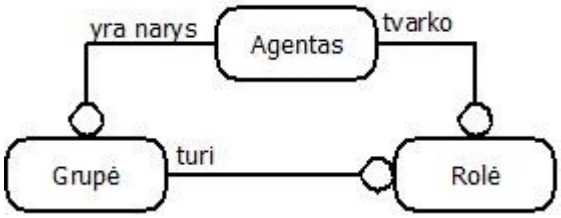
BDI – programinės įrangos kūrimo modelis, sukurtas sumaniesiems agentams. Iš esmės, modelis pateikia mechanizmą kaip atskirti planavimą nuo vykdymo. BDI agentų architektūros panaudoja tikėjimo, troškimo, tikslo koncepciją ir planuoja, kad agentų kūrimas, sukurti platformomis įgyvendinančiomis šias architektūras, pareikalaus subalansuoto laiko sugaišto apgalvojimui, ką agentas turės padaryti ir planavimui kaip reikės tai padaryti BDI modelis apjungia keletą atributų [TIS06]:

- pagrįstas Bratmano žmonių samprotavimo teorija;
- realizuotas ne vieną kartą;
- panaudotas sudėtingose aplikacijose;
- BDI teorija griežtai formalizuota.

BDI domimasi bent šiose tyrimų srityse: žmogaus elgesio modeliavime, BDI teorijos ir realizacijų kūrime, sudėtingų ir patikimų agentų programų kūrime.

3.2.5. AGR – Agent-Group-Role modelis

Naudojantis šiuo modeliu, į organizaciją galima suprasti kaip sistemą, kurioje darbui atlikti agentai sąveikauja tarpusavyje, remiantis grupėmis, rolėmis ir ryšiais tarp jų. AGR modelis paremtas trimis primityvais: agentu, grupe ir role, kurie yra tarpusavyje sujungti ir negali būti apibrėžti kitais primityvais.



8 pav. AGR modelis

Agentas yra aktyvi, komunikuojanti esybė, turinti rolę grupės viduje. Agentas gali turėti kelias roles, ar priklausyti kelioms grupėms. Tačiau modelis turi trūkumą, kadangi neįtraukia mobilumo problemos, kuomet agentas migruodamas fiziškai tampa nariu kitoje grupėje, kitoje vietoje.

3.3. Agentų migracijos tipai

Skiriami du migracijos tipai: silpnoji ir stiprioji. Agentų sistemose labiau paplitusi silpnoji dėl technologinių kliūčių, kaip kad Java kalboje negalima išsaugoti ir perduoti steko ir programos skaitliukų[9], todėl Java programavimo kalba paremtose agentų platformose sutiksime tik silpnąją migraciją.

3.3.1. Silpnoji migracija

Šis migravimo būdas yra paprastesnis, reikalauja mažiau resursų. Agento vykdymas visuomet pradedamas nuo pradžių, todėl svarbu prieš migravimą išsisaugoti duomenis. JADE sistemoje šio tipo migravimas reikalauja, kad agentas būtų realizuotas kaip mašina su baigtine būsena, jei norima kad agento vykdymo rezultatai būtų išsaugoti.

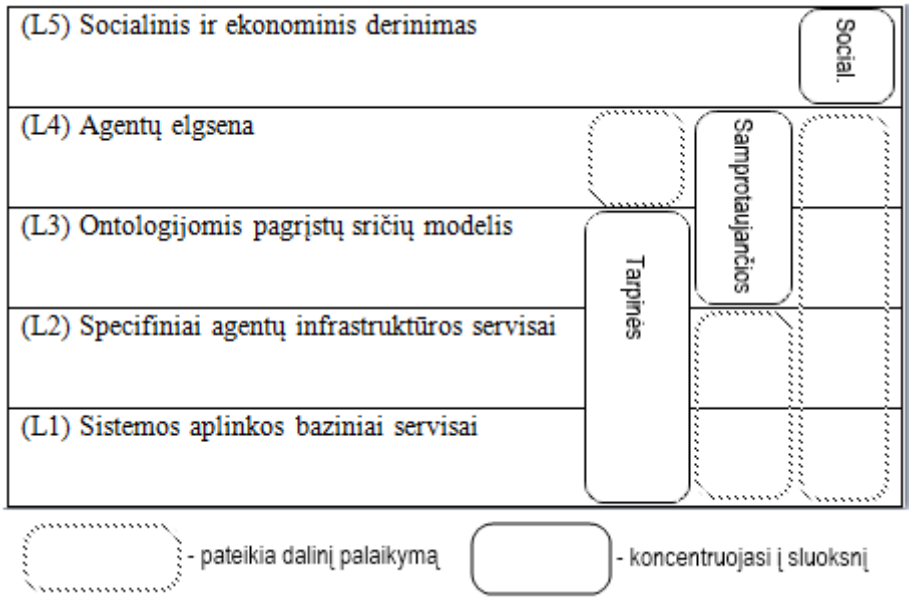
3.3.2. Stiprioji migracija

[9] (angl. counters)

Stipriosios migracijos metu perduodamas ne tik kodas ir duomenys, bet ir agento vykdymo būsena (stekas ir skaitliukai). Tai ne visuomet pavyksta padaryti dėl kalbos trūkumų ar kitų reikalavimų. Agentui migravimus, jo darbas pratęsiamas būtent nuo tos vietos, kur ir buvo sustabdytas prieš migravimą. Šio migravimo tipo sudėtingumą didina tai, kad jis reikalauja priėjimo prie vidinių agentų vykdymo skaitliukų ar parametrų, kurie gali būti prieinami tik operacinei sistemai ir gali būti architektūriškai priklausomi, kas daro tokį migravimą sudėtingesnį tarpplatforminėms[10] agentų sistemoms.

3.4. Agentų platformų kategorijos

Agentų platformos gali būti suskirstytos pagal tai, kokius lygmenis jos pabrėžia. [BPL05] išskiria penkis lygmenis. Jie išvardinami 3 paveiksle. Kad agentų platforma būtų pilnavertė tarpinė platforma, ji turi užtikrinti pirmus tris lygius L1-L3. Tarpinės platformos pateikia bazines priemones atviroms, tarpusavyje sąveikaujančioms sistemoms, kadangi jų tikslai yra tarpsąveikavimas, agentų valdymas ir komunikacija. Tačiau ne visi agentų kūrimo aspektai gali būti palaikomi vienodai gerai. Dauguma tarpinių platformų neužtikrina sudėtingo agentų samprotavimo, o remiasi paprastu užduotimis paremtu modeliu, kuris leidžia komponuoti sudėtingą agentų elgseną iš smulkių užduočių.



9 pav. Skirtingų kategorijų apimami sluoksniai [BPL05]

[10] (angl. cross-platform)

Samprotaujančios sistemos remiasi trečiu ir ketvirtu lygiu ir taiko vidinę samprotavimo architektūrą tam, kad nustatytų agento veiksmus iš realaus pasaulio žinių. Kadangi samprotavimo mechanizmas gali būti labai painus, todėl pirmųjų trijų lygių palaikymas yra labai skirtingas ir priklauso nuo poreikių.

Samprotaujančios ir tarpinės platformos nepateikia priemonių agentų bendruomenių programavimui, tai atlieka socialinės platformos. Socialinėse platformose tai užtikrinama organizacinės architektūros įgyvendinimu, kaip kad MadKit platformoje AGR modelio įgyvendinimas. Svarbus klausimas šio tipo platformose yra tai, ar įgyvendinama architektūra priklauso nuo vidinės architektūros agentų elgsenos koncepcijos, nes nuo to priklauso kokio sudėtingumo struktūros, skirtos agentų valdymui, gali būti įgyvendintos. Kita vertus, tokios architektūros ir platformos pritaikymas gali būti apribotas agentų tipų, kuris yra nepageidaujamas atvirose sistemose.

3.4.1. Tarpinės platformos

Paskirstytų skaičiavimų sistemose tarpinės[11] platformos suprantamos būtent kaip tai nuo tinklo rūšies, tipo ir t.t. nepriklausanti programinė įranga, „patalpinta“ tarp programos, operacinės sistemos, tinklo transporto sluoksnių, kurių tikslas yra palengvinti bendrą apdorojimą. Pavyzdžiu gali būti žinučių perdavimo mechanizmai, paskirstytas transakcijų apdorojimas, objekto užklausų tarpininkas (ORB), nuotolinis procedūrų kvietimas (RPC) ir pan.

Kadangi agentų technologija remiasi paskirstytų skaičiavimų koncepcijomis ir technologijomis, pasirinkta tarpinė įranga yra taip pat svarbi kaip ir kitos agentų sistemos kūrimo dalys, kaip projektavimo įrankiai ir kt. Agentų platformose, žodis „tarpinė“ naudojamas nurodo bendrinius servisus, tokius kaip tarpusavio komunikacija (žinučių perdavimas), naudojamus standartus ar mobilumo principus. Vien todėl, kad pačios agentų platformos apibrėžiamos kaip tarpinės, iš jų reikalaujama bendrinio praktikų, kurios turi būti standartizuotos. Agentai gali būti matomi kaip komponentai naudojantys tuos standartinius pateikiamus tarpinės platformos servisus. Kita vertus, tarpinės agentų platformos, įgyvendinančios tuos standartus, turi supaprastinti ir pagreitinti plečiamų, daugelio vartotojų ir atsparių sutrikimams programų kūrimą. [BPL05]

[11] (angl. middleware)

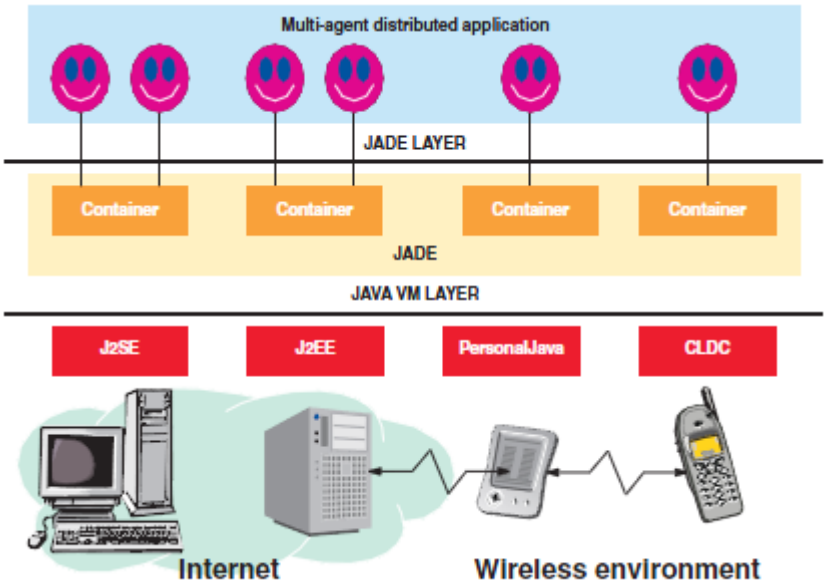
Tokie esminiai standartai ir yra FIPA, kurio įgyvendinimas padeda tarpsąveikauti skirtingoms tarpinėms sistemoms ir papildomas MASIF standartas, pateikdamas koncepciją agentų mobilumui. O šių abiejų standartų įgyvendinimas leidžia kurti agentus ir jų sistemas tenkinančius bendresnius agentų reikalavimus.

3.4.1.1. JADE

JADE (Java Agent DEvelpment framework) yra agentų kūrimo platforma parašyta Java programavimo kalba. JADE platformą sukūrė Telecom Italia Lab 1998 metais, o nuo 2000 vasario išleista kaip atvirojo kodo. JADE skirta multiagentinių sistemų, paremtų P2P komunikavimo architektūra, kūrimui. Naujausia šios platformos versija yra 4.0.1, išleista 2010 metų liepą. JADE platforma supaprastina multiagentinių programų kūrimą pateikdama priemones atitinkančias FIPA 2000 specifikaciją. Kartu su platforma pateikiami grafiniai įrankiai, kurie padeda agentų kūrimo, derinimo ir naudojimo metu. JADE platforma gali būti panaudota įvairiuose įrenginiuose, turinčiuose ryšį su internetu, kurie palaiko Java, įskaitant mobiliuosius telefonus, asmeninius ir nešiojamus kompiuterius ir kt.

JADE vykdymo aplinka yra sudarytą iš konteinerių, kurių viduje yra laikomi agentai. Konteineriai savo ruožtu sudaro platformą. Platforma paslepia po ja esančią techninę ir programinę įrangą nuo agentų ir programuotojų, taip supaprastindama agentų kūrimą, derinimą ir naudojimą. JADE yra suderinama su J2ME, J2EE ir CLDC. Nedidelis atminties naudojimas yra svarbus mobiliesiems įrenginiams. Agentas platformoje yra sudarytas iš skirtingų vienu laiku ir be pirmenybės vykstančių elgsenų[12], kurios gali būti pridedamos dinamiškai, vykdymo metu.

[12] (angl. behaviour)



10 pav. JADE architektūra [BCPR03]

Kiekvienas JADE konteineris sudarytas iš agentų stebėjimo sistemos (AMS) agento, nuotolinio stebėjimo interfeiso[13] (RMI) agento, katalogo paslaugų modulio (DF) agento (FIPA modelio standarto dalys). Agentai gali surasti kitus agentus pasinaudodami DF agentu ir komunikuoti tarpusavyje naudodami P2P paradigma. JADE sistemoje naudojama asinchroninė žinučių perdavimo technika, kuri yra dažniausiai naudojama tarp „netvirtai“ surištų komunikavimo galų. Žinučių keitimasis tarp agentų atitinka FIPA apibrėžtą agentų komunikavimo kalbą ACL. JADE taip pat palaiko keleto užduočių vienalaikį vykdymą toje pačioje Java gijoje – tai leidžia lengviau atlikti didėjančios apimties užduotis, kurios yra apribotos vykdymo aplinkos resursų.

JADE platforma palaiko agentų mobilumą, kurio metu migruojamas ne tik kodas ir duomenys, bet ir būsena, tačiau tai nėra laikoma stipriąja migracija, kadangi Java programavimo kalboje stekas ir programos skaitliukai negali būti išsaugoti ir perduoti. JADE naudojamas nuosavas autentifikacijos mechanizmas, patikrinantis agento teises, užtikrina agentų saugumą. JADE objektų valdytojas[14] pateikia susijungimų autentifikaciją, naudotojo validavimą ir RPC žinučių šifravimą.

Kadangi ši platforma yra suderinama su FIPA standartu, tarpsąveikavimas yra naudinga savybė, kuomet agentai sukurti JADE platformoje gali bendrauti su kitais agentais, naudojantis tuo pačiu standartu. JADE palaiko CNP protokolą, kuriuo supaprastinamos sudėtingos multiagentinių programos užduočių vykdymas. Ontologijų palaikymas „išplečia“

[13] (angl. Remote Monitoring Interface)
[14] (angl. Object Manager)

agentų žinias, o integravimas su Jess, leidžia JADE agentams argumentuotai panaudoti žinias, apibrėžtas deklaratyviomis taisyklėmis. Pagrindinis platformos trūkumas yra tas, kad ji nėra orientuota į agentų migravimą, o daugiau vysto kitas multiagentinių sistemų savybes. JADE agentų įgimtas migravimas galimas tik platformos viduje tarp konteinerių, tuo tarpu norėdami migruoti agentus tarp platformų, reikia panaudoti platformos priedą Inter-Platform Mobility Service, sukurtą Savarankiškojo Barselonos Universiteto tyrėjų. Tokia migravimo ideologija panaši į Grasshopper agentų platformos regionus. JADE sistemos priedas WSIG, leidžia agentams elgtis kaip klientui arba serveriui tinklo servisų[15] programose.

3.4.1.2. Aglets

Aglets yra programinių agentų kūrimo aplinka, kuri specializuojasi mobiliuosiuose agentuose. Po ilgos pertraukos, nuo 2002 metų, kuomet buvo išleista 2.0.2 versija, 2010 metų liepą projektas buvo atnaujintas išleidžiant 2.5 Alpha platformos versiją. Ši platforma dar buvo vadinama ASDK (Agents Software Development Kit). Platforma yra atvirojo kodo. Su ja pateikiamas grafinis vartotojo interfeisas agentų kūrimui. Jis pagrinde susideda iš dviejų paketų: Aglets Building Environment (ABE) kūrimo aplinkos ir Aglets Workbench. Aglets Workbench paketas skirtas autonominių agentų kūrimui. ABE apima Aglets programavimo interfeisą[16], Aglets serverį žinomą kaip Tahiti, Fiji aplinką, skirtą integruoti agentų platformą į interneto puslapius panaudojant Java Applets technologiją, dokumentacija ir pavyzdžiais. Aglets agentas iš esmės yra Java objektas, susidedantis iš dviejų dalių – Aglet Core branduolio ir Aglet Proxy tarpininko. Branduolyje saugomi vidiniai agento metodai ir kintamieji, o tarpininkas veikia kaip interfeisas su branduoliu, apsaugodamas jį nuo kenksmingų įsikišimų iš išorės. Tahiti yra programa, kuri veikia kaip serveris Aglets agentams. Ji pateikia grafinį interfeisą ir leidžia kurti, naikinti, stebėti, persiųsti agentus, kai to prireikia, taip pat yra galimybė agentams suteikti prieigos teises serveryje. Tam, kad agentas migruotų į nuotolinį tašką, jame turi būti paleistas Tahiti serveris. Aglets palaiko agentų mobilumą ir iš anksto apibrėžtą judėjimą tinkle, vadinamą maršrutu[17]. Aglets palaiko silpnąjį migravimą, tačiau tai ribojama iš Tahiti serverio pusės ir Java programavimo kalbos. Aglets agentų platforma yra suderinama su MASIF standartu, o migravimo mechanizmas yra įgyvendintas panaudojant Java Sockets. Žinutės gali būti siunčiamos tiek sinchroniškai, tiek

[15] (angl. Web Service)
[16] (angl. Application Programming Interface)
[17] (angl. Itinerary)

asinchroniškai. Agentų persiuntimo protokolas (ATP[18]) kartu su Java agentų persiuntimo ir komunikavimo interfeisu (J-ATCI) taip pat leidžia įgyvendina šias savybes sistemoje. Nors Aglets platforma yra pripažinta, tačiau jos saugumo mechanizmas yra silpnas, kadangi saugumas realizuotas apribojant agentų migravimą į Aglet platformos serverius. Dėl saugumo apribojimų, agentų būsena, negali būti išsaugota kitame serveryje. Sistemos plečiamumas taip pat yra problema, kadangi Aglets platforma nėra suderinama su jokia kita sistema ir negali tarpsąveikauti, o agentai būtinai turi veikti Tahiti serveryje.

Aglets naudoja „proxy“, kad pasiektų nutolusius agentus, tačiau tie „proxy“ nėra dinaminiai, ir nuotoliniam agentui pakeitus vietą, nebegalima panaudoti to paties „proxy“, programuotojas turi užtikrinti, kad nuotolinių agentų „proxy“ būtų atnaujinami prieš panaudojant juos iš naujo. Kadangi kiekvienam agentui skiriami viena gija, reikia vengti ilgai trunkančių užduočių, kas galėtų sutrukdyti apdoroti įvykiams, kaip kad žinučių gavimui. Vienos gijos naudojimas gali privesti prie agentų užsiblokavimo, jei du agentai vienas kitam išsiųstų sinchronines žinutes tuo pačiu metu, tokiu atveju ateinanti žinutė negalėtų būti apdorota, kol kitas agentas negavo siųstos žinutės. Platforma nepalaiko nuotolinių agentų kvietimų ar vartotojui suprantamų agentų identifikatorių. Tai pat platformos trūkumu galima įvardinti ir tai, kad negalima nurodyti metodo migravimo operacijoms, kurį atliktų agentas baigęs migravimą (iš anksto nurodytas veiksmas atliekamas visada atliekamas atvykus agentui į platformą).

3.4.1.3. Grasshopper

Grasshopper agentų platforma sukurta 1999 metais. Kartu su Aglets, Concordia ir Voyager, ji buvo tarp populiariausiųjų agentų platformų. Paskutinė versija yra 2.2.4, išleista 2003 sausį, kuomet tapo komercinė Enago Mobile dalis ir šiuo metu platformos tobulinimas yra nutrauktas. Rasta laisvai prieinama versija buvo tik 1.2.2, išleista dar 1999 metais. Versija 1.2.2 turėjo patogią, įprastą Windows operacinei sistemai, instaliacijos vedlį, tačiau instaliavus ją, platformos išmėginti nepavyko, kadangi ši nepasileido. Kituose šaltiniuose nurodoma, kad sistema yra suderinama su FIPA ir MASIF standartais. Ji turi nuosavą JADE sistemos konteinerių atitikmenį – regionus. Pateikiamas grafinis interfeisas, kuriuo galima valdyti agentus, regionus. Apibrėžus regionus, galima naudotis dinaminiais „proxy“, kas labai supaprastintų komunikavimą su nutolusiais agentais

[18] (angl. Agent Transfer Protocol)

Nepaisant, kad sistema turi nemažai naudingų savybių, tačiau tokios sistemos naudojimas yra labai pavojingas, kadangi ji nebebus tobulinama. Regionų serveris, kurį naudoją platformą, turės įtakos sistemos plečiamumui, kadangi kaskart prieš pat dinaminių „proxy“ naudojimą, jis atnaujina jas. Taip pat įvardinamas trūkumas, kad kreipimasis į agentą, kuris tuo metu migruoja, gali sukurti kopiją agento gimtame serveryje. Panašiai kaip ir Aglets platformoje, agento kelionės pabaigoje, iš anksto nurodytas veiksmas yra įvykdomas.

3.4.1.4. Tryllian

Tryllian agentų platforma sukurta 2001 metais, to paties pavadinimo kompanijos. Paskutinė šios platformos versija 3.2 išleista 2005 metų lapkritį. Agentų platforma paremta jutiminiu-samprotaujančiu veiksmų mechanizmu. Šis mechanizmas leidžia programuotojams apibrėžti reaktyvų (įeinančių pranešimų pagrindu) ir proaktyvų agentų elgesį. Tryllian siūlo užduotimis paremtą programavimo modelį, o komunikacija užtikrinama žinučių, atitinkančių FIPA standartą, apsikeitimu. Siunčiant pranešimą, platformą neužtikrina vietos „permatomumo“, todėl reikia žinoti agento, kuriam bus siunčiamas pranešimas, buvimo vietą. Pačios platformos užduotimis paremtas ir asinchroninis modelis yra kiek sudėtingas naudoti, kuomet programuotojas yra pripratęs prie procedūrinio programavimo. Pati platforma yra labai lanksti, pateikdama labai daug nustatymų, kuriuos galima keisti, kas daro jos naudojimą netgi sudėtingą. Tryllian platformoje metodų kvietimas yra netradicinis ir neįprastas dėl savo modelio. Taip pat, nors ir nėra labai svarbu, tačiau sinchroninis žinučių siuntimas nėra įmanomas. Saugumas platformoje užtikrinamas panaudojant Java saugumo modelį su sertifikatais ir teisėmis. Programuotojams saugumo užtikrinimas sumažintas iki to, kad reikia pasirašyti agentą su jar-signer įrankiu, kuris pateikiamas su Java JDK.

3.4.1.5. Zeus

Zeus yra integruota kūrimo aplinka (IDE)[19], skirta greitam tarpusavyje bendraujančių agentų kūrimui. Šią sistemą sukūrė Advanced Applications & Technology Department of British Telecommunication laboratorija. Ji yra atviro kodo, sukurta su Java. Zeus yra suderinama su FIPA standartu. Zeus užtikrina pagrindinį agentų funkcionalumą ir turi priemones agentų veiksmų planavimui ir numatymui. Zeus sistema pateikia įrankius, kuriais

[19] (angl. Integrated Development Environment)

naudojantis galima modeliuoti, kurti ir organizuoti agentų sistemas. Zeus grafiniai įrankiai ir papildomi komponentai kaip agentų ataskaitų įrankis, statistinis įrankis, agentų ir agentų bendruomenės peržiūros įrankis, leidžia stebėti ar analizuoti kuriamas sistemas. Komunikacija tarp agentų gali būti atlikta arba ACL kalba, arba panaudojant jos pirmtakę KQML. Komunikavimo saugumas yra užtikrinamas panaudojant viešą raktą ar skaitmeninio parašo technologijas, tačiau ZEUS platforma nepalaiko agentų mobilumo.

3.4.2. Samprotaujančios platformos

Samprotaujančios platformos pagrįstos specifinėmis vidinėmis agentų architektūromis. Būtent tos vidinės architektūros yra sugalvotos realizuoti agentų samprotavimo ir agento veiksmų nustatymo (to reikalauja agento proaktyvumas) mechanizmus. Šios architektūros gali būti skaidomos į „reaktyvias“[20], „svarstančias“[21] ar hibridines. „Svarstančiose“ architektūrose reikalingas panaudojamas pasaulio modelis ir sprendimai priimami remiantis loginiu (arba bent jau pseudologiniu) pagrindimu, kuomet reaktyviose architektūrose nenaudojamas toks modelis, o veiksmų nustatymui ir priėmimui panaudojama elgesių hierarchija, kur žemesnio lygio veiksmas atitinka primityvesnį veiksmą už aukščiau esantį (tokia architektūra tinkama roboto judėjimui). Hibridinės architektūros stengiasi perimti ir panaudoti geriausias praktikas iš abiejų architektūrų. [BPL05]

3.4.2.1. JACK

JACK platforma yra komercinis produktas sukurtas Agent Oriented Software. Pati platforma yra paremta BDI modeliu ir agentų kūrimui naudoja nuosavą JACK agentų kalbą (JAL[22]). JAL yra Java kalbos praplėtimas, pateikiantis kai kurias BDI koncepcijas ir keletą kalbos naujovių, kaip kursoriai ir loginiai kintamieji. Agentas JACK platformoje yra JAL failų rinkinys, kuriuose yra paties agento kodas, jo planai, „tikėjimų“ bazė ir įvykiai. Norint paleisti JACK agentą, pirmiausia reikia „prekompiliuoti“ JAL failus į Java išeities tekstus ir tuomet sukompiliuoti į Java „baitkodą“.

JACK sistema, būdama verslo lygio produktu, pateikia gausybę dokumentacijos ir įrankių. Vienas pagrindinių įrankių yra pateikta IDE aplinką, kurios pagalba galima

[20] (angl. reactive)
[21] (angl. deliberative)
[22] (angl. JACK Agent Language)

projektuoti ir kurti agentus. Pateiktoje IDE aplinkoje pateiktas primityvus projekto valdymas, joje taip pat galima vykdyti kodą tiesiogiai. Grafinis plano redaktorius leidžia vizualiai kurti planus ir stebėti jų vykdymo laikus.

3.4.3. Socialinės platformos

Grupinio elgesio palaikymas socialinėse[23] platformose yra jų pagrindinis išskirtinumas iš kitokių agentų architektūrų, kas labai svarbu daugelio agentų sistemose, kur agentų komunikavimas yra neatsiejamas dalykas. Komandinis darbas įtraukia kitokias struktūras, todėl tokiose sistemose naudojamos skirtingos nuo samprotaujančių ir tarpinių elgesio teorijos ir architektūros. Vienas iš mechanizmų agentų komandoms įgyvendinti yra Agentas-Grupė-Rolė (AGR) modelis. Agentas yra suprantamas kaip aktyvus, komunikuojantis vienetas ir turintis rolę grupės viduje. Grupė, savo ruožtu, yra laikoma agentų aibe besidalinančiu kokia nors savybe. Vienas pagrindinių AGR modelio principų yra tai, kad organizaciniame lygyje jokie agento sugebėjimai ir jo apibūdinimai nėra naudojamos, kas atskiria šį modelį nuo kitų architektūrų ir leidžia tiek paprastiems, tiek sudėtingiems agentams būti toje pačioje organizacinėje struktūroje. Tai pasiekiama todėl, kad pačių agentų aprašymas jau pats savyje išskiria ir parodo agentų tarpusavio skirtumus ir net nenorom verčia per nauja burti į kitas grupes, dėl ko per daug skirtingų agentų struktūra gali „griūti“ ir ją reikės permąstyti iš naujo.

3.4.3.1. MadKit

MadKit yra socialinė multiagentinė platforma, sukurta remiantis organizaciniu modeliu. Ji pateikia pagrindines priemones, kurių pagalba užtikrinamas agentų gyvavimo ciklas, žinučių perdavimas, pasirinkimas tarp agentų architektūrų ir komunikavimo kalbų ir įgalina aukštą pritaikomumą. MadKit naudoja P2P paremtą komunikavimo mechanizmą ir leidžia programuotojams kurti paskirstytas programas panaudojant agentų programavimo principus. Paskutinė stabili versija yra 4.2.0, platformą kuria grupė žmonių, vadovaujamų Ferber. Šiuo metu yra kuriama 5 platformos versija ir yra išleista 5 Alpha 8 versija.

MadKit platforma yra modulinė ir plečiama, paremta AGR organizaciniu modeliu, kurt agentai yra grupės nariai ir turi savo priskirtas roles jose. AGR modelis yra nepriklausomas

[23] (angl. social)

nuo vidinio agentų modelio, jis leidžia panaudoti įvairias agentų architektūras ir komunikacijos priemones ir standartus.

MadKit platforma yra realizuota remiantis trimis principais:

- sistema paremta mikrobranduolio architektūra, kuri pateikia pagrindinius agentų ir grupės valdymo ir komunikavimo servisus
- servisai platformoje yra realizuoti kaip agentai
- pateikiamas komponentinis modelis tam, kad atvaizduoti agentus grafiškai pačioje platformoje

Dėl trečios savybės pati platforma atrodo kaip atskiros operacinės sistemos langas. Su sistema taip pat yra pateikiama įrankių skirtų agentų stebėjimui ir derinimui. Pati platforma buvo panaudota agentų simuliavimo aplinkos TurtleKit kūrime, taip pat ir SEdit, struktūrinių diagramų projektavimui ir atvaizdavimui. Saugumo užtikrinimui naudojamas agentas, pavadintas AgencyKeeper, kuriuo užtikrinamas saugumas tiek agentams (statiniams ir mobiliems) tiek ir platformai.

3.5. Kokybinė agentų platformų analizė

Kokybinei agentų platformų analizei atlikti daugiausia dėmesio, kur buvo įmanoma, skirta agentų platformos techninei literatūrai nagrinėti, pateikiamai platformos gamintojo puslapyje kartu su agentų platforma. Ne visų agentų platformų techninė literatūra buvo skirta tai pačiai platformos versijai, kurią galima buvo parsisiųsti su platforma, pavyzdžiui Jack naujausia išleista versija yra 5.4, o pateikiama dokumentacija yra skirta 5.3 versijai ir yra 2008 metų. Todėl savybėms nagrinėti ir vertinti buvo naudota dokumentacija, tokia kokia pateikta gamintojo, arba jei dokumentacijos nepavyko surasti, tuomet naudoti kiti šaltiniai.

Nagrinėjant įvairias agentų platformas, literatūrą apie jas, rasti kriterijai, apie kuriuos kalbama ne viename šaltinyje. Išrinkus daugiau tokių bendrų kriterijų, juos surinkau į vieną lentelę ir nustačiau visoms iš aprašytoms agentų platformoms. Aprašytų agentų sistemų pagrindinių savybių lentelė pateikiama 2 lentelėje. Lentelėje vertinamos šios savybės: agentų platformos licencija (komercinė, atviro kodo ir kt.), palaikomi standartai, žinučių siuntimo, komunikavimo technika, bendras saugumo įvertinimas, agentų migravimo mechanizmas, „silpnas“ ar „stiprus“ agentų mobilumas, pateikiami įrankiai, agentų sistemos architektūros kategorija (tarpinė, socialinė ar samprotaujanti), paskutinė išleista ir prieinama galutinė ir/ar negalutinė versija.

Iš visų aprašytų agentų platformų, JADE platforma atrodo patraukliausiai. Ji ir buvo pasirinkta kaip kuriamos sistemos pagrindas, kadangi JADE platforma yra atviro kodo, parašyta su Java, suderinama su agentų standartais, pakankamai aprašyta, yra literatūros, dokumentacijos, API aprašų, turi priedų praplečiančių jos standartinį funkcionalumą, palaiko agentų mobilumą. Jack yra komercinė platforma, todėl kitos sistemos kūrimas paremtas ja gali būti sudėtingas ir tobulinimas neįmanomas ateityje,. Zeus nepalaiko agentų mobilumo, tačiau yra suderinama su FIPA standartais. Aglets nėra suderinama su FIPA standartais ir jos saugumo mechanizmas ir silpnas. Grasshopper yra morališkai pasenusi. Tryllian buvusi kai p komercinė platforma, 2005 metais išleista kaip atvirojo kodo, tačiau jos nei instaliacijos, nei išeities tekstų gauti nepavyko. MadKit nepalaiko jokio standarto. JADE agentų platforma savo savybėmis yra geriausiai subalansuota iš visų bandytų ir aprašytų šiame darbe.

2 lentelė. Agentų platformų palyginimas

	JADE	Aglets	Grasshopper	Tryllian	Zeus	Jack	MadKit
Licencija	Atviro kodo	Atviro kodo	Atviro kodo	Atviro kodo	Atviro kodo	Komercinė	Atviro kodo
Modelis	BDI	Įvykiais paremtas	Procedūrinis	Užduotimis paremtas	Įvykiais paremtas	BDI	AGR
Palaikomi standartai	FIPA, MASIF	MASIF	MASIF, FIPA	FIPA	FIPA	FIPA (per išplėtimą)	–
Komunikacijos technika	Asinchroninė	Sinchroninė, asinchroninė	Sinchroninė, asinchroninė	Sinchroninė, asinchroninė	Asinchroninė	Asinchroninė	Asinchroninė
Saugumo vertinimas	Geras	Vidutinis	Geras	Geras	Geras	Prastas	Geras
Agentų mobilumas	Silpnas + būsena, bet ne stekas	Silpnas	Silpnas	Silpnas	–	Silpnas	Silpnas
Migravimo mechanizmas	RMI	Socket	RMI	nežinoma	–	nežinoma	nežinoma
Architektūra	Tarpinė	Tarpinė	Tarpinė	Tarpinė	Tarpinė	Samprotaujanti	Socialinė
Pateikiami įrankiai	Administravimo, derinimo	Administravimo, derinimo	Administravimo, derinimo	Kūrimo aplinka, derinimo	Kūrimo aplinka, administravimo	Kūrimo aplinka, derinimo	Administravimo, derinimo
Versija	4 (2010 m.)	2.5 Alpha (2010 m.)	2.2.4 (2003 m.)	3.2 (2005 m.)	2.0e (2006 m.)	5.4c (2011 m.)	Stabili 4.2, 5 Alpha 8 (2011 m.)

3.6. Kiekybinė agentų platformų analizė

Platformų kiekybinės analizės testų tikslas – išsiaiškinti agentų platformų spartą, todėl sudaryti trys testai, kuriuos atlikus gauti duomenys parodytų, kuri platforma yra spartesnė. Testų metu atliekamas darbas parinktas taip, kad atkartotų pagrindinę platformų agentų darbo specifiką, ir kas ypatingas dėmesys skirtas būsimos sistemos darbo specifikai imituoti. Todėl pagrindiniais testų bruožais išskirti agentų migravimas ir komunikavimas. Agentų migravimas, atliekant testus, buvo vykdomas tame pačiame kompiuteryje. Taip mažiau imituojamas realus agentų platformų darbas, tačiau tokiu būdu pašalinami trukdžiai ir vėlavimai, kurie gali atsirasti dėl tinklo trūkumų, todėl gauti rezultatai yra mažiau paveikti tinklo vėlavimų.

Kiekybinei analizei atlikti pasirinkau kompiuterį, kurio techninės charakteristikos yra šios: Intel Q6600 2,4GHz procesorius, 3GB darbinės atminties ir Windows XP operacinė sistema. Šis kompiuterinis buvo darbo kompiuteris, juo buvo atliekami visi testai. Testai sukurti it atliekami tokia tvarka, kad pirma atliekamas nesudėtingas testas, kurio metu bandomas migravimas, antrajame – pridedamas komunikavimas, o trečiajame teste mėginamas agentų platformos atsparumas dideliems krūviams.

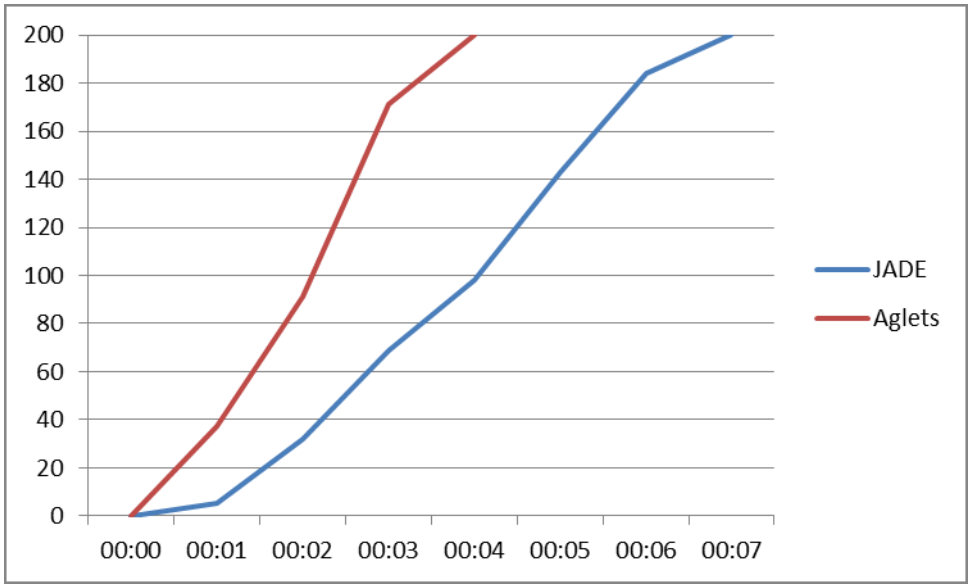
Sukurti trys testai:

1. Dviejų agentų migravimas tarp dviejų agentų platformų, veikiančių tame pačiame kompiuteryje.
2. Šimto agentų migravimas tarp dviejų agentų platformų, veikiančių tame pačiame kompiuteryje.
3. Šimto agentų migravimas tarp dviejų agentų platformų, veikiančių tame pačiame kompiuteryje ir komunikavimas.

Pirmasis testas buvo atliekamas po tris kartus, jo rezultatai suskaičiuojami ir galutiniam įvertinimui imamas trijų testų aritmetinis vidurkis. Antras ir trečias testai dėl rezultatų gausos ir didelio darbo po bandymo buvo atliekami po kartą. Testai buvo atliekami su pasirinktomis dvejomis agentų platformomis – tai Aglets ir JADE. Šios platformos buvo pasirinktos dėl to, kad jos yra parašytos Java kalba, yra tarpinės sistemos ir jų naujausios versijos yra ir išleistos bent per paskutinius du metus. Testų rezultatai buvo skaičiuojami iš „log4j“ bibliotekos išvedamos informacijos į failą, vėliau rezultatai perkelti į Microsoft Excel skaičiuoklę, sugrupuoti bei apdoroti. Pranešimai suskaičiuoti ir grafikai nubraižyti panaudojant Microsoft

Excel skaičiuoklę. Pranešimas į failą išvedamas tuomet, kai agentas pilnai atlikdavo iteraciją, t.y. darbo vienetą, kuris ir buvo skaičiuojamas ir vėliau analizuojamas. Pirmojo ir antrojo testų metų agento iteracija buvo agento migravimas iš vienos platformos į kitą (Aglets agento atveju iš Aglets platformos į kitą Aglets platformą, JADE – iš JADE platformos į JADE platformą). Platformos testams atlikti buvo pakurtos tame pačiame kompiuteryje, o trečiajame teste iteracija yra vienas migravimas į kitą platformą ir vienas komunikavimas su užduotu agentu. Agentai kiekvienos iteracijos metu kaupdavo informaciją apie iteracijų vykdymą, konkrečiai apie tai kokį veiksmą jie atliko ir sistemos laiką. Todėl su didėjančiu informacijos kiekiu, kurį agentas turėjo kaupti kiekvienos iteracijos metu, agentams ir agentų sistemoms atlikti vieną iteraciją prireikdavo didesnių resursų.

Pirmasis testas yra pats paprasčiausias ir skirtas bandymams kaip veikia agentų parinktos platformos, jų neapkraunant dideliu darbu, tačiau išmėginant esmines būsimos kuriamos sistemos veikimo dalis (agentų migravimas). Jo metu agentų užduotis buvo migruoti iš vienos sistemos į kitą, kiekvienam agentui reikėjo atlikti šimtą tokių iteracijų, kad būtų laikoma, kad agentas atliko visą savo darbą.

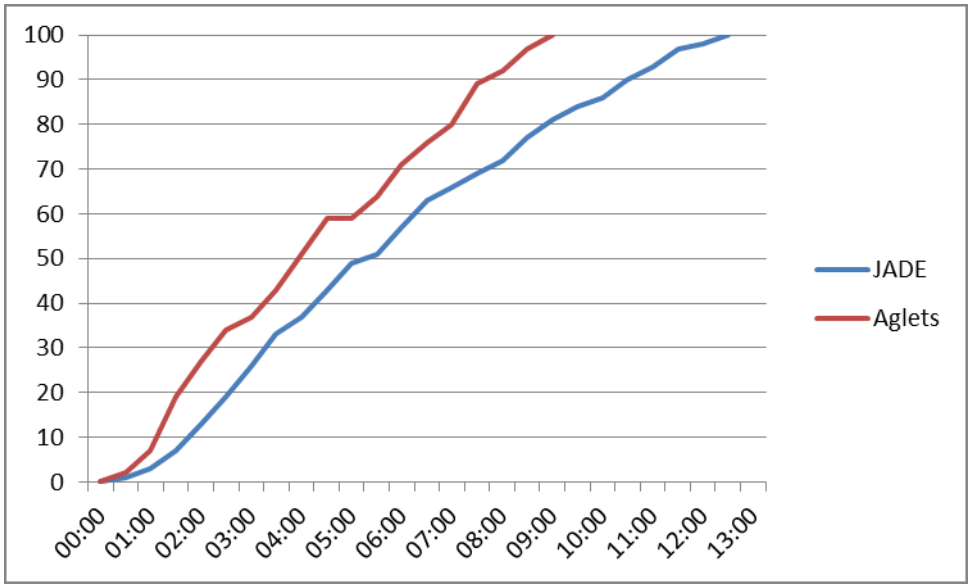


11 pav. Dviejų agentų migravimas (agentų iteracijų skaičius)

Pirmojo testo metu Aglets sistema parodė geresnius rezultatus. Testo pradžioje visos sistemos nerodė didelės spartos, todėl, kad pirmieji funkcijų kvietimai pareikalavo užkrovimo į atmintį ir pirmosios migracijos galėjo užtrukti dėl sekančios platformos, į kurią turi būti migruojama, paieškos. Pirmojo testo grafike (11 pav.) horizontalioje ašyje nurodytas laikas

sekundėmis, o vertikalioje agentų iteracijų skaičius. Iteracijos buvo skaičiuojamos suskaičiavus iteracijos baigimo pranešimus iš log4j bibliotekos sukuriamo failo, į kurį išvedama informacija, pranešimus sugrupavus ir susumavus kas sekundę ir pateikus grafike. Šio testo metu iteracija laikyta vienas agento migravimas iš vienos platformos į kitą platformą, esančią tame pačiame kompiuteryje. Platformų sąrašas agentams buvo žinomas iš anksto ir jį sudarė dvi atitinkamos platformos, tarp kurių agentai migruodavo. Kaip matyti iš grafiko Aglets sistemos startas yra „greitesnis“, joje agentai greičiau pradeda darbą ir jį greičiau pabaigia. Tačiau po pirminių veiksmų Aglets ir JADE sistemos, pakartotinai tuos pačius veiksmus atliko sparčiau ir iteracijų kiekis per tą patį laiko tarpą, palyginus su pirmosiomis testo sekundėmis, padidėjo. Šio testo metu, vykdant testą abejoms platformoms atskirai, procesorius buvo pilnai apkrautas, kas parodo, kad agentų platformoms yra svarbus laisvų resursų kiekis, todėl svarbu išsiaiškinti kaip elgsis platformos, kai joms bus užduota daugiau darbo. Agentų skaičiaus padidinimas leis ištestuoti sistemas didesnės apkrovos sąlygomis ir išbandyti jų stabilumą.

Antrojo testo metu agentų skaičius padidintas iki vieno šimto. Agentai atliks tokį patį darbą kaip ir pirmojo testo metu, tačiau bus stebima kaip sistema „susidoros“ su darbo apimties padidėjimu.

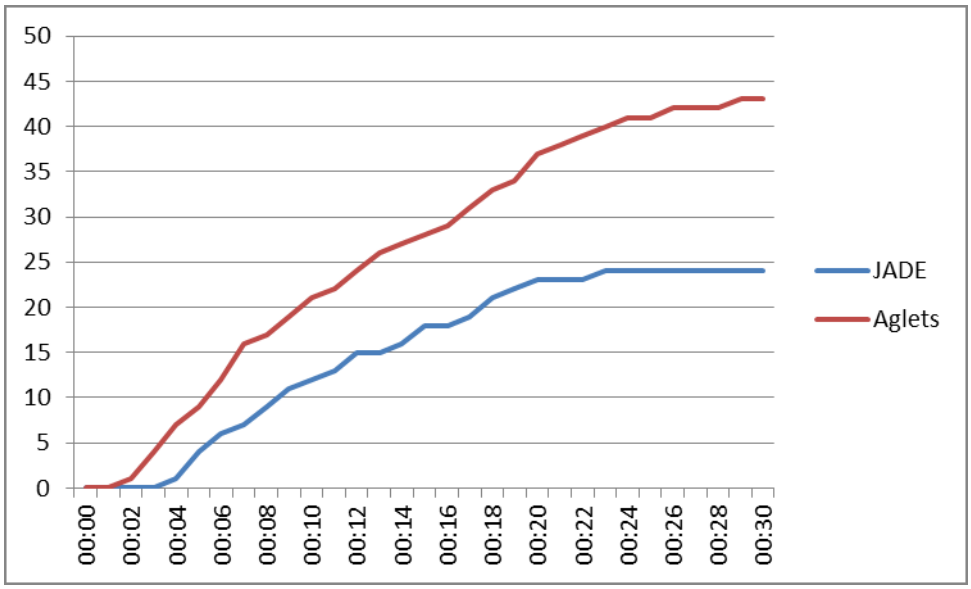


12 pav. Šimto agentų migravimas (baigę darbą agentai)

Šiame teste, taip pat kaip ir pirmajame, agentams reikėjo atlikti šimtą iteracijų. Antrojo testo rezultatų grafike horizontali ašis yra laiko atžvilgiu (minutės), o vertikalioji yra užbaigusių darbą agentų kiekis tam tikru laiko momentu. Trečiame paveiksle (12 pav.) matyti

panašios į pirmojo testo kreivės. Kartojasi tos pačios tendencijos kaip ir pirmajame teste: gana lėtas platformų startas, vėliau normalus ir spartus iteracijų vykdymas ir sulėtėjimas pabaigoje, kuris geriau matomas JADE platformos atveju, kadangi dalis agentų jau buvo pabaigę darbą, ir testo pabaiga – tai laukimas dar darbo neužbaigusių agentų. Šiame teste Aglets agentų platforma vėl pasirodė geriau nei JADE platforma. Aglets sistemos startas buvo greitesnis, jos agentai pirmieji pradėjo užbaiginėti darbus ir testui einant į pabaigą, kai likdavo vis mažiau agentų, likę agentai turėjo panaudoti atsilaisvinusius resursus po kitų agentų, kadangi pabaigoje nejaučiamas toks darbo sulėtėjimas kaip JADE sistemoje.

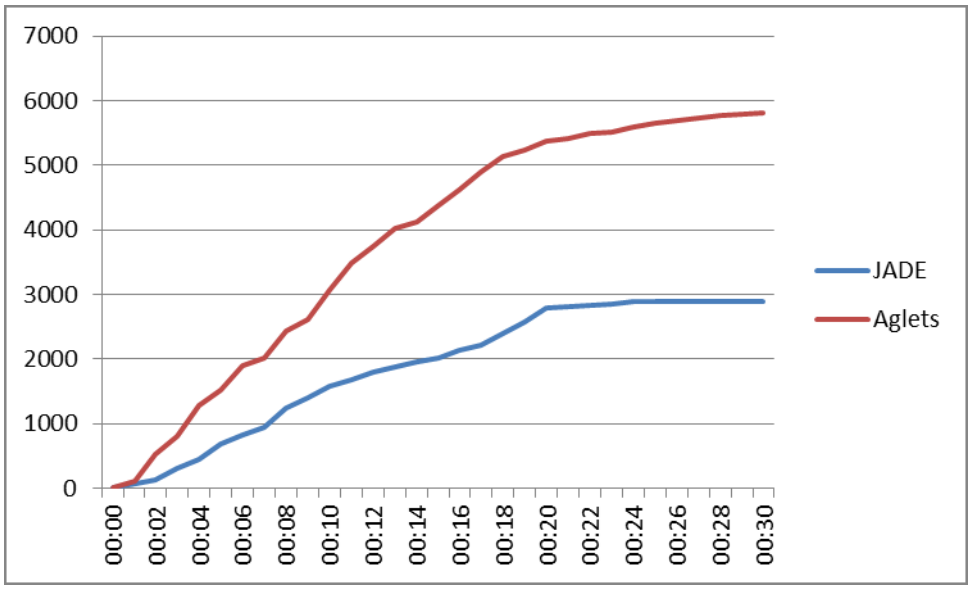
Paskutiniojo, trečiojo testo metu siekta išsiaiškinti kitos svarbios agentų platformos savybės efektyvumas, prie šio testo darbo pridėtas agentų komunikavimas su užduotu agentu. Agento sukūrimo metu, jam buvo pateikiamas agentas, su kuriuo jis turės komunikuoti testo eigoje, tai buvo vienas pastovus lokalus agentas platformos viduje, su kuriuo jis keisis žinutėmis. Agentų skaičius liko nepakitęs, t.y. jų sukurta ir paleista šimtas.



13 pav. Šimto agentų migravimas ir komunikavimas (baigę darbą agentai)

Paskutinįjį testą galima prilyginti „stress“ testui, kuomet mėginamas visos sistemos stabilumas. Šio bandymo metu, sistema, tikėtina, galėjo „nulūžti“ ir nebaigti darbo. Buvo pasirinkta tokia testo strategija, kad jei agentui nepavyks migruoti ar nusiųsti žinutės ir gauti atsakymo, veiksmas (žinutės siuntimo ar migravimo) bus pakartotas iš naujo, tol kol agentas pilnai atliks savo darbą. Pirmame grafike (13 pav.) pateikiamas baigusių darbą agentų skaičius laiko atžvilgiu (minutėmis horizontalioje ašyje), antrajame grafike (14 pav.)

atvaizduojamas iteracijų skaičius (vertikalioje ašyje) laiko atžvilgiu (minutėmis horizontalioje ašyje).



14 pav. Šimto agentų migravimas ir komunikavimas (agentų atliktos iteracijos)

Šio testo metu Aglets sistema taip pat parodė geresnius rezultatus nuo bandymo pradžios. Jos agentai buvo pirmieji, kurie pilnai baigė darbą. Teste buvo tikėtina, kad agentų sistema galėjo „nulūžti“, tuomet būtų tekę jį kartoti arba keisti pati testą, tačiau atsitiko kitas nenumatytas įvykis, sistemoms atlikus dalį darbo, jų sparta sulėtėjo, o JADE sistemai atlikus apie ketvirtadalį testo, darbas visai sustojo, ir agentai neatliko visai jokio darbo. Tuo tarpu Aglets sistemos agentai labai lėtai, bet rodydavo rezultatus ir atlikdavo iteracijas, tačiau po pusvalandžio testo, jis buvo nutrauktas dėl prastos spartos. Tai galima paaiškinti, kad pati agentų platforma negalėjo apdoroti agentų dėl per didelio resursų poreikio, kuomet agentai pradėdavo ilgiau vykdyti vieną iteraciją tam pačiam darbui atlikti. Iš pateiktų grafikų (13 pav. ir 14 pav.) matyti, kad tame pačiame laiko taške Aglets sistema atlieka daugiau iteracijų, nei yra darbą užbaigusių agentų, tuo tarpu JADE sistemai atliktų iteracijų skaičius laiko momentu yra artimesnis užbaigusių darbą agentų ir jų atliktų iteracijų skaičiaus sandaugai. Todėl galima daryti išvadą, kad Aglets sistemos agentai yra „gyvybingesnį“ už JADE sistemos agentus, tuo požiūriu, kad jie nesiblokuoja atlikdami darbą ir nelaukia, kol kiti agentai atliks darbą ir pati Aglets agentų platforma geba geriau valdyti agentus nei JADE paskirstydama darbui atlikti reikalingus resursus agentams.

3.6.1. Kiekybinės analizės išvados

Buvo tiriamos dvi agentų platformos – tai Aglets ir JADE. Analizės metu buvo atlikti trys testai, kuriais buvo siekiama išsiaiškinti tiriamų sistemų spartą:

1. Dviejų agentų migravimas tarp dviejų agentų platformų, veikiančių tame pačiame kompiuteryje
2. Šimto agentų migravimas tarp dviejų agentų platformų, veikiančių tame pačiame kompiuteryje
3. Šimto agentų migravimas tarp dviejų agentų platformų, veikiančių tame pačiame kompiuteryje ir komunikavimas

Pirmus du testus abi agentų platformos baigė. Trečią testą abi platformos dėl prasto resursų valdymo atliko tik iš dalies. JADE sistemos agentai po dvidešimt ketvirtos bandymo minutės neatliko visai jokio darbo, o Aglets sistemai testas dėl prastos spartos po trisdešimtos minutės buvo nutrauktas nors ir agentai atlikdavo iteracijas.

Bendru rezultatu Aglets agentų platforma greičiau geba susidoroti su užduotimis, paskirstyti ir valdyti resursus, o dėl savo migravimo mechanizmo, Aglets agentai geba sparčiau migruoti, o JADE agentams migruojant kartu migruojama ir agento būsena.

4. Mobiliosios komercijos agentų sistemos kūrimas

Mobiliosios komercijos agentų sistemos sukūrimu bus patvirtintas pasiūlyto prototipo įgyvendinamumas ir praktiškumas. Realizavus prototipą, bus parodyta, kad tokios sistemos veikimas yra realus mobiliojoje komercijoje, išnaudojant agentų teikiamus privalumus.

Kuriant sistemą, reikėjo rasti sprendimus šioms problemoms:

- problemos konstatavimas, pritaikymas mobiliems įrenginiams ir WAP'ui;
- sprendinio pasiūlymas, prototipo realizavimas (pasirinkimas iš alternatyvų);
- sprendinio sukūrimas (pačios sistemos realizavimas).

Problemos, su kuriomis buvo susidurta kuriant sistemą, pagal pateiktą prototipą:

- patirties trūkumas dirbant su Java, Apache, WML, JSP;
- dokumentacijos, pavyzdžių stoka (pateikti pavyzdžiai pasenę);
- ne visada sklandus darbas su agentų platforma (konfigūracijos problemos);
- kitoks mąstymas kuriant.

4.1. Modelio sukūrimas ir galimos alternatyvos

Atsiradus galimybei telefone naudotis internetu ir įvairiems tiekėjams siūlant plataus pobūdžio paslaugas panaudojant WAP protokolą, tikslinga sukurti paslaugą, kuria naudojantis vartotojas galėtų atlikti reikiamo produkto paiešką iškart tarp kelių skirtingų prekių ar paslaugų tiekėjų tiesiog telefono pagalba, naudojantis WAP protokolu, kad jam nereikėtų pačiam apklausinėti kiekvieno tiekėjo atskirai.

Tinkamas kuriamos sistemos modelio pasirinkimas, nulemia įgyvendinimo sėkmingumą ir atsako į šiuos klausimus: ar sistemą galima bus realizuoti, ar ji atliks darbą (t.y. atliks paiešką panaudojant mobiliosios komercijos agentus), ar ja bus patogu naudotis ir kt.

Remdamasis stacionariųjų ir mobiliųjų agentų žiniomis bei pasirinkta JADE mobiliųjų agentų platforma, kuriamai sistemai sukūriau tris alternatyvius modelius, kurie gali būti panaudoti sistemos kūrime:

1. Stacionarus agentas užklausia iš anksto žinomų tiekėjų DB ir gražina rezultatus
2. Vienas agentas apkeliauja iš anksto žinomus tiekėjus ir apklausia DB per kitą agentą

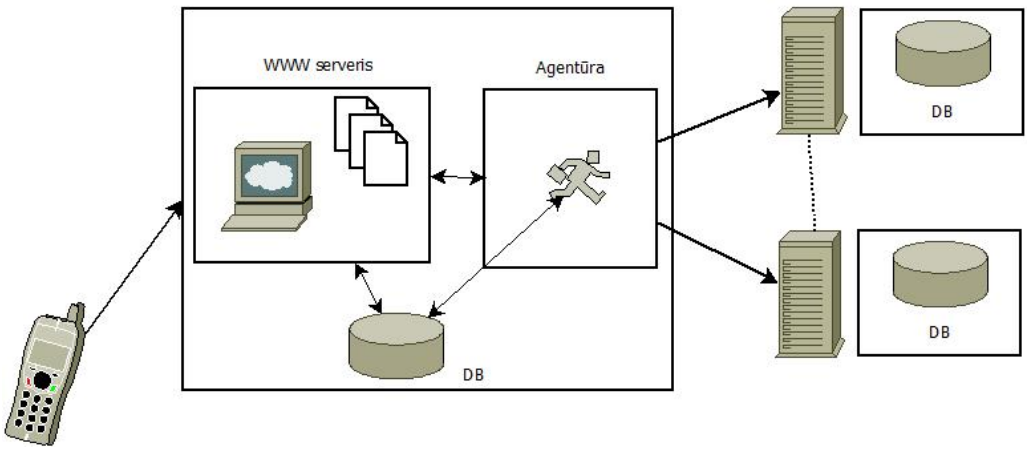
3. Vienas agentas apkeliauja iš anksto žinomus tiekėjus ir apklausia tiesiogiai SQL DB

Šie modeliai aprašyti 4.1.1, 4.1.2 ir 4.1.3 skyriuose.

4.1.1. Alternatyva nr. 1 „Stacionaraus agento sistema“

Pirmoji alternatyva, kurią teko svarstyti buvo sistemos modelis, kuris naudojo stacionarų agentą, kuris gavęs užklausą, ją apdoroja, o paiešką atlieka tarp iš anksto žinomų duomenų bazių. Tokiam modeliui realizuoti, užtektų paprastos agentų sistemos, kuri neužtikrina agentų mobilumo, ir nereikėtų jokių agentų komunikacijos priemonių, kadangi pats agentas tuomet atliktų visą produkto paiešką ir filtravimą. Įdiegtos agentų platformos užtektų tik pagrindiniame serveryje, kuris priima užklausas, kadangi kituose serveriuose bus tik duomenų bazės, kurios apklausinėjamos agento. Tačiau toks sprendimas pasirodė neefektyvus, nes agentų platforma tokiu modeliu yra įtraukiamą į sistemą bereikalingai. Jos konfigūravimui, diegimui ir priežiūrai reikia papildomai laiko, o agento panaudojimas neužtikrina pagrindinių agentų technologijos teikiamų privalumų. Agentas apklausęs visas duomenų bazes, rezultatus filtruoja, rūšiuoja ir tinkamiausia pateikia vartotojui.

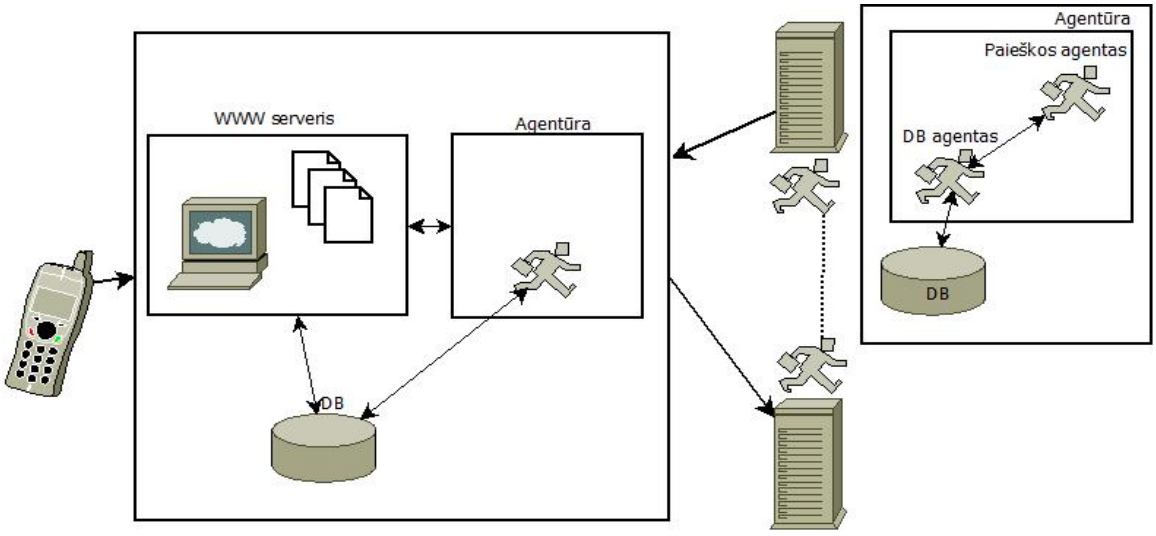
Bendras sistemos modelis galėtų būti panašus į pavaizduotą 15 paveiksle.



15 pav. Stacionaraus agento sistema

4.1.2. Alternatyva nr. 2 „Mobilusis agentas su stacionariu DB agentu“

Antrame modelyje, agentų platforma gavusi užklausą, sukuria agentą su tiekėjų sąrašu ir kuomet agentas pasiruošęs pradėti produkto paiešką, jis migruojamas į pirmąjį serverį, kuriame jį priima kita agentų platforma. Tokiu būdu agentų platformos yra įrašytos kiekviename iš serverių. Atkeliavęs į serverį, paieškos agentas susiranda tiekėjo agentą ir komunikuodamas su juo pateikia užklausos parametrus. Tuomet tiekėjo agentas gavęs užklausos parametrus pats bendrauja su duomenų baze, netinkančius rezultatus atmeta, o užklausą atitinkančius pateikia paieškos agentui, ir paieškos agentas tęsia paiešką toliau. Taip apkeliavęs visas duomenų bazes, agentas gražina rezultatus, kurie įrašomi į pirminio serverio duomenų bazę, apie ką vartotojas gali būti informuojamas žinute. Sistemos modelis pavaizduotas 16 paveiksle.

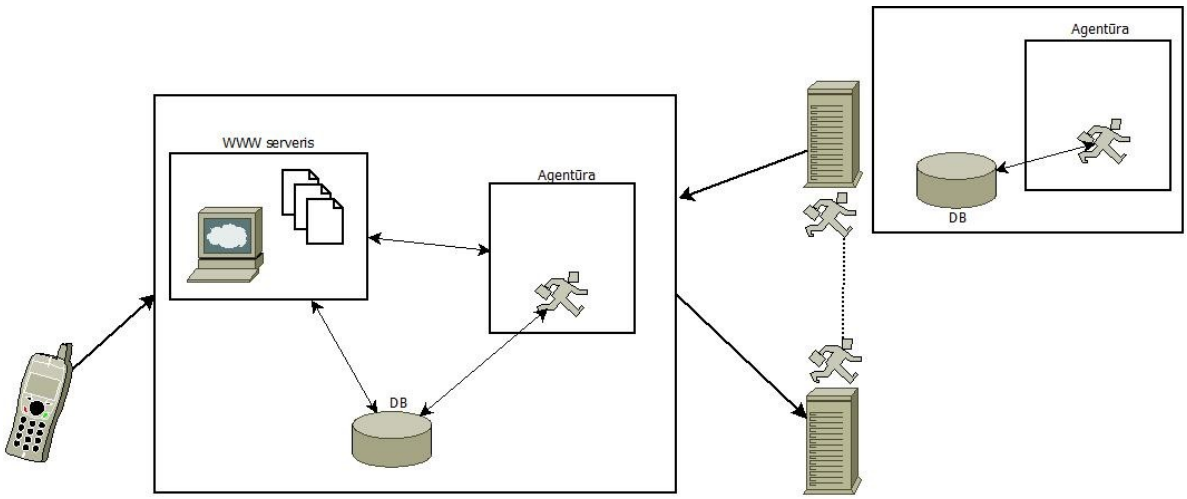


16 pav. Mobilusis agentas su stacionariu DB agentu

Toks modelis iš karto padidina sudėtingumą, kadangi pridedamas mobilumas ir komunikaciją tarp agentų. Tačiau yra efektyvus, kuomet agentų platforma palaiko dinaminius „proxy“. Ši savybė leistų paieškos agentui tiesiog apkeliauti visas tiekėjų agentų platformas ir pateikti užklausas jų duomenų bazių agentams, ir agentas galėtų keliauti toliau, o tiekėjų duomenų bazių agentai atlikę darbą ir palikę tik reikiamus rezultatus, naudodamiesi dinaminiais „proxy“, rezultatus pateiktų jau migravusiam paieškos agentui, tokiu atveju agentas gavęs visus rezultatus, arba per tam tikrą laiką jų negavęs, iš gautų, atliktų galutinę analizę ir pateiktų geriausią variantą vartotojui. Pasirinktoji agentų platforma JADE dinaminių „proxy“ nepalaiko, todėl toks sprendimas iš karto tampa nenaudingas dėl savo perdėto realizacijos sudėtingumo, kuomet jis neatneša realios naudos.

4.1.3. Alternatyva nr. 3 „Mobilusis agentas be stacionaraus DB agentu“

Šis modelis yra labai panašus į antrąjį modelį, kadangi jame naudojamas mobilus agentas. Esminis skirtumas tarp šių dviejų modelių yra tas, kad šiame modelyje yra atsisakyta stacionaraus duomenų bazių agento dėl JADE agentų platformos trūkumo. Šiame modelyje, agentas migravęs į agentų platformą pats atlieka užklausą ir filtruoja duomenų bazę ir nekaupia perteklinių rezultatų, o nuolatos juos atnaujina, tam kad migruojant nedidėtų pernešama ir nereikalinga informacija. Aprašytas modelis atvaizduotas 17 paveiksle.



17 pav. Mobiliosios komercijos agentų sistemos architektūra

Toks modelis pasirinktas dėl savo racionalumo ir paprastumo, kadangi agentas keliaudamas nekaupia papildomų duomenų, o tik pasilieka geriausią. Apkeliavęs visus tiekėjus, agentas grįžtą į pirminę platformą, kur rezultatas yra įrašomas duomenų bazę ir vartotojas gali jį peržiūrėti. Remiantis šiuo modeliu buvo sukurta mobiliosios komercijos sistema panaudojant agentų technologiją ir WAP protokolą, aprašyta 4.2 skyriuje.

4.2. Produktų paieškos mobiliosios komercijos sistemos kūrimas

Pagrindinis magistro baigiamojo darbo tikslas yra sistemos, naudojančios agentų technologiją, sukūrimas. Anksčiau sukurti ir aprašyti modeliai buvo svarstomi kaip kuriamos

sistemos pagrindas. Sukūręs šiuos modelius, įvertinęs jų sudėtingumą ir JADE sistemos galimybes, „Mobiliojo agento be stacionaraus DB agento“ modelį, aprašytą 4.1.3 skyriuje, pasirinkau kaip sistemos veikimo modelį, kadangi 4.1.1 skyriuje aprašytas „Stacionaraus agento sistemos“ modelis neišnaudoja agentų privalumų, o agentų technologijos panaudojimas tik daro jį sudėtingesniu, o 4.1.2 skyriuje aprašytas „Mobiliojo agento su stacionariu DB agentu“ modelis būtų tinkamas, jei JADE sistema turėtų galimybę keistis žinutėmis tarp agentų, kurie keičia savo vietą, kitiems agentams nežinant, ką stacionarus duomenų bazės agentas ir praneštų migravusiam mobiliajam agentui apie rezultatus.

Užklausa pateikiama mobiliuoju telefonu WAP puslapyje, kuriame įvedamas ieškomo produkto pavadinimas ir pageidaujama kaina. Šis puslapis panaudoja JSP, jame taip pat pateikiamas paskutinės atliktos užklausos rezultatas: ieškotas kriterijus su kaina, tiekėjo identifikatorius ir užklausos atlikimo data. Tuomet spaudžiamas mygtukas „Ieškoti“ ir užklausa patenka į web serverį. Kuriamoje sistemoje juo buvo pasirinktas Apache Tomcat web serveris, kuris apdoroja gautas WAP puslapio užklausas ir kviečia web servisą, kuris pateikiamas panaudojant JADE agentų platformos WSIG[24] web servisų praplėtimą. Renkantis web serverį, reikalavimai buvo tokie, kad web serveris galėtų apdoroti WAP užklausas, būtų gerai žinomas ir viešai prieinamas. Apache Tomcat web serveris buvo pasirinktas dėl savo prieinamumo, galimybių ir kitų sistemos keliamų reikalavimų. Web serverio paieškoms daug laiko neskirta, kadangi tai nėra šio darbo pagrindinis objektas.

Logika web serverio pusėje buvo realizuota remiantis JSP technologija. JSP technologija įgalina kurti dinaminį turinį internete, taip pat ir mobiliajam internetui ir WML puslapiams. Forma, kurioje įvedami užklausos duomenys, perduodavo GET metodu į kitą WML puslapį, kuris atlikdavo parametrų ištraukimą iš URL adreso ir kviesdavo JADE agentų platformos web servisą. WSIG praplėtimas leidžia apdoroti per web servisus gautas užklausas, be šio praplėtimo JADE agentų platformai dar buvo keletas panašių praplėtimų, bet šis pasirinktas dėl to, kad pateiktas JADE svetainėje ir kartu pateikiamo dokumentacijos kiekio. Paleistas WSIG agentas ir laukiantis kreipimosi į web servisą, gavęs užklausimą, ištraukia iš WSDL užklausos parametrus ir sukuria paieškos agentą. Sukurtasis paieškos agentas tiekėjų sąrašą turėjo sukūrimo metu, t.y. jis buvo žinomas iš anksto, tačiau dinaminis tiekėjų sąrašas galėtų būti vartotojui palankesnis, kuomet paieška galėtų būti atliekama įvedant papildomus tiekėjus ar išimant nepageidaujamus. Paieškos agentas patikrinęs savo aplankytų tiekėjų skaitliuką ir tiekėjų sąrašo ilgį pradėdavo darbą. Duotojoje sistemoje

[24] (angl. Web Service Integration Gateway)

lankytinų tiekėjų sąrašas buvo laikomas kodo eilutėje. Visos lankytinos agentų platformos buvo paleistos darbo kompiuteryje, besinaudojančios skirtingais kompiuterio prievadais žinutėms keistis. Kadangi visos platformos buvo viename kompiuteryje, tokiu būdu buvo pašalintos dėl skirtingos konfigūracijos galinčios pasireikšti problemos, taip pat ir ugniasienių ir tinklo apsaugos, galinčios sutrukdyti migruoti mobiliesiems agentams. Kadangi tai yra konfigūracijos ir saugumo klausimas, o tai nuspręsta neaptarinėti dar darbo pradžioje, tai į tokią sistemos modelio realizaciją nebuvo žiūrima kaip į trūkumą.

Žemiau pateikiamas agento migravimo Java kodo pavyzdys:

```
    AID remoteAMS = new AID("ams@UserOn:1000/JADE",
AID.ISGUID);
    remoteAMS.addAddresses("http://UserOn:1100/acc");
    PlatformID destination = new PlatformID(remoteAMS);
    doMove(destination);
```

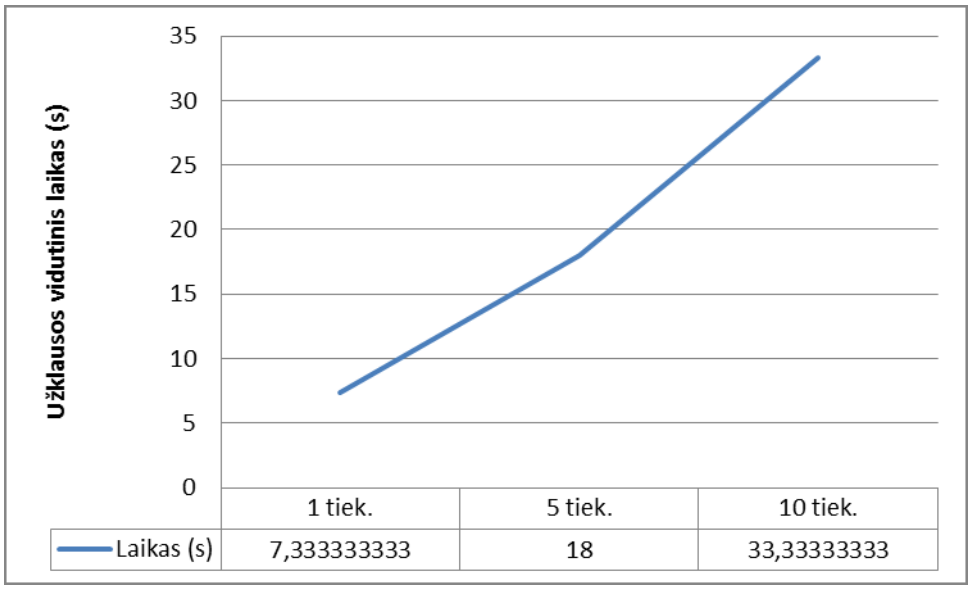
18 pav. Agento migravimo Java kodo pavyzdys

Migruodamas agentas žinojo „stotelių“ sąrašą ir kiekvienoje iš jų atlikdavo užklausą duomenų bazėje. Duomenų bazei buvo pasirinkta MySQL duomenų bazių valdymo sistema, į kurioje agentas atlikdavo užklausą. Ši duomenų bazių valdymo sistema pasirinkta dėl gero žinomumo, viešo prieinamumo ir gebėjimo dirbti su Java programavimo kalba. Kadangi simuliuojama aplinka buvo tame pačiame kompiuteryje, buvo naudojamas tas pats duomenų bazės serveris, tačiau vykdomai užklausai atlikti nurodoma vis kita duomenų bazės, atitinkamai prie duomenų bazės pavadinimo pabaigoje pridedant eiliškumo numerį. Taip apeinamos visos platformos ir galiausiai migruojama į pirminę agentų platformą, kurioje įrašomas rezultatas į duomenų bazę. Pirmos duomenų bazės eiliškumo numeris suteiktas nulis, į kurią agentas įrašo paieškos rezultatą. Atlikus užklausą, gautas geriausias produkto pasiūlymas buvo palyginamas su jau saugomu vidiniame agento kintamajame ir, jei naujas rezultatas pasirodydavo geresnis už prieš tai buvusį arba prieš tai nebuvo jokio, tuomet jis būdavo išsaugomas agento vidiniuose kintamuosiuose, kartu su platformos numeriu, kurioje aptiktas. Užklausa, kuri vykdoma ieškant produkto pavadinimo, buvo nesudėtinga SQL užklausa, išrenkanti pirmą panašų į ieškomo produkto pavadinimo rezultatą su žemiausia kaina.

4.3. Sukurtos sistemos vertinimas

Sukūrus sistemą, reikėjo ją įvertinti, kiek pasirinktas modelis yra priklausomas nuo tiekėjų skaičiaus, kadangi, kai tiekėjų daug, užklausos vykdymo laikas gali išaugti ir tapti nebepriimtinas vartotojui. Kita vertus, labai didelis užklausos laikas verstų ieškoti kito modelio pasirinkimo ar kitos agentų sistemos, kuri galėtų panaudoti dinaminius „proxy“ komunikavimui.

Bandymui buvo pasirinktas 1, 5 ir 10 tiekėjų skaičius. Bandymai buvo atliekami po tris kartus ir išvestas bandymų aritmetinis vidurkis. Nors ir visų tiekėjų agentų platformos paleistos tame pačiame kompiuteryje, tačiau dėl didelės darbinės atminties kiekio, tai problemų nesukėlė. Agentų platformos naudojamas atminties kiekis, kuomet ji neatlikdavo jokio darbo, siekė apie 25-35 megabaitus. Duomenų kiekis tiekėjo duomenų bazės lentose buvo toks pat (po 20 įrašų). Bandymų rezultatai pateikti 19 paveiksle. Užklausos trukmė nustatyta išvedant paieškos agento darbo pradžios ir darbo pabaigos laikus ir imant tų laikų skirtumą.



19 pav. Bandymas su skirtingais tiekėjų skaičiais

Iš pateiktos kreivės matyti, kad didėjant tiekėjų skaičiui, užklausos vykdymo laikas lieka panašus, tik pirmajam tiekėjui apklausti reikia pradinio pasiruošimo, kol paleidžiamas agentas, ir ji trunka ilgiausiai (apie 7 sekundes), o su 5 ar 10 tiekėjų, vidutinis tiekėjo apdorojimo laikas yra apie 3-4 sekundes. Kai apklausiama 10 tiekėjų, toks, pusės minutės, laikas yra priimtinas vartotojui, žinant, kad tiek pat laiko galėtų užtrukti vieno tiekėjo apklausimas, paiešką vykdant pačiam vartotojui savarankiškai.

REZULTATAI IR IŠVADOS

Darbe ištirtos programinių agentų platformos. Jos aprašytos kiekviena atskirai, o jų savybės pateiktos lentelėje. Pasirinktoms dviem agentų platformoms suprogramuoti ir atlikti testai. Testų rezultatai atvaizduoti grafikuose. Atlikus testus ir palyginnus rezultatus, tolesniam darbui buvo pasirinkta JADE agentų platforma. Nors ji ir truputį lėtesnė už Aglets platformą, tačiau didesnė dirbo su ja patirtis ir didesnis mokomųjų resursų kiekis lėmė būtent šios platformos pasirinkimą.

Sukurtas mobiliosios komercijos sistemos, naudojančios WAP, prototipas. Jis įgalina atlikti informacijos paiešką tarp skirtingų tiekėjų, įvedus pageidaujamo produkto pavadinimą ir kainą. Sukurtu mobiliosios komercijos sistemos prototipu parodyti agentų technologijos privalumai. Taip pat parodyta, kad pati agentų technologija yra gyvybinga ir jos atsisakyta per anksti. O praktinis prototipo panaudojimas gali turėti taikomąją vertę.

Sistemos sparta buvo vertinama keičiant apklausiamų tiekėjų skaičių. Net ir parinkus 10 tiekėjų, pasiektas užklausos vykdymo laikas kiek daugiau nei pusė minutės yra priimtinas vartotojui, kadangi tokį laiką dabar vartotojas sugaišta vienam tiekėjui užklausti. Tolesnis tokios sistemos plėtojimo darbas galėtų būti draugiškesnio mobiliojo įrenginio interfeiso vartotojui, kuriuo pateikiamos užklausos, tobulinimas, paieškos algoritmo ir rezultatų gražinimo mechanizmo tobulinimas.

Mobiliojo ryšio nepastovumas (trūkinėjimas ir prisijungimas prie tinklo iš naujo), mažas greitis, didelė duomenų perdavimo kaina kelia pagrindinius nepatogumus vartotojui. Agentai savo ruožtu leidžia atlikti užklausą ir nebesirūpinti atsakymu, kadangi jis turi būti pateiktas, kai užklausa bus atlikta. Todėl nebūtinas ryšio pastovumas, kaip kad kliento-serverio architektūroje, ir agentas, naudodamasis vidine suprogramuota logika, automatiškai filtruoja rezultatus iš tiekėjo taip pernešdamas mažiau duomenų.

Kadangi agentas mobiliojoje komercijoje atstovauja vartotoją ir jo poreikius, todėl vartotojui pateikiamas turinys gali būti suasmenintas ir skirtas konkrečiam vartotojui. Naudodami aplinkos daviklius, agentai geba reaguoti į aplinką, taip prisitaikydami prie jos ir pateikdami vartotojui tikslesnius rezultatus, pavyzdžiui, už mažesnę ieškomo produkto kainą, nei vartotojas nustatė. Taip pat agentų tarpusavio komunikavimas gali dar labiau patikslinti paieškos rezultatus.

Agentų technologijos panaudojimas mobiliojoje komercijoje gali ne tik spręsti nepastovaus ryšio, didelio duomenų perdavimo kiekio problemas, tačiau jų naudojimas produktų ar paslaugų paieškoje pateikia labiau poreikius atitinkančius, tikslesnius ir siauresnius rezultatus vartotojui.

ŠALTINIAI

[ADG09] Aglets Development Group. The Aglets 2.0.2 User's Manual., 2009. [žiūrėta 2011-02-07]. Prieiga per Internetą: <http://sourceforge.net/projects/aglets/files/User_s%20Manual/March%202009/manual031209.pdf/download>

[AOS] AOS. JACK White Paper., 1998. [žiūrėta 2011-02-07]. Prieiga per Internetą: <http://www.aosgrp.com/downloads/JACK_WhitePaper_US.pdf>

[BBCP] Fabio Bellifemine, Federico Bergenti, Giovanni Caire, Agostino Poggi. JADE – A Java Agent Development Framework.

[BCG07] Fabio Bellifemine, Giovanni Caire, Dominic Greenwood. Developing Multi Agent Systems with JADE. Wiley, 2007.

[BCPR03] Bellifemine F., Caire G., Poggi A. and Rimassa G. JADE: A white Paper, 2003. [žiūrėta 2011-04-07]. Prieiga per Internetą: <JADE.tilab.com/papers/WhitePaperJADEEXP.pdf>

[BCT10] F. Bellifemine, G. Caire, T. Trucco. JADE Programmer‘s Guide., 2010. [žiūrėta 2011-02-07]. Prieiga per Internetą: <http://JADE.tilab.com/doc/programmersguide.pdf>

[BCTr10] F. Bellifemine, G. Caire, T. Trucco. JADE Administrator‘s Guide., 2010. [žiūrėta 2011-02-07]. Prieiga per Internetą: <http://JADE.tilab.com/doc/administratorsguide.pdf>

[Bia05] Eric Bianchi. UK leading Europe in smart phone sales. [žiūrėta 2011-02-06]. Prieiga per Internetą: <http://calvinayre.com/2011/02/04/business/uk-leading-europe-in-smart-phone-sales/>

[BPL05] Lars Braubach, Alexander Pokahr, Winfried Lamersdorf. Tools and Standards, 2005.

[BPR98] Fabio Bellifemine, Agostino Poggi, Giovanni Rimassa. JADE – A FIPA-compliant agent framework, 1998.

[CA04] J. J. Chen, C. Adams. Enabling Secure, Interoperable, and User-friendly Mobile Payments, Proceedings of the 6th international conference on Electronic commerce, 2004.

[Cai09] G. Caire. JADE Programming for Beginners., 2009. [žiūrėta 2011-02-07]. Prieiga per Internetą:

<http://JADE.tilab.com/doc/tutorials/JADEProgramming-Tutorial-for-beginners.pdf>

[Che07] R.-Y. Chen. A web services agent-based framework for mobile payment process, 2007.
[žiūrėta 2009-09-10]. Prieiga per Internetą:
<http://itgroup.blueshop.com.tw/fleget/FileDownLoad.aspx?CDE=1086>

[Col00] J. Collis. The Zeus Agent Building Toolkit – The Runtime Guide., 2000.

[Fer09] J. Ferber MadKit pas à pas, 2009.
[žiūrėta 2011-02-07]. Prieiga per Internetą:
<www.lirmm.fr/~ferber/madkit/Madkit-pasapas.1.pdf>

[FIPA] Foundation for Intelligent Physical Agents.
[žiūrėta 2011-02-07]. Prieiga per Internetą:
<http://www.fipa.org>

[FDAS02] E. Ferreira, P. Duarte, J. C. Abrantes, F. R. Sampaio. Implementing Mobile Agent Design Patterns in the JADE framework, 2002.
[žiūrėta 2009-09-10]. Prieiga per Internetą:
<http://JADE.tilab.com/papers/EXP/Ferreira.pdf>

[GFM00] O. Gutknecht, J. Ferber, F. Michel. The MADKIT Agent Platform Architecture., 2000.

[Hun08] J. C. Hung. Foreword of Special Issue on "Mobile System, Agent Technology, and Web Services", Journal Of Software, vol. 3, no. 8, 2008.
[žiūrėta 2009-09-10]. Prieiga per Internetą:
<http://www.academypublisher.com/jsw/vol03/no08/jsw03080102.pdf>

[IKV98] IKV++. Grasshopper Development System – Basics and Concepts., 1998.

[Jha00] Rahul Jha. Mobile agents for e-commerce, 2000.

[KUU02] R. Kowalczyk, M. Ulieru, R. Unland. Integrating Mobile and Intelligent Agents in Advanced e-Commerce: A Survey, 2002.
[žiūrėta 2009-09-10]. Prieiga per Internetą:
<http://www.netobjectdays.org/www_old_netobjectdays_org/pdf/02/papers/malceb/0679.pdf>

[Li05] X. Li. The Role of Mobile Agents in M-commerce, The Sixth Wuhan International Conference on E-Business, e-Business Track, 403-408 psl, 2005.
[žiūrėta 2009-09-10]. Prieiga per Internetą:
<http://it.swufe.edu.cn/UploadFile/other/xsjl/sixwuhan/Paper/EB037.pdf>

[LK02] F. C. Lin, C. N. Kuo. Cooperative multi-agent negotiation for electronic commerce based on mobile agents, 2002.
[žiūrėta 2009-09-10]. Prieiga per Internetą:
< http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1176097>

[MSS99] P. J. Marques, L. M. Silva, J. G. Silva. Security Mechanisms for Using Mobile Agents in Electronic Commerce, 1999
[žiūrėta 2009-09-10]. Prieiga per Internetą:
<http://www2.computer.org/portal/web/csdl/doi/10.1109/RELDIS.1999.805127>

[MT00] Mihhail Matskin and Amund Tveit. Mobile Commerce Agents In WAP-Based Services, 2000.

[MVV04] Antonio Moreno, Aida Valls, Alexandre Viejo. Using JADE-LEAP to implement agents in mobile devices, .

[Nwa96] H. S. Nwana. Software Agents: An Overview. Knowledge Engineering Review, 205-244 psl., 1996.

[NM09] Cynthia Nikolai, Gregory Madey. Tools of the Trade: A Survey of Various Agent Based Modeling Platforms, 2009.

[PAG08] R. Petrauskas,K. Agafonov, R. Gamulis. E-verslo informacinės sistemos, Vilnius, 2008.

[PCVM99] M. K. Perdikeas, F. G. Chatzipapadopoulos, I. S. Venieris, and G. Marino, "Mobile agent standards and available platforms" Computer Networks, vol. 31, no. 19, pp. 1999–2016, 1999.

[Pet05] J. Peters. Integration of Mobile Agents and Web Services, 2005.
[žiūrėta 2009-09-10]. Prieiga per Internetą:
<http://semoa.sourceforge.net/docs/papers/peters2005a.pdf>

[Sam02] G. Samtani. B2B Integration: A Practical Guide to Collaborative E-commerce, Imperial College Press, 2002.

[SBPL04] Sudeikat, J.; Braubach, L.; Pokahr, A.; Lamersdorf. Evaluation of Agent-Oriented Software Methodologies – Examination of the Gap Between Modeling and Platform., 2004.
[žiūrėta 2011-02-07]. Prieiga per Internetą:
<http://vsis-www.informatik.uni-hamburg.de/getDoc.php/publications/207/AOSE04-03.pdf>

[SE03] O. K. Sahingoz, N. Erdogan. A two-leveled mobile agent system for electronic commerce, Journal of aeronautics and space technologies, July 2003, vol. 1, no. 2, 2003.
[žiūrėta 2009-09-10]. Prieiga per Internetą:
<http://www3.itu.edu.tr/~nerdogan/huten-sahingoz-erdogan.pdf >

[Sho97] Y. Shoham. An Overview of Agent-oriented Programming. In Software Agents, AAAI Press, 1997.

[TIS06] J. Tweedalea, N. Ichalkaranjeb, C. Sioutisb ir kt. Innovations in multi-agent systems, 2006.

[Try01] Tryllian B. V. Agent Development Kit 1.3 – Technical White Paper., 2001.

[Vys08] E. Vyšniauskas. Programiniai agentai, Kaunas, 2008.

[ZZ02] W. Zhimin, Q. Zheng. Mobile agent oriented architecture to build open mobile electronic commerce system, Proceedings of IEEE Tencon'02, 2002 [žiūrėta 2009-09-10]. Prieiga per Internetą: <ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1181242>

SĄVOKŲ APIBRĖŽIMAI

Agentas – **agentų technologija** (sumanieji agentai, mobilieji agentai ir kt.) – programinės įrangos vienetas, turintis tikslą ir įgyvendinantis jį.

Contract Net Protocol (CNP) – užduočių paskirstymo protokolas, naudojamas multiagentinėse sistemose.

Mobilioji komercija – komercijos rūšis su mobiliaisiais įrenginiais.

Multiagentinė sistema – agentų sistema, sudaryta iš kelių, tarpusavyje sąveikaujančių agentų.

„Stress“ testas – testo rūšis, kurio metu yra bandomas sistemos patikimumas, ją labai apkraunant, su tikslu išbandyti ar sistema liks stabili.

WAP – protokolų rinkinys, sukurtas mobiliems įrenginiams.

WAP paslaugos – paslaugos, teikiamos WAP protokolu.

SANTRUMPOS

ACL – *Agent Communication Language*, agentų komunikavimo kalba

API – *Application Programming Interface*, taisyklių ir specifikacijų rinkinys

B2C – *Business-to-Customer*, verslas vartotojui

B2B – *Business-to-Business*, verslas verslui

CNP – *Contract Net Protocol*, sandorių protokolas

DTD – *DataType Definition*, duomenų tipo apibrėžimas

FIPA – *Foundation for Intelligent Physical Agents*, ne pelno siekianti orgranizacija, kurianti standartus, skirtus agentų technologijai ir multiagentinėms sistemoms

JADE – *Java Agent DEvelopment framework*, Java agentų kūrimo platforma

JDK – *Java Development Kit*, naudojamas kompiliuoti Java programoms ir susidedantis iš Java kalbos API, kompiliatoriaus ir virtualios mašinos

JSP – *Java Server Pages*, technologija, leidžianti dinamiškai generuoti HTML, WML ar kito turinio puslapius

KQML – *Knowledge Query Manipulation Language*

MA – *Mobile Agent*, mobilusis agentas

MAS – *Multi-Agent System*, multiagentinė sistema

MASIF – *Mobile Agent System Interoperability Facility*, OMG sukurtas agentų standartas

OMG – *Object Management Group*, ne pelno siekianti orgranizacija, kurios pagrindinis tikslas yra kurti standartus paskirstytoms sistemoms

ORB – *Object Request Broker*, tarpinės programinės įrangos dalis, kuri tinklu leidžia atlikti kvietimus programos viduje iš vieno kompiuterio į kitą

RPC – *Remote Procedure Call*, nuotolinis kreipimasis į funkciją, viena iš ORB realizacijų

URL – *Uniform Resource Locator*, standartas, skirtas nurodyti adresui

WAP – *Wireless Application Protocol*, bevielio ryšio programų protokolas

WML – *Wireless Markup Language*, kalba, naudojama WAP puslapių kūrimui

WSDL – *Web Service Definition Language*, web paslaugų apibrėžimo kalba

WSIG – *Web Services Integration Gateway*, web paslaugų integravimo „vartai“

XML – *eXtensible Markup Language*, plečiama žymėjimo kalba