KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Internetinės mokymosi sistemos informacijos analizės agento sudarymas ir tyrimas

Magistro darbas

Darbo vadovas:
prof. E. Kazanavičius

Kaunas, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Internetinės mokymosi sistemos informacijos analizės agento sudarymas ir tyrimas

Magistro darbas

Recenzentas
Doc. Dr. E. Toldinas
2008-05-23

Vadovas
prof. E. Kazanavičius
2008-05-23

Atliko
IFM–2/1 gr. stud.
Saulius Kreivaitis
2008-05-23

Kaunas, 2008

INTERNETINĖS MOKYMOSI SISTEMOS INFORMACIJOS ANALIZĖS AGENTO SUDARYMAS IR TYRIMAS

SANTRAUKA

Nuotolinis mokymasis tampa svarbus šiandieninėje visuomenėje, vienas pagrindinių nuotolinio studijavimo privalumų – galimybė priartinti mokymą prie besimokančiojo. Šiame darbe analizuotas ir sudarytas Internetinės mokymosi sistemos informacijos analizės agentas.

Tam, kad užtikrinti kokybiškesnį, efektyvesnį nuotolinio mokymosi sistemos funkcionalumą, SMVS - studijų modulių valdymo sistemą papildyta integruojant informacijos analizės agento sistemą paremta šiandieninėmis tendencijomis - agentų technologija. Atlikus informacinių agentų tipologijos analize pasirinktas reaktyvinis agentas, paremtas jutiklių ir vykdiklių pagrindu. Realizuota sistema veikia kaip dėstytojų ar administratorių pagalbininkas – asistentas kuris seka ir valdo mokymosi veiklas. Informacijos analizės agento sistema reaguoja į kurso dalyvių dalyvavimą, aktyvumą, pateikia susisteminta informaciją apie modulius ir vartotojus. Sukurta sistema aktuali vartotojams, kurie naudojasi SMVS sistema, nes ji pateikia išsamia informaciją į naršyklės langą arba siunčia e-paštu.

COMPILATION AND RESEARCH OF THE AGENT OF THE INTERNET-BASED LEARNING SYSTEM INFORMATION ANALYSIS

SUMMARY

Distance learning is gradually becoming important in modern society and one of the main advantages of distance learning is the possibility to draw the learning and the learner closer together. In this study the agent of information analysis of the internet-based learning is analyzed and created.

To ensure a high quality and a more effective functionality of the system of distance learning, a study module management system, an SMVS, has been supplemented by integrating the system of information analysis agent, based on present-day tendencies of agent technology. Having completed the typological analysis of information agents, a reactive agent has been selected. The system works as an assistant for teachers, lecturers and administrative staff, observing and managing the process of learning. The system of information analysis agent reacts to the level of participation and activeness of the course participants. It also produces systematised information about modules and users. The system is especially advantageous to the users who exploit the SMVS system since it presents comprehensive information in a browser window and can be sent over an e-mail.

TURINYS

Lentelės

Paveikslėlių sąrašas

1. ĮVADAS

Informacinės technologijos vis dažniau veržiasi į mūsų gyvenimą. Dauguma jaunų žmonių neįsivaizduoja savo gyvenimo be kompiuterio ar Interneto. Gyvenant aktyvų gyvenimą, laikas yra viena svarbiausių vertybių. Nuotolinis mokymasis tampa svarbus šiandieninėje visuomenėje, nes vienas pagrindinių nuotolinio studijavimo privalumų – galimybė priartinti mokymą prie besimokančiojo, studijos gali išplėsti mokymosi apimtį neribojant laiko ir vietos, kursams reikia mažiau dėstytojų tam pačiam studentų kiekiui, suteikia daug daugiau informacijos ir galimybių studentams išvystyti gebėjimą galvoti, kurti, ir tapti savarankiškais.

Siekiant, nuotolinių studijų efektyvumo, jų lankstumui kuriamos studijų kokybei užtikrinančiai nuotolinio mokymosi sistemos, kurių kokybė priklauso nuo kurso parengimo, medžiagos, pateikimo, interaktyvumo užtikrinamo.

Tiekiant net ir kokybiškai parengtą kursą, reikia sukoncentruoti dėmesį į bendravimą, sąveika, studentų aktyvumo paskatinimui bei kontrolei.

Šiam tikslui nuotolinio mokymosi sistemose į jų priežiūros procesus gali būti integruojami programiniai agentai, kurie veikia kaip dėstytojų ar administratorių pagalbininkai – asistentai sekdami ir valdydami mokymosi veiklas. Programinių agentų sistemos reaguoja į kurso dalyvių dalyvavimą, aktyvumą, užduočių atlikimą, pažangumą pateikdamos susisteminta informaciją dėstytojui ar studentui.

Šio darbo tikslas - užtikrinti kokybiškesnį, efektyvesnį nuotolinio mokymosi sistemos funkcionalumą. Šiam tikslui pasiekti reikia:

- Atlikti nuotolinio mokymosi valdymo sistemų palyginamąją analizę;
- Atlikti agentų tipologijos analizę;
- Sukurti informacijos analizės agento sistemą;
- Integruoti į SMVS - Studijų Modulių Valdymo Sistemą;
- Atlikti informacijos agento sistemos eksperimentą ir tyrimą;
- Pateikti informacijos agento sistemos susistemintą informaciją vartotojui.

2. NUOTOLINIO MOKYMOSI SISTEMŲ IR INFORMACIJOS VALDYMO METODŲ ANALIZĖ

2.1 Tiriamojo darbo objekto analizė

Nuotolinis mokymasis – tai mokymasis, paremtas naujausiomis informacinėmis ir komunikacinėmis technologijomis, kai galima mokytis patogiu metu, patogioje vietoje ir priimtina sparta.

Nuotolinio mokymosi privalumai:

- Galimybė priartinti mokymą prie besimokančiojo;
- Galima mokytis norimu metu, patogioje vietoje ir priimtina sparta;
- Kursų medžiaga nuolat atnaujinama;
- Įgytos žinios yra praktiškai patikrinamos laboratoriniais darbais;
- Nereikia daug laiko ir lėšų (kelionėms, pragyvenimui kitame mieste, reikalingai literatūrai pirkti ir t.t.).

Nuotolinio mokymosi trūkumai:

- Sunku organizuoti operatyvią mokymosi rezultatų kontrolę;
- Autorinių teisių problemos;
- Neapdraudžia nuo pasyvaus mokymosi;
- Skurdžios poveikio studentams priemonės;
- Ilgas parengimo ciklas.

2.2 Nuotolinio mokymosi valdymo sistemų (NMVS) analizė

Šiuo metu, plačiai nagrinėjamos nuotolinio mokymo valdymo sistemos (NMVS), dažnai naudojamas angliškas terminas Learning Management System (LMS). Nuotolinio mokymo valdymo sistemos sudaro didesnes galimybes ne tik valdant mokymo ir mokymosi procesus, bet ir kuriant e.kursus bei juos teikiant. Nuotolinio mokymo valdymo sistemos atveria galimybes jungtis institucijoms, teikiančioms e.kursus netgi skirtingose virtualiose mokymo terpėse, studentams galima pasiūlyti žymiai daugiau įvairių elektroninių kursų, iš kurių jie gali rinktis jiems labiausiai patinkančius. Tokiu atveju, turėtų didėti konkurencija tarp nuotolinių studijų tiekėjų ir gerėti jų kokybė.

Šiuo metu siūlomos tokios nuotolinio mokymo valdymo sistemos: Moodle, WebCT Vista, Blackboard, Learning Management Suite LMS, Adventus LMS, ir kt.

Nuotolinio mokymo valdymo sistemos (NMVS) tai yra programinė įranga, skirta valdyti e.mokymo ir mokymosi procesus dideliu mastu. *Nuotolinio mokymo valdymo sistemos* yra programinės įrangos sistemos, kurios apjungia kompiuterinės programinės įrangos (e.pašto, skelbimų lentos, naujienų grupių ir kt.) ir e. kursų teikimo (pvz. WWW) funkcijas. Tai yra e.mokymosi portalai.

NMVS svarbiausios funkcijos yra studentų duomenų valdymas, sistemos administravimas, priėjimo prie kursų kontroliavimas. Dažnai NMVS apibūdinamos kaip mokymosi portalai, kurie jungia vartotojus su įvairiomis mokymosi priemonėmis. Kai kuriais atvejais NMVS naudojama kursų katalogui valdyti ir įvairioms nuotolinio mokymosi priemonėms apjungti – tai leidžia pateikti mišrų sprendimą vartotojui.

Šiuolaikinės NMVS turi atlikti šias funkcijas:

- registruoti studentus,
- tvarkyti studentų duomenis,
- sekti studentų rezultatus,
- kurti ataskaitas,
- sudaryti tvarkaraščius,
- ieškoti žinių spragų,
- valdyti mokymo įstaigos išteklius,
- valdyti virtualią klasę,
- organizuoti kursų ir kitų mokymo išteklių katalogą,
- valdyti sistemą.

2.3 Studijų modulių valdymo sistemos (SMVS) analizė

SMVS – tai studijų modulių valdymo sistema. Ji suprojektuota ir sukurta Kompiuterių katedrai https://ifko.ktu.lt/kkpes4, informacijos organizavimui ir mainams tarp dėstytojo ir studento.

Šioje sistemoje pagrindiniai veikiantys asmenys būtų:

- *Dėstytojas*. Tvarko visą su jo dėstomais moduliais susijusią informaciją.
- *Studentas.* Aktyviai domisi jam reikalingais moduliais ir yra pastoviai informuojamas apie visus pakitimus bei naujienas.
- *Administratorius.* Turi visas teises susijusias su šia sistema.

Pagrindinės veiklos sritys modulio vedimo metu yra tokios:

- Paskaitos – Dėstytojas veda paskaitas pagal iš anksto paruoštą modulio planą. Daugelis dėstytojų medžiagą pateikia elektronine forma, kuri yra lengviau suprantama ir patogesnė naudoti, nei rašytinė informacija. Studentai gali patys pasiimti jiems reikalingą medžiagą patogiu metu, gali nepirkti brangių knygų bei nestovėti eilėse bibliotekose.
- Laboratoriniai darbai – Dažniausiai prie kiekvieno modulio būna tam tikri laboratoriniai darbai. Dėstytojams suteikiama galimybė iš anksto pateikti laboratorinių darbų aprašymus, tai palengvina studentų pasiruošimą darbo atlikimui.
- Tvarkaraštis – Dėstytojas gali pateikti tvarkaraštį ar pranešti apie tvarkaraščio pokyčius. Studentai gali sužinoti apie tvarkaraščio pasikeitimus būdami namuose.
- Modulio skelbimai – Dėstytojas gali pranešti svarbias naujienas studentams, kurie nelanko paskaitų ar neskaito skelbimų lentos.
- Vertinimo sistema – Vertinimo sistema yra apibrėžta iš anksto ir dėstytojas privalo ją pateikti studentams.

2.4 Informacijos sistemų agentų analizė

Agentai yra nepriklausomi objektai (angl. *Stand-alone entities*), kurie kaupia informaciją apie save bei apie kitus agentus tam tikroje sistemoje ir, žinoma, jie gali bendradarbiauti vienas su kitu, siekiant kokio nors tikslo jų aplinkoje[2].

Programinis agentas – tai programinės įrangos objektas, kuris nepertraukiamai ir autonomiškai funkcionuoja sistemoje ir gali bendrauti su kitais agentais ar procesais [1].

Programinio agento koncepcija susideda iš tam tikro mąstymo būdo, kuris yra taikomas kuriant programines sistemas. Programinės sistemos funkcionavimas yra apibūdinamas sudėtinga skirtingų sąveikaujančių ir nepriklausomų programinių objektų kooperacija. Agentas yra vienas programinis objektas, turintis apibrėžtą tikslą. Jis siekia šio tikslo nepriklausomai funkcionuodamas, ir tuo pačiu sąveikauja su savo aplinka arba kitais agentais. Agentas yra charakterizuojamas konkrečiomis autonomijos savybėmis, tikslo orientacija, reakcija, iš anksto numatomu funkcionavimu, atkaklumu (tvirtu valdymo srautu), sąveika (sugebėjimu komunikuoti), ir tam tikrais atvejais adaptyvumu ir mobilumu.

2.5 Informacijos sistemų agentų tipologija

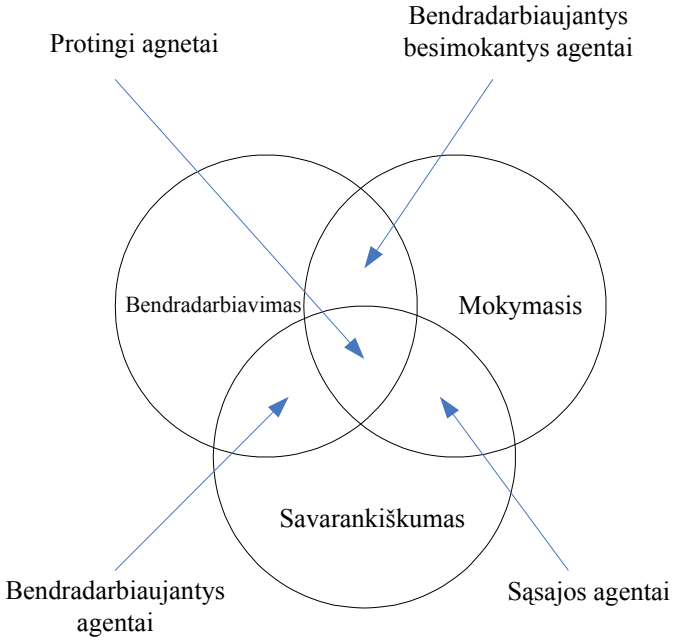
Siekiant suskirstyti egzistuojančius agentus į skirtingas agentų klases, kuriama agentų tipologija. Tipologija nurodo studijuojamų objektų tipus. Yra keletas aspektų, pagal kuriuos galima klasifikuoti egzistuojančius agentus.
Pagal [3] pasiūlytą agentų tipologiją, agentai gali būti klasifikuojami:

Pirma, agentus galima klasifikuoti pagal jų mobilumą, tai yra jų sugebėjimą judėti keliuose tinkluose. Taip agentai skirstomi į *statinius* ir *mobilius* agentus.

Antra, jie gali būti klasifikuojami į *svarstančius* (angl. *deliberative*) ir *reaktyvius* (angl. *reactive).* Svarstantys agentai kilo iš simbolinio samprotavimo modelio: agentai turi vidinę simboliką, priežastingumo modelį ir jie užsiima planavimu ir derybomis, siekiant koordinacijos su kitais agentais.

Trečia, agentai gali būti klasifikuojami pagal idealių ir svarbiausių atributų parodymą, tokių, kaip: *autonomiškumas, kooperacija, mokymasis*. Iš šių charakteristikų išvesti keturi agentų tipai (1 pav.):

1. Bendradarbiaujantieji agentai (angl. *collaborative agents*).
2. Bendradarbiaujantieji besimokantys agentai (angl. *collaborative learning agents*).
3. Sąsajos agentai (angl. *interface agents*).
4. Sumanūs agentai (angl. *smart agents*).



1 pav. - Agentų tipai pagal tipologiją.

Pavyzdžiui, bendradarbiaujantieji agentai (angl. *collaborative agents*) labiau yra linkę į bendradarbiavimą ir autonomiškumą, negu į mokymąsi; tai pat reikia pasakyti, kad bendradarbiaujantys agentai niekad nesimoko. Sąsajos agentai yra daugiau autonomiški ir besimokantys, nei bendradarbiaujantys. Taip pat reikia suprasti, kad daugelis ekspertinių sistemų yra daugiau autonomiškos, bet, kaip tai būdinga, jos nesikooperuoja ir nesimoko. Idealiu atveju, agentai turėtų visas tris veiklas atlikti vienodai gerai, bet tai daugiau siekimas, nei realybė [16].

Ketvirta, agentai kartais gali būti klasifikuojami pagal jų vaidmenis: *informacijos* arba *interneto* agentai. Taip pat informacijos agentai gali būti statiniai, mobilūs arba svarstantys (angl. *deliberative*). Aišku, beprasmiška sudaryti klases tokių nedidelių vaidmenų, kaip pranešimų agentai, pristatymų agentai, analizės ir kūrimo agentai, testavimo agentai, pakavimo (angl. *packing*) agentai ir pagalbos agentai – pagaliau, klasių sąrašas gali būti dar ilgesnis.

Penkta, reikia įtraukti ir *hibridinių* agentų kategoriją, kurie savyje – vienete – apjungia dviejų ar daugiau agentų filosofiją.

Prie visų paminėtų savybių, reikėtų paminėti ir antrinius atributus, 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[3].

Išskiriame septynias agentų tipų kategorijas:

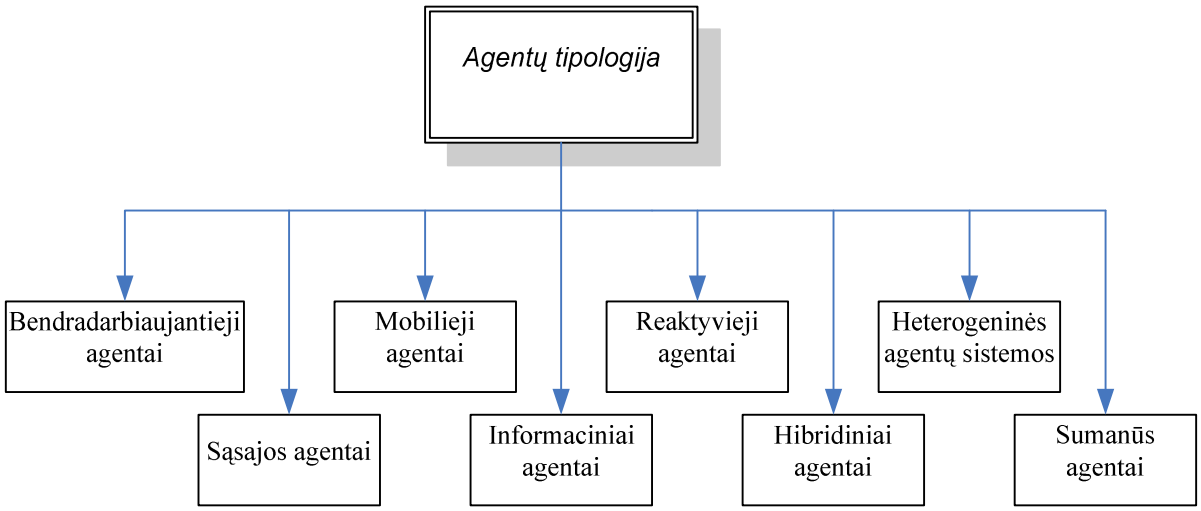
1. Bendradarbiaujantieji agentai (angl. *collaborative agents*).
2. Sąsajos agentai (angl. *interface agents*).
3. Mobilieji agentai (angl. *mobile agents*).
4. Informaciniai agentai (angl. *information/Internet agents*).
5. Reaktyvieji agentai (angl. *reactive agents*).
6. Hibridiniai agentai (angl.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

Gali būti taikymų, kai apjungiami agentai iš dviejų arba daugiau minėtų kategorijų, todėl tai galima vadinti *heterogeninė agentinė sistema*.

2.6 Informacijos sistemų agentų tipų apžvalga literatūros šaltiniuose

Aptarus agentų tipologiją, svarbu apžvelgti agentų tipų sąrašą, tokia tvarka, kaip jie buvo tiriami. Agentai gali būti apžvelgiami, įvertinus jų esmines metaforas, hipotezes/tikslus,

motyvaciją, vaidmenis, prototipų pavyzdžius, potencialias naudas, pagrindinius sunkumus ir kitus bendrus klausimus apie agentų tipus.



2 pav. - Programinių agentų klasifikacija.

Bendradarbiaujantieji agentai (*angl. collaborative agents*), pavaizduoti *1 pav.,* pasižymi autonomija ir bendradarbiavimu su kitais agentais, siekiant atlikti savo savininkų užduotis. Agentai gali mokytis, bet šis aspektas dažniausiai jų darbe nenaudojamas. Tam, kad turėti koordinuotą bendradarbiaujančiųjų agentų grupę, jiems gali tekt derėtis tam, kad būtų pasiekti abipusiai susitarimai tam tikrais klausimais. Pagrindinės šių agentų charakteristikos apima autonomiškumą, socialinius gebėjimus, jautrumą, iniciatyvumą. Jie gali veikti racionaliai ir autonomiškai atvirose, laike apribotose, daugiaagentinėse aplinkose. Agentai turi tendenciją būti statiškais. Jie gali būti geranoriški, racionalūs, teisingi, arba turėti šių savybių kombinaciją, bet gali ir neturėti nė vienos iš šių savybių.[17]

Sąsajos agentai (*angl. interface agents*), palaiko ir suteikia pagalbą, dažniausiai vartotojui, besimokančiam naudotis su tam tikra programa (pvz.: teksto redaktorius, operacinė sistema). Sąsajos agentas stebi vartotojo veiksmus, siūlo geresnius būdus tiems veiksmams atlikti. Taigi sąsajos agentas veikia, kaip autonominis asistentas, kuris dirba kartu su vartotoju, siekiant atlikti tam tikrus darbus programoje. Sąsajos agentai mokosi tam, kad pagerintų pagalbą vartotojui, tai jie atlieka 4 pagrindiniais būdais:

1. Stebėdami ir imituodami vartotoją (mokymasis iš vartotojo).
2. Gaudami teigiamus ir neigiamus vartotojo atsiliepimus (mokymasis iš vartotojo).
3. Gaudami tikslias instrukcijas iš vartotojo (mokymasis iš vartotojo).
4. Klausdami kitų agentų patarimo (mokymasis iš tos pačios grupės).

Jų bendradarbiavimas su kitais agentais yra apribotas, paprastai – klausiant patarimo ir paprastai neužtęsiamas taip, kaip vykdant derybas, kaip tai vyksta bendradarbiavimo agentų atveju.[18]

Mobilieji agentai (*angl. mobile agents*), kompiuteriniai programiniai procesai sugebantys klajoti kompiuteriniuose tinkluose (pvz.: WWW), sąveikauti svetimais šeimininkais, savo šeimininko vardu rinkti informacija, ir grįžti „namo“ atlikus vartotojo numatytas užduotis. Šios užduotys gali būti nuo bilietų rezervavimo iki telekomunikacijų tinklų valdymo. Tačiau mobilumas nėra nei būtina nei pakankama sąlyga leidžianti procesui vadintis agentu. Mobilūs agentai yra agentai dėl to, kad jie yra autonomiški ir bendradarbiauja, nors ir kitaip nei bendradarbiaujantieji agentai. Pavyzdžiui jie gali bendradarbiauti vienam agentui informuojant kitus apie savo vidinių objektų buvimo vietą. Tokiu būdu agentas keičiasi informacija su kitai agentai, bet nebūtina perduoda visą informaciją.[19]

Informaciniai agentai (*angl. information agents*), sukurti dėl įrankių stokos, kurie padėtų susitvarkyti su vis augančiais informacijos kiekiais. Informacijos agentai atlieka vadybos vaidmenį, sumaniai veikdami arba sulygindami informaciją iš daugelio skirtingų šaltinių.

Tačiau reikia paminėti, kad skirtumas tarp informacinių ir bendradarbiaujančių bei sąsajos agentų yra labai mažas. Pagrindinis skirtumas tame, kad informaciniai agentai apibūdinami pagal tai ką jie daro, o bendradarbiaujantieji ir sąsajos agentai apibudinami pagal tai kas jie yra. Dauguma informacinių agentų yra autonominiai ir gali mokintis, bet kadangi jie naudojami WWW aplinkoje, todėl tam tikra prasme jie ir yra informaciniai agentai.[20]

Reaktyvieji agentai (*angl. reactive agents*), priklauso specifinei agentų grupei, kuri neturi vidinio simbolinio aplinkos modelio, vietoj to jie veikiai pagal veiksmas-atoveiksmis būdą. Svarbiausia šių agentų savybė, kad jie labai paprastai sąveikauja su kitais agentais. Tačiau jei į agentų grupę pažvelgsime globaliai, tai galime pastebėti sudėtingų elgesio modelių.[14, 22]

Hibridiniai agentai (*angl. hybrid agents*), 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.[14, 22]

Heterogeninės agentų sistemos (*angl. heterogeneous agent systems*), sistemos, kaip ir hibridinės sistemos, yra integruotos, mažiausiai dviejų ar daugiau agentų, kurie priklauso dviems ar daugiau skirtingų agentų klasių, sistemos. Heterogeninė sistema gali turėti vieną ar daugiau hibridinių agentų.[23]

Sumanūs agentai (*angl. smart agents*), tikslaus šių agentų apibūdinimo nėra. Apskritai sumanūs agentai šiuo metų yra labiau mokslininkų siekimas negu realybė.[8]

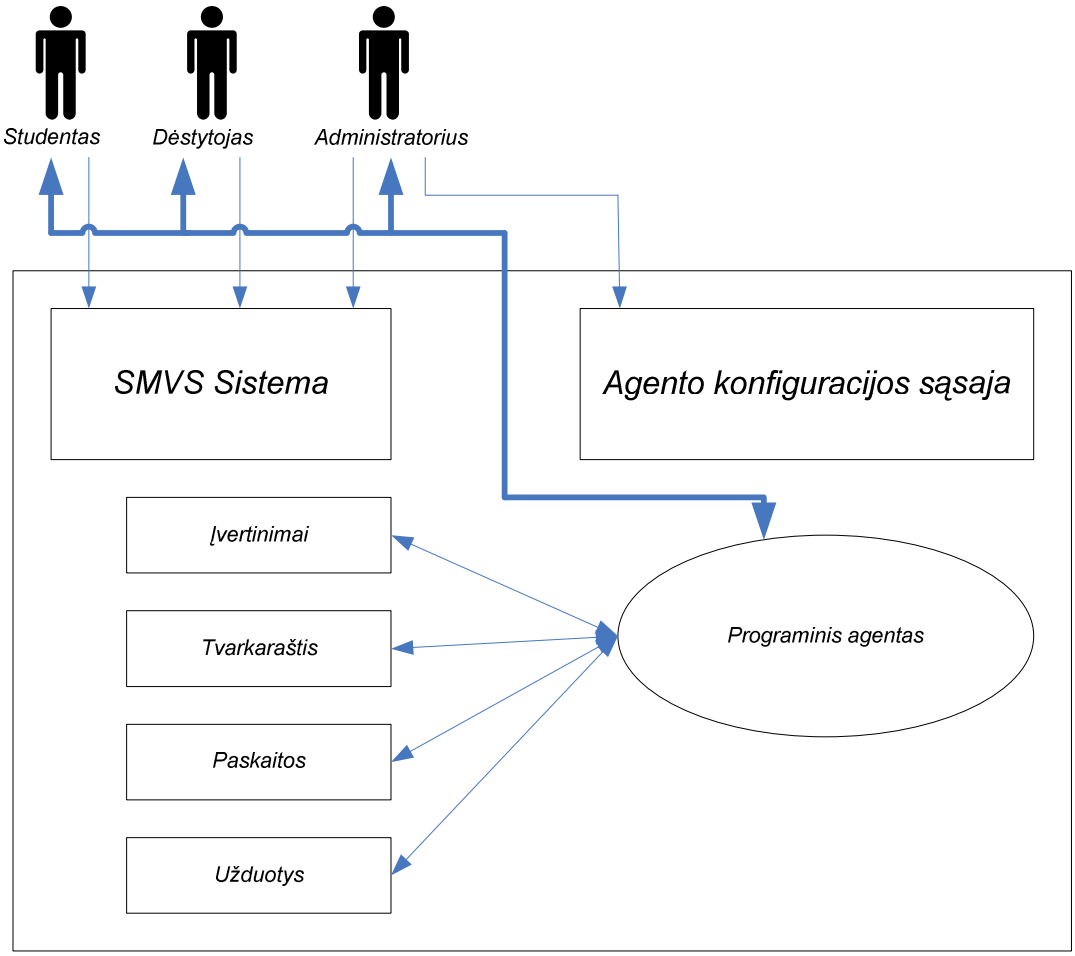
2.7 Agentinės sistemos panaudojimas nuotoliniame mokyme

Nuotolinio mokymosi procese gali būti panaudojami programiniai - informaciniai agentai, kurie veikia kaip dėstytojų asistentai. Programiniai agentai atlieka kurso asistento ar administratoriaus vaidmenį, reaguodami į studentų lankomumą, aktyvumą, pažangumą, siųsdami informacijos pranešimus ar priminimus. Programinių agentų sistemos gali suefektyvinti nuotolinį mokymąsi bei palengvinti jo teikimo procesą.

2.8 Programinio agento tikslo struktūra nuotolinio mokymosi valdymo sistemoje

Programiniai agentai, kaip mokymosi asistentai. Kaip jau minėta anksčiau, pagrindiniai nuotolinio mokymosi dalyviai (studentai, dėstytojai, administratoriai) beveik visą mokymosi laiką yra geografiškai ir socialiai atskirti vienas nuo kito, o nuotolinio mokymosi kursas, patalpintas serveryje, naudojamas kaip aplinka, kurioje jie mokosi, bendrauja, siunčia užduotis.

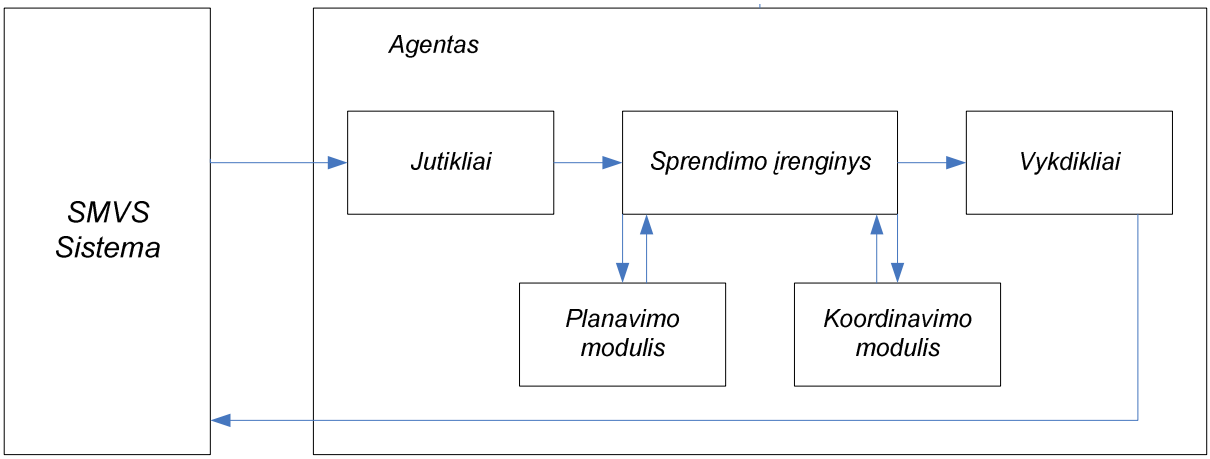
Programiniai agentai nuotolinio kurso aplinkoje gali tarnauti kaip kurso administratoriai, atlikdami užduotis, kurių reikalauja tiesioginis jungimasis prie kurso serverio. *3 pav.* vaizduoja kaip gali veikti programiniai agentai mokymosi aplinkoje. Agentas gali konfigūruoti per agentų konfigūracijos sąsają atlikdamas darbą, kuris priklauso kurso administratoriui. Jis gali gauti informaciją, pvz. apie studentų mokymosi pažangumą, elgseną studijų metu, lankomumą, sukaupti informaciją, ir pateikti tam tikros formos ataskaitą kurso dėstytojui. Agentas gali pasiųsti pranešimą e-paštu, pranešti kurso, dėstytojui ar studentams apie padėtį studijose, pasiųsti įspėjimo signalą ar kitokį priminimą[5].



3 pav. - Programinio agento veikla internetiniame kurse.

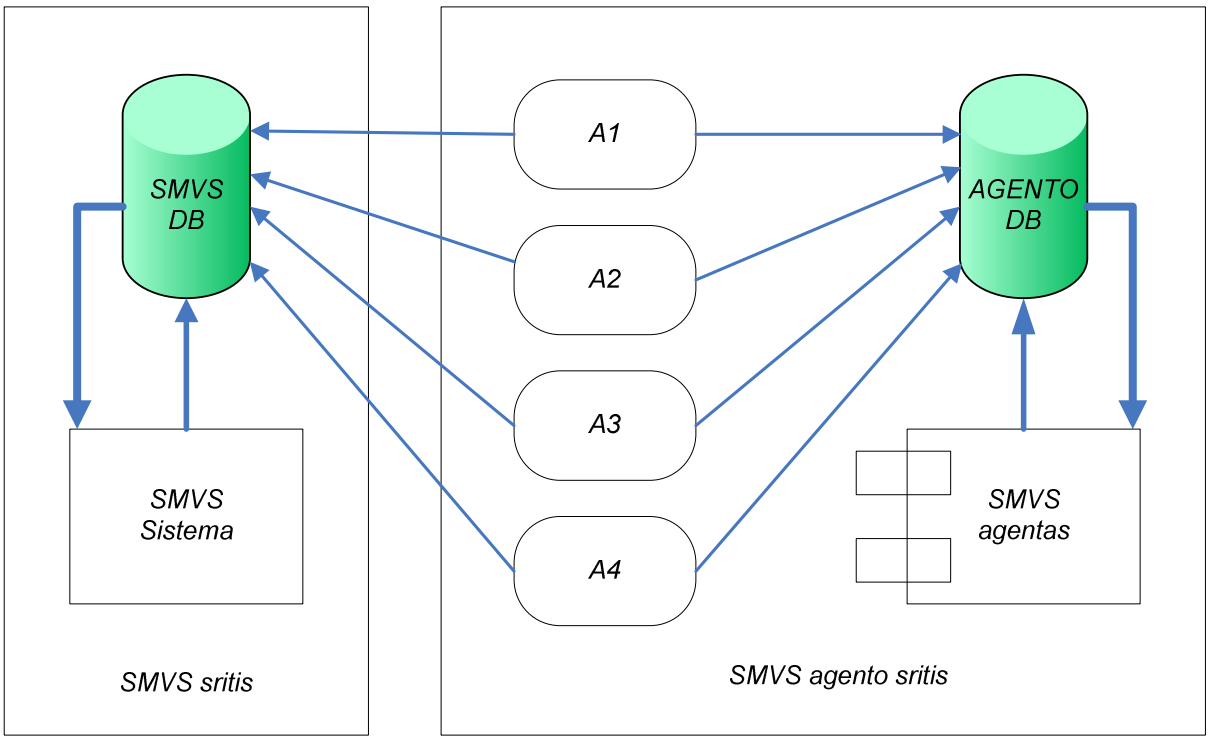
2.9 Agento struktūrinis modelis nuotolinio mokymosi valdymo sistemoje

Klasikinė programinio agento architektūra, kai agentas priima informacija iš aplinkos per jutiklius ir atlieka veiksmus, kurie įtakoją aplinką panaudojant efektorius, pavaizduota *4 pav*.



4 pav. - Klasikinė programinio agento architektūra

Veiksmų pasirinkimas yra atliekamas uždavinių sprendimų įrenginyje (*angl. problem solver*), kuris jei reikia gali pasitelkti planavimo (*angl. scheduling*) ir koordinavimo (*angl. coordination*) moduliais. 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. Nepertraukiamumo ir autonomiškumo reikalavimas kyla iš mūsų noro, kad agentas be žmogaus pagalbos ir pastovaus įsikišimo sugebėtu lanksčiai ir protingai reaguoti į pokyčius tam tikroje aplinkoje.

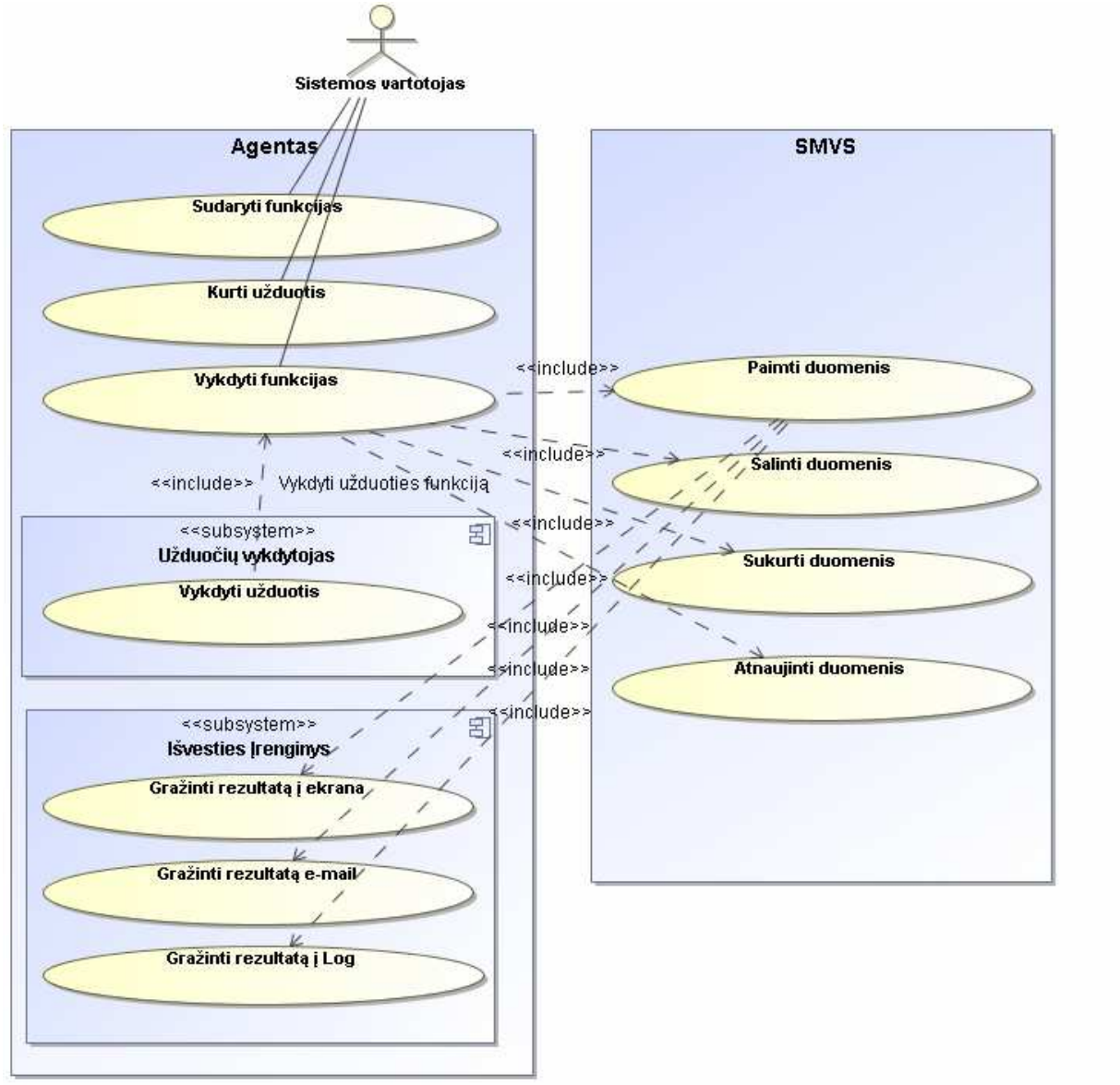


5 pav. - Sistemos su agentais struktūrinis modelis.

3. TEORINĖ DALIS

3.1 Agentinės sistemos funkcionalumo modelis

Agentinės sistemos funkcionalumo modelis pateikiamas 6 *pav.*. Tikslas pavaizduoti grafiškai bendrą funkcionalumo supratimą aprūpinant sistemą aktorių išraišką. Išraiškos tikslas pavaizduoti kaip veikla - funkcionalumo priklausomybe tarp veiklų.



6 pav. - Sistemos Use Case diagrama.

Informacijos analizės agento sistemos vartotojai skirstomi į grupes pagal turimas teises. Teisės tai parametras, kuris parodo kurias grupes funkcijų, gali vartotojas vykdyti. Funkcijos skirstomos į grupes: administravimo (sukurti vartotoją, pašalinti vartotoją), vartojimo 1 (leidimas kurti/šalinti užduotis), vartojimo 2 (funkcijos kurios modifikuoja DB – duomenų bazės duomenis), stebėjimo (funkcijos, kurios atlieka duomenų surinkimą). Teisių lygi nurodo

DB lentelėje users esantis stulpelis number tipo user_type. Šis laukelis atitinkamai parodo vartotojo teisių lygi:

- 0 – administravimo;
- 1 – vartojimo 1;
- 2 – vartojimo 2;
- 3 – Stebėjimo.

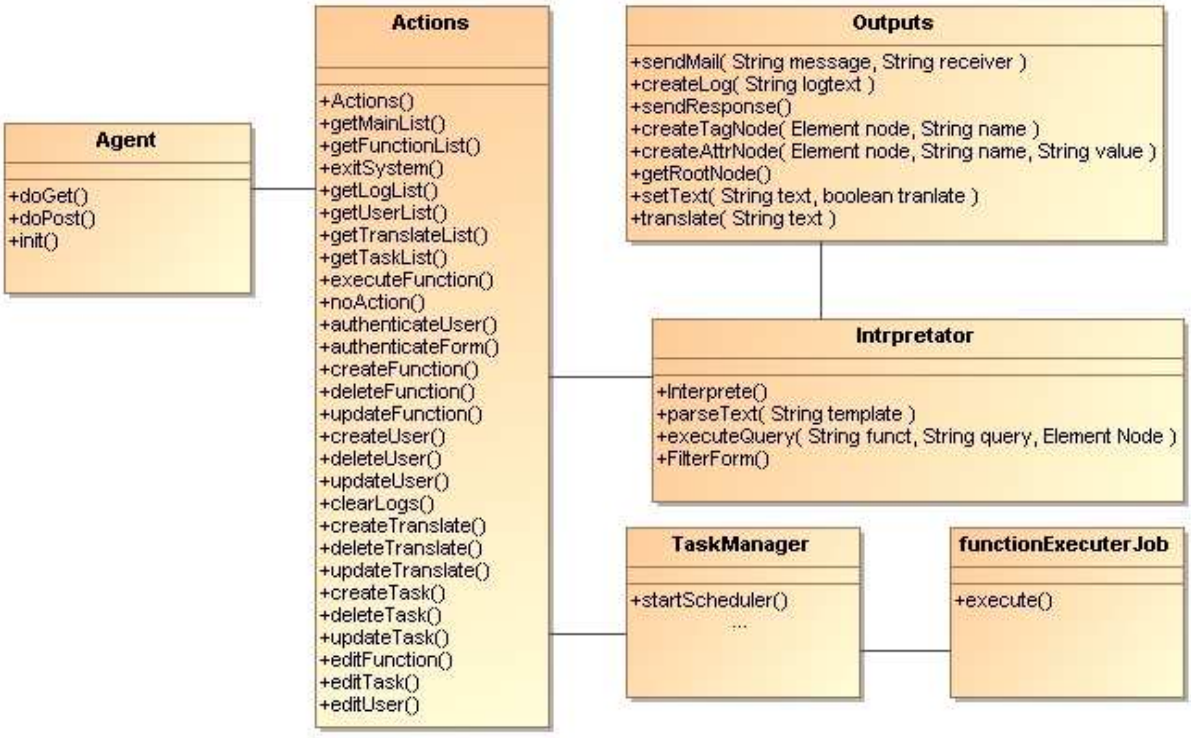
User_type = 0 parodo, kad vartotojas turi visas teises 1 ir 2 ir 3. User_type = 1 turi teises į 2 ir 3.

Vartotojų pagrindinės funkcijos yra kurti naujas funkcijas ir vykdyti jau suformuotas funkcijas. Tik vartotojai ne visas funkcijų grupes gali kurti ar vykdyti priklausomai pagal teises. Visos vartotojų kuriamos funkcijos tai SQL operacijų (*angl. insert, delete, update, drop, ...*) aibės. Tai iliustruoja, funkcionalumo modelio diagramoje esantys ryšiai su SMVS sistema. Visos SMVS sistemos funkcijos tai būtent ir yra jau paminėtos SQL užklausos. Tik funkcijos paimti duomenis veikimas yra išplėstas, surinkti duomenys turi būti pagražinti per posistemę „Išvesties Įrenginiai“. Pasinaudojus posistemės atitinkamą metodą duomenis galima išvesti į naršyklės ekraną, elektroninio pašto laišką, Log sistemą.

Funkcijas be vartotojo gali vykdyti ir užduočių vykdytojas. Užduotis tai funkcija, kuri vykdoma nustatytu laiku ir esant nurodytai sąlygai. Pastarosios sąlygos atitinka agento logika. Galima vykdyti elementarias logines operacijas - reikšmių palyginimą. Tai atitinka reaktyvinį agento modelį, kai jutiklio parodymas susiejamas su tam tikru veiksmu. Užduočių vykdymas atliekamas per posistemę „Užduočių vykdytojas“. Posistemė kuria kiekvienai naujai užduočiai naują procesą, bei administruoja sukurtų procesų aibę. Skirtingai nuo vartotojo, užduočių vykdytojas rezultatą galės gražinti elektroniniu paštu arba log sistemai.

3.2 Informacijos analizės agento statinis modelis

(7 *pav.*) Pateikta statinės struktūros diagrama, aprašanti sistemos klases bei ryšius tarp jų.



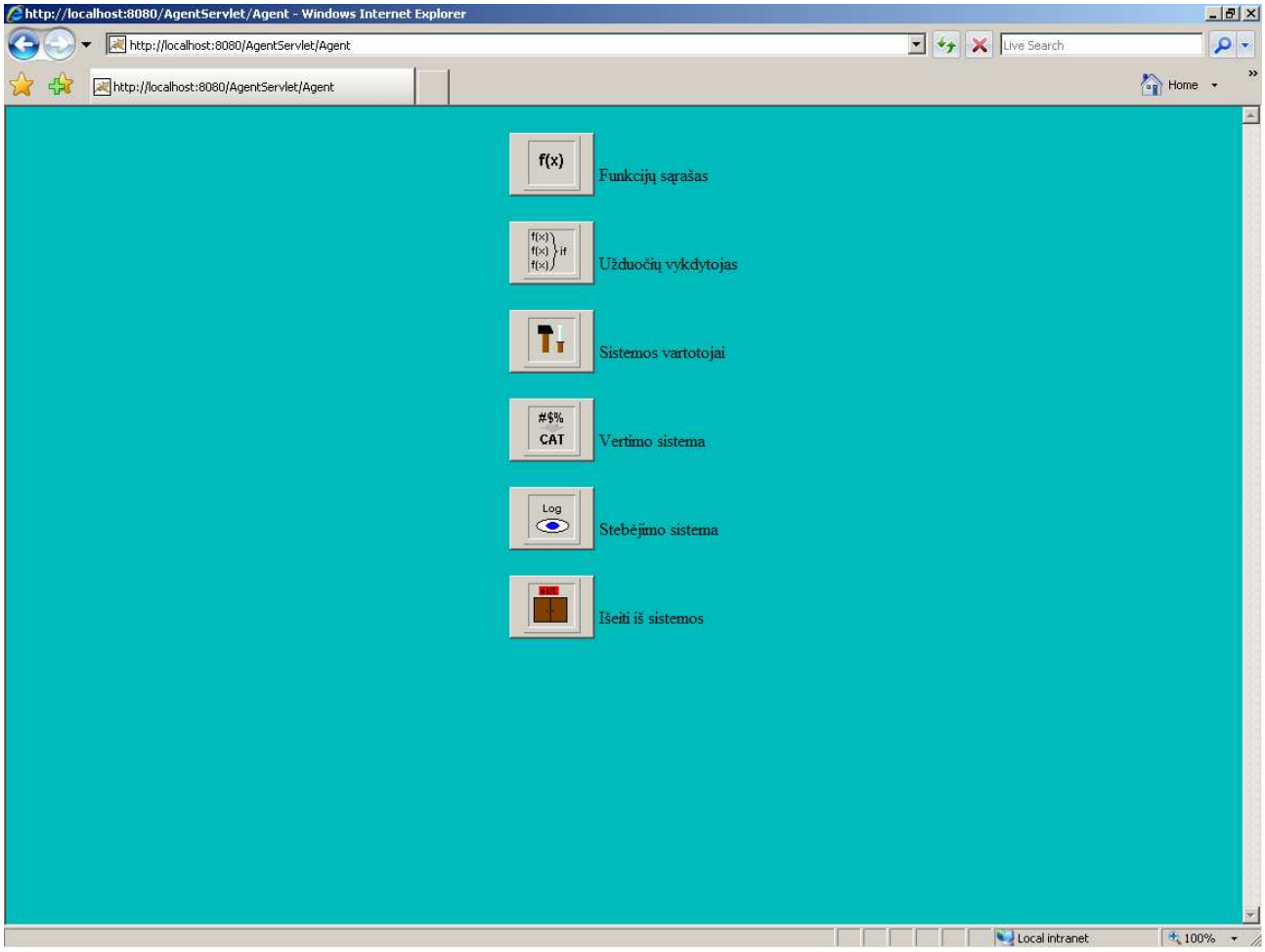
7 pav. - Sistemos klasių diagrama.

Informacijos analizės agento sistema iš šešių klasių:

- Agent – pradinė servleto klasė, kuri turi pagrindinius komunikavimo serverio su klientais metodus *doPost()* ir *doGet()*. Šių metodų pagalba gaunamos užklausos iš kliento su prašymais atlikti veiksmus. Standartinę užklausą sudaro pagrindiniai input tagai.
    - Action – reikšmė kuri pasako servletui kokius veiksmus reikia atlikti;
    - Session – reikšmė kuri pasako per kokią sesiją vartotojas kreipiasi;
    - UserID – reikšmė kuri pasako koks vartotojas atsiuntė užklausą;

  Priklausomai nuo <Action> reikšmės galį būti perduota ir daugiau parametrų. Pvz.: norint įvykdyti funkciją reikia Action nurodyti, kad įvykdyti funkciją, bet taip pat perduoti ir funkcijos kurią norima įvykdyti pavadinimą. *doPost()* metodas gavęs užklausą pirmiausia pasiima Action reikšmę ir startuoja Action klasės atitinkamą metodą, kuris apdoros užklausą. Agent klasė yra *Init()* metodas kurio pagalbą startuojamas foninio veikimo Užduočių vykdytojas.
- Actions – klasė skirta gautom užklausom apdoroti. Šioje klasėje saugomi visi metodai kurie formuoja HTML atsakymo puslapius.
    - *Actions()* – klasės konstruktorius, kuris inicijuoja bendrus klasės kintamuosius.

- *getMainList()* – metodas, kuris startuojamas po prisijungimo. Jis suformuoja pagrindinį sistemos langą su meniu sąrašu (žr. 8 *pav.*), kurio pagalba galima patekti į atskirus sistemoms modulius:
    - Funkcijų sąrašas;
    - Užduočių vykdytojas;
    - Sistemos vartotojai;
    - Vertimo sistema;
    - Stebėjimo sistema (Log);
    - Išeiti iš sistemos;



8 pav. - Sistemos pagrindinis meniu langas.

- *getFunctionList()* – metodas iškviečiamas pasirinkus iš meniu „Funkcijų sąrašas“. Metodas suformuoja visų sistemoje esančių ir prisijungusiam vartotojui priklausančių funkcijų sąrašą. Iš bendro funkcijų sąrašo galima patekti į konkrečios funkcijos redagavimo langą pasirinkus „Edit“ arba „Sukurti“. Taip pat galima grįžti į pagrindinį sistemos langą pasirinkus „Atgal“

be to galima tiesiogiai paleisti funkciją ir stebėti gautus rezultatus paspaudus mygtuką „Vykdyti“.

- *exitSystem()* – metodas inicijuojamas pagrindiniame meniu paspaudus mygtuką „Išeiti iš sistemos“. Metodas išvalo visas vartotojo buvusias sesijas, bei nukreipią į prisijungimo langą.
- *getLogList()* – metodas inicijuojamas pasirinkus „Stebėjimo sistema“ pagrindiniame meniu. Suformuoja visų log pranešimų sąrašą. Šiame sąraše yra tik galimybė išvalyti esančius logus arba grįžti atgal į pagrindinį meniu.
- *getUserList()* – metodas grąžiną visų sistemos vartotojų sąrašą, paspaudus pagrindinio meniu mygtuką „Sistemos vartotojai“. Iš šio sąrašo galima toliau patekti į naujo arba jau įvesto vartotojo kūrimą ar redagavimą. Taip pat galima grįžti į pagrindinį meniu.
- *getTranslateList()* – metodas grąžina keičiamų žodžių sąrašą. Translate modulis gan paprastas, galima daryti naujus įrašus iš dviejų laukų verčiamos frazės ir keičiamą į frazę. Grąžintoje formoje galima atlikti visus redagavimo darbus: kurti naują frazę iššaukus metodą *createTranslate(),* redaguoti seną su metodu *updateTranslate()* ir šalinti nebereikalingus vertimus su metodu *deleteTranslate();*
- *getTaskList()* – metodas inicijuojamas pasirinkus „Užduočių vykdytojas“ iš pagrindinio meniu. Metodas grąžina visų sistemoje aprašytų užduočių sąrašą. Suformuotoje formoje, galima grįžti atgal į pagrindinį meniu arba pereiti į konkrečios užduoties kūrimą ar redagavimą paspaudus atitinkamus mygtukus Sukurti/Edit.
- *executeFunction()* – metodas įvykdo pasirinktą funkciją iš sąrašo ir pagal funkcijos rezultato šabloną suformuoja atsakymą.
- *noAction()* – metodas inicijuojamas tada kai gaunamas neaiškus prašymas iš kliento. Šis metodas suformuoja pranešimą apie tai, kad prašymas neturi priskirto metodo iš Action klasės.
- *authenticateForm()* – metodas inicijuojamas kreipiantis į sistemą per *doGet()* metodą. Tada sistema žino, kad jungiasi naujas vartotojas ir jam sukuria naują sesija, bei grąžiną formą su „VartotojoID“ ir „Slaptažodis“ laukais. Šis metodas inicijuojamas ir visais atvejais, kai pasibaigia sesijos galiojimo laikas arba kreipiantis vartotojui, kuris dar nėra prisijungęs.
- *authenticateUser()* – metodas inicijuojamas kai prisijungimo formoje suvedamas Vartotojo ID ir slaptažodis ir paspaudžiamas mygtukas prisijungti.

Metodas patikrina ar vartotojas atitinką duomenų bazėje suvestiems parametrams ar ne. Jeigu taip tai vykdoma funkcija kuri perduodama per formos Action parametrą, jei ne inicijuoja *authenticateForm()* iš naujo.

- *editFunction()* – šis metodas inicijuojamas tada, kai vartotojas nori redaguoti seną funkciją arba kurti naują. Metodas suformuoja ir grąžina formą su laukais skirtais funkcijai ir jos parametrams įvesti. Šios formos pagalba galima inicijuoti *createFunction(), updateFunction(), deleteFunction()* metodus, kurie atitinkamai sukurią naują funkciją pagal įvestus duomenis, atnaujina duomenis pagal įvestus duomenis arba pašaliną pasirinktą funkciją.
- *editUser()* – metodas inicijuojamas tada, kai vartotojas nori sukurti naują arba redaguoti seną vartotoją iš bendro vartotojo sąrašo. Metodas suformuoja formą su laukais skirtais įvesti vartotojo duomenis. Pasirinkus sukurti, atnaujinti ar pašalinti atitinkamai inicijuojami metodai *createUser()*, kuris sukuria naują vartotoją sistemoje, *updateUser(),* kuris atnaujiną seno vartotojo parametrus, bei *deleteUser()* kuris pašalina pasirinktą vartotoją.
- *editTask()* – metodas inicijuojamas, kai vartotojas užduočių sąraše paspaudžia mygtukus „Sukurti“ arba „Edit“. Tada metodas suformuoja užduoties parametrų įvedimo formą ir pateikią ją vartotojui. Užpildžius laukus ir pasirinkus „Sukurti“, ar „Atnaujinti“ ar „Pašalinti“ inicijuojami atitinkami metodai *createTask()/updateTask()/deleteTask(),* kurie atitinkamai kurią naują užduotį, atnaujiną senąją arba pašaliną pasirinktą užduotį. Šie metodai daro atnaujinimus duomenų bazės lentelėje tasks, bei vykdo metodą *startSheduler()* iš klasės TaskManager, kuris stabdo visas užduotis, kurios jau startavusios ir užduočių sąrašą startuoja iš naujo.

- Interpretator – funkcijų atsakymo formavimas susideda iš gana sudėtingų veiksmų, tam sukurta atskira klasė kuri atliktų šablono analizę, duomenų surinkimą patalpinimą į atsakymą ir išsiuntimą klientui. Ši klasė turi šiuos metodus:
  - *Interprete()* – tai pagrindinis metodas, kuris inicijuoja, atsakymo formavimą pagal gautą šabloną per parametrus.
  - *parseText()* – metodas skirtas šabloną išskaidyti į tagus ir juos sudėti į atsakymo buferį.
  - *executeQuery()* – metodas pagal gautą <SQL> tago turinio užklausą iš duomenų bazės surenka duomenis ir juos patalpiną į to pačio tago vietą.
  - *FilterForm()* – jeigu atsakymas formuojamas tiesiogiai klientui į ekraną, o ne į e-paštą tai papildomai prie kiekvienos užklausos <SQL> tago suformuoja filtro

zoną. Filtras formuojamas pagal tuos elementus kurie duomenų bazėje parameters lentelėje yra nurodyti kaip „filter“ tipo.

- *Outputs* – klasė kuri yra galutinė, ji turi buferinę atmintį ir metodus skirtus buferiui užpildyti HTML tekstu. Taip pat turi metodus suformuoti atsakymą kurį gali išsiųsti klientui į ekraną ar į e-pašto adresą arba į Log sistemą. Šios sistemos visi duomenys talpinami duomenų bazės lentelėje agent_logs.
- *createTagNode()* – metodas naujam tagui suformuoti;
- *createAttrNode()* – metodas nurodyto tago atributams suformuoti;
- *setText()* – metodas tekstui tarp tagų įterpti;
- *sendResponse()* – išsiunčią jau suformuotą HTML tekstą kaip atsakymą klientui į ekraną;
- *sendMail()* – išsiunčia suformuotą HTML tekstą, kaip atsakymą į e-paštą;
- *addLog()* – įterpia į Log sistemą norimą pranešimą;

- TaskManager – klasė skirta startuoti užduočių vykdytoją. Tai foninė programa, kuri nepriklausomai ar gautos užklausos ar ne atlikinėja duomenų analizę, duomenų persiuntimą vartotojui elektroniniu paštu. Ši klasė turi tik vieną metodą *startScheduler()* kuris surenka iš duomenų bazės visas aprašytas užduotis ir jas sudeda į atskiras fonines programas, kurios vykdomos pagal nurodytus laiko tarpus. Šioje sistemoje naudojamas (*ang. Quartz scheduler*) darbo planavimo sistema. Pagal jos architektūrą metodas sukuria trigerį kuris veikia tam tikrais laiko momentais. Trigeriui galima priskirti darbus (*ang. Job's*). Darbas turi būti aprašomas standartizuota klase. Sukurtoje agentinėje sistemoje ši klasė realizuota functionExecuterJob pavadinimu. Šios standartizuotos klasės turi turėti privalomą metodą *execute(),* kuris ir vykdomas suveikus trigeriui. Kadangi šios klasės pagrindinė paskirtis atlikti funkcijų vykdymą tam tikrais laiko momentais tai jos funkcionalumas yra identiškas Action klasės metodo *executeFunction()* funkcionalumui. Tik nėra formuojama Filtro zona ir atsakymas siunčiamas visada tik e-paštu.

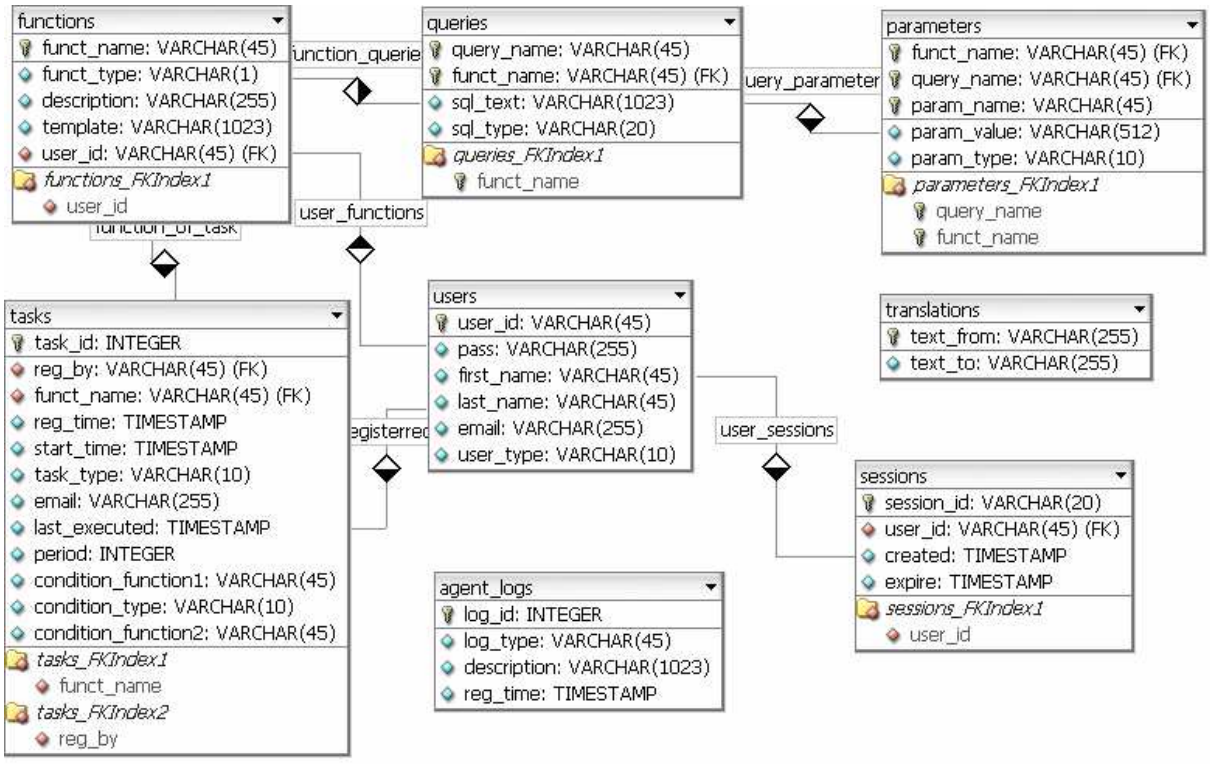
3.3 Informacijos analizės agento duomenų bazės struktūra

Tam kad sistemą būtų lengviau derinti ir papildyti naujomis funkcijomis, Servleto veikimas realizuojamas, kaip taisyklių vykdytojas, o visos taisyklės saugomos duomenų bazėje. Informacijos analizės agento sistema atlikinės aibę funkcijų, kurias vykdys pasijungęs vartotojas, šiuo atveju dėstytojas. Jis galės ištraukti statistika apie pažymius, lankomumą, sukurti priminimą apie skolininkus. Kiekviena funkcija sistemoje suprantama kaip SQL

operacijų (*angl. insert, update, select, create, drop*) aibė. Savo ruožtu kiekviena SQL operacija turės parametrų aibę, kurie bus naudojami kaip filtrai „where“ sąlygose arba kaip „values“ laukeliai „insert“. Servletas visas funkcijas vykdys naudodamas bendrą mechanizmą.

- Identifikuos funkciją;
- Kiekvienai funkcijos SQL operacijai surinks reikalingus laukus pagal užklausai priskirtus parametrus;
- Įvykdys užklausas;
- Gražins gautą rezultatą pagal funkcijoje nurodytą šabloną.

Visas reikalingos taisyklės saugomos duomenų bazėje. Tokios sistemos funkcijų apdorojimas realizuojamas duomenų bazės struktūra pateikta (žr. 7 *pav*.)



9 pav. - Agento duomenų bazės struktūra.

3.4 Informacijos analizės agento duomenų bazės modelis

Sistemoje realizuota galimybė aprašinėti funkcijas, kurios kuria naujus trigerius SMVS sistemos duomenų bazėje kkpes4_kursams. Be trigerių kūrimo, galima aprašyti funkcijas kurių pagalba kuriamos duomenų bazėje naujos, šalinamos lentelės ar papildomos SQL užklausos. Negalima naudoti Stored procedūrų ir funkcijų. (Stored procedūros - tai TSQL ar PL/SQL kalbomis aprašytos SQL užklausų skriptas)

Tam, kad galėtume papildomai rinkti informacija apie SMVS sistemos vartotojus sukurti du trigeriai (count_user, act_user):

- Trigeris skirtas fiksuoti laiką apie naujai sukurtą vartotoją SMVS sistemoje.

1 lentelė. Trigeris count_user

Laukas	**Aprašymas**
Trigeris	`count_user`
Veiksmas	`UPDATE`
Lentelė	`kkpes4_kursams.vartotojai`
Būsena	`insert into kkpes4_kursams.vartotoju_aktyvumas SET user_id = new.user_id, action_time = now(), action_type = 'new user', description = 'Sukurtas naujas vartotojas';`

- Trigeris skirtas fiksuoti laiką apie vartotojų aktyvumą SMVS sistemoje.

2 lentelė. Trigeris act_user

Laukas	**Aprašymas**
Trigeris	`act_user`
Veiksmas	`UPDATE`
Lentelė	`kkpes4_kursams.vartotojai`
Būsena	`insert into kkpes4_kursams.vartotoju_aktyvumas SET user_id = new.user_id, action_time = now(), action_type = 'Action', description = 'Vartotojas buvo aktyvus';`

Pateikiamos agentinės sistemos duomenų bazės lentelės su aprašais ir kam jos naudojamos.

Duomenų bazės lentelės functions (žr. 9 *pav.*) aprašas:

Functions = < funct_name, description, template, funct_type, user_id, fuct_permit >

- Funct_name - Funkcijos pavadinimas ir raktas (Raktinis);
- Description - Funkcijos aprašymas;
- Template - Funkcijos rezultatui gražinimui skirtas šablonas;
- Funct_type - Parodo ar funkcija skirta jutikliųrių (S) (Sensor) ar valdikliui (E)(Executer);
- User_id - Vartotojo vardas;
- Fuct_permit - Parodo funkcijos saugumo lygį. Jis skirtas tam, kad nevisi vartotojai galėtų atlikinėti.

Naudojantis programinio agento klasikine architektūra (žr. 4 *pav.*) Funkcijos skirstomos į dvi grupes: Jutiklių ir Vykdiklių. Šias grupes nurodo funct_type laukas. Jutikliai tai funkcijos, kurios bus skirtos nurodyti sąlygoms kuriant užduotis. Jos turi daugiau apribojimų lyginant su vykdiklių funkcijomis. Jutiklio funkcija privalo grąžinti tik vieną reikšmę, negali gražinti reikšmių aibės. Jutikliams būdinga informacijos surinkimo operacijos, taigi bus naudojama tik "select" SQL užklausos. Visos jutiklių funkcijos gali būti naudojamos kaip valdiklio funkcijos, bet atvirkščio ryšio nėra, dėl

jutikliams būdingų apribojimų. Vykdikliai tai veiksmai, kuriuos sistema darys įvykus tam tikram įvykiui. Įvykiais gali būti koks nors tikėtas ar netikėtas jutiklio rezultatas arba gautas prašymas įvykdyti funkciją (dėstytojas paspaudė mygtuką, kuris realiu laiku atvaizduoja skolininkų sąrašą). Kadangi funkciją sudaro aibė SQL užklausų tai tam, kad funkciją gražintų tik vieną rezultatą, o ne jų aibė naudojamas šablonas (Skriptinis aprašas). Šabloną gali sudaryti statinis tekstas (HTML tekstas) arba dinaminis (konkrečių užklausų gautas rezultatas). Dinaminės užklausoms bus realizuotas interpretatorius, kuris tarp simbolių # # įrašytą tekstą interpretuos kaip kodo dalį, kurią reikią įvykdyti ir gautą rezultatą gražinti šablonui. Pavyzdžiui #SQL(skolininkai)# bus įvykdytą užklausą skolininkai ir gautas skolininkų sąrašas grąžintas šablonui. Šablonas persiunčiamas išvesties įrenginiui. Šioje sistemoje bus naudojami trys išvesties įrenginiai:

- elektroninis paštas;
- ekranas per naršyklę;
- log'ai.

Pastarasis skirtas labiau pranešimams apie sistemoms klaidas. Visos funkcijos bus siejamos su vartotoju, kiekvienas prisijungęs vartotojas galės turėti savo asmenišką funkcijų sąrašą.

Duomenų bazės lentelė queries (žr. 9 *pav.*) skirta funkcijos užklausom saugoti.

Queries = < query_name, sql_query, query_type, funct_name >

- Query_name - Užklausos pavadinimas (Raktinis);
- Sql_query - Užklausos pradinis tekstas. Po to ši užklausa bus naudojama kaip view‘as, kuris po to bus papildomas užklausos parametrais;
- Query_type - Nurodo užklausos tipą. Pagal jį Servletas supras kokioje formoje pagražinti rezultatą. Pvz jeigu tai select užklausą tai servletas rezultatą įdės į <table>;
- Funct_name - funkcijos pavadinimas, kuriai priklauso ši užklausa(Raktinis).

Sistema vykdydama funkcija įvykdo iš eilės visas funkcijas, kurios nėra skirtos duomenims surinkti t.y. "select". Naudojamas funkcijos rezultato šablonas ir vykdomos visos rezultatui suformuoti skirtos užklausos. sql_type laukelis naudojamas dekoduoti kokio tipo užklausą. Jis labiau aktualus suskirstyti "select" užklausas ar surinkus duomenis įdėti į <table>, ar <item> html elementus.

Duomenų bazės lentelė parameters (žr. 9 *pav.*) skirta queries lentelėje saugomų užklausų parametrams nurodyti.

Parameters = < query_name, Funct_name, param_name, param_value, Param_type >

- Query_name - Ryšys su užklausa, kuriai šis parametras priklausys (Raktinis);
- Funct_name - Funkcijos pavadinimas (Raktinis);

- Param_name - Parametro pavadinimas(Raktinis);
- Param_value - Parametro reikšmė;
- Param_type - Parametro tipas. Jis servletui parodys kam šis parametras skirtas. Pvz.: „where“ nurodys, kad šį parametrą reikia naudoti where sąlygoje. „display“ nurodys, kad šį parametrą reikia naudoti formuojant grafinį informacijos rezultatą. Sakykim jei rezultatui naudojamas <table> tai čia bus galima nurodyti table atributus. (<table border = “1”>).

Vykdant kiekvieną užklausa bus norima nufiltruoti konkrečia informaciją, tam Servletas sugeneruos atitinkama filtro lauką pagal parameters lentelėje esančius parametrus. Taip pat parametrų lentelėje gali būti parametrų skirtų užklausos rezultatui pagražinti reikiamų HTML atributų reikšmių. Pvz. query lentelėje nurodoma, kad rezultatas turi būti gražintas į <table> taigi parameters lentelėje bus galima rasti border=”1” ir panašius atributus. Kam skirtas parametras parodo param_type laukelio reikšmė.

Duomenų bazės lentelė users (žr. 9 *pav.*) skirta agento sistemos vartotojams saugoti.

Users = < user_id, pass, first_name, last_name, email, description, User_type >

- User_id - Sistemos vartotojo ID (raktinis);
- Pass - Vartotojo prisijungimo prie sistemos slaptažodis;
- First_name - Vartotojo vardas;
- Last_name - Vartotojo pavardė;
- Email - Vartotojo elektroninio pašto adresas;
- Description - Vartojo aprašymas;
- User_type - Vartotojo tipas skirtas privilegijos valdyti.

Ši lentelė turi ryšį su functions ir tasks lentelėmis. Šie ryšiai reikalingi tam, kad būtų galima suskaidyti funkcijas pagal vartotojus ir užduotis pagal vartotojus. Taip pat svarbus laukelis yra user_type, kuris nurodys teisių lygi. Teisių lygis nurodys galimas funkcijas. Pvz.: bus galima suskirstyti vartotojus kurie turės teisę į funkcijas, kurios atlieka tik duomenų surinkimą „select“.

Duomenų bazės lentelė translations (žr. 9 *pav.*) skirta pavadinimams versti.

Translations = < text_from, text_to >

- Text_from - Tekstas kurį reikią pakeisti;
- Text_to - Tekstas į kurį reikia pakeisti.

Servletas naudoja duomenų bazės lentelių stulpelių pavadinimus kaip informacijos atvaizdavimui. Taigi reikia tokiems žodžiams vertimo.

Duomenų bazės lentelė tasks (žr. 9 *pav.*) skirta formuoti užduotims, kurios bus vykdomos fone.

Tasks = < task_id, funct_name, reg_time, start_time, task_type, reg_by, email, last_executed, period, condition_function1, condition_type, condition_function2 >

- Task_id - Užduoties id (raktinis);
- Funct_name - Funkcijos pavadinimas;
- Reg_time - Sukūrimo laikas;
- Start_time - Funkcijos vykdymo laikas;
- Task_type - Užduoties tipas. Periodinis, vienkartinis ir pan;
- Reg_by - Tai vartotojo user_id, kuris sukūrė užduotį;
- Email - Pašto adresas į kurią reikia siųsti rezultatą, jeigu jis yra. Pagal nutylėjimą e-paštas lygus vartotojo e-paštui. Jeigu tuščia tai rezultatas negražinamas;
- Last_executed - Paskutinį kartą vykdymo laikas arba žymė, kad užduotyje yra klaidų;
- Period - Vykdymo, periodas. Kas valanda, kas diena ir t.t.;
- Condition_function1 - Funkcija arba statinė reikšmė pagal kurią bus tikrinama sąlyga;
- Condition_type - Sąlyga (< > =);
- Condition_function2 - Funkcija arba statinė reikšmė pagal kurią bus lyginama pirmos funkcijos rezultatas.

Užduotis tai funkcija, kurią reikia įvykdyti tam tikru laiku arba vykdyti periodiškai tam tikru laiku. Gautas rezultatas visada gražinamas elektroniniu laišku arba visai negražinamas. Taip pat užduotis turės galimybę būti valdomos per elementarias logines operacijas (<,>,=). Taigi periodas nurodys funkcijos vykdymui reikalingų sąlygų tikrinimo periodą, o sąlygos parodys ar vykdyti nustatytą funkcija ar ne. Sąlygų tikrinimo mechanizmas elementarus. If (Condition_function1 operatorius (Condition_type) Condition_function2) then vykdoma funct_name;

Duomenų bazės lentelė agent_logs (žr. 9 *pav.*) viena iš informacijos išvesties formų.

Agent_logs = < id, log_type, description, reg_time >

- ID - Logo id (raktinis);
- Log_type - Tipas kuris nurodo ar tai, klaida, ar tai tiesiog pranešimas apie įvykį (Pvz. apie įvykdytą užduoti) ir kt.
- Description - Įvykio aprašymas;
- Reg_time - Įvykio užfiksavimo data.

Jį labiau skirta išvedinėti pranešimus apie sistemos veiklą, esama būkle ir kitą tarnybinę informacija.

Duomenų bazės lentelė sessions (žr. 9 *pav.*) ji skirta agento saugumui užtikrinti.

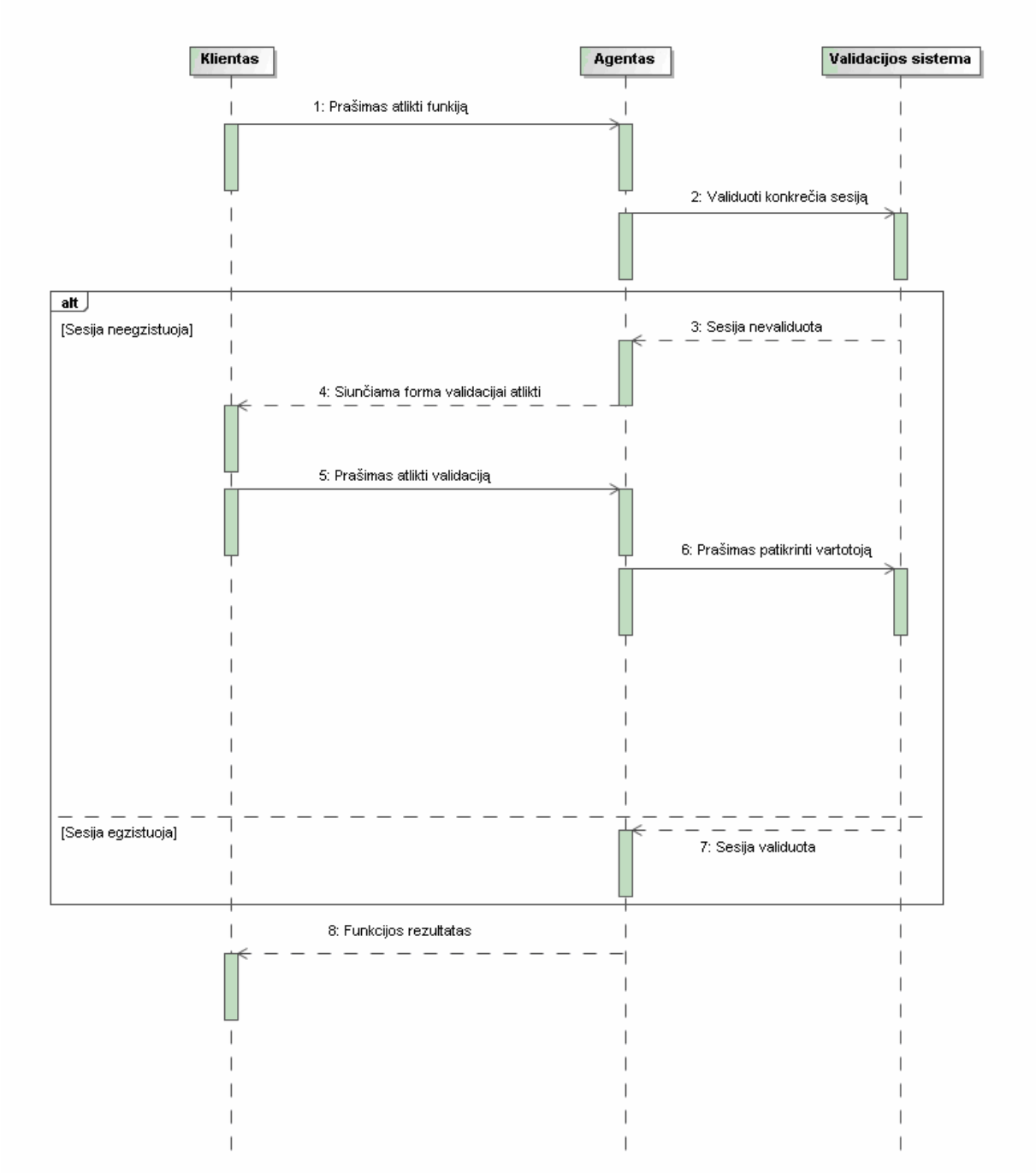
Sessions = < session_id, user_id, created, expire >

- Session_id – kliento prisijungimo sesijos numeris. Tai unikalus skaičius identifikuojantis prisijungusį prie sistemos;
- User_id – vartotojo id, kuriam priklauso sesija;
- Created – laikas kada sesija buvo sukurta;
- Expire – laikas kada sesijos galiojimo laikas pasibaigs; Kiekvieną kartą serveriui gavus užklausą iš kliento tikrinamas esamas laikas jai jis didesnis už expire laiką tai sesiją naikinama ir prašoma vartotoją prisijungti iš naujo. Jei esamas laikas mažesnis už expire laiką didinama sisteminė konstanta laiko tarpu ir toliau agentas vykdo kliento nurodytus veiksmus.

Lentelė skirta agento saugumui užtikrinti. Atėjus kiekvienai užklausai iš kliento bus tikrinama ar vartotojas jau buvo prisijungęs. Jeigu taip tai bus įvykdyta jo užklausa jei ne bus išmestas puslapis, kad reikia atlikti saugumo patikrinimą.

3.5 Informacijos analizės agento sistemos saugumas

Sekų diagramą iliustruoja objektų, jų būsenų, veiksmų lygiagretų išsidėstymą laike bei pranešimus tarp jų, kuri aprašo patikimumo patvirtinimo mechanizmą (žr. 10 *pav*.)



10 pav. - Agento sekų diagrama.

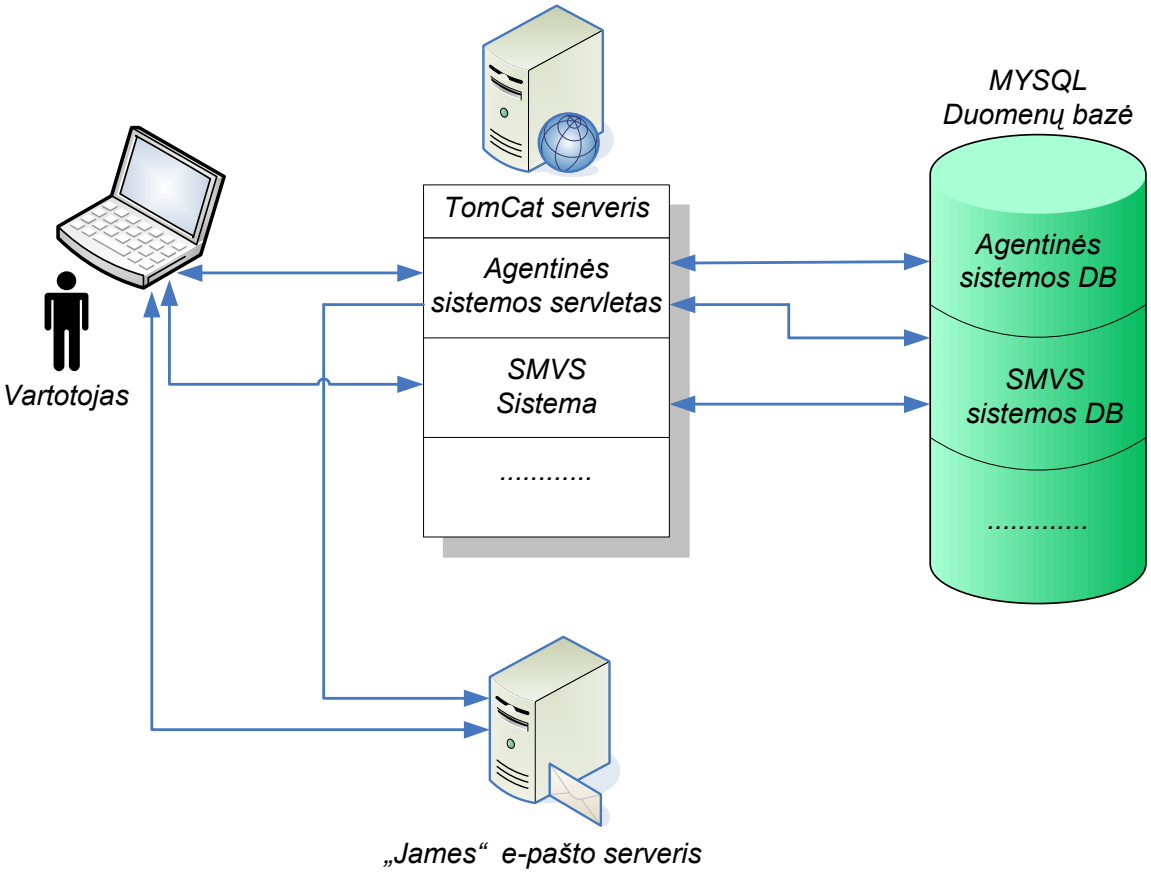
Visą sistemos saugumo patikrinimą atlieka servletas. Prisijungus prie sistemos pradžioje reikalaujama atlikti identifikaciją. Prašoma įvesti vartotojo vardą ir slaptažodį. Atlikus įvestų duomenų patikimumo patvirtinimą kuriamas naujas sesijos numeris duomenų bazėje ir suteikiamas jis konkrečiam vartotojui. Su kiekviena nauja užklausa vartotojas siunčią sesijos numerį. Serveris gavęs užklausą patikrina ar sesijai dar nepasibaigęs galiojimo laikas. Jeigu laikas pasibaigęs tai klientui siunčiama pradinė vartotojo įvedimo forma tam, kad patvirtinti sesiją iš naujo. Jeigu patikimumo patvirtinimo laikas nepasibaigęs tai įvykdoma vartotojo nurodyta funkcija. Sistemoje nėra kliento būsenos tikrinimo. Taigi nėra apibrėžta kokia tvarka klientas turi atlikinėti veiksmus. Jeigu Agento sistemai taisyklingai

suformuojamas prašymas ir prašymas praėjo sesijos patikimumo patvirtinimo mechanizmą tai funkciją gali būti atlikta nebūtinai tik tuo momentu kai bus paspausta atitinka mygtukų (nuorodų) seka. Įvykus sistemoje bet kokiai neapibrėžtam veiksmui bus naikinam sesija ir prašoma per naujo prisijungti prie sistemos. Neapibrėžti veiksmai - sistemai nežinomos užklausos arba užklausos su trūkstamais parametrais.

Saugesniam sistemos naudojimui Serveltas vietoj doGet() standartinio metodo naudoja tik doPost() metodą. Pastarasis metodas parametrus pasiima per formos elementus, kurie vizualiai nematomi. Metodas doGet() parametrus gauna per naršyklės adresą, taigi lengvai matomi visi perduodami parametrai. Sistemoje slaptažodžiai koduojami md5 koduote.

3.6 Agentinės sistemos architektūra

Sistemą sudaro trys pagrindinės dalys TomCat serveris, MySQL duomenų bazė ir e-pašto serveris, naudojamas nemokamas „James“ e- pašto serveris. (žr. 11 *pav.*)



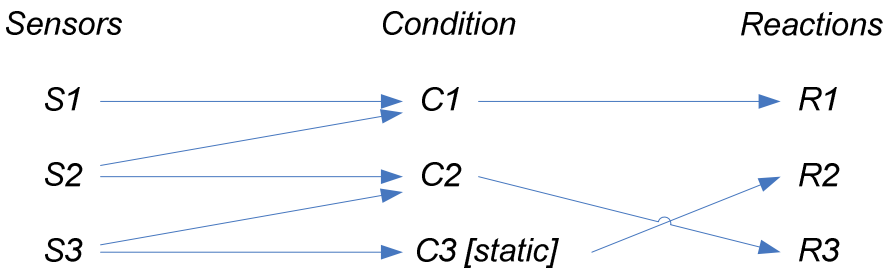
11 pav. - Agentinės sistemos architektūra.

TomCat serveryje patalpintas informacijos agento sistemos servletas, kuris atliks visus sistemos valdymo darbus. Ryšys su Studijų Modulių Valdymo Sistema atiliekamas duomenų bazės lygyje. Agento sistema tai autonomiška sistema kurią galima programuoti

atlikti įvairius duomenų manipuliavimo darbus įvairiose duomenų bazėse. Šiuo atveju Studijų Modulių Valdymo Sistema duomenis. Tačiau per nustatymus galima prisilinkinti bet kurią fiziškai pasiekiamą duomenų bazę. Atliekamos funkcijos tai SQL užklausų aibė kurios būdingos visoms duomenų bazėms. Vienintelis skirtumas tas, kad funkcijos turi būti siejamos su konkrečia duomenų bazės struktūra. Tai atliekama panaudojus funkcijos parametrus. Įvykdytos funkcijos rezultatas - tai funkcijos programuojamojo šablono apibrėžtas tekstas. Agentas įvykdęs funkcijos SQL užklausas, sekančiu etapu formuoja atsakymą, pagal šablone nurodytas taisykles. Suformuotas atsakymas gali būti siunčiamas „išvesties įrenginiui“. Pastarasis įrenginys tai klasė kuri savyje saugo atsakymo pranešimą, kurią vėliau per atitinkamą metodą galimą gražinti klientui tiesiogiai į naršyklės ekraną HTTP protokolu arba rezultatą gražinti e-pašto forma per „James“ serverį arba suformuotą žinutę perduoti sistemos sekimo sistemai (Log). Ji funkcijos rezultatą išsaugo duomenų bazėje. Rezultatas gali būti peržiūrimas vėliau atsidarius „Log“ sistemą. Visos Agento sistemos funkcijos gali būti atliekamos netiesiogiai per „užduočių vykdytoją“. Pastaroji sistema tai sheduler pagrindu veikiantis mechanizmas. Tam naudojama „Quartz“ biblioteka.

3.7 Reaktyvinio agento realizavimas sistemoje.

Šiame darbe realizuotas “Reaktyvinis” agentas. Tai viena paprasčiausių agentų grupių. Jų veikimas paremtas Veiksmų (Sense) ir Atoveiksmių (Reaction) pagrindų, kai kiekvienas sistemoje esanti jutiklius yra susietas su reakcija. Tiksliau jutiklis parodo ar jo administruojamą reakcija reikia įvykdyti ar ne. Realizuotoje sistemoje jutikliai yra select tipo užklausos kurios turi nurodytą tipą ‘S’ (jutiklius). Tokios užklausos vykdymo metu, iš duomenų bazės gali ištraukti įvairius rodiklius. Gautas rodiklis lyginamas su kitu rodikliu kuris gali būti statinis arba kito jutiklio gautas rodiklis tai pat iš duomenų bazės ar kitos sistemos. Rodiklių lyginimo rezultatas ir parodo ar vykdyti reakciją ar ne. Reakcijos realizuotoje sistemoje yra funkcijos (SQL užklausų aibė). Pavyzdžiui jutiklių gali būti užklausa, kuri iš duomenų bazės ištraukia studento semestro modulių pažymių vidurkius kurie mažesni už 5. Lyginamasis rodiklis galėtų būti 1, kuris parodytų ar tokių modulių yra daugiau ar mažiau už 1, o reakciją galėtų būti elektroninio pašto su pranešimu apie gresiančią skolą išsiuntimas visiems studentams, kurie papuola į ribinę sąlygą >=1. Tai pat reiktų nustatyti laiką, kad šią funkciją vykdyti sesijos metu kas savaitę.
Agento veikimą būtų galima pavaizduoti diagrama pateikta (žr. 12 *pav*.). Arba funkcija: f := Rx(Ca(Sm,Sn)).



12 pav. - Reaktyvinio agento veikimo diagrama.

Atitikimas sukurtai sistemai Sensors = užklausos su tipu 'S';

Conditions = {<, >, =, >=, <= };

Reactions = visos sistemoje egzistuojančios funkcijos;

3.8 Naudojamos bibliotekos sistemoje:

- Mail.jar tai JavaMail API sudedamoji dalis, kuri skirta formuoti elektorines žinutes. JavaMail API yra sudaryta iš komplekto abstrakčių klasių, kurios modeliuoja įvairias tipiškos pašto sistemos dalis.
- Activation.jar tai JavaMail API dalis.
- Mysql-connector-java-5.1.5-bin.jar – tai Java driver - tvarkyklė, kuri leidžia JDBC (Javos Duomenų bazės Sujungimo galimybe) jungtis į tinklo protokolą, naudojantį MySQL duomenų bazės. JDBC prisijungimas skirtas ryšiui tarp servelto ir MySQL duomenų bazės palaikyti.
- Quartz.jar - tai atviro kodo Scheduling system - darbo planavimo sistema, kuri gali būti naudojama su bet kuria J2EE ar J2SE programa - nuo mažiausiai vienos užduoties iki didžiausios elektroninės komercijos sistemos užduočių atlikimui. Quartz gali būti naudojamas atlikti paprastus ar sudėtingus dešimties, šimto ar dešimtis tūkstančių sąrašus užduočių, kurios apibrėžtos kaip standartiniai Java komponentai ar EJBs.

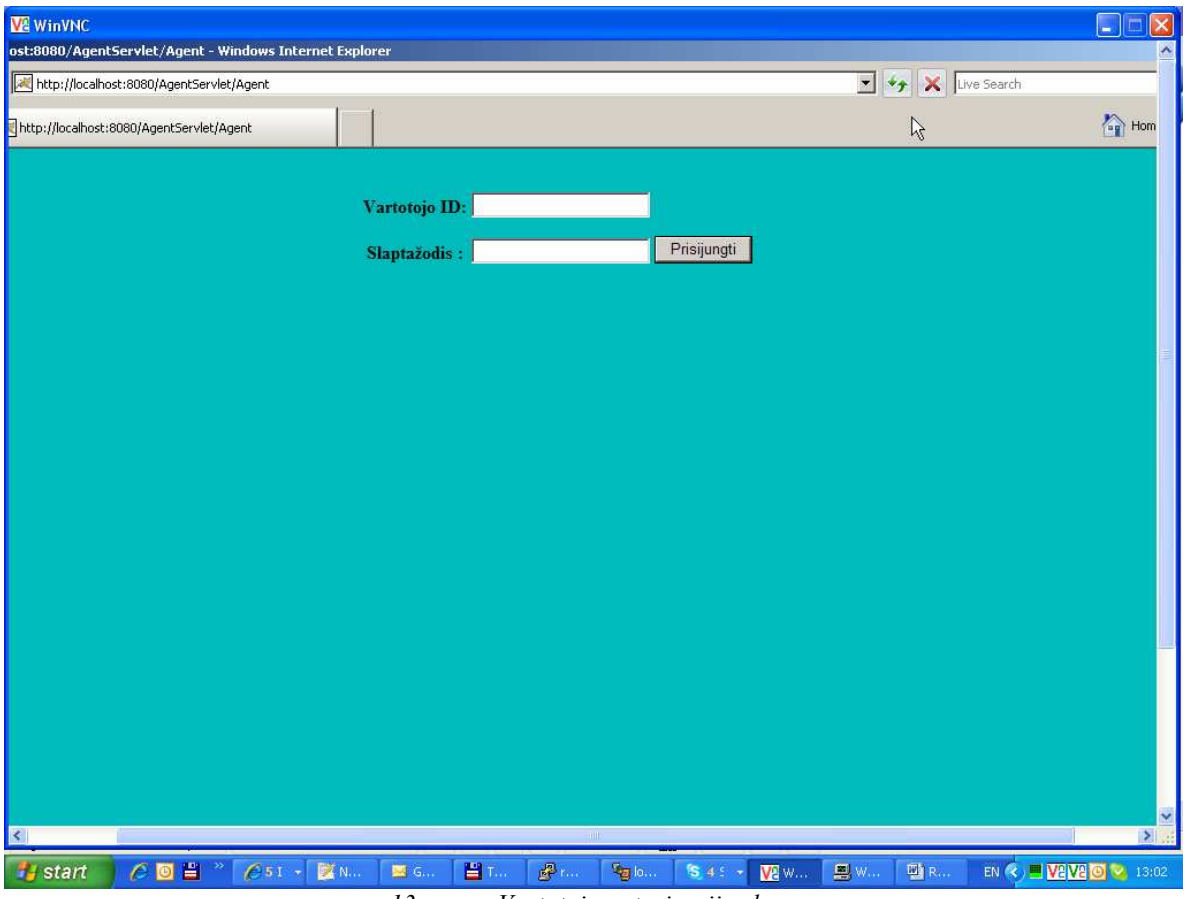
4. EKSPERIMENTINIS TYRIMAS - REALIZACIJA

4.1 Eksperimentinio tyrimo eiga.

Eksperimento tikslas yra patikrinti sukurtą sistemą. Testuojant sistemą naudojame kelias sukurtas funkcijos: Literatūra, Moduliai, Vartotojai, Vartotoju_aktyvumas. Rezultatai vartotojui pateikiami naršyklėje ir siunčiami elektroniniu paštu.

4.2 Eksperimentinio tyrimo rezultatai.

Paleidus agento sistema ir naršyklės pagalba atvėrus programa matome (žr. 13 *pav.*), kad pagrindinis agento puslapis skirtas vartotojui autorizuoti. Prisijungus prie sistemos sistema tikrina ar toks vartotojas jau yra prisijungęs prie sistemos su nurodytu sesijos numeriu. Jei vartotojas dar nebuvo prisijungęs pateikiamas sesijos numeris = “”. Taip sistema supranta, kad vartotojui reikia sukurti naują sesijos numerį, jeigu atlikus vartotojo įvestų duomenų patikrinimą gaunamas patvirtinimas, sistema sukuria naują sesijos numerį ir pažymi sesijos galiojimo laiką penkias minutes. Jeigu per penkias minutes vartotojas neatliks jokių veiksmų, sistema paprašys pakartotinai prisijungti. Po kiekvieno aktyvaus klieno veiksmo, galiojimo laikas visą laiką pratęsiamas penkioms minutėms.



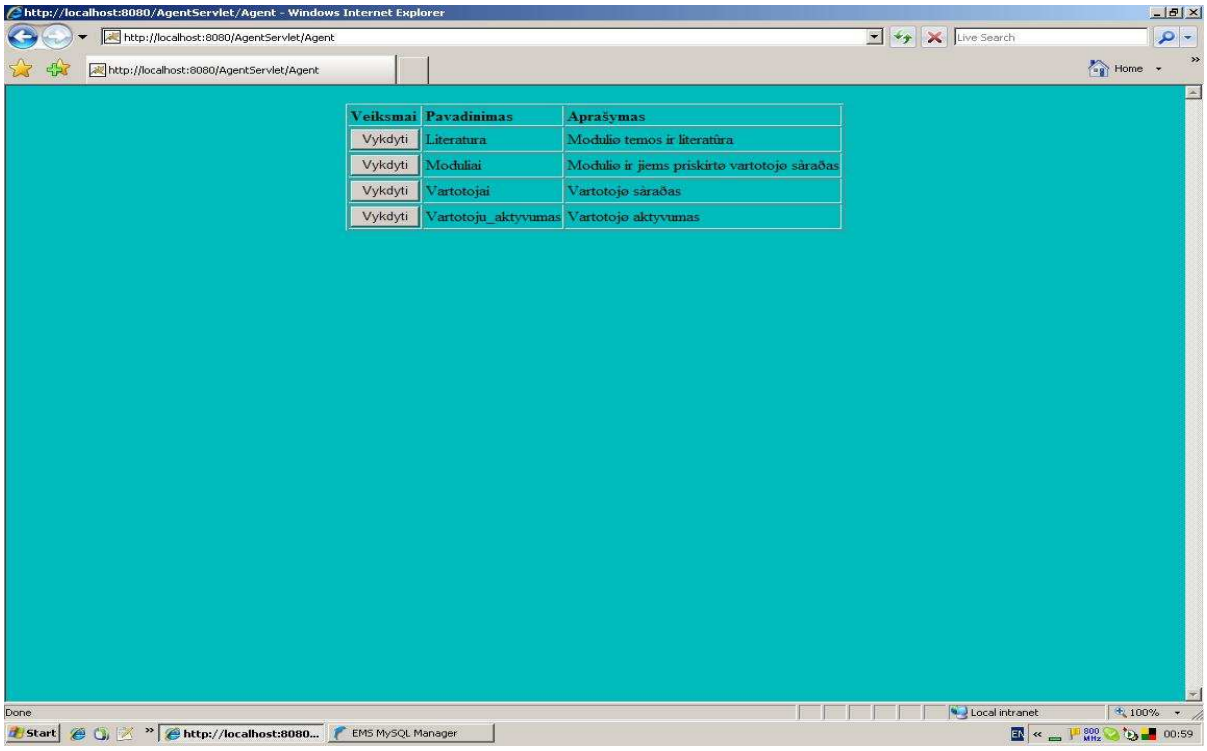
13 pav. - Vartotojo autorizacijos langas.

(žr. 14 *pav.*) Pateiktas pagrindinis funkcijų sąrašo puslapis. Šiame sąraše matosi visos funkcijos, kurios priskirtos prisijungusiam vartotojui. Kitas vartotojas gali turėti savo funkcijų sąrašą. Sąraše matosi mygtukas „Vykdyti“, kurį paspaudus bus įvykdyta atitinkama funkcija, funkcijos pavadinimas, bei trumpas funkcijos aprašymas.

Dabartiniame sąraše yra funkcijos:

- Literatūra – Gražiną sąrašą literatūros pagal modulius ir temas;
- Moduliai – Gražiną sąrašą modulių su aprašais;
- Vartotojai – SMVS sistemoje esančių vartotojų sąrašą;
- Vartotoju_aktyvumas – SMVS vartotojų aktyvumas.

Funkcija Vartotoju_aktyvumas grąžina sąrašą vartotojų iš vartotojai lentelių, bei gražina papildoma informaciją iš lentelės vartotoju_aktyvumas. Ši lentelė buvo sukurta tam, kad kaupti informaciją apie vartotojų prisijungimo ir aktyvumo laikus, bei naujo vartotojo sukūrimo laiką. Visą šią informaciją surenka prie vartotojai lentelės prikabinti trigeriai. Vienas jų veikia „insert“ metu tai yra kai sukurtas naujas vartotojas. Kitas trigeris veikia „update“ metu, tai yra kai vartotojas buvo aktyvus. Ši informacija gali būti naudojamą tam, kad būtų galima kontroliuoti koks vartotojas kokiu laiku buvo sukurtas ir kokiu laiku buvo aktyvus – lankėsi SMVS sistemoje.



14 pav. – funkcijų sąrašas.

(žr. 15 , 16, 17, 18 *pav.*) vaizduojami sukurtų funkcijų rezultatai: Funkcijos atliekamos pagal veiksmų seką:

1. Identifikuojame pasirinktą funkciją;

2. Įvykdomos visos tai funkcijai priskirtos užklausos, kurios nėra “select” tipo. T.y. „update”, „insert”, „create“, „drop“, „alter“ operacijas. Tai funkcijas, kurios atnaujins, įterps, susikurs, papildys, šalins - lenteles, trigerius, stulpelius lentelėse.
3. Šiuo etapu paleidžiamas “Inerpretatorius”, kuris pagal funkcijos šabloną grąžiną funkcijos rezultatą; pvz.: funkcijos Literatūra šablonas:

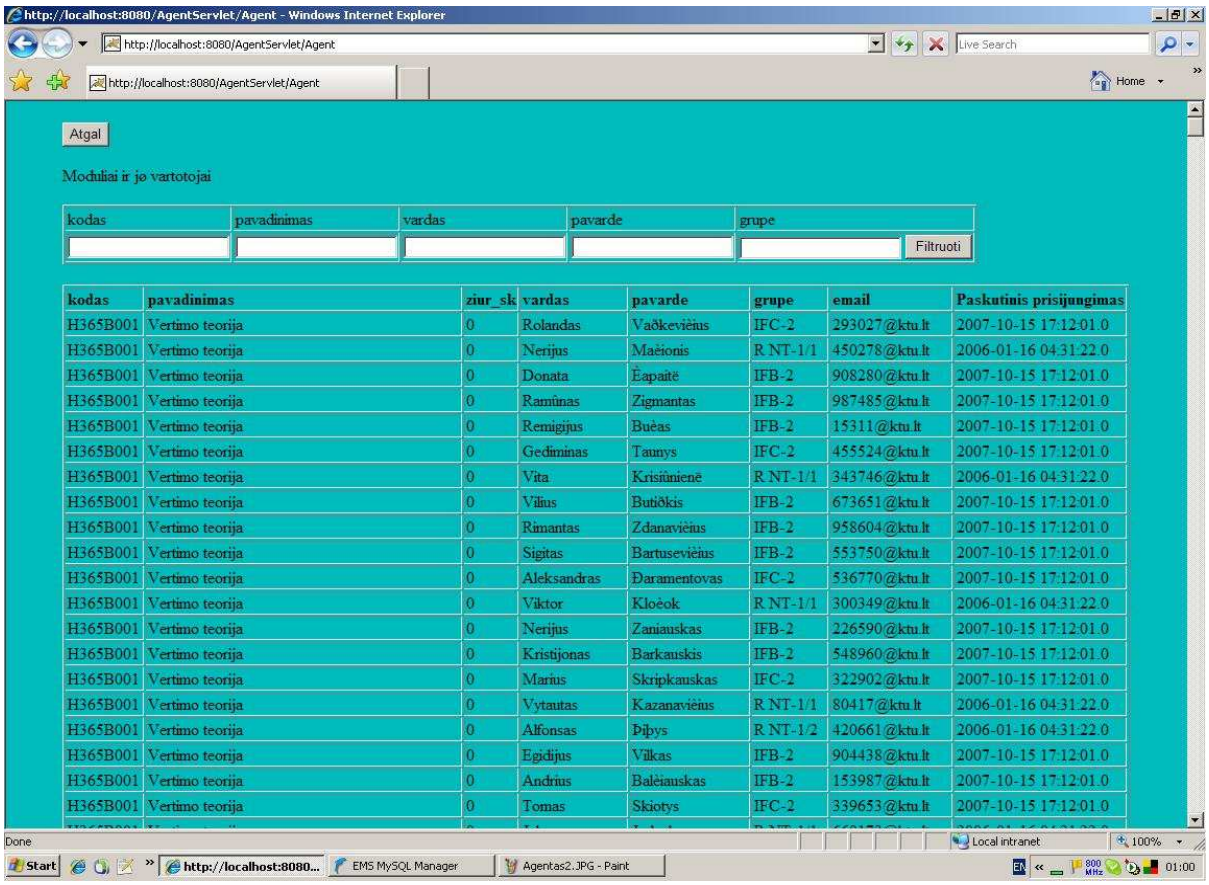
```
<P>Literatûra
        <P>
            <SQL>literatura</SQL>
        </P>
        <P>pabaiga</P>
</P>
```

   “Interpretatoriaus” sistema gavusi toki šabloną pirmiausia patalpina visą šabloną į “Agento” atsakymo (“respons“) žinutę.
4. Patalpinus šabloną “Interpretatoriaus” sistema susiranda visus šablone esančius SQL tagus. Iš tago turinio pasiima užklausos pavadinimą. Pagal ją susiranda visus “filter” tipo parametrus priskirtus tai užklausai. Pagal šiuos parametrus Interpretatorius sugeneruoja filtravimo zoną, kuri skirta rezultato paieškai tarp rezultato.
5. Po filtro seką duomenų aibė kuri gaunamą įvykdžius pačią užklausą, kurios pavadinimas yra tarp <SQL></SQL> tagų. Visas gautas rezultatas talpinamas į <table></table> HTML kodo elementą.
6. Sugeneravus atsakymo žinutę, ji išsiunčiama klientui. Jeigu funkcija vykdoma per naršyklę tiesiogiai, tai funkcijos rezultatas gražinamas į naršyklės ekraną. Jei netiesiogiai, o per “Užduočių vykdytoją”, tai rezultatas gražinamas elektroniniu paštu vartotojui. Abiem atvejais rezultatas gražinamas HTML tekstu. Tam, kad matyti HTML tekstą, elektroninio pašto sistemą turi “suprasti“ – sugeneruoti HTML kodą į vartotojui suprantama tekstą. Tokios elektroninio pašto programos kaip Outlook, GMail šias funkcijas turi ir gali atvaizduoti duomenis vartotojui aiškia suprantama forma.

Literatūros funkcijos gautas rezultatas pateiktas (žr. 15 *pav.*). Funkcija grąžiną sąrašą literatūros pagal modulius ir temas.

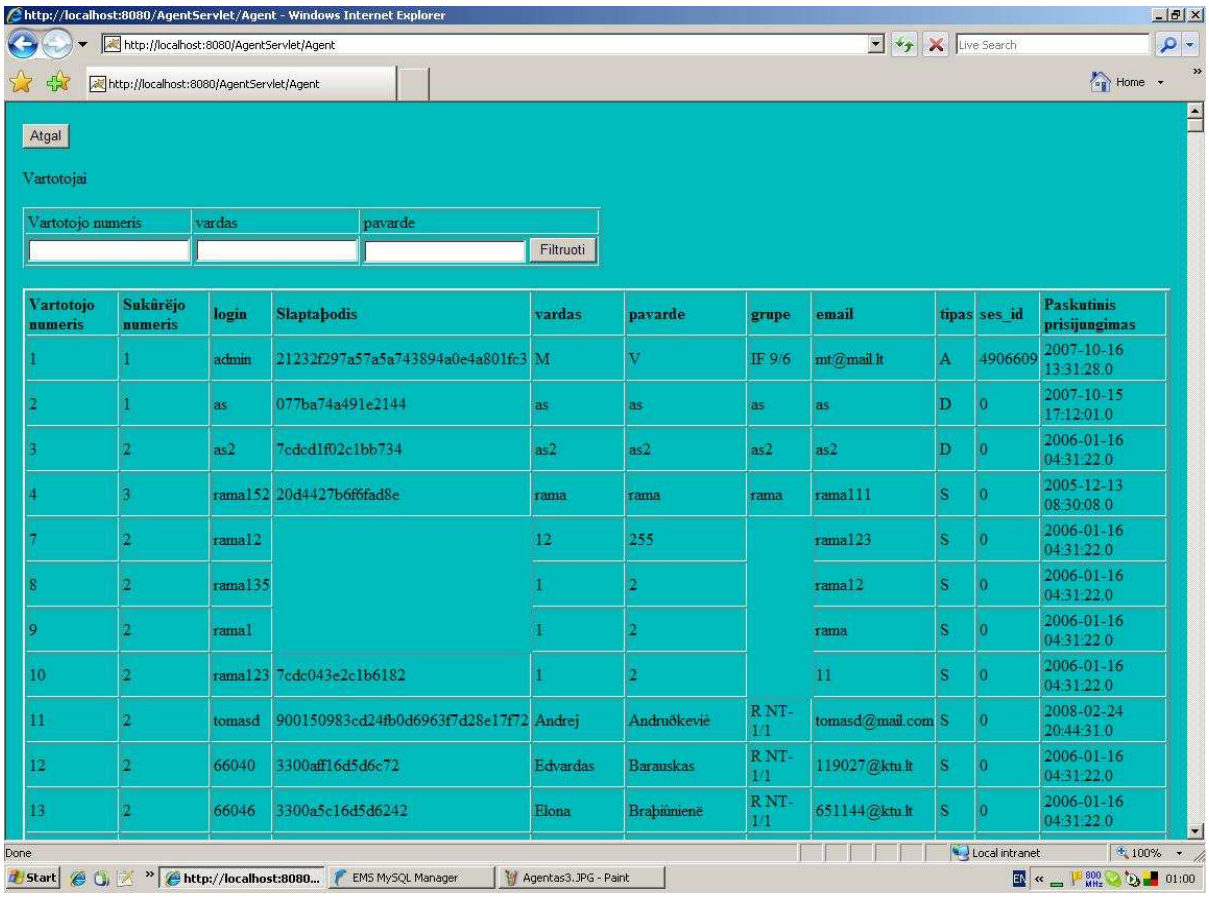
15 pav. – Literatūros funkcijos gautas rezultatas.

Moduliai ir vartotojai funkcijos gautas rezultatas pateiktas (žr. 16 *pav.*). Funkcija grąžina sąrašą modulių su aprašais.



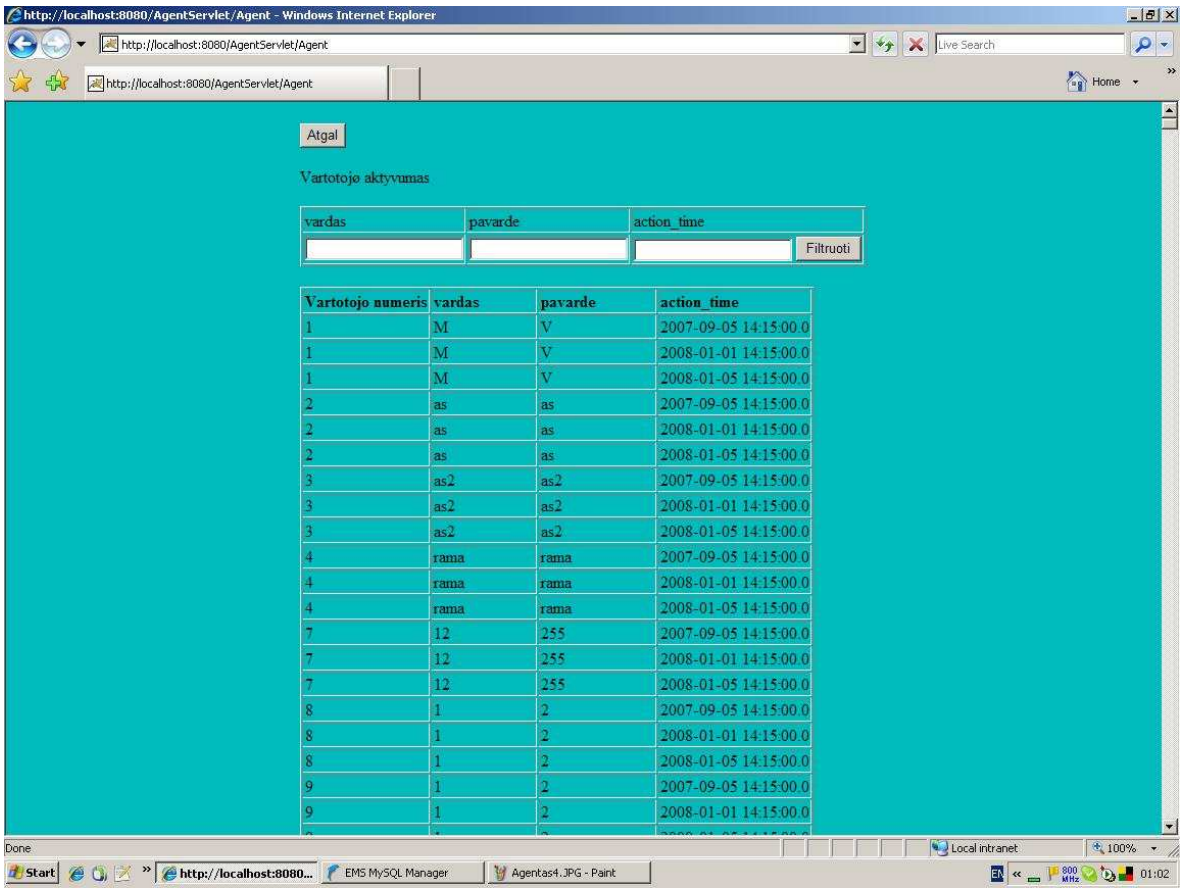
16 pav. – Moduliai ir vartotojai funkcijos gautas rezultatas.

Funkcijos Vartotojai gautas rezultatas pateiktas (žr. 17 *pav.*). Funkcija grąžina SMVS sistemoje esančių vartotojų sąrašą.



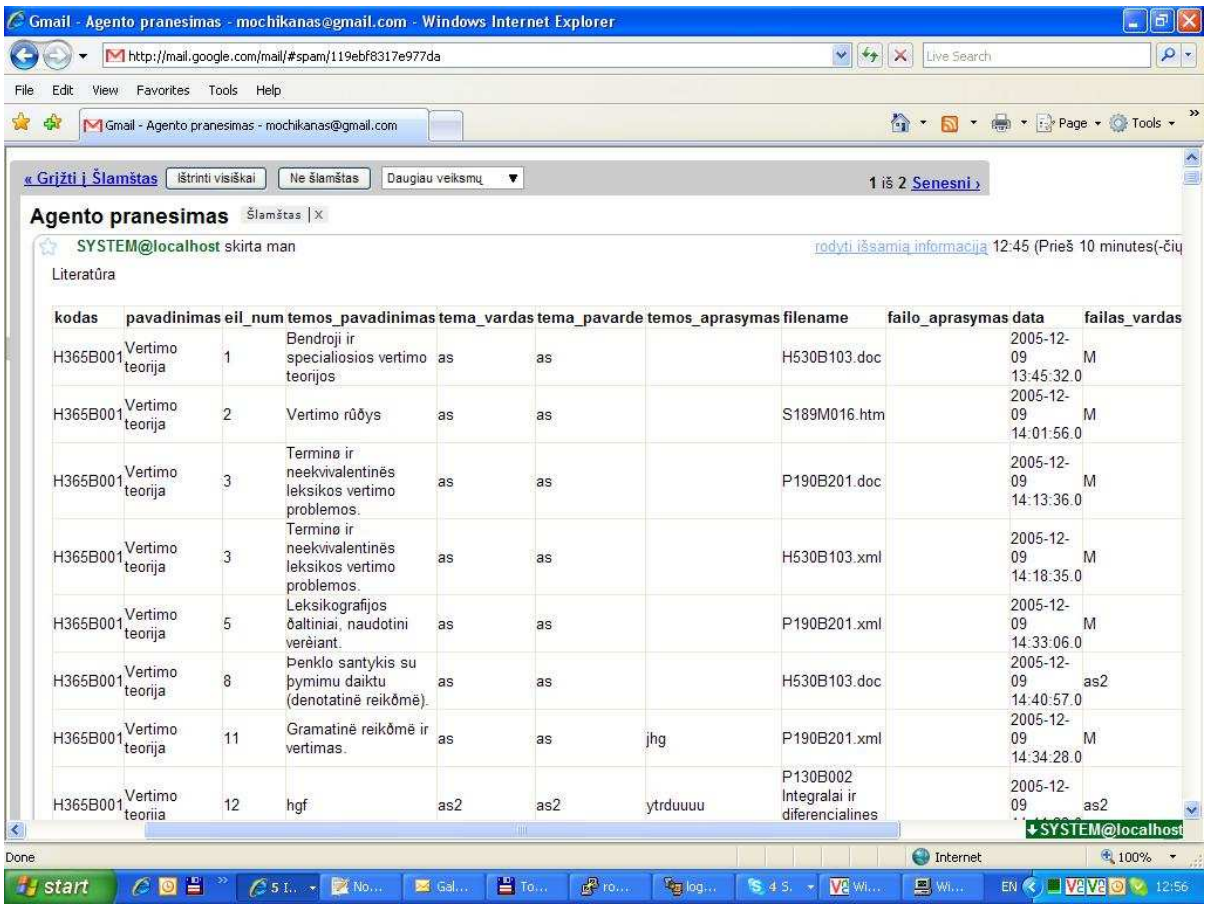
17 pav. – Funkcijos Vartotojai gautas rezultatas

Vartotojų aktyvumo funkcijos gautas rezultatas pateiktas (žr. 18 *pav.*). Funkcija grąžina SMVS vartotojų aktyvumą – kada buvo prisijungę paskutinį kartą.



18 pav. – Vartotojų aktyvumo funkcijos gautas rezultatas.

"Užduočių vykdytojas" – tai "foninė" sistema, kuri savo veiksmus atlieka tam tikrais nustatytais laiko momentais. Skirtingai nuo tiesioginio funkcijos vykdymo "Užduočių vykdytojas" savo gautus rezultatus siunčia elektroniniu paštu. Agento sistemoje buvo nurodytą, kad funkciją "Literatūra" sistema vykdytų nuo "Užduočių vykdytojo" paleidimo pradžios laiko intervalais, kas 20 000 milisekundžių. Taip pat buvo nurodyta, kad gautą rezultatą siųstu elektroniniu paštu. Pašto dėžutėje buvo gautas rezultatas (žr. 19 *pav.*).



19 pav. – Užduočių vykdytojo rezultatai.

4.3 Rekomendacijos sistemos vystymui.

Ateityje būtų galima atlikti šias rekomendacijas sistemos tobulinimui:

- Padaryti sudėtingesnį sąlygų tikrinimo mechanizmą. Naudojant neuroninį tinklą, kurio mazguose būtų galima naudoti dabartinius jutiklius. O reakcijos vykdymas būtų išrenkamas pagal gautą situaciją tinkle. Tokioje sistemoje agentas prieš atlikdamas veiksmą analizuotų didesnę rodiklių aibę kurie ir pasakytų ar norimas veiksmas tinkamas esamoje situacijoje ar ne. Be to agentas galėtų padaryti analize apie tai, kaip veiksmas paveiks dabartinę sistemą.
- Dabartinė sistema nėra pilnai surišta su konkrečia duomenų baze. Ją galima naudoti lygiagrečiai kelių sistemų analizei. Tik ateityje reikia funkcijas rišti su konkrečia sistema. T.y. viena funkcija būtų skirta tik tam tikros sistemos analizei. Iš esmės būtų galima daryti ir bendras funkcijas visoms sistemoms tačiau dėl skirtingų duomenų bazių kūrėjų SQL sintaksių tai padaryti per sudėtinga. Taigi ateityje reiktu rišti funkcijas su duomenų baze, bei padaryti duomenų bazės struktūros analizės WEB aplikaciją. Dabar tam, kad sudaryti bet kokią funkciją reikia daug žinių tiek apie pačią SQL sintaksę tiek apie duomenų struktūra vienoje ar kitoje sistemoje. Taigi įrankis

būtų skirtas tam, kad būtų galima lengvai analizuote sistemoje esančius duomenis bei lengvai sukonstruoti norimas funkcijas.

- Funkcijas susieti ne tik su SQL užklausomis, bet ir fizinėmis programomis ar procesais esančiais serveryje.
- Padaryti universalų grafiką. Dabar visą informaciją pateikiama lentelių pavidalu. Tačiau įvairios statistinės informacijos lengviau analizuojasi grafiko pavidale. Taigi šiai sistemai tiktų WEB aplikaciją kuri gavusi iš klientų dviejų sql užklausų aibes sugeneruotų grafiką x,y koordinačių sistemoje.
- Padaryti, kad be pačių laiškų būtų galimą siųsti laiškus su prisegtais failas. Tai leistų agentui atsiradus naujai literatūrai išsiųsti ją visiems temai priskirtiems vartotojams. Dabartinėje sistemoje galima tik išsiųsti pranešimus apie naują literatūra konkrečia tema.

5. IŠVADOS

1. Atlikus nuotolinio mokymosi valdymo sistemų analizę, išsiaiškinti jų veikimo principai ir architektūra, palygintos teikiamos funkcijos. Išanalizavus įvairius literatūros šaltinius, pastebėta, kad nuotolinio mokymosi sistemos paremtos agentinėmis sistemomis nėra plačiai paplitusios. Galime teigti jog poreikis tokių sistemų paremtų agentinėmis sistemomis tikrai yra ir šioje srityje sparčiai vystomos technologijos.
2. Atliktus agentų tipologijos, projektavimo metodų analizę, nustatyta, kad tinkamiausias mūsų kuriamai sistemai projektuoti reaktyvinis agentas, dėl keliamų reikalavimų ir realizavimo sudėtingumo.
3. Atlikus analizę įsitikinta, kad informacijos analizės agento sistemos įgyvendinimui geriausia tinka realizuoti Java Servlet ir Java JSP technologija. Nes šios technologijos palaiko dinaminių puslapių generavimą.
4. Informacijos analizės agento sistemos architektūra sudaryta reaktyvinio agento pagrindu, kurio veikimas paremtas veiksmų (Sense) ir atoveiksmių (Reaction) pagrindų, kai kiekvienas sistemoje esanti jutiklis yra susietas su reakcija. Pasirinkta, nes agentas turi atlikinėti funkcijas tam tikrais laiko momentais, o ne daryti analizę ar spręsti kurią funkciją atlikti iš aibės.
5. Informacijos analizės agento sistema integruota į SMVS – Studijų Modulių Valdymo Sistemą. Kuri realizuota tradicinės duomenų bazės modeliu, tokios architektūros sistemos leidžia integruoti agentines sistemas į nuotolinio mokymo valdymo portalus.
6. Eksperimento metu sudaryta sistema kurioje įdiegtos ir ištyrinėtos sistemos funkcijos ir elgsena atitikusios funkcionalumą. Šiuo metu informacijos analizės agento sistema yra realizuota WEB aplinkoje.
7. Informacijos agento sistema susistemintą informaciją pateikiama atitinkamais metodais tiesiogiai į naršyklės ekraną HTTP protokolu arba siunčiama e-paštu vartotojui.

6. LITERATŪRA

1. **Y. Shoham**. An Overview of Agent-oriented Programming. In Software Agents, ed. J.M. Bradshaw. Menlo Park, Calif.: AAAI Press, 1997
2. **P. Jaques, A. Andrade, J. Jung and others.** Using Pedagogical Agents to Support Collaborative Distance Learning. http://newmedia.colorado.edu/csc/275.pdf
3. **H. S. Nwana.** Software agents: An Overview. Knowledge Engineering Review, 11 (3). http://www.sce.carleton.ca/netmanage/docs/AgentsOverview/ao.html
4. **Johnson, W.L.,** "Pedagogical Agents", Proceedings of, the Sixth International Conference on Computers in Education (ICCE '98), Beijing, China, October 14 – 17, pp.13 – 22, (1998). http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/8445/26602/01185855.pdf?temp=x
5. **S. Choy, S. Ng, Y. Tsang.** Software Agents to Assist in Distance Learning Environments. http://www.educause.edu/ir/library/pdf/eqm6523.pdf
6. **Maes, P.,** "Software Agents Tutorial", http://www.media.mit.edu/people/pattie/CHI97 (1997), (accessed March 18, 2000).
7. **Shoham, Y.** 1997. An Overview of Agent-oriented Programming. In *Software Agents,* ed J. M. Bradshaw. Menlo Park, Calif.: AAAI Press.
8. **Etzioni, O., and Weld, D. S.** 1995. Intelligent Agents on the Internet: Fact, Fiction, and Forecast. *IEEE Expert* 10(4): 44–49.
9. **Franklin, S., and Graesser, A**. 1996. Is It an Agent or Just a Program? A Taxonomy for Autonomous Agents. In *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*. New York: Springer-Verlag.
10. **Newell, A.** 1982. The Knowledge Level. *Artificial Intelligence* 18:87–127.
11. **Wooldridge, M. J., and Jennings, N. R.** 1995. Agent Theories, Architectures, and Languages: A Survey. In *Intelligent Agents: ECAI-94 Workshop on Agent Theories, Architectures, and Languages,* eds. M. J. Wooldridge and N. R. Jennings, 1–39. Berlin: Springer-Verlag.
12. **Genesereth, M. R., and Ketchpel, S. P.** 1994. Software Agents. *Communications of the ACM* 37(7): 48–53, 147.
13. **Jennings, N., Sycara, K., & Wooldridge, M.** (1998). A roadmap of agent research and development. Journal of Autonomous Agents and Multi-Agent Systems, 1, 275-306.
14. **Wooldridge, M., & Jennings, N.** (1995). Intelligent agents: Theory and practice. Knowledge Engineering Review, 10 (2).

15. **Harrison, C. G.; Chess, D. M.; and Kershenbaum, A**. 1995. Mobile Agents: Are They a Good Idea? IBM T. J. Watson Research Center.
16. **Brodie, M. L**. 1989. Future Intelligent Information Systems: AI and Database Technologies Working Together. In *Readings in Artificial Intelligence and Databases,* eds. J. Mylopoulos and M. L. Brodie, 623–642. San Francisco, Calif.: Morgan Kaufmann.
17. **Jun Peng; Min Wu; Xiaoyong Zhang; Ya Xie; Fu Jiang; Ya Liu.** A collaborative multi-agent based workflow system. 14-17 May 2006 Page(s):341 – 348.
18. **Shirley Lincicum**. Introduction to Interface Agents.
19. **Hentea, M.** Multi-agent security service architecture for mobile learning. 28 June-1 July 2004 Page(s): 91 – 95.
20. **M. Klusch.** Information Agent Technology for the Internet: A Survey. Journal Data & Knowledge Engineering, Elsevier Science, 36(3), 2000.
21. **M. Klusch**. Special issue on Intelligent Information Agents: Theory and Applications, Intelligent Cooperative Information Systems, vol. 10(1&2), March 2001.
22. **Jos´e M Vidal.** Reactive and Hybrid Agents. Department of Computer Science and Engineering University of South Carolina September 2, 2005.
23. **V. S. Subrahmanian, Piero Bonatti, Jürgen Dix, Thomas Eiter, Sarit Kraus, Fatma Ozcan and Robert Ross.** Heterogeneous Agent Systems. Page(s): 21. ISBN:0262194368 (2000).
24. **Chang, D. T., and Lange, D. B.** 1996. Mobile Agents: A New Paradigm for Distributed Object Computing on the WWW. In Proceedings of the OOPSLA 96 Workshop "Toward the Integration of WWW and Distributed Object Technology."
25. **diSessa, A. A**. 1986. Notes on the Future of Programming: Breaking the Utility Barrier. In *User-Centered System Design,* eds. D. A. Norman and S. W. Draper. Hillsdale, N.J.:Lawrence Erlbaum.
26. **Kay, A**. 1990. User Interface: A Personal View. In *The Art of Human-Computer Interface Design,* ed. B. Laurel, 191–208. Reading, Mass.: Addison-Wesley.
27. **Miller, J. R., and Neches, R**. 1987. Tutorial on Intelligent Interfaces Presented at the Sixth National Conference on Artificial Intelligence, 14–16 July, Seattle, Washington.
28. **G. Weiß and S. Sen, eds**., *Adaptation and Learning in Multiagent Systems*. Berlin: Springer Verlag, 1996.
29. **Hayes-Roth, B., Hewett, M., Waashington, R., Hewett, R., Seiver, A.** (1995) Distributing intelligence within an individual. In: L. Gasser, M. N. Huhns (Eds.) *Distributed AI*, Volume II, 385412. Morgan Kaufmann.

7. TERMINŲ IR SANTRUMPŲ ŽODYNAS

3 lentelė. Terminai ir santrumpos

Terminas / Santrumpa	Pilnas pavadinimas	Vertimas / Paaiškinimas
SMVS	Studijų Modulių Valdymo Sistema	Esama Kompiuterių katedros Studijų Modulių Valdymo Sistema
LMS	Learning Management System	Mokymo valdymo sistemos
NMVS	Nuotolinio mokymo valdymo sistemos	
SQL	Structured Query Language	Struktūrizuota užklausų kalba
DB	Database	Duomenų bazė
WEB	World Wide Web	Visuotinis, pasaulinis žiniatinklis
HTML	HyperText Transfer Protocol	Hiperteksto persiuntimo protokolas
TomCat		Tai Java parašytas daugiaplatformis savarankiškas HTTP (Žiniatinklio) Serveris, palaikantis servletus ir JavaServer Page
MySQL		Viena iš reliacinių duomenų bazių valdymo sistemų
James		Elektroninio pašto serveris
Servletas		Sun Microsystems sukurta technologija dinaminių puslapių generavimui.
JSP	JavaServer Pages	Technologija, leidžianti dinamiškai generuoti HTML, XML, ar kito tipo puslapius
JAVA		Objektiškai orientuota programavimo kalba
Sheduler		Darbų planuotojas
Quartz		Darbo planavimo sistema
JDBC	Java Database Connectivity	Prisijungimas skirtas ryšiui tarp Servelto ir MySQL duomenų bazės
J2EE	Java 2 Platform, Enterprise Edition	Standartinė daugialyčių programų kūrimo Java kalba platforma, paremta moduliniais komponentais, vykdomais programų serveryje
J2SE	Java 2 Platform, Standard Edition	Tai Javos širdis, šioje platformoje yra pateikiamos visos bazinės bibliotekos ir įrankiai, kurie naudojami komandinės eilutės ir vizualių programų (Swing karkasas) kūrimui
EJBs	Enterprise JavaBeans	Modulių metodai Interneto komponentų pasiekiami arba lokaliai, arba per kompiuterinį tinklą (naudojant CORBA ar RMI). Jie nebūtinai veikia tame pačiame kompiuteryje, kaip ir Interneto komponentai.
Outlook		Pašto programa
GMail		Nemokama web ir POP3 el. pašto dėžutė.
IS	Information System	Informacinė sistema
Use Case	Unifield Modeling Language	Vieninga modeliavimo kalba

8. PRIEDAI

8.1 Priedas Nr.1 SISTEMŲ ĮDIEGIMO VADOVAS (TomCat ir James serverių diegimas, MYSQL duomenų bazės diegimas, Agentinės sistemos startavimas)

Prisegto kompaktinio disko kataloge „Serverio programine iranga“ yra programinė įranga reikalinga serverio paleidimui:

- apache-tomcat-6.0.16 – HTTP serveris;
- Mysql 5.0.41 - Duomenų bazių valdymo sistema;
- James-binary-2.3.1 – e-pašto serveris.

Instrukcijos paruosiant serverį informacijos analizės agento sistemai:

1. Instaliuojamas Tomcat serveris įvykdant failą apache-tomcat-6.0.16.exe ir sekami tolesni nurodymai.
2. Tomcat serveris paruoštas darbui.
3. Instaliuojama MySQL įvykdant failą setup.exe ir sekant tolesnius nurodymus.
1. Sukuriami MySQL duomenų bazės vartotojai: agent.
2. Įrašomas James e-pašto serveris.

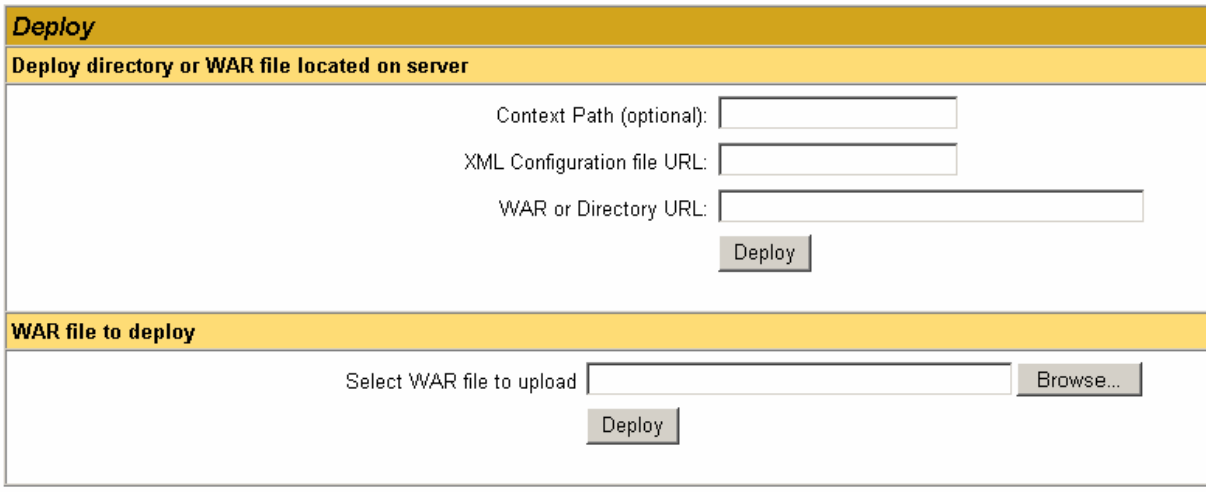
Įdiegus serverio programinę įrangą, paleidžiame informacijos analizės agento sistemą.

Kataloge „Agentas“ sudėtos naudojamos bibliotekos, ir informacijos analizės agento sistema:

- mysql-connector-java-5.1.5-bin.jar - tvarkyklė, kurią naudoja sukurta programinė įranga jungimuisi prie Mysql duomenų bazių.
- Activation.jar - JavaMail API dalis.
- mail.jar - JavaMail API sudedamoji dalis, kuri skirta formuoti elektorines žinutes.
- quartz.jar - darbo planavimo sistema.
- Sukurtos sistemos išeities kodai.
- Sukurta sistema suglaudinta WAR formatu, AgentServlet.war.
- agentodb.sql – sukurtos sistemos duomenų bazės kopija.

Sukurtos agentinės sistemos startavimo vadovas:

1. Sukeliame pateiktas bibliotekas ir AgentServlet.war byla į serverio darbo direktoriją *x:\\..\Tomcat 6.0\webapps*
2. Naršyklėje įvedus adresą, jei serveris lokalus: http://localhost:8080/manager/html. Prisijungiame prie Tomcat serverio valdymo (žr. 20 *pav*.) ir įkeliame(*ang. Deploy*) AgentServlet.war failiuką.



20 pav. – Tomcat valdymo sistema.

3. Prisijungiame prie MySQL duomenų bazės ir importuojame informacijos analizės agento sistemos duomenų bazę iš failo agentodb.sql
4. Informacijos analizės agento sistema paruošta darbui, naršyklės lange įvedame adresą http://localhost:8080/AgentServlet/Agent jei naudojamas lokalus serveris. Rezultate gauname sistemos prisijungimo langą (žr. 13 *pav.*).

8.2 Priedas Nr.2 Kompaktinis diskas (CD)