KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

TVIRTINU
Katedros vedėjas
prof. E. Kazanavičius
2005 m. gegužės mėn. ....d.

Mobiliosios informavimo sistemos tyrimas

Informatikos magistro baigiamasis darbas

Lietuvių kalbos konsultantė
(parašas)
dr. J. Mikelionienė
2005 m. gegužės mėn. ..... d.

Vadovas
(parašas)
prof. E. Kazanavičius
2005 m. gegužės mėn. ..... d.

Recenzentas
(parašas)
doc. dr. S. Maciulevičius
2005 m. gegužės mėn. ..... d.

Atliko
(parašas)
IFM 9/1 gr. studentas
Darius Kaškelevičius
2005 m. gegužės mėn. ..... d.

KAUNAS, 2005

Turinys

Paveikslų turinys

1. Įvadas

Šiomis dienomis, kai mokslo ir technikos laimėjimai vis sparčiau žengia į priekį, turime neatsilikti ir spėti kartu. Vis didėjantys projektai ir trumpėjantis atlikimo laikas, tam kad sukurti rinkoje konkurencingą produktą, diktuoja sąlygas, kad produktas būtų kuo greičiau sukurtas, išbandytas ir išleistas į rinką. Šiuolaikinėse kompiuterizuotose sistemose dažniausiai susiduriame su veiksmais realiame laike, kai duomenų apdorojimas turi vykti nenutrūkstamu srautu ir pakankamai greitai. Tokie procesai vyksta vaizdo ir garso apdorojimo informacijos apdorojimo sistemose, komunikacinėse priemonėse, kalbos atpažinimo sistemose, kuriose veiksmai vyksta realiu laiku ypač svarbu iš anksto žinoti ir numatyti, kurioje vietoje sistema gali būti pažeidžiama, galimus trikdžius. Tai leidžia sutrumpinti projekto laiką, nes nereiks papildomai gaišti taisant iš anksto nenumatytų spragų. Tai galima pasiekti turint sistemos modelį

Šio magistrinio darbo tyrimo objektas – internetinė mobili informavimo sistema. Jos paskirtis – studentams pateikti informaciją apie dėstytojus bei jų tvarkaraščius, o dėstytojams – koreguoti savo dienotvarkes, informuoti apie pasikeitimus norimus asmenis. Vartotojas su sistema dirba interneto naršyklės pagalba.

Šiuo metu Kauno Technologijos Universitete nėra nei vienos realiai veikiančios mobiliosios informavimo sistemos, bei tokios, per kurią studentai ar dėstytojai galėtų sužinoti dėstytojų tvarkaraščius, jų laisvą laiką, nusiųsti pranešimą elektroniniu paštu ar trumpąja SMS žinute. Informacija apie dėstytojų tvarkaraščius, jų elektroniniai pašto adresai, jų fakultetai, katedros ir kabinetai yra žinomi ir prieinami visiems, tačiau ši informacija nėra susisteminta, sudėta į vieną vietą ir patogiai bei greitai pasiekiama. Ši sistema, vartotojui suteikia galimybę pasiekti visą reikalingą informaciją iš vienos vietos. Dėstytojai gali prisijungti prie sistemos ir koreguoti savo tvarkaraštį, dienotvarkę, paskirti susitikimus su studentais ir pan. Be to, jie gali siųsti pranešimus elektroniniu paštu ir/arba trumpąja SMS žinute suinteresuotiems dėstytojams, bei studentams.

Magistrinio darbo tikslas – išanalizuoti egzistuojančios mobiliosios informavimo sistemos procesų modelius ir jų struktūrą, atlikti transformaciją iš UML procesų į Petri tinklų procesus, atlikti jų savybių, bei efektyvumo tyrimą. Pasiūlyti lanksčiausią sistemos valdymo algoritmą, atsižvelgiant į sistemai keliamus reikalavimus.

Šiam tikslui pasiekti reikia atlikti tokius uždavinius:

- atlikti mobiliosios informavimo sistemos procesų analizę;
- atlikti modeliavimo metodų analizę ir pasirinkti tinkamiausią modelio sudarymo metodą;
- sudaryti sistemos procesų modelį;
- patikrinti šio modelio patikimumą ir stabilumą.

Darbo analizės dalyje atlikta formalių metodų, bei mobiliosios informavimo sistemos procesų analizė. Teorinėje dalyje, tiksliai suformulavus uždavinį, pasiūlytas sistemos procesų modelio veikimo algoritmas, bei sudarytas mobiliosios informavimo sistemos veikimo modelis. Eksperimentinėje dalyje patikrintas šio modelio veikimas.

2. Valdymo procesų, bei formalių metodų analizė

2.1 Tyrimo sritis, objektas, problema

Pagrindinė tyrimo sritis – mobiliosios informavimo sistemos valdymo algoritmo parinkimas pagal sistemai keliamus reikalavimus, panaudojant UML ir Petri tinklus. Visą šiame darbe nagrinėjamą tyrimo sritį galima išskirti į kelias atskiras sritis (1 pav.).

Sritis	**Srities aprašymas**
Mobili informavimo sistema ir joje vykstantys procesai	• Mobiliosios informavimo sistemos samprata • Mobiliosios informavimo sistemos procesų charakteristikos
Mobiliosios informavimo sistemos procesų modelio sudarymas	• Vartotojui suprantamo modelio sudarymas ir privedimas prie formalaus Petri tinklų modelio
Mobiliosios informavimo sistemos procesų modelio tyrimas	• Mobiliosios informavimo sistemos modelis • Sistemos modelio veikimas

1 pav. Nagrinėjimų tyrimų sritys

Visos paminėtos tyrinėjimo sritys yra tarpusavyje susijusios, jas reikia analizuoti ir įvertinti. Mobiliosios informavimo sistemos procesus reikia išnagrinėti ir įvertinti tam, kad sudaryti teisinga matematinį modelį ir parinkt tinkamą valdymo algoritmą. Kad sistema veiktų stabiliai, reikia kad jau projektavimo fazėje būtų pastbėtos ir išanalizuotos galimos spragos. Jei sistema veiks ne pagal pasirinktą algoritmą, ji paprasčiausiai nebus tinkama naudojimui, nes esant dideliam sistemos apkrautumui padidėja tikimybė prarasti duomenis ar tiesiog išvesti pačia sistemą iš rikiuotės.

Tiriamasis objektas – mobiliosios informavimo sistemos procesų tyrimas, UML modelio sudarymas privedimas prie Petri tinklų modelio.

Jokių tyrimų apie mobiliają informavimo sistemą nebuvo atlikta, taip pat nėra pristatyta jokių mokslinių straipsnių apie jos procesų analizę ir modelio sudarymą. Šio darbo tikslas – ištirti ir sudaryti mobiliosiosios informavimo sistemos UML modelius, privesti juos iki Petri tinklo modelių, atlikti jų tyrimą.

Tiriant ir sudarant mobiliosios informavimo sistemos veiklos procesų modelį, iškyla kelios esminės problemos, kurios yra pateikiamos 2 paveiksle.

Problema	**Problemos aprašymas**
Sistemos patikimumas	- Sistemos priežiūra turi būti paprasta
Sistemos stabilumas	- Kaip užtikrinti sistemos stabilumą - Kokias priemones naudoti sistemos stabilumui užtikrinti
Naudojimo paprastumas ir išplečiamumas	- Sistema turi būti lengvai plečiama ir plėtimo procesas nereikalaujantis daug resursų

2 pav. Nagrinėjamos problemos

Mobili informavimo sistema yra skirta informuoti suinteresuotus asmenis, leisti tvarkyti dienotvarkę, talpinti skelbimus, todėl sistemmos patikimumas yra vienas iš svarbiausių sistemos veiksnių, kurie įtakoja jos populiarumą. Vartotojos nesinaudos sistema, jei ji bus nepatikima, nestabili ir blogai veikianti.

Vienas iš svarbiausių uždavinių yra sistemos saugumas, todėl sistema turi būti apsaugota nuo pašalinių faktorių, kurie galėtų pažeisti sistemos veikimą. Sistemos saugumui turi būti parinktos tinkamos saugumo technologijos ir priemonės.

Mobilioji informavimo sistema turi būti turi būti lengvai papildoma naujais komponentais. Šis išplečiamumas turi būti lengvai vykdomas ir nereikalaujantis didelių išteklių. Iš sistemos turėtų taip pat lengvai būti pašalinami nereikalingi komponentai, kurie nereikalingais tampa dėl naujų komponentų atsiradimo arba jie morališkai pasensta..

Sistemos vartotojai – dažniausiai paprasti žmonės. Tik nedidelė jų dalis turi gilias žinias ir patirtį kompiuterijos srityje, todėl jiems reikalinga nesudėtinga ir jų galimybes atitinkanti sistema.

2.2 Naudojami metodai ir priemonės sistemoms sudaryti ir tirti

Standartu, objektinio modeliavimo srityje, pripažinta UML (*Unified Modeling Language*). Šios kalbos notacija leidžia grafiškai atvaizduoti sistemos reikalavimus, struktūrą ir elgseną, nepriklausomai nuo sistemos realizavimo specifiškumų. Be to, UML pirmiausiai vis tik yra vaizdinio specifikavimo kalba, daugeliui specifikacijų naudojanti diagramas. Tai labai svarbu aprašant sudėtingus sąryšius, kurie retai turi paprastą nuoseklią struktūrą ir yra sunkiai aprašomi paprastu tekstu.

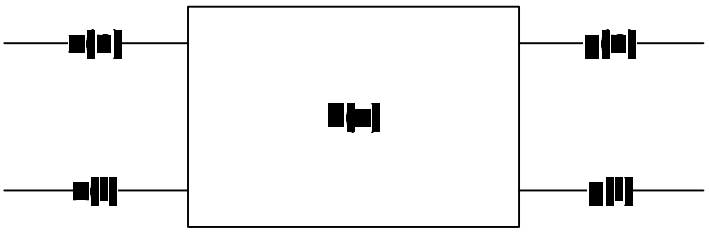
Dažnai lygiagrečiais tampantys valdymo srautai, skirtingos rolės ir jų įgaliojimai, resursų būsenų pokyčiai itin svarbūs bandant suprasti sistemos elgseną, ir juos būtina išreikšti vizualiai. Be to, dažnai būtų naudinga atskirti veiksmo apdorojamas esybes nuo esybių, atliekančių tą veiksmą. Deja, UML konstruktai, skirti elgsenai modeliuoti, nėra taip gerai išvystyti kaip konstruktai, modeliuojantys statinę sistemos struktūrą.

Sistemai modeluoti struktūriniu požiūriu objektiniame modeliavime buvo pritaikytas vienintelis – esybių-ryšių modelis, kuris įvairiose metodologijose skiriasi nebent grafiniais žymėjimais. Tuo tarpu elgsenai modeliuoti yra taikomi keli iš esmės skirtingi modeliai. Bendradarbiavimo modeliavimui, kuris orientuotas į roles ir jų sąveikas, reikalingas vykdant užduotis, skirtos bendradarbiavimo (*collaboration*) ir sekų (*sequence*) diagramos; veiklos (*activity*) ir būsenų (*statechart*) diagramos vaizduoja sistemos reakcijas į veiksnius.

Dar vienas trūkumas – UML nepateikia vientiso elgsenos modelio. Statinis modelis (klasių diagrama), puikiai vaizduoja sistemos klases ir tinka sudaryti bendram sistemos vaizdui. Dinaminis (veiklos diagramos) parodo kaip veikia sistema, tačiau neįmanoma patikrinti jo veikimo teisingumo, nes UML nėra pritaikytas formaliam specifikavimui. Šiai kalbai trūksta integracijos tarp elgsenos modelio ir struktūrinių komponentų, tokių kaip posistemės, klasės, interfeisai ir sąryšiai. Ideali objektinė kalba turėtų pasižymėti tikslia semantika, padengiančia ir struktūrinius sistemos aspektus (pvz., paskirstymą), ir elgsenos aspektus (pvz., lygiagretumą, gebėjimą reaguoti ir atsparumą lūžiams). Jeigu kalbėtume apie laiko specifikavima, tai kol kas UML kalboje tai nėra iki galo išbaigta, tarkime naudodami sekų diagramas galime įvesti laiko apribojimus, tačiau yra sunku sudėtingą sistemą išskaidyt į atskirus lygius ir juos apjungti, diegrama tampa paini ir sukiai suprantama Trumpai tariant turėdami vien tik UML modelį negalime tiksliai prognozuoti sistemos elgsenos.

2.3 Sistemos modeliavimo metodų apžvalga

Mobilioje informavimo sistemoje dažnai vyksta procesai, kuriuos reikia apdoroti realiame laike, taigi mobilią informavimo sistemą galime laikyti realaus laiko sistema. Pačią sistemą laikome kaip tam tikrą funkciją



3 pav. Sistemos bendra funkcinė schema

Tokioje sistemoje padavus įėjime duomenis ar jų rinkinų, išėjime gauname atitinkamą rezultatą.

Siekiant sudaryti sistemos modelį reikia atlikti realaus laiko sistemų sudarymo bei tyrimo analizę. Šio tipo sistemoms tirti yra naudojami įvairūs metodai bei priemonės. Sistemas galima aprašyti naudojant įvairių tipų programavimų kalbas.

- įprastos programavimo kalbos (C++, C#, Delphi). Naudojantis šiomis kalbomis, sukurta daugelis sistemų modeliavimo įrankių;
- modeliavimo kalbos (UML). Naudojant įprastas programavimo kalbas bei atliekant sistemos modeliavimą ir tyrimą kiekvieną kartą tenka iš naujo kurti bazinius algoritmus. Naudojant specialiai tam skirtas modeliavimo kalbas, nereikia rūpintis baziniu tyrimo priemonių kūrimu. Tai ne tik patogu, bet ir padeda išvengti dalies klaidų kuriant sistemos aprašus.

Projektuojant sistemas ir aprašant jas programavimo kalba yra naudojamis įvairūs sistemų modeliai. Sistemų formalius modelius galima aprašyti naudojant Petri, Kano tinklų bei baigtinių būsenų automatais.

Egzistuoja trys tipinės sistemos modelių savybės, kurios yra svarbios sistemos projektavimui:

1. laiko modeliavimas ir sinchroniškumas;
2. modelio vaizdavimas (paremtas veikla, būsenomis arba laiku);
3. pagrindinės pritaikymo sritys.

2.3.1 Laiko modeliavimas ir sinchroniškumas

Kai modeliuojant yra susiduriama su laiku, labai naudinga yra panaudoti sinchroninius veiksmus. Šiuo atveju sistemos išėjimai yra sinchronizuoti su sistemos įėjimais, o sistemos reakcijos laikas yra nulinis. Tokiu būdu laikas yra tarsi nustumiamas į šalį. Sinchroniškumo hipotezė yra pagrindas matematiniam formalizmui.

Tokiose sinchroninėse kalbose, kiekvienas signalas yra susietas su laiko signalu. Laiko signalas yra susijęs su kitais laiko signalais ir taip yra apibrėžiamas sistemos įvykių seka. Kai lyginami du signalai, susiję laiko signalai parodo, kurie veiksmai yra vienalaikiai ir kurie seka vienas po kito.

2.3.2 Vaizdavimas

Paprastai naudojama Gajskio sistematika išskiria penkis specifikacijų modelius:

- paremtus būsenomis;
- paremtus veikla;
- paremtus struktūra;
- paremtus duomenimis;
- heterogeninius.

Šios kategorijos atspindi skirtingus požiūrius į sistemas, jų funkcionalumą, struktūrą ir duomenis jose.

2.3.3 Pagrindinės pritaikymo sritys

Pradinis modeliavimo įrankis buvo pasirinktas UML atsižvelgus i magistrinio darbo vadovo pasiulymus bei remiantis jo straipsniu[1]. UML yra galingas modelavimo įrankis, tačiau jinai nėra naudojama formalizavimui, formalių metodų sudarymui, ji yra tik modeliavimo kalba, bet ne metodas. UML pasirinktas dar ir todėl, kad turi puikiai išvystyta grafinę sąsają, sudaromi modeliai yra lengvai suprantami ir paaiškinami.

Atlikta daugelis tyrimų, bandant skirtingomis kalbomis parašytas programas. Yra sukurta modelių, kurie yra skirti duomenų srautams apdoroti, taip pat yra tokių, kurie labiau tinka valdymo sistemoms. Vis dėlto, šiam požiūriui stinga bendrumo, kadangi, daugelis sistemų nėra pritaikomos abejoms kategorijoms. Turėtume atkreipti dėmesį į tai, kad skirtumai tarp valdymo ir duomenų srautų sistemų modelių yra svarbūs tiek valdymo, tiek duomenų srautų sistemoms.

3 paveiksle pateikiami pagrindinės baigtinių automatų, Kano ir Petri tinklų charakteristikos[7].

Sistemos modelis	Laiko modeliavimas	Sinchroniškumas	Vaizdavimas	Pagrindinės pritaikymo sritys
Baigtinių automatų	Įvykiai su laiko požymiu	Taip	Būsena	Posistemių valdymas
Kano tinklų	Nėra tikslios laikinės sekos	Ne	Veikla	Skaitmeninis signalų apdorojimas, tinklai, protokolai
Petri tinklų	Nėra tikslios laikinės sekos, tik perėjimų seka	Ne	Būsena arba veikla	Planavimas, valdymas, tinklo protokolai

4 pav. Sistemos modelių charakteristikos

Pagrindiniai Petri tinklų privalumai lyginant su kitais metodais yra:

1. paprastas ir lengvai suprantamas modelio vaizdavimas;
2. formali ir efektyvi semantika (tinkama sudėtingai analizei, gali būti kaip programinio kodo pagrindas);
3. modelis sudarytas, naudojant Petri tinklus, yra lengvai išplečiamas;
4. lengvai analizuojami;
5. atvaizduoja lygiagrečius procesus.

Apibendrinus aukščiau pateiktą privalumus, galima daryti išvadą, kad Petri tinklai yra tinkamiausi mobiliosios informavimo sistemos modeliui sudaryti, kadangi, jie atitinka visus kuriamo modelio reikalavimus. Taigi atsižvelgus į Petri tinklų privalumus prieš kitus modeliavimo būdus, būtent Petri tinklai ir buvo pasirinkti modelio sudarymui.

2.4 Unifikuota modeliavimo kalba

2.4.1 UML samprata ir istorija

Objektiškai orientuotos modeliavimo kalbos, kaip metodologijos atsirado tarp 1970 ir 1980 metų kartu su naujais objektiškai orientuotų programavimo kalbų stiliais. Tačiau augant panaudojimo atvejų (aplikacijų) skaičiui, siekiant analizuoti ir projektuoti alternatyvius metodus, atsirado objektiškai orientuotų modeliavimo kalbų poreikis. UML pagrindus suformulavo Rational Software Corporation ir trys žymiausi informacinių sistemų ir technologijų industrijos metodologai: Grady Booch, James Rumbaugh ir Ivar Jacobson[2]

Oficialiai UML bandymai prasidėjo 1994 metų spalio mėnesį, kada J.Rumbaugh ir G.Booch, UML pradininkai suvienijo pastangas. Projekto iniciatorius buvo G.Booch

UML – universali modeliavimo kalba, apibrėžianti standartinę notaciją objektiškai orientuotai sistemai. Ji yra skirta programinei įrangai specifikuoti, konstruoti, vizualizuoti ir dokumentuoti. UML atveria standartinius kelius aprašyti konceptualius dalykus, biznio procesus ir sistemos funkcijas, kaip konkrečius dalykus - kaip klases[2].

Universalią modeliavimo kalbą UML sudaro tokie modeliai ir juos aprašančios diagramos:

Statinės struktūros diagramos *(static structure diagrams)*:

- klasių diagrama *(class diagram)*;
- objektų diagrama *(object diagram);*

Panaudojimo atvejų diagramos *(use case diagrams)*

Sąveikos diagramos *(interaction diagrams):*

- sekos diagramos *(sequence diagrams);*
- bendradarbiavimo diagramos *(collaboration diagrams).*

Būsenų diagramos *(statechart diagrams)*

Veiklos diagrama *(activity diagram)*

Diegimo diagramos *(implementation diagrams)*

- komponenčių diagrama *(component diagram);*
- įrangų diagrama *(deployment diagram).*

UML nenurodo standartinio informacinės sistemos kūrimo proceso ar metodo, tai yra tik objektinė grafinio modeliavimo kalba – notacija ir jos sintaksės bei semantikos aprašymas. Yra žinoma keletas populiarių objektinio modeliavimo metodologijų, kurios aprašo konkrečius objektinio modeliavimo metodus, naudojančius UML ir papildomus modelius[2].

Šiuo metu labiau žinomos tokios OO metodologijos [2]:

Catalysis - ..;

Objectory - ..;

Shaler/Mellor …;

Fusion …;

OMT …;

Booch…;

Daugelis firmų sukuria IS kūrimo metodologijas savo programinių produktų kūrimui. Catalyst metodologija, firma *Computer Sciences Corporation* (CSC). WSDDM – *Worldwde Solution Design and Delivery Method*, firma IBM. Šios metodologijos sujsungia į nuoseklią technologiją UML modelius (visus ar dalį) ir papildomus modelius, pavyzdžiui, CRC korteles, Workflow ir kitus.

2.4.2 Klasių diagramos

Klasių diagramos CD (ankščiau buvo vadinamas objektų modeliu) aprašo objektų klases ir jų tarpusavio ryšius, apimdamos paveldėjimą, agregavimą, asociacijas.

CD yra OO modeliavimo pagrindas, jos nurodo, aprašo *analizės rezultatus*: kam informacijos sistema yra skirta, ką IS gali vykdyti (atlikti), ir *projektavimo rezultatus*: iš ko informacijos sistema bus sudaryta (iš kokių dalių suprojektuota)[2].

Informacija, esanti klasių diagramose, tiesiogiai projektuojasi į programos išeities tekstą, kuris realizuoja taikomąją IS, todėl klasių diagramas CD yra būtina sudaryti.

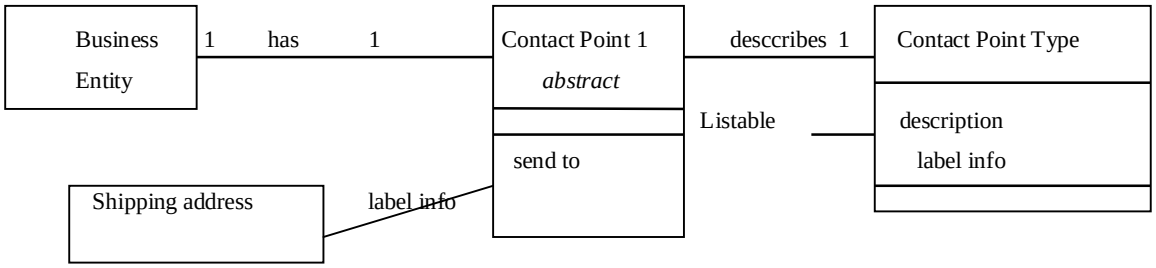
Klasės yra aprašomos nurodant objektu (patenkančių i aprašoma klase) atributus (parametrus), atliekamus veiksmus (metodus) ir ryšius (asociacijas) su kitomis klasėmis.

- atributai aprašomi nurodant jų paskirti, tipą ir galimas jų reikšmes;
- metodai yra dokumentuojami atskirai, parašant jų logika;
- ryšiai tarp klasių yra dokumentuojami, nurodant ryšio paskirti, kardinalumą ir ryšio sudarymo būtinumą

Klasių diagrama gali būti sudaroma skirtingiems IS lygmenims aprašyti:

- organizacijos veiklos (biznio) lygmeniui – nurodo klases, kurios dalyvauja vartotojo sąsajoje *(user interface);*
- panaudojimo atvejų *(application)* lygmeniui – nurodo klases, kurios realizuoja taikomosios programos logiką;

- duomenų *(data)* lygmeniui – nurodo saugomų duomenų klases.



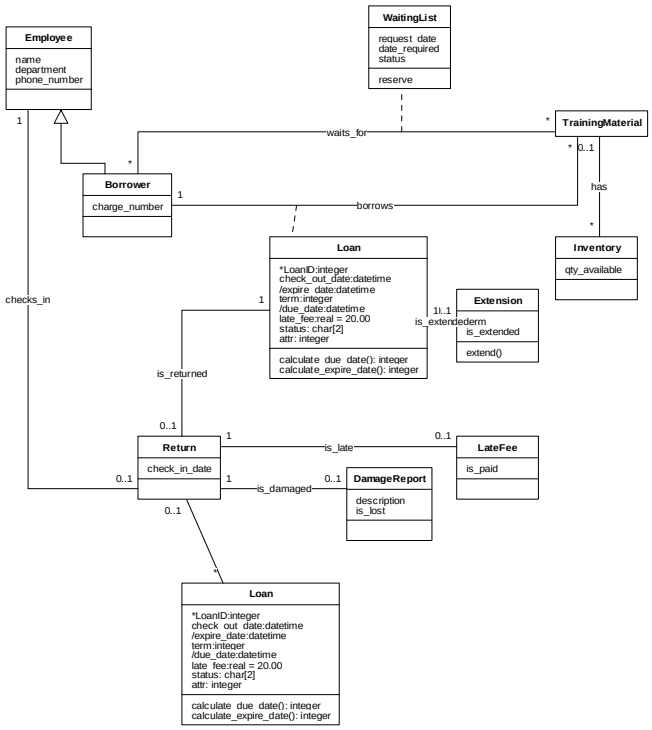
5 pav. Klasių diagramos pavyzdys

Klasių diagramos siejasi su kitomis UML diagramomis:

- būsenų diagramomis *(state d.)*, kurios yra sudaromos aprašyti sudėtingų klasių gyvavimo ciklą;
- veiklos diagramomis *(activity d.)*, kurios gali būti naudojamos specifikuoti klasės metodų vykdymo žingsnius;
- komponenčių diagramomis *(component d.)*, kurios sugrupuoja klases į komponentes arba modulius;
- įrangų diagrama *(deployment d.)*, kuri specifikuoja techninės ir programinės įrangų tarpusavio sąryšį, išdėstymą (IS realizavimo specifikacija).

2.4.3 Objektų diagrama OD (Object Diagram)

Objektų diagrama yra klasės elementų (instances) grafas, kuris vaizduoja objektus ir duomenų reikšmes. Statinė modelis yra klasių diagramos konkretus atvejis.



6 pav. Objektų diagrama

Objektų diagramos retai vartojamos, reikalingos aprašant konkrečiu duomenų struktūrų pavyzdžius.

2.4.4 Panaudojimo atvejų diagramos UCD (Use Case Diagrams)

Panaudojimo atvejų diagramos (use case diagrams - UCD) yra įeigos taškas reikalavimų IS analizei atlikti,pagrinde naudojama sistemos funkcionalumo analizei ir aprašymui, bei funkciniams reikalavimams aprašyti.. Pavyzdžiui, Popkin Software and Systems firma IS kuria pagal “use case-driven approach”, t.y. panaudojimo atvejų *(use case)* diagramos yra kertinis modelis IS kurti..

UCD susideda iš dalyvių, procesų (panaudojimo atvejų) ir įvedamų bei išvedamų duomenų. Pradinė informacija UCD sudaryti yra problemos aprašymas tekstu, vadinamas panaudojimo atvejų scenarijumi[2].

Taikomasis uždavinys (Use Case) - nurodo kaip realaus pasaulio objektas (asmuo, organizacija ar kita informacijos sistema), vadinamas *dalyviu ("an actor"),* sąveikauja su projektuojama sistema – kokius duomenis įveda ir gauna. Taikomasis uždavinys aprašo biznio logiką, kurią reikia kompiuterizuoti ("Use cases discribe the basic bussines logic of an application").

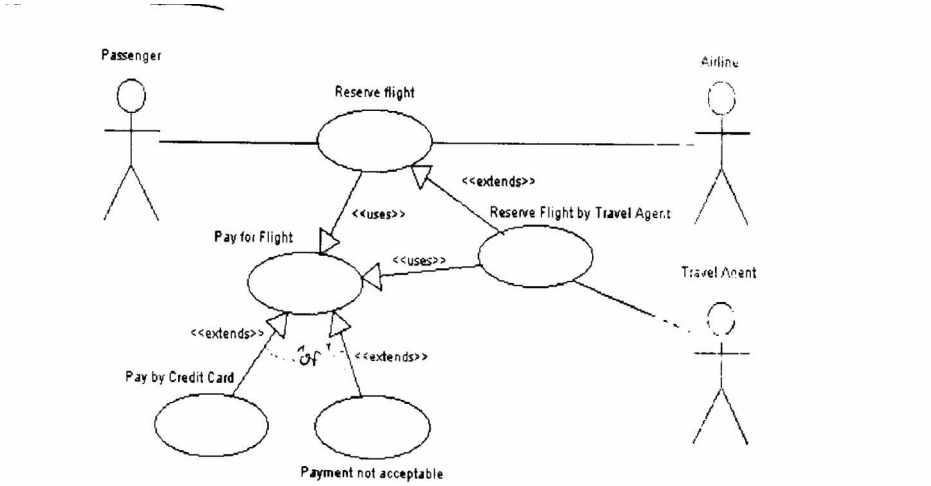
UCD elementai: taikomasis uždavinys diagramoje žymimas elipse, dalyvis – žymimas stačiakampiu, duomenų srautas – rodykle. Visi UCD elementai būtinai turi pavadinimus. Tarp panaudojimo atvejų gali būti nurodomi ryšių tipai: "extends" ir "uses".

Ryšys "extends" (išplečia) nurodo, kad vienas uždavinys panaudoja kitą uždavinį (išplečia savo funkcionalumą), paveldi to kito uždavinio savybes arba jas perdengia (overrides). Šis ryšys tarp uždavinių įgalina modeliuoti alternatyvias veiksmų sekas "jeigu…, tai..), susidarančias detalizuojant (tikslinant) pradines *use case* diagramas, t.y kuriant detalias (žemesnio lygmens) *use case* diagramas uždaviniams, nurodytiems pradinėje *use case* diagramoje. "Extends" ryšį galima palyginti su apibendrinimo ryšiu klasių diagramose.

Ryšys "uses" (naudoja) reiškia, kad vienas taikomasis uždavinys naudoja kitą taikomąjį uždavinį konkrečiam savo etapui (veiksmui, procesui) atlikti. Ryšį "uses" galima palyginti su agregavimo ryšiu klasių diagramose.

Kiekvienas diagramoje nurodytas taikomasis uždavinys yra aprašomas tekstu arba detalizuojamas kitomis UML diagramomis:

- veiklos diagrama *(activity d.)* – modeliuoja scenarijaus vykdymo eigą;
- sekų diagrama *(sequence d.)* ar/ir bendradarbiavimo diagrama (collaboration d.);
- modeliuoja objektų tarpusavio santykius (siunčiamus pranešimus ir atliekamus metodus/operacijas).



7 pav. Panaudojimo atvejų diagramos (use case diagram) pavyzdys

Panaudojimo atvejų diagramos (UCD) yra naudojamos testuoti sukurtą IS – tikrinti, ar realizuoti pirminiai vartotojo reikalavimai (informaciniai poreikiai) bei reikalavimai IS , kurie užfiksuoti visose sudarytose projektavimo metu UCD. UCD pagrindu sudaromi IS testavimo skriptai[2].

2.4.5 Sekos diagramos SeD (Sequence Diagrams)

Sekos diagramos SeD naudojamos panaudojimo atvejų scenarijaus (use case scenario) logikos aprašymui.

Sekos diagramos SeD modeliuoja uždavinio ar veiklos, aprašytos panaudojimo atvejų scenarijuje, vykdymo eilės tvarką: nurodo, kokie objektai dalyvauja, kokia eilės tvarka ir kokius pranešimus (bei atsakymus) jie siunčia vienas kitam. Eiliškumo diagramos yra pats geriausias būdas nagrinėti projektą, nes priverčia patikrinti, ar gali būri įvykdytas “use case” scenarijus. Sekos diagramas sudaro programuotojai arba projektuotojai, kurie aiškinasi, kaip bus realizuojamas uždavinio (problemos) scenarijus.

2.4.6 Bendradarbiavimo diagramos CLD (Collaboration Diagrams)

Klasių bendradarbiavimo diagramos (CLD) skirtos klasių elgsenos, vykdant taikomąjį uždavinį, aprašymui: jos modeliuoja pranešimų srautus tarp objektų OO taikomajame uždavinyje. CLD nėra atvaizduojamos objekto būsenos.

Pranešimus vaizduoja tekstas kartu su rodykle, esantis prie asociacijos tarp klasių ženklo. Atsakymus į pranešimus vaizduoja tekstas ir rutuliuku prasidedanti rodyklė.

Klasių diagramose CD nėra vaizduojami pranešimai. Tai apsunkintų CD analizę, skaitymą. Kaip tik objektų sąryšiui per pranešimus modeliuoti ir skirtos CLD.

2.4.7 Būsenų diagramos SD (State Diagrams)

Objektus apibūdina du dalykai: būsena ir elgsena. Kai kurie objektai yra sudėtingi. Siekiant suprasti jų elgseną, sudaromos klasių būsenų diagramos (state diagrams StD).

Būsenų diagrama modeliuoja vienos klasės objektų elgseną realiame laike – objektų būsenas ir perėjimus (transitions) iš vienos būsenos į kitą. Būsena žymima apskritimu su pavadinimu, perėjimas – rodykle.

Objekto būseną *(state)* aprašo visuma atributų reikšmių, kurias gali įgyti objektas.

Perėjimus *(transitions*) aprašo įvykiai (events), kurie keičia objekto būsenas *(trigger states)*. Perėjimai yra siejami su sąlygomis *(conditions),* kurios modeliuoja biznio taisykles.

StD yra dvi psiaudo būsenos : pradinė, kuri reiškia, kad objektas sukurtas; pabaigos būsena, iš kurios objektas negali išeiti, jei jau pateko.

Būsenų diagrama siejasi su kitomis UML diagramomis:

- Būsenų diagrama aprašo objektų klasės vidinę sandarą ir elgseną, o bendradarbiavimo diagramos CLD *(collaboration diagrams)* aprašo išorines klasių sąveikas;
- Veiklos diagrama *(activity d.)* gali būti naudojama kaip būsenų diagramos išplėtimas nurodyti veiksmus, kurie atliekami konkrečioje būsenoje, kai įvyko konkretus įvykis būsenos viduje.

2.4.8 Veiklos diagrama AD (Activity Diagram)

Veiklos diagramos paskirtis – aprašyti sistemos elgseną. Veiklos diagrama yra procesų modeliavimo priemonė, kuri gali modeliuoti:

- atskiro veiklos proceso ar funkcijos (t.y. *use case*) eigą (logiką);
- klasės elgseną;
- sudėtingo metodo (operacijos) logiką.

Viena veiklos diagrama yra siejama su viena klase, arba viena operacija, arba vienu taikomuoju uždaviniu *(use case).* Veiklos diagramos paskirtis - aprašyti vidinius sistemos procesus.

Veiklos diagrama yra objektinis modelis, kuris atitinka DFD (duomenų srautų diagramą) struktūriniame-funkciniame modeliavime[2].

Svarbi veiklos diagramų savybė – jos gali modeliuoti lygiagrečius procesus.

Veiklos diagramos yra dokumentuojamos: aprašomas kiekvienas procesas (grafo viršūnė) veiksmai, susiję su procesu, ir jų sąsajos (nurodant įvykių pavadinimus). Be to, kiekvienam procesui detaliau aprašyti gali būti sudaromos veiklos diagramos (žemesnio lygmens).

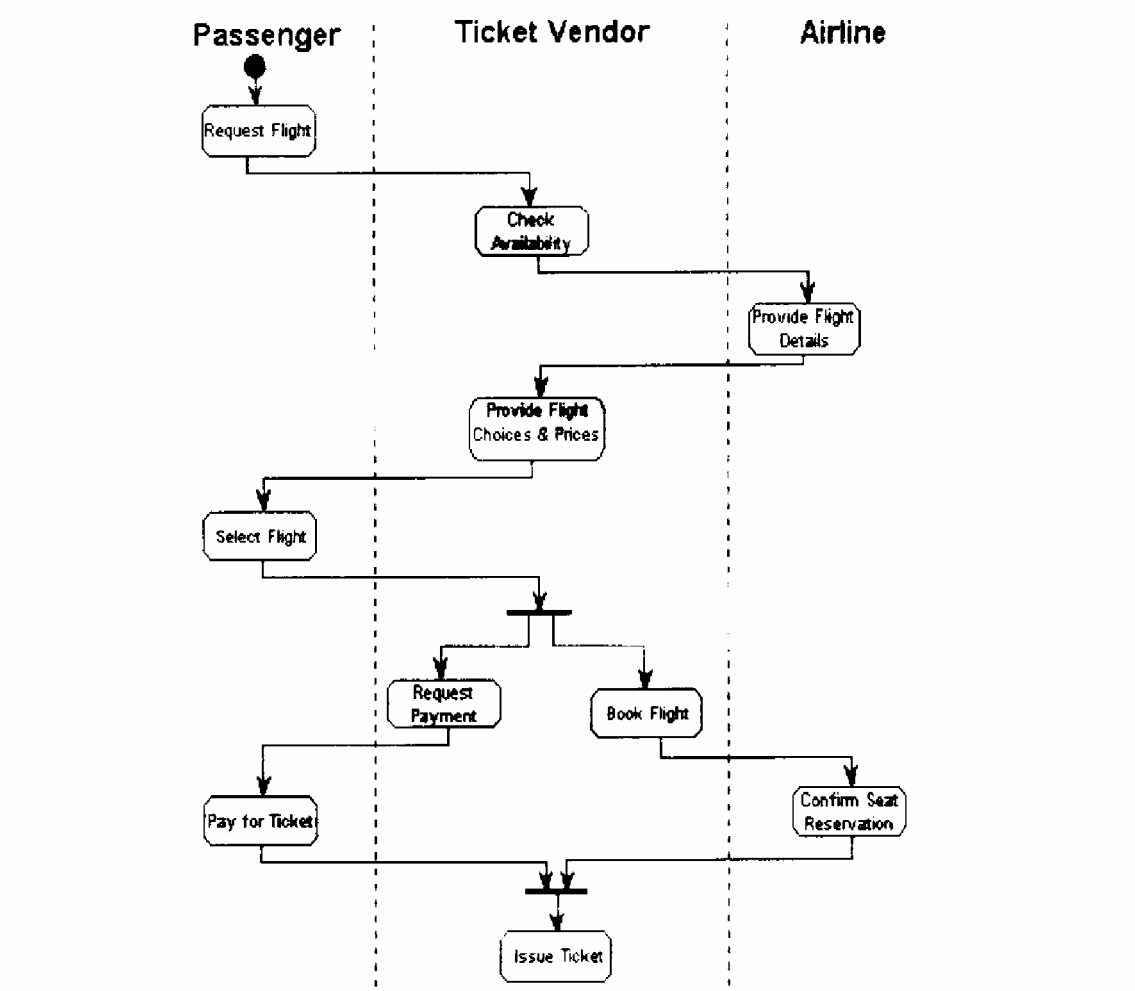
- Veiklos diagramų panaudojimas biznio procesams ar funkcijoms (t.y. *use case*) modeliuoti

Veiklos diagrama galima aprašyti (vietoj teksto) taikomąjį uždavinį (use case diagramoje). Tokioje veiklos diagramoje galima nurodyti taikomojo uždavinio žingsnius. Kiekvieną uždavinio žingsnį galima detaliau aprašyti tekstu arba sudarant jam veiklos diagramą (žemesnio lygmens).

- Veiklos diagramų panaudojimas klasėms modeliuoti

Paprastai klasės elgseną aprašo būsenų diagramos *(state diagrams)*. Veiklos diagramos naudojamos tais atvejais, kai įvykius sukelia būsenose atliekami veiksmai. Tokia veiklos diagrama yra atskiras būsenų diagramos *(State diagram)* atvejis, kai dauguma būsenų (grafo viršūnės) yra veiksmo būsenos *(action states)* ir transakcijas inicijuoja *(are triggered by)* veiksmai, atliekami pradinėse būsenose. Todėl, modeliuojant klases, galima priskirti klasėms veiksmus (veiklas) iš anksto, dar nesudarius veiklos diagramų[2].

Veiklos diagramas patartina naudoti, kai dauguma ivykių yra sukelti dėl vidinių veiksmų, t.y. jos naudojamos valdymo procesų aprašymui *(procedural flow of control).*



8 pav. Veiklos diagrama

UML metodikos pagrindas – anksčiau naudotos OOAD, OMT ir OOSE koncepcijos. Jos susideda iš diagramų, skirtų analizei, modeliavimui, projektavimui ir testavimui, tačiau iš vartotojo sąsajos modeliavimo ir projektavimo klausimų yra nagrinėjami tik elementinė ir funkcinė sudėtis bei veiksmų seka.

Iš visų UML diagramų apžvalgos galima daryti daryti išvadą. Pereiti nuo UML modelio prie Petri tinklo modelio labiausiai tiktų UML veiklos būsenų diagrama, nes ji yra artimiausia ir panašiausia į Petri tinklo struktūrai, sekų diagrama yra netinkama, kadangi analizuojant ir sudarant sistemos modelį jis gaunasi painus ir sudėtingas.

2.5 Petri tinklai ir jų panaudojimas

Petri tinklai (PT) – tai vienas iš formalių lygiagretumo metodų. Jie sėkmingai naudojami sistemose, kuriose vyksta lygiagretūs diskretūs procesai. PT ypač dažnai sutinkami paskirstytose sistemose, pvz., operacinėse sistemose, gamybos, vadybos, verslo ar programinės įrangos kūrimo procesuose.

Šie tinklai pritaikomi jau egzistuojančiose sistemose siekiant atlikti įvairius veiksmus: poreikių analizę, specifikavimą, projektavimą, testavimą, imitavimą ir formalią elgsenos analizę [ISO 97]. Su tokiomis sistemomis susijusių projektų vykdymo metu yra sukuriama programinė įranga, kuri arba tarnauja kaip sistemos veiksmų palaikymo įrankis, arba valdo jos elgseną, arba yra galutinės realizacijos dalis. Tokiais atvejais PT yra naudojami tik pradinėse projekto stadijose ir visai nenaudojami programinės įrangos realizacijos metu. Tai lemia faktas, jog PT nėra programavimo kalba. Šis apribojimas verčia pakeisti koncepcinę sistemos supratimo schemą.

2.5.1 Klasikinis Petri tinklas

Klasikinis Petri tinklas – tai orientuotas dvipusis grafas, turintis du mazgų tipus: vietas ir perėjimus. Tinklo mazgai sujungti orientuotais lankais. Sujungti du to paties tipo mazgus draudžiama. Vietos yra žymimos apskritimais, o perėjimai – kvadratais [7].

Petri tinklas – tai trejetas (P, T, F), kur:

- P yra baigtinė vietų aibė,
- T yra baigtinė perėjimų aibė ($P \cap T = 0$),
- $F \subseteq (P \times T) \cup (T \times P)$ yra lankų aibė.

Vieta p yra vadinama perėjimo t įėjimo vieta, jeigu egzistuoja orientuotas lankas iš p į t. Vieta p yra vadinama perėjimo t išėjimo vieta, jeigu egzistuoja orientuotas lankas iš t į p. Bet kuriuo laiko momentu kiekviena vieta turi nulį ar daugiau žymių[7].

Žymių skaičius gali keistis vykdant tinklą. Petri tinkle aktyvūs komponentai yra perėjimai. Jie keičia tinklo būseną pagal tokią taisyklę:

1. perėjimas t yra laikomas galimu, jeigu kiekviena iš t įėjimo vietų p turi bent po vieną žymę;
2. aktyvus perėjimas gali įvykti. Jei perėjimas t įvyksta, t sunaudoja po vieną žymę iš kiekvienos t įėjimo vietos p ir pagamina po vieną žymę kiekvienai t išėjimo vietai p;

2.5.2 Aukšto lygio Petri tinklai

Klasikinis Petri tinklas suteikia galimybę modeliuoti būsenas, įvykius, sąlygas, sinchronizavimą, lygiagretumą, pasirinkimą ir iteracijas. Tačiau Petri tinklai, aprašantys realius procesus yra sudėtingi ir labai dideli. Be to, klasikiniai Petri tinklai nesuteikia galimybės modeliuoti duomenis ir laiką. Siekiant išspręsti šias problemas, buvo pasiūlyta daug tinklo papildymų ir patobulinimų. Trys žinomiausi papildymai yra šie[7]:

1. spalvoti Petri tinklai (CPN) - spalvos įvedimas duomenims modeliuoti;
2. laikiniai Petri tinklai (TPN) - laiko įvedimas;
3. hierarchijos panaudojimas didelių modelių struktūroms aprašyti.

Petri tinklas, papildytas spalvos, laiko ir hierarchijos panaudojimo galimybėmis, vadinamas aukšto lygio Petri tinklu. Plačiau apie kiekvieną iš trijų papildymų:

1. *CPN*. Žymės dažnai atitinka objektus (resursus, duomenis, signalus) modeliuojamoje sistemoje. Dažnai norima atvaizduoti šių objektų atributus (pavadinimą, numerį, kiekį). Tai nėra paprasta padaryti naudojant klasikinio Petri tinklo žymes, todėl įvedamos spalvotos žymės. Spalvotame Petri tinkle, kiekvienos žymės tipą ar netgi reikšmę nurodo jos spalva. Perėjimai aprašo gaminamų žymių spalvas, remiantis sunaudotų žymių spalvomis (aprašo ryšį tarp įėjimo žymių ir išėjimo žymių). Taip pat galima aprašyti „prieš sąlygas“, kurios atsižvelgia į sunaudojamų žymių spalvas [7].
2. *TPN*. Realiose sistemose dažnai svarbu aprašyti sistemos elgesį laike, t. y. reikia modeliuoti trukmes ir vėlinimus. Kadangi klasikinis Petri tinklas negali aprašyti laiko, jis yra papildytas laiko sąvoka. Yra daug būdų, kaip įvesti laiką Petri tinkluose. Laikas gali būti susietas su žymėmis, vietomis ir/arba perėjimais [7].
3. *Papildymas hierarchija*. Nors papildytų laiku ir spalva Petri tinklų pakanka aprašyti daugeliui procesų, bet tikslios realių sistemų specifikacijos dažnai yra didelės ir sudėtingos. Dėl šios priežasties yra įvedama hierarchijos konstrukcija, vadinama potinkliu. Potinklis agreguoja tam tikrą skaičių vietų, perėjimų ir posistemių. Tokios konstrukcijos dėka galima struktūrizuoti didelių procesų aprašymus. Viename lygyje galima pateikti paprasčiausių proceso aprašą (nedetalizuojant), o kitame lygyje – šį aprašą detalizuoja [7]

2.5.3 Petri tinklų savybės

Pirma iš savybių yra susijusi su ribota atminties talpa realiuose sąlygose realizuojant įvykius. Dirbant Petri tinklams kai kurios tinklo vietos gali sukaupti neribotą žymių skaičių. Jeigu interpretuoti vietą kaip kaupiklį (duomenų, signalų ar kitokios informacijos buferį), tai reikia pareikalauti, kad bet kokiu sistemos funkcionavimo atveju neįvyktų kaupiklių perpildymas, kurie realiuose sąlygose turi realią talpa.

Vieta p Petri tinkluose M = (P, T , I, O, m) vadinama ribota, jei egzistuoja toks skaičius n, kad esant bet kokiam atvejui žymių skaičius vietoje m(p)=<n.

Tinklas vadinamas ribotu, jei visos vietos jame yra ribotos.

Vieta p vadinama nepavojinga, jei joje gali būti ne daugiau kaip viena žymė. Atitinkamai tinklas vadinamas nepavojingu, jei kiekviena jo vieta nepavojinga.

Konservuoti tinklai - tai tinklai kuriuose žymiu suma yra visą laiką pastovi t. y. kiekvieno perėjimo suveikimas nepakeičia žymių skaičiaus tinkle. Perėjimas tinkle gali sudirbti esant atitinkamoms sąlygoms, susijusioms su žymių išsidėstymu jo įėjimo vietose. Gali taip atsitikti, kad kuriam nors perėjimui jo suveikimo sąlyga negali būti išpildyta, kaip tik neveiktu tinklas. Toks perėjimas vadinamas mirusiu. Jis yra nereikalingas tinkle, jį galime išmesti iš tinklo be žalos. Gali taip pat atsitikti tokia situacija, kad po tam tikrų perėjimų suveikimų tinkle, pakeitusių žymių išsidėstymą, kai kurie perėjimai, jų tarpe ir suveikę, niekada daugiau nesuveiks, kokie tik žymių išsidėstymo variantai nebūtų. Tokie perėjimai vadinami potencialiai mirę. Priešingos sąvokos yra gyvi ir potencialiai gyvi perėjimai. Gyvas perėjimas vadinamas toks, kuris turi galimybę sudirbti esant bet kokiam žymių išsidėstymui tinkle. Potencialiai gyvas perėjimas yra toks, kuris turi galimybę suveikti esant tam tikram žymių išsidėstymui tinkle. Atitinkamai, jei visi perėjimai tinkle gyvi - tinklas vadinamas gyvu, jei visi mirę - mirusiu. Tinklas vadinamas potencialiai gyvu, jei jame yra potencialiai gyvu perėjimu ir potencialiai mirusiu, jei jame yra potencialiai mirusių vietų.

Dažniausiai Petri tinklai yra naudojami norint įrodyti sistemos ypatybes, tokias kaip gyvybiškumas, ribotumas bei stabilumas, o ne pačiam sistemos specifikavimui. Taigi sistemos arba, tikriau sakant, tam tikros sistemos dalys yra modeliuojamos naudojant Petri tinklus. Paprastas Petri tinklų modelis yra gana ribotas. Vis dėlto pasitelkus aukšto lygio Petri tinklus galima aprašyti lygiagrečias sistemos dalis [7].

Apskritai, Petri tinklai yra labai lankstūs ir tinkami tiek pastovių, tiek besikeičiančioms sistemų modeliavimui.

3. Mobiliosios informavimo sistemos procesų teorinis tyrimas

Pagrindinis tiklas yra mobiliosios informavimo sistemos procesų sistema. Išanalizavę procesų sistemą, galime sudaryti UML modelius ir prijungti juos prie Petri tinklų.

3.1 Modelio analizė

Kauno technologijos universitete studentai pakankamai dažnai susiduria su problema – dėstytojas neatėjo į paskaitą, nes buvo pakviestas į susirinkimą, ar perkėlė susitikimo laiką. Iki šiol dažniausiai ši problema būdavo sprendžiama pakabinant skelbimą apie pasikeitimus fakulteto skelbimų lentoje. Dėl šios ir panašių problemų yra gaištamas studentų laikas, kurie nežino dėstytojo tvarkaraščio, juo labiau, kada jiems skiriami susirinkimai ar iškyla kitokios iš anksti nenumatytos problemos, dėl kurių neįvyksta paskaita ar susitikimas.

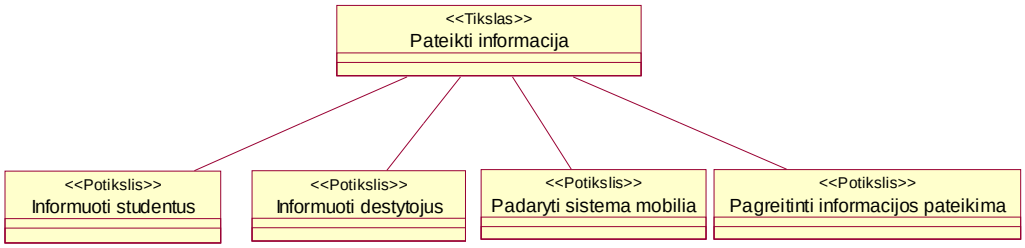
Siekiant išvengti tokių nesusipratimų ir informuoti studentus bei dėstytojus apie dienotvarkę ir jos pasikeitimus, reikalinga greitai, patogiai bei kokybiškai pateikti informaciją. Vienas iš sprendimo būdų – internetinėmis bei mobiliomis technologijomis pagrįsta mobili informavimo sistema, padedanti informuoti suinteresuotus asmenis iš bet kurio kompiuterio, turinčio prieigą prie interneto.

3.1.1 Modelio procesų analizė

Magistrinio darbo tiksle, buvo apibrėžta, kad analizuosime UML diagramas ir jų perėjimą prie Petri tinklų, todėl pradinį sistemos modelį modeliuojame UML kalba.

3.1.1.2 Veiklos tikslų modelis

Visi organizacijos tikslai yra atvaizduojami tikslų modelyje (9 pav.). Organizacijos tikslai - visos veiklos, kuriais įgyvendinami organizacijos pagrindiniai uždaviniai.



9 pav. Veiklos tikslų modelis

Šiais organizacijos tikslais paremta ir „Mobiliosios informavimo sistemos“ struktūra bei jos funkcionalumas.

3.1.1.3 Veiklos procesų modelis

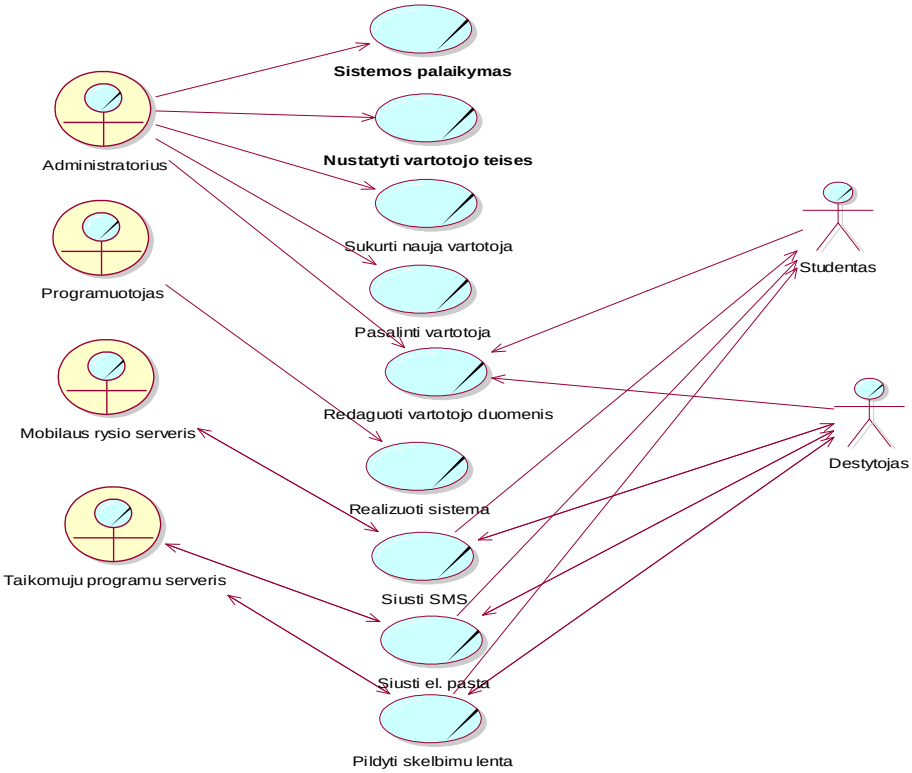
Iš veiklos panaudojimo atvejų modelio (8 pav.) matosi, kad „Mobili informavimo sistema“ skirta dėstytojams ir studentams [1] . Vartotojai skirstomi į dvi dalis, studentus ir dėstytojus.

Studentai gali atlikti šiuos veiklos procesus:

- gauti pranešimus iš dėstytojų apie planų pakeitimą;
- peržiūrėti dėstytojo dienotvarkę;
- siųsti elektroninius paštus dėstytojams ir grupių, kurioms jis priklauso, studentams;
- skaityti skelbimus;
- redaguoti savo asmeninę informaciją.

Dėstytojai gali atlikti šiuos veiksmus:

- sudaryti ir tvarkyti savo dienotvarkę;
- peržiūrėti kolegų dienotvarkes;
- siųsti pranešimus dėstytojams ir studentams apie planų pasikeitimą (pvz.: susitikimą, susitikimo perkėlimą į kitą laiką ir t.t.);
- gauti pranešimus ;
- įdėti naują skelbimą;
- redaguoti savo asmeninę informaciją.



10 pav. Sistemos vartotoju panaudojimo atvejų modelis

Iš šios diagramos galima sudaryti dėstytojo bei studento panaudojimo atvejų modelį.

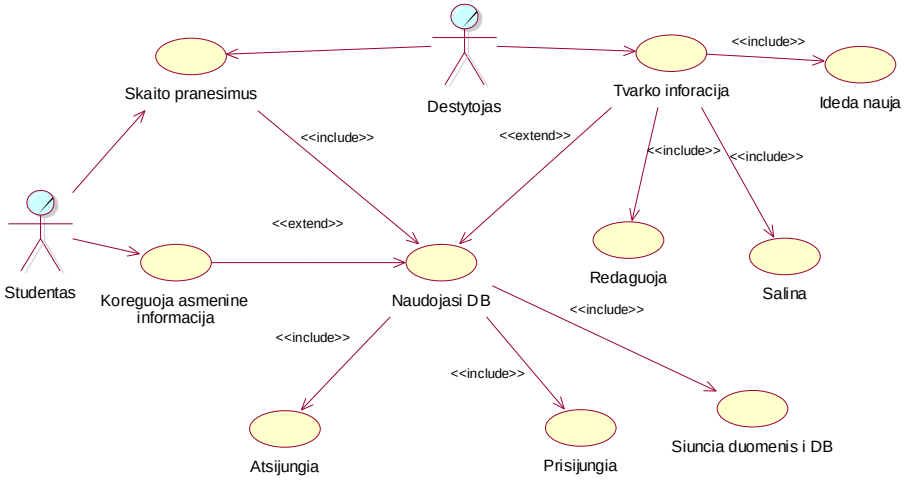
3.1.2 Analizės išvados

Šiuo metu rinkoje egzistuoja panašių į „Mobilią informavimo sistemą“ sistemų, tačiau dauguma jų yra komercinės ir mokamos. Nemokamos sistemos nepasižymi dideliu funkcionalumu bei patogumu naudoti.

Šios sistemos analizės modeliai nubraižyti UML pagrindu. Iš visų modelių aiškiai matyti, kokius veiksmus gali atlikti sistemos vartotojai, sistemos objektų ryšiai, veiklos procesai. „Mobiliosios informavimo sistemos“ vartotojai yra 3 rūšių: sistemos administratorius, dėstytojai ir studentai. Administratorius gali daryti tai, ko negali kiti – ištrinti vartotoją, redaguoti bet kurio iš jų duomenis. Dėstytojas turi žymiai daugiau teisių nei studentas, jis gali ne tik peržiūrėti pateiktą informaciją, bet ją pateikti ir pats, siųsti žinutes elektroniniu paštu bei trumpąja SMS žinute. Visi duomenys yra saugomi pagrindiniame serveryje, todėl jie yra apsaugoti. Kadangi sistema yra valdoma vien interneto naršykle, todėl papildomos programinės įrangos pas vartotoją nereikia įdiegti. Vartotojai gali matyti ir keisti tik savo asmeninę informaciją, tokiu būdu yra užtikrinamas konfidencialumas.

3.2 Vartotojų panaudojimo atvejų modelis

11 paveiksle pateiktas vartotojų panaudojimo atvejų modelis. Iš šio modelio matyti, kad tik dėstytojas gali kurti pranešimus, tačiau ir studentas ir dėstytojas gali peržiūrėti kitų pedagogų tvarkaraščius, bei skelbimus.

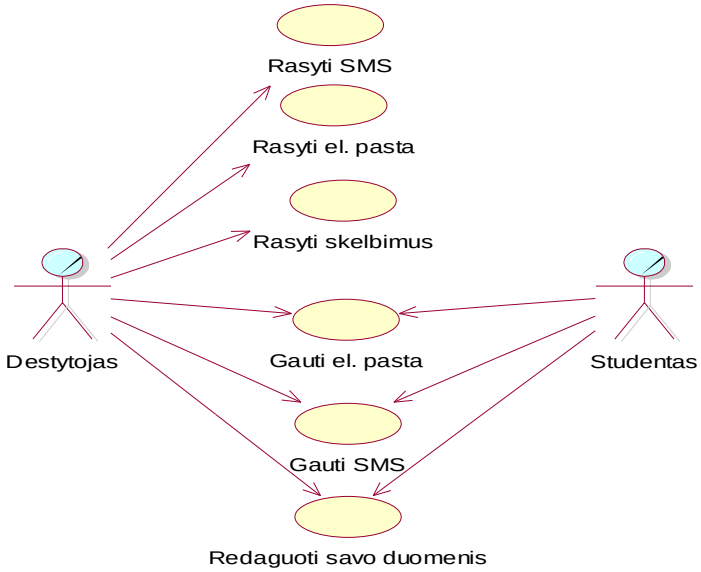


11 pav. Vartotojų panaudojimo atvejų modelis

Šiame modelyje matosi visi vartotojų veiksmai, jų galimybės bei veiksmai sistemoje. Visa tai realizuota realioje sistemoje.

3.3 Studento ir dėstytojo veiklos procesas

Tam, kad sužinoti, kokias veiklas gali atlikti sistemos vartotojai, yra sudaroma studento ir dėstytojo panaudojimo atvejų modelis [4] . Šiame modelyje (12 pav.) aiškiai matyti, kad dėstytojas turi žymiai daugiau teisių ir gali atlikti daugiau veiksmų sistemoje, nei studentas.

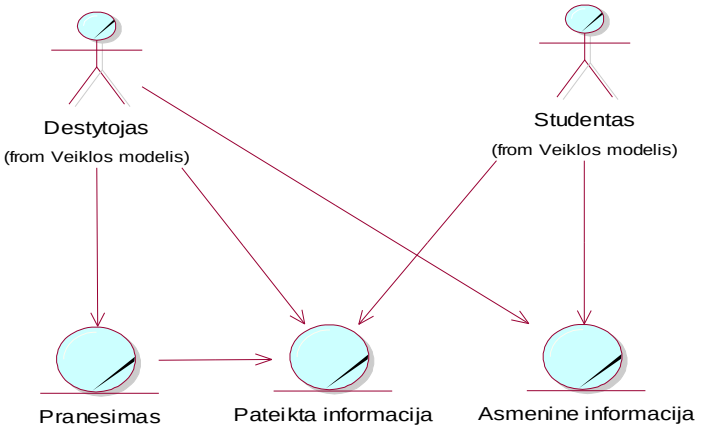


12 pav. Studento ir dėstytojo panaudojimo atvejų modelis

Tokiu principu yra atskiriami du vartotojų tipai bei jų teisės. Ši struktūra yra įgyvendinta realioje sistemoje.

3.4 Veiklos objektų modelis

Pagrindiniai veiklos objektai yra: dėstytojas ir studentas. Jie tarpusavyje dalinasi informacija(13 pav.). Asmeninė informacija yra pasiekiama tik patiems vartotojams. Tokiu būdu yra užtikrinamas konfidencialumas.

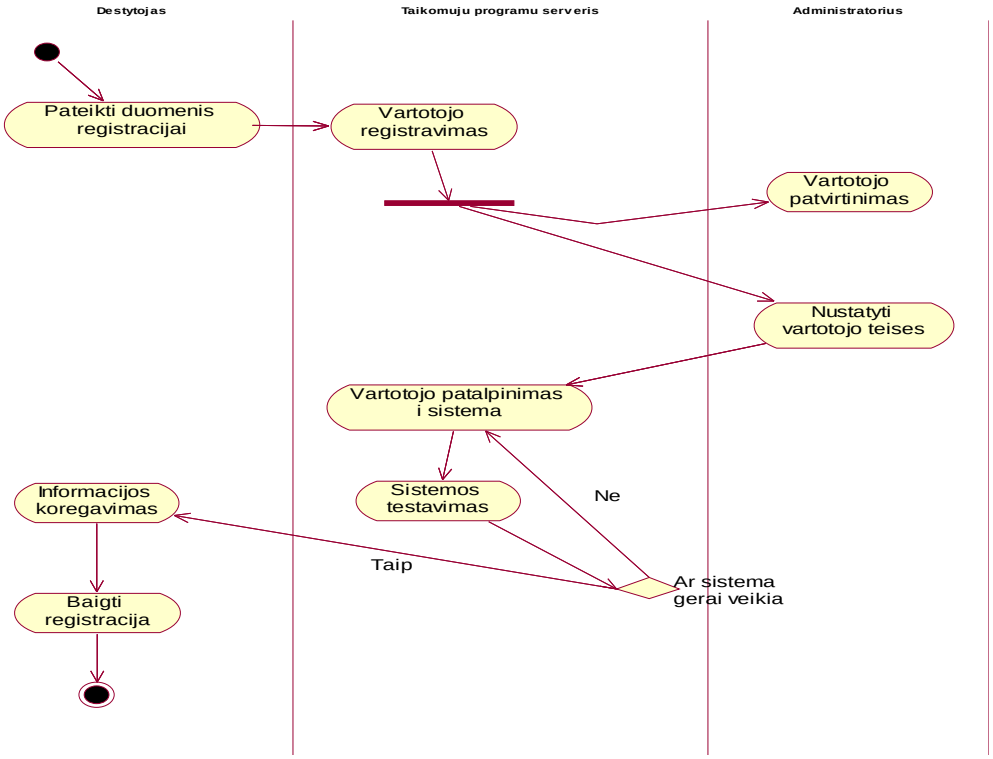


13 pav. Veiklos objektų modelis

Kadangi „Mobili informavimo sistema“ skirta tik informuoti, veiklos objektams nėra numatyta papildomų veiklų.

3.5 Veiklos procesų modelis

Šiame modelyje (14 pav.) matosi, kokie procesai ir kaip vyksta sistemoje. Šioje diagramoje pavaizduotas naujo vartotojo prisijungimas prie sistemos, duomenų apie naująjį vartotoją talpinimas į sistemą. Šiuo atveju dalyvauja ir sistemos administratorius, kuris tvirtina naują vartotoją. Vartotojo patvirtinimas reikalingas tik registruojantis dėstytojui, o studentui – ne, nes studentas šioje sistemoje turi ribotas teises.

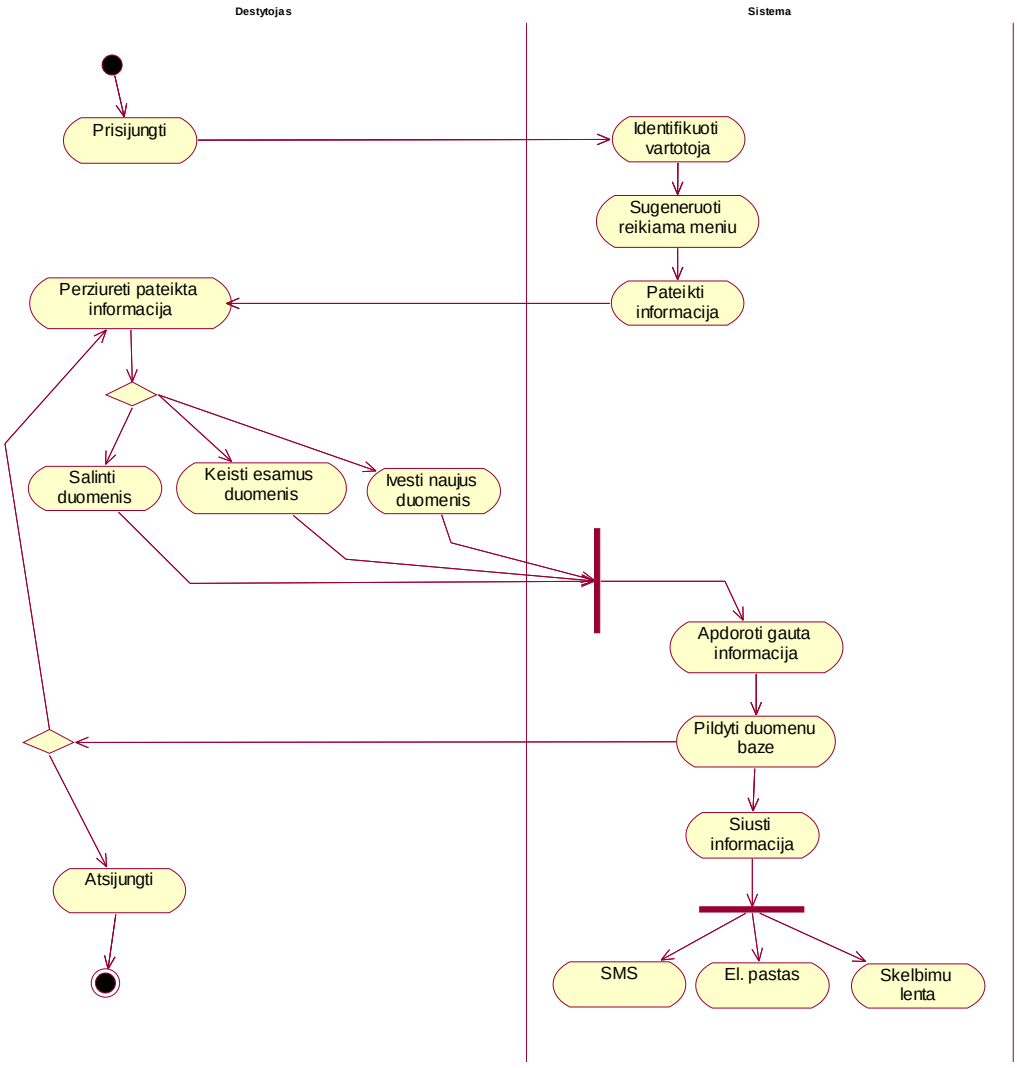


14 pav. Veiklos procesų modelis

Veiklos procesų modelyje pavaizduoti visi sistemoje vykstantys procesai, jų eiliškumas bei sąlygos naujo vartotojo registravimo atveju. Tokiu principu yra užtikrinamas saugumas, nes ne bet kuris prie sistemos prisijungęs asmuo gali gauti dėstytojo teises.

3.6 Sistemos elgsenos modelis, kai vartotojas dėstytojas

Sistemos elgsenos, kai vartotojas dėstytojas, (15 pav.) pavaizduota visos sistemos elgsena, kai prisijungęs dėstytojas ir jis vykdo įvairius jam leistinus veiksmus.

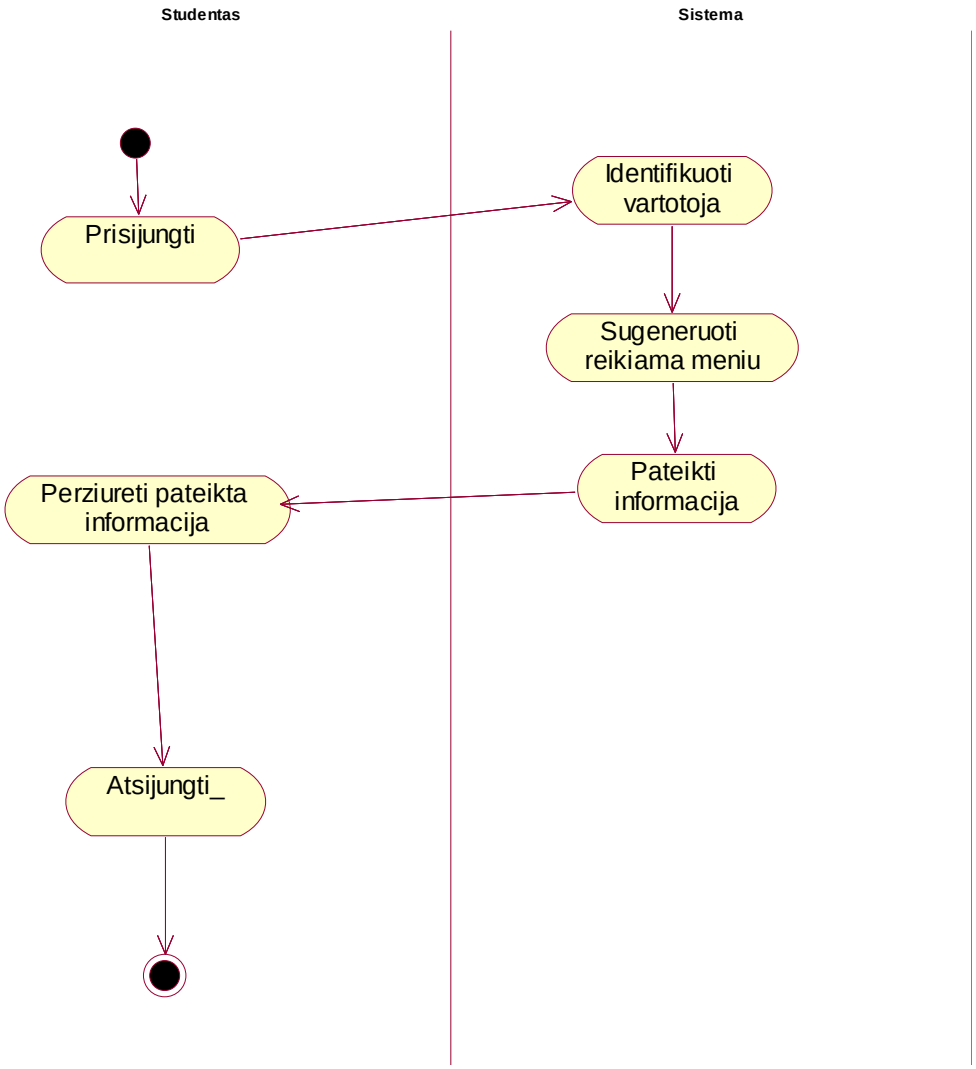


15 pav. Sistemos elgsena, prisijungus dėstytojui

Analizuojant pateiktą diagramą pastebime, kad atsiranda papildomos šakos tik tose vietose, kur keli veiksmai gali būti vykdomi vienu metu. Lygiagretūs veiksmai sujungiami, todėl, kad sekantis veiksmas gali priimti tik vieno iš jų rezultatus

3.7 Sistemos elgsenos modelis, kai vartotojas studentas

Šiame modelyje pavaizduota (16 pav.) visos sistemos elgsena, kai prisijungęs studentas ir jis vykdo įvairius jam leistinus veiksmus.



16 pav. Sistemos elgsena, prisijungus studentui

Studentas sistemoje gali atlikti mažiau veiksmų, nei dėstytojas, todėl modelis paprastesnis, nei dėstytojo atveju.

3.8 UML diagramų vertimas į Petri tinklą

3.8.1 Laiko įvertinimas

Mobiliai informavimo sistemai nagrinėti pasirinkome laikinius Petri tinklus. Tarp kiekvieno sistemoje vykstančio įvykio yra laiko tarpas, po kurio yra inicijuojamas kitas įvykis. Kiekvienas įvykis turi įvykti per tam tikrą nustatyta laiką.

Mobili informavimo sistema yra įvykiais premta sistama. Vieno įvykio inicijavimas, sąlygoja kito įvykio inicijavimą. Laiko ribos φ ir ω tarp vykstančių įvykių γ_a ir γ_b įvertinamos tokia nelygybe

$$\phi \le \tau(\gamma_a) - \tau(\gamma_b) \le \varpi$$

3.8.2 Petri tinklas

Šiame darbe nagrinėjamos sistemos Petri tinklo modelis turi tokią matematinę išraišką.

$$N = (P, T, I, O, M_0, t, t2),$$

kur P yra būsenų aibė, T yra ryšių aibė, I yra įėjimų aibė, O yra išėjimų aibė ir M_0 yra pradinė tinklo būsena.[1]

t: $T \rightarrow t \in R_0^+$ yra laikas susietas su perėjimu.

t2: $T \rightarrow t \in R_0^+$ antras laiko parametras (variacija)

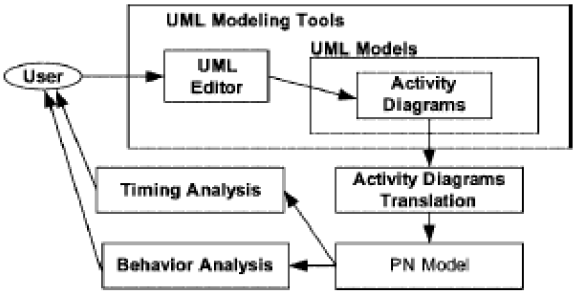
Tinklo žymėjimą nusako žetonai. Kur žetonas yra *k=(v,r) i*r k yra žetono reikšmė, o r yra žetono galiojimo laikas,

Kiekviename perėjime $t \in T$, egzistuoja minimalus perėjimo uždelsimas *d* ir maksimalus perėjimo uždelsimas *D.*

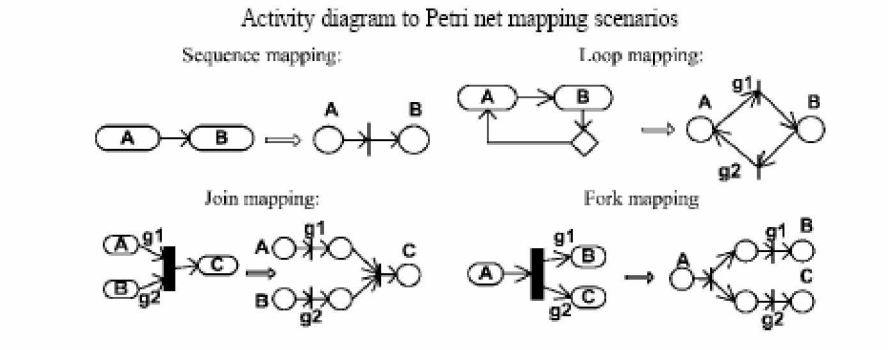
3.8.2 Perėjimas prie Petri tinklo

Perėjimas iš UML prie Petri tinklų yra aprašytas daugelyje šaltinių. Jungiant UML prie Petri tinklo yra suderinamos UML specifikacijos ir Petri tinklų formalizmas. Pirmas žingsnis yra suprojektuoti sistemą su UML, sekantis aprašyti sistemos elgseną veiklos diagrama ir paskutinis žingsnis – prijungti UML veiklos diagramą į Petri tinklą.

Perėjimo idėja yra gana paprasta, tai yra pakeisti kiekvieną veiklos diagramos būseną į Petri tinklo būseną ir kiekvieną veiklos diagramos perėjimą į Petri tinklo perėjimą.



17 pav. Perėjimas iš UML modelio į Petri tinklo modelį



18 pav. Perėjimo scenarijai

Galimi keli perėjimo scenarijai. 18 Pav. parodyti sekos, žiedinio, sujungimo ir šakutės perėjimo scenarijai.

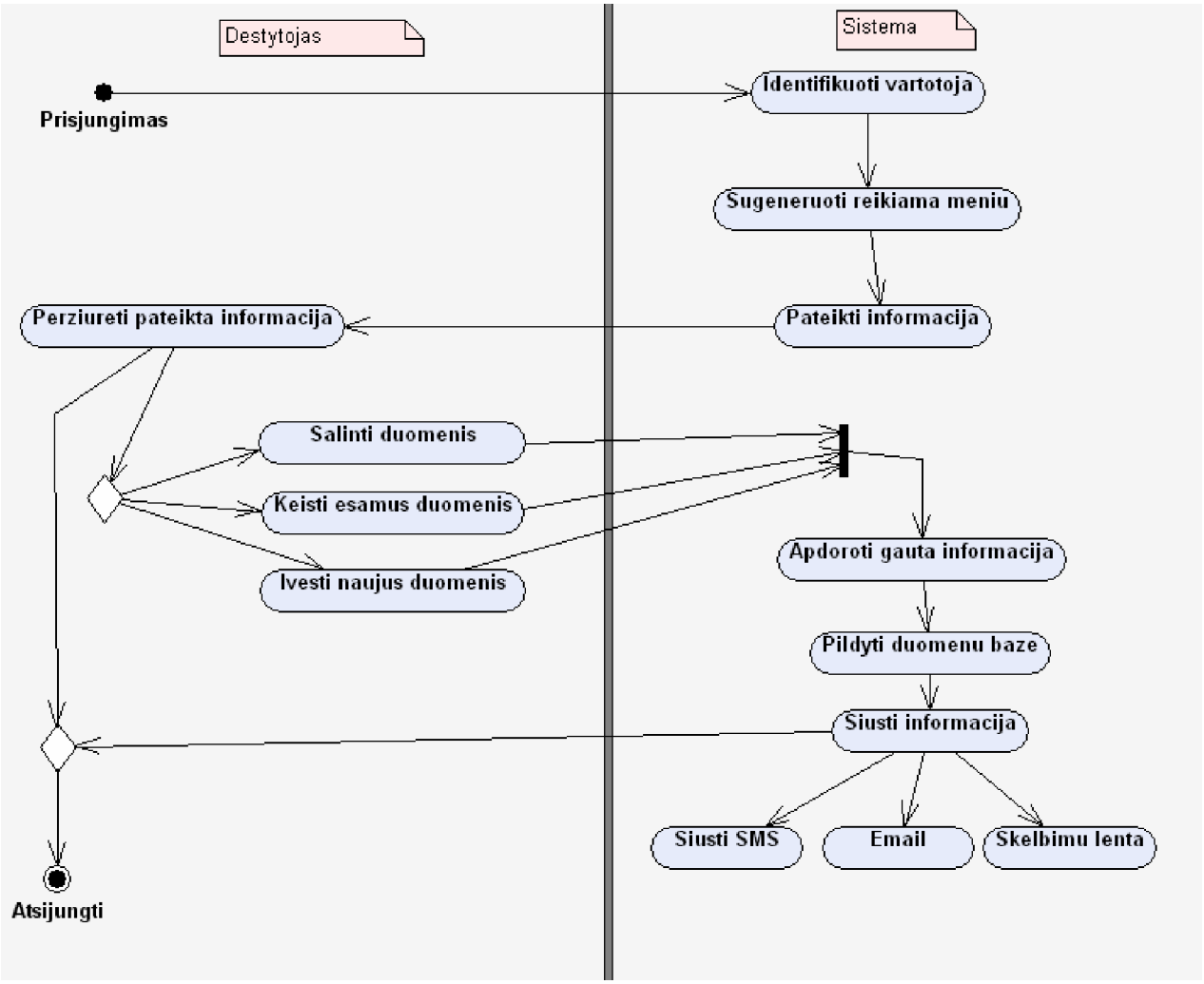
Kaip jungimo rezultatą, mes gauname saugų Petri tinklą. Saugus Petri tinkles yra $\Sigma = (S, T, F, M_0)$ yra toks tinklas, kuris kiekvienoje iteracijoje turi mažiausiai vieną žetoną kiekvienoje tinklo vietoje (Murata, 1989):

$$\forall_s \in S, \forall M \in [M_0]: M(s) \leq 1$$

Ši sąlyga yra pritaikoma pradinei veiklos diagramai, todėl galime teigti, kad mūsų atveju, pradinė veiklos diagrama yra saugus Petri tinklas.

3.8.3 Veiklos diagramos modelis

Mobilioje informavimo sistemoje galimi du veiklos scenarijai, kada su sistema dirba dėstytojas ir kada su sistema dirba studentas, šie procesai gali vykti ir kartu, t.y realiam laike dėstytojas gali talpinti į sistemą duomenis, o tuo metu studentas gali juos peržiūrinėti. Šiame darbe, analizuosime tik sistemos elgsena prisijungus prie sistemos dėstytojui, nes studentas realiai jokiu veiksmu neatlieka, išskyrus dienotvarkių peržiūrą.



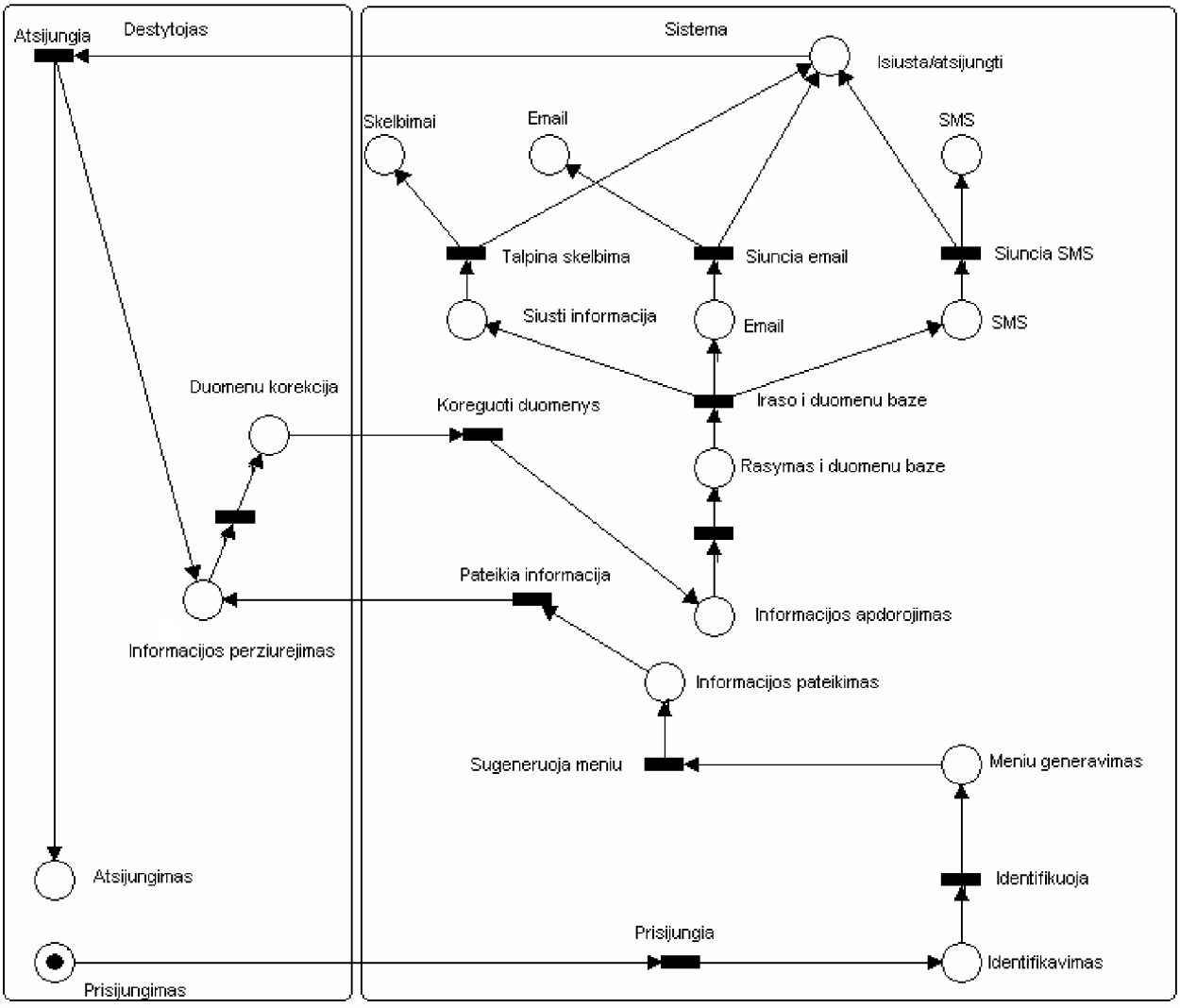
19 pav. Sistemos veiklos diagrama

Šioje veiklos diagramoje matome kokie galimi dėstytojo veiksmai prisijungus prie sistemos ir kokia yra sistemos veiksmų seka. Prisijungus prie sistemos pirmiausia identifikuojamas vartotojas, jam naršyklės lange sugeneruojamas meniu ir pateikiama informacija. Dėstytojas gali peržiūrėti informacija ir rinktis tolesnę veiksmų seką. Tolesni veiksmai yra atsijungimas nuo sistemos arba duomenų korekcija, kuris apjungia duomenų šalinimą, duomenų koregavimą ir naujų duomenų įvedimą į sistemą. Jei buvo pasirinkta duomenų korekcija, sistema apdoroja naujus duomenis, pildo informaciją duomenų bazėje, siunčia apdorota informaciją pateikimui ir informuoja visus suinteresuotus asmenis.

3.8.4 Petri tinklo modelis

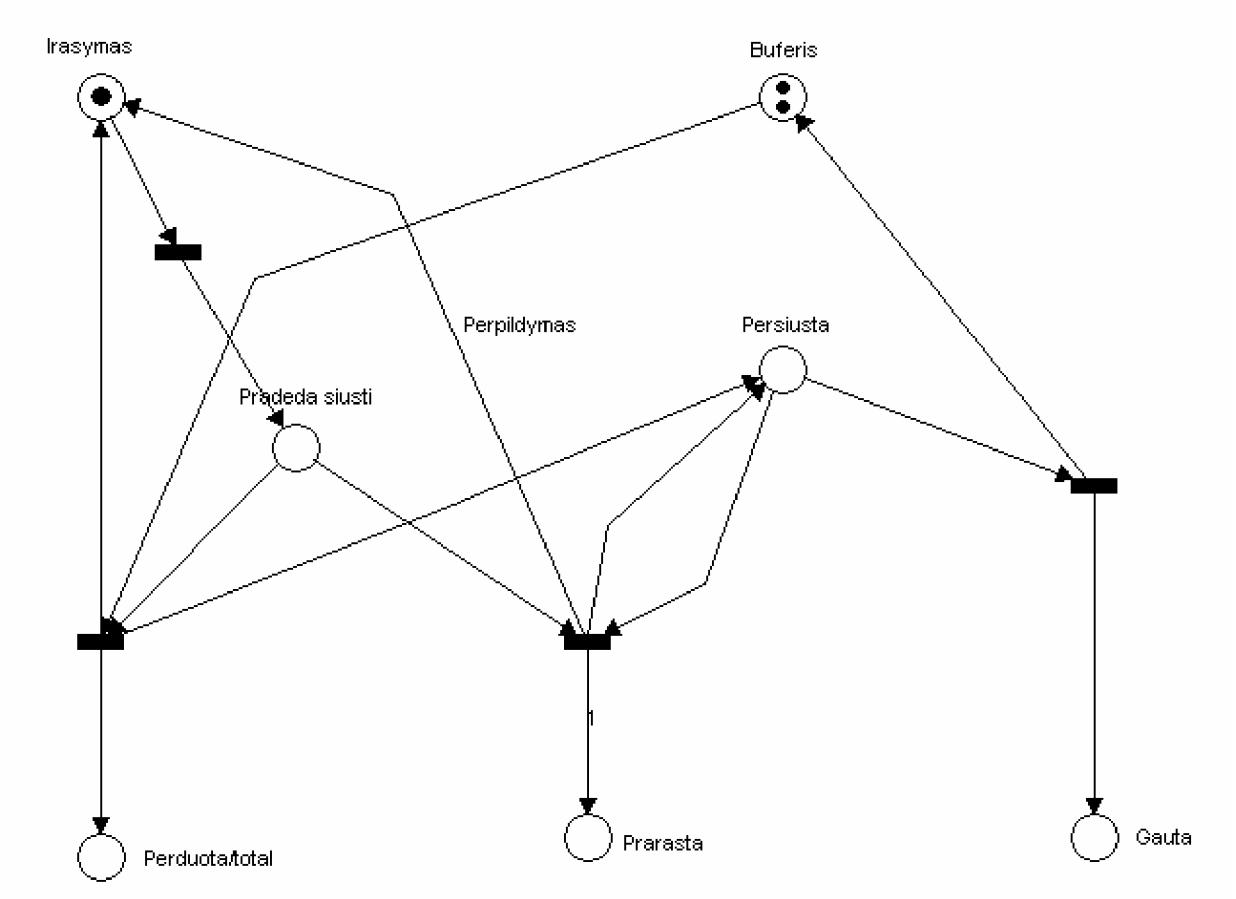
Perėjimas nuo UML modelio prie Petri tinklo modelio vykdomas gana paprastai, keičiant veiklos diagramos būsenas i Petri tinklo būsenas ir veiklos diagramos perėjimo lankus į Petri tinklo lankus.

Kai UML diagrama yra paversta į Petri tinklą, mes turime saugų, laikinų Petri tinklą, kur uždelsimo intervalai *d* ir *D* yra susieti su tinklo vieta. Uždelsimo intervalai *d* ir *D* nusako uždelsimo intervalus kiekvienai tinklo vietai. Grafiškai vieta susieta su intervalu *[d(s),D(s).* Perėjimas t turi suveikti tada, kao visi žetonai yra reikiamose tinklo vietose pries laiką *t*, nebent perėjimą sulaiko nuo paleidimo veiksmo inicijacimo kito perėjimo inicijavimas[1].



20 pav. Petri tinklo modelis sumodeliuotas is UML diagramos

Papildomai sumodeliuosime SMS pranešimų posistemę norėdami ištirti SMS posistemės stabilumą. Siunčiant SMS pranešimus yra svarbus kiekvienas duomenų praradimas, nes jam įvykus sistemos vartotojai lieka neinformuoti apie įvykius, kas mažina sistemos naudojamumą.



21 pav. SMS siuntimo posistemės Petri tinklo modelis

Visų procesų grafų viršūnėms nustatome didžiausią δ ir mažiausią ε sustadydami tokią neligybę

$$\delta \le \tau(s_{iki}) - \tau(s_{nuo}) \le \varepsilon$$

Du skaičiai δ ir ε yra maksimalus ir minimalus skirtumas tarp perėjimų t_{nuo} ir t_{iki} su ryšiu nusakomu laikiniu Petri tinklu.

Svarbiausia yra maksimalaus skirtumo analizė, nes minimalus skirtumas gali būti rastas iš sąryšio su maksimaliu skirtumu

$$\tau(s_{nuo}) - \tau(s_{iki}) \leq -\delta$$

Tai yra, apibrėžti minimalų skirtumą tarp t_{nuo} ir t_{iki}, maksimalus skirtumas tarp t_{nuo} ir t_{iki} yra apibrėžtas (jis gali būti neigiamas) ir rezultatas yra neginčijamas. Laiko analizė analizuoja dvi begalines aibes: aibės P elementus p, kur kiekvienam p elementui priskirtas pastovus laikas. Taigi, jei ε *(p)* yra maksimalus skirtumas konkrečiam p aibėje P, tada ε *(p)* yra maksimalus skirtumas tarp s_{nuo} ir s_{iki} procese *p*

$$\varepsilon(p) = \max\{\tau(s_{iki}) - \tau(s_{nuo}) \mid \tau \text{ yra pastovus užduotas laikas įvykiui } p\}$$

Jeigu grafo viršūnė *p* neturi užduoto laiko ε *(p)* bus apibrėžtas kaip -∞

Taigi maksimalus skirtumas apibrėžiamas dėsniu:

$$\varepsilon = \max\{\varepsilon(p) \mid p \in P\}.$$

Mūsų atveju modeliavimas vyksta naudojant santykinius laiko vienetus, prjektuotojas pats nusprendžia, kokie jie bus. Kiekviena posistemė veikia realiu laiku, bet mes tiksliai nežinome kiek laiko užtruks kiekvienos posistemės veikimas, todėl buvo pasirinkti santykiniai dydžiai labiausiai atspindintys realią situaciją.

Išvados. Teorinio tyrimo metu buvo ištirtas namo mobiliosios informavimo sistemos UML modelis, nuo UML modelio pereita prie laikinio Petri tinklo modelio (TPN), sudarytos Petri tinklo laikinės charakteristikos(TPN). Papildomai sudarytas SMS posistemės Petri tinklo modelis.

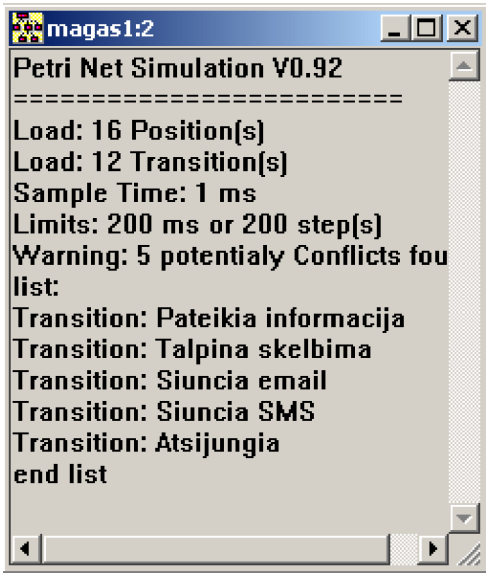
4. Eksperimentai ir tyrimai

Siekiant nustatyti sudaryto modelio tinkamumą tolesniems tyrimams, jį reikia sumodeliuoti bei parinkti sąlygas, artimas toms, kurioje veiks reali sistema. Modeliavimu paremta analizė yra dažnai naudojama kaip alternatyva analitiniams metodams. Modeliavimas retai pateikia tikslius atsakymus, tačiau įmanoma įvertinti, kokios tikslios yra padarytos prielaidos, tačiau to pilnai užtenka, atliekant modelio įvertinimą.

4.1 Sistemos Petri tinklo modelio tyrimas

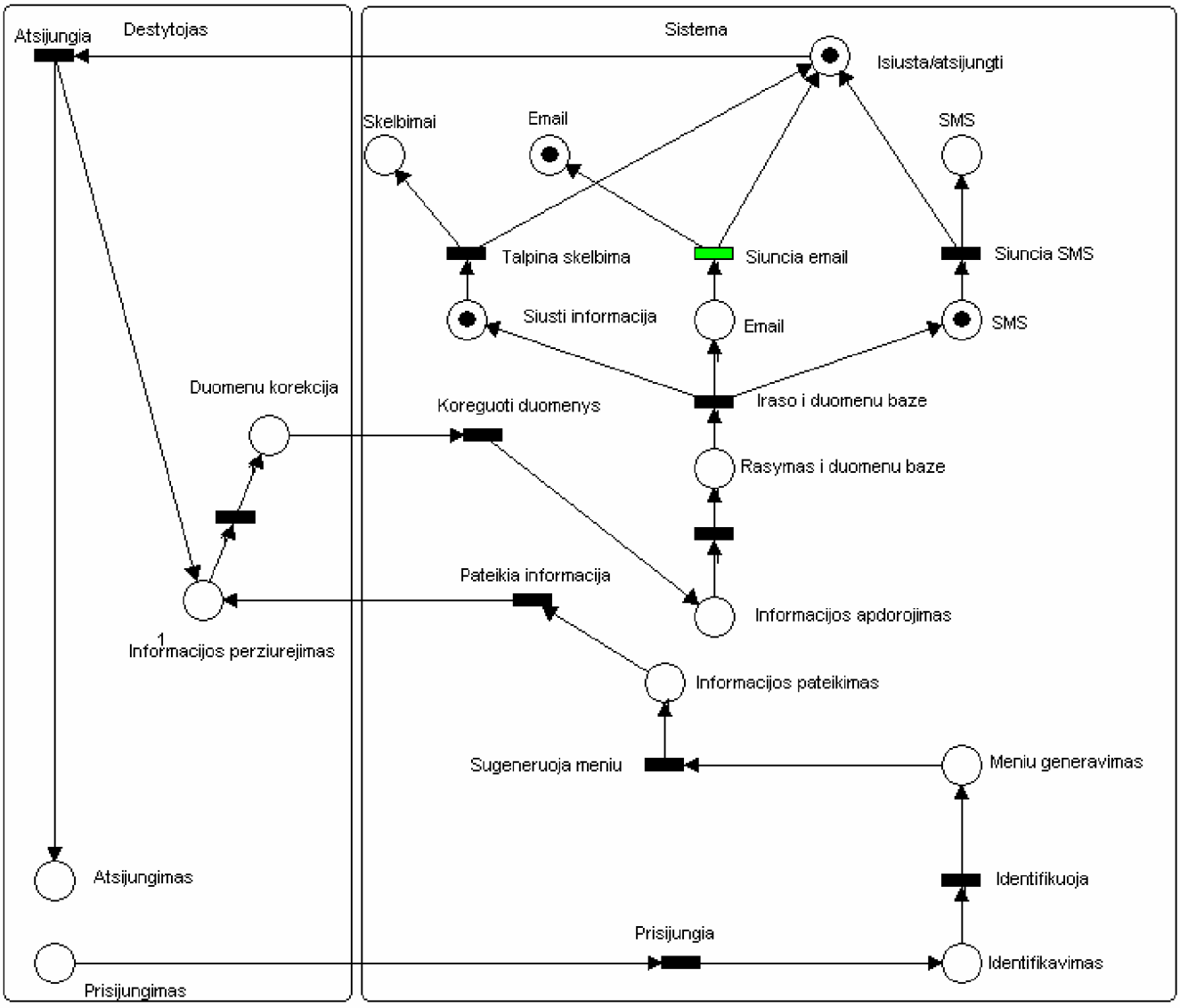
Kadangi sistemos modelis sudarytas naudojant laikinius Petri tinklus (TPN), šio modelio elgsenai tirti pasirinkau HPSim1.1 programą, skirtą TPN analizei. Ši programinė įranga taip pat pasižymi ir kitomis savybėmis: greitu modeliavimu, paprasta veiklos analize bei tinklo veikimo animacija [7].

Šios programos pagalba sudarius TPN, programa patikrina tinklą ir pateikia trumpą tinklo santrauką bei galimas klaidas, tai pavaizduota 22 paveiksle.



22 pav. HPSim 1.1 programos pateikta santrauka apie sudaryta laikinį Petri tinklą

Modeliavimas buvo vykdomas 200 žingsnių. Viso šio laiko metu modelis veikė stabiliai ir jokių nesklandumų nepastebėta.

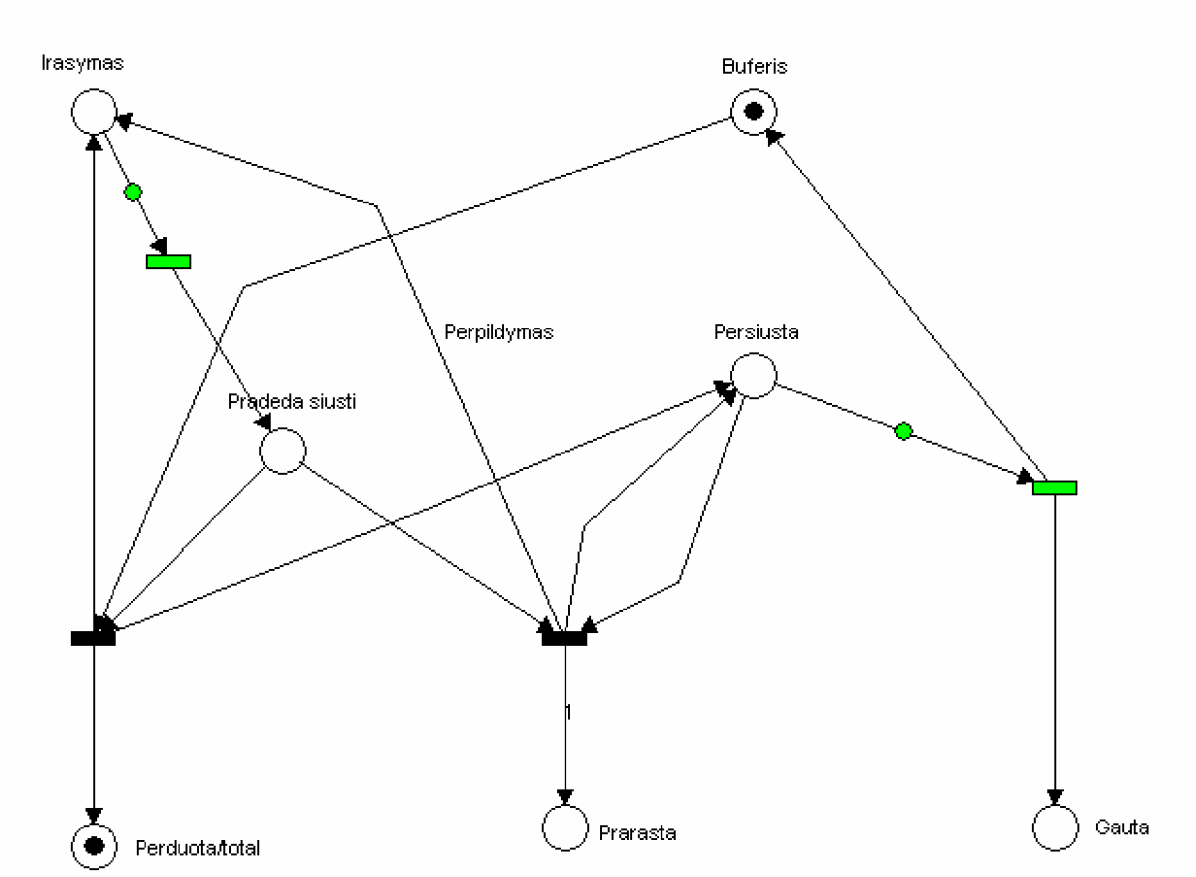


23 pav. Laikinio Petri tinklo modeliavimas

HPSim programa pasižymi eksportavimo į tekstinę bylą galimybe. Tokiu būdu visas tinklo veikimo procesas buvo eksportuotas į tekstinę bylą. Šiomis bylomis pasinaudojau kurdamas tinklo charakteristikų grafikus.

4.2 SMS posistemės Petri tinklo modelio tyrimas

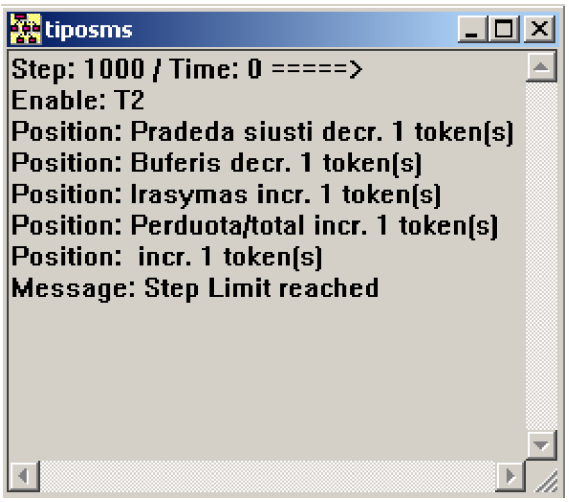
SMS posistemei tirti buvo naudojama ta pati HPSim programa. Tyrinėjome šį modelį



24 pav. SMS posistemės TPN modelis

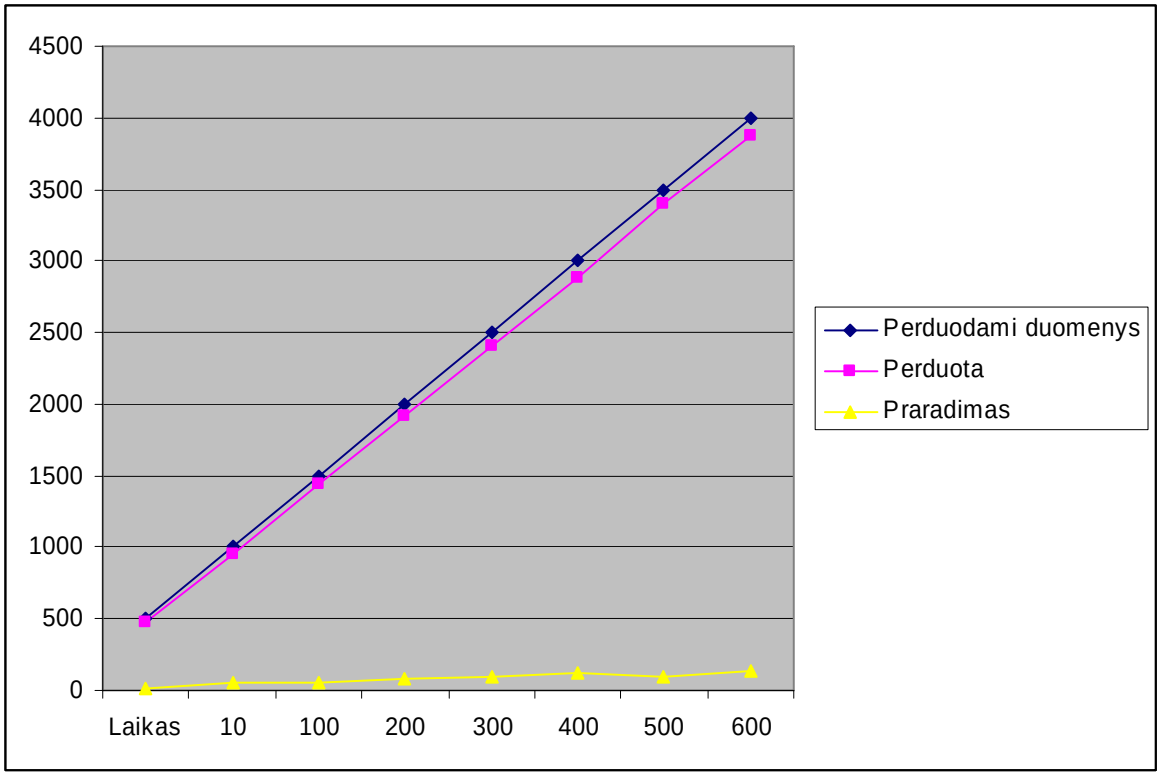
Šis modelis vaizduoja SMS posistemės veikimą. T.y. iš pradžių SMS tekstas sinčiamas į duomenų bazę, perduodamas į buferį ir tada persinčiamas vartotojui, čia modeliuojame buferio perpildymo atvejį, buferio dydį pasirenkame 2 (buferis gali talpinti du pranešimus).

SMS posistmės modelio tyrimas buvo vykdomas 1000 žingsnių. Didelių nukrypimų ir duomenų praradimų nebūdavo, nebent buvo pasirenkamas mažas buferio dydis. Duomenų dydis skaičiuojamas žetonais, vienas žetonas atitinka vieną siunčiamą SMS pranešimą.



25 pav. HPSim programos pranešimas tiriant SMS posistemės modelį

Naudojama programinė įranga gali ekpsportuoti tinklo veikimo duomenis į tekstinį failą, tuo pasinaudodami galime nubrėžti duomenų perdavimo grafiką.



26 pav. SMS posistemės duomenų perdavimo grafikas

Buvo pastebėta, kad mažinant buferio dydį didėja duomenų praradimas. Todėl projektuojant sistemą reikia numatyti, jeigu bus didelės grupės vartotojų, kuriems sistema turės pristatyti N pranešimų reikės užtikrinti pakankamą buferio dydį. Didelėms realaus laiko sistemoms duomenų praradimo modeliavimas yra labai svarbus norint iš anksto numatyti galimas sistemos spragas.

5. Išvados

Šiame darbe buvo ištirtos galimybės sudaryti formalų mobiliosios informavimo sistemos procesų valdymo modelį. Atlikus analizę buvo nustatyta, kad tinkamiausias metodas šiam modeliui sudaryti yra UML diagramos ir Petri tinklai. UML diagramos yra lengvai suprantamos paprastam vartotojui, Petri tinklai yra gerai pritaikyti formaliai analizei. Buvo sudaryti mobiliosios informavimo sistemos UML modeliai. Nuo UML modelių buvo pereita prie laikinių Petri tinklų modelių, kurie apibrėžia modelių veiklos kitimą laike. Nežinant realių mobiliosios sitemos posistemių veikimo savybių, laikui modeliuoti buvo pasirinkti santykiniai laiko vienetai, kuriuos galima keisti pagal sistemos projektuotojo nuožiūrą. Pasiūlytas modelis yra apibendrintas, tačiau yra lengvai keičiamas ir pritaikomas posistemėms su specifiniais reikalavimais.

Modelis buvo modeliuojamas ir tiriamas naudojant HPSim 1.1 programinę įrangą. Tyrimo metu buvo sumodeliuoti mobiliosios informavimo sistemos modelis, kuris veikė tiksliai pagal jų sudarytus. Tęsiant eksperimentą, buvo ištirtas SMS siuntimo posistemės veikimas. Buvo modeliuojamas SMS siuntimas esant tam tikram duomenų buferio dydžiui, vienas SMS pranešimas atitinka vieną žetoną tinkle. Buvo pastebėta, kad ribojant buferio dydį ir padidinus duomenų atsiranda tikimybė prarasti duomenis. Padidinus duomenų kiekį, sistema praradinėjo duomenis, tačiau dirbo stabiliai ir be trūkių. Todėl norint užtikrinti normalų sistemos darbą dirbant su dideliais duomenų kiekiais, reikia numatyti pakankamą buferio dydį.

Sudarytas modelis gali būti naudojamas tolimesniems tyrimams ir patobulinimams.

6. Literatūra

Kazanavičius, E. The Evaluation and Design Methodology for Real Time Systems. Iš „Informatica“ [interaktyvus]. 2004, sausis [žiūrėta 2004-04-01]. Prieiga per internetą http://www.vtex.lt/informatica/htm/INFO547.htm

Gudas, S. Uml modeliai ir papildomos diagramos. Iš Darbų bazės [interaktyvus]. 2003, kovas [žiūrėta 2001-12-12]. Prieiga per internetą: http://www.sociumas.lt/

Jensen, K. Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Volume 1. EATCS Series Monographs in Theoretical Computer Science, Springer-Verlag, Chapter 1&2 , 1996.

Sibertin-Blanc, C. CoOperative Objects: Principles, Use and Implementation. Iš University Tolouse [interaktyvus] 1998, balandis [žiūrėta 2005-01-24]. Prieiga per internetą: http://www.univ-tlse1.fr/CERISS/COOpubli.html

Durant, E. Embedded Real-Time System Considerations. Iš Endurant software [ineraktyvus] 1998, vasaris [žiūrėta 2004 10 11]. Prieiga per Internetą: http://www.edurant.com/papers/cs384embeddedrealtime.pdf

Free Petri net simulation software tool HPSim for download [interaktyvus]. 2002, spalis [žiūrėta 2004 04 04]. Prieiga per Internetą: http://www.winpesim.de

Petri Nets World: Online Services for the International Petri Nets Community [interaktyvus] 2001 kovas [žiūrėta 2003 12 10]. Prieiga per Internetą: http://www.daimi.au.dk/PetriNets/

Knudsen, C. WebCalendar Database Documentation. Source Forge [interaktyvus]. 2002, spalis [žiūrėta 2003-04-15]. Prieiga per Internetą: http://webcalendar.sourceforge.net/docs/database

Project Management [interaktyvus]. 2003 m. kovas [žiūrėta 2003-05-04]. Prieiga per Internetą: http://www.project-management-software.org/calendar/

Rational Software Corporation. Rational Unified Process 2000. Žinių bazė (komercinis produktas) [interaktyvus] 2000 vasaris, [žiūrėta 2004-03-25]. Prieiga per Internetą http://www.rational.com/products/rup/index.jsp

Miller, R. Practical UML™: A Hands-On Introduction for Developers [interaktyvus]. 2003 balandis [žiūrėta 2005-02-23]. Prieiga per Internetą http://bdn.borland.com/article/0,1410,31863,00.html

Kiauleikis, M. Vartotojo sąsaja duomenų inžinerijoje [interaktyvus]. 2001, gegužė [žiūrėta 2005-04-11]. Prieiga per internetą http://www.ifko.ktu.lt

7. Summary

This work is dedicated to the making and research of mobile information system. Following tasks were completed in order to tackle this problem:

- mobile information system processes analysis;
- uml modeling analysis;
- formal modeling methods analysis;
- composition of system model;
- system model verification.

Petri nets were chosen as formal modeling method for control system. System model was created using timed Petri nets (TPN). System models was simulated using HpSim software. Following conclusions were made after model research:

- models operated correctly;
- models can be used for systems with specific requirements;
- system model dispensed time properly;
- composed model can be easily changed and expanded.

8. Terminų ir santrumpų žodynas

TPN *(angl. Timed Petri Networks)* – laikiniai Petri tinklai

UML *(angl. Unified Modeling Language)* – projektavimo kalba

API *(angl. Application Program Interface)* - įrankių, protokolų rinkinis aplikacijų kūrimui

ASP *(angl. Active Server Pages)* - aktyvūs serverio puslapiai

COM *(angl. Component Object MOdel)* - komponento objekto modelis

GSM *(angl. Global System for Mobile Communications)* – globali mobilių komunikacijų sistema

IIS *(angl. Internet Information Server)* - interneto informacijos serveris

PHP *(angl. Hypertext Preprocessor)* - hiperteksto pirminis redaktorius (programavimo kalba)

RSA *(angl. Rivest, Shamir, Adelman)* – kūrėjų vardais pavadintas šifravimo algoritmas

SMS *(angl. Short Message Service)* - trumpųjų žinučių paslauga

SQL *(angl. Structured Query Language)* - struktūrizuotų užklausų kalba

TCP/IP *(angl. Transmission Control Protocol/Internet Protocol)* – protokolų rinkinys naudojamas susijungimams ineternete tarp kompiuterių

UTP *(angl. Unshielded Twisted Pair)* – kabelio tipas iš dviejų be šarvo laidų

WAP *(angl. Wireless Application Protocol)* – protokolas naudojamas keistis informacija tarp bevielių įrenginių