KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Andrius Uznys

UML elgsenos modelių integravimas informacinių sistemų projektavime

Magistro darbas

Vadovas
Prof.dr. L. Nemuraitė

KAUNAS, 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Andrius Uznys

UML elgsenos modelių integravimas informacinių sistemų projektavime

Magistro darbas

Vadovas
Prof.dr. L. Nemuraitė

Recenzentas
Lekt.dr. A. Janavičiūtė

Atliko
IFM 3/1 gr. stud.
A. Uznys

KAUNAS, 2009

Turinys

Integration of UML Behavioural Models in Design of Information Systems

Summary

In the paper, using of UML behavioural diagrams such as use cases, state machines, activity diagrams and interactions during Information system development is analyzed. The methodology of using these diagrams for creation of class diagrams and other specifications needed for generating code is proposed. The importance of state machines is highlighted in this methodology. MagicDraw UML tool is used for modeling and code generation.

1. Įvadas

Kuriant sudėtingas, dinamiškas informacines sistemas, sistemos elgsenos modeliavimas tampa būtinybe. Kuriant sudėtingas, dinamiškai kintančias informacines sistemas kyla grėsmė, kad realizuota informacinė sistema netenkins vartotojo poreikių, dėl to projektuojant informacines sistemas pasitelkiamos elgsenų modeliavimo priemonės tokios, kaip UML, BPMN ir BPEL. Remiantis literatūros šaltinių analize, šiuo metu geriausiai vertinamos dvi veiklos procesų modeliavimo kalbos: *Business Process Modeling Notation* (BPMN) ir UML 2 veiklos (angl. *Activity*) diagramos.

Pirmą kartą pradėjus modeliuoti informacines sistemas iškyla daug klausimų. Pirmieji klausimai dažniausiai būna susiję kokią modeliavimo kalbą reikės pasirinkti, kuri populiari modeliavimo kalba tinkamesnė projektuojamai informacinei sistemai. Pasirinktus modeliavimo kalba atitinkamai reikia pasirinkti IS projektavimo metodiką, kurį būtų aiški, suvokiama ir tenkintų specifinius projektuojamos sistemos reikalavimus. Pradiniuose antrame darbo skyriuje atlikta populiarių projektavimo procesų modeliavimo kalbų analizė. Informacinei sistemai projektuoti pasirinkta populiari procesų modeliavimo kalba UML veiklos diagramos.

Šio darbo tikslas - pateikti metodiką, kaip kurti elgsenos modeliais pagrįstas informacines sistemas. Šiam tikslui pasiekti reikės:

1. ištirti UML elgsenos modelius,

2. palyginti jų galimybes su kitomis elgseną modeliuojančių kalbų galimybėmis.

3. sudaryti elgsenos modelių integravimo metodiką, susiejant veiklos procesus ir esybių būsenas bei klasių modelius.

4. eksperimentiškai ištirti metodikos tinkamumą, pagal ją sukuriant modelį ir realizuojant informacinę sistemą.

Išnagrinėtos procesų modeliavimo kalbų šablonų modeliavimo galimybes. Kadangi IS projektavimui pasirinktos UML veiklos diagramos. Veiklos diagramos išnagrinėtos ir nagrinėtos modeliavimo kalbos galimybės. Išanalizavus nustatyta, kad UML 2 palaiko įvairius elgsenos mechanizmus – neformalius aprašus, vaizduojamus panaudojimo atvejais; automatus, vaizduojamus būsenų mašinomis; *Petri* tinklų pavidalo grafais, vaizduojamus veiklomis; dalinai sutvarkytas įvykių sekas, vaizduojamas sąveikų diagramomis. UML 2 vartotojo apibrėžta elgsena taip pat yra klasė. Elgsenos klasės, kaip visos klasės, gali turėti atributus, asociacijas, operacijas, ir net kitą elgseną. Šios klasės atitinka įprastoje IS kūrimo praktikoje naudojamas valdymo klases, kurios valdo procesus, pavyzdžiui, veiksmų eigą.

UML kalboje elgsenai modeliuoti turi sekų, veiklos ir būsenų diagramas. Šios diagramos dalinai dengiasi ir turi būti suderintos tarpusavyje ir su statiniu klasių modeliu. Kad suderinti ir susieti tarpusavyje UML diagramas sudaryta informacinių sistemų projektavimo metodika. Sukurta metodika skirta sudėtingoms sistemoms, kurios pasižymi dideliu kiekiu būsenų ir reikalingas greitas sistemos reagavimas.

UML standarte tokios galimybės nėra nagrinėjamos, todėl šio tyrimo idėja yra patobulinti IS kūrimo procesą, gaunant valdymo klases tiesiai iš elgsenos modelių. Susiejant veiklas su baigtinėmis būsenų mašinomis turime valdymo klases su metodais būtinais realizuoti veiklos funkcionalumą. Metodikos dėka yra suformuojamos modeliuojamas valdymo klasės, kurios vėliau bus panaudotos toliau projektuojant, tobulinant informacinę sistemą. UML modeliai naudojami projektuojant sistemas, bet nebuvo nagrinėjamos galimybės susieti veiklos *(*angl. *Activity)* diagramas su būsenomis. UML 2 elgsenos modeliai naudojami veiklos procesams ir duomenų srautams modeliuoti. Elgsenos modeliai leidžia apibrėžti modeliuojamos informacinės sistemos būsenas tam tikruose veiklos procesų vykdymo taškuose.

Šiame darbe siekiama pateikti metodiką, kaip kurti elgsenos modeliais pagrįstas informacines sistemas. Nagrinėjami elgsenos modeliai leidžia apibrėžti, kaip sistema turi keisti savo būsenas priklausomai nuo įvykių, kuriuos iššaukia vidiniai ar išoriniai veiksmai. Būsenoms kisti gali būti reikalingas vartotojo įsikišimas arba tai gali būti automatizuotas sistemos reagavimas į veiksmus.

IS projektavimo, pagrįsto UML elgsenos modeliais, metodas susideda iš šių pagrindinių žingsnių:

1. Identifikuojamos ir sukuriamos dalykinės srities esybių klasės;
2. Analizuojamas procesas ir sukuriamos pagrindinės proceso veiklos;
3. Identifikuojamos šių veiklų įėjimų ir išėjimų esybių būsenos ir sukuriamos pradinės esybių būsenų mašinos;
4. Analizuojami alternatyvūs scenarijai ir nustatomi alternatyvūs veiksmai, kurie vykdomi nesėkmingų rezultatų atveju, papildomos esybių būsenų mašinos;
5. Sugeneruojama proceso valdymo klasių diagrama su operacijomis, kurios vaizduoja

Metodika pritaikyta vertybinių popierių apskaitos informacinės sistemos modeliui sukurti. Remiantis metodikos įvertinimu metodika padėjo lengviau suvokti sistemos veiklos procesą. Sukurtas IS modelis padės tolimesniame kuriamos IS tobulinimo darbe. Remiantis pateiktais klausimyno atsakymais IS kūrimo metodika turi teigiamą poveikį informacinės sistemos modeliavimo procesui.

Darbo struktūra: antrame skyriuje apžvelgiami UML 2 veiklos procesų modeliavimo šablonai, analizuojami UML veiklos modeliai. Palyginama su populiaria procesų modeliavimo kalba BPMN ir palyginamos kitų populiarių kalbų galimybės realizuojant standartinius šablonus. Trečiame skyriuje aprašoma sudaryta IS kūrimo metodika. Ketvirtame skyriuje pateikta pagal trečiame skyriuje pateiktą metodiką realizuotas informacinės sistemos modelis. Sukurta metodikos įvertinimas aprašytas ketvirtame skyriuje. Penktame skyriuje pateikiamos išvados, o šeštame literatūros analizė.

2. Veiklos procesų modeliavimo kalbų analizė

2.1. Veiklos procesų modeliavimo šablonai

Wil van der Aalst, Arthur ter Hofstede, Bartek Kiepuszewski, ir Alistair Barros tiriamojo darbo rezultatas buvo identifikuota šablonų aibė (21 šablonas), kuri apibūdina verslo procesų elgesį. Šie šablonai naudojami įvertinti veiklos procesų modeliavimo kalbų, modeliavimo ir vykdymo įrangos potencialias galimybes. Šablonai sudėtingumas yra nuo labai paprastų iki labai sudėtingų. Šiame darbe šablonai bus naudojami UML elgsenos modeliams palyginti su kitų modeliavimo kalbų galimybėmis. Bus lyginamos galimybės pritaikyti informacinių sistemų projektavimui.

2.1.1. Baziniai valdymo šablonai

Pirmi penki šablonai yra ganėtinai paprasti procesų elgsenos pavyzdžiai. Šablonai nusako pagrindinius verslo proceso modelius.

2.1.2. Seka (angl. Sequence)

Sekos šablonas yra surūšiuotas veiklų sąrašas, viena veikla prasideda, kai prieš tai buvusi veikla baigia darbą (WfMC)[1].

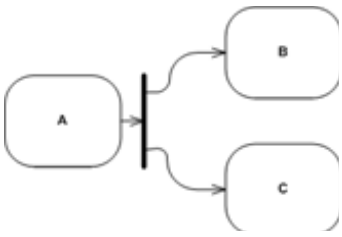
UML veiklos diagrama apibrėžia šabloną kaip seriją veiklų sujungtų tarpusavyje valdymo srautais. Valdymo srautų kryptis apibrėžia sekos tvarką.

2.1 pav. Seka

2.1.3. Lygiagretus išsišakojimas (angl. Fork Node)

Lygiagretus išsišakojimo šablonas yra mechanizmas, kuris leidžia veiklos diagramoms būti atliekant vienu metu. Proceso vienas kelias yra išskaidomas į du ar daugiau kelių, taip kad viena ar daugiau veiklų būtų pradėtos tuo pačiu metu. WfMC apibrėžia šią elgseną kaip „AND-split“.

[1] The Workflow Management Coalition Terminology & Glossary (1999)

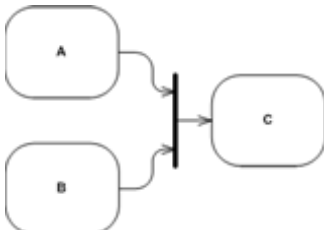


2.2 pav. Lygiagretus išsišakojimas

UML veiklos (angl. A*ctivity*) diagrama naudoja išsišakojimo mazgą (angl. *Fork Node*) sukurti lygiagrečių kelius. Paveiksle pavaizduota juosta nurodo kas visi išeinantys valdymo srautai iš juostos sukurs eilę lygiagrečių srautų. Požymis bus sugeneruotas kiekvienam išeinančiam sekos srautui (angl. *Sequence Flow*). Galutinės veiklos kiekvienam iš srautų bus galima pradėti tuo pačiu metu.

2.1.4. Sinchronizacija

Sinchronizacijos šablonas jungia kelius, kurie buvo sukurti lygiagretaus išsišakojimo šablono. Jungiamos veiklos tūri būti baigusios darbą, tik tada procesas toliau tęs veiklą. Tai yra sinchronizacija lygiagrečių kelių. WfMC apibrėžia tokią elgseną kaip „AND-Join“.



2.3 pav. Sinchronizacija

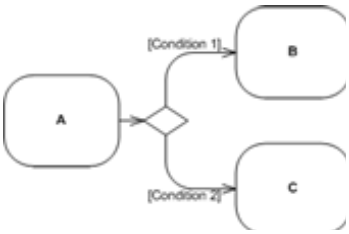
UML veiklos diagramoje naudojamas jungimo mazgas, kuris apjungia seką lygiagrečias veiklas (2.3 paveikslas). Juosta pavaizduota paveiksle atrodo identiškai išsišakojimo mazgui. Šis žymėjimas gali atlikti abi funkcijas. Sujungimo juosta nurodo kad simboliai (angl. *Tokens*) turi atkeliauti iš visų į juostą ateinančių valdymo (angl. *Control*) srautų. Tada vienas požymis toliau eis iš juostos (angl. *Bar*).

2.1.5. Išskirtinis pasirinkimas

Išskirtinio pasirinkimo šablonas yra proceso vieta, kur srautai yra išdalinti į du ar daugiau vienetinius kelius. Šablonas vienpusiškumas yra dėl to kad tik vienas iš daugelio kelių gali būti pasirinktas, kuriuo procesas toliau tęsis. WfMC apibrėžia šią elgseną kaip arba išsišakojimą.

Veiklos diagramos atveju naudojamas kūrimui sprendimo mazgas, kurio pagalba realizuojami alternatyvūs keliai. Išeinantys valdymo srautai iš sprendimo mazgo turės logines

išraiškas, kurių pagalba bus nustatyta kuris valdymo srautas bus naudojamas tęsti procesui. Kai simbolis (angl. *token*) pasieks mazgą, reikšmės bus įvertintos (neapibrėžta tvarka) ir reikšmė kuri yra nustatyta teisinga (angl. *true*) atitinkamas valdymo srautas bus pasirinktas ir simbolis toliau keliaus šiuo keliu. Tik vienas simbolis išeis iš sprendimo mazgo kiekvienam įeinančiam simboliui kuris atkeliauja į mazgą.

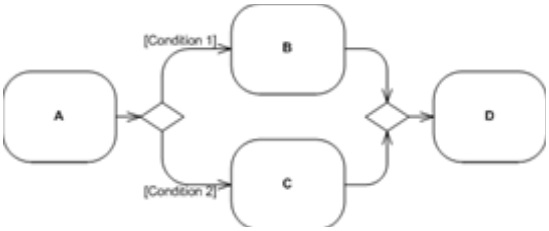


2.4 pav. Vienintelis pasirinkimas

2.1.6. Paprastas jungimas (angl. Exclusive Choice)

Paprasto jungimo šablonas yra vieta procese, kur rinkinys alternatyvių kelių yra apjungiami į vieną kelią. WfMC apibrėžia šią elgseną kaip „OR-Join“.

UML veiklos diagramose naudojamas jungimo mazgą sujungti alternatyvius kelius. Rombas parodytas paveikslėlyje yra identiškas sprendimo mazgui. Rombu simbolis gali atlikti abi funkcijas. Kai simbolis pasiekia mazgą jis iš karto keliaus į išeinantį srautą. Jungimo mazgas yra nebūtinas ir įeinančius valdymo srautus galima palikti veikoje, bet tai nėra tinkamas metodas modeliuojant veiklų elgsenas.



2.5 pav. paprastas jungimas

2.1.7. Sudėtingesni išsišakojimo ir sinchronizacijos šablonai

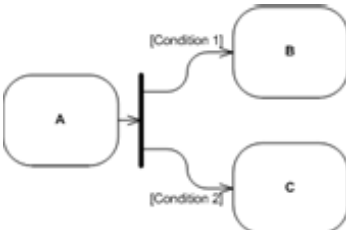
Žemiau pateikti šablonai apibrėžia sudėtingesnius modeliavimo atvejus, kaip reikia išdalinti ir sujungti verslo proceso srautus.

2.1.8. Daugialypis pasirinkimas (angl. Multiple Choice)

Daugialypio pasirinkimo šablonas skiriasi nuo vienintelio pasirinkimo šablono (angl. *Exclusive Choice*), kad daugialypio pasirinkimo šablonas leidžia nuo vieno iki visų veikloje esančių alternatyvių kelių pasirinkimo vykdymo metu. Daugialypio pasirinkimo šablonas gali

leisti nepasirinkti nei vieno kelio, bet tai gali būti neleistina situacija, kai proceso eiga sustoja netikėtai.

Veiklos diagramoje naudojamas išsišakojimo mazgas, kur išeinantys valdymo srautai turi apsaugas (angl. *Guards*), kurių pagalba sukuriamas daugialypio pasirinkimo šablonas. Išsišakojantis mazgas sukuria lygiagrečių takus. Kai simbolis atvyksta į mazgą, jis bus išskaidytas į išeinančius valdymo srautus kur apsaugos tenkinamos. Teoriškai išsišakojimo mazgas yra nereikalingas ir apsaugos gali būti uždėtos ant išeinančių iš veiklos valdymo srautų, bet tai nėra rekomenduojama.



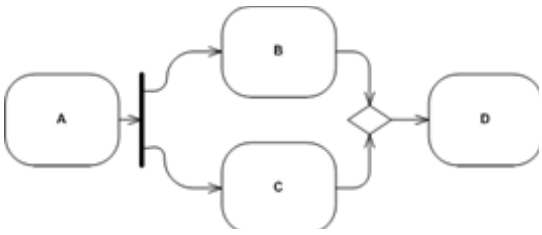
2.6 pav. Daugialypis pasirinkimas

2.1.9. Daugelio srautų jungimas (angl. Multiple Merge)

Daugialypio srautų jungimo šablonas yra vieta procese, kai yra keliai jungiami, bet be jokio simbolių srautų valdymo. Kai simbolis pasieks veiklą, veikla bus pradedama. Jei yra keleto kelių apjungimas veikloje be valdymo simbolių yra įmanoma, kad veikla bus pradės veikti tuo pačiu metu kai takai yra sujungiami. Simboliai toliau vyks nepriklausomai per likusį procesą (jei liko).

Daugelio proceso apjungimo šablonas skiriasi nuo sinchronizacijos ar paprasto jungimo kad šie šablonai skirti veiklai kuri bus paleidžiama tik kartą. Dėl to daugelio procesų apjungimo šablonas yra ganėtinai sunkiai suvokiamas. Sunku suvokti proceso elgseną, kai jos proceso dalis yra paprasčiausiai kopijuojama. Dėl to šablonas skirtas specifinėms situacijoms.

Veiklos diagramose naudojamas jungimo mazgas kelių jungimui. Rombas pažymėtas figūroje atrodo taip kaip sprendimo mazgas, ir iš tiesų rombu pažymėtas simbolis gali atlikti abi funkcijas. Kai simbolis pasieks mazgą judės toliau į išeinantį valdymo srautą. Jungimo mazgas yra tik grafinis jungimo žymėjimas dėl to nesustabdo kitų simbolių, kurie tuo metų gali pasiekti mazgą. Jei pradinis kelių šaltinis pasiekia jungimo mazgą, ir tas mazgas yra išsišakojimo mazgas (kaip parodyta 2.7 iliustracijoje), tada veikla kuri pasiekia sujungimo mazgą bus paleista keletą kartų. Techniškai apjungimo mazgas yra nereikalingas ir įeinantys srautai yra leidžiami veiklai, bet tai nėra laikoma rekomenduojama .

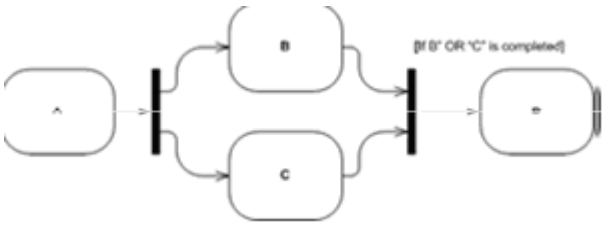


2.7 pav. Daugelio srautų apjungimas

2.1.10.Diskriminatorius

Diskriminatoriaus šablonas yra vienas iš būdų sujungti kelius kurie buvo sugeneruoti naudojant lygiagretaus išsišakojimo šabloną. Šablono pagalba lygiagretūs srautai yra sujungiami. Tai skiriasi nuo sinchronizacijos šablono, kuriame procesas toliau tęs veiklą, kai pirmas simbolis pasieks diskriminatorių. Diskriminatoriaus atveju visi likę simboliai kurie buvo sukurti lygiagrečiame išsišakojime pasiekę diskriminatorių bus blokuojami.

UML veiklos diagrama naudoja jungimo mazgą (angl. *Join*) diskriminatoriaus elgsenos realizacijai. Jungimo mazgas naudojamas sujungti kelius. Apsauga gali būti nustatyta sujungimo mazgui taip tikrinamos bet kokios pasibaigusios pradinės veiklos būsenos. Kai pirmas simbolis pasieks apjungimo mazgą, simbolis toliau judės link išeinančio srauto. Kiti simboliai atvykstantys į mazgą bus sustabdyti.



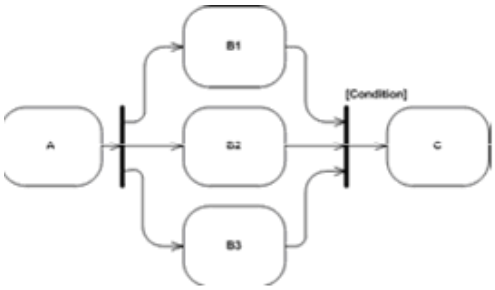
2.8 pav. Diskriminatorius

2.1.11. N iš M jungimas

N iš M jungimo šablonas yra tarpinis šablonas tarp sinchronizacijos ir diskriminatoriaus šablonų. Šiuo atveju vietoj vieno ar kelių įeinančių lygiagrečių simbolių, kurie turi judėti toliau už apjungimo mazgo, N ir M apjungimas leidžia modeliuotojui apibrėžti kiek įeinančių srautų simbolių yra būtina praleisti. Visi likę simboliai bus sustabdyti (kaip diskriminatoriuje).

Veiklos diagramoje naudojamas jungimo mazgą lygiagrečių kelių sujungimui. Kaip ir su sinchronizacijos šablonu, sujungimo juosta (angl. *join bar*) reiškia, kad simboliai turi pasiekti iš visų įeinančių į juostą srautų. Apsauga pridedama prie jungimo. Apsaugos pagalba

nustatome kiek simbolių yra reikalingi. Likę papildomi simboliai kurie pasieks sujungimo mazgą bus blokuojami.

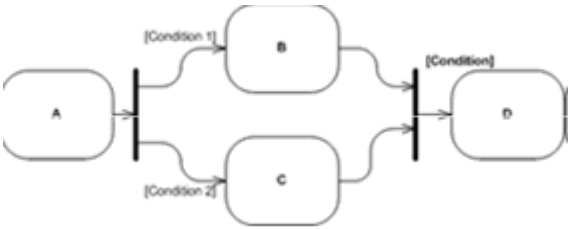


2.9 pav. N iš M apjungimas

2.1.12. Sinchronizuojamas jungimas

Šis šablonas yra vienas iš sinchronizavimo šablonų. Šablono tikslas yra sinchronizuoti simbolius iš visų lygiagrečių kelių kurie pasiekia veiklą. Sudėtingumas atsiranda dėl to kad nežinoma kiek simbolių atvyks. Ši situacija sukuriama daugelio veiklų pasirinkimo šablone. Sinchronizuojantis sujungimo šablonas turi nustatyti kiek simbolių buvo sugeneruota iki tol, tada sinchronizuoti šiuos simbolius, bet nelaukti jokių kitų simbolių.

UML veiklos diagrama naudoja jungimo mazgą sinchronizuojamo jungimo šablonui. Sujungimo mazgas su veiklos sąlyga, kuri kontroliuoja kiek simbolių turi atvykti iš atvykstančio valdymo srauto, prieš tai kol simboliai bus vykdomi per išeinantį valdymo srautą.

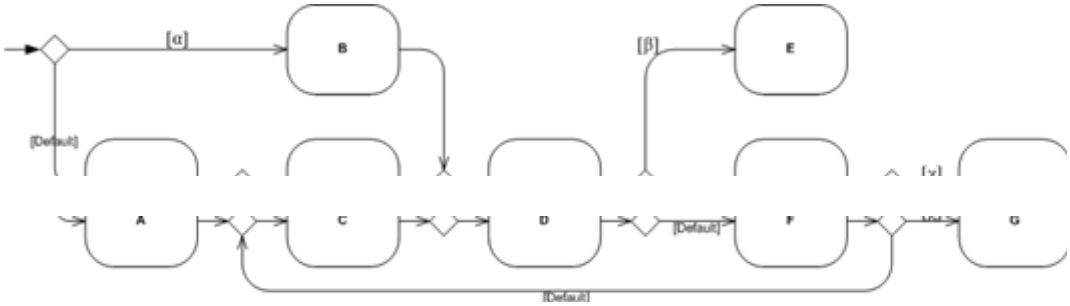


2.10 pav. Sinchronizuojantis jungimas

2.1.13. Pasirenkamieji ciklai (angl. Arbitrary Cycles)

Pasirenkamųjų ciklų šablonas yra mechanizmas leidžiantis daliai proceso būti pakartotai – tai ciklas. Šis šablonas leidžia kurti ciklus kurie yra nenuoseklus ir neblokinės struktūros. Ciklo segmentas procese gali turėti daugiau nei vieną įėjimo ar įėjimo tašką. Šablonas yra svarbus dėl lengvo skaitomumo to kad galima sutalpinti sudėtingas ciklų situacijas vienoje diagramoje. Žymėjimai kurie skirti blokiniams struktūriniams ciklams neleistų pavaizduoti viso proceso vienoje diagramoje neleistų vaizduoti proceso lygio ar vaizduojant išskirstytų elgseną ne itin suvokiamai.

Veiklos diagramose galima sukurti pasirenkamąjį ciklo šabloną sujungiant valdymo srautus su prieš tai buvusiomis veiklomis.

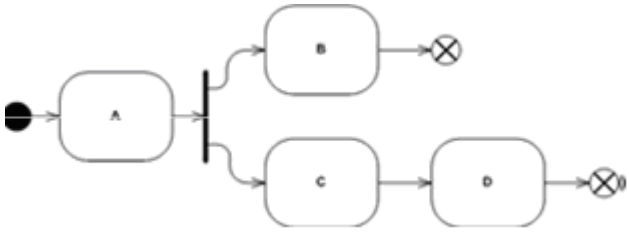


2.11 pav. Pasirenkamieji ciklai

2.1.14. Visiškas nutraukimas

Šis šablonas leidžia sukurti specifinį proceso kelią, kuris pasibaigtų kartu su likusiais lygiagrečiais keliais. Tai prieštarauja tam tikriems žymėjimams, kurie reikalauja, kad visas procesas pasibaigtų, kai bet koks mazgas yra pasiektas.

Veiklos diagramose srauto pabaigos (angl. *flow final*) mazgas pažymėtas X veiklos diagramoje yra indikatorius, kad tam tikras kelias baigtas ir tai yra visiško nutraukimo šablonas. Veiklos pabaigos mazgas (apskritimas su mažesniu užpildytu apskritimu) reikštų viso proceso sustabdymą, net jei yra veiklos kurios nebuvo pradėtos ar yra vis dar aktyvios.



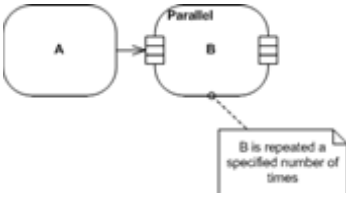
2.12 pav. Visiškas nutraukimas

2.1.15. Procesai paleisti vienu metu su prioritetiniu projektavimu ir laiko žinojimu

Šis šablonas apibrėžia kaip veikla gali būti paleista lygiagrečiai ir žinomą kiekį kartų. Projektuotojas žino kiek kartų veikla bus atlikta ir nustato tą skaičių procesų modelyje.

Šablonas neapibrėžia tiksliai kas nutinka kai veiklos egzemplioriai paleisti. Jei procesai veiklos egzemplioriai ir jų simboliai turi būti sinchronizuoti prieš paleidimą, tada procesų paleidimui reikia panaudoti sinchronizacijos šabloną. Jei procesas gali tęstis nepriklausomai nuo kitų proceso veiklos egzempliorių (leidžiamas nepriklausomas simbolių judėjimas), tokiu atveju modifikuotas šablonas su daugelio procesų paleidimo be prioritetų turėtų būti naudojamas.

UML veiklos diagramos naudoja išplėtimo sritį aplink veiklą. Taip sukuriamos veiklos kopijų. Tai yra nurodoma ant veiklos esančių įvesčių ir išvesčių išplėtimo mazgais (mažais kvadratais). Vienalaikiškumo (angl. *Concurrency*) atributas nustatomas į lygiagrečiai, taip modelyje sukuriama lygiagrečiai veikiančios veiklos egzemplioriai. Kiekis elementų regiono įėjimo nustato kiek turinio kopijų bus paleista. Elgsenos šablonui realizuoti elementų kiekis nustatomas statinis.



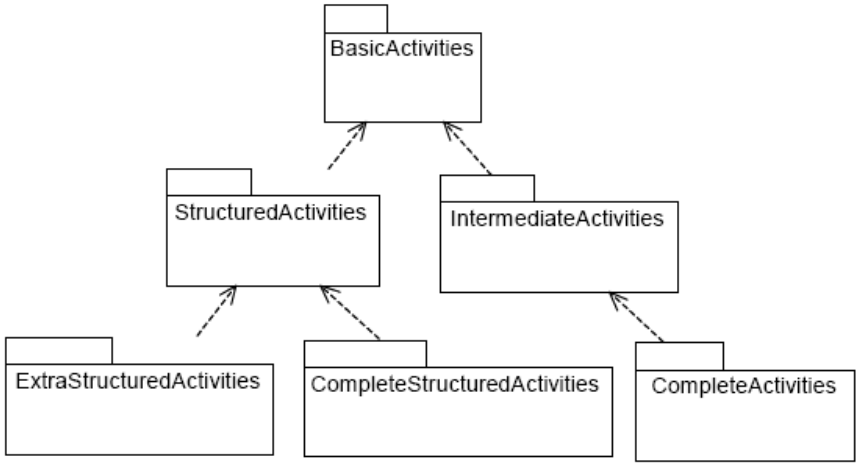
2.13 pav. Prioritetinių procesų modelis su laiko žinojimu

2.2. UML veiklos modelių analizė

2.2.1. UML veiklos modelių ryšiai

UML elgsenos modeliai yra pagrįsti bendru modeliu, kuris sieja tarpusavyje veiksmus, veiklas, būsenas, įvykius, klases, kviečiamas operacijas ir t. t. Šie modeliai yra apibrėžti nepriklausomai nuo programinės įrangos, tačiau kai kurios ypatybės yra tinkamesnės vienoms taikymo sritims negu kitoms. Pavyzdžiui, veiksmai skirti iškviesti klaidas bus dažniau naudojami programuotojų, tuo tarpu projektuotojai labiau naudos srautų modelius. Kuriami modeliai yra pertekliniai, nes yra skirti įvairioms vartotojų grupėms. Tačiau šį pertekliškumą kompensuoja efektyvi komunikacija.

Veiklų (angl. *Activity*) modelis yra labiau detalizuotas negu kiti elgsenos modeliai, ir turi sudėtingesnes priklausomybes tarp paketų, kaip parodyta 2.14 paveiksle.



2.14 pav. Veiklos paketų priklausomybės

Ypatybės bendrai tarp programų, tokios kaip abstraktus veiksmų modelis, kontrolės/duomenų srautai, ir įvestis/išvestis, yra pagrindas (angl. *Root*) (pagrindiniai veiksmai). Nuo ten, priklausomybės išsišakoja dviem kryptimis, programinės įrangos modeliavimui (struktūrinės veiklos), ir kitos bendram proceso modeliavimui (tarpinės ir užbaigimo veiklos). Struktūrinės veiklos įveda konstrukcijas, tam, kad modeliuotų sąlygos sakinius, kintamuosius, try/catch, ir taip toliau. Tarpinės (angl. *Intermediate*) ir užbaigimo veiklos įveda aiškų lygiagretumą, skirsnius (angl. Partitions), srautais grindžiamų išimčių apdorojimą ir daugelį kitų smulkių konstrukcijų skirtų srautų valdymui. Tai yra stačiakampės šakos, tokiu būdu jos gali būti apjungtos. Pavyzdžiui, struktūrinės veiklos gali persidengti su tarpinėmis veiklomis palaikydamos aiškų lygiagretumą ir struktūrizuotas sąlygos tuo pačiu laiku. Taip užtikrinamas įvairių paketų funkcionalumas.

Plataus programų diapazono palaikymas užtikrinti reikia kelių tipų žymėjimų. Pavyzdžiui, programuotojai yra linkę naudoti tekstinius žymėjimus, tuo metu srities ekspertai teikia pirmenybę grafiniams žymėjimams (angl. *Notation*). UML atkreipė į tai dėmesį ir sukūrė saugyklos modelį (repository). Saugykloje saugomos specifikacijas, kurios gali būti papildytos naujais žymėjimais. Programuotojai galės naudoti tekstinę sintaksę veiklos modelių kūrimui ir skaitymui veiklos modelių esančių saugykloje, projektuotojai gali naudoti grafinį žymėjimą. UML modelių saugykla yra komunikacijos terpė tarp skirtingų žymėjimų tipų, ir yra šaltinis lengvai adresuojamų (angl. *Directable*) kompiliatorių skirtų kelioms platformoms [4].

2.2.2. Sekos modeliai ir semantika

Elgesio modeliai nusako kada elgsena turi prasidėti ir kokios elgsenos įvestys yra. UML 2 veiklos modeliai laikosi standartinio valdymo ir duomenų srauto metodų metodo, pradeda subelgsenas tada kai kitos elgsenos baigiamos vykdyti ir elgsenos įvestys yra pasiekiamos. Tai tipiški kontrolės ir duomenų srauto modeliams. Kad suvokti ir įsivaizduoti veikimą reikia sekti diagramoje linijas nuo pradžios iki galo, valdymo ir duomenų judėjimas nusakomas linijomis. Todėl simbolių srauto semantika, įkvėpta Petri tinklų, yra labiausiai intuityvi vartotojams. Srautų semantikoje simbolis (angl. „Token“) yra terminas skirtas apibrėžti valdymo ir duomenų vertėms.

UML 2 taiko tą pačią elgsenos semantiką skirtą modeliuoti virtualias mašinas kaip ir UML 1.x. Tai įgalina vartotojus ir kūrėjus matyti modelių veikimą ir jų veiklos rezultatą. UML 2 veiklos diagramos apibrėžia virtualią mašiną, kuri paremta valdymo ir duomenų pasiskirstymu grafo viršūnėse, kurios yra sujungtos lankais. Kiekviena viršūnė ir lankas apibrėžia, kaip valdymo ir duomenų srautai juda. Modeliuojant grafo elgseną simbolių (angl.

Token) judėjimo taisyklės gali būti sujungtos. Valdymo ir duomenų judėjimo taisyklės yra skirtos numatyti veiklos padarinius, kurie bus, kai elgsenos bus paleistos su kitokiomis įvestimis. Modeliai nevaržo realizacijos. Taisyklės nenurodo kad veiklos modeliai turi būti įgyvendinti kaip virtualios mašinos, atitinkančios nustatytą schemą. Vienintelė sąlyga yra, kad sukurtos IS veikimo metu, modeliuojamos virtualios mašinos poveikiai vyktų realioje sistemoje.

Veiklos modelis yra modeliuojamas su keliais pakeitimais, kurie leidžia realizacijai pasirinkti alternatyvią veikimo elgseną, neįrašant tų pasirinkčių modelyje. Pakeitimai priverčia modelį vienoje realizacijoje veikti kitaip lyginant su kita to paties modelio realizacija. Veiklos vykdymo pakitimai yra daugiausia modeliuojami kaip atributo vertės vartotojo modelyje. Tai reiškia, kad jie yra kontroliuojami vartotojo ir yra perduoti kitiems vartotojams.

2.2.3. Veiklos viršūnės ir lankai

UML 2 veiklos turi viršūnes, sujungtas lankais, kurių pagalba suformuojamas pilnas srautų grafas. Viršūnių valdomos valdymo ir duomenų vertės juda lankais, nukreipiamos į kitas viršūnes, arba laikinai patalpinamos saugoti vertes. Yra trys viršūnių rūšys veiklos modeliuose:

1. Veiksmo (angl. *Action*) viršūnės dirba su valdymo ir duomenų vertėmis, kurias jos gauna, ir perduoda valdymą ir duomenis kitoms veikloms.

2. Valdymo viršūnės parskirsto valdymą ir duomenis grafe. Valdymo viršūnės turi konstrukcijas skirtas išrinkimui iš alternatyvių srautų (sprendimo punktai), judėjimui daugialypiuose srautuose lygiagrečiai (šakose), ir taip toliau.

3. Objekto viršūnės laiko duomenis laikinai, kol simboliai laukia judėjimo grafu.

2.15 iliustracija rodo žymėjimus kai kurioms iš veiklos viršūnių. Valdymo viršūnės valdo duomenų srautą, srautą grafe. Objekto viršūnės gali saugoti objektus, ir duomenis.

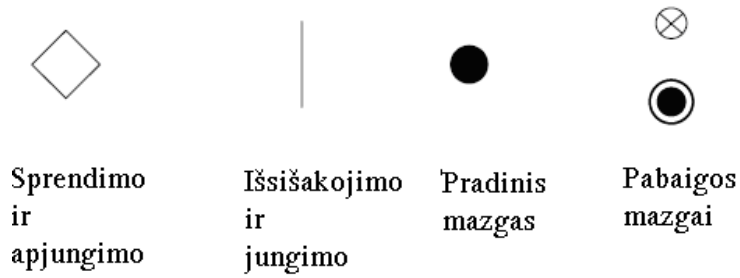
Veiklos viršūnės yra sujungiami dviejų kryptinių lankų rūšių:

1. Valdymo srauto lankai jungia veiksmus kurie rodo, kad veiksmas lanko tikslo gale (strėlės smaigalys) negali prasidėti iki pradinio veiksmo pabaigos. Tiktai valdymo simboliai gali pereiti per valdymo srauto lankus.

2. Objekto srauto lankai jungia objektų viršūnes, kurie suteikia įvestis veiksmams. Tiktai objektai ir duomenys gali pereiti per objekto srauto lankus.

Veiklos (angl. *Action*) viršūnė:

Kontrolės viršūnė:

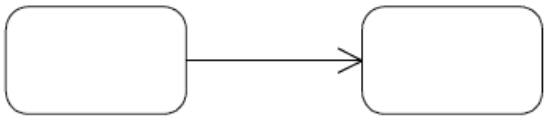


Objektų viršūnės (naudojami keliais būdais):

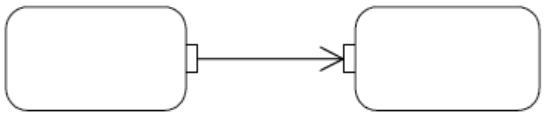
2.15 pav. Veiklos viršūnės

2.16 pav. rodo lankų žymėjimus. Valdymo ir objekto srautų lankai yra atskiriami pagal naudojimą. Valdymo lankai apjungia veiksmus tiesiogiai, kai objekto srautų lankai sujungia veiksmų įėjimus ir išėjimus.

Valdymo srauto lankas:



Objektų srautų lankas:

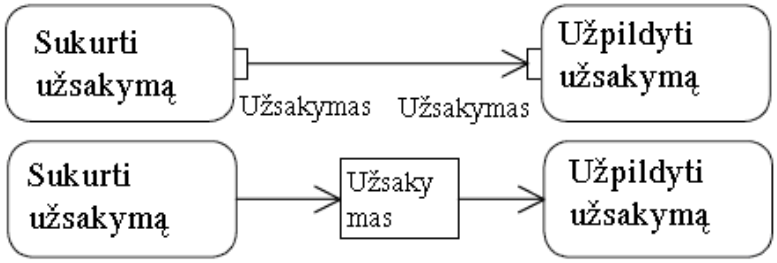


2.16 pav. Veiklos briaunos

2.2.4. Veiksmai

Veiklos modeliai koordinuoja veiksmus. Veiksmai gali iškviesti vartotojo apibrėžtą elgseną ir kitas veiklas. Visi veiksmai yra apibrėžti iš anksto. Pavyzdžiui, UML 2 turi veiksmus objektų sukūrimui, atributų verčių nustatymui, objektų sujungimui, ir vartotojo apibrėžtų elgsenų iškvietimui. Veiksmai gali turėti įvestis ir išvestis, kurie yra vadinamos jungtimis (pins), kurios yra prijungtos prie objekto srauto lankų, kurie parodo, kaip vertės juda veikloje. Galima pradėti vykdyti tik kai visos įvestys veiksmui pasiekiamos.

2.17 pav. rodo pavyzdį veiksmo, kuriančio naują užsakymo klasę, tada kitas veiksmas, iškviečia elgseną, kuri užpildo objektą. Objekto sukūrimo veiksmas sukuria tuščią užsakymą, kurį „Užpildyti užsakymą“ užpildo. Veiksmo viršūnės yra užrašytos sutartiniais ženklais - stačiakampiais užapvalintais kampais. 2.17 pav. viršuje, maži stačiakampiai, pridėti prie veiksmų, yra įvesčių ir išvesčių jungtys (angl. *Pins*). Tipas objekto, priimto kaip įvestis ar išvestis, vaizduojamas kaip papuošimas. Žymėjimas 2.17 pav. apačioje gali būti panaudotas, kai tipai įvesties ir išvesties yra tie patys. Jungtys yra objekto viršūnės tipas, tokiu būdu jungtys laikinai saugo srauto duomenų vertes.

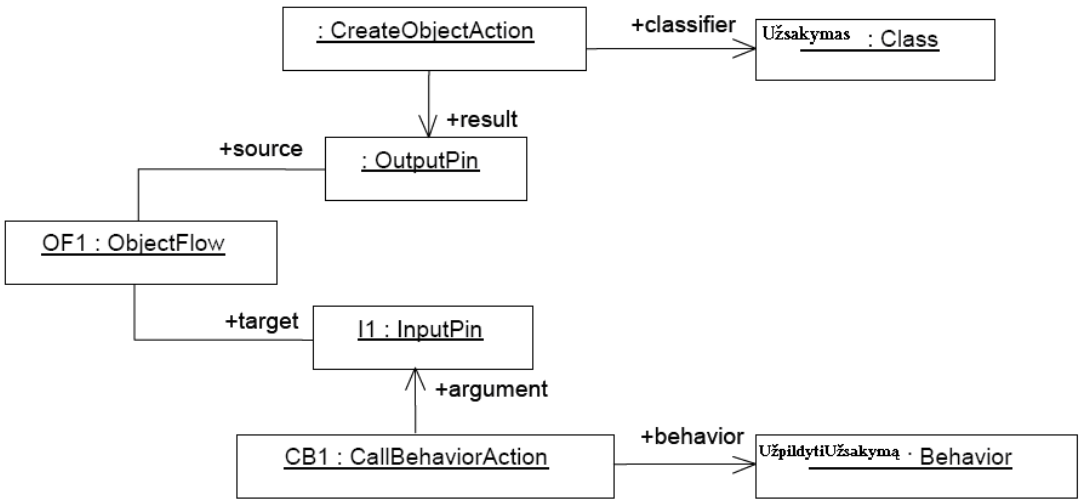


2.17 pav. Pavyzdiniai veiksmai ir objekto srautų lankai

Kaip jau buvo minėta anksčiau nebūtina naudoti žymėjimus parodytus 2.17 iliustracijoje. Programuotojai labiau bus linkę naudoti tekstinius žymėjimus tokius kaip:

```
Užsakymas o;
o = new Užsakymas;
UžpildytiUžsakym (o);
```

Saugyklos modelis, apibrėžtas UML 2, yra toks pat anksčiau minėtam žymėjimui, kuris parodytas 2.17 paveiksle, laikoma, kad tekstinės kalbos kintamieji yra modeliuojami kaip duomenų srautai. Kiekvienas saugyklos elementas yra egzempliorius meta-klasės, aprašytos UML specifikacijoje, kurios vardas yra į dešinę nuo dvitaškio. Saugyklos vardas yra kairėje dvitaškio pusėje. Modelis rodo CREATEOBJECTACTION su išėjimu kontaktu, perduodančiu jo vertę į CALLBEHAVIORACTION įvesties kontaktą. CREATEOBJECTACTION yra sujungtas su vartotojų klase tai iškviečia (Užsakymas), ir CALLBEHAVIORACTION yra sujungtas su vartotojo apibrėžtu elgsena, kuri iškviečia (UžpildytiUžsakymą).



2.18 pav. Saugyklos modelis

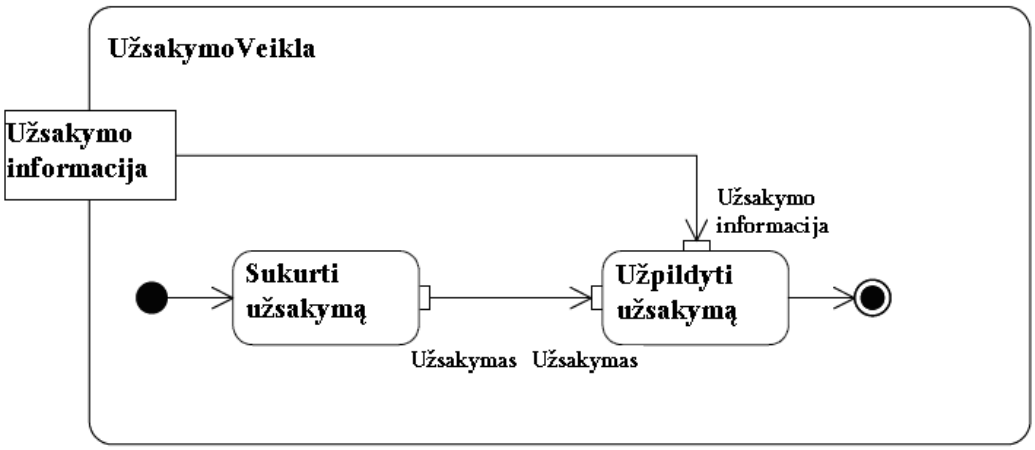
„UžpildytiUžsakymą“ elgsena gali būti iškviestas daugelyje veiklos diagramų, arba daug kartų toje pačioje veiklos modelyje, bet kiekvienam iškvietimui bus naudojamas atskiras CALLBEHAVIORACTION saugyklos egzempliorius. „UžpildytiUžsakymą“ naudojamas skirtinguose srautuose, ar skirtinguose to paties srauto taškuose, ir kiekvienas paleidimas gali turėti skirtingus veiksmus prieš ir po to. Pavyzdžiui, 2.18 pav. turi CREATEOBJECTACTION prieš „UžpildytiUžsakymą“, tuo tarpu kitas srautas šios situacijos galėjo neturėti.

Veiksmai yra taip pat gali būti naudojami kelis kartus, bet tai modeliuojama skirtingu būdu nei vartotojo apibrėžtos elgsenos. Kiekvienas veiksmas sraute yra nauja klasė iš UML meta-modelio. Pavyzdžiui, jei elgsena turi savyje dvi objektų kūrimo veiksmo viršūnes, tai vartotojo objektų modelis turi du pavyzdžius CREATEOBJECTACTION klasės iš UML meta-modelio, ir atskiras jungtis kiekvienam pavyzdžiui. Daugkartinis panaudojimas yra pasiekiamas naudojant kelis egzempliorius (angl. *Instances*) to paties UML meta-klasės veiksmų.

2.2.5. Veiklos (angl. Activity)

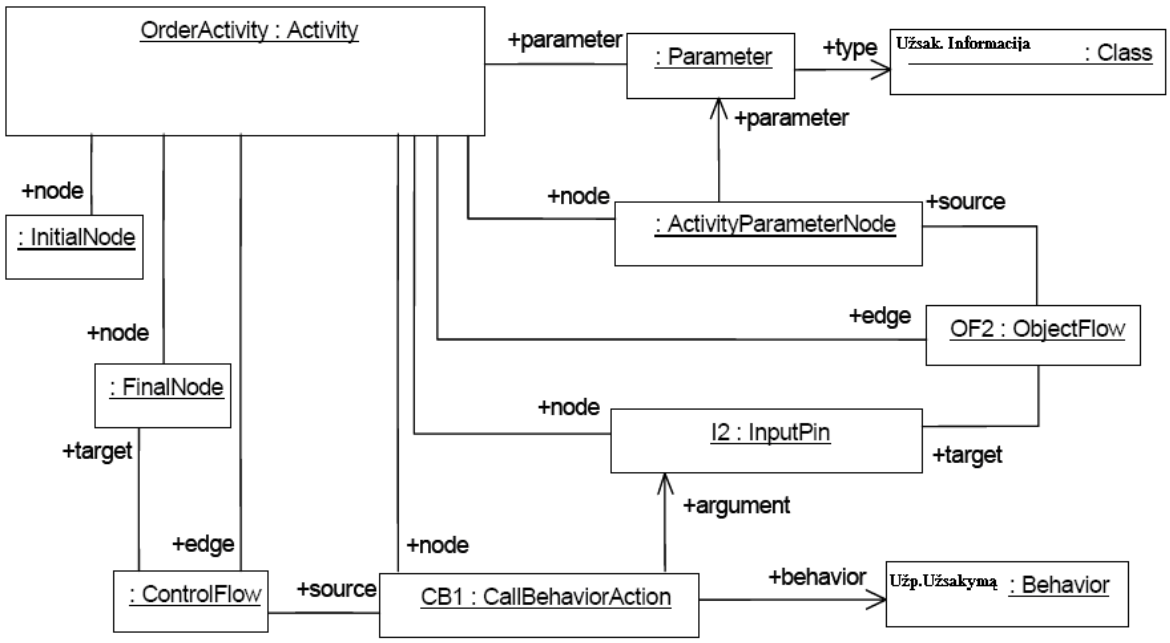
Veiklos yra vartotojo apibrėžtos elgsenos, ir kaip visos UML 2 elgsenos gali prasidėti nuo veiksmų, iškvietimų, kurios turi pagalbinius parametrus skirtus duomenų gavimui ir perduoti duomenis veiklą iškvietusiam objektui. Veiklos apibrėžimo dalis yra parametrai. Parametrai gali būti daugkartinio naudojimo. Parametrai nėra jungtys, todėl, kad jungtys yra naudojamos sujungti veiksmus į srautą, tuo tarpu veiklos yra elgsenos iškviečiamos veiksmų. Tačiau, kad nuskaitytų parametro vertes iš veiksmų veikloje, veiklos parametrai yra modeliuojami kaip objekto viršūnės rūšis, kuri naudojama laikinam parametro verčių laikymui, kol vertės juda į ir iš veiklos. 2.19 pav. rodo parametrizuotą veiklą su parametrų

objekto viršūne, sujungta su veiksmų jungtimis. Parametro objekto viršūnes rodomos ant krašto, su objekto srauto kraštais, jungiančiais juos su jungtimis. Objekto tipą dažniausiai nurodo objekto viršūnėje rodoma pavadinimas. Šiame pavyzdyje, informacija naudojama užpildyti užsakymą yra pateikiama kaip įvesties parametras ir yra perduodamas į „UžpildytiUžsakymą“ elgsenos iškvietimą.



2.19 pav. Pavyzdinė veikla

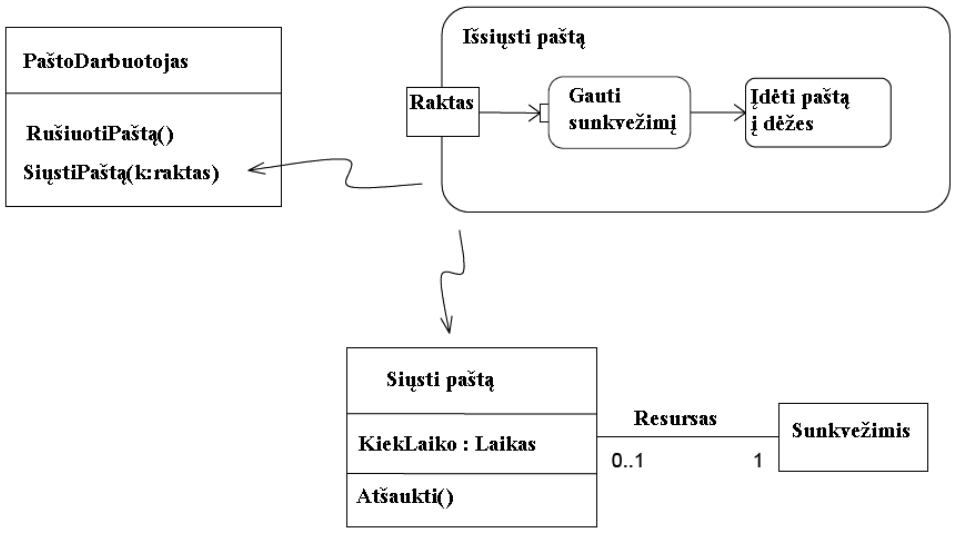
Valdymo viršūnės pradžioje ir pabaigoje srauto 2.19 pav. yra pradinės ir paskutinė viršūnės. Kai „UžsakymoVeikla“ yra iškviečiama, valdymo simbolis yra patalpinamas į pradinę viršūnę, ir duomenys su užsakymo informacija yra patalpinami į įvesties parametro objekto viršūnę. Valdymo simbolis juda nuo pradinės viršūnės į „UžsakymoVeikla“ veiksmą, kuris pradedamas vykdyti. Duomenys teka nuo parametro į iškvietimo veiksmą „UžpildytiUžsakymą“, kuris turi laukti kol „SukurtiUžsakymą“ suteiks įvesties jungtį prieš pradedant vykdyti. Kai FILLORDER yra pabaigęs vykdymą, valdymo simbolis perduodamas į paskutinę viršūnę, ir veikla baigiasi, grąžindama valdymą iš kurio viskas pradėjo. Dalinis 2.19 paveikslo saugyklos modelis yra vaizduojamas 2.20 paveiksle, kuris yra apačioje. Sujungiamas 2.20 paveikslo objekto modelį per CB1 CALLBEHAVIORACTION. „SukurtiUžsakymą“ veiksmas ir jo srautai yra praleisti dėl aiškumo.



2.20 pav. Dalinis 2.19 paveiksle pavaizduotų veiklų saugyklos modelis

2.2.6. UML 2 elgsenos

UML 2 palaiko parametrizuotas elgsenas visoms elgsenų rūšims ir veikloms. Tai reiškia būsenų mašinos, sąveikos, ir veiklos, viskas gali būti parametrizuota. Objektų metodai ar iškviečiami tiesiogiai vienodu būdu. Viršutinė 2.21 paveikslo kairė pusė rodo veiklos modelį elgesiui, pavadintam „Išsiųsti paštą“. „Išsiųsti“ galėjo būti iškviestas, kaip ir CALLBEHAVIORACTION, ar kaip metodas iš „PaštoDarbuotojas“ klasės su CALLOPERATIONACTION. Kiekvienu atveju, elgsena ima KEY kaip įvestį. Kadangi elgsena gali būti iškviesta tiesiogiai ar kaip operacija.



2.21 pav. UML 2 Elgsenos

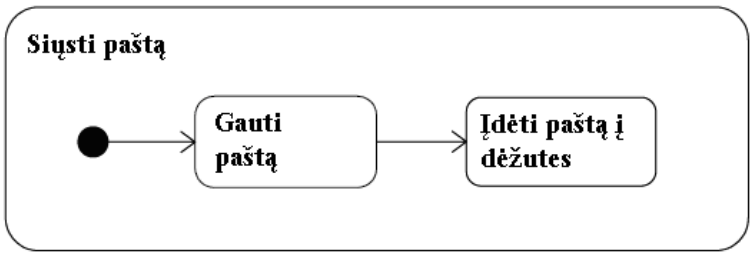
UML 2 vartotojo apibrėžta elgsena yra klasė. Kiekvieną kartą kai elgsena yra vykdoma, naujas vartotojo elgsenos klasės egzempliorius yra sukuriama. Paleistis (angl.

Instance) yra sunaikinama, kai elgsena baigiama. Elgsenos klasės, kaip visos klasės, gali palaikyti atributus, asociacijas, operacijas, ir net kitas elgsenas, tokias kaip būsenų mašinas. Tai atspindi įprastą praktiką sistemose, kurios valdo procesus, pavyzdžiui, darbų sekas (angl. *Workflow*) ir operacines sistemas. 2.21 pav. apačioje parodyta elgsenos klasė „Siųsti paštą“ veiklai su požymiu, kiek laiko kiekvienas Siųsti paštą“ veiklos vykdymas užtruko ir kokį sunkvežimį, kuris buvo naudotas. Elgsenos klasė gali turėti būsenų mašinas, kurios padėtų aprašyti kiekvieną vykdymo būseną, tokią kaip NOT_STARTED, SUSPENDED ir taip toliau.

UML 2 elgsenos klasės įgalina apibrėžti proceso standartinį valdymo funkcionalumą. Elgsenos klasės ypatybės gali būti apibrėžtos srities standartų, pardavėjų, ar vartotojų grupių kaip daugkartinio naudojimo pavyzdinės bibliotekos, turinčios savyje abstrakčias elgsenų klases su normatyviniais atributais ir operacijomis tokiomis kaip ABORT ir taip toliau. Tada šitos klasės gali būti panaudotos kaip super-tipai vartotojo apibrėžto elgsenų tokių, kaip „Siųsti paštą“ 2.21 iliustracijoje.

2.2.7. Pradinės viršūnės

Srautai veiklų diagramoje prasideda pradinėse viršūnėse. Pradinės viršūnės gauna valdymą, kai veikla yra pradėta. Viršūnės perduoda valdymą iškarto į iš jos išeinančias briaunas. Jokia kita elgsena nėra susieta su pradinėmis viršūnėmis UML modelyje. Pradinės viršūnės negali turėti briaunų, nukreiptų į save. Pavyzdžiui, 2.22 paveiksle, kai „Siųsti paštą“ veikla yra pradėta, valdymo simbolis patalpinamas ant pradinės viršūnės, pažymėtos kaip užpildytas apskritimas, ir nedelsiant juda į priekį, kad pradėdamas „Gauti paštą“ veiksmą.



2.22 pav. Pradinė viršūnė

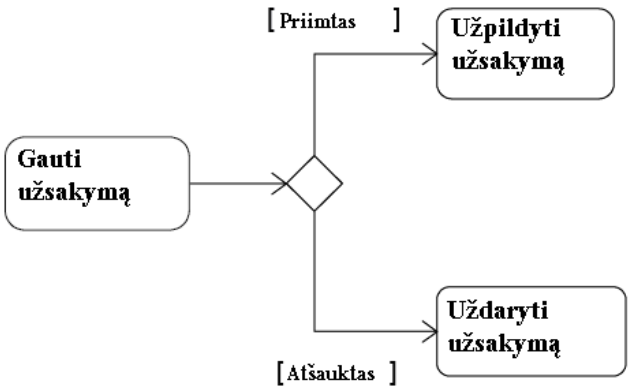
Veikla gali turėti savyje daugiau nei vieną pradinį viršūnę. Valdymo simbolis yra kiekvienoje pradinėje viršūnėje, veikla yra pradedama su visais srautais iš pradinių viršūnių. Aiškiau būtų panaudoti vieną pradinę viršūnę, prijungtą prie išsišakojančios viršūnės, kuri paleistų srautus tuo pačiu laiku. Pasirinktas sprendimas priklauso nuo projektuotojo.

Jei pradinė viršūnė turės daugiau kaip vieną išeinančią briauną, tiktai viena briauna gaus kontrolę, todėl, kad pradinės viršūnės negali nukopijuoti simbolių, kaip išsišakojantys mazgai gali (angl. *Fork Node*). Briaunos, išeinančios iš pradinių viršūnių, gali turėti sąlygas,

ir semantiškai bus identiška sprendimo viršūnei. Dėl patogumo pradinės viršūnės yra išskirtos iš bendrų taisyklių, todėl kontrolės viršūnės negali laikyti simbolių, kurie laukia judėjimo pasroviui, tai gali atsitikti jei visos sąlygos neįvykdytos. Bendru atveju aiškiau yra panaudoti detalius sprendimo taškus ir objekto viršūnes negu priklausyti nuo šitų detalizuotų taškų pradinėse viršūnėse.

2.2.8. Sprendimo viršūnės

Sprendimo viršūnės veda srautą viena ar kita kryptimi, bet tiksliai kokia kryptis yra nustatoma veikimo laiku briaunose, išeinančių iš viršūnės. Paprastai briaunos einančios nuo sprendimo viršūnės turi sąlygas, tai yra loginių verčių tikrinimas veikimu metu, kurių pagalba nustatoma ar kontrolė ir duomenys gali judėti šia briauna. Sąlygos yra nustatomos kiekvienai atskirai kontrolės ir duomenų simboliui atvykusiam į sprendimo viršūnę tam, kad nustatytų tiksliai vieną briauną, kurią simbolis pereis. Pavyzdžiui, 2.23 pav. rodo sprendimo viršūnę, užrašytą sutartiniais ženklais panašiais į rombą, kuri išrenka srautą priklausomai nuo to, ar užsakymas gali būti užpildytas ar ne. Vertės UML 2 yra dažnai tik simbolių masyvai interpretuojami priklausomai nuo realizacijos. Šiame pavyzdyje simbolių masyvai "Priimtas" ir "Atšauktas", turi būti suprantami iš realizacijos arba tūri būti apibrėžta patikslintame modelyje. Patikslintame modelyje galima įvesti papildomą detalią elgseną, tokią kaip sprendimo įvesties elgsena.

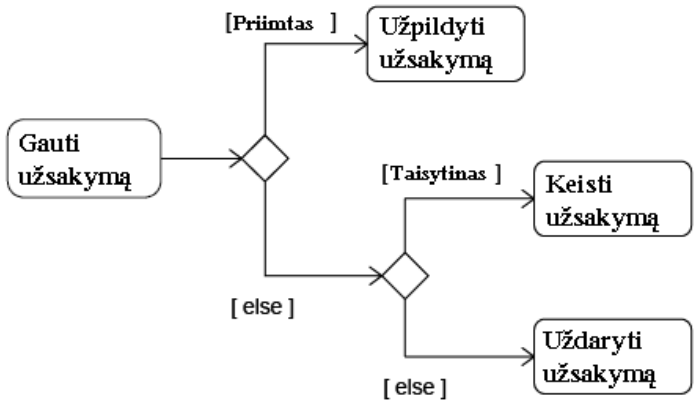


2.23 pav. Sprendimo viršūnės

Vykdymo tvarka, kurios metu anksčiau minėtos sąlygos įvertinamos nėra suvaržyta UML specifikacijos, todėl sąlygos gali būti nustatytos tuo pačiu metu. Dėl šios priežasties, sąlygos negali turėti neigiamo šalutinių poveikio. Jei sąlygos turi būti įvertintos tam tikra tvarka, taip kaip yra tipiškose sąlyginėse programinėse konstrukcijose, tai sprendimo viršūnės gali būti surištos kartu, kiekviena sąlyga turi viršūnę, sujungtą su apibrėžta iš anksto nustatyta sąlyga "else". Else sąlyga gali būti naudojama sprendimo viršūnėmis su viena išeinančia

briauna, kuri parodo kad turi būti trasuojama, jei visos kitos sprendimo viršūnės sąlygos neįvykdytos.

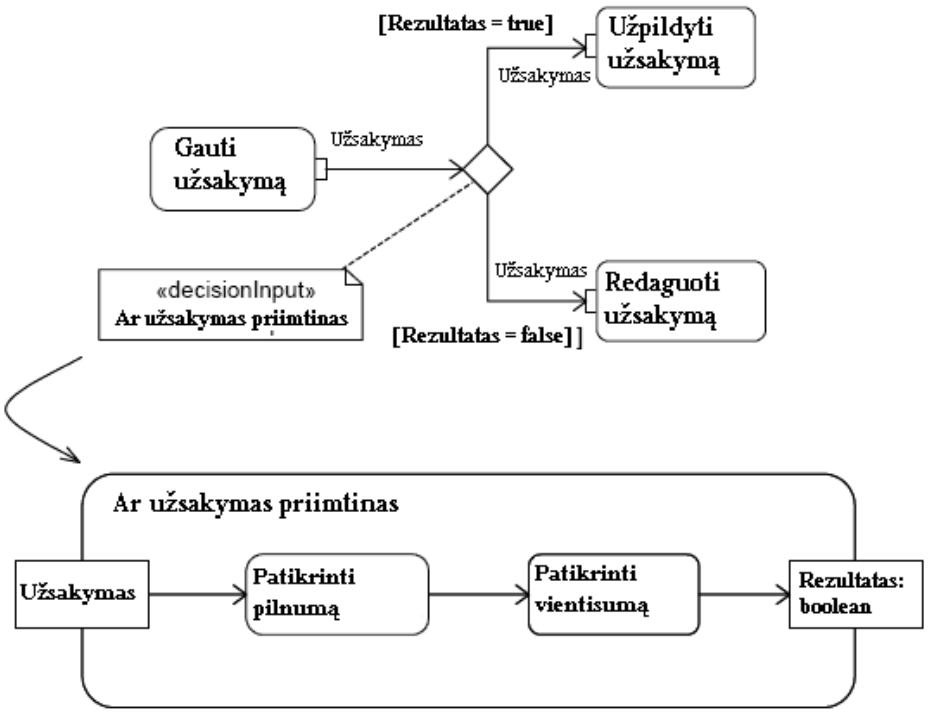
2.24 pav. rodo pavyzdį sujungtų sprendimo viršūnių su else sąlygomis. „Uždaryti užsakymą“ veiksmas yra pasiekiamas, kai ne „else“ sąlygos neįvykdymos.



2.24 pav. sujungtų sprendimo viršūnių grandinė

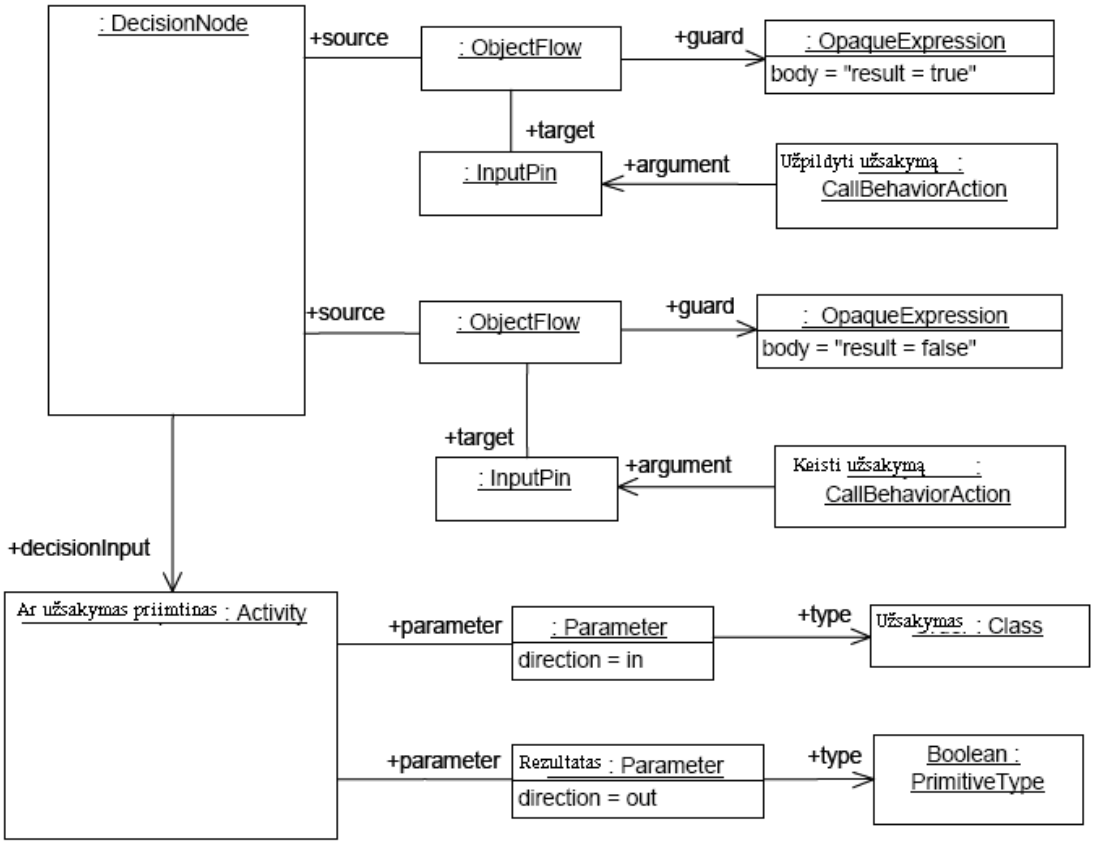
Kadangi sąlygų nustatymo tvarka priklauso nuo realizacijos, modeliuojant reikia taip sutvarkyti, kad tiktai viena sąlyga būtų sėkminga, kitaip bus lenktynių būsena tarp sąlygų. Tai priklauso nuo realizacijos, ar baigti sąlygų įvertinimą po to, kai vienas iš sąlygų buvo tenkinama. Pagal teoriją, visos sąlygos gali būti tenkinamos vieną kartą, bet tokiu atveju semantika nėra apibrėžta. Jei visos sąlygos netenkinamos, tai neveiksni kontrolė ar duomenų simbolis lieka objekto viršūnėje, iš kurios atvyko, kadangi kontrolės viršūnės negali laikyti simbolių laukiančių judėjimo toliau, tai gali padaryti tik objekto viršūnės.

Jei sąlygos apima pakartotinį tos pačios vertės skaičiavimą, sprendimo viršūnės elgsena gali nustatyti šią vertę, tik kai simbolis pasieks sprendimo viršūnę, ir paskui bus nukreiptas į išėjime esančias sąlygas, kuriose bus patikrintos. Pavyzdžiui, 2.25 pav. rodo, kad sprendimo įvesties elgsena „Ar užsakymas priimtinas“ suteikia rezultatui loginę reikšmę, kuri buvo patikrinta išėjime esančių sąlygų. Kiekvieną užsakymą, atvykusį į sprendimo viršūnę perduoda į, „Ar užsakymas priimtinas“ anksčiau, negu sąlygos bus nustatytos ant išeinančių briaunų. Elgsenos išėjimas yra pasiekiamas sąlygoms priklausomai nuo realizacijos kaip ir su visomis verčių specifikacijomis. Verčių specifikacija 2.25 pav. panaudoja sprendimo elgsenos išėjimo parametro vardą.



2.25 pav. Sprendimo įvesties elgsena

Objekto modelį 2.25 paveikslui yra parodytas 2.26 paveiksle. Du anoniminiai objekto srautai yra atskiri objekto elementai dviem objekto srautams, atvykstantiems iš sprendimo viršūnės. Kiekvienas turi nematomą išraišką, kuri yra kaip sąlyga, kuri yra tam tikros vertės specifikacijos rūšis interpretuojama priklausomai nuo realizacijos. Kiekvienas objekto srautas kreipiasi į savo atskirą anoniminį įvesties jungtį, iš kurių kiekvienas suteikia įvestis jų atitinkamoms elgsenoms, kiekvienai srauto krypčiai iš sprendimo.



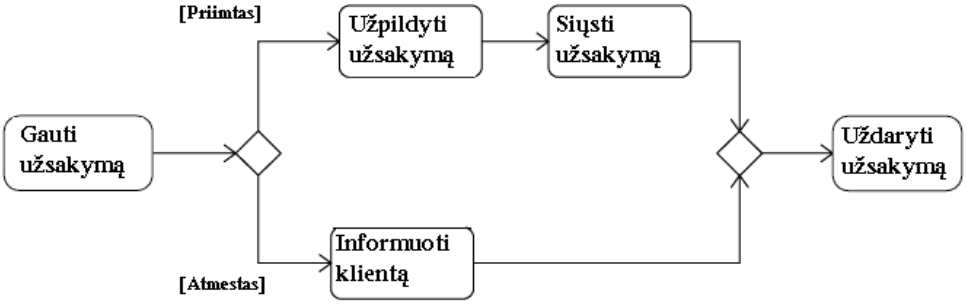
2.26 pav. dalinis 2.25 pav. saugyklų modelis

Kiti faktoriai be sąlygų gali nustatyti, ar kontrolė ir duomenys gali pereiti per briauną, ir kokia briauna iš viršūnių bus trasuojama, įskaitant sprendimo viršūnes. Kad ir kokie faktoriai yra įtraukti, sprendimo viršūnės tikslas yra užtikrinti, kad kiekvienas kontrolės ir duomenų simbolis pasiekiantis sprendimo viršūnę pereitų ne daugiau kaip per vieną iš išeinančių briaunų.

2.2.9. Jungimo viršūnės

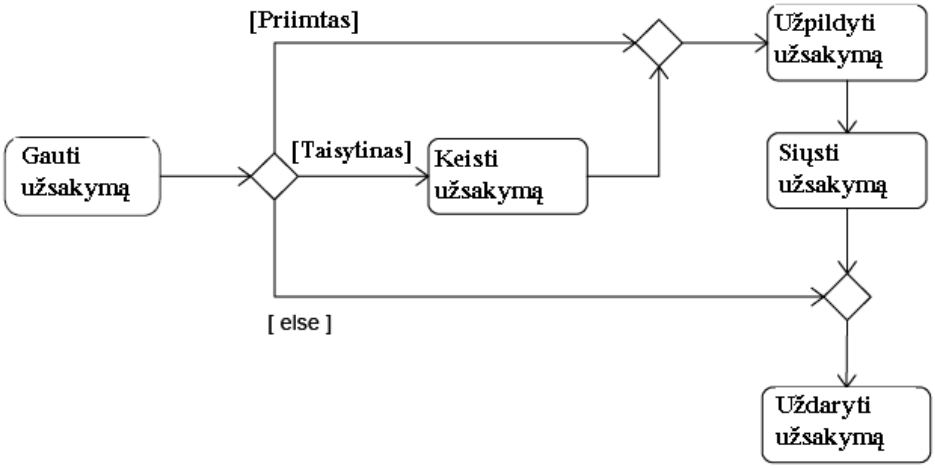
Jungimo viršūnės suburia daugialypius srautus. Visą kontrolę ir duomenis, atvykdama į jungimo viršūnę nedelsiant perduoti į briauną, išeinančią iš sujungimo. Jokia kita elgsena nėra siejama su sujungimo viršūnėmis UML specifikacijoje. Jungimo viršūnės turi tą patį žymėjimą ženklais kaip sprendimo viršūnės, bet jungimo viršūnės turi didelį kiekį įeinančių briaunų ir vieną išeinančią briauną, tuo tarpu tai yra priešingybė sprendimo viršūnėms. Srautai įeidami į jungimą yra paprastai alternatyvos iš prieš jį esančio sprendimo viršūnės. Pavyzdžiui, 2.27 pav. rodo sujungimo viršūnę, suburiantį du srautus, kurie uždaryto užsakymą. Sujungimas yra reikalingas, todėl, kad, jei du srautai nuėjo tiesiogiai į „Uždaryti

užsakymą“, abu srautai turėtų atvykti prieš uždarant užsakymą, kuris gali neįvykti. Jungimas gali būti panaudotas su lygiagrečiais srautais.



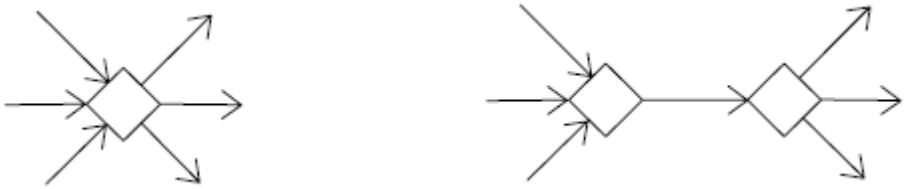
2.27 pav. Sujungimo viršūnės su alternatyviais srautais

Srautai iš sujungtų sprendimo viršūnių gali būti jungiami lanksčiau negu naudojantis sąlyginėmis konstrukcijomis struktūrinio programavimo kalbose. Pavyzdžiui, 2.28 pav. rodo dviejų iš trijų srautų išeinančio iš sprendimo viršūnės jungiami atskirai nuo trečio. Srautai, atvykstantys iš sprendimo viršūnės, neturi būtinai būti suburti jungimo viršūnėje.



2.28 pav. Sujungimo viršūnės

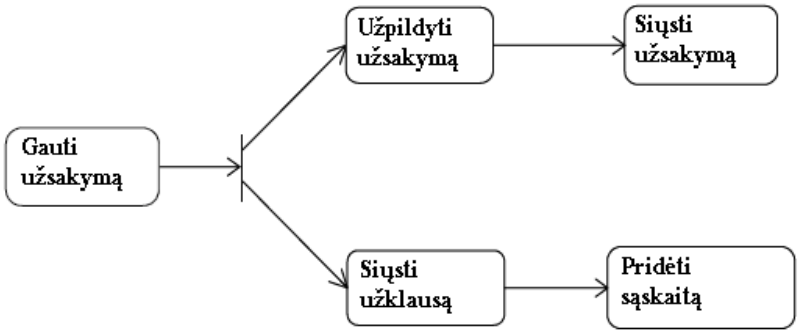
2.29 pav. kairėje pusėje rodo jungimo ženklo žymėjimą, kuriame iš karto seka sprendimas. Tai turi tą patį poveikį kaip atskiras sujungimas ir sprendimas, parodytas dešinėje. Abu turi tą patį objekto modelį, kuris turi savyje atskiras jungimą ir sprendimo viršūnes.



2.29 pav. Sujungimo / sprendimo viršūnių darinys

2.2.10. Išsišakojančios viršūnės

Išsišakojančios viršūnės padalina srautus į daugialypius lygiagrečius srautus. Kontrolė ir duomenys pasiekę išsišakojimą yra kopijuojami iš viršūnės išeinančiuose briaunose. Jokia kita elgsena nėra susieta su išsišakojančiomis viršūnėmis UML specifikacijoje. Pavyzdžiui, 2.30 pav. kontrolė ar duomenų simbolių išėjime iš „Gauti užsakymą“ yra kopijuojamos išsišakojime, pažymėta kaip tiesės atkarpa, ir pereis į „Užpildyti užsakymą“ ir „Siųsti užklausą“ tuo pačiu metu. Kadangi objekto simboliai yra tiktai nuorodos į objektus, kopijuojant juos nekopijuoja pačių objektų, tiktai nuorodas į juos. Nėra numatyta sinchronizacija lygiagrečiuose elgsenų srautuose UML 2 veiklos specifikacijoje, kadangi yra UML 1.x veiklos, kurios yra tam tikros būsenų mašinų rūšis. 2.30 pav. srautas, kad „Siųsti užsakymą“ gali būti užbaigtas, anksčiau nei „Siųsti užklausą“ yra pasibaigęs ir atvirkščiai.



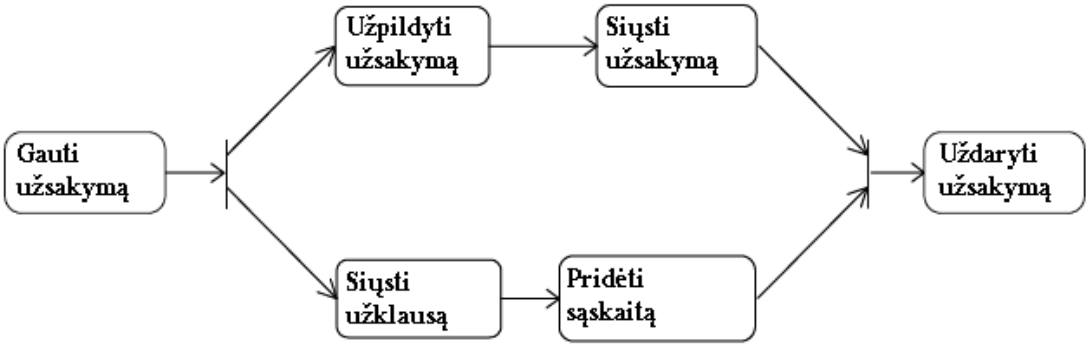
2.30 pav. Išsišakojusi viršūnė

Standartinė semantika srautams, atvykstantiems iš veiksmo, yra, kad jie yra visi srautai prasideda, kai veiksmas pabaigiamas. Taip sukuriami lygiagretūs srautai, bet išvedami duomenys iš veiksmų nėra kopijuojami. Veiksmas išveda atskirą vertę kiekvienam srautui. Veiksmo išvestys yra ant jungčių, kurios yra objekto viršūnės tipas, ir todėl laiko vertes, kol laukiama judėjimo toliau. UML 1.x, duomenų srautai yra pagrįsti būsenų perėjimu, tokiu būdu tiktai vienas srautas yra pradėtas, kai būsena (veiksmas) yra baigtas.

2.2.11. Jungimo (angl. Join) viršūnės

Jungimo viršūnės sinchronizuoja daugialypius srautus. Bendrajame atveju, kontrolė ar duomenys turi būti pasiekiami kiekvienoje įeinančioje briaunoje, kad būtų galima perduoti į išeinančią briauną. Jungimo viršūnės turi tuos pačius ženklų žymėjimus kaip išsišakojimo viršūnės, bet sujungimai turi daugialypes įeinančias briaunas ir vieną išeinančią briauną, tuo tarpu tai yra priešingybė išsišakojimo viršūnėms. Srautai įeidami į sujungimą yra dažnai lygiagretūs srautai iš prieš tai buvusio išsišakojimo. Pavyzdžiui, 2.31 iliustracija rodo sujungimo viršūnę, sinchronizuojančią du srautus, kad įvykdytų CLOSE ORDER. Ir SHIP

ORDER ir ADD ACCOUNT PAYABLE turi būti užbaigti anksčiau, negu CLOSE ORDER gali prasidėti.



2.31 pav. Jungimo viršūnė

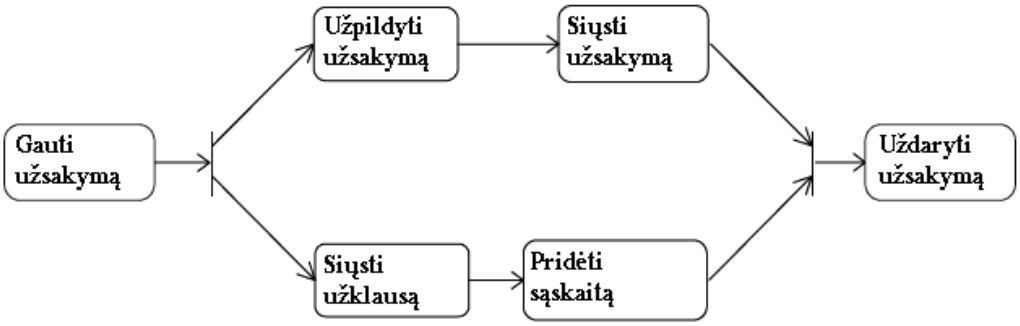
Jungimo viršūnės paima vieną simbolį nuo kiekvieno iš įeinančių briaunų ir apjungia juos pagal šias taisykles:

1. Jei visi įeinantys simboliai yra kontrolei, tai jie yra apjungiami į vieną kontrolės simbolį skirtą išeinančiai briaunai.

2. Jei kai kurie iš įeinančių simbolių yra kontrolei, o kiti yra duomenys, tai jie yra apjungiami, kad nukreiptų tiktai duomenų simbolius išeinančiai briaunai. Kontrolei skirti simboliai yra sunaikinami

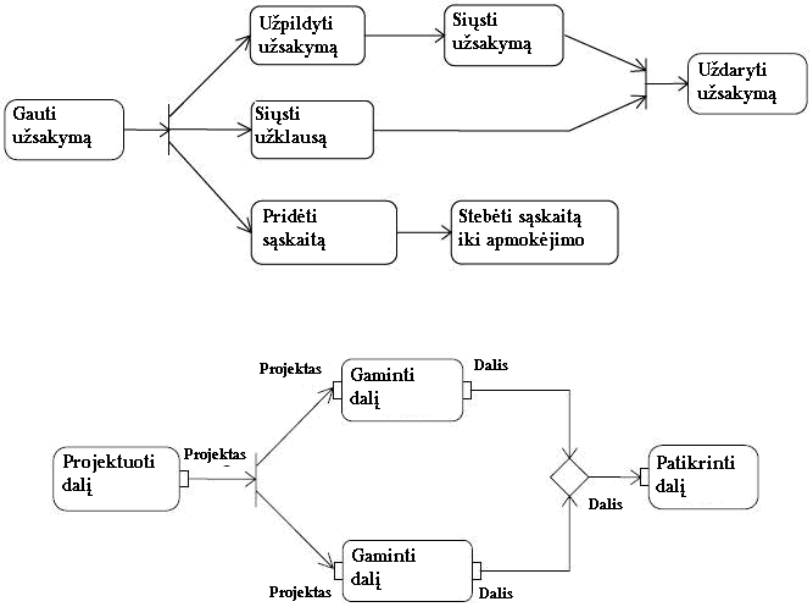
Pavyzdžiui, 2.32 pav. jungimas apjungia kontrolės simbolius iš „Siųsti užsakymą“ ir „Pridėti sąskaitą“ į vieną, kad „Uždaryti užsakymą“ būtų atliekamas vieną kartą vietoj dviejų kartų.

Būtų tas pats, jei jungimas yra praleistas, ir du srautai eina tiesiogiai į „Uždaryti užsakymą“, todėl, kad veiksmas lauktų abiejų srautų bet kokiu būdu. Būtų aiškiau panaudoti jungimą, tuo labiau, kad UML 1.x veikloms reikės tiktai vieno srauto, kad pasiektų, kaip ir su visoms būsenų mašinoms. Tačiau, jei srautai nešė duomenis, du simbolius perduos per išeinančią briauną po sinchronizacijos nueis į vieną „Uždaryti užsakymą“ jungtį. Tai turėtų nepageidaujamą „Uždaryti užsakymą“ poveikį - paleidimą, du kartus ir net nebus įvykdoma, jei duomenys yra nesuderinamų tipų, todėl, kad juos abu nukreiptų į tą patį įvesties jungtį. Pavyzdžiui, „Siųsti užsakymą“ galėtų išvesti sekimo (angl. *tracking*) įrašą, ir „Pridėti sąskaitą“ nauja sąskaita, kurios yra būtinos „Uždaryti užsakymo“ įvestys. Šiuo atveju, duomenų srautus turi nukreipti į dvi „Uždaryti užsakymą“ jungtis, be jungimo, kaip parodyta 2.32 pav. Tai yra kitas pavyzdžio patobulinimo pavyzdys. 2.30 pav. galėtų būti paimtas kaip proceso pradinis modelis, kuris vėliau buvo patobulintas į 2.32 pav., kai aišku, kokios įvestys yra būtinos, kad būtų užbaigtas užsakymas.



2.32 pav. Jungimas duomenų srautų naudojant jungtis

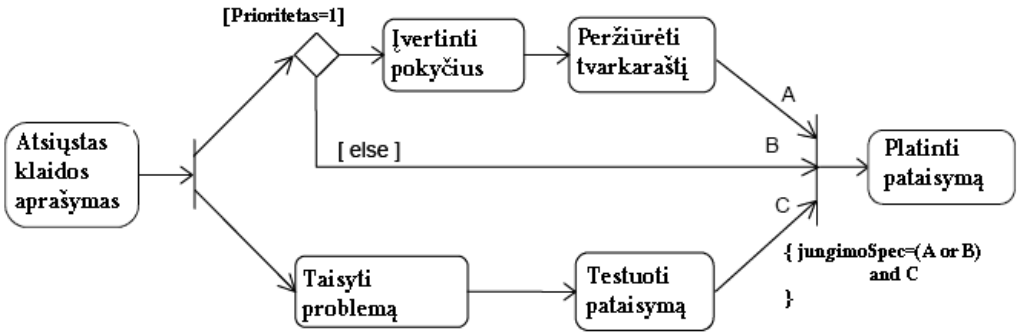
Nėra būtina, kad srautai, atvykstantys iš išsišakojimo būtų sinchronizuoti. Pavyzdžiui, 2.33 iliustracija rodo tiktai kai kuriuos iš srautų einančių iš šakutės juda į jungimą. Užsakymas yra uždarytas po to, kai jis yra nusiųstas ir išrašyta sąskaita, bet mokėtina sąskaita galėtų būti kontroliuota ilgą periodą vėliau, dėl to nėra sinchronizuotas su užsakymo uždarymu. Lygiagretūs srautai gali taip pat būti sujungti, o ne prijungti, kaip parodyta 2.33 iliustracijoje. Šiame pavyzdyje, dalies apžiūra yra leista dalimis, tuo metu, kai dvi dalys gali būti padaromos lygiagrečiai. INSPECT PART veiksmas bus atliktas dukart, po vieną veiksmą kiekvienai daliai, atvykstančiai lygiagrečiu srautu. Tai reikalauja daugiau kaip vieno simbolio, judančio tą pačią srauto liniją vienu metu. Jie yra daugiau išraiškingesnių pavyzdžių įvesto UML 2 specifikacijos veiklose lyginant su UML 1.x veiklos specifikacija.



2.33 pav. Išsišakojimas su sujungimu (merge)

Modeliuotojai gali apibrėžti sąlygas, su kuriomis jungimas priima įeinančią kontrolę ir duomenis, naudodamas jungimo specifikaciją (angl. *join specification*), kuri yra loginių sąlygų specifikacija susieta su sujungimo viršūnėmis. UML paveldėtas standartas yra "and" su semantika apibūdinta anksčiau. Kitos jungimo specifikacijos gali būti duotos, naudojant įeinančių briaunų vardus susiejimui kontrolės ar duomenų ateinančių į sujungimą. Pavyzdžiui,

2.34 pav. parodo alternatyvą 2.32 pav. jungimo (angl. *merge*) viršūne pakaitalui naudojant sujungimo viršūne (angl. *join*). Briaunas pavadintos naudojant vieną raidę šiame pavyzdyje, bet gali būti bet koks pavadinimas.



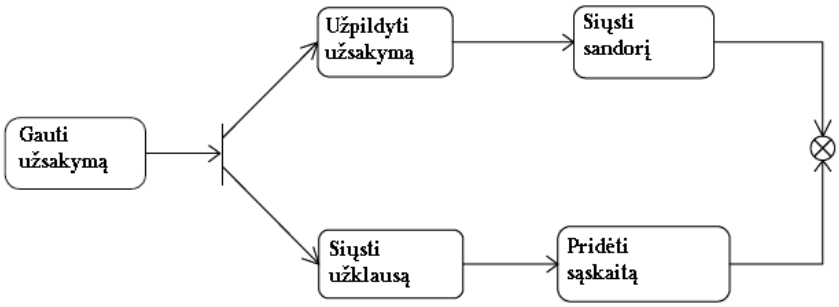
2.34 pav. Sujungimo specifikacija

2.35 pav. kairėje pusėje renginiuose trumpą ženklinimą sujungimui po kurio iš karto seka išsišakojimas. Tai taip pat realizuojama kaip atskiras sujungimas ir išsišakojimas, kaip parodyta dešinėje. Abu turi tą patį objekto modelį, kuris turi atskiras jungimo ir išsišakojimo viršūnes.

2.35 pav. Jungimo/šakojimosi variantai

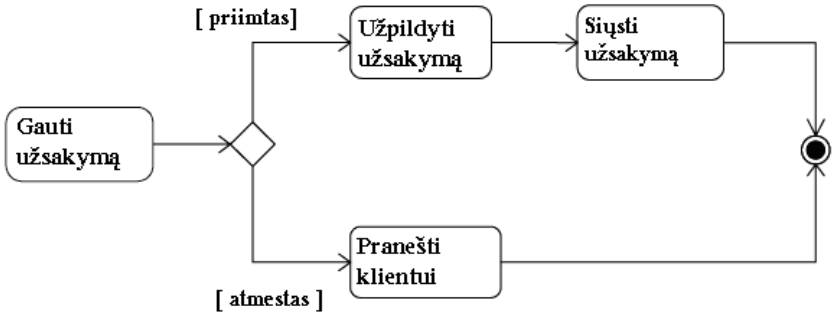
2.2.12. Paskutinės viršūnės (angl. Final nodes)

Srautas veikloje pasibaigia paskutinėse viršūnės. Nepavojingiausia forma yra srauto pabaiga, kuri paima kontrolę ar duomenis, kurie įeina ir nedaro nieko. Srauto paskutinės viršūnės negali turėti išeinančių briaunų tokiu būdu nėra jokio įeinančių simbolių poveikio, nes jie yra tiesiog sunaikinami. Kadangi objekto simboliai yra tik nuorodos į objektus, sunaikindamas objekto simbolį nesunaikini objektą. 2.36 pav. išplečia 2.32 iliustraciją, kurio pabaigą papildo srauto paskutine viršūne. Kiekvienas srautas galėjo turėti savo savą paskutinį srautą, ir poveikis bus tas pats. Veiklos baigiasi, kai visi simboliai grafe yra sunaikinti, tokiu būdu ši veikla bus baigta, kai abu srautai pasieks srauto pabaigą.



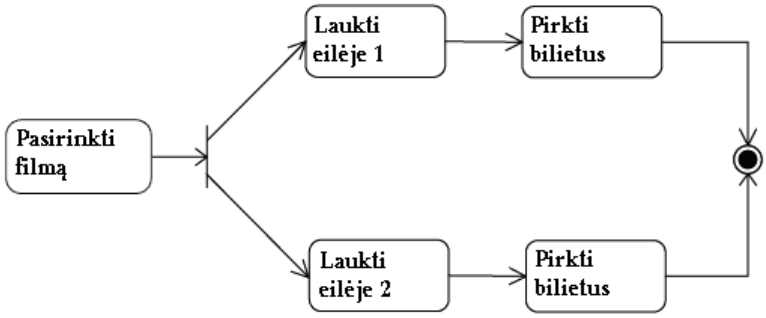
2.36 pav. Paskutinio srauto viršūnė

Veiklos finalo viršūnės panėšėja į srauto finalo viršūnes, išskyrus tai, kad kontrolė ar duomenys, atvykdama į juos nedelsiant baigia visą veiklą. Tai turi reikšmę, jei daugiau kaip vienas kontrolės ar duomenų simbolis gali dar judėti grafu, tuo laiku kai veiklos pabaiga yra pasiekta, kaip yra 2.36 pav. Veiklos pabaiga negali būti panaudotas vietoj srauto pabaigos, todėl, kad vieno lygiagretaus srauto baigimas nutrauktų kitą srautą. 2.37 pav. neturi reikšmės, ar srauto pabaiga ar veiklos pabaiga yra panaudojama, vykdymo eigos rezultatas yra tas pats. Taip pat kiekvienas srautas galėjo turėti savo savą veiklos pabaigą gale, ir rezultatas bus tas pats.



2.37 pav. Veiklos pabaigos viršūnė

2.38 pav. yra pavyzdys, kur veiklos pabaigos nutraukimo funkcionalumas yra panaudotas tyčinėse lenktynėse tarp srautų. Tai yra kino filmo bilietus pirkimo procesas, žmonės stovi atskirose linijose, kol vienas iš jų gauna bilietus grupei. Greičiausia linija pagamins veiklos pabaigos simbolį ir užbaigs kitą srautą.



2.38 pav. Veiklos pabaigos viršūnė, lenktyniavimo sąlyga

2.3. BPMN kalbos analizė

Perėjimas nuo duomenų į procesų orientaciją ir susitelkimas ties procesais paremtomis informacijos sistemų (PAIS – Process Aware Information Systems) praėjusiame dešimtmetyje yra artimai susijęs su naujos kartos kalbų ir įrankių proceso apibrėžimui išsivystymu. Egzistuojančios (vyraujančios) kalbos informacinių sistemų tobulinime buvo patobulintos, pavyzdžiui UML 2.0 išleistas su esminiais atnaujinimais elgsenų srityje, t.y. UML Veiklos Diagramomis (AD). Lygiagrečiai naujos kalbos kaip BPMN ir BPEL4WS buvo kuriamos ir greitai pasklido.

Šitų trijų kalbų bendras bruožas yra susikoncentravimas ties galingu ir standartizuotu žymėjimu skirto atvaizduoti įvairius verslo elgsenos aspektus. Tarp skirtumų gali būti paminėta, kad UML AD ir BPMN yra grafiniai, bet ne - formalizuoti žymėjimai, BPEL4WS yra vykdomoji kalba (ir todėl yra formalizuota), kuri stokoja grafinio žymėjimo.

Šios grubios charakteristikos nesuteikia jokių įžvalgų kokios yra kalbos išraiškos galimybės ir kaip jos iš tikrųjų susisieja viena su kita. Tuo metu, dalis supranta UML 2.0 AD ir BPMN kaip potencialus grafinius žymėjimus BPEL4WS kalbos nors šios kalbos yra konkurentai, pagal citatą,

Kur BPMN turi susitelkia ties verslo procesais, UML turi koncentruojasi ties programinės įrangos projektavime, ir todėl kalbų žymėjimais tarpusavyje nekonkuruoja, be to tai yra skirtingi požiūriai apie sistemas.[17]

gynėjai BPMN, būtent BPMI.org, aišku nesutinka su šią nuomone. Kad sugebėtų palaikyti ar atmesti tokius tvirtinimus, dar svarbiau, kad ištirtų panašumus ir skirtumus tarp kalbų, nuodugni analizė ir palyginimas yra būtinas.

Workflow šablonų (www.workflowpatterns.com) karkasas suteikia pagrindą palyginamajai analizei. Ši karkasas susideda iš daugelio šablonų, kurie suteikia bendrų, grįžtančių koncepcijų ir konstrukcijų taksonomiją tinkamą procesą suvokiančių informacijos sistemų (PAIS) kontekstui. Pagal Jablonski ir Bussler originalią [19] klasifikaciją, ši struktūra apima kontrolės srautus, duomenis, ir išteklių perspektyvas ir buvo palaipsniui išvystyta, kad įtrauktų 20 valdymo srauto šablonų [18], 40 duomenų šablonų [20], ir 43 išteklių šablonų [22]. Metodiškai tirdamas galimybę reikšti kiekvieną šabloną iš karkaso pasirinkta kalba, sukuria visapusį paveikslą, kuris nurodo sritį ir tinkamumą šios kalbos dėl trims matmenims, paminėtais anksčiau

Yra neišvengiamų sunkumų pritaikant Workflow šablonų karkasą kalbos įvertinimui, kai kalba neturi bendros suderintos formalios semantikos, nei vykdymo aplinkos. BPMN [21] specifikacija suteikia atvaizdavimą iš BPMN į BPEL, kuri turi vykdymo variklį ir

egzistuojančią formalizaciją. Atidžiau nagrinėjant atvaizdavimas yra dalinis, paliekant modelius su nestruktūrizuotomis topologijomis ir konstrukcijomis tokiomis kaip OR-join ir sudėtingi maršrutai (angl. *gateway*) [22] Be to, sudarytas atvaizdavimas yra verstas iš modelio, todėl galimos skirtingos modelio interpretacijos. Be to daug dviprasmybių gali būti surastos BPMN specifikacijoje dėl trūkstamos formalizacijos.

Lentelė 2.1 Kontrolės srautų šablonai

Šablonai	BPMN	UML 2.0 Activity Diagram	BPEL	Oracle BPEL PM v.10.1.2
Paprastos valdymo sekos				
Seka	+	+	+	+
Lygiagretus išsišakojimas	+	+	+	+
Sinchronizacija	+	+	+	+
Pasirinkimas	+	+	+	+
Paprastas jungimas	+	+	+	+
Sudėtingesnė sinchronizacija				
Daugialypis pasirinkimas	+	+	+	+
Sinchronizuojamas jungimas	+/-	-	+	+
Daugialypis jungimas	+	+	-	-
Diskriminatorius	+/-	+	-	-
Struktūriniai šablonai (Structural Patterns)				
Pasirenkami ciklai(Arbitrary Cycles)	+	+	-	-
Visiškas nutraukimas (Implicit Termination)	+	+	+	+
Keleto paleisčių šablonai (Multiple Instances Patterns)				
Kelios paleistys su sinchronizacija	+	+	+	+
Kelios paleistys projektuotos žinant laiką	+	+	+	+
Kelios paleistys projektuotos žinant paleidimą	+	+	-	+
Kelios paleistys projektuotos nežinant palaidimo	-	-	-	+/-
Būsenomis paremti šablonai				
Atidėtas pasirinkimas	+	+	+	+
Nepersidengiantis lygiagretus paskirstymas (Interleaved parallel routing)	+/-	-	+/-	-
Etapas (Milestone)	-	-	-	+/-
Atšaukimo šablonai				
Atšaukimo veikla	+	+	+	+/-
Atšaukimo atvejis	+	+	+	+

Rezultatai valdymo srautų analizės požiūriu yra tai, kad daugelis šablonų gali būti atvaizduojami keliais būdais. Paprastesnės struktūros gali turėti tris skirtingas BPMN išraiškas. Iš kitos pusės, išsamios žinios atributų susietų su BPMN modeliuojamomis konstrukcijomis (kurie neturi grafinio pavaizdavimo) yra būtinos norint atvaizduoti dalį sudėtingesnių šablonų.

Lentelė 2.2 Duomenų šablonų palaikymas 1–BPMN, 2–UML 2.0 AD, 3–BPEL ir 4–Oracle BPEL PM v. 10.1.2

Duomenų matomumas	1	2	3	4	Veiksmai su duomenis(išorinis)	1	2	3	4
1. Užduočių duomenys	+	+/-	+/-	+/-	21. Aplinka į tipą	-	-	-	-

2. Bloko duomenys	+	+	-	-	22. Tipas į aplinką	-	-	-	-
3. Srities duomenys	-	-	+	+	23. Darbų seka į aplinką (stūmimas)	-	-	-	-
4. Kelių paleisčių duomenys	+/-	+	-	+/-	24. Aplinka į seką (traukimas)	-	-	-	-
5. Pavyzdžio (angl. *case*) duomenys	+	-	+	+	25. Aplinka į darbų seką (stumimas)	-	-	-	-
6. Katalogo duomenys	-	-	-	-	26. Darbų seką į aplinką (traukimas)	-	-	-	-
7. Darbų sekų duomenys	-	+	-	-	Duomenų perdavimas				
8. Aplinkos duomenys	-	-	+	+	27. Pagal vertę - įeinantis	+	-	+	+
Duomenų sąveika (vidinė)					28. Pagal vertę – išeinantis	+	-	+	+
9. Tarp užduočių	+	+	+	+	29. Kopijuoti į / iš	+/-	-	-	+
10. Blokinės užduoties suskaidymas	+	+	-	-	30. Pagal adresą – neužrakinta	-	-	+	+
11. Suskaidymas į blokinę užduotį	+	+	-	-	31. Pagal adresą – užrakinta	+	+	+/-	-
12. Į daugialypes paleisčių užduotis	-	+	-	+/-	32. Duomenų transformacija įvestis	+/-	+	-	-
13. Iš daugialypių paleisčių užduočių	-	+	-	+/-	33. Duomenų transformavimas – įvestis	+/-	+	-	-
14. Pavyzdys į pavyzdį	-	-	+/-	-	Duomenimis pagrįstas skirstymas				
Duomenų sąveika išorinė					34. Užduoties prielaida – Duomenys egzistuoja	+	+	+/-	-
15. Užduotis į aplinką (stumimas)	+	-	+	+	35. Užduoties prielaida – validuoti duomenys	-	+	+	+
16. Aplinka į užduotį (traukimas)	+	-	+	+	36. Užduoties būsena – duomenys egzistuoja	+	+	-	-
17. Aplinką į užduotį (stumimas)	+	-	+/-	+	37. Užduoties būsena – validuoti duomenys	-	+	-	-
18. Užduotis į aplinką (traukimas)	+	-	+/-	+	38. Įvykiais pagrįstas užduočių trigeris	+	+	+	+
19. Pavyzdys į aplinką (stumimas)	-	-	-	-	39. Duomenimis pagrįstas trigeris	+	-	+/-	-
20. Aplinka į pavyzdį (traukimas)	-	-	-	-	40. Duomenimis pagrįstas kelio parinkimas	+	+	+	+

Dėl BPMN duomenų šablono palaikymo iš 2.2 lentelės matyti, kad darbų seka ir aplinkos duomenų šablonai nėra palaikomi. Duomenų sąveika į ir iš daugialypės paleisčių užduočių nėra palaikoma, todėl, kad bet kokie nuo specialūs paleisties duomenys skirti užduočiai ar sub-procesas su “daugialype paleistimi” žymekliu negali būti apibrėžti. Taip pat išorinės duomenų sąveikos šablono palaikymas yra ribotas. Tiktai šablonai atvaizduojantys sąveiką tarp užduočių ir aplinkos yra palaikomi, kadangi jie gali būti atvaizduoti modeliuojant aplinką kaip atskirą procesą, kuris gali būti atvaizduojamas pilnai kaip abstrakcija/viešas procesas, ar netiesiogiai per nuorodas siunčiant ir gaunant užduotis/įvykius.

BPMN išteklių požiūriu palaikymas yra minimalus. Tai patvirtinta specifikacijoje ([18], p. 22), kad organizacinės struktūros ir išteklių modeliavimas yra už BPMN apimamos srities. Tačiau sąvokų kelias (angl. *lane*) ir saugyklos (angl. *pool*) egzistavimas, kurių pagalba atvaizduojamos grupės ir rolės sudaro prieštaraujantį įspūdį. Akivaizdu, kad saugyklos ir keliai nesuteikia būdų atvaizduoti subtilių detalių susijusių su parenkamo (angl. *selective*) darbo paskyrimu per daugybę galimų išteklių ir veiklos metu ir iš to sekančių darbo dalykų valdymą.

Lentelė 2.3 Išteklių šablonų palaikymas 1–BPMN, 2–UML 2.0 AD, 3–BPEL ir 4–Oracle BPEL PM v. 10.1.2

Kūrimo šablonai	1	2	4	**Traukimo šablonai (turinio)**	1	2	4
1. Tiesioginis išskyrimas	+	+	+	24. Sistemos apibrėžti.Darbų eilės turinys.	-	-	-
2. Rolėmis pagrįstas išskyrimas	+	+	+	25.Resursų apibrėžti. Darbų eilės turinys	-	-	+
3. Atidėtas išskyrimas	-	-	+	26. Pasirinkimo autonomija	-	-	+
4. Autorizacija	-	-	-	**Aplinkkelio šablonai**			
5. Pareigų atskyrimas	-	-	-	27. Delegavimas	-	-	+
6. Atvejų apdorojimas	-	-	+	28. Eskalavimas	-	-	+
7. Išsaugoti panašumą	-	-	+	29. Nepriskyrimas	-	-	+
8. Galimybėmis pagrįstas išskyrimas	-	-	+	30. Teginių perskirstymas	-	-	+
9. Istorija paremtas išskyrimas	-	-	+/-	31. Perskirstymas be būsenų	-	-	-
10. Organizacinis Išskyrimas	-	-	+/-	32. Apribojimas / teisių atgavimas	-	-	+
11. Automatinis vykdymas	+	+	+	33. Praleisti	-	-	+
Stumimo (angl. *Push*) šablonai				34. Perdaryti	-	-	-
12. Paskirstymas naudojant vieną resursą	-	-	+	35. Prieš padaryti	-	-	-
13. Paskirstymas naudojant daugelį resursų	-	-	+	**Automatinio paleidimo šablonai**			
14. Skirstymas paskiriant vieną resursą	+	+	+	36. Pradžia sukūrimo metu	+	+	-
15. Atsitiktinis skirstymas	-	-	+/-	37. Pradžia išskyrimo metu	-	-	-
16. Round Robin skirstymas	-	-	+/-	38. Krūvos vykdymas	-	-	-
17. Trumpiausios eilės	-	-	+/-	39. Susietas vykdymas	+	+	-
18. Ankstyvo paskirstymo	-	-	-	**Matomumo šablonai**			
19. Paskirstymas naudojant įgalinimą	+	+	+	40. Konfigūracija. Neišskirtas darbo vieneto matomumas	-	-	-
20. Vėlyvas paskirstymas	-	-	-	41. Konfigūracija. Išskirtas darbo vieneto matomumas	-	-	-
Traukimo (angl. *Pull*) šablonai				**Daugelio resursų šablonai**			
21. Resursų pradinis išskyrimas	-	-	-	42. Vienalaikis vykdymas	+	+	+
22. Resursų išskyrimas paleidžiant – išskiriamas darbo vienetas	-	-	+	43. Papildomi resursai.	-	-	+
23. Resursų išskyrimas vykdant – išskiriamas pasiulytas darbo vienetas	-	-	+				

Lentelėse taip pat palyginta BPMN su UML 2.0 AD ir BPEL. Pagal Controlflow perspektyvą, BPMN ir UML 2.0 AD didžia dalimi persidengia. BPMN yra truputį stipresnis kai pereinama prie persidengiančio lygiagretaus pasiskirstymo atvaizdavimo ir sinchronizuojančio jungimo šablono ir truputį silpnesnis diskriminatoriaus šablono palaikyme, bet šie skirtumai yra smulkūs. Taip pat iš 1 lentelės matome, kad dalis kontrolės srautų šablono yra palaikomi BPMN, bet ne BPEL ir atvirkščiai. Tokiu būdu, apibrėžimai šių šablonų BPMN modelyje reikalauja ypatingo atidumo kai modelis verčiamas į BPEL. Vertimas iš BPMN į BPEL yra ne toks paprastas, kaip dažnai manoma.

Duomenų šablonų požiūriu palaikymas BPMN ir UML 2.0 AD yra truputį skiriasi. UML 2.0 AD yra stipresnis atvaizdavime daugialypių paleisčių duomenyse taip pat ir duomenų sąveikoje į ir iš daugialypių paleisčių užduočių, tuo metu, kai BPMN yra stipresnis duomenų sąveikoje tarp užduoties ir aplinkos, dėl to, kad aplinka gali būti tiksliai modeliuojama. Yra kitų skirtumų duomenų perkėlimo ir duomenimis pagrįstose paskirstymo šablonų kategorijų, taip pat kaip ir šablonų skirtumai atvaizdavimui naudojant BPEL. Tačiau, net jei rinkinys šablonų atvaizduotų šiuo požiūriu yra skirtingas kiekvienai kalbai ir net jei nė viena iš kalbų visiškai neatvaizduoja visų šablonų, gali būti teigiama, kad duomenų atžvilgiu šablonai yra pakankamai gerai paruošti.

Deja, to paties negalima pasakyti išteklius. Egzistuojančios sąvokos kelias (angl. *lane*) ir saugykla (angl. *pool*) BPMN modelyje parodo išteklių šablonų reikalingumą ir ketinimą toliau juos palaikyti. Tačiau tik minimalus išteklių šablonų komplekto palaikymas parodo šios kalbos nesubrendimą. BPMN naudai, galima pasakyti, kad išteklių palaikymas yra minimalus ir UML 2.0 AD. Tuo pačiu metu, BPEL išplėtimas skirtas atvaizduoti išteklių šablonus buvo pasiūlytas (pavyzdžiui. BPEL4People [23]), ir dalis šito išplėtimo yra įgyvendinta komerciniuose įrankiuose tokiuose kaip Oracle BPEL PM, tokiu būdu pabrėžiama būtinybė atvaizduoti šią perspektyvą. Bendrai, žiūrint iš perspektyvos BPMN ir UML 2.0 AD išteklių palaikymo stygius, prieštarauja tebevykstančiomis pastangomis BPEL bendruomenėje atkreipti dėmesį į šią perspektyvą, priešingybė BPEL bendruomenės dabartinėms pastangoms skirtoms šiai perspektyvai, atveria plyšį tarp šiuolaikinių proceso modeliavimo įrankių, ir proceso vykdymo variklių (pastarieji apskritai palaiko išteklių šablonus vienu ar kitu būdu). Kad būtų pasiektas nuoseklus ir susietas proceso modelių naudojimas, nuo analizės iki realizacijos ir priėmimo, svarbu, kad ištekliai būtų plačiau pripažystami kaip verslo proceso modeliavimo neatimama dalis. Užuot kūręs naują proceso modeliavimo žymėjimus, kuris didžia dalimi persikloja su egzistuojančiais kontrolės sekų srautais, turėtų greičiau susikoncentruoti ties tolimesniu egzistuojančių žymėjimu tobulinimo, kuris pakankamai plačiai apimtų visus aspektus taikomus PAI.

2.4. Elgsenos modeliavimo kalbų palyginimas

Šiame skyriuje pateikti įvertinimo rezultatai gauti iš išsamios kontrolės-srautų šablonų (angl. *control-flow patterns*) analizės[24], kurioje buvo palyginta keturiolika komercinių produktų. Produktų testavimas apėmė workflow sistemas, atvejų apdorojimo sistemą (angl. *Case Handling*), verslo proceso vykdymo kalbas ir verslo proceso modeliavimo formalizavimas. Specifiniai produktai/kalbos, kurios buvo nagrinėtos:

Staffware Process Suite 10;

IBM WebSphere MQ Workflow 3.4;

- FLOWer 3.5.1;
- COSA 5.1;
- Sun ONE iPlanet Integration Server 3.0;

SAP Workflow version 4.6c

- FileNet P8 BPM Suite version 3.5
- BPEL version 1.1;
- WebSphere Integration Developer 6.0.2, the development environment for the Business Process Choreographer (BPC) part of WebSphere Process Server v6.0.2;
- Oracle BPEL v10.1.2;
- BPMN version 1.0;
- XPDL version 2.0;
- UML 2.0 Activity Diagrams; and
- EPCs as implemented by ARIS Toolset 6.2.

Pabandysime apžvelgti naujoves ir naujas kryptis siūlomose produktuose.

Tradiciškai darbų sekos (angl. *workflow*) sistemos naudojo savo originalias įdėjas realizuojant laiko proceso modelius ir taip kaip tie modeliai turėjo elgtis veikimo metu. Pagrindo realizacija tokio kaip kontrolės gijos, jungimo semantika ir ciklai tarpusavyje smarkiai skyrėsi. Fundamentalus proceso modelis, naudotas specifiniams produktams, buvo ypač miglota sritis. Atrodo, tai pasikeitė, naujausiuose produktuose į labiau formalizuotai apibrėžti ir geriau suprantami modeliai naudojant BPEL, BPMN, XPDL ir UML 2.0 Activity Diagrams. Visi teigia, kad pagrindinis vykdymo modelis yra „token-based“.

Nors papildyti apibrėžimai ir originalūs šablonai turi šiek tiek daugiau apribojimų– tokiems šablonams kaip struktūrinis sinchroninis apjungimas (angl. *Structured Synchronizing Merge*), struktūrinis diskriminatorius (angl. *Structures Discriminator*), daugialypės paleistys su laikinio projektavimo žiniomis (angl. *Multiple Instances with Design-Time Knowledge*) ir persidengiantis lygiagretus paskirstymas (angl. *Iterleaved Parallel Routing*) –yra plačiai

palaikomi. Visi pagrindiniai šablonai (WCP1 iki WCP-5) yra vis dar palaikomi išanalizuotų produktų.

Peržiūrėtas struktūrinio sinchroninio jungimo apibrėžimas labiau tinkamas struktūrizuotoms kalboms tokioms kaip WebSphere MQ ir BPEL nors šablonas yra palaikomas COSA, FileNet, BPMN, XPDL ir EPC nepriklausomai nuo apribojimų kaip šį šabloną būtų galima panaudoti. Lankstesni šio šablono variantai gali būti apibrėžti naudojant neciklinį sinchronizuojantį jungimą (angl. *acyclic synchronizing merge*) ir bendrinį sinchronizuojantį jungimą. Abi šios formos yra palaikomas tik minimaliai. Taip pat galima pastebėti, kad tai galioja ir struktūrizuotam diskriminatoriui ir struktūriniam daliniam jungimui (angl. *Structured Partial Join*), kurie turi minimalų palaikymą tarp produktų su realia vykdymo aplinka ir kitos lanksčios šių šablonų formos (blokavimas, atšaukimo diskriminatorius (angl. *Blocking Discriminator*) ir atšaukiamas dalinis jungimas (angl. *Cancelling Partial Join*), kurie iš viso turi labai menką palaikymą iš esmės.

Lentelė 2.4 Produktų įvertinimas – originalūs šablonai (WCP1 – WCP20)

Šablonas	Staffware	WebSphere MQ	FLOWer	COSA	iPlanet	SAP Workflow	FileNet	BPEL	WebSphere BPEL	Oracle BPEL	BPMN	XPDL	UML AD	EPC
1 Seka	+	+	+	+	+	+	+	+	+	+	+	+	+	+
2 Lygiagretus išsišakojimas	+	+	+	+	+	+	+	+	+	+	+	+	+	+
3 Sinchronizacija	+	+	+	+	+	+	+	+	+	+	+	+	+	+
4 Išskirtinis pasirinkimas	+	+	+	+	+	+	+	+	+	+	+	+	+	+
5 Paprastas jungimas	+	+	+	+	+	+	+	+	+	+	+	+	+	+
6 Daugialypis pasirinkimas	-	+	+	+	+	-	+	+	+	+	+	+	+	+
7 Paprast ciklas	-	+	+	-	-	-	+	+	+	+	+	+	-	+
8 Daugialypis sujungimas	-	-	+/-	+/-	+	-	+	-	-	-	+	+	+	-
9 Paprastas diskriminatorius	-	-	-	-	+	+/-	-	-	-	-	+/-	+/	+/-	-
10 Pasirenkamieji ciklai	+	-	-	+	+	-	+	-	-	-	+	+	+	+
11 Vienintelis pasirinkimas	+	+	+	-	-	-	+	+	+	+	+	+	+	+
12 Daugelio egzempliorių be sinchronizacijos	+	-	+	+	+	+/-	+	+	+	+	+	+	+	-
13 Daugelio egzempliorių su laiku	+	-	+	-	-	+	-	-	-	+	+	+	+	-
14 Daugelio egzempliorių veikimas realiu laiku	+	-	+	-	-	+	-	-	-	+	+	+	+	-
15 Daugelio egzempliorių	-	-	+	-	-	-	-	-	-	-	-	-	-	-
16 Atidėto pasirinkimo	-	-	+	+	-	-	+/-	+	+	+	+	+	+	-
17 Persidengiantis paskirstymas	-	-	+/-	+	-	-	-	+/-	+/-	-	-	-	-	-
18 Tikslų	-	-	+/-	+	-	-	-	-	-	-	-	-	-	-
19 Atšaukti būseną	+	-	+/-	+	+	+	+	+	+	+	+	+	+	-
20 Atšaukti veiklą	-	-	+/-	-	-	+	+	+	+	+	+	+	+	-

Įdomūs pastebėjimai kyla iš šių šablonų nepriklausomai nuo teiginių apie verslo modeliavimo apibrėžto žymėjimo kalbų egzistavimą BPMN ir XPDL iki vykdomosios kalbos BPEL, yra daugybė šablonų tokių kaip daugialypis jungimas (angl. *Multi-Merge*), visos diskriminatoriaus formos, dalinis jungimas ir sprendimo ciklai (angl. *Arbitrary Cycles*), kurie yra palaikomi ankstesnėse kalbose, bet nepalaikomi BPEL iškyla klausimas kaip šie šablonai galėtų būti realizuoti. Šis pastebėjimas su kitais tyrimais ir atspindi faktą, kad modeliavimo kalbos geriau kuria šablonus nei realūs veikiantys produktai, kadangi jie nesusiduria su iššūkiu realiai realizuoti tai ką sugeba modeliuoti.

Bendrai vykdymo rezultatai dviejų BPEL atstovų – WebSphere Integration Developer ir Oracle BPEL parodė, kad jie suteikia reliatyviai patikimą realizaciją BPEL specifikacijos. Viena pažymėtina išimtis yra tai, kad papildoma <flowN> konstrukcija esanti Oracle BPEL, kuri leidžia realizuoti keletą daugialypių paleisčių šablonų, kurių kiti BPEL variantai nesugeba palaikyti.

Sprendimo ciklų šablono forma (angl. *Arbitrary Cycles*) nėra palaikoma tokiose blokinės struktūros kalbose tokiose kaip WebSphere MQ ar BPEL. Nors labiau griežtesnė atkartojimo forma – struktūrinis ciklo šablonas yra plačiai palaikomas daugelyje produktų įskaitant WebSphere MQ, FLOWer, iPlanet, FileNet, BPEL, BPMN ir XPDL.

Neskaitant daugialypių paleisčių be sinchronizacijos (angl. *Multiple Instances without Synchronization*) šablono, kuris yra plačiai palaikomas, toliau tęsiasi minimalus palaikymas valdymui skirtų kontroliuojamų veiklų lygiagretumo su įvairiomis jos formomis. Staffware, FLOWer ir Oracle BPEL yra vieninteliai, kurie dalinai palaiko įvairius daugialypės paleisties šablonus, kol BPMN ir XPDL suteikia galimybę sukurti juos laikinio projektavimo modeliuose (Design-time Models).

Aiškaus užbaigimo šablonas buvo pristatytas tam, kad apibrėžtų situacijas, kai yra numatyta užbaigimo viršūnė procese, nei, kad laikoma, kad proceso paleistis pasibaigia kai nebėra daugiau darbo (angl. *Implicit Termination*). Daugelis produktų palaiko vieną ar kitą iš šių šablonų, bet gerai yra tai, kad BPMN, XPDL ir UML 2.0 AD palaiko abu šablonus.

Kalbant apie būsenų mašinų šablonus, atidėto pasirinkimo (angl. *Deferred Choice*) efektyvi realizacija tinkama produktuose, kurie turo simbolinį pagrindą (angl. *token based*) tokiuose kaip COSA, BPEL, BPMIN, XPDL ir UML 2.0 AD. Tik COSA tiesiogiai palaiko persidengiantį lygiagretų paskirstymą, bendru atveju atrodo kad integruotas būsenos suvokimas yra būtinas norint efektyviai realizuoti šį šabloną. Panašus komentarai gali būti pritaikyti tikslų (Milestone) šablonui. Kai dalinis užsakymo reikalavimas yra ne toks griežtas ir leidžia savavališką vykdymo tvarką (angl. *Interleaved Routing*), BPEL (bet ne Oracle BPEL) taip pat tai palaiko. Būtina užtikrinti kad veiklos yra veikiančios ne lygiagrečiai

FLOWer produkte nėra. Bet tikro lygiagretumo trūkumas (kitokio nei persidengimas) neleidžia pilnai palaikytį šį šabloną. Adhoc veiklos konstrukcija BPMN ir XPDL suteikia galimybę netiesiogiai panaudoti šį šabloną, bet tai negarantuoja kad ši veikla veiks bent kartą tiksliai. Kritinės sekcijos šablonas yra palaikomas tik COSA ir BPEL.

Atšaukimo veikla yra plačiai palaikoma nors WebSphere MQ yra išimtis. Skirtumas tarp veiklos pašalinimo dar prieš jai įvykstant (palaikoma Staffware, COSA, SAP) ir atšaukimo viduryje vykdymo (palaikoma BPEL, BPMN, XPDL ir UML 2.0 AD). Panašiai su atšaukimo atveju (angl. *Cancel Case*) šablonu, kuris plačiai palaikomas (SAP, FileNet, BPEL, BMPN, XPDL, UML 2.0 AD), bet atšaukimas sprendimo regiono (angl. *Cancel Region*) nėra pilnai palaikoma UML 2.0 AD.

Lentelė 2.5 Produkto įvertinimai – nauji šablonai (WCP21 – WCP43)

Šablonas	Staffware	WebSphere MQ	FLOWer	COSA	iPlanet	SAP Workflow	FileNet	BPEL	BPELWebSphere	Oracle BPEL	BPMN	XPDL	UML AD
21 Struktūrinis ciklas	-	+	+	-	+	+	+	+	+	+	+	+	-
22 Rekursija	+	+	-	+	+	+	-	-	-	-	-	-	-
23 Laikinas trigeris	+	-	-	+	-	+	-	-	-	-	-	+	-
24 Pastovus trigeris	-	-	+	+	-	+	+	+	+	+	+	+	+/-
25 Atšaukimo sritis	-	-	-	+/-	-	-	-	+/-	+/-	+/-	+/-	+	-
26 Atšaukimas daugelio veiklų egzempliorių	+	-	-	-	+	-	-	-	+	+	+	+	-
27 Pilna daugelio egzempliorių veikla	-	-	+/-	-	-	-	-	-	-	-	-	-	-
28 Blokuojamas diskriminatorius	-	-	-	-	-	-	-	-	-	+/-	+/-	+/-	-
29 Atšaukimo diskriminatorius	-	-	-	-	-	+	-	-	-	+	+	+	-
30 Struktūrinis dalis jungimas	-	-	-	-	+	+/-	-	-	-	+/-	+/-	+/-	-
31 Bluokuojamas dalinis jungimas	-	-	-	-	-	-	-	-	-	+/-	+/-	+/-	-
32 Dalinis atšaukimo jungimas	-	-	-	-	+	-	-	-	-	+/-	+/-	+	-
33 Bendrinis and-join	-	-	-	-	-	+	-	-	-	+	+	-	+/-
34 Daugelio egzempliorių statinis dalinis jungimas	-	-	-	-	-	-	-	-	-	+/-	+/-	-	-

35 Daugelio egzempliorių dalinis atšaukimo jungimas	-	-	-	-	-	-	-	-	-	+/-	+/-	-	-
36 Daugelio egzempliorių dinaminis dalinis atšaukimo jungimas	-	-	-	-	-	-	-	-	-	-	-	-	-
37 Aciklinis sinchronizuojamas jungimas	-	+	+	+	-	-	+	+	+	-	-	+/-	+
38 Bendras sinchronizuojamas jungimas	-	-	-	-	-	+	-	-	-	-	-	-	-
39 Kritinė sekcija	-	-	+/-	+	-	-	-	+	+	+	-	-	-
40 Persidengiantis skirstymas	-	-	+/-	+	-	-	-	+	+	+	-	-	-
41 Gijų jungimas	-	-	-	-	-	-	+/-	+/-	+/-	+	+	+	-
42 Gijų dalinimas	-	-	-	-	-	-	+/-	+/-	+	+	+	+	-
43 Išskirtinis nutraukimas	-	-	-	+	+	+	-	-	-	+	+	+	-

Trigeriais pagrįsti šablonai, kurie pasiūlo galimybę išoriniams faktoriams kontroliuoti proceso vykdymą yra plačiai palaikomi. Staffware realizavo trumpalaikius trigerius (angl. *transient*), pastovūs trigeriai buvo pasiūlyti FileNet, BPEL, BPMN ir XPDL. COSA, SAP workflow ir UML 2.0 AD realizuoja abu trigerių tipus.

2.5. Analizės išvados

Remiantis literatūros šaltinių analize galime sakyt, kad BPMN turi susitelkia ties verslo procesais, UML turi koncentruojasi ties programinės įrangos projektavimu, dėl to turime skirtingus požiūrius apie skirtingus požiūrius apie sistemas[17].

Nagrinėjant šablonus nustatyta, kad modeliavimo kalbos geriau kuria šablonus nei realūs veikiantys produktai, kadangi jie nesusiduria su iššūkiu realiai realizuoti tai ką sugeba modeliuoti. Kalbant apie būsenų mašinų šablonus, atidėto pasirinkimo (angl. *Deferred Choice*) efektyvi realizacija tinkama produktuose, kurie turo simbolinį pagrindą (angl. *token based*) tokiuose kaip COSA, BPEL, BPMIN, XPDL ir UML 2.0 AD.

UML 2 vartotojo apibrėžtas elgsena yra klasė. Kiekvieną kartą kai elgsena yra vykdoma veikimo metu, nauja egzempliorius vartotojo elgsenos klasė yra sukuriama. Egzempliorius (angl. instance) yra sunaikinamas, kai elgsena baigiama. Elgsenos klasės, kaip visos klasės, gali palaikyti atributus, asociacijas, operacijas, ir net kitas elgsenas, tokias kaip būsenų mašinos. Tai atspindi įprastą praktiką sistemose, kurios valdo procesus, pavyzdžiui,

darbų sekas (angl. workflow) ir operacines sistemas. Programos gali taip pat patalpinti būsenų mašinas į elgesio klasę, kad padėtų aprašyti kiekvieną vykdymo būseną.

3. UML elgsenos modeliais pagrįsta informacinių sistemų kūrimo metodika

3.1 Metodikos aprašymas

IS projektavimo, pagrįsto UML elgsenos modeliais, metodas susideda iš šių pagrindinių žingsnių:

1. Identifikuojamos ir sukuriamos dalykinės srities esybių klasės;

2. Analizuojamas procesas ir sukuriamos pagrindinės proceso veiklos;

3. Identifikuojamos šių veiklų įėjimų ir išėjimų esybių būsenos ir sukuriamos pradinės esybių būsenų mašinos;

4. Analizuojami alternatyvūs scenarijai ir nustatomi alternatyvūs veiksmai, kurie vykdomi nesėkmingų rezultatų atveju, papildomos esybių būsenų mašinos;

5. Sugeneruojama proceso valdymo klasių diagrama su operacijomis, kurios vaizduoja veiklos proceso metu vykdomus veiksmus.

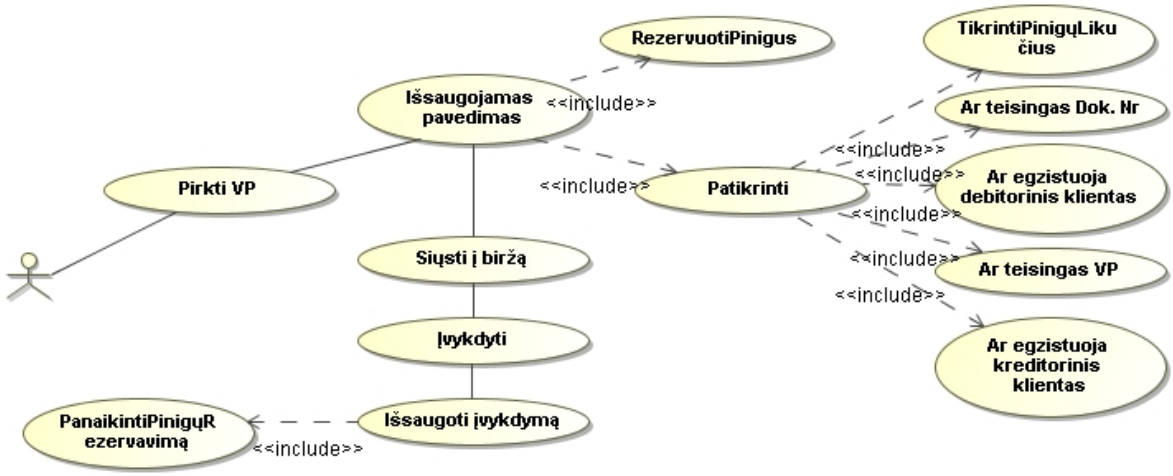
Kadangi UML CASE įrankiai gali tik dalinai užtikrinti modelių suderinamumą, projektuotojas turi papildomai patikrinti modelių suderinamumą ir jų atitikimą dalykinei sričiai. Nepaisant to, modeliavimo metodika padeda supaprastinti projektavimą ir sumažinti galimų projektavimų klaidų skaičių. Gautą klasių diagramą galima modifikuoti, kad atitiktų reikalavimus ir panaudojus kodo generavimo įrankius gauti sugeneruotą kodo šabloną.

UML elgsena yra klasė, todėl iš sumodeliuotos veiklos diagramos galima gauti valdymo klases, realizuojamas programine įranga. Pasitelkus šia metodiką ir modeliuojant veiklos procesus ir esybes galima tiksliai susieti esybes, jų būsenas, veiklos procesus ir valdymo klases.

3.2 Metodikos taikymas vertybinių popierių apskaitos informacinei sistemai projektuoti

3.2.1 Informacinės sistemos pradiniai panaudos atvejai

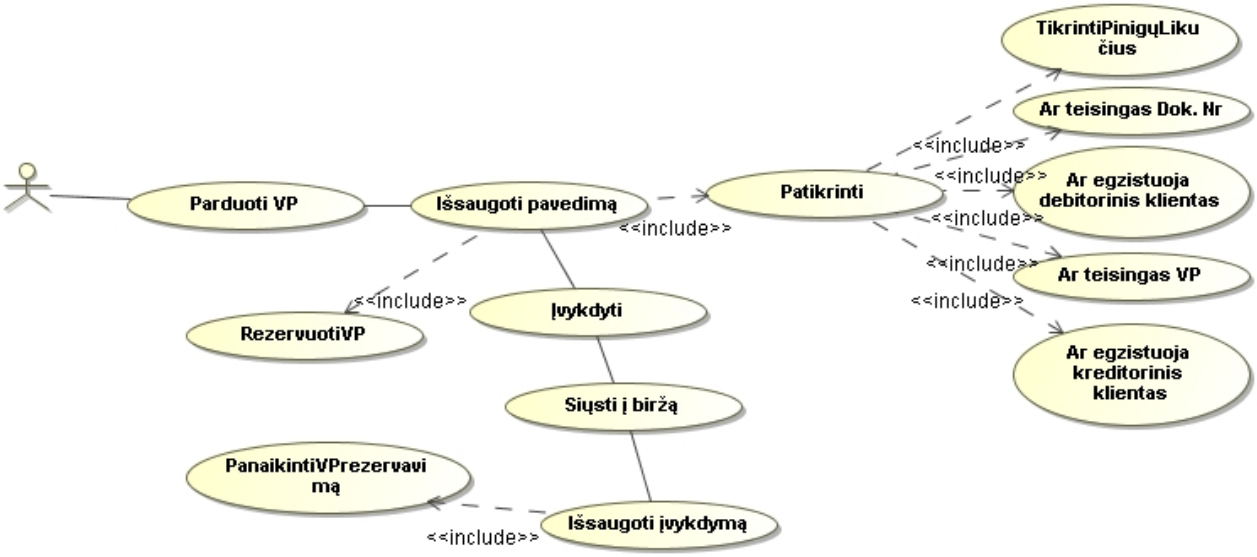
VP pirkimo panaudos atvejis nusako kliento VP pirkimą. Vartotojui perkant VP atliekami pagrindiniai žingsniai: išsaugojamas pavedimas, siuntimas į biržą, įvykdymas. Išsaugojant pavedimą patikrinami pavedimai.



3.40 pav. VP pirkimo panaudos atvejis

VP pardavimo panaudos atvejis nusako VP pardavimo veiklą. Kaip ir VP pirkimo veiklos atveju išsaugojamas pavedimas. Saugojant pavedimą patikrinami pavedimo duomenys, rezervuojami VP. Įvykdomos pavedimas, pavedimas išsiunčiamas į biržą ir

išsaugojamas įvykdymas.

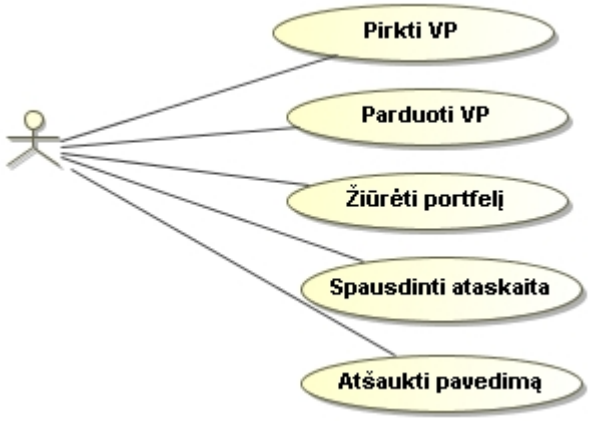


3.41 pav. VP pardavimo panaudos atvejis

Atšaukimo veikla vykdoma tik, kai pavedimas buvo įvykdytas.

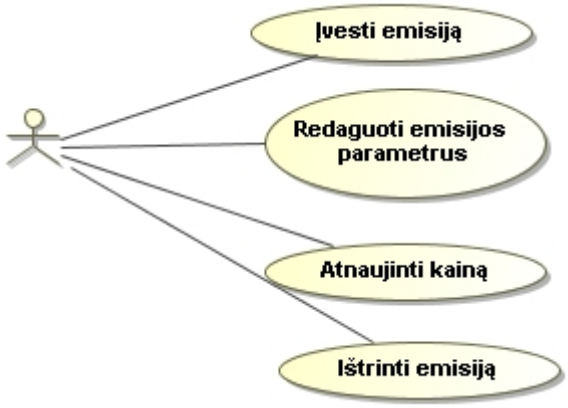
3.42 pav. Pavedimo atšaukimo panaudos atvejis

Klientas sistemoje turi turėti galimybę atlikti pagrindines funkcijas tokias kaip pirkti, parduoti ir atšaukti VP. Klientas taip pat gali spausdinti ataskaitas ir peržiūrėti savo vertybinių popierių portfelį.



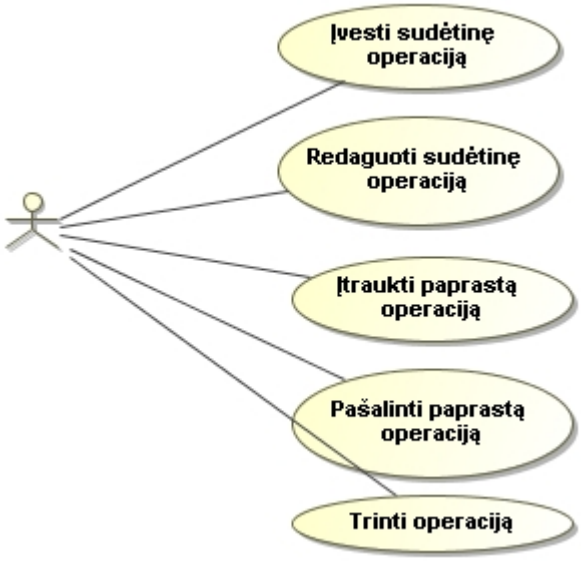
3.43 Kliento panaudos atvejai

Vertybiniai popieriais nusakomos finansinės priemonės, kuriomis prekiaujama vertybinių popierių biržoje. VP gali būti ir kuriais prekiaujama užbiržinuose sandoriuose. Sistemoje svarbu, kad vartotojai turėtų reikiamų emisijų parametrus ir galėtų keisti kainas.



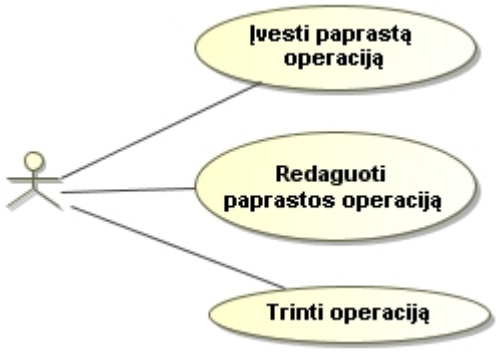
3.44 Vertybinio popieriaus panaudos atvejai

Sudėtinėmis operacijomis nusako kokia operacija bus atliekama, ar tai bus pirkimas ar pardavimas. Sudėtinę operacija sudaro paprastos operacijos. Dėl to kuriamoje informacinėje sistemoje būtina užtikrinti galimybę vartotojams keisti sudėtinės operacijos paprastas operacijas.



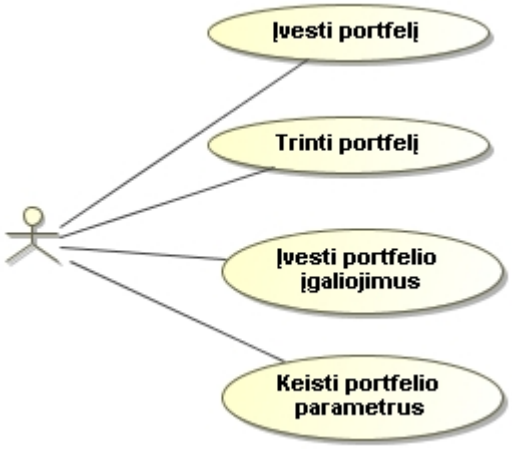
3.45 Sudėtinė operacijos panaudos atvejai

Paprastomis operacijomis nusakoma pagrindiniai VP apskaitos parametrai, kokia informacija bus saugojama. Paprastos operacijos į sudėtinę įtraukti panaudos atvejų diagramoje nėra, nes įtraukimas paprastų operacijų į sudėtines atliekamas redaguojant sudėtinę operaciją.



3.46 Paprastos operacijos panaudos atvejai

Kiekvienas klientas turi vertybinių popierių portfelyje. Šiame portfelyje laikomi VP ir kliento pinigai. Portfelių klientas gali turėti kelis. Portfelio įgaliojimai reikalingi skirtingoms veikloms. Pvz. Klientas turi savo asmeninį portfelį, kurio VP gali pirkti ir parduoti be įgaliojimo. Kitas kliento portfelis yra bendra jungtinė nuosavybė tokiu atveju klientas negali parduoti VP, jei neturi įgaliojimo.



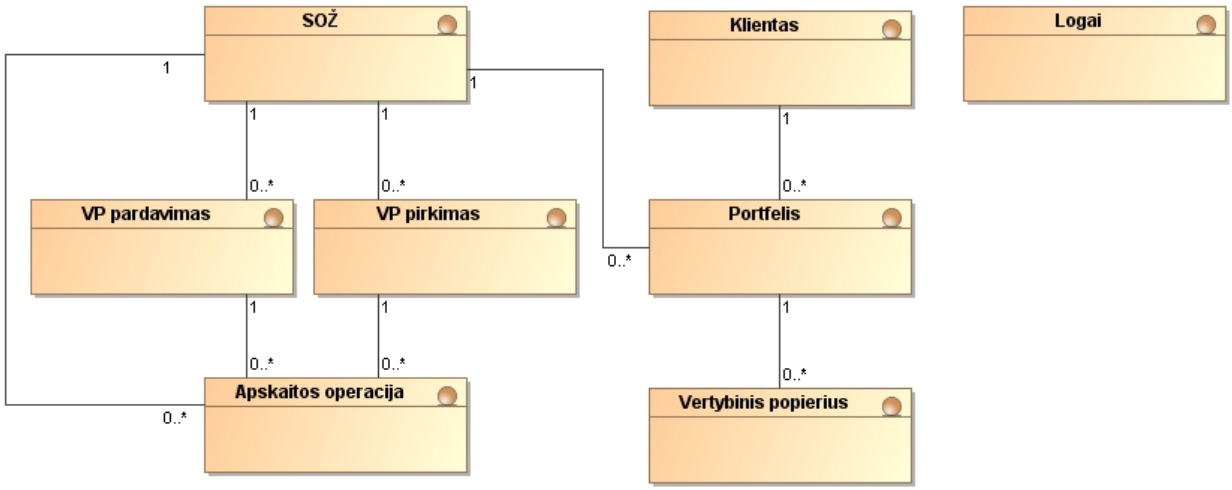
3.47 Portfelio panaudos atvejai

3.2.2 Esybių klasės

UML metodikoje pradinis žingsnis yra išskirti esybių klases.

Sistema susideda iš tokių pagrindinių esybių klasių:

- VP – vertybinių popierių;
- Apskaitos operacijų;
- Pirkimo;
- Pardavimo;
- Portfelio;
- SOŽ (Sudėtinių operacijų žurnalo);
- Klientų;
- Logų;



3.48 pav. Esybių klasių modelis

Šiomis klasėmis nusakoma kuriamos informacinės sistemos veikla. VP klasė skirta aprašyti vertybinius popierius naudojamus biržoje ir už biržos ribų (užbiržiniai sandoriai).

Apskaitos operacijos klasė nusako apskaitos operacijų veikimą. Kiekvienam pirkimo ar pardavimo sandoriams apskaityti naudojamas iš anksto nustatytos operacijos.

Atitinkamai remiantis operacijos konfigūracija informacija saugojama sudėtinių operacijų žurnale (SOŽ). Sudėtinių operacijų žurnalas skirtas saugoti apskaitos operacijų įrašus pagal depozitoriumo reikalavimus. Be informacijos, kurią būtina saugoti pagal depozitoriumo reikalavimus, saugojami sandorio parametrai, kliento portfelis.

Kliento klasė nusako kliento funkcionalumą. Sistemos ypatumai: klientas gali būti juridinis ar fizinis asmuo, ir kartu gali būti darbuotojas. Darbuotojai nėra atskira esybė.

Portfelio esybė skirta nusakyta portfelyje esančius VP (vertybinius popierius), pinigus. Portfelių klientas gali turėti neribotą kiekį.

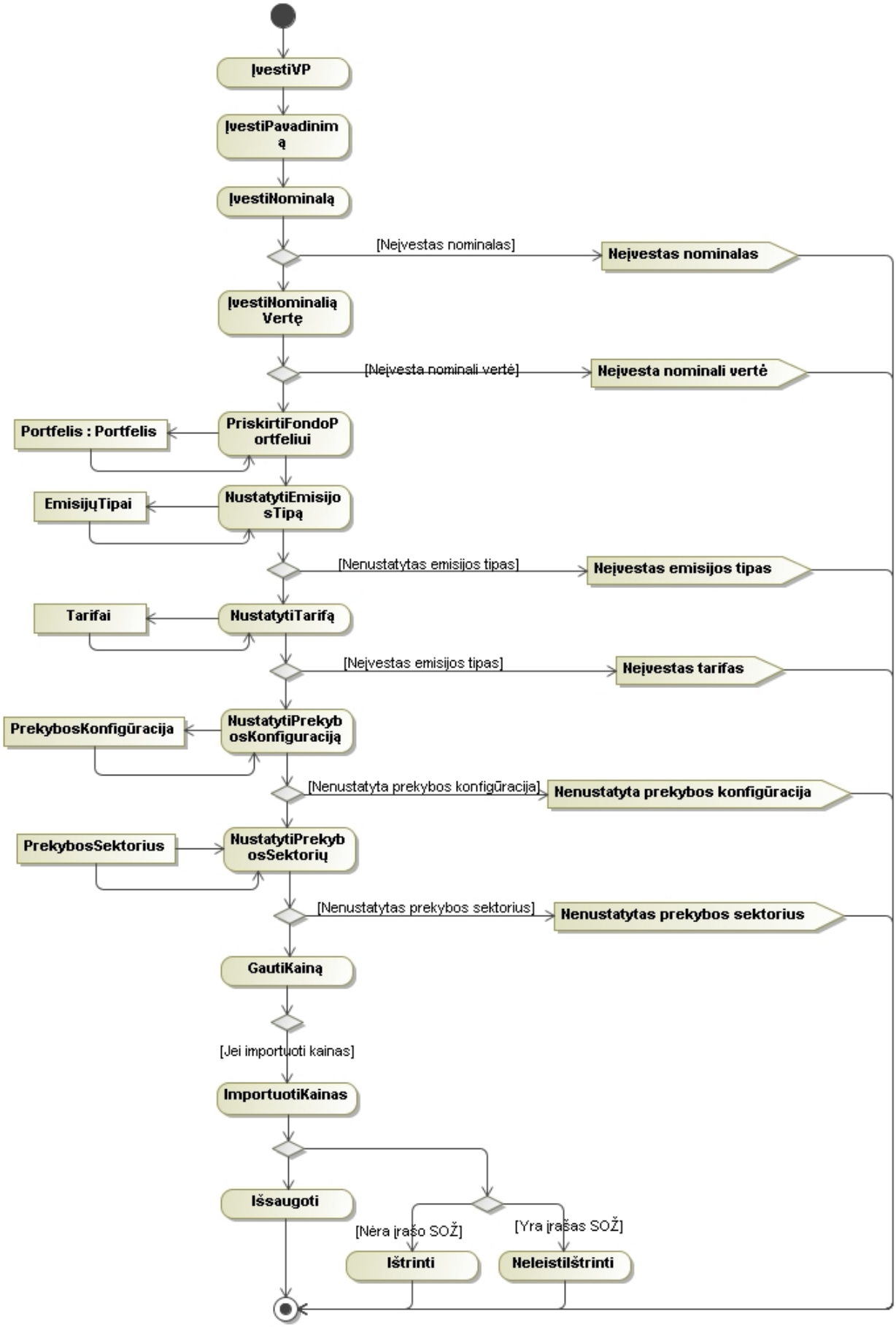
Pirkimo ir pardavimo operacijų esybėmis nusakoma pirkimo/pardavimo veikimas. Nespecifikuota sistemos dalis remiasi šių esybių sugeneruotų duomenų apdorojimu. Duomenys naudojami formuojant sąskaitas ir išrenkant duomenis klientams.

Informacinės sistemos modelis bus pateiktas pagal pateiktos metodikos žingsnius. Modelyje pateikiamas esybės veiklos procesas, veiklos proceso būsenų mašina su alternatyviais scenarijais ir informacinės sistemos langais.

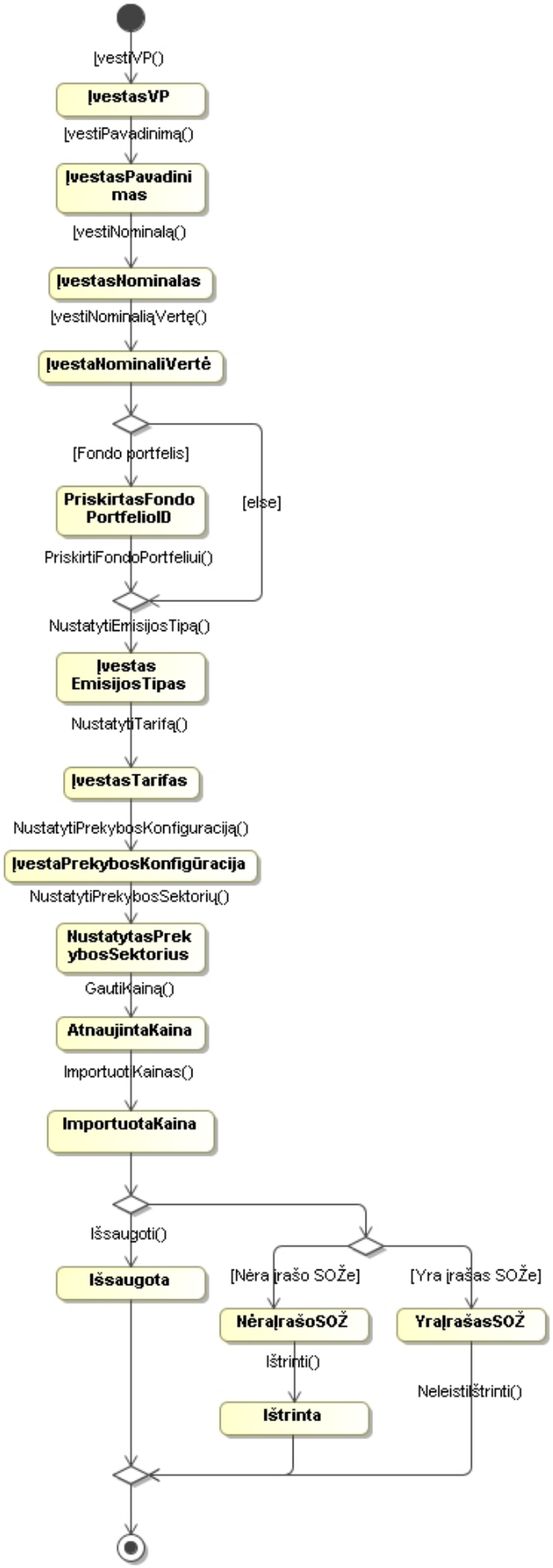
3.2.3 Vertybinis popierius

Vertybinių popieriaus apskaitos informacinė sistema dirba su vertybinių popierių prekyba ir apskaita. Dėl to nusakyti vertybinių popierių apskaitos veiklos procesą sukurta

vertybinio popieriaus esybės klasė. Pirmame paveiksle pateikta vertybinio popieriaus veikla. Vertybinio popieriaus klasėje saugojama pagrindinė informacija susieta su vertybiniais popieriais. Vertybiniai popieriai turi atskirą prekybos konfigūracija. Konfigūracija reikalinga nusakyti kokioje rinkoje ir kokiomis operacijomis atliekamas pirkimas pardavimas. Tarifai priskiriamas kiekvienam VP, taip galima užtikrinti skirtingus tarifus finansiniams instrumentams, kuriais prekiaujama užsienio biržose. VP kuris yra listinguojamas ir vykdoma prekyba biržoje turi kainą, kuri iš biržos gaunamų duomenų yra atnaujinimą. Jei VP kuriais prekiaujama biržoje, su kuria nėra sąsajos galima importuoti kainas iš kitų sistemų, taip realizuotas kainų importas iš Bloomberg prekybos sistemos. Nereikalingas emisijas galima ištrinti. Trinant tikrinama ar yra sudėtinių operacijų žurnale (SOŽ) operacija su VP, kuris yra trinamas, jei yra trinti emisijos negalima. Taip užtikrinama, kad duomenys apie operacijas būtų pilni.



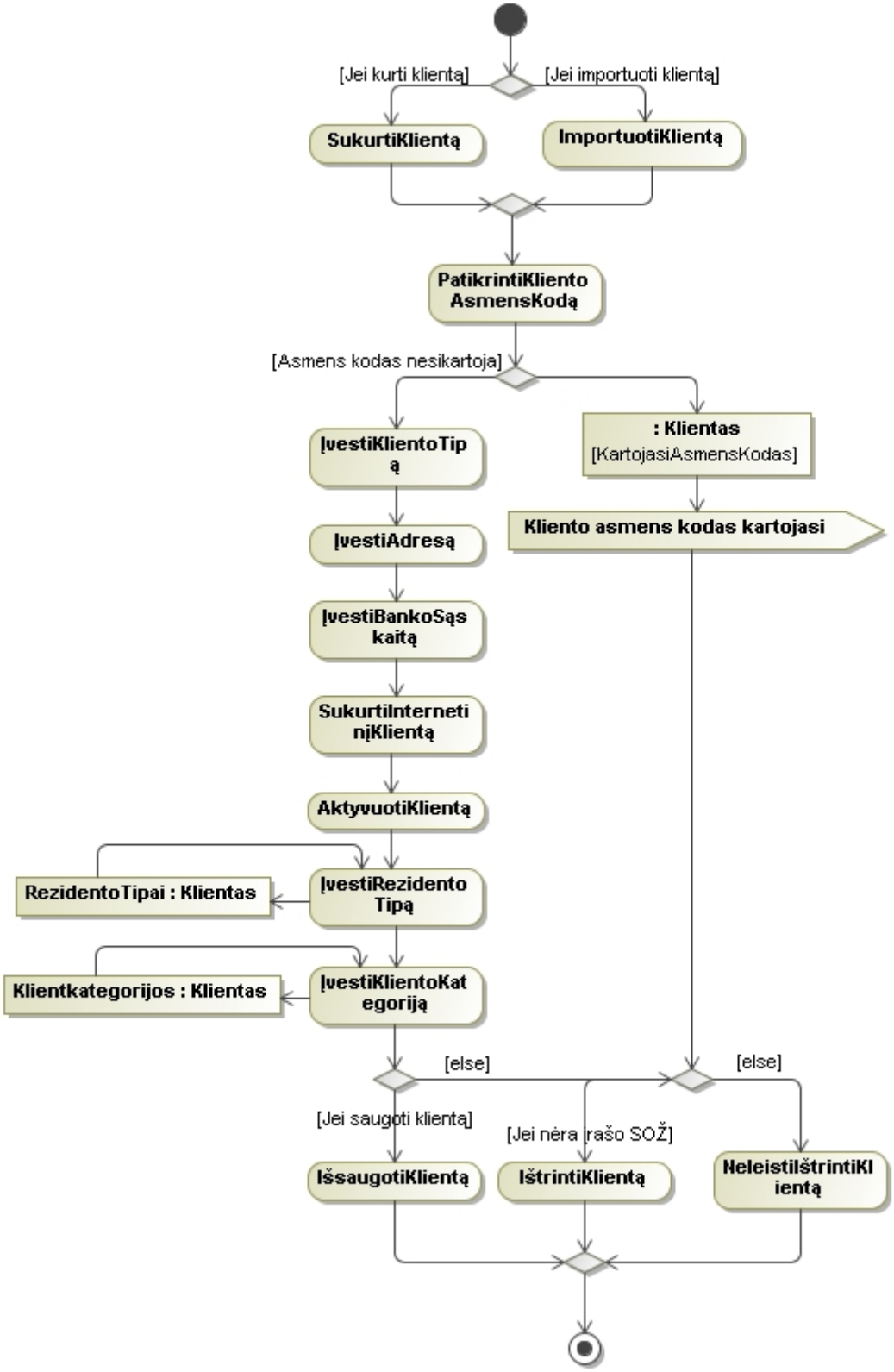
3.49 pav. Vertybinio popieriaus veikla



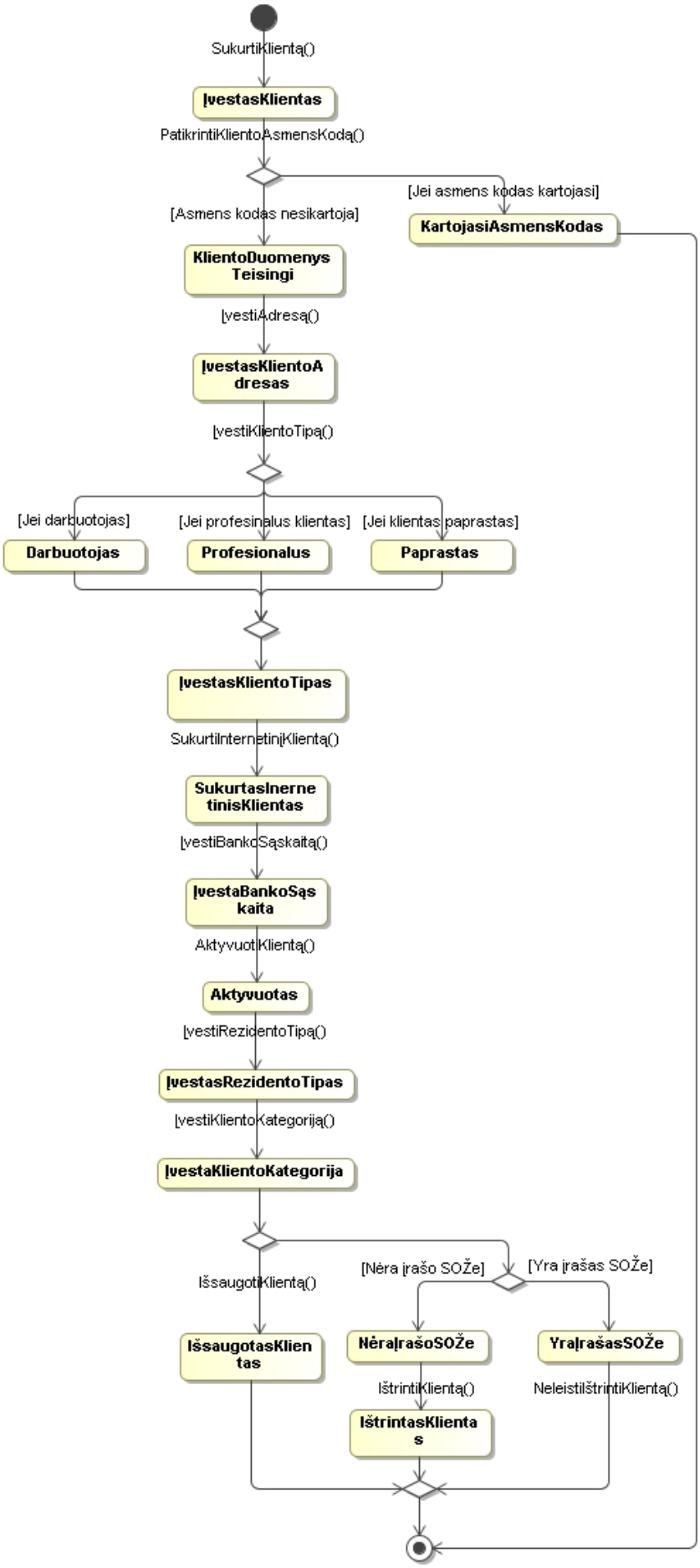
3.50 pav. Vertybinio popieriaus būsenos

3.2.3 Klientas

Projektuojamoje informacinėje sistemoje kliento esybė nusako ne tik klientus, tai yra fizinius ir juridinius asmenis, bet ir darbuotojus. Visa informacija apie klientus ir darbuotojus laikoma kliento esybėje. Sistemoje numatyta galimybė importuoti klientus. Vartotojų unikalumas užtikrinamas tikrinant asmens kodą. Sistemoje nėra numatyta galimybės, kad asmens kodai kartosis. Sistemoje taip pat realizuotas kliento atskirimas nuo VP, vertybiniai popieriai laikomi portfelyje, todėl kuriant naują klientą jam reikės sukurti naują portfelį. Portfelių klientas gali turėti kelis ir daugiau. Klientas galės naudotis programa tik tada, kai jis bus aktyvus. Todėl kiekvieną naują klientą reikia aktyvuoti.



3.51 pav. Kliento veiklos modelis

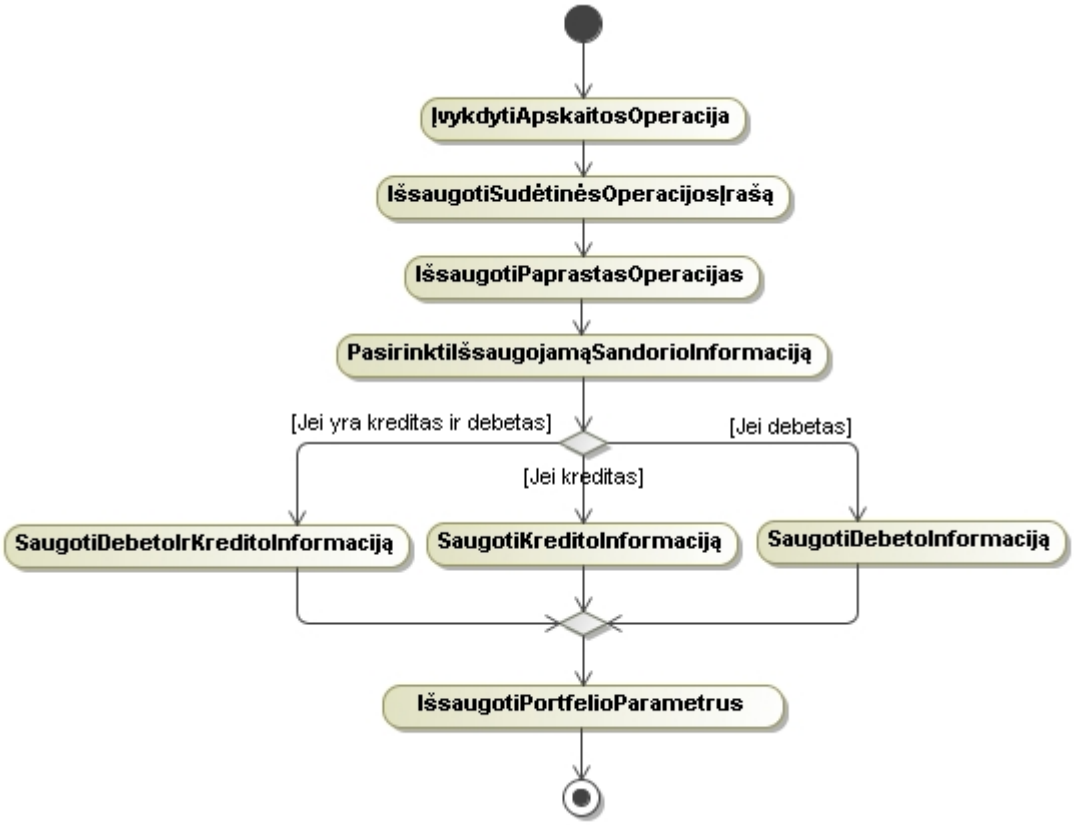


3.52 pav. Kliento būsenos

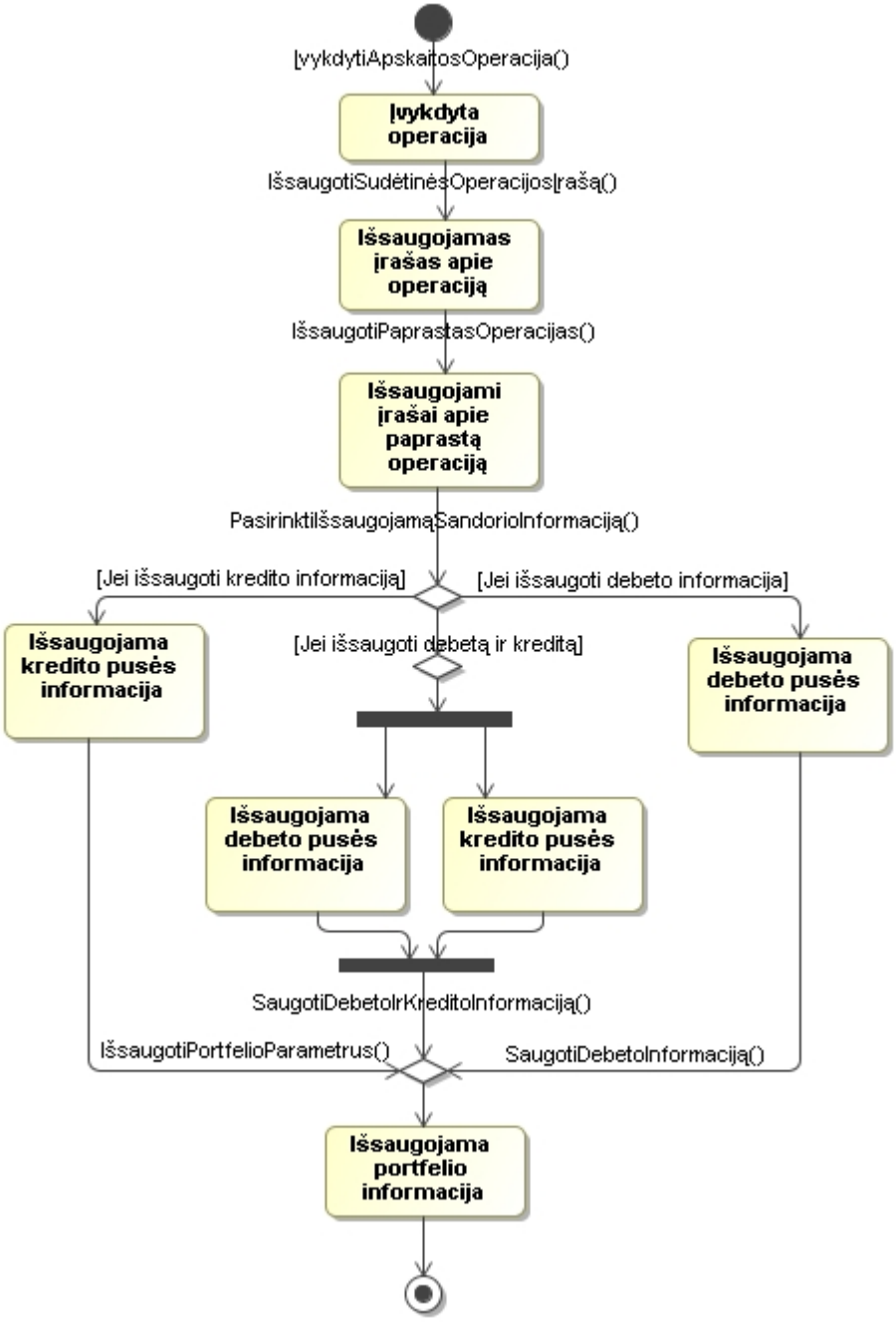
3.2.4. Sudėtinių operacijų žurnalas (SOŽ)

Sudėtinių operacijų žurnale saugojama informacija apie visas įvykdytas operacijas. Ši informacija vėliau panaudojama formuojant ataskaitas. Sudėtinių operacijų žurnalas taip pat užtikrina duomenų vientisumą. Negalima trinti sudėtinės ar paprastos operacijos, jei tokia operacija buvo įvykdyta ir yra informacija apie operaciją sudėtinių operacijų žurnale. Operacijų įvykdymų informacija yra saugojama dėl depozitoriumo reikalavimų. Kiekvieno kliento atlikto sandorio informacija turi būti saugojama 10 metų.

Sistemoje realizuota galimybė atlikti du sandorius vienu metu, tai labai patogu atliekant tam tikras operacijas tokias kaip užbiržiniai sandoriai, VP pervedimas iš įmonės sąskaitos klientui ir t. t. Dviejų sandorių kūrimas realizuojamas įvedant debeto ir kredito puses į pavedimo parametrus. Debeto pusė parduoda, o kredito klientas perka VP. Ar bus atliekamas sandoris su debeto ir kredito klientu nustatoma paprastos operacijos konfigūracijoje, tokiu atveju į sudėtinių operacijų žurnalą bus saugojami du susiję įrašai, tai yra kredito kliento pirkimo sandoris ir debeto kliento pardavimo sandoris.



Pav. 3.53 Sudėtinių operacijų žurnalas (SOŽ) veikla



3.54 pav. Sudėtinių operacijų žurnalo būsenos

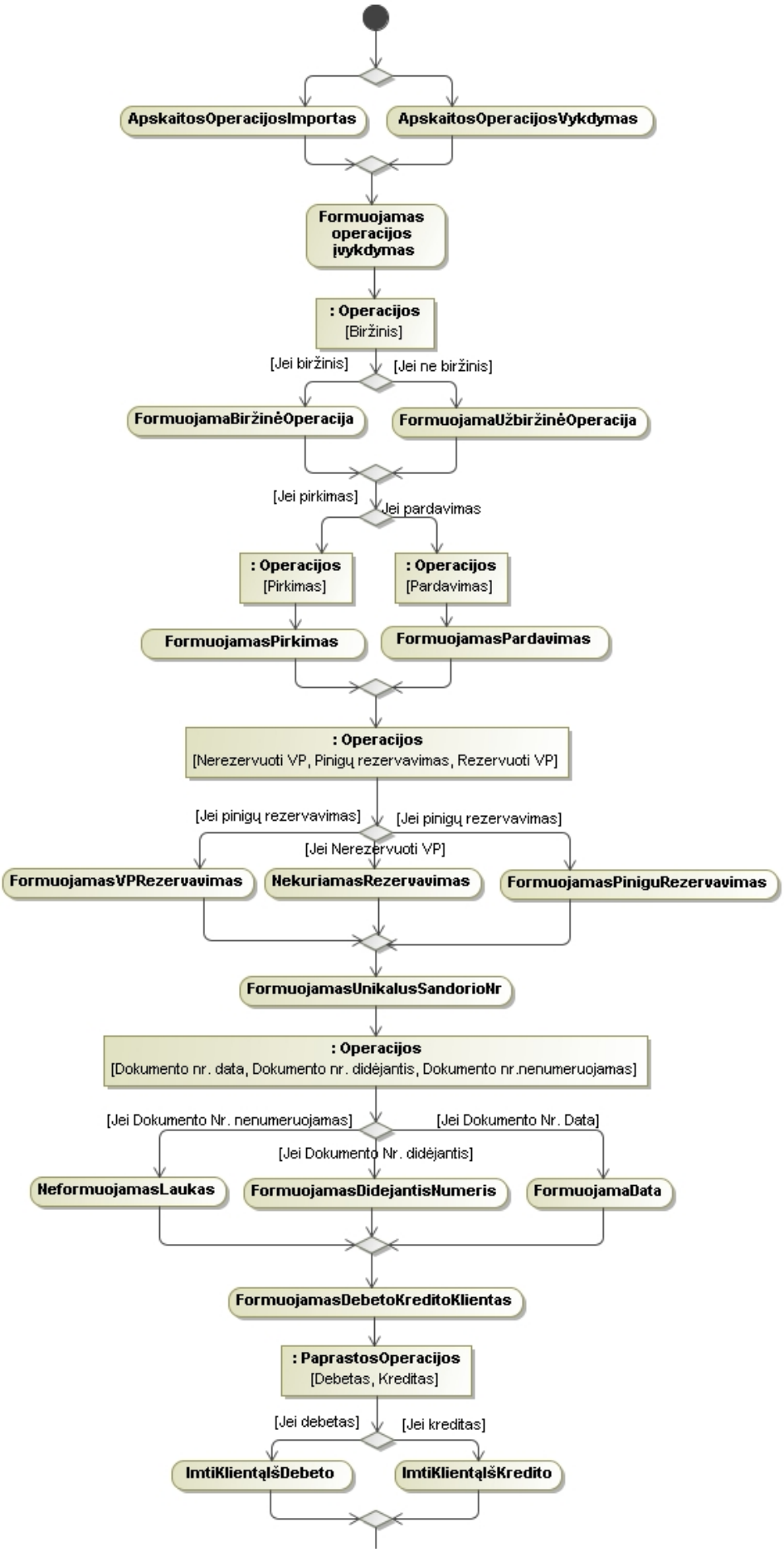
3.2.5. Operacija

Operacijomis aprašomi visi veiksmai susieti su VP prekyba ir apskaita. Tai realizuojama naudojant operacijų junginius, tai yra sudėtinių ir paprastų operacijų poros. Sudėtinė operacija gali turėti daug paprastų operacijų. Sudėtinė operacija nustatomi pagrindiniai parametrai reikalingi sandoriui suformuoti. Šios operacijos konfigūracija nustatoma ar sandoris pirkimo, pardavimo, bus rezervuojami VP ar pinigai, kaip bus numeruojami sandoriai.

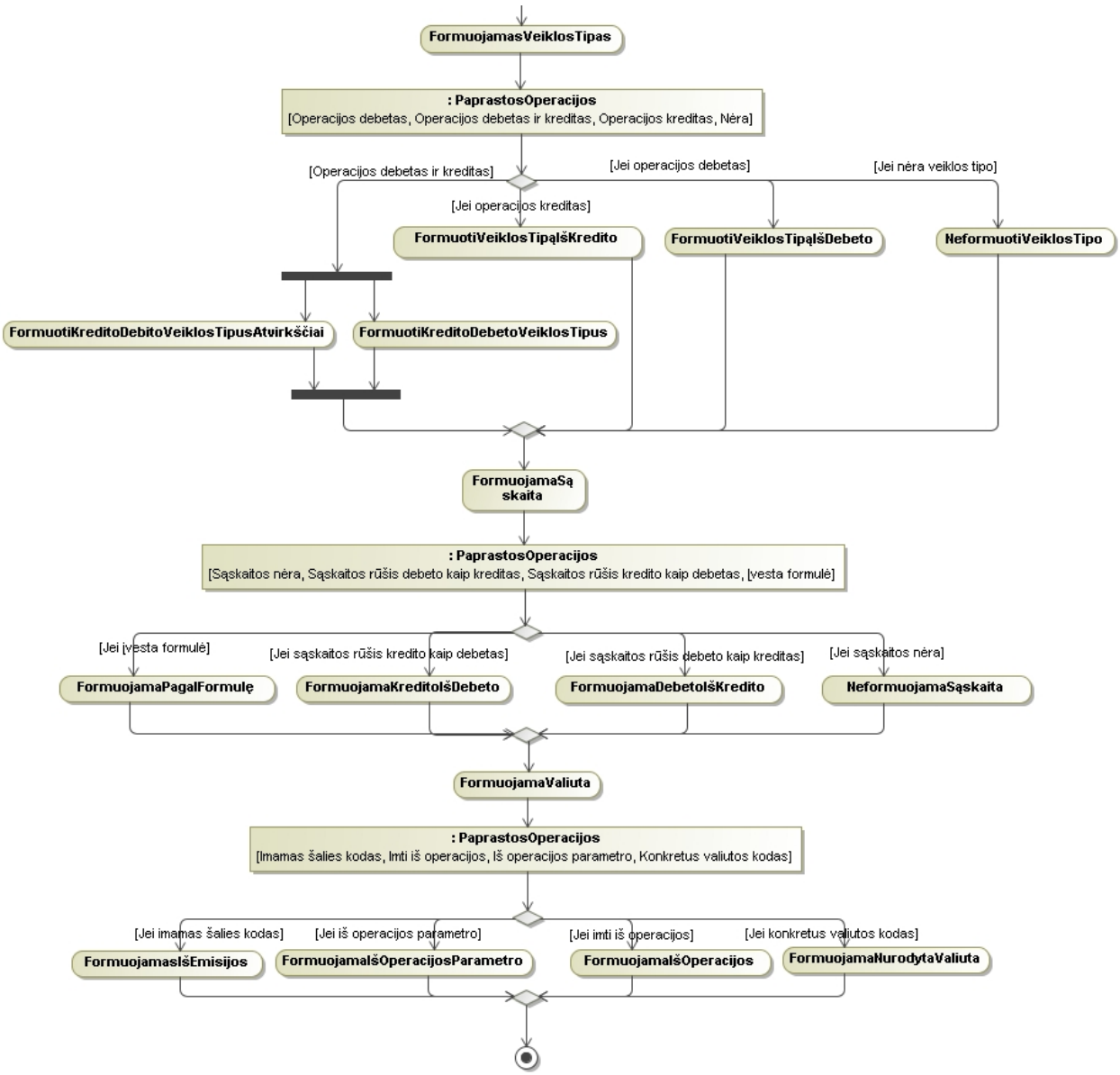
Paprastomis operacijomis nusakomas operacijos debeto ir kredito pusės, į kokia sąskaitą kokie pinigai ar VP paklius. Kadangi paprastų operacijų skaičius neribojamas galima apsibrėžti fiktyvias paprastas operacijas, kurių pagalba galima palengvinti apskaitą. Sukūrus fiktyvią operaciją į tam tikrą operacijoje nurodytą sąskaitą galima nurodyti pervesti komisą, sandorio sumą ir kitus parametrus iš sandorio formos. Kad užtikrinti lanksčią apskaitą paprastos operacijos gali dirbti su formulėmis, duomenys tokiu atveju imami iš sandorio.

Projektuojamas sistemos reikalavimas yra suderinamumas su kitomis informacinėmis sistemomis dėl to yra galimybė operacijas importuoti. Importo parametrai yra lengvai keičiami aprašant reikiamus parametrus sudėtinėse ir paprastose operacijose. Trūkstami duomenys užpildomi iš operacijų konfigūracijos.

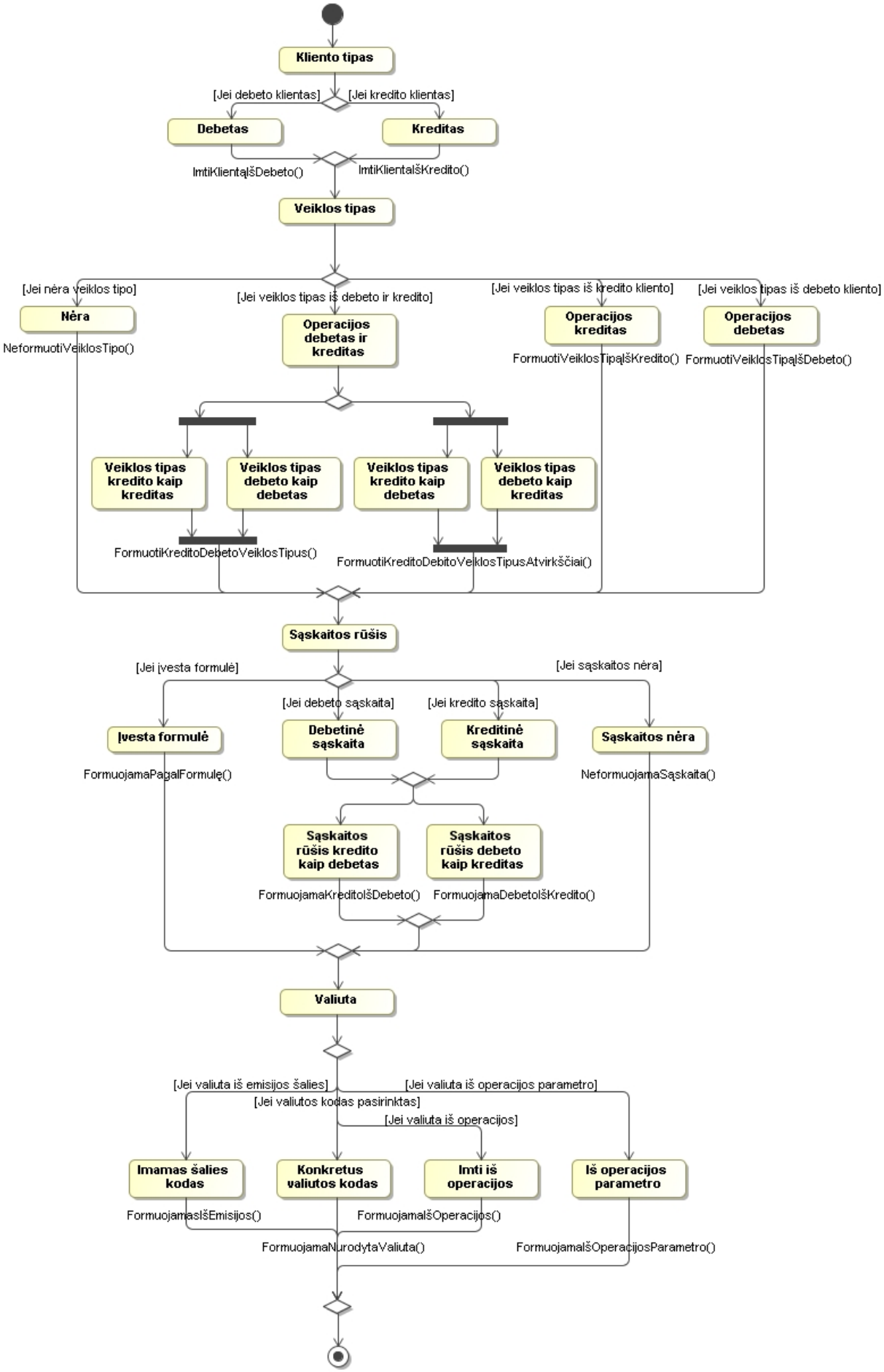
Operacijos konfigūracijoje galima nustatyti kurią sandorio pusę naudoti, tai yra debetą ar kreditą. Sandoriam su dviem pusėmis galima sukeisti debetą su kreditu.



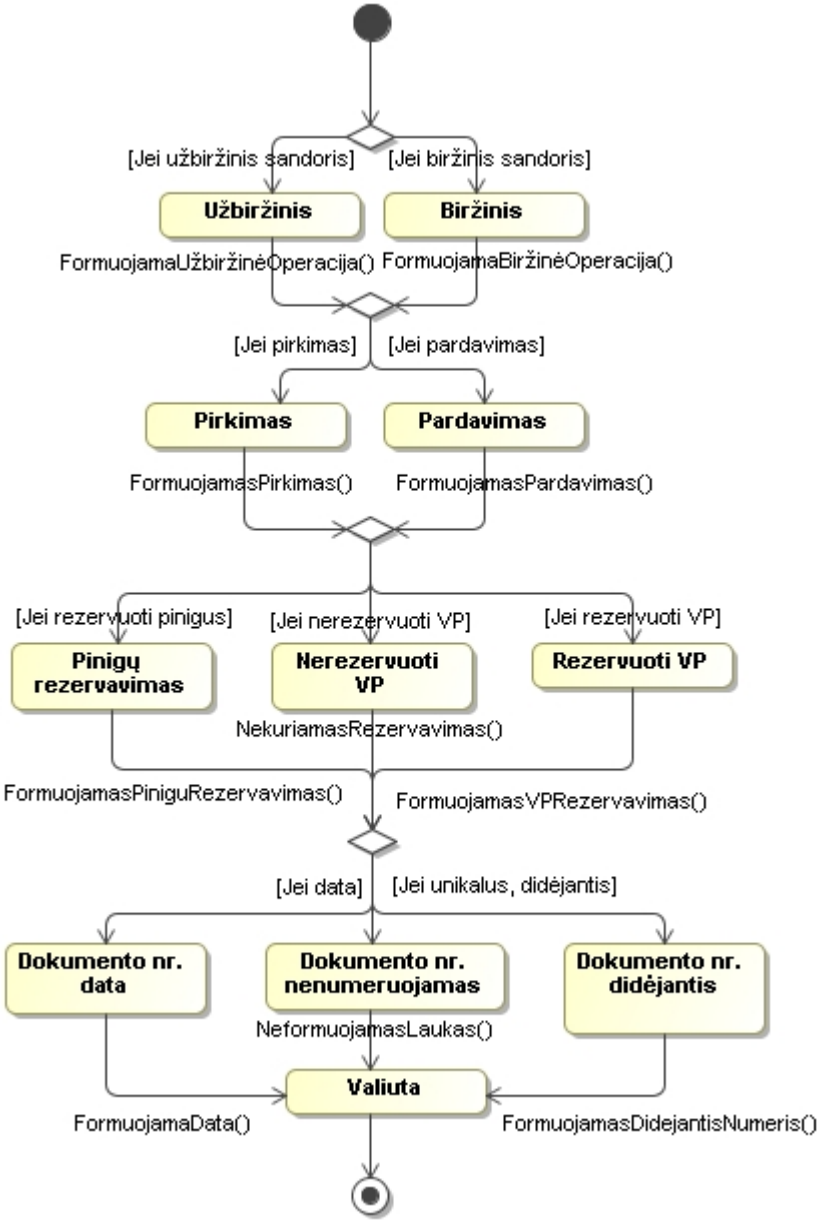
3.55 pav. operacijų veiklos fragmentas



3.56 pav. Operacijos veiklos fragmentas



3.57 pav. Paprastos operacijos būsenos



3. 58 pav. Sudėtinės operacijos būsenos

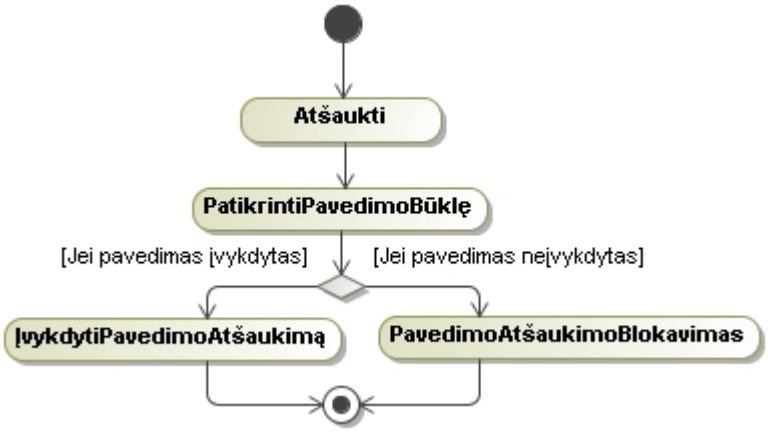
3.2.6. Vertybinių popierių prekyba

Vertybinių popierių prekybai realizuoti naudojamas trys veiklos: VP pirkimas, pardavimas ir atšaukimas. Projektuojamoje sistemoje realizuota sąsaja su birža, dėl to galima atlikti aukščiau paminėtus veiksmus. Pirkimo veiksmu sistemos vartotojas gali nusipirkti VP. Pardavimo veiksmu parduodami VP.

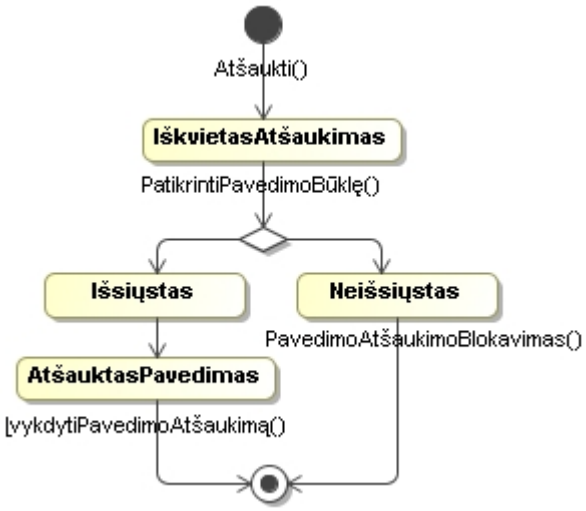
Vertybinių popierių prekyba galima tik biržos darbo valandomis, jei suformuojamas sandoris ne biržos darbo laiku sandoris bus išsaugotas, bet apskaitytas bus tik per sekančią prekybos sesiją. Nusiųstas sandoris į biržą bus apskaitytas jei tenkins sąlygą, kad siuntimas į biržą buvo sėkmingas ir pirkimo ar pardavimo sandoris buvo be klaidų.

Pardavimas skiriasi nuo pirkimo dėl to kad reikia papildomai atsižvelgti į kliento portfelio įgaliojimus. Įgaliojimai yra taikomi kelioms vartotojų grupėms, kai portfelis yra bendroji jungtinė nuosavybė, kuriai reikia įgaliojimo. Kitas atvejis kai kliento VP portfelis yra areštuotas. Tokiu atveju jei klientas nepateikia reikiamų dokumento VP parduoti negalės.

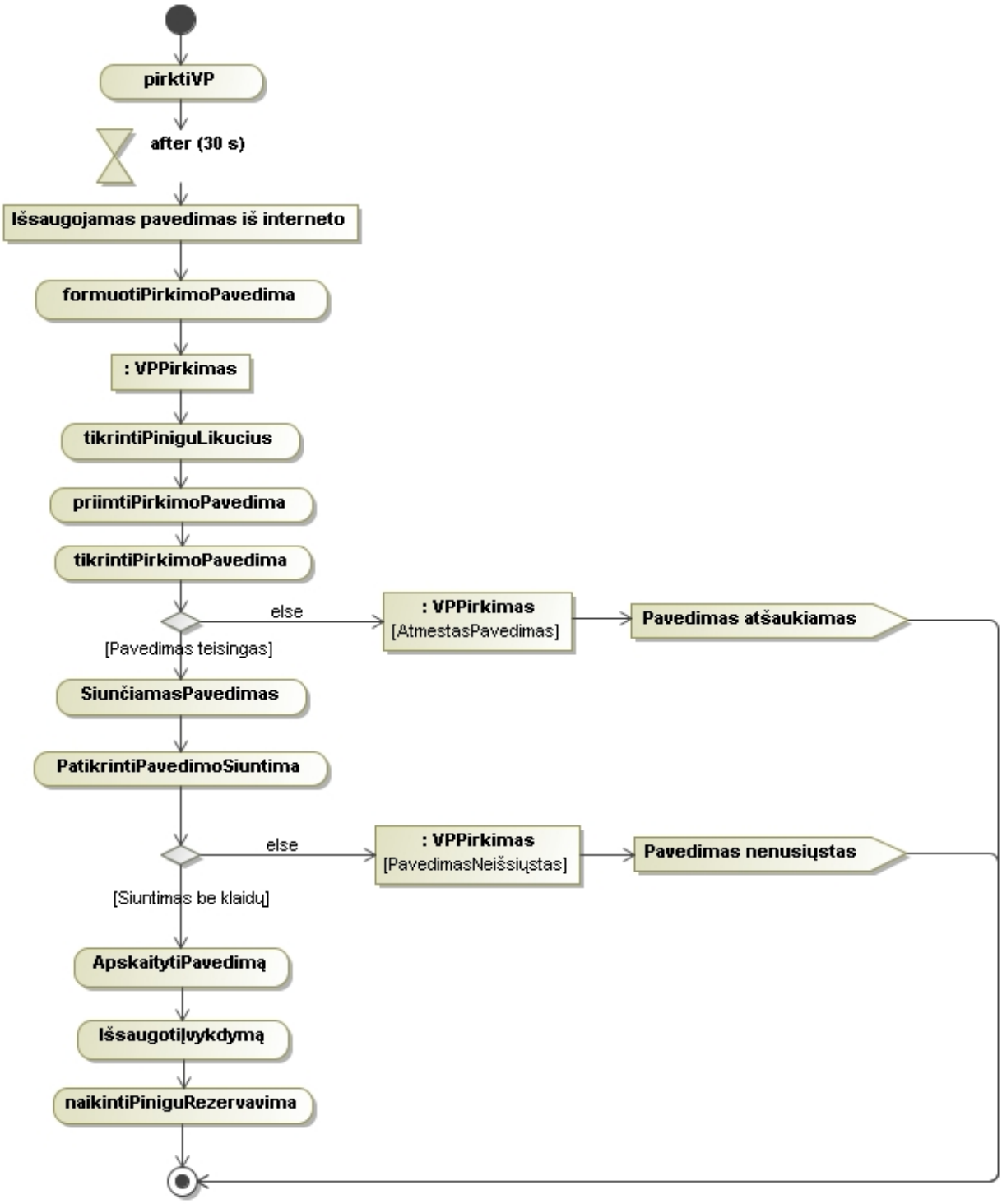
VP atšaukimai reikalingi jei klientas nori atšaukti pavedimą. Atšaukti galima tik priimtus pavedimus.



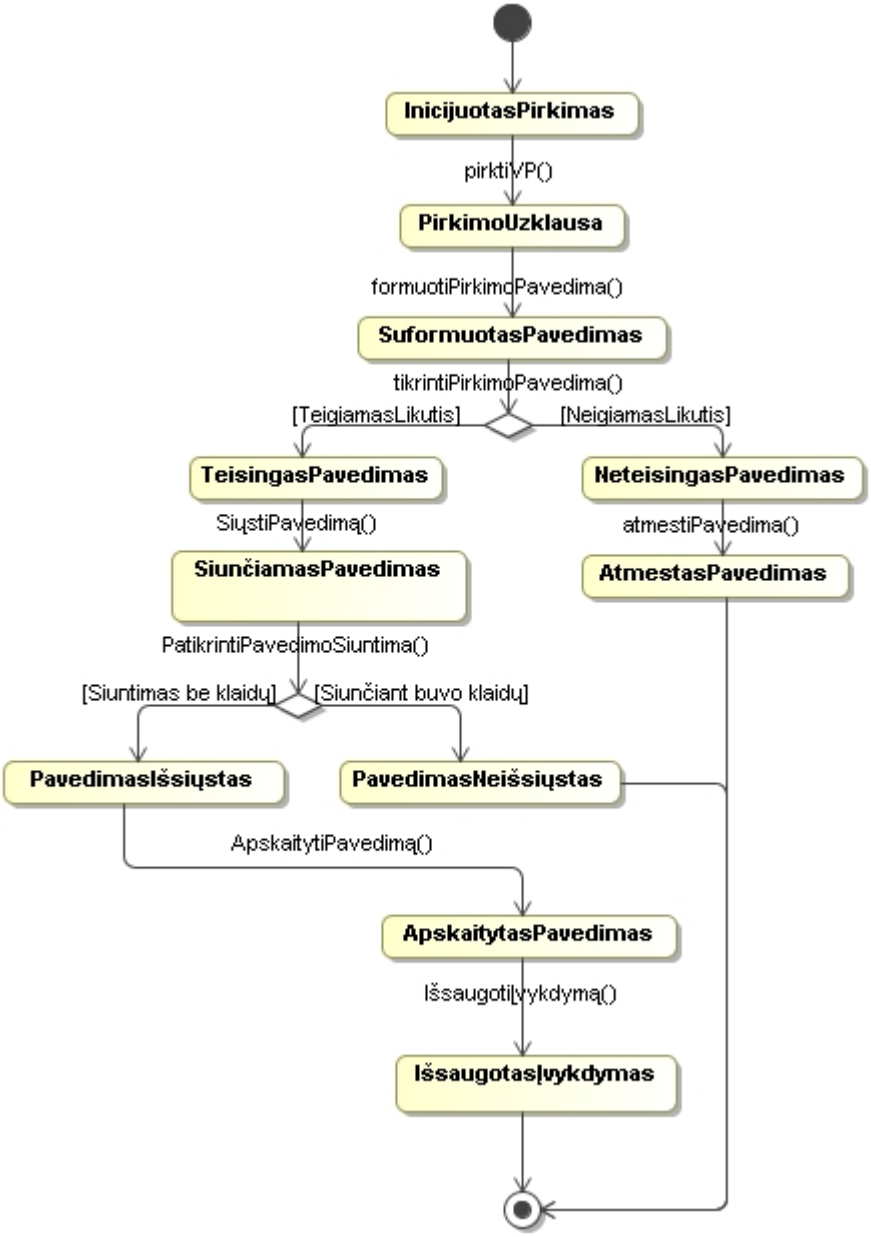
3.49 pav. VP atšaukimo veikla



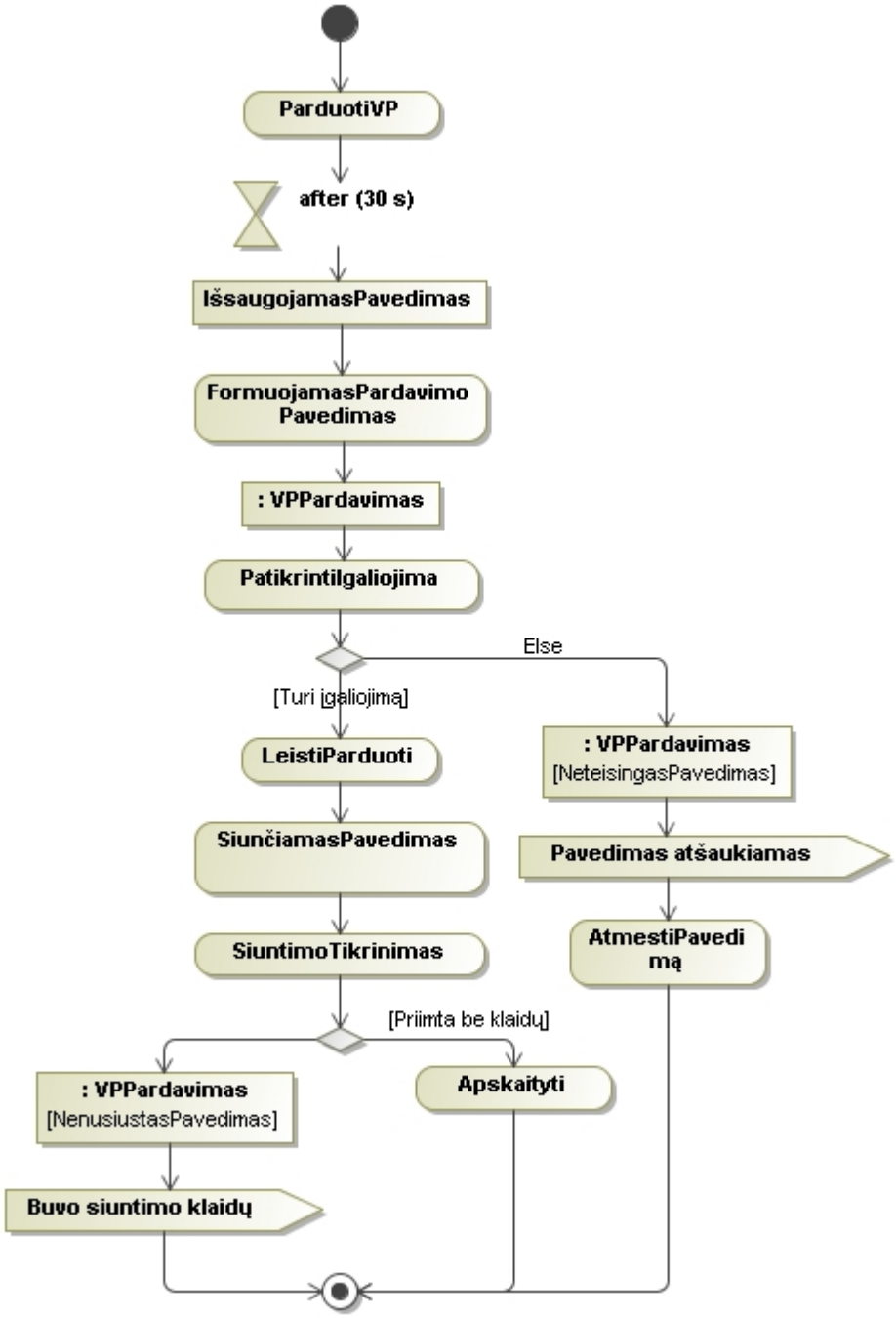
3.59 pav. VP atšaukimo būsena



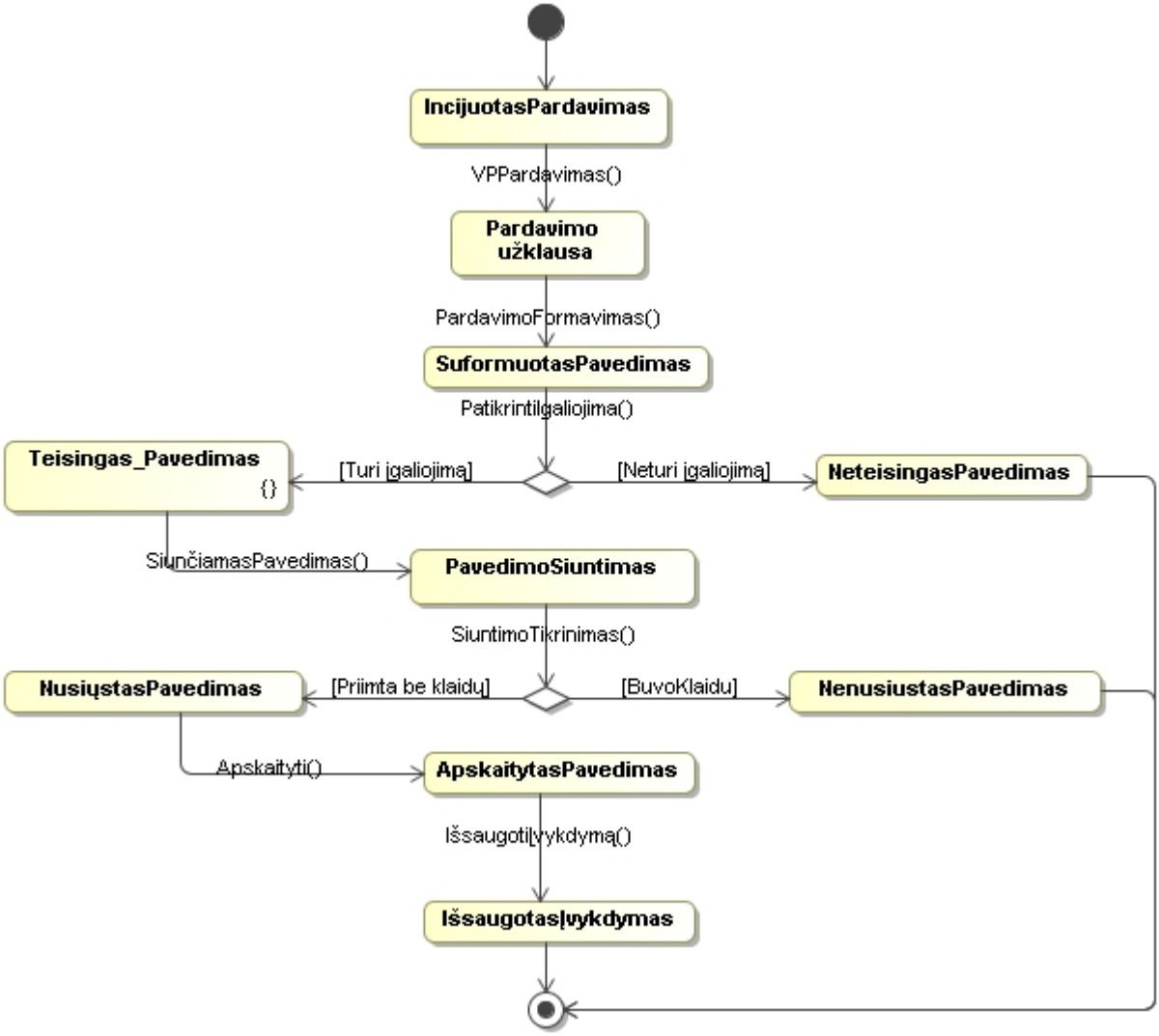
3.60 pav. VP pirkimo veikla



3.61 pav. Pirkimo būsenos



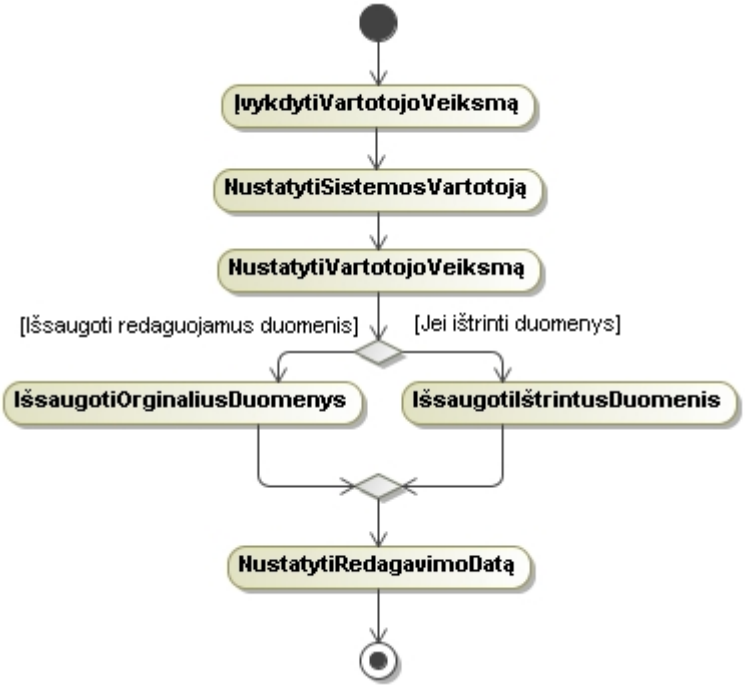
3.62 pav. VP pardavimo veikla



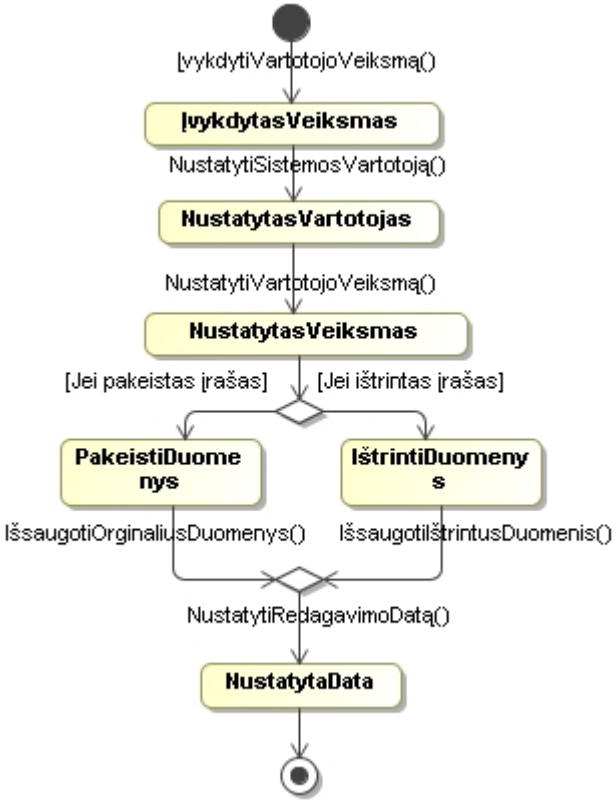
3.63 pav. VP pardavimo būsenos

3.2.7. Logai

Logai naudojami registruoti vartotojo veiksmus, vartotojui atliekant duomenų redagavimo, trynimo veiksmus sukuriamas įrašas su originaliais duomenimis. Taip užtikrinama konkretaus vartotojo atsakomybė už savo veiksmus.

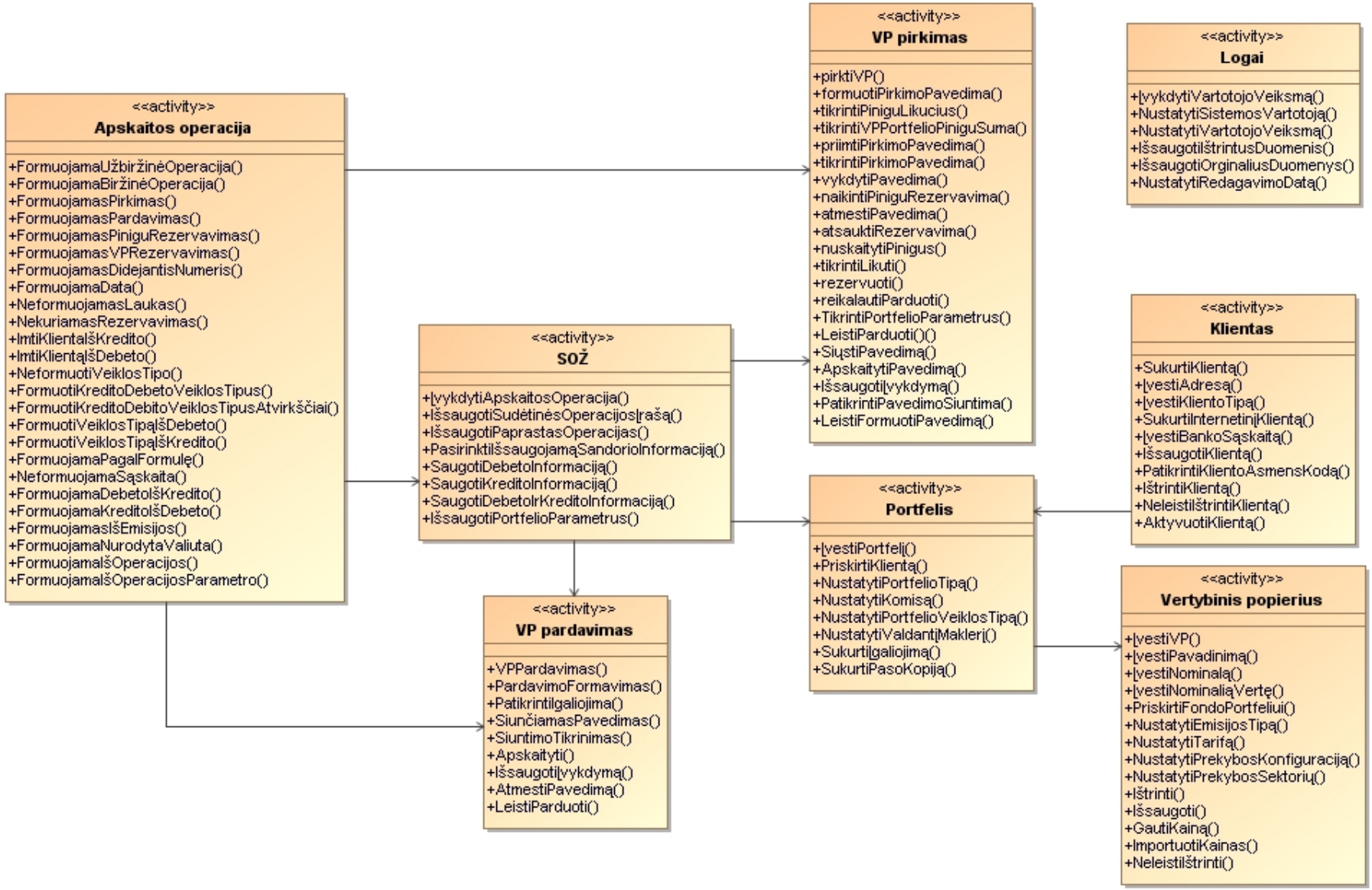


3.64 pav. Logų veiklos modelis



3.65 pav. Logų būsenos

3.2.9. Klasių diagrama

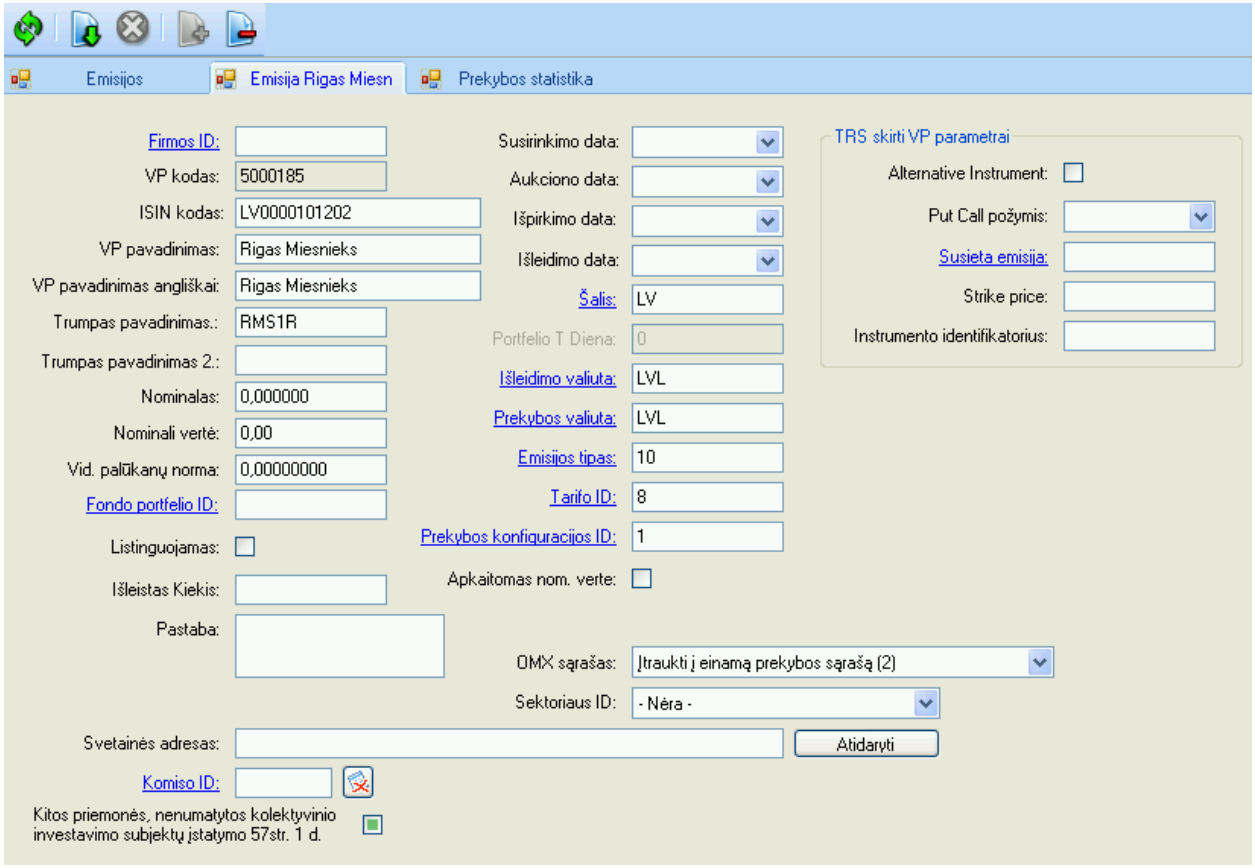


3.66 pav. Klasių diagrama

4. Vertybinių popierių apskaitos informacinės sistemos realizacija ir metodikos įvertinimas

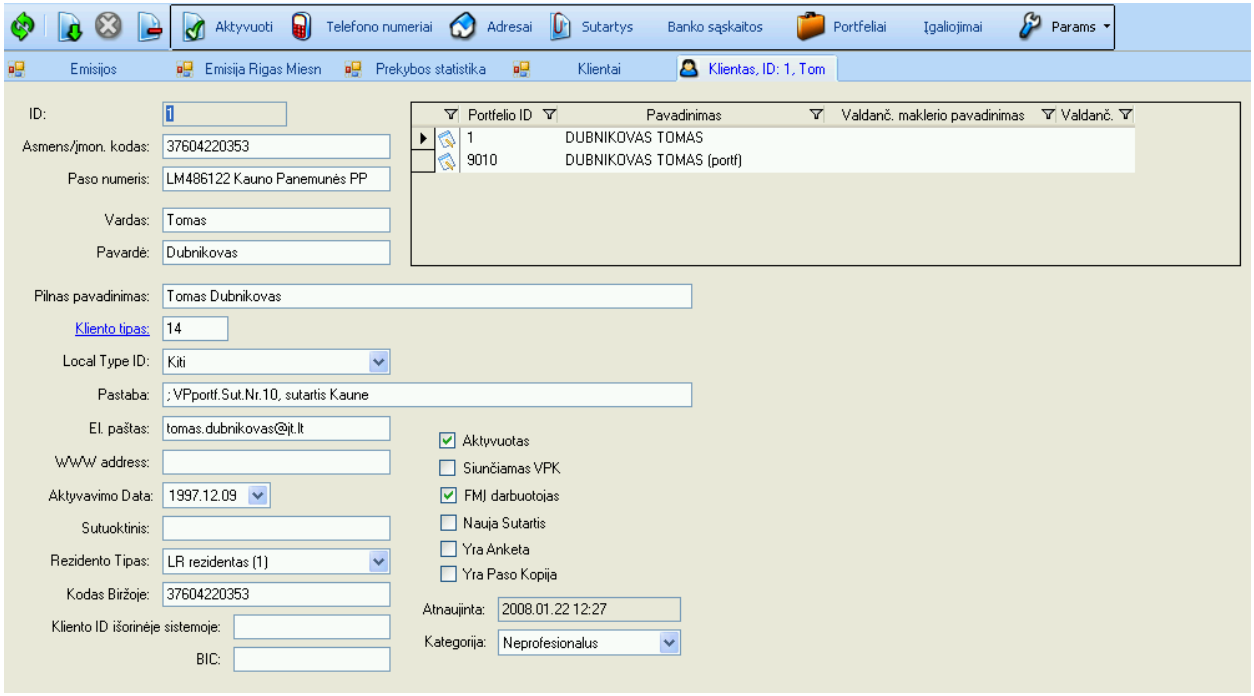
4.1. Informacinės sistemos realizacija

Žemiau pateikiami realizuotos informacinės sistemos langai. 4.67 pav. pateikiamas kuriamos IS vertybinio popieriaus (emisijos) langas, kuriame nustatomi vertybinio popieriaus vertės. Mėlynu tekstu pažymėti tekstiniai pavadinimai nustatomi iš dialogų, duomenys išrenkami iš duombazės. Dalis per dialogus išrenkamų reikšmių nustatomas iš statinių lentelių.



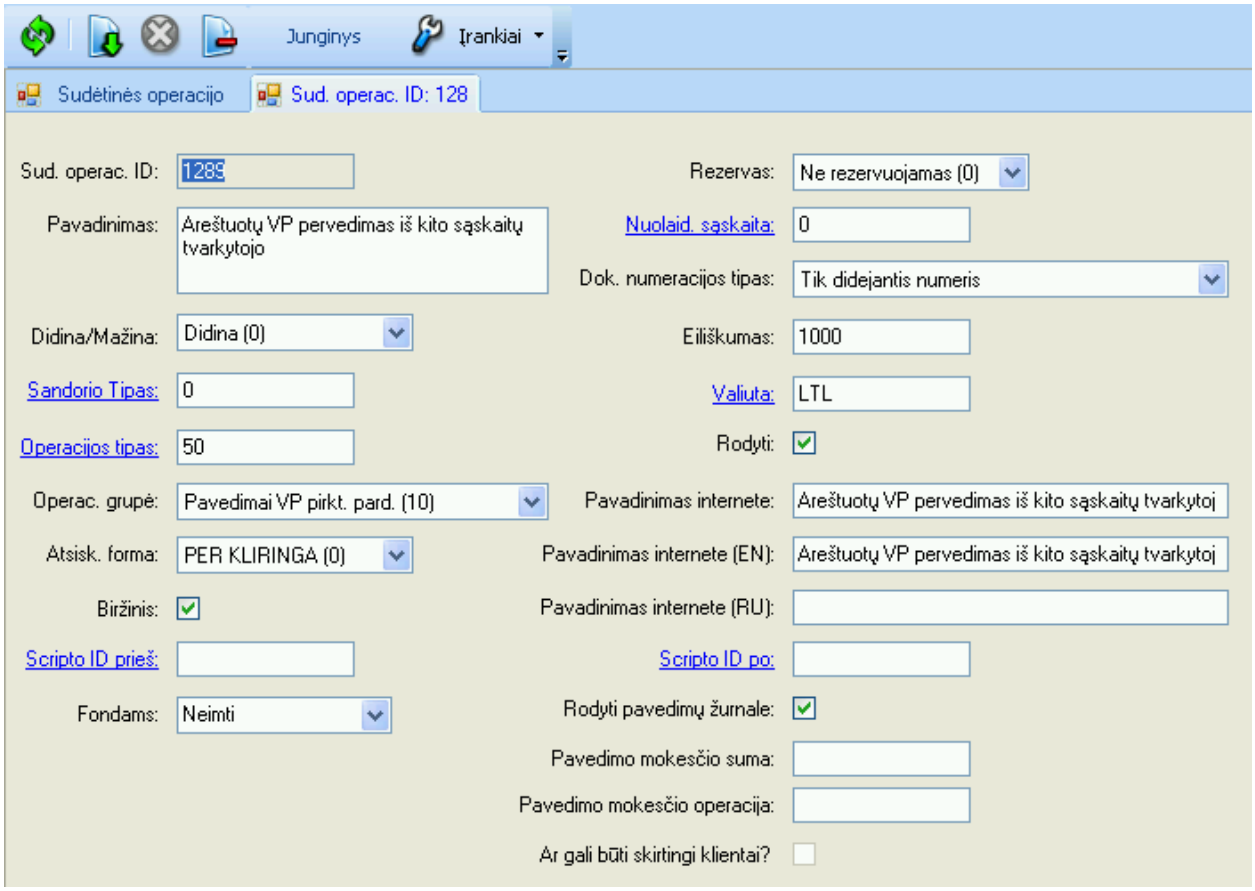
4.67 pav. Vertybinio popieriaus (Emisijos) langas

Kliento redagavimo lange darbuotojas gali pakeisti kliento duomenis. Šalia kliento duomenų pateikiami kliento portfeliai. Kliento kontaktinių duomenys yra išdalinti į atskirus langus, šiuos duomenis galima redaguoti per pagrindiniame kliento lange viršuje esančius meniu mygtukus. Portfelio įgaliojimus galima priskirti per viršutinio meniu punktą Įgaliojimai.



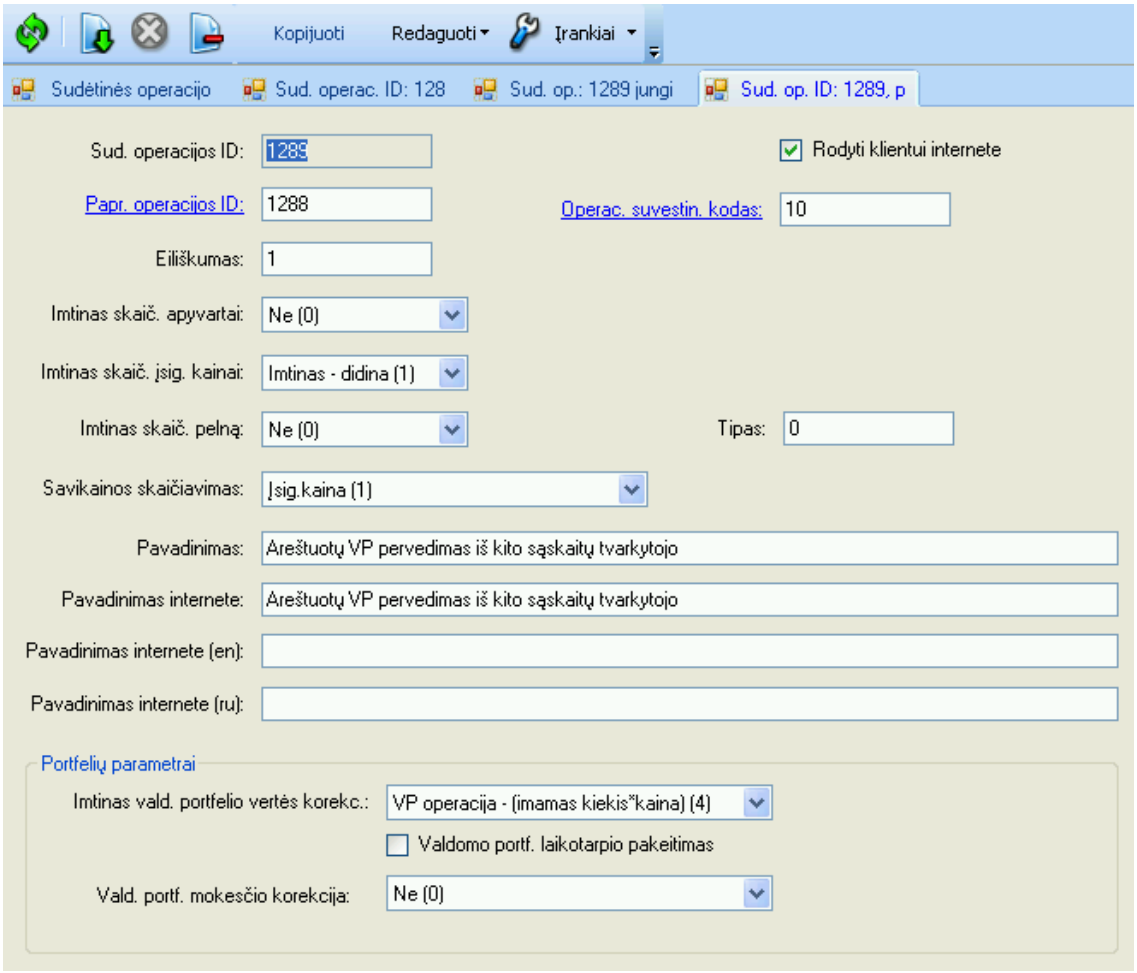
4.68 pav. Informacinės sistemos kliento redagavimo langas

Sudėtinės operacijos lange kaip buvo minėta nustatomi pagrindiniai pavedimo parametrai. Vartotojų patogumui iš sudėtinės operacijos redagavimo lango galima atsidaryti operacijos junginį ir pamatyti susietas su redaguojama operacija paprastas operacijas.



4.69 pav. Sudėtinės operacijos redagavimo langas

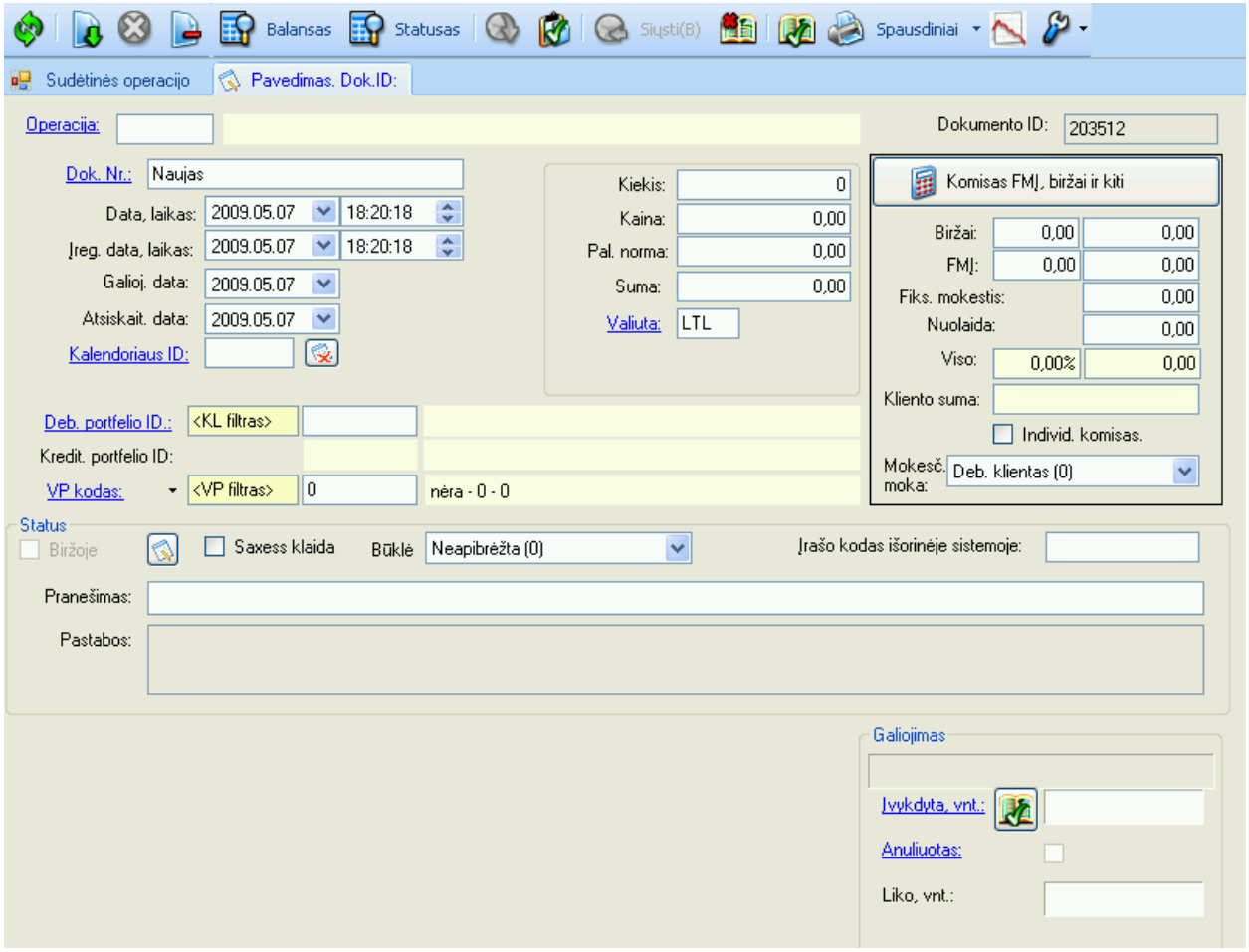
Paprastos operacijos redagavimo lange galima pakeisti paprastos operacijos parametrus ir kopijuoti paprastą operaciją. Tokia funkcija labai patogi kai reikia sukurti operaciją, kuri beveik nesiskiria nuo redaguojamos operacijos. Be šios funkcijos paprastai operacijai skirtas įrankių meniu su įvairiomis pagalbinėmis operacijų kopijavimo funkcijomis.



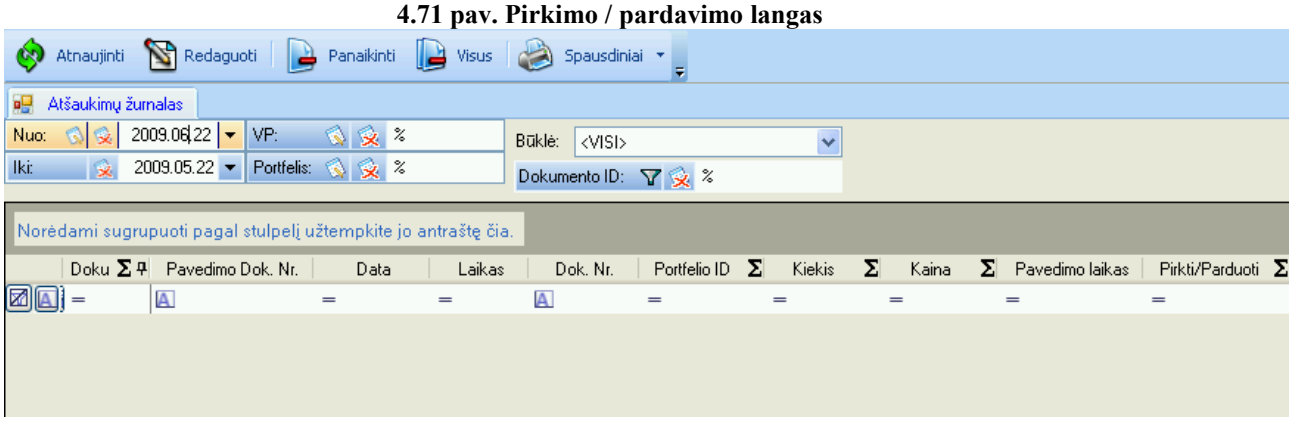
4.70 pav. Paprastos operacijos redagavimo langas

Pirkimui ir pardavimui realizuoti sistemoje užtenka vieno lango, visi reikiami parametrai formuojami iš pavedimo konfigūracijos. Komiso skaičiavimo lankstumui užtikrinti yra integruota pagalbinė komiso skaičiavimo priemonė. Pavedimo būklę galima matyti galiojimo langelyje, o būseną būklės meniu. Būklės išskleidžiamo meniu pagalba vartotojas gali sužinoti visas VP pavedimo būsenas. Atšaukimas realizuojamas iš pavedimo pateikimo lango. Atšaukimus galima peržiūrėti atskirame atšaukimo lange, kuriame yra filtras skirtas atrinkti atšaukimus pagal norimus parametrus.

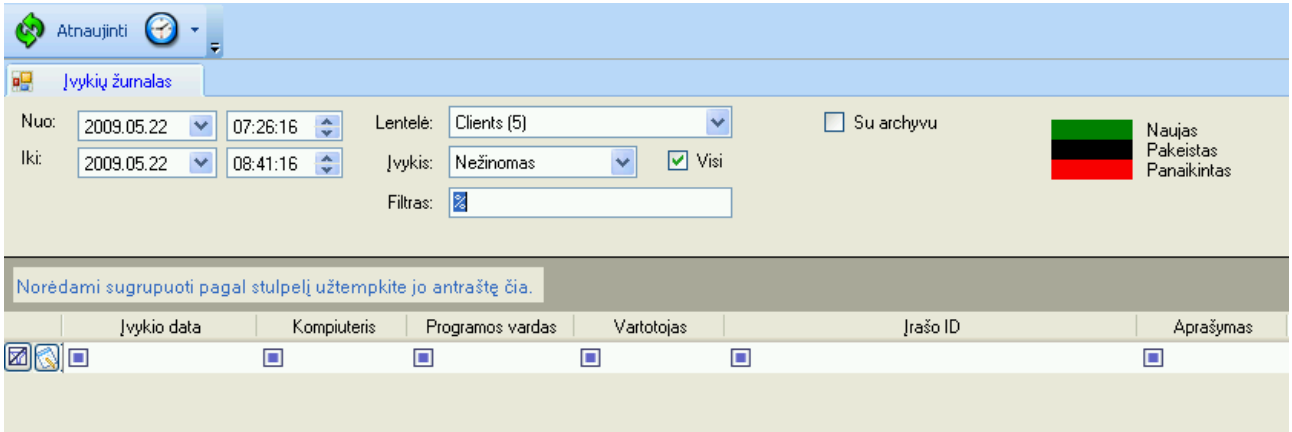
Įvykių (logų) lange peržiūrimi informacinės sistemos registruojami vartotojų veiksmai, sistemos klaidos ir duomenų pakeitimas. Įvykiai registruojami kiekvieno vartotojo, kompiuterio t.t. IS lange realizuoti filtrai lengvesnei įvykių peržiūrai.



4.71 pav. Pirkimo / pardavimo langas



4.72 pav. Pavedimų atšaukimų langas



4.73 pav. Ivykių (logų) peržiūros langas

4.2. Metodikos įvertinimas

Sukurtai metodikai įvertinti buvo atlikta trijų programuotojų (P1, P2, P3) apklausa. P1 programuotoju pažymėtas J. Zajančkauskas, P2 pažymėtas F. Tubelevičius ir P3 žymėjimu pažymėtas programuotojas Ž. Gedutis. Visi apklausti programuotojai dirbo prie šios informacinės sistemos kūrimo. Kiekvienam programuotojui buvo užduoti identiški klausimai:

1. Ar metodika suprantama?
2. Ar modeliai padėjo greičiau perprasti sistemos kūrimo užduotį?
3. Ar modeliai padėjo geriau perprasti sistemos kūrimo užduotį?
4. Ar taikant metodiką pagreitėjo kūrimas?
5. Ar metodika turėjo įtakos vartotojo sąsajos kūrimui?
6. Ar metodika turėjo įtakos duomenų bazės kūrimui?
7. Ar metodika turėjo įtakos programos komponentų kūrimui?
8. Ar pagerėjo projekto kokybė naudojant pateiktą metodiką ?
9. Ar modeliai padėtų tobulinti sistemą?
10. Kiek laiko reikėjo perprasti metodiką?
11. Kiek laiko reikėjo perprasti kuriamos sistemos modelius?
12. Kiek santykinai Jūsų nuomone pagreitėjo kūrimas?
13. Metodikos pranašumai
14. Metodikos trūkumai

Atsakymai į pirmuosius 9 klausimus buvo pateikiami balais 1-5 balų skalėje. Atsakant į 10-12 klausimus buvo paprašyta pateikti apytikslius įvertinimus. Atsakant į 13-14 klausimus buvo prašoma išdėstyti savo nuomonę. Atsakymai į 1-12 klausimus pateikiami lentelėse, 13-14 klausimų atsakymai buvo išanalizuoti ir iš jų gautos išvados.

Surinkti atsakymai

Lentelė 4.6. Klausimai įvertinti balais

Klausimai		P1	P2	P3	Vidurkis
1	Ar metodika suprantama?	5	3	4	4
2	Ar modeliai padėjo greičiau perprasti sistemos kūrimo užduotį?	4	3	5	4
3	Ar modeliai padėjo geriau perprasti sistemos kūrimo užduotį?	5	4	5	4,6
4	Ar taikant metodiką pagreitėjo kūrimas?	4	3	4	3,6
5	Ar metodika turėjo įtakos vartotojo sąsajos kūrimui?	1	2	1	1,3
6	Ar metodika turėjo įtakos duomenų	4	5	4	4,3

	bazės kūrimui?				
7	Ar metodika turėjo įtakos programos komponentų kūrimui?	4	4	5	4,3
8	Ar pagerėjo projekto kokybė naudojant pateiktą metodiką ?	5	4	4	4,3
9	Ar modeliai padėtų tobulinti sistemą?	4	5	5	4,6

Lentelė 4.7. Klausimai (10 – 12)

Klausimai		P1	P2	P3	Vidurkis
10	Kiek laiko reikėjo perprasti metodiką?	1 h	2 h	3 h	2 h
11	Kiek laiko reikėjo perprasti kuriamos sistemos modelius?	3 h	2 h	3 h	2,6 h
12	Kiek santykinai Jūsų nuomone pagreitėjo kūrimas?	10%	5%	5%	6,6%

Pastabos dėl metodikos ir modelių suvokimo

Programuotojai dirbantys prie šios informacinės sistemos tobulinimo metodikai perprasti užtruko sąlyginai nedaug laiko, dėl to kad supranta ir dirba prie kuriamos IS, suvokia ir naudoja UML. Modelyje vaizduojama kuriama informacinės sistemos modelis, dėl to programuotojams lengviau suvokti metodikos veikimą, per kuriamos sistemos pavyzdį.

Kuriamiems modeliams analizuoti užtrukta daugiau laiko, nes modelius reikia išnagrinėti. Atsiranda nesupratimų, ar iškyla klausimų dėl veiklos proceso, alternatyvių kelių ir modelio grafinio žymėjimo. Kadangi tie patys veiklos proceso modelio elgsenų gali būti žymimi skirtingai modelyje atsiranda dviprasmiškų veiksmų elgsenų.

Metodikos pranašumai klausimo išvados

Anksčiau naudojama sistemos projektavimas buvo remtasi tik rašytinėmis specifikacijomis, ar specifikacijų nebuvo iš viso. Dėl to sistemos dalis funkcionalumo nebuvo dokumentuota. Taikant sukurtą metodiką galima papildyti ir sutvarkyti IS sistemos modelį, aprašyti trūkstamą funkcionalumą, kuris padės toliau plėtoti IS. Taikant metodikos paminėtus žingsnius labai svarbi dalis yra tai, kad sudėtinga sistemos proceso veiklas vaizduojant grafiškai lengviau įvertinti, kuriamos sistemos trūkumus, pridėti alternatyvius scenarijus proceso veikloms. Atliekant sistemos pakeitimus lengviau pastebėti, kaip sistemai atsilieps funkcionalumo pakeitimai.

Modelis sudarytas pagal pateiktą metodiką, padeda nežinantiems programuotojams susipažinti su sistema, taip išvengiama didelė dalis klaidų. Naujas programuotojas nežinodamas viso funkcionalumo, tobulindamas naują funkciją gali pažeisti visą esamą proceso eigą.

Metodikos trūkumai klausimo išvados

Kaip ir bet kokios metodikos trūkumas yra, tai, kad jei nebus IS modelio kiekvienas pakeitimas ar papildymas veiklos proceso nebus dokumentuojamas, turimas modelis nebe neatspindės realybės. Dėl to dirbant prie dinaminių sistemų, kuriose reikalingi dažni funkcionalumo pakeitimai svarbu laiku papildyti IS modelį.

Pagal metodiką sukurtas modelis padeda geriau matyti veiklos procesą, nustatyti problemines vietas. Aišku, tai, kad IS kokybei užtikrinti reikia turėti pakankamai žinių ir gerą analitinį mąstymą. Būtina probleminės srities suvokimas ir patirtis IS sistemų kūrimą, kitaip sudėtinga numatyti galimus alternatyvius proceso scenarijus, tobulinimo atvejus ir t. t.

4.2.1. Metodikos įvertinimo išvados

Apklausus programuotojus, kurie dirba prie vertybinių popierių apskaitos sistemos nustatyta, kad pagal darbe pateiktą metodiką sukurtas modelis padės suprasti sistemos veikimą. Lengviau pastebėti pakeisto funkcionalumo įtaka veiklos proceso modeliui ir pačiai sistemai. Perprasti elgsenos modelių integravimu pagrįsta metodiką nėra sunku, kai modeliuose pateikiama kuriama informacinė sistema, kuri dabar yra tobulinama. Daugiau laiko užtruks pačių modelių analizė. Tai iš tiesų yra normalu, kadangi grafinėse žymėjimuose projektuojant gali likti dviprasmybių, kurias nagrinėjant modelius reikia ištirti ir pataisyti, jei tai yra būtina proceso veiklų aiškumui užtikrinti. Metodikos pagalba bus galima paspartinti IS kūrimą beveik 7%. Paspartinti IS kūrimą bus galima kai programuotojai bus geriau susipažinę su metodika ir turės aiškias nustatytas proceso modeliavimo gaires, kurių turės griežtai laikytis norint užtikrinti IS modelių aiškumą ir teisingumą. Metodika beveik neturėjo įtakos grafinei sąsajai, tai iš tikrųjų nėra blogai, kadangi metodika nėra skirta apibrėžti kuriamą vartotojo sąsają.

5. Išvados

1. Kuriant sudėtingas informacines sistemas, kuriuose realizuoti sudėtingi verslo procesai, projektavimui tikslinga panaudoti elgsenos modelius.
2. UML ir populiarių veiklos procesų modeliavimo kalbų (BPMN ir BPEL) analizė parodė, kad UML veiklos diagramos atitinka standartinius reikalavimus veiklos procesams modeliuoti, be to, jas galima susieti su būsenų ir klasių modeliais.
3. UML kalba geriau tinka projektuoti IS, kadangi ji leidžia aprašyti ne tik veiklos procesus, bet ir klasių bei architektūrinius modelius, duomenų bazių schemas – viską, ko reikia sukurti IS.
4. UML elgsenos yra klasės, todėl iš veiklos diagramų modelių galima gauti valdymo klases, kurias galima realizuoti programinio kodo generavimo įrankiais.
5. Šio tyrimo rezultate sudaryta metodika paremta UML elgsenos modeliais ir skirta IS, pasižyminčioms sudėtingais veiklos procesais, projektuoti.
6. Kuriant veiklos proceso modelius ir jų esybių būsenas pagal pateiktą metodiką galima susieti esybes, jų būsenas, veiklos procesus ir valdymo klases.
7. Sukurta metodika buvo pritaikyta projektuojant vertybinių popierių apskaitos informacinę sistemą. Projektuojant tokiu būdu, kokybė išaugo dėl to, kad aiškiai matomas veiklos procesas, lengviau pastebėti alternatyvius procesus.
8. Atliktas metodikos įvertinimas, remiantis įvertinimo rezultatais sukurta metodika padeda suvokti veiklos procesus, palengvina IS sistemos tobulinimą ir paspartina programinės įrangos kūrimo procesą.

6. Literatūros sąrašas

1. Conrad Bock „UML 2 Activity and Action Models“ Journal of Object Technology, Vol. 2 No. 4, Birželis-Spalis 2003.
2. „Unified Modeling Language: Superstructure version 2.0“ OMG.
3. „OMG Unified Modeling Language“ Object Management Group version 1.5.
4. Bock, Conrad „Three Kinds of Behavior Model“ Journal of Object-Oriented Programming, Birželis/Spalis, 1999.
5. Conrad Bock „UML Without Pictures“ IEEE Computer Special Issue on Model-driven Development, Rugsėjis/Spalis, 2003.
6. „Workflow Management Facility Specification” Object Management Group Version 1.2.
7. „Workflow Standard - Interoperability Abstract Specification“, Workflow Management Coalition, http://www.wfmc.org/standards/docs/TC-1012_Nov_99.pdf žiūrėta [2009.05.18].
8. Conrad Bock „UML 2 Activity and Action Models Part 3: Control Nodes“ Journal of Object Technology, Vol. 2, No. 6, Lapkritis-Gruodis 2003.
9. Conrad Bock „UML 2 Activity and Action Models, Part 2: Actions“ Journal of Object Technology, vol. 2, no. 5, Rugsėjis-Spalis 2003, p. 41-56.
10. http://www.jot.fm/issues/issue_2003_09/column4 žiūrėta [2009.05.18].
11. Conrad Bock „UML 2 Activity and Action Models“ Journal of Object Technology, vol. 2, no. 4, Gegužė-Spalis 2003, p. 43-53.
12. http://www.jot.fm/issues/issue_2003_07/column3 žiūrėta [2009.05.18].
13. „MOF 2.0 Query/Views/Transformations RFP“ Object Management Group.
14. Shooter, S.B., Keirouz, W.T., Szykman, S., Fenves, S. J., „A Model for the Flow of Design Information in Product Development“ Journal of Engineering with Computers, vol. 16, 2000, p. 178-194.
15. Booch, G., Rumbaugh, J., and Jacobson, I., The Unified Modeling Language User Guide, Addison-Wesley, 1999.
16. Conrad Bock „Unified Behavior Models“ Journal of Object-Oriented Programming, vol. 12, no. 5, Rugsėjis 1999.
17. „OMG Unified Modeling Language, version 1.5“ Object Management Group
18. „Business Process Modeling Notation (BPMN) Information: Frequently Asked Questions“, http://www.bpmn.org/Documents/FAQ.htm žiūrėta [2009.05.18].

19. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros „Workflow Patterns. Distributed and Parallel Databases“, Journal of Object Technology, vol. 14, no. 1, p. 5 – 51, 2003.
20. S. Jablonski and C. Bussler „Workflow Management: Modeling Concepts, Architecture, and Implementation“, International Thomson Computer Press, London, UK, 1996.
21. N. Russell,W.M.P. van der Aalst, A.H.M. ter Hofstede, and D. Edmond „Workflow Resource Patterns: Identification, Representation and Tool Support“ O. Pastor and J. Falcaoe Cunha editors, Proc. of 17th Int. Conf. on Advanced Information Systems Engineering (CAiSE05), leidinys 3520 iš LNCS, p. 216–232. Springer, 2005.
22. 18. S. White. Business Process Modeling Notation (BPMN). www.bpmi.org žiūrėta [2009.05.18].
23. 12. C. Ouyang, M. Dumas, S. Breutel, and A.H.M. ter Hofstede „Translating Standard Process Models to BPEL“ To appear in Proceedings of 18th International Conference on Advanced Information Systems Engineering (CAiSE 2006), June 2006.
24. M. Kloppmann, D. Koenig, F. Leymann, G. Pfau, A. Rickayzen, C. von Riegen, P. Schmidt, and I. Trickovic. „WS-BPEL Extension for People – BPEL4People“, http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people žiūrėta [2009.05.18].
25. Petia Wohed1, Wil M.P. van der Aalst, Marlon Dumas Arthur H.M. ter Hofstede, Nick Russell Pattern-based „Analysis of BPMN - an extensive evaluation of the Control-flow, the Data and the Resource Perspectives“
26. Stephen A. White „Process Modeling Notations and Workflow Patterns“ IBM Corp.