KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Linas Gudonavičius

Veiklos procesų vykdymo modeliavimas, taikant UML veiklos grafus

Magistro darbas

Darbo vadovė
doc. dr. L. Nemuraitė

Kaunas, 2004

TURINYS

WORKFLOW MODELLING USING UML ACTIVITY DIAGRAMS

SUMMARY

The goal of this thesis is to define a semantics of activity diagrams that is suitable for workflow modelling. The semantics should allow verification of functional requirements using model checking. The semantics should be accurate, yet easy to analyse by a model checker.

The defined semantics must represent workflow behaviour accurately. We mean, that the semantics of an activity diagram as workflow specification is a realistic and faithful representation of the real-world behaviour of the corresponding workflow. From an inaccurate semantics, nor reliable analysis results can be inferred. Then analysis is useless and ineffective. Therefore, the semantics must represent all relevant aspects of workflow behaviour.

The defined semantics must be "easy to analyse" for a model checker. Ideally, to allow for efficient model checking, the state space of the semantics of an activity diagram must be as small as possible.

Since a workflow specification prescribes how a workflow system behaves, the semantics is defined and motivated in terms of workflow systems.

There is analysed the correctness for verification ability of UML activity diagrams by using hypergaphs. There suggested the transformation from UML activity diagram to hypergraph and verification algorithm of hypergraph, which can be used for expanding the capabilities of CASE tools for workflow and e-business modelling.

ĮVADAS

Šis darbas susideda iš keturių dalių:

- Analitinė dalis (veiklos procesų analizė);
- Pagrindinė dalis (veiklos procesų vykdymo modeliavimas, taikant UML veiklos grafus). Ši dalis apima veiklos diagramos privedimą prie hipergrafų bei funkcionalumo reikalavimų tikrinimą;
- Eksperimentinė dalis (verifikavimo algortimo pritaikymas, sudarant Wing Tsun kungfu klubo veiklos procesų sistemos valdymo modelį);
- Išvados (analitinės bei pagrindinės dalies).

Automatizuojant verslo procesus ar kuriant darbų srautų valdymo sistemas tiek įmonės viduje, tiek kelių organizacijų bendradarbiavimo metu, reikia sukurti proceso apibrėžimą, aprašantį jo vykdymo logiką. Šis aprašas turi būti korektiškas ir tikslus, kadangi jį naudoja arba jo pagrindu sukuriamos programos, valdančios proceso vykdymą. Verslo procesams apibrėžti naudojamos UML veiklos diagramos [1], [9] arba panašios notacijos, pvz., [7]. Jų privalumas toks, kad jos leidžia grafiškai aprašyti sudėtingus valdymo srautus. Ši forma intuityviai suprantama verslo analitikams ir informacinių technologijų specialistams. Tačiau UML CASE įrankiai neturi galimybių valdymo logikos teisingumui patikrinti.

Pradinėse informacinių sistemų projektavimo fazėse reikia sudaryti veiklos procesų (*business process)* modelį. Vieni tai vadina verslo (*business*) modeliavimu, kiti – organizacijos (*enterprise*) modeliavimu. Nors skirtinguose projektavimo metoduose veiklos procesų (verslo, organizacijos) modelis šiek tiek skiriasi, viskas susiveda į tai, kad reikia apibrėžti veiklos panaudojimo atvejus (*use cases*), juos detalizuoti veiklos procesų modeliais bei sudaryti atitinkamų veiklos objektų (*business objects*) modelį. Tokie uždaviniai iškyla tiek tiesioginiame informacijos sistemų projektavime, tiek atliekant informacijos sistemų reinžineriją ar apibrėžiant darbų sekų valdymo procesus. Objektinės orientacijos (*OO*) analizės ir projektavimo metodai nepakankamai gerai sprendžia šį uždavinį. UML veiklos diagramų semantika veiklos procesams aprašyti yra labiau pritaikyta programinės įrangos elgesiui (algoritmams) vaizduoti ir nelabai tinka veiklos procesams modeliuoti, todėl siūloma veiklos diagramas privesti prie hipergrafų (įvesti papildomus apribojimus, naudoti laukimo

būsenas ir kt.), kuriuos galima formaliai patikrinti. Tai labai aktualu kompiuterizuojant el.verslo procesus, kai programine įranga siekiama generuoti automatiškai pagal proceso aprašymą. Tačiau hipergrafai naudoja pakeistą UML notaciją (nenaudoja sudėtinių veiksmų ir kt.), kas nepriimtina, norint naudoti standartinius UML CASE įrankius.

Šio darbo tikslas – patikslinti veiklos diagramų semantiką, kuri būtų tinkama veiklos procesams modeliuoti ir leistų atlikti modelio patikrą bei sudaryti modelio tikrinimo algoritmą.

UML veiklos diagramų pradininkas yra J. Odell [5], [6]. Tiesa, dabartinės veiklos diagramos įgijo visai kitokį pavidalą, nei pradiniuose šaltiniuose. UML veiklos grafai susideda iš viršūnių ir lankų. Viršūnės interpretuojamos kaip veiklos (susidedančios iš vieno ar daugiau veiksmų), perėjimai reiškia valdymo srautus. UML yra ir kitokių tipų viršūnės, vadinamos pseudoviršūnėmis, skirtos srautų sujungimams vaizduoti. Galimos sudėtinės viršūnės, kurios aprašo atskirus veiklos procesus, tačiau tikrinimo metu arba veiklos diagrama išskleidžiama iki paprastų viršūnių, arba sudėtinė viršūnė traktuojama kaip paprasta, o joje aprašytas procesas tikrinamas atskirai.

Grafų korektiškumą galima nusakyti tokiais kriterijais:

- iš pradinės viršūnės turi būti pasiekiamos visos galinės viršūnes (negali būti aklaviečių);
- negali būti „mirusių“ viršūnių (visos jos turi turėti galimybę būti aktyviomis, negali būti kabančių viršūnių);
- negali būti „mirusių“ lankų (visi jie turi jungti viršūnes, t.y., turėti pradžią ir pabaigą);
- negali būti „amžinų ciklų“ (visuomet turi būti galimybė pereiti iš vienos viršūnės į kitą ir į galinę viršūnę).

Kai kurie reikalavimai užtikrinami CASE įrankiuose, pavyzdžiui, Rational Rose negalima pavaizduoti kabančio lanko. Automatizuotą tikrinimą apsunkina sprendimų viršūnės bei sinchronizavimo juostos. Todėl šiame darbe nagrinėjama galimybė transformuoti UML veiklos grafus į hipergrafus [2], [3], [4], turinčius sudėtinius perėjimus.

Pateikiamos pervedimo į hipergrafą taisyklės ir hipergrafų tikrinimo algoritmas, susidedantis iš struktūros tikrinimo (iš karto identifikuojama netinkama grafo sandara) ir

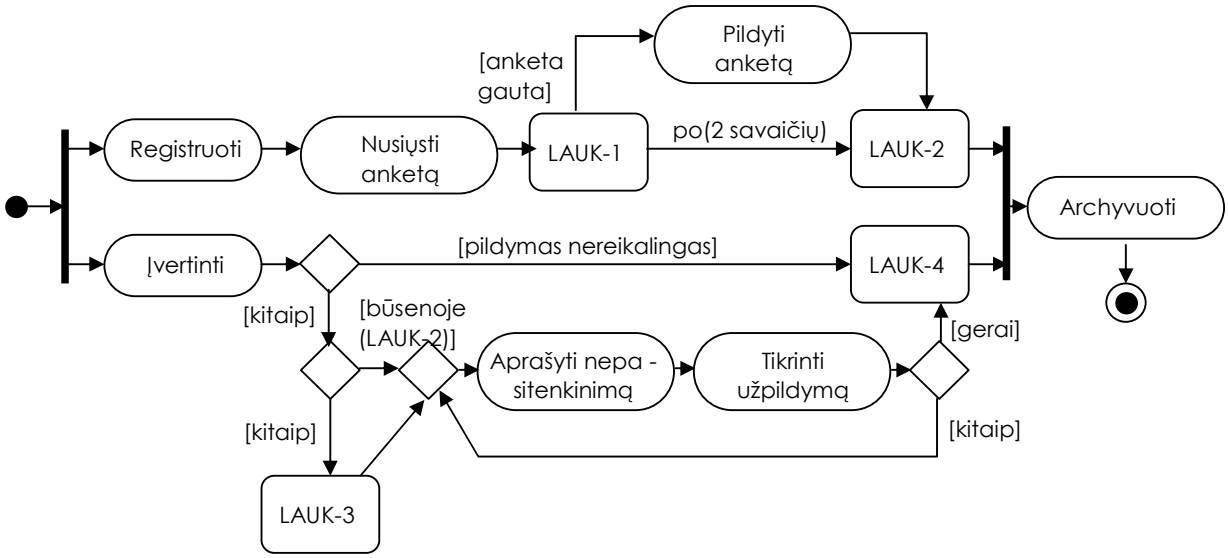
vykdymo tikrinimo algoritmo (galima jį vadinti dinaminiu tikrinimu). Šiuo tikrinimo algoritmu būtų galima išplėsti veiklos procesų modeliavimo CASE įrankių galimybes.

Šio darbo rezultatai buvo paskelbti devintoje tarpuniversitetinėje magistrantų ir doktorantų konferencijoje „Informacinė visuomenė ir universitetinės studijos“ ir parašytas straipsnis [15].

1 VEIKLOS PROCESŲ ANALIZĖ

1.1 VEIKLOS PROCESŲ MODELIAVIMAS

Darbų srautų modelis nurodo organizacijoje atliktą įvykių eiliškumą. Tipinės rikiavimo konstrukcijos yra sekos, pasirinkimo ir paralelizmo. Naudinga tokio pobūdžio išdėstymui notacija yra atvaizduojama naudojant UML veiklos grafus. Ovalai žymi aktyvumo būsenas (*1 pav.*), o stačiakampiai suapvalintais kampais – laukimo būsenas. Veiksmo būsenoje kai kurie veiksmai yra užimti. Veiklos procesai prasideda juodame taške (pradinė būsena) ir baigiasi „jaučio akimi“ (pabaigos būsena). Sija vaizduojama išsišakojimu (daugiau kaip vienas išėjimas) arba jungtimi (daugiau kaip vienas įėjimas). Deimantas vaizduoja pasirinkimą (daugiau kaip vienas išėjimas) arba skirtingų pasirinkimų sujungimą (daugiau kaip vienas įėjimas).



1 pav. UML veiklos grafas

Veiksmus atlieka aktorius. Veiksmo būsena yra veiksmas su įėjimo ir mažiausiai vienu išėjimo perėjimu, kur turi būti besąlyginis įvykis, įvykdantis įėjimo veiksmą (tokių perėjimų gali būti keletas, jei jie turi sąlygas). Veiksmai negali turėti vidinių perėjimų ir išorinių perėjimų, paremtų apibrėžtais įvykiais. Šiuo atveju naudojami paprasti veiksmai. Įprastinis veiksmų naudojimas yra algoritmo vykdymo ar veiklos proceso modeliavimas. Veiksmo metu aktorius atnaujina CASE atributus, kurie laikomi duomenų bazėje. CASE atributai yra veiklos elementai ir kiti tinkami duomenys CASE. CASE gali būti paskirstyta

tarp kelių aktorių. Kiekviena padalinta CASE dalis turi vietinę būseną. Čia yra trys vietinės būsenos rūšys:

- *Veiksmo būsenoje* aktorius vykdo veiksmą CASE dalyje. Kiekvienam veiksmui turėtų būti bent viena veiksmo būsena, tačiau ir skirtingos veiksmų būsenos gali vykdyti tą patį veiksmą.
- *Laukimo būsenoje*, CASE laukia išorinio arba laikino įvykio.
- *Eilės būsenoje*, CASE laukia, kol aktorius bus laisvas bei pasiruošęs atlikti sekantį CASE veiksmą.

Aktoriai yra žmonės arba mašinos. Jie rūšiuojami pagal vaidmenys. Vaidmenys aprašo keletą charakteristikų, būdingų aktoriams. Žmonių atveju – vaidmuo gali remtis įgūdžiais, atsakomybe arba žmonių autoritetais. Mašinų – kompiuterio galimybėmis ir panašiai. Vaidmenys jungia aktorius ir veiksmus (veiklą).

1.2 DARBŲ SRAUTŲ VALDYMO SISTEMOS

1.2.1 DSVS TIKSLAS

Darbų srautų sistema valdo (apdoroja) procesus organizacijoje. CASE – valdo specifinius kliento reikalavimus atlikti vienokias ar kitokias paslaugas. Pavyzdžiui, pasirūpinimas draudimo ieškiniu, kad reikalavimai būtų priimti, apmokėti arba sumažinti, galiausiai, būtų atšauktas ir pats ieškinys. CASE kai kurie veiksmai yra atliekami tam tikra tvarka. Tam tikrų CASE klasių veiksmų eiliškumas yra apibrėžtas darbų srautų modelyje. DSVS veiklos procesų modelyje naudoja nuoseklią informaciją, kad nukreipti CASE po to, kai veiksmas nutraukiamas arba įvykis įvyko.

Įvykiai gali būti generuojami DSVS vartotojo. DSVS reaguoja į įvykius nukreipinėdamas CASE. Čia išskiriami keturi įvykių tipai:

- Tipinis įvykio tipas veiklos procese yra *nutraukimo įvykis*, kuris nurodo, jog tam tikras veiksmas buvo nutrauktas (čia nėra svarbu, kas buvo aktorius) ir, kad kitas veiksmas gali būti pradėtas. Kas bus tas sekantis veiksmas – tai yra

nustatoma DSVS modeliu pagrįsta veiklos proceso sistema. DSVS tuomet nukreipia CASE kito veiksmo link. Nutraukimo įvykiai nėra aprašomi UML 1.3.

- Šiek tiek painia terminologija, *baigties įvykis* yra aprašytas UML kaip įvykis, kuris pereina į laukimo būseną. Taigi UML 1.3 baigties įvykis įvyksta tuomet, kai visi įėjimo veiksmai ir veiklos-veiksmai (do-activities) toje būsenoje baigėsi. Veiklos-veiksmai UML 1.3 versijoje yra veiksmai vykdomi programinės sistemos. Tačiau veiksmų būsenas veiklos grafuose atspindi aktorių atliekami veiksmai, o ne DSVS. Taigi – veiklos-veiksmai čia nenaudotini.

- *Išorinis įvykis* yra atskiras kurios nors aplinkos sąlygos keitimas. Šis pakeitimas gali būti paremtas vardo suteikimu pačiam pasikeitimui arba sąlygai, kuri keičia:

  - *Pavadintas išorinis įvykis* yra įvykis, kuris suteikia unikalų pavadinimą;

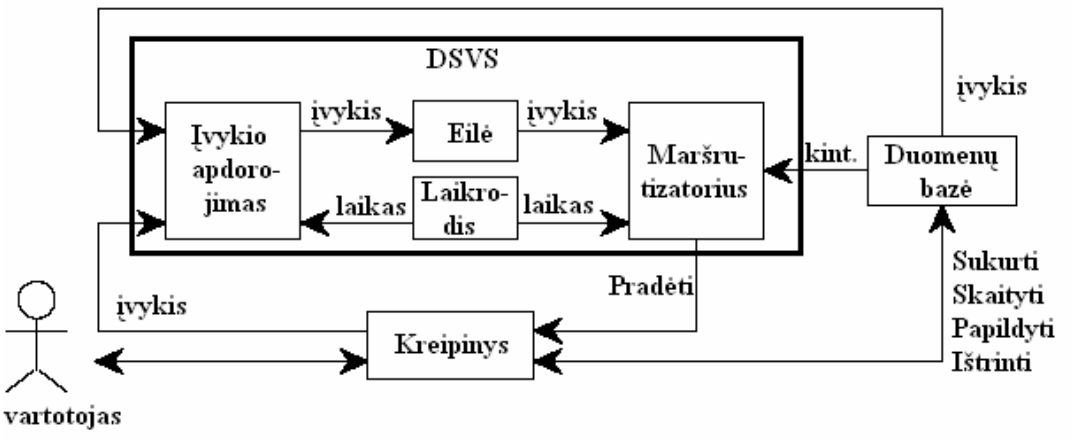
  - *Reikšmės pakeitimo įvykis* yra įvykis, kuris nusako, jog sąlyga tapo patenkinta (įgijo reikšmę „true“). Pavyzdyje (*1 pav.*) sąlyga [anketa gauta] nurodo reikšmės pakeitimo įvykį.

- *Laikinas įvykis* yra tas laiko momentas, kurio metu sistema tikisi reaguoti, pvz., į kokią nors kritinę situaciją. Pavyzdyje (*1 pav.*) žymė *[po(2 savaičių)]* nurodo kritinę situaciją, po kurios DSVS turėtų reaguoti praleisdamas *[pildyti anketą]* veiksmą. Laikini įvykiai yra generuojami pačios DSVS. Tarkime, kad DSVS turi vidinį laikrodį, kuris seka laiką.

1.2.2 DSVS ARCHITEKTŪRA

Semantika grindžiama tokia DSVS architektūra (*2 pav.*). Tai yra panašu į UML mašinų būsenos architektūrą. DSVS susideda iš dviejų komponentų, tai įvykio valdytojas ir maršrutizatorius. Šie komponentai veikia lygiagrečiai, taip pat jie tarpusavyje komunikuoja

eilės pagrindu. Įvykio valdytojas gauna įvykių ir juos įdeda į eilę. Įvykio valdytojas taip pat atsakingas už laikinų įvykių generavimą.

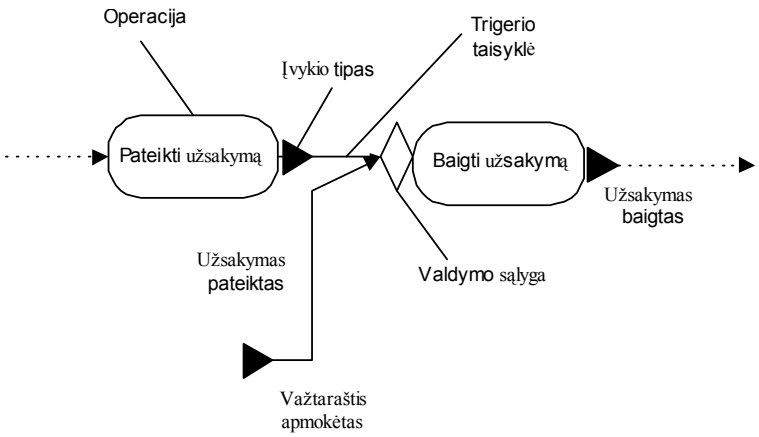


2 pav. Abstrakti DSVS vykdymo architektūra

Maršrutizatorius paima įvykį iš eilės ir nukreipia CASE. Nukreipinėjant pradedami nauji veiksmai bei generuojami nauji įvykiai. Maršrutizatoriui prireikia laiko apdoroti įvykį. Vadinasi, kai naujas įvestas įvykis atvyksta – maršrutizatorius tuo metu gali būti užimtas kito įvykio apdorojimu. Maršrutizavimo rezultatas gali startuoti naujus veiksmus. Tokio pobūdžio žinutė siunčiama svarbiems aktoriams. Duomenų bazė generuoja keitimo įvykius. Vartotojas ir/arba programa generuoja nutraukimo ir pavadintus išorinius įvykius. Baigties įvykius generuoja tik maršrutizatorius.

1.3 OBJEKTINĖS ORIENTACIJOS VEIKLOS PROCESŲ MODELIAVIMO METODAI (*MARTIN, ODELL*)

Veiksmų modeliavimo požiūriu vienas labiausiai išvystytų objektinės orientacijos (*OO*) metodų yra *Martin, Odell* metodas [5], [6], kurio pagrindu buvo sudarytos *UML* [9] veiklos (*activity*) diagramos. *Martin, Odell* modelis remiasi įvykių diagramomis. Įvykių diagramos išreiškiamos naudojant keturis, pagrindinius su procesais susijusius, elementus: operacijos, įvykius, trigerius ir valdymo sąlygas. Operacijos yra procesai, kurie vykdo būsenų pasikeitimus. Įvykių tipai apibrėžia būsenų pasikeitimus, kurie naudodami trigerių taisykles iš operacijų išrenka ir startuoja kitas operacijas. Valdymo sąlygos užtikrina, kad egzistuoja tam tikra būsena prieš sukeltą tam tikrą operaciją. Šios sąvokos apibendrinamos, naudojant grafinę notaciją, vadinamą įvykių diagramomis (*3 pav.*):



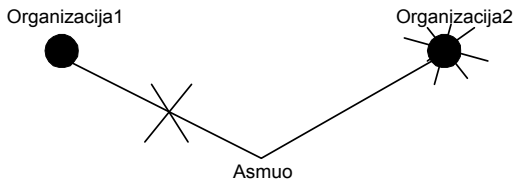
3 pav. Įvykių diagramos grafinis atvaizdavimas

Kiekvieną kartą iškvietus operaciją „pateikti užsakymą", tikimasi rezultato — įvykio „užsakymas pateiktas". Kai šis įvykis pasirodo, trigeris iškviečia operaciją "baigti užsakymą". Tačiau prieš operacijos iškvietimą yra tikrinama valdymo sąlyga. Šiame pavyzdyje valdymo sąlygos įvertinimas užtikrina, kad užsakymas buvo pateiktas ir klientas sumokėjo už jį. Tik tuomet gali būti vykdoma operacija "baigti užsakymą". Jei operacijai "baigti užsakymą" nėra aprašyta valdymo sąlyga, operacija įvykdoma tuoj pat po bet kurio įvykio atlikimo.

Įvykio tipas "apmokėtas važtaraštis" neturi jokios susijusios operacijos, todėl būdas, kuriuo apmokamas važtaraštis, yra už specifikacijos ribų. Įvykio tipas pateikiamas, siekiant parodyti proceso vykdymo stimulą.

1.3.1 ĮVYKIAI

Iš esmės, yra dvi būsenų pokyčių rūšys: pridėti (*add*) ir panaikinti (*remove*). Būsenos pokytis „pridėti" įveda naują objektą arba ryšį. Būsenos pokytis „panaikinti" pašalina objektą arba ryšį. Pavyzdžiui, asmuo (*4 pav.*) nusprendžia palikti „organizaciją 1" ir formuoja savo „organizaciją 2". Būsenos pokytis nereikalauja panaikinti „organizaciją 1", bet tik nutraukia ryšį su ja. Kitas būsenos pokytis reikalingas įvesti naują objektą „organizacija 2". Tuomet kitas būsenos pokytis reikalingas įtraukti ryšį tarp „asmens" ir „organizacijos 2".

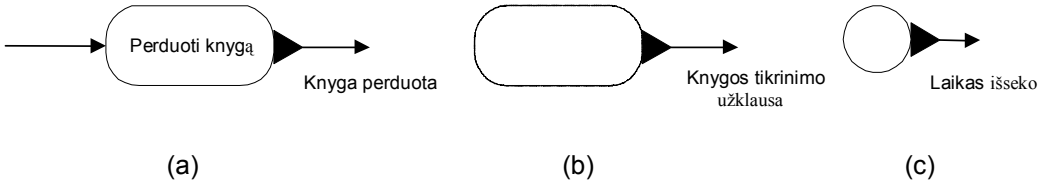


4 pav. Būsenų pokyčiai pridedant ir pašalinant objektus

Objektai gali būti kuriami, baigiami, grupuojami, išgrupuojami, pergrupuojami, jungiami, atjungiami, suvienijami, atskiriami. Kiekvienas įvykis gali būti klasifikuojamas kaip vienas iš šių būsenų pokyčių rūšių – kiekvienas įveda pridedamus ir pašalinamus objektus.

Įvykiai generuojami ne spontaniškai. Jie yra procesų, iššaukiančių būsenų pokyčius, pasekmė. Įvykiai gali būti vidiniai, išoriniai arba laiko.

Vidinis įvykis pasirodo kaip operacijos, esančios analitiko srityje, rezultatas. Pavyzdžiui, įvykis „Knyga perduota*"* yra operacijos „Perduoti knygą" rezultatas (*5 pav. (a)*).



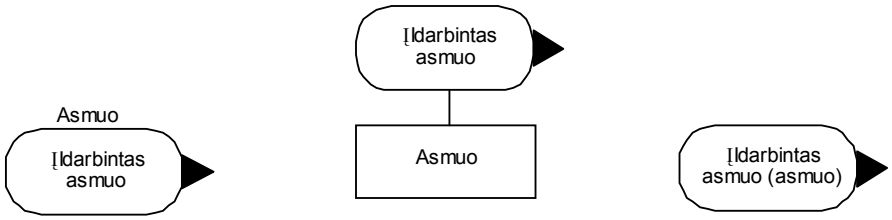
5 pav. Vidinių, išorinių ir laikinių įvykių išraiška

Išorinis įvykis yra išorinės analitiko srities operacijos rezultatas. Šie įvykiai identifikuojami tuomet, kai jie gali įtakoti analitiko sritį. Pavyzdžiui, išorinis įvykis (*5 pav. (b)*) rodo, kad tikrinimo užklausa yra už bibliotekos ribų, nors skaitytojui šis įvykis turėtų būti vidinis, nes knygos užklausa yra tai, ką skaitytojas daro. Tačiau bibliotekai šis įvykis yra išorinis, nes procesai, vadovaujantys įvykiui, priklauso skaitytojui, o ne bibliotekai.

Laiko įvykiai yra laiko operacijų rezultatas (*5 pav. (c)*). Pavyzdžiui, laiko įvykiu galėtų būti metų pabaiga, balandžio 14 diena, arba „mano gimtadienis". Priklausomai nuo to, ar įvykiai yra analitiko srityje, juos galima suskirstyti į vidinius ir išorinius.

1.3.2 OPERACIJOS

Tuo metu, kai įvykiai atlieka būsenų pokyčius, operacijos apdoroja elementus, kurie atlieka tuos pokyčius. Kiekvienai operacijai reikalingas objektas. Be jo operacija neturi paskirties. Visos operacijos reikalauja objektų kaip kintamųjų. Operacijų įvedimo kintamieji atitinka indeksus, kurie organizuoja operacijų žinias. *6 pav.* parodo tris grafinius operacijų dokumentavimo būdus.



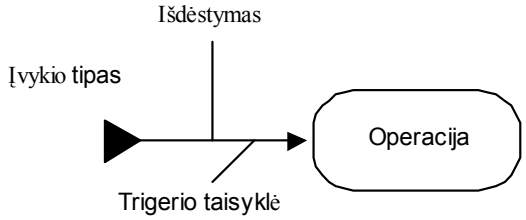
6 pav. Operacijų dokumentavimo būdai

OO analizėje tai užtikrina operacijų organizavimo mechanizmą. OO projektavime tai rodo klasę, prie kurios turėtų būti prijungta operacija.

OO operacijos gali būti suvokiamos kaip transakcijos. Transakcijos yra procesai arba procesų sekos, veikiančios kaip elementai, keičiantys objektų būsenas. Tokiu būdu operacijos turi būti baigtinės – nepriklausomai nuo to, ar tai truktų sekundes, dienas ar metus. Tokia operacijų specifikacija vadinama metodu. OO programavimo kalbose metodai specifikuojami kodo eilutėmis.

1.3.3 TRIGERIŲ TAISYKLĖS

Kaip jau buvo minėta, įvykiai yra būsenų pokyčiai. Jų pasirodymas iššaukia reakciją. Reakcija yra procesas. Ryšys tarp įvykio ir atsakomojo proceso vadinamas trigeriu. Kitaip tariant, kuomet pasirodo įvykis, iššaukiama operacija. Galima apibrėžti tam tikras trigerių taisykles. Jos pažymi, kad pasirodžius įvykiui – iškviečiama operacija.

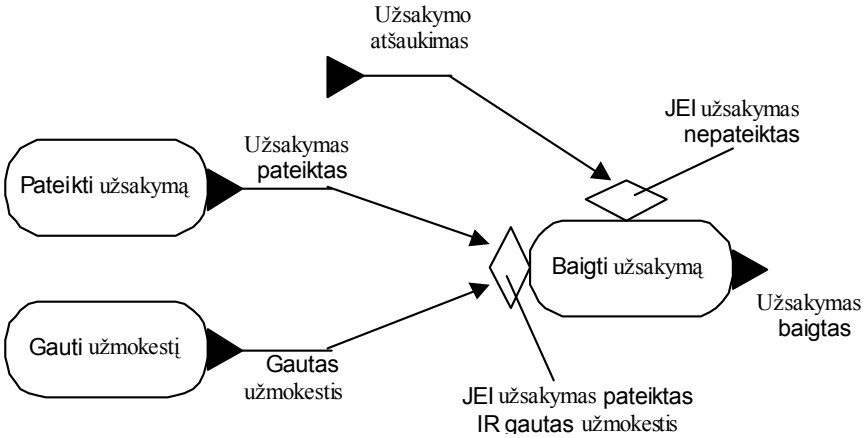


7 pav. Trigerių taisyklių priežastis ir pasekmė

Kaip parodyta *7 pav.* , kiekviena trigerio taisyklė apima tris elementus: įvykio tipą (priežastį), operaciją (veiksmą) ir atvaizdavimą, kuris perima vienos operacijos rezultatinius kintamuosius ir perkelia juos į tuos objektus, kurie reikalauja jų kaip įvedimo kintamųjų.

1.3.4 VALDYMO SĄLYGOS

Valdymo sąlygos yra apribojimai, kuriuos reikia patikrinti prieš vykdant operaciją. Operacijos prieš sąlygas gali būti laikomos operacijos dalimi, nes jos apribojimai turi būti patenkinti, iššaukiant operaciją.



8 pav. Valdymo sąlygos pavyzdys

Valdymo sąlyga negali būti operacijos dalis, nes ji gali būti ir neteisinga. Ji privalo būti teisinga, kuomet iškviečiamas atitinkamas trigeris (arba trigeriai). Įvykių diagramose valdymo sąlygos simbolis prijungiamas operacijos išorėje. Tai rodo, kad ji nėra operacijos dalis, o priklauso nuo trigerio (*8 pav.*).

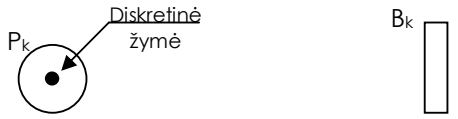
Be įvykių diagramų, *Martin, Odell* pasiūlė objektų srautų diagramas, kuriose vaizduojami operacijų apdorojami (įėjimo) ir gaunami (išėjimo) objektai. Ši idėja buvo pritaikyta *UML* veiklos diagramose su objektų srautais.

1.4 PETRI TINKLAI

Petri tinkluose dalyvauja darbai ir sąlygos [11]. Visa tai tinkluose vaizduosime pozicijomis. Jei darbas atliekamas – žymima diskretinė žymė (*9 pav. (a)*). Veiksmai, kurie yra atliekami, žymimi barjerais (*9 pav. (b)*). Petri tinklas aprašomas taip:

$$PN = \{P, B, A^{IN}, A^{OUT}, \Phi, M[0]\}, \quad (1)$$

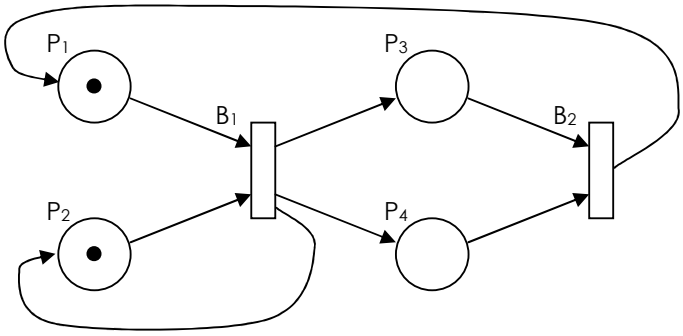
kur P – pozicijų aibė, B – barjerų aibė, A^{IN} – įėjimų aibė, A^{OUT} – išėjimų aibė, Φ – loginė taisyklė/formulė, M[0] – pradinė įėjimų matrica.



(a) (b)

9 pav. Pozicijų (a) ir barjerų (b) vaizdavimas Petri tinkluose

Ryšiai tarp pozicijų ir barjerų aprašomi grafais (*10 pav.*). Įėjimų aibė A^{IN} – tai lankai, kurie iš pozicijos P_k pereina į barjerą B_k. Išėjimų aibė A^{OUT} – tai lankai, kurie iš barjero B_k pereina į poziciją P_k. Φ (funkcionavimo logika) – tai taisyklė (dažniausiai loginė sąlyga Petri tinkluose „IR“, tačiau gali būti ir „ARBA“), kuri nurodo, kada ir kaip įvyksta barjeras. M[0] – pirminis žymėjimas – vektorius, kuris nurodo, kurioje pozicijoje yra diskretinė žymė. Tokios žymės gali būti kelios. Klasikiniai Petri tinklai nemodeliuoja laiko.



10 pav. Ryšiai tarp pozicijų ir barjerų

Pateiktu atveju (*10 pav.*) – barjeras B_1 yra aktyvus, nes visos pozicijos, kurių išėjimai įeina į barjerą B_1 – yra pažymėtos (visi darbai atlikti). Tokia būsena aprašoma taip:

P = {P1, P2, P3, P4}; B = {B1, B2}; M[0] = (1 1 0 0);

$$A^{IN} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix},$$

$$A^{OUT} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix},$$

kur stulpeliai atitinka pozicijas, o eilutės – barjerus.

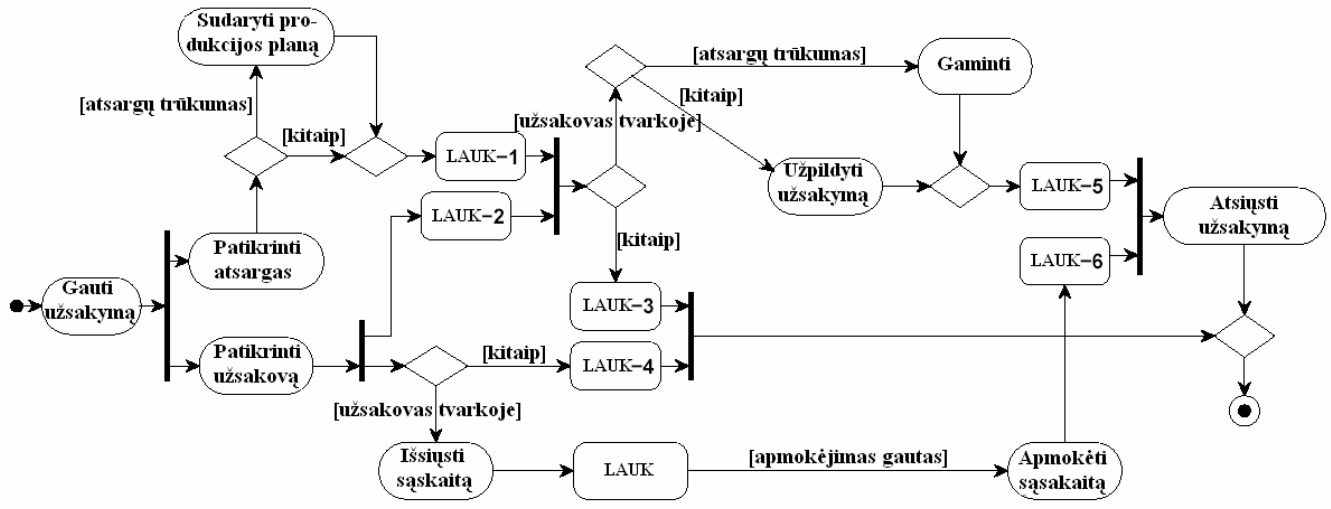
Perėjimų iteracinė procedūra aprašoma taip:

$$M[n] = M[n-1] + \beta[n] \cdot \left(A^{OUT} - A^{IN}\right), \tag{2}$$

kur β – barjerų vektorius.

$$M[1] = M[0] + \beta[1] \cdot \left(A^{OUT} - A^{IN}\right) = \begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 \end{pmatrix} \cdot \left(\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \right) =$$
$$= \begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 1 & 1 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 1 \end{pmatrix}$$

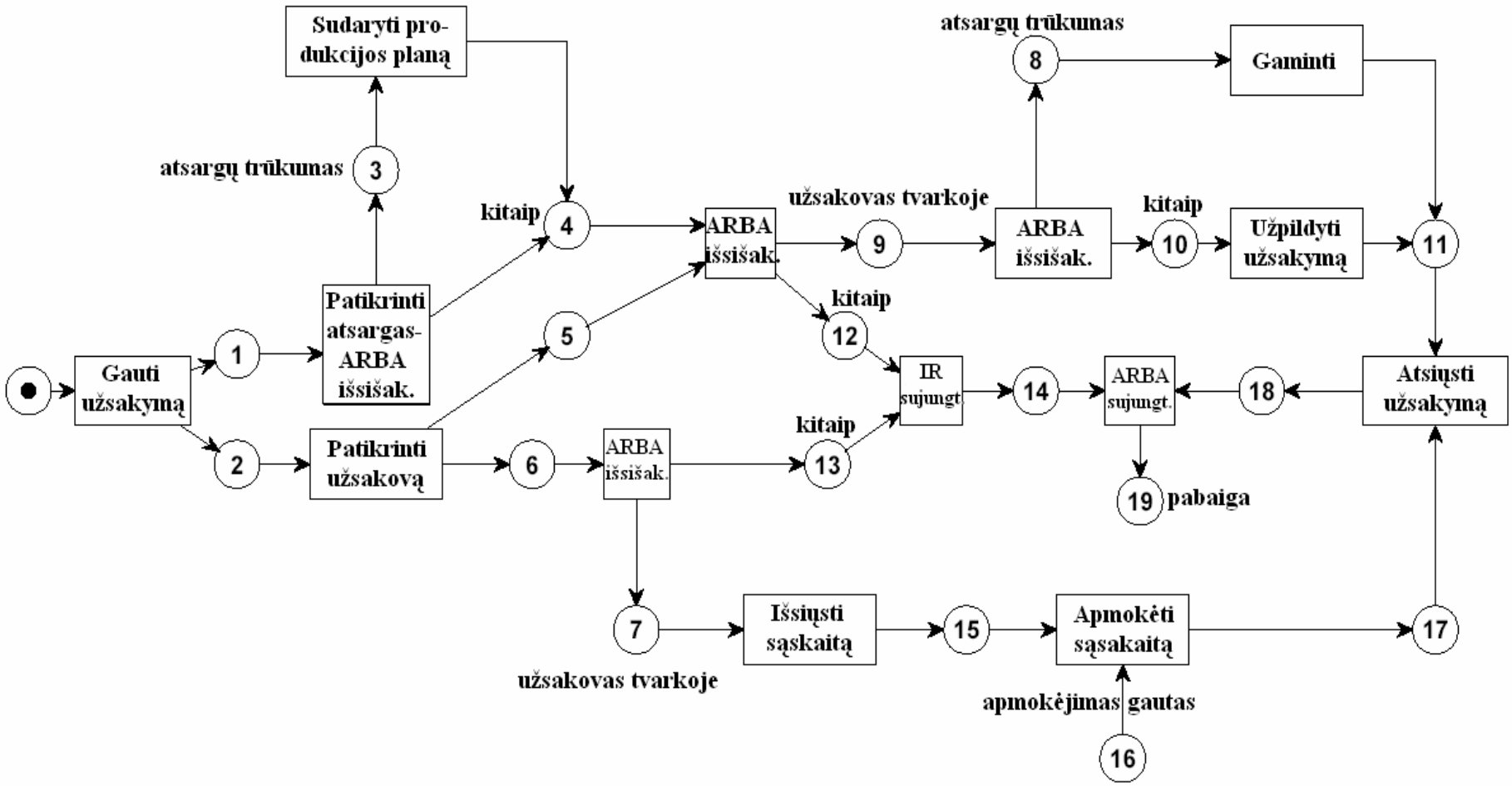
Eilė Petri tinklų variantų formuluota veiklos procesų modeliams. Kadangi įprastiniai Petri tinklai nėra pakankamai galingi veiklos procesų modeliavimui, tokie autoriai, kaip Van der Aalst [12], Ellis ar Nutt [13] – naudojo aukšto lygio Petri tinklus [14]. Tai laiku, duomenimis ir hierarchija išplėsti Petri tinklai. Čia labiau bus nagrinėjamas aukšto lygio Petri tinklo modelis, sudarytas Van der Aalst [12].



11 pav. Veiklos diagramos pavyzdys

11 paveiksle yra pavaizduota produkcijos užsakymo veiklos diagrama, iš kurios yra suformuotas Petri tinklo modelis (12 pav.). Šiame grafe naudojama tokia notacija: apskritimai vaizduoja vietas; stačiakampiai vaizduoja veiklas, IR-išsišakojimas, IR-sujungimas, ARBA-išsišakojimas arba ARBA-sujungimas. Globali būsena (Petri tinklų teorijoje vadinama žymėjimu) yra vaizduojama žymių paskirstymu tarp vietų. Tokia žymė nurodo, jog būsena yra aktyvi.

Van der Aalst [12] nepateikia aiškios vykdymo semantikos. Petri tinklų semantika ne modeliuoja aplinkos: žingsnis priklauso tik nuo esamos konfigūracijos, o ne nuo įėjimo įvykių. Aukšto lygio Petri tinklo semantikos atitinka technologiją, kai perėjimai įvyksta momentaliai. Perėjimai šiame modelyje vaizduoja veiksmo pradžią, bet ne veiksmo vykdymą. Viena gali būti ginčytina, jog aukšto lygio Petri tinklo semantika turi kažką panašaus į laikrodžio asinchroninę semantiką, kadangi perėjimas įvyksta iškart po aktyvacijos, kas būtent ir yra laikrodžio asinchroninėje semantikoje.



12 pav. Petri tinklo modelis sudarytas iš 11 pav.

1.5 ANALIZĖS IŠVADOS

1. UML veiklos diagramos pagal jų aprašytą semantiką gerai tinka kompiuterinėms sistemoms vaizduoti, bet ne visai tinka verslo procesams vaizduoti, nors yra gana plačiai tam naudojamos. Vaizduojant B2B verslo procesus, naudojamos įvairios UML diagramos, ir UML veiklos diagramoms suteikiama kitokia semantika.

2. VH (Veiklos Hipergrafai) neapima objektų srautų, kadangi jų semantika UML nėra pakankamai apibrėžta: kaip vyksta objektų srautų išskaidymas? Kas vyksta, kai veikla vykdoma daug kartų – ar vieno objekto kopija perduodama daugeliui veiklų, ar perduodama daug objektų? Ką reiškia objektų srautas veiklos išėjime – ar tie objektai sukuriami, ar atnaujinami? Kas vyksta, kai keli objektų srautai sujungiami į vieną? Ką reiškia, jei neparodytas objektų srautas – ar, kad jo nėra, ar, kad jis neparodytas?

   Veiklos hipergrafai priimama, kad objektai saugomi duomenų bazėje ir išreiškiami per proceso būsenos kintamuosius.

   B2B elektroniniame versle objektų srautai yra labai svarbūs, kadangi jie išreiškia siunčiamą informaciją apie būsenas – bendros duomenų bazės nėra, sąveikaujantys dalyviai naudojasi bendru kontekstu, kuris gali būti išreikštas standartiniu (arba bent jiems visiems suprantamu) žodynu. Todėl, modeliuojant B2B verslo procesus, objektų srautų ignoravimas netikslingas .

   2.1 Veiklos hipergrafų apribojimas, kad kiekviena veiklos diagrama turėtų tik vieną pabaigos būseną – nėra tikslingas, kadangi proceso pabaiga gali turėti daug variantų (pvz., sėkmingą ir nesėkmingą). Vienas pabaigos simbolis mažina aiškumą.

   2.2 Laukimo būsenų įvedimas apkrauna vaizdumą. Tačiau veiklos elemento semantika turi būti pakeista. Įvykdžius įėjimo veiksmą, perėjimas prie kitos veiklos galimas tik tada, kai:

   - jis vyksta automatiškai, tai yra, jį vykdo programinė įranga;
   - nesinchronizuojamas su kita veikla (priešingu atveju reikalingas laukimas).

Perėjimas visada turėtų reikšti laukimą (laukiama dalyvio įvykio ir/arba kitų veiklų užbaigimo), išskyrus automatinius perėjimus, kurie turėtų būti kaip nors pažymėti.

2.3 Ką reikėtų panaudoti iš veiklos grafų: naudingi apribojimai:

- iš kiekvienos veiklos gali būti tik vienas perėjimas į jungimą;
- kiekviena veiklos būsena turi turėti bent vieną įėjimo ir išėjimo perėjimą.

3. Petri tinklų semantika nėra pakankama verslo procesams modeliuoti:
    - negalima pavaizduoti įėjimo įvykių;
    - negalima pavaizduoti duomenų;
    - veiksmai turi būti vykdomi perėjimų metu, tai neatitinka B2B procesų semantikos, kai veiksmus turi vykdyti verslo dalyviai.

   Tuo galima paaiškinti, kodėl Petri tinklai nėra plačiai naudojami verslui modeliuoti.

   Tiek hipergrafai, tiek Petri tinklai formalizuoja veiklos procesą, tačiau daro jį mažiau intuityviai suprantamą, tai irgi svarbu aprašant verslo procesus, ypač kai aprašymą daro verslo analitikai.

4. Todėl pasirinkta šio darbo kryptis – patikslinti UML veiklos diagramų semantiką, naudojant kai kurias hipergrafų savybes, bei panaudoti hipergrafų galimybes verslo proceso modeliui patikrinti.

2 VEIKLOS PROCESŲ VYKDYMO MODELIAVIMAS, TAIKANT UML VEIKLOS GRAFUS

2.1 VEIKLOS PROCESŲ SEMANTIKA IR UML NOTACIJA

Standartinė veiklos procesų ir informacijos sistemų projektų modeliavimo kalba yra UML. UML veiklos diagrama (kuri dažnai vadinama darbų srautų diagrama) galima aprašyti organizacijos aktorių vykdomus procesus. UML veiklos diagramos elementai yra aktoriai, objektų būsenos, perėjimai, veiksmai. Ji tinka tiek projektuojant įprastas taikomąsias programas, tiek kompiuterizuojant darbų srautų valdymą. Gerai aprašyta veiklos diagrama duoda labai daug informacijos apie kompiuterizuojamus procesus: iš jos galima gauti panaudojimo atvejų, objektų tipų ir kitas diagramas, nuo kurių prasideda projektavimas.

UML pasirinkta todėl, kad tai nauja standartinė objektiškai orientuotų projektų aprašymo kalba, jungianti daugelio objektiškai orientuotų metodų savybes ir turinti labai dideles galimybes. Naudojant šią kalbą, galima projektuoti įprastas, vieno vartotojo vykdomas taikomąsias programas, ko neleidžia specialios darbų srautų apibrėžimo kalbos. Projektuojant darbų srautų apibrėžimus, reikia projektuoti ir vartotojų taikomąsias programas. UML įgalina projektuotoją abiem atvejais naudoti vieningą įrankį.

UML pagrindinė naudojama notacija:

Aktoriai;
Veiksmai;
Būsenos;
Priklausomybės;
Tikslai, problemos, galimybės;
Komunikacinis veiksmas.

2.1.1 AKTORIAI

Aktorius apibrėžia nuoseklų vaidmenų rinkinį, kurį esybės vartotojai gali atlikti, kuomet sąveikaujama su esybe. Aktorius gali būti skirtas atskiro vaidmens atlikimui, atsižvelgiant į kiekvieną panaudojimo atvejį, su kuriuo jis komunikuoja.

UML notacijoje (*13 pav. (b)*) standartinė aktoriaus piktograma yra žmogaus figūra su vardu po ja. Aktorius taip pat gali būti atvaizduotas kaip klasės kvadratas su raktiniu žodžiu „aktorius“ (*13 pav. (a)*) ir įprasta notacija visiems skyriams.

13 pav. Aktoriaus atvaizdavimo pavyzdys UML

Aktoriaus vardai turi atitikti skyrybos taisykles, naudojamas modelio tipams ir klasėms.

Galimi du ryšių tarp aktorių variantai: standartinis ryšys tarp aktorių ir standartinis ryšys tarp aktorių ir panaudojimo atvejų:

Asociacija – aktoriaus dalyvavimas panaudojimo atvejyje, t.y. aktorius ir panaudojimo atvejis komunikuoja vienas su kitu. Tai yra ryšys tik tarp aktoriaus ir panaudojimo atvejo. Asociacija tarp aktoriaus ir panaudojimo atvejo parodoma vientisa linija tarp aktoriaus ir panaudojimo atvejo.

Apibendrinimas – aktoriaus A apibendrinimas aktoriumi B rodo, kad A egzempliorius komunikuoja su tos pačios rūšies panaudojimo atvejo egzemplioriumi. Apibendrinimas tarp aktorių atvaizduojamas apibendrinimo rodykle, t.y. vientisa linija su uždara rodyklės galvute. Rodyklės galvutė nurodo apibendrinamą aktorių.

2.1.2 VEIKSMAI

Veiksmo būsena yra veiksmas su įėjimo ir mažiausiai vienu išėjimo perėjimu, įtraukiant besąlyginį įvykį išbaigiantį įėjimo veiksmą (tokių perėjimų gali būti keletas, jei jie turi sąlygas). Veiksmai negali turėti vidinių perėjimų ir išorinių perėjimų, paremtų apibrėžtais įvykiais. Šiuo atveju naudojami paprasti veiksmai. Įprastinis veiksmų naudojimas yra algoritmo vykdymo ar *darbo srautų valdymo* proceso modeliavimas.

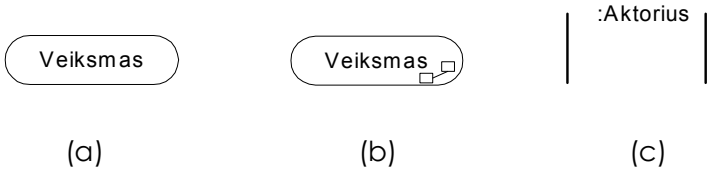
Veiksmas gali būti aprašytas natūralia kalba, pseudokodu ar programinės kalbos kodu. Jis gali naudoti tik jam priklausančio objekto atributus ir ryšius.

Pažymėtina, kad veiksmo notacija naudojama ne tik veiklos (*activity*) diagramose, bet ir būsenų diagramose. Veiklos diagramos yra atskiras būsenų diagramų atvejis.

UML notacijoje veiksmas atvaizduojamas forma su tiesiu viršumi ir apačia bei suapvalintais kampais (14 *pav. (a)*). Veiksmo išraiška nurodoma figūros viduje. Ji gali nebūti unikali diagramoje.

Subveiksmas (*subactivity*) naudojamas veiklos diagramoje. Jis nebaigiamas tol, kol nepasiekiama diagramos galinė būsena arba pasirodo trigerio įvykis perėjimuose, išeinančiuose iš subveiksmo būsenos. Jei veiklos diagramoje nėra trigerių įvykių, subveiksmas baigiamas, pasiekus galinę būseną. Veiklos diagrama gali būti paremta daugeliu subveiksmų būsenomis.

Subveiksmas atvaizduojamas kaip ir veiksmas, tik su ikona apatinio kampo dešinėje pusėje (*14 pav. (b)*). Pavadinimas–vardas atvaizduojamas simboliu ir jis gali nebūti unikalus diagramoje.



14 pav. Veiksmo (a), subveiksmo (b) ir juostos (c) atvaizdavimo pavyzdys UML

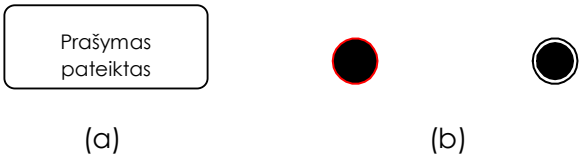
Veiksmai ir subveiksmai gali būti rikiuojami į juostas (*swimlanes*). Jos naudojamos veiksmų ir subveiksmų atsakomybės organizavimui, remiantis klase. Ji dažniausiai atitinka organizacinį vienetą arba vaidmenį veiklos modelyje.

Veiklos diagrama gali būti sudalyta į juostas (*14 pav. (c)*), kurių kiekviena atskiriama vertikaliomis linijomis iš abiejų pusių. Kiekviena juosta reiškia veiklos dalies atsakomybę, kurią atlieka vienas ar daugiau objektų. Santykinė juostų eilės tvarka neturi semantinės reikšmės, bet gali išreikšti panašumus. Kiekvienas veiksmas priskiriamas vienai juostai. Perėjimai gali kirsti jas.

2.1.3 BŪSENOS

Būsena yra sąlyga objekto arba sąveikos, kurios metu patenkinamos sąlygos, atliekami veiksmai arba laukiama įvykių, gyvavimas. Objektas vienoje būsenoje išlieka tam tikrą laiko intervalą.

UML būsena atvaizduojama stačiakampiu suapvalintais kampais (*15 pav. (a)*), kuriame nurodomas būsenos vardas bei objektas, kuris priklauso šiai būsenai.



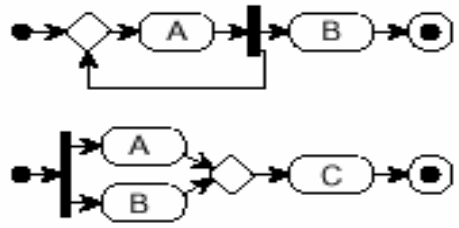
15 pav. Būsenų atvaizdavimo pavyzdys UML

UML poreikis (pradinė veiksmo būsena) ir patenkinimas (galinė veiksmo būsena) turi atitinkamus žymėjimus (*15 pav. (b)*).

2.2 UML VEIKLOS GRAFAI

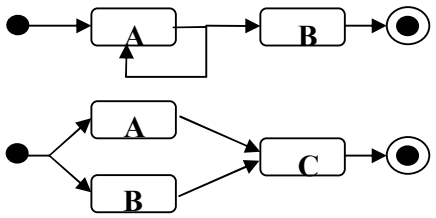
UML veiklos grafas susideda iš viršūnių ir lankų. Kai kurios viršūnės yra naudojamos paprastų lankų sujungimui į kompleksinius lankus. Tokio tipo viršūnės vadinamos pseudoviršūnėmis, o kompleksiniai lankai, kuriuos tos viršūnės sukuria – UML vadinami sudėtiniu keliu (perėjimu) [8], p.2-147].

UML veiklos diagramose naudojant sprendimų taškus ir sinchronizavimo juostas, galima pavaizduoti sudėtingas situacijas, kai vienu metu turi būti aktyvios kelios viršūnės (*16 pav.*).



16 *pav. Veiklos grafai*

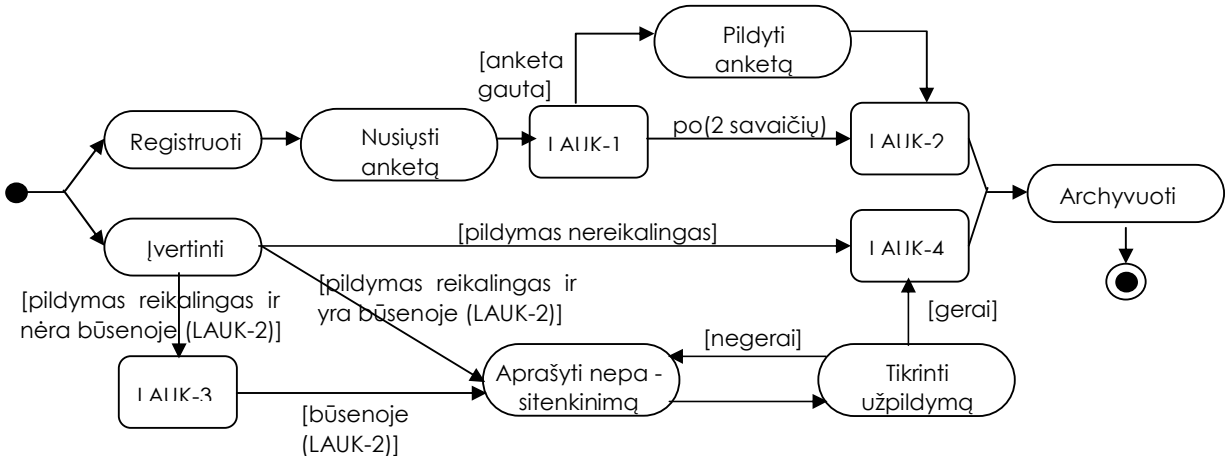
Tokiu atveju korektiškumą sunku patikrinti. Todėl veiklos grafas privedamas prie hipergrafo (*17 pav.*).



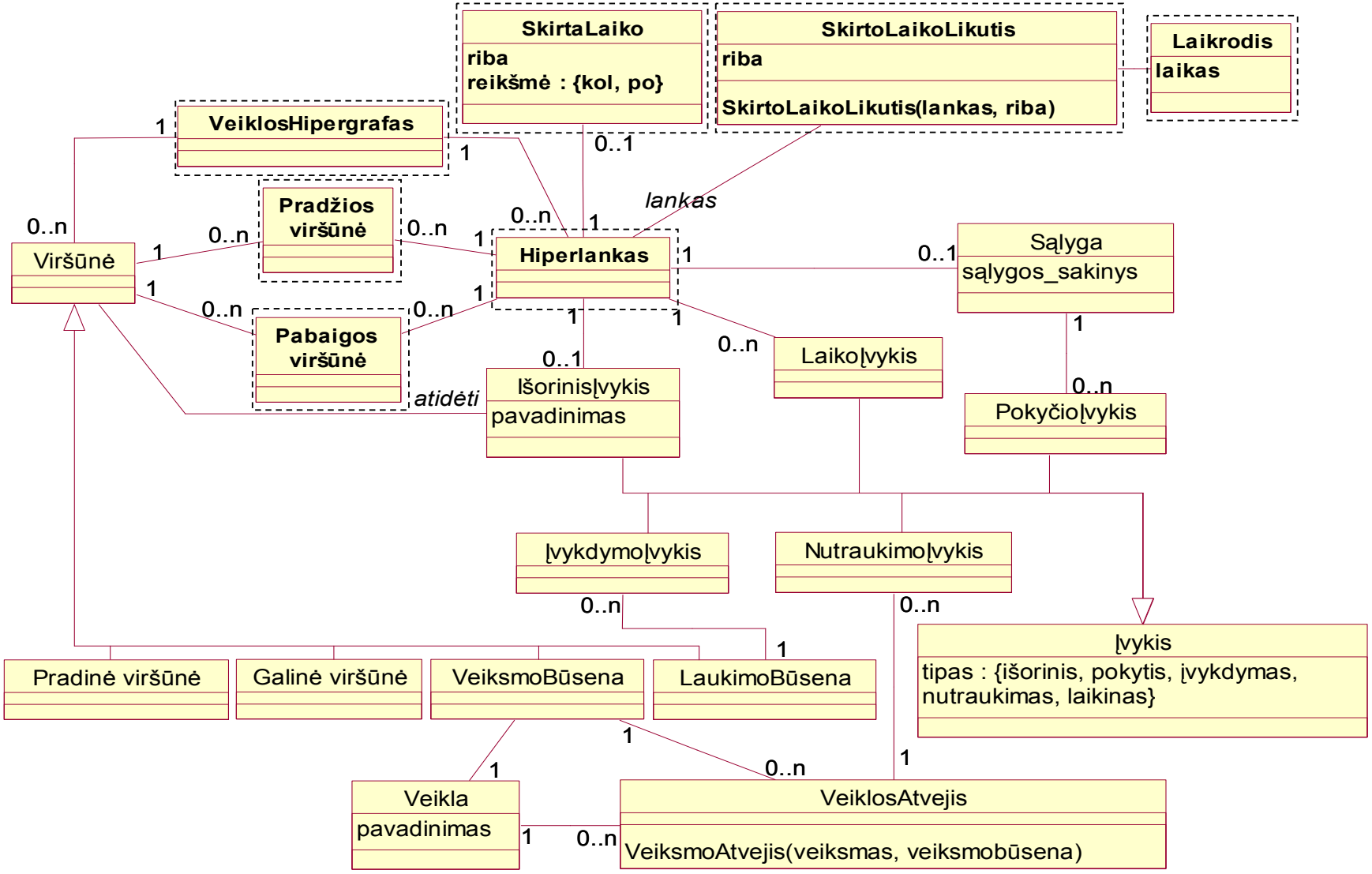
17 pav. Hipergrafai, sudaryti iš veiklos grafų 16 paveiksle

2.3 VEIKLOS HIPERGRAFAI

Apibrėžiant veiklos grafų semantiką, visų pirma reikia veiklos grafus privesti iki veiklos hipergrafų (*18 pav.*). Paveiksle (*19 pav.*) yra pavaizduotas veiklos hipergrafų metamodelis. Punktyriniais stačiakampiais apvesti bei pastorintu šriftu pavaizduoti laukai, kurie būdingi veiklos hipergrafams. Veiklos hipergrafas susideda iš viršūnių ir hiperlankų. Hiperlankas yra lankas, kuris gali turėti daugiau nei vieną įėjimo ir daugiau nei vieną išėjimo viršūnę.



18 pav. Veiklos hipergrafas sudarytas iš veiklos grafo (1 pav.)



19 pav. Veiklos hipergrafo metamodelis

Hipergrafo viršūnės gali būti tik tokių tipų: veiksmo būsenos viršūnė, laukimo būsenos viršūnė, pradinės būsenos viršūnė arba pabaigos būsenos viršūnė. Kiekviena veiksmo būsenos viršūnė yra susieta su tam tikra veikla. Laukimo viršūnė įvedama tuo atveju, kai veiklos vykdymo pradžia priklauso nuo išorinio įvykio arba nuo sinchronizavimo su kitais veiksmais, tai yra, veiksmo vykdymas negali prasidėti automatiškai, įvykdžius prieš jį einantį veiksmą. Sudarant veiklos procesų modelius, ši aplinkybė dažnai ignoruojama, todėl gaunamas neteisingas proceso modelis (taip vyktų procesas, jei veiksmus atliktų programa, tačiau veiklos procesų valdymo sistemose veiksmus dažnai atlieka vartotojai, ir proceso vykdymo metu laukiama, kol vartotojas įvykdys tam tikrą veiksmą).

2.4 NUO VEIKLOS DIAGRAMOS PRIE VEIKLOS HIPERGRAFO

Veiklos diagramų privedimas prie veiklos hipergrafų susideda iš trijų žingsnių:

- kai kurių sintaksinių išraiškų perrašymas;
- hierarchijos eliminavimas;
- hiperlankų sujungimas.

2.4.1 KAI KURIŲ SINTAKSINIŲ IŠRAIŠKŲ PERRAŠYMAS

Perrašant sintaksines išraiškas, sprendimų taškų sąlygos perrašomos kiekvienam lankui. Kiekviena laukimo pabaigos sąlyga priskiriama tiksliai tam tikram lankui. Taigi, *[kitaip] (else)* yra pakeičiamas tokia išraiška, kuri jį apibūdina. Antra – keičiame kiekvieną *po(t)* hiperlankų žymę *h* su *po(t, h)*. Šis pakeitimas *po* ribojimą padaro unikaliu abejiems hiperlankams, užtikrinant, kad jeigu abu hiperlankai turi tą pačią *po* žymę, čia nebus jokios painiavos suprasti, kuriam iš šių lankų skirta laiko baigtis.

2.4.2 HIERARCHIJOS ELIMINAVIMAS

Eliminuojant hierarchijas, t.y., sudėtines veiklos viršūnes, kiekviena sudėtinė veiklos viršūnė, kuri buvo atvaizduota kita veiklos diagrama, įtraukiama į pagrindinę veiklos diagramą. Jungiant sudėtinę viršūnę su kitomis, perėjimai turi būti necikliški. Pradinė ir galinė atskiros diagramos viršūnės pagrindinėje veiklos diagramoje virsta ARBA viršūnėmis, kurios suformuojamos taip: kiekvienas sudėtinis lankas, kuris įeina į veiklos diagramą, sujungiamas su kiekvienu sudėtiniu lanku, kuris išeina iš pradinės viršūnės. Kiekvienas sudėtinis lankas, kuris išeina iš atitinkamos veiklos diagramos, sujungiamas su kiekvienu sudėtiniu lanku, kuris įeina į vieną iš pabaigos viršūnių.

2.4.3 HIPERLANKŲ SUJUNGIMAS

Įvedant hiperlankus, pirmiausia nustatomi sudėtiniai lankai. Sudėtinis lankas – tai toks lankas, kuris susideda iš lankų, kurie yra sujungti su IR (išsišakojimas/sandūra) ir ARBA (sprendimas/jungimas) viršūnėmis ir tenkina apribojimus:

- jeigu lankas įeina arba išeina iš IR viršūnės, jis yra sudėtinio lanko dalis;
- jeigu lankas įeina arba išeina iš OR viršūnės, tuomet yra vienas sudėtinis lankas kuris išeina arba įeina į ARBA viršūnę.

Ne kiekvienas sudėtinis lankas, kuris tenkina aukščiau minėtas taisykles, yra gerai apibrėžtas. Sudėtiniai lankai nėra gerai apibrėžti, jeigu jie yra cikliniai. Ciklai yra blogai dėl šių priežasčių:

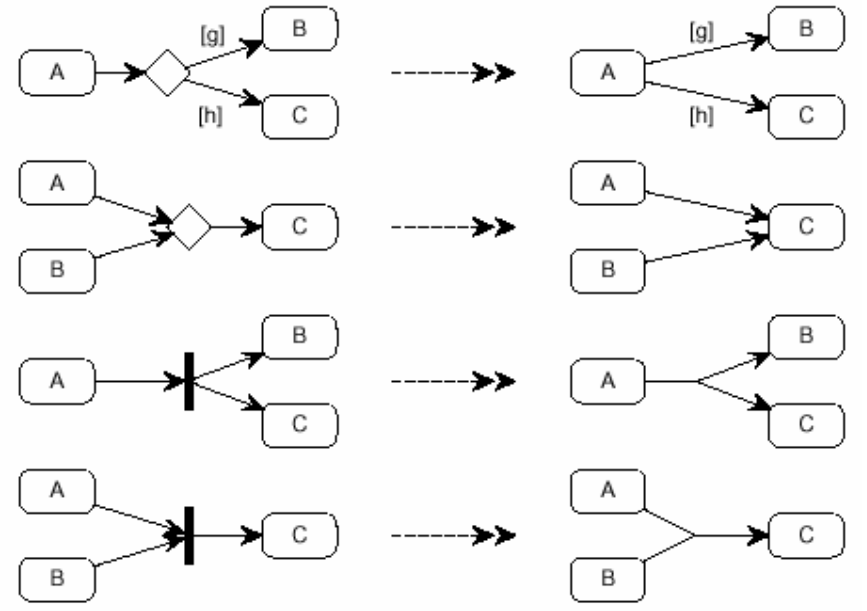
- Kartais sudėtiniai lankai su ciklais yra nepasiekiami: sudėtinis lankas neprasideda be pseudoviršūnės. Pavyzdyje (*20 pav. (a)*) yra du sudėtiniai lankai, pavadinimais {e1, e2, e3} ir {e2, e3}. Sudėtinis lankas {e2, e3} nėra gerai apibrėžtas, kadangi jis nepasiekiamas;
- Kartais sudėtinis lankas su ciklu yra pasiekiamas, bet negali būti vykdomas. Pavyzdžiui paveiksle (*20 pav. (b)*) yra vienas sudėtinis lankas, pavadinimu

{e5, e6, e7,e 8}. Šis sudėtinis lankas yra pasiekiamas, bet nėra vykdomas. Lankas e6 gali būti vykdomas po to, kai e7 bus įvykdytas. O e7 gali būti įvykdytas tik po to, kai e6 bus įvykdytas (atminkite, jog lankas pažymi eigos seką). Taigi, e6 ir e7 negali būti vykdomi, vadinas ir sudėtinis lankas negali būti vykdomas.

Sudėtinį lanką galima atvaizduoti hiperlanku. Hiperlankas apibrėžia įėjimo ir išėjimo viršūnes (iš kur į kur pereinama perėjimo metu). Hiperlankai neturi pseudoviršūnių.

20 pav. Du apibrėžti sudėtiniai lankai

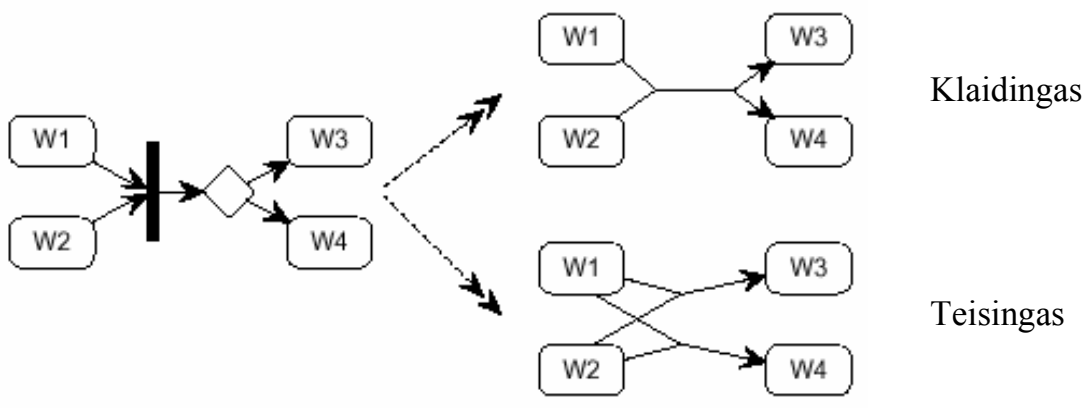
Veiklos grafas privedamas prie hipergrafo, eliminuojant pseudoviršūnes. Eliminavimo taisyklės grafiškai parodytos (*21 pav.*).



21 pav. Pseudviršūnių eliminavimas

Taigi, hiperlankai sudaromi nuosekliai performuojant veiklos diagramos pseudoviršūnes. ARBA viršūnė su *n* įėjimų arba *n* išėjimų lankų paverčiama į *n* naujų hiperlankų. IR viršūnė paverčiama vienu nauju hiperlanku. Paveiksle (*21 pav.*) yra pavaizduoti paprasčiausi sudarymai.

Vienintelis sunkumas iškyla tuomet, kai IR viršūnė jungiama su ARBA viršūne. Tuomet veiksmų seka yra labai svarbi: vykdant IR viršūnę prieš ARBA viršūnę, gaunamas skirtingas rezultatas nei vykdant ARBA viršūnę prieš IR viršūnę. Pavyzdžiui, veiklos diagramoje pseudoviršūnės, kurios pavaizduotos *22 paveikslo* kairėje pusėje, gali būti vykdomos dviem būdais.



22 pav. Du galimi hiperlankų sudarymai

Tuo atveju, kai ARBA viršūnė vykdoma prieš IR viršūnę, sudaryto hipergrafo vaizdas pateiktas dešiniame viršutiniame kampe, priešingu atveju – dešiniame apatiniame kampe. Pradinė veiklos diagrama aiškiai rodo, jog bus pasirinkta W3 arba W4 viršūnė, tačiau jokiu būdu ne abi. Taigi, IR viršūnės turi būti vykdomos prieš ARBA viršūnes.

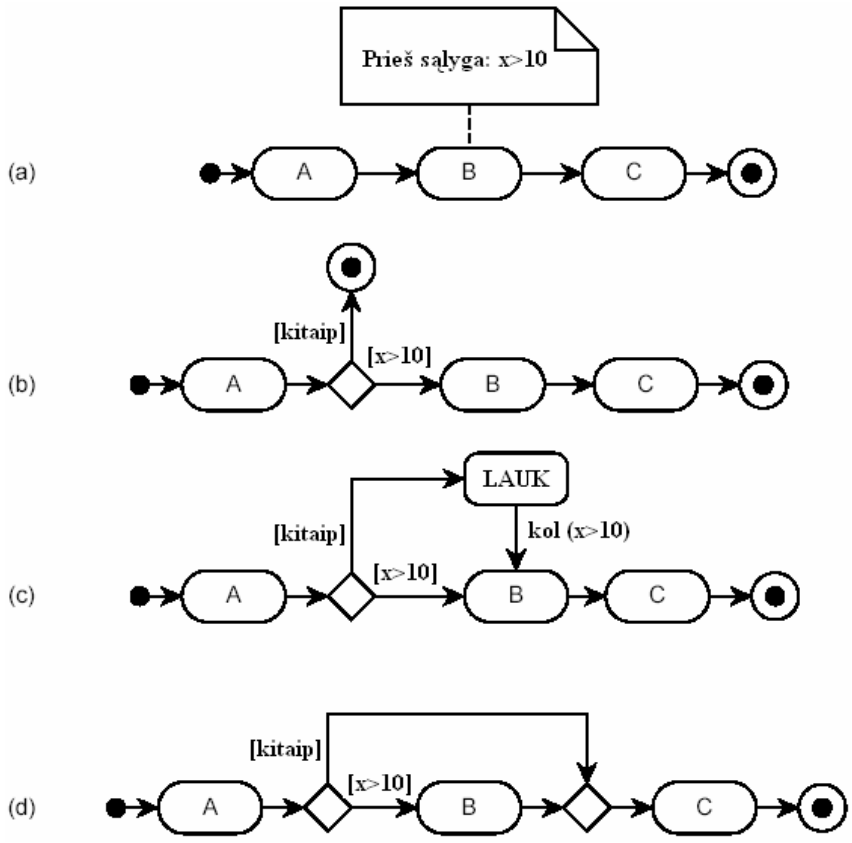
2.5 VEIKLOS VIRŠŪNIŲ APIBŪDINIMAS

Aukščiau buvo aprašoma lankų naudojimo semantika veiklos diagramose. Šiame skyriuje bus sutelktas dėmesys ties viršūnių semantika (tiksliau – ties veiklos viršūnių semantika).

Darykime prielaidą, jog kai veikla tampa aktyvi – ji po kažkurio laiko pasibaigs. Vienintelis tinkamas veiklos vykdymo aspektas yra toks: kai veikla pasibaigia – kai kurie valdymo duomenys gali būti pakeisti.

Veiklos elgesį mes specifikuosime klasifikuojant pagal „prieš“ ir „po“ sąlygas. „Prieš“ sąlyga yra loginė išraiška, kuri naudojama DSVS nuspręsti, kada veikla gali būti pradėta. „Po“ sąlyga yra loginė išraiška, kuri naudojama DSVS nuspręsti, kada veikla turi būti pabaigta. Tiek „prieš“ ir „po“ sąlygos remiasi valdymo duomenimis.

Deja, tiek „prieš“ ar „po“ sąlygos neturi standartinės semantikos. T.y., jog negali būti interpretuojamos skirtingai su skirtingomis DSVS. Pavyzdžiui, jeigu proceso (*23 pav. (a)*) veikla A pasibaigia ir x = 5, tuomet kai kurios DSVS sustabdys visą darbų srautų veiklą. Procesas pavaizduotas *23 (a) paveiksle* elgiasi panašiai kaip ir procese, pavaizduotame *23 (b) paveiksle*. Kitos DSVS lauks tol, kol x kintamasis įgys reikšmę didesnę už 10 (*23 pav. (c)*). Tačiau kitos DSVS praleis B ir pereis prie C veiklos (*23 pav. (d)*). Lygiai taip pat „po“ sąlyga irgi turi skirtingas interpretacijas. Jei „po“ sąlyga neįvyksta – tuomet DSVS gali nuspręsti, jog veikla turi būti pakartota arba visas procesas sustabdytas.



23 pav. Darbų srautai su išankstine sąlyga ir trejomis galimomis jos interpretacijomis

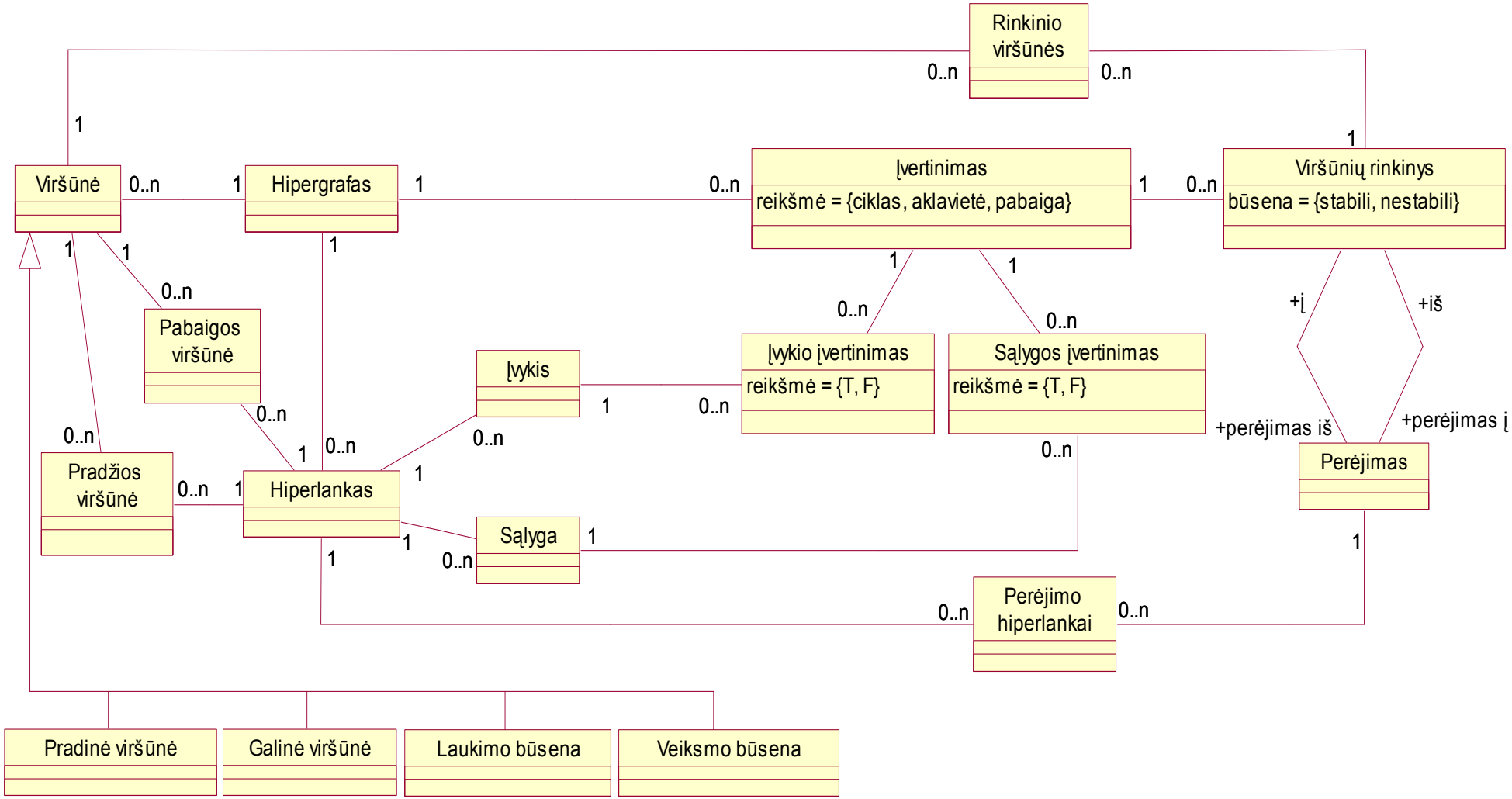
Kadangi „prieš“ ir „po“ sąlygų semantika yra tokia neaiški, nusprendėm, jog nemodeliuosime jų tiesiogiai taip, kaip pavaizduota *23 (a) paveiksle*, tačiau naudosimės aiškiais sąlygos sakiniais (*23 pav. (b)-(c)*.).

2.6 FUNKCIONALUMO REIKALAVIMŲ TIKRINIMAS

Šiame skyriuje pateiktas tikrinimo meta modelis, sudarytas naudojant UML notaciją (*24 pav.*). Šis modelis neapima veiklos diagramų pervedimo į hipergrafus sąvokų.

Vėliau pateikiamas pats veiklos hipergrafo tikrinimas, susidedantis iš tokių pagrindinių etapų:

- Grafo struktūros tikrinimas;
- Vykdymo tikrinimas.

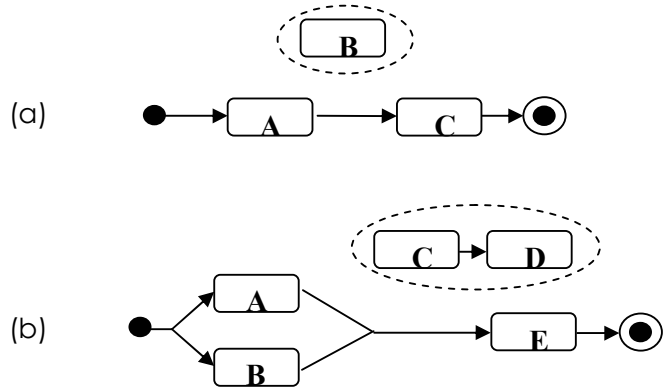


24 pav. Tikrinimio metamodelis

2.6.1 GRAFO STRUKTŪROS TIKRINIMAS

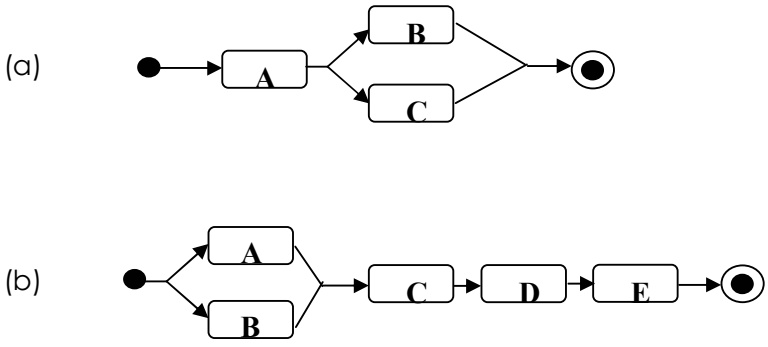
Struktūros tikrinimas yra gana paprastas. Jo metu galima iš karto identifikuoti netinkamą grafo sandarą. Tam tikslui, pradedant nuo pradinės viršūnės, nustatomos viršūnės, į kurias galima patekti per 1, 2, … žingsnius. Tikrinimas nutraukiamas, jei negalima pereiti į naujas būsenas arba perėjimų seka pradeda cikliškai kartotis.

Jei buvo pasiekta galinė būsena, galima nustatyti nepasiektas viršūnes. Tai gali būti „kabančios“ viršūnės (neturinčios įėjimo hiperlanko su bent viena pereita viršūne (*25 pav. (a)*)) arba viršūnių seka, kurios pradžios viršūnė taip pat neturi įėjimo hiperlanko su bent viena pereita viršūne (*25 pav. (b)*).



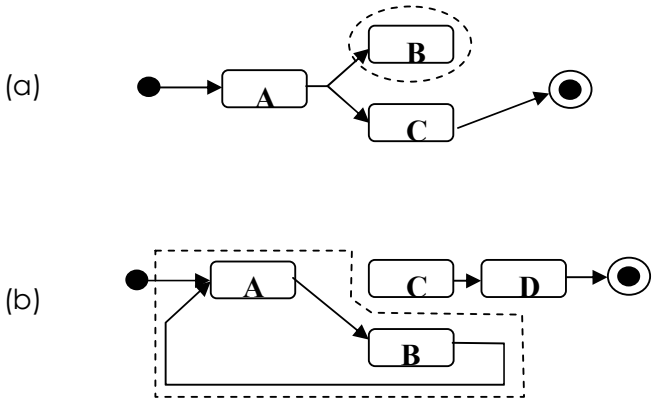
25 pav. Hipergrafe apvestos punktyriniu ovalu pavaizduotos „kabančios“ viršūnės

Tokias viršūnes reikia eliminuoti arba sujungti hiperlanku(-ais) su pradžios viršūne (*26 pav.*).

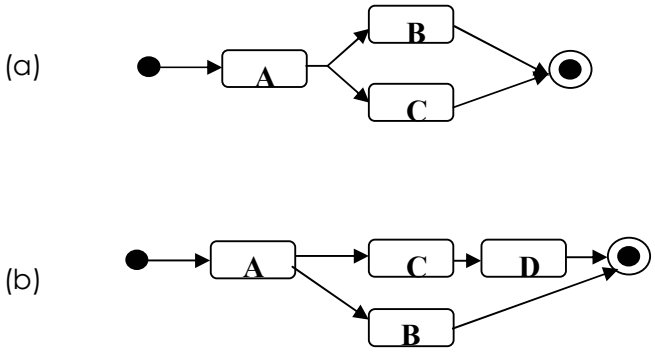


26 pav. „Kabančių“ viršūnių sujungimas hipergrafe, pavaizduotame 24 paveiksle

Jei nebuvo pasiekta galinė viršūnė, galima identifikuoti aklavietes (*27 pav. (a)*) arba ciklus (*27 pav. (b)*) ir pakoreguoti hipergrafo struktūrą (*28 pav.*).



27 pav. Aklavietės (a) bei ciklai (b) aptikti hipergrafe (apibraukti punktyru)



28 pav. Hipergrafo (27 pav.) koregavimas, eliminuojant aklavietes (a) bei ciklus (b)

2.6.2 VYKDYMO TIKRINIMAS

Jei struktūrinis tikrinimas bei hipergrafo atliktos korekcijos sėkmingai atliktos, tuomet galima pereiti prie vykdymo tikrinimo algoritmo, kuris konceptualiai pavaizduotas *29 paveiksle*. Tokį tikrinimą galime vadinti dinaminiu patikrinimu.

Dinaminio tikrinimo metu sistemos veikimas imituojamas nuosekliai fiksuojant, kokie perėjimai įvyksta išorinių įvykių įtakoje ir į kokias būsenas pereina sistema kiekvieno žingsnio metu. Laikoma, kad išoriniai įvykiai patenka į sistemą logiškai nuoseklia tvarka ir laukia eilėje. Sudėtinį žingsnį iššaukia išorinis įvykis. Žingsnio metu galima užfiksuoti pasiektas viršūnes (tai bus tos viršūnės, į kurias ateina įvykę perėjimai).

Proceso būseną nusako pasiektų viršūnių rinkinys. Veiksmų viršūnės įvykdomos per 1 žingsnį, o laukimo viršūnės įvykdomos tada, jei atėjo reikiamas išorinis įvykis arba yra įvykę atitinkami perėjimai. Užfiksavus viršūnes, analizuojami galimi perėjimai. Perėjimas galimas tada, kai įvykdytos visos įėjimo viršūnės ir tenkinamos perėjimo sąlygos. Jei negalima pereiti prie naujo viršūnių rinkinio – sistema yra stabili. Tuomet imamas naujas išorinis įvykis ir vėl fiksuojamas vykdymas, kol pasiekiama galinė viršūnė (identifikuojamas sėkmingas vykdymas). Jei baigiasi visi išoriniai įvykiai, o dar galinė viršūnė nėra pasiekta – tuomet identifikuojama aklavietė. Ciklišk ai besikartojant viršūnių rinkiniams – identifikuojamas ciklas.

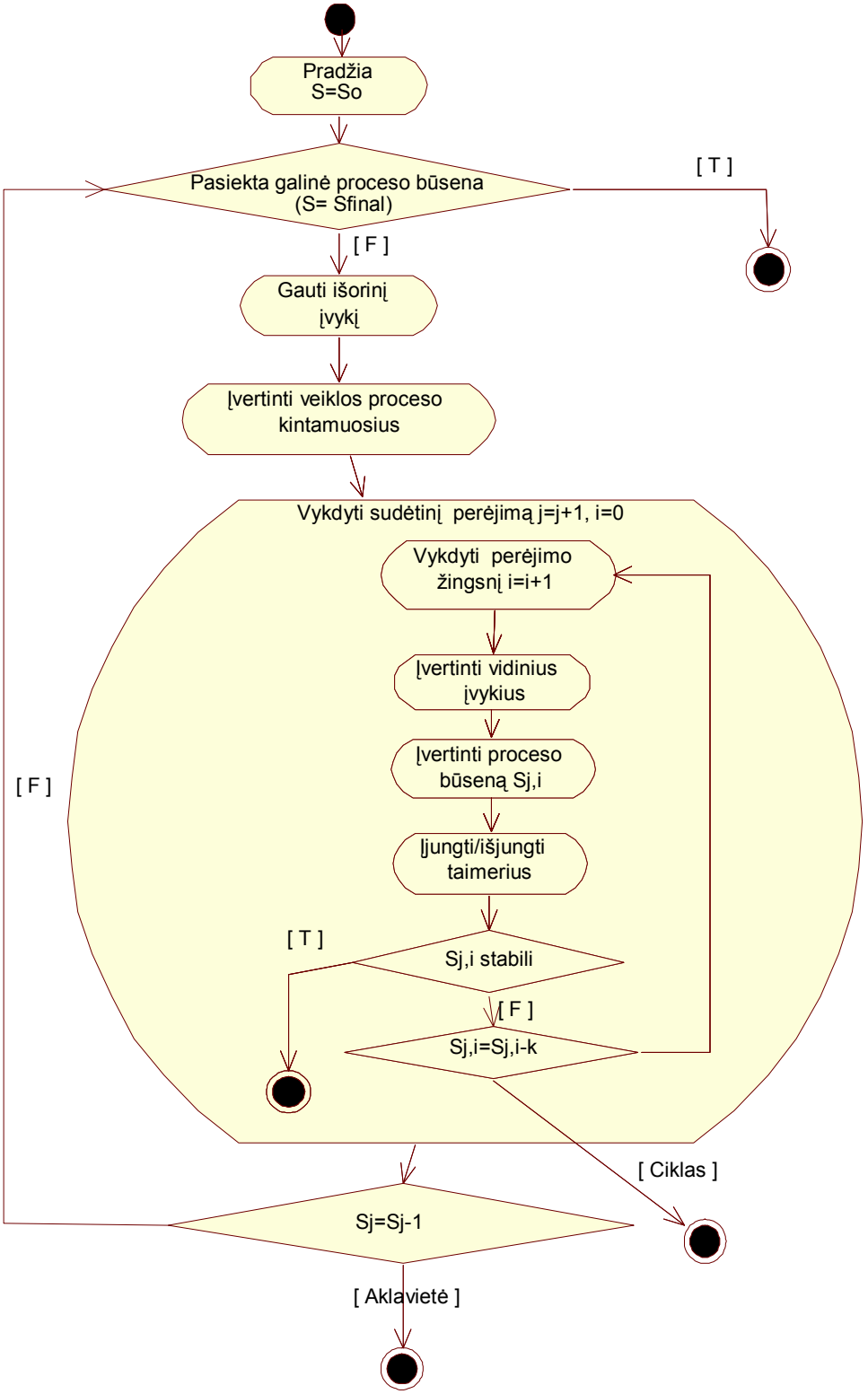
Žemiau pateiktame vykdymo tikrinimo algoritme (*29 pav.*) buvo naudotos šios išraiškos:

S – viršūnių rinkinio būsena;

j – rinkinio numeris

i – žingsinis;

k – viršūnių rinkinio būsena, kuri pasikartoja.



29 pav. Vykdymo tikrinimas

Atliekant vykdymo tikrinimą, reikia perrinkti daug variantų, kadangi reikia modeliuoti tokius atvejus, kai kiekviena sąlyga gali būti patenkinta arba nepatenkinta ir išorinis įvykis gali pasirodyti arba nepasirodyti. Bet kokiu atveju turi būti pasiekta kuri nors galinė viršūnė. Kita problema – didelis viršūnių skaičius. Siekiant sumažinti tikrinimo apimtį, veiklos hipergrafą galima sumažinti (redukuoti). Apimčiai sumažinti visumoje taikomos tokios taisyklės:

- Nagrinėjami tik reikšmingi išoriniai įvykiai logine tvarka;
- Laikoma, kad du išoriniai įvykiai negali pasirodyti vienu metu;
- Iš hipergrafo pašalinamos nuoseklios viena po kitos einančių viršūnių sekos. Tikrinimo metu jas galima sujungti, nes hipergrafo korektiškumui jos neturi įtakos;
- Iš hipergrafo pašalinamos lygiagrečiai vykdomos viršūnių sekos, kurių įėjimo lankai turi bendrą pradinę viršūnę ir išėjimo lankai turi bendrą įėjimo viršūnę.

3 EKSPERIMENTINIS SISTEMOS MODELIAVIMAS, TAIKANT PASIŪLYTĄ MODELIAVIMO METODĄ

3.1 SISTEMOS TIKSLŲ ANALIZĖ

Pasiūlytam veiklos procesų modeliavimo metodui išbandyti pasirinktas „Wing Tsun kungfu klubo“ (toliau WT klubas) modelis. Išskirti tokie pagrindiniai WT klubo tikslai, funkcijos, uždaviniai:

- Rengti pažintinio pobūdžio, kvalifikacinius ir kitokius seminarus ar renginius;
- Suteikti kvalifikuoto trenerio paslaugas Wing Tsun kungfu kovos menui apmokyti;
- Suteikti informaciją apie įvyksiančius seminarus ar renginius;
- Parduoti metodinę ar kitokią medžiagą apie Wing Tsun kungfu meną;
- Dalyvauti seminaruose ar renginiuose;
- Naudotis kvalifikuoto trenerio paslaugomis;
- Gauti informaciją apie įvyksiančius seminarus ar renginius, susijusius su WT kovos menu;
- Įsigyti metodinę ar kitokią medžiagą apie Wing Tsun kungfu meną;

WT klubui priklausantys nariai – tai fiziniai ar juridiniai asmenys (toliau *narys*), kurių tikslai:

- Seminaruose bei renginiuose įgyti pageidautiną kiekį informacijos, keltis kvalifikaciją;
- Naudotis kvalifikuoto trenerio paslaugomis, tobulinant savo asmeninius įgūdžius;
- Gauti informaciją apie įvyksiančius seminarus ar renginius, susijusius su WT kovos menu;
- Įsigyti metodinę ar kitokią medžiagą apie Wing Tsun kungfu meną;

Taigi yra du pagrindiniai aktoriai: *WT klubas* ir *narys*.

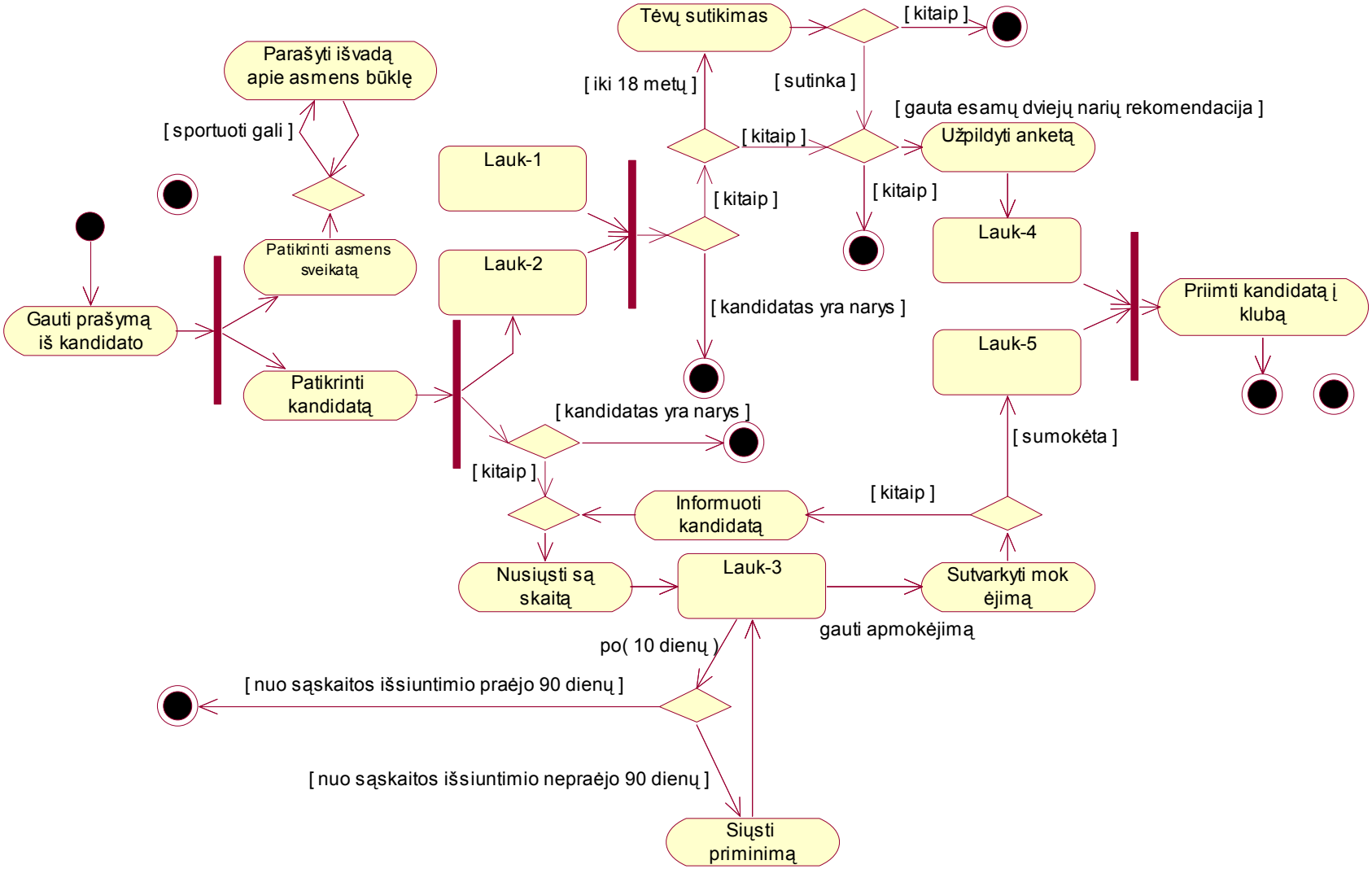
Sudaromas tikslų modelis. Nagrinėjami aktorių tarpusavio tikslai ir problemos, su kuriomis susiduria norėdami pasiekti savo tikslą. Tikslai bus nagrinėjami eilės tvarka.

Pagrindinė pradinė vartotojo siekiama situacija – tapti WT klubo nariu. WT klubo siekiama situacija yra WT kovos meno propagavimas plečiant savo veiklą. Plėtra tiesiogiai susijusi su klubo narių skaičiumi. Kuo WT klubo narių skaičius didesnis, tuo didesnės finansinės galimybės siekiant užsibrėžtų tikslų. Galutinius aktorių tikslus galima apibrėžti atitinkamai "Įgyta narystė" ir "Suteikta narystė". Veiklos objektas yra "Narystė". Vartotojo tikslas "Įgyti narystę" teigiamai įtakoja WT klubo tikslą „Suteikti narystę". Tai rodo, kad vartotojo tikslas leidžia pasiekti WT klubo.

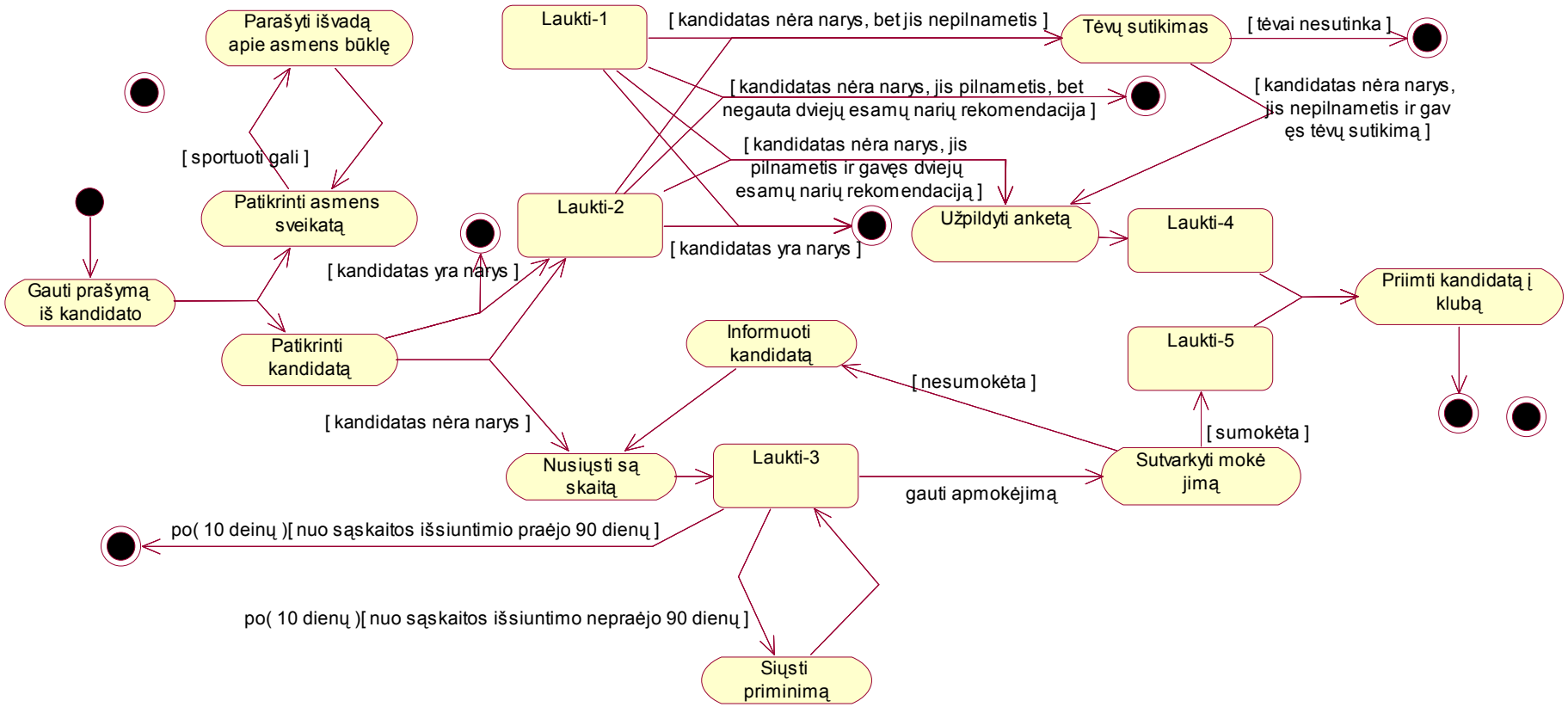
Toliau nagrinėjamos aktorių su tikslais susijusios problemos, su kuriomis susiduria aktorius ir kokios galimybės yra joms išspręsti. Situacija "Reikalinga informacija" WT vartotojui yra problema, o WT klubui – galimybė gauti abonementinį nario mokestį ir suteikti reikiamą informaciją. Ši situacija gali iššaukti WT klubo problemą "Trūkstamas kiekis". Tokio pobūdžio problema atsiranda tuomet – kai turimos informacijos nepakanka patenkinti WT vartotojo norus. Dėl šios priežasties WT klubui iškyla panaši situacija, kaip ir WT vartotojui „Reikalinga informacija", o BWTA – galimybė gauti nario mokestį iš WT klubo ir nuolat teikti reikiamą informaciją. BWTA informacija apie Wing Tsun kungfu kovos meną yra maksimali.

3.2 EKSMPERIMENTINIS VEIKLOS PROCESO MODELIS

30 paveiksle yra pavaizduota veiklos diagrama, kuri aprašo naujo nario priėmimo į klubą procesą. Šio proceso hipergrafas, gautas taikant anksčiau minėtas taisykles, pavaizduotas *31 paveiksle*.

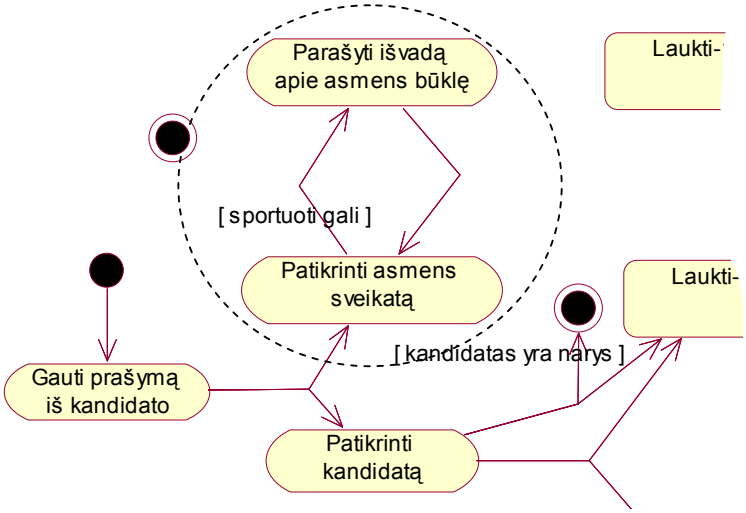


9 pav. Naujo nario priėmimo į klubą veiklos diagrama



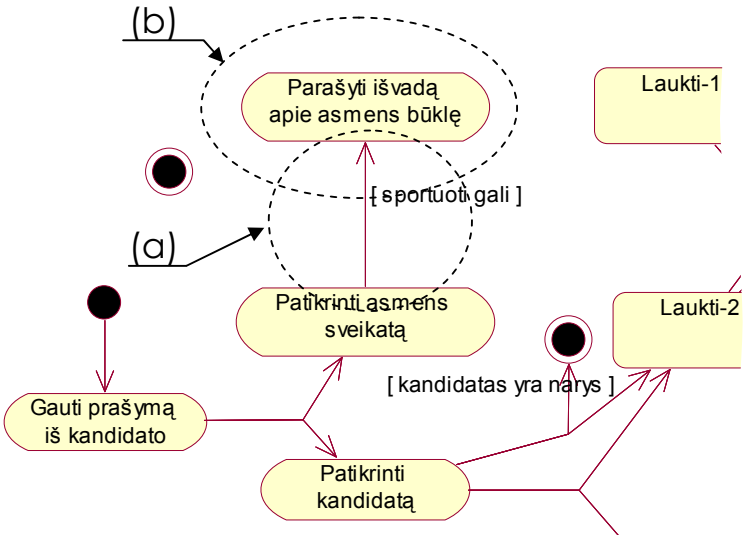
10 pav. Veiklos hipergrafas, sudarytas iš veiklos diagramos, pavaizduotos 30 paveiksle

Gautą hipergrafą dabar galime patikrinti. Struktūros tikrinimo metu yra nustatyta, jog iš viršūnės *„Patikrinti asmens sveikatą“* yra tik vienas kelias – pereiti prie viršūnės *„Parašyti išvadą apie asmens būklę“*. Iš šios viršūnės išėjimo hiperlankas nukreipia į prieš tai buvusią viršūnę *„Patikrinti asmens sveikatą“*, tad konstatuojamas ciklas (*32 pav.*).



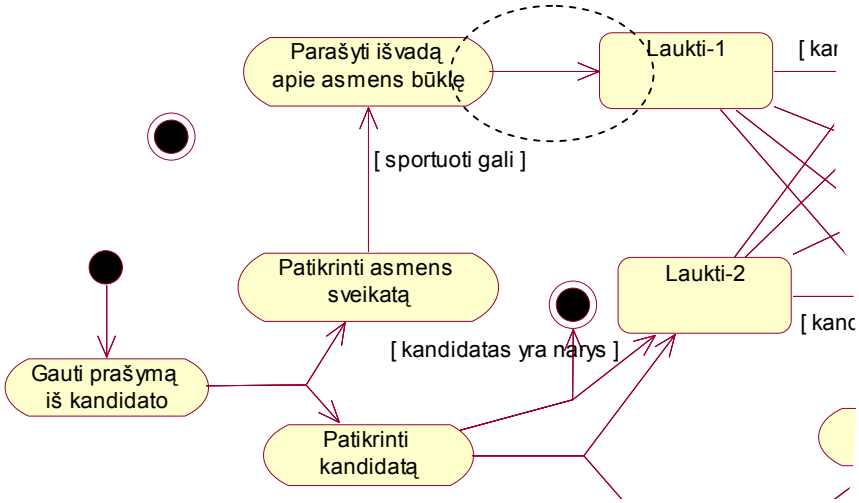
32 pav. Atlikus hipergrafo struktūros (31 pav.) tikrinimą, nustatytas ciklas (apibrauktas punktyru)

Atliekama hipergrafo korekcija ir sėkmingai eliminuojamas ciklas pašalinant nereikalingą hiperlanką (*33 pav. (a)*). Vėliau tęsiame struktūros tikrinimą ir aptinkame, jog po ciklo eliminavimo perėjus į viršūnę *„Parašyti išvadą apie asmens būklę“* – iš jos nėra galimybės pereiti į jokią kitą viršūnę. Nustatoma aklavietė (*33 pav. (b)*).



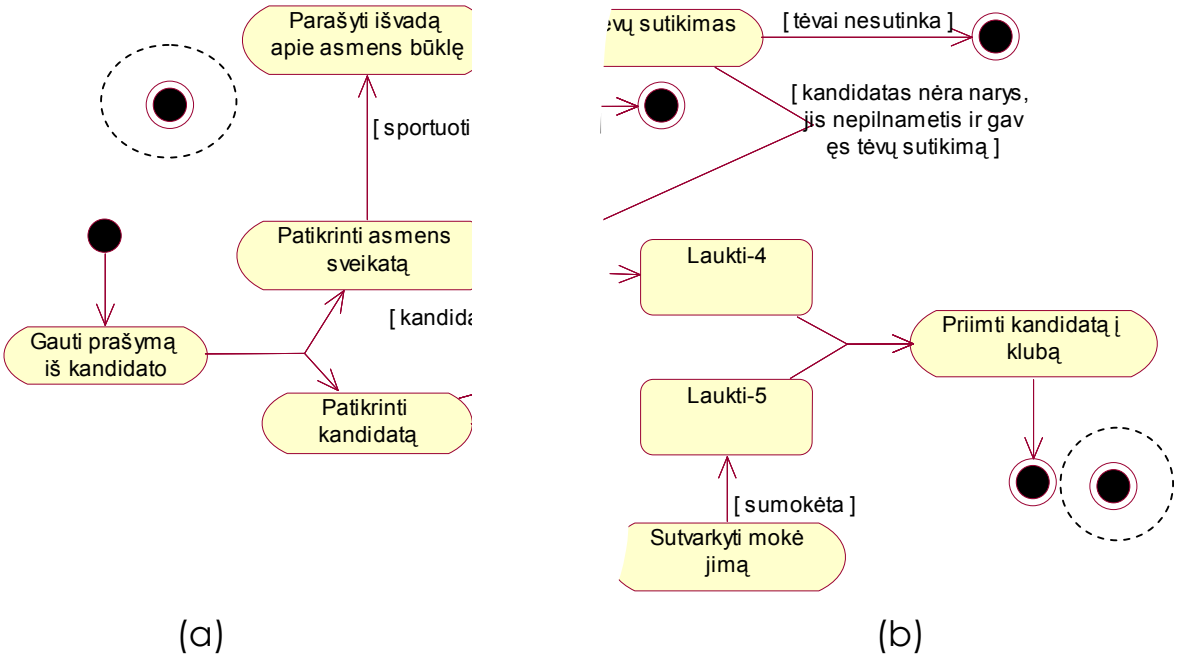
33 pav. Eliminavus nustatytą ciklą hipergrafe – nustatoma aklavietė (b)

Kad rasta aklavietė būtų eliminuota ir hipergrafas būtų korektiškas – reikia sujungti viršūnę *„Parašyti išvadą apie asmens būklę“* su viršūne *“Laukti-1“* taip, kaip pavaizduota (*34 pav.*).



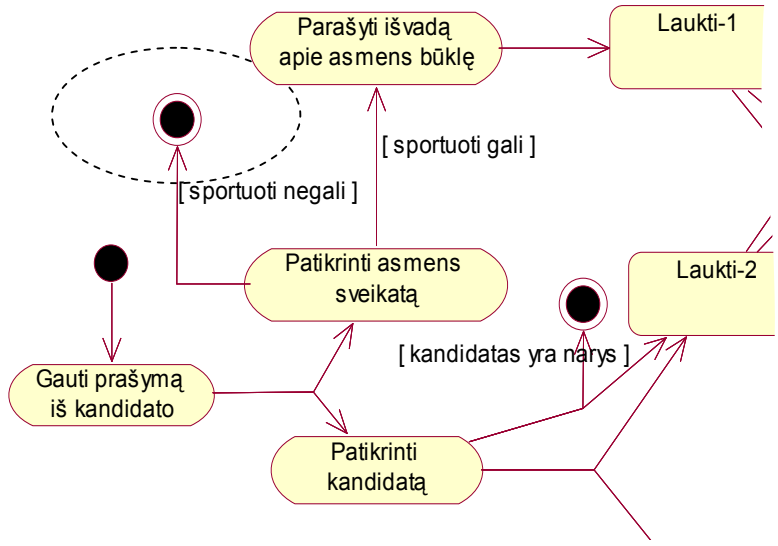
34 pav. Sujungus viršūnes tarpusavyje – eliminuojama aklavietė

Po šios korekcijos reikia vėl tęsti hipergrafo struktūros tikrinimą. Šį kartą yra sėkmingai pasiekta galinė būsena, tačiau yra nustatomos dar dvi klaidos – nepasiektos („kabančios“) viršūnės (*35 pav.*).



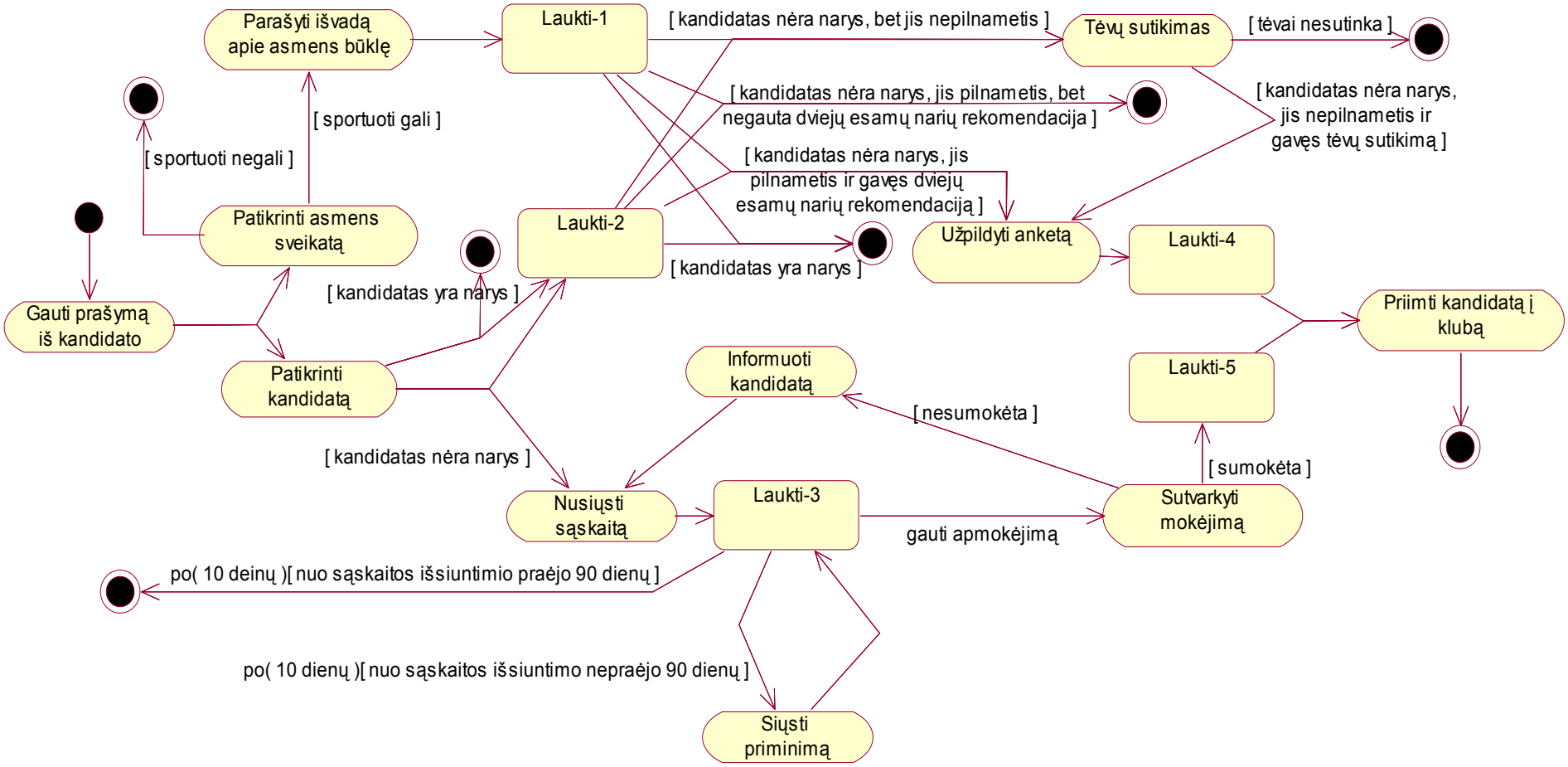
35 pav. Nustatytos nepasiektos viršūnės (apibrauktos punktyru)

Norint, kad veiklos hipergrafas būtų korektiškas, reikia atlikti jo korekciją. Vieną viršūnę (*35 pav. (a)*) reikia sujungti su viršūne „*Patikrinti asmens sveikatą“*, kadangi situacija, kuomet sportininkas yra nesveikas ir negali sportuoti – nėra numatyta (*36 pav.*). Antroji nustatyta „kabanti“ viršūnė (*35 pav. (b)*) yra nereikalinga, todėl ją reiktų paprasčiausiai eliminuoti (panaikinti).



36 pav. Atliktas „kabančios“ viršūnės (35 pav. (a)) prijungimas prie hipergrafo

Atlikus visas struktūros korekcijas yra gaunamas struktūriškai teisingas hipergrafas (*37 pav.*).



11 pav. Veiklos hipergrafas, sudarytas iš tikrinamo hipergrafo (31 pav.), atlikus struktūros korekciją

Dabar, kai nėra „kabančių“ viršūnių ir hipergrafo struktūros tikrinimas sėkmingai pavyko – galima pereiti prie dinaminio tikrinimo.

Sistemos (veiklos hipergrafo) apėjimas pradedamas nuo pradinės būsenos S_0. Veikla sustoja, kuomet pasiekia vieną iš trijų laukimo būsenų (*„Laukti-1“*, *„Laukti-2“* arba *„Laukti-3“*). Iš būsenos *„Laukti-1“* niekur negalim eiti, kol procesas nepereina į būseną *„Laukti- 2“*, ir atvirkščiai, iš būsenos *„Laukti-2“* niekur negalim eiti, kol procesas nepereina į būseną *„Laukti- 1“*. Iš būsenos *„Laukti-3“* niekur negalim eiti tol, kol nebus gautas *„kandidato“* apmokėjimas, arba kol nepraeis 90 dienų po sąskaitos išsiuntimo, arba kol neįvyks išorinis įvykis *„gauti apmokėjimą“*. Vėliau, iš laukimo būsenų *„Laukti-1“* bei *„Laukti-2“* yra pereinama į laukimo būseną *„Laukti-4“*, iš kurios niekur negalima pereiti tol, kol procesas nepereina į būseną *„Laukti-5“*. Čia yra fiksuojama stabili būsena, nes negalima pereiti prie kito viršūnių rinkinio. Tad imamas naujas išorinis įvykis *„gauti apmokėjimą“*. Gavus apmokėjimą iš *„kandidato“,* procesas pereina į būseną *„Laukti-5“*. Esant aktyviom abiem laukimo būsenom – procesas vienu žingsniu pereina į pabaigos viršūnę (pasiekta galinė proceso būsena S_{final}). Kadangi visi išoriniai įvykiai yra pasibaigę ir būsena $S=S_{final}$ – identifikuojame sėkmingą vykdymą.

IŠVADOS

1. Šiame darbe buvo siekiama patikslinti veiklos diagramos semantiką, kuri leistų teisingai atvaizduoti veiklos procesus ir atlikti modelio patikrą.

2. Atlikta UML veiklos grafų, Petri tinklų bei hipergrafų analizė. Petri tinklų semantika nėra pakankama verslo procesų modeliavimui, nes nėra galimybės pavaizduoti įėjimų įvykių, duomenų bei veiksmai turi būti vykdomi perėjimų metu, kas neatitinka B2B procesų semantikos, kai veiksmus turi vykdyti verslo dalyviai. Tiek hipergrafai, tiek Petri tinklai formalizuoja veiklos procesą, tačiau daro jį mažiau intuityviai suprantamą, kas irgi svarbu aprašant verslo procesus, ypač kai aprašymą daro verslo analitikai.

3. UML veiklos grafų tikrinimui pasirinkti hipergrafai, kurie sudaromi iš veiklos grafų. Hipergrafas leidžia supaprastinti veiklos diagramų struktūrą, nes išvengiama sprendimo taškų ir sinchronizavimo juostų. Hipergrafams galima taikyti panašias taisykles, kaip ir grafams: rasti aklavietes, nepasiekiamas viršūnes, ciklus.

4. Šiame darbe siūlomas hipergrafo dinaminio tikrinimo metamodelis ir konceptualus algoritmas, taip pat variantų skaičiaus sumažinimo taisyklės. Naudojant tokią metodiką, veiklos diagramos tikrinimas susidėtų iš jos transformavimo į hipergrafą, pradinio struktūrinio patikrinimo ir gana sudėtingo dinaminio patikrinimo.

5. Veiklos procesų apibrėžimo patikrinimo automatizavimas labai svarbus, kuriant elektroninio verslo ar darbų srautų valdymo sistemas, kadangi pastaruoju metu plinta metodai, leidžiantys generuoti tokio tipo programinę įrangą iš veiklos procesų apibrėžimų. To reikalauja dažni verslo sistemų pokyčiai. Nekorektiškas verslo proceso apibrėžimas gali niekais paversti visus informacinių technologijų laimėjimus verslo automatizavimo srityje.

6. Atliktas eksperimentinis algoritmo tyrimas rodo, kad tokį tikrinimą būtų tikslinga įgyvendinti UML CASE įrankiuose. Verslo analitikas sudarytų modelį UML diagramos pavidale, o pavertimas į hipergrafą ir tikrinimas būtų vykdomas automatiškai.

7. Greta tradicinių verslo procesų srautų modelių atsiranda ir alternatyvūs modeliai, padedantys užtikrinti korektišką procesų struktūrą. Tačiau sudėtingos verslo logikos išsiaiškinimas ir verifikavimas išlieka reikalingas. Šioje srityje dar daug klausimų

reikalauja išsamaus tyrimo, pavyzdžiui, laiko apribojimų nustatymas, kuris turi didžiulę įtaką verslo proceso apibrėžimo korektiškumui ir paties verslo efektyvumui.

TERMINŲ IR SANTRUMPŲ ŽODYNAS

- **DB** – Duomenų bazė
- **WFS (Workflow System)** – DSVS (Darbų srautų valdymo sistemos)
- **AH (Activity Hypergraphs)** – Veiklos hipergrafai
- **OO (Object-Oriented)** – Objektinės orientacijos
- **CASE (Computer-Aided Software Engineering)** – Automatizuoto projektavimo įrankis
- **UML (Unified Modeling Language)** – Modeliavimo kalba, skirta aprašyti sistemai, duomenų srautams ir pan. Plačiai naudojama populiariose *CASE* priemonėse
- **B2B (Business to Business)**

LITERATŪRA

[1] Booch G., Rumbaugh J., Jacobson I. The Unified Modelling Language User Guide. Addison Wesley, 2000.

[2] Eshuis R., Wieringa R. A Formal Semantics for UML Activity Diagrams – Formalising Workflow Models. University of Twente, Department of Computer Science, Netherlands.

[3] Eshuis R., Wieringa R. An Execution Algorithm for UML Activity Graphs. University of Twente, Department of Computer Science, Netherlands.

[4] Eshuis R., Wieringa R. Verification Support for Workflow Design with UML Activity Graphs. University of Twente, Department of Computer Science, Netherlands.

[5] Martin J., Odell J. Object-Oriented Methods: A Foundation. Prentice Hall, 1995.

[6] Martin J., Odell J. Object-Oriented Methods: Pragmatic Considerations. Prentice Hall, 1996.

[7] Marshall Ch. Enterprise Modelling with UML. Addison Wesley, 1999.

[8] UML Revision Taskforce. OMG UML Specification V. 1.3. Object Management, Group, 1999.

[9] OMG Unified Modeling Language. Version 1.4. Object Management Group, 2001.

[10] Weigand H., Heuvel W. J., Dignum F. Modelling Electronic Commerce Transactions. LAP'98, 1998.

[11] Peterson J. L., Petri Net Theory and the Modelling of Systems, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1981, p. 290.

[12] Van der Aalst W. M. P. The application of Petri nets to workflow management. The Journal of Circuits, Systems and Computers, 8(1):21-66, 1998.

[13] Ellis C. A., Nutt. G. J. Modelling and Enactment of Workflow Systems. In M. Ajmone Marsan, editor, Application and Theory of Petri Nets 1993, volume 691 of Lecture Notes in Computer Science, p. 1-16. Springer-Verlag, Berlin, 1993.

[14] Van der Aalst W. M. P. Interval Timed Coloured Petri Nets and theri Analysis. In M. Ajmone Marsan, editor, Application and Theory of Petri Nets 1993, volume 691 of Lecture Notes in Computer Science, p. 453-472. Springer-Verlag, Berlin, 1993.

[15] Gudonavičius, L. UML Veiklos diagramų verifikavimas naudojant hipergrafus// Informacinė visuomenė ir universitetinės studijos – 9: 9-sios magistrantų ir doktorantų konferencijos pranešimų medžiaga [Kaunas, 2004 m. balandžio 15 d.]. Kaunas, 2004, p. 193-199.

PRIEDAI