KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Inga Blažytė

VEIKLOS TAISYKLIŲ MODELIAVIMO UML KALBOS PAGRINDU TYRIMAS

Magistro darbas

Darbo vadovas
Doc. R. Butleris

Kaunas
2005

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

VEIKLOS TAISYKLIŲ MODELIAVIMO UML KALBOS PAGRINDU TYRIMAS

Informacinių sistemų inžinerijos magistro baigiamasis darbas

Darbo vadovas
Doc. R. Butleris
2005 01 11

Recenzentas
doc. V. Kiauleikis
2005 01 11

Atliko
IFM 9/4 gr. stud.
I. Blažytė
2005 01 11

Kaunas
2005

TURINYS

PRATARMĖ

Tikslai. Šio darbo tikslai yra tirti veiklos taisykles, jų sandarą ir klasifikaciją. Kadangi kiekvieną veiklą riboja tam tikros taisyklės ir jos yra keičiasi dinamiškai, tai svarbu tirti ar jas galima modeliuoti remiantis objektine metodologija. Taip pat siekiama analizuoti, kokia struktūra galima būtų patalpinti veiklos taisykles duomenų bazėje, kad jos būtų suprantamos ne tik profesionaliems sistemų projektuotojams, bet ir sukurtų sistemų vartotojams.

Išpopuliarėjus objektiškai orientuotam požiūriui buvo sukurta daug informacinių sistemų projektavimo įrankių, kurie palaiko UML (Unified Modeling Language) ar OML (Open Modeling Language) metodologijas. Šiais įrankiais tiesioginės inžinerijos metodu galima sukurti ir duomenų bazes, naudojant duomenų modelį. Bet tik nedidelė dalis tokių įrankių modeliuoja veiklos taisykles, kaip galinčias paveikti veiklos sistemos elgesį. Dažniausiai įrankiuose jos modeliuojamos kaip apribojimai, kurie „įsiuvami“ į programos kodą ir paprastam vartotojui jos nematomos ir negalima operatyviai jų pakeisti.

Pasirinktas įrankis MagicDraw UML, kuriame veiklos taisykles galima aprašyti objektų apribojimo kalba, bet taip jos yra paslepiamos nuo sistemos vartotojo ir be specialisto pagalbos sistemos vartotojas jų negali pakeisti.

Darbo aktualumas. Pagrindinis sistemų analizės tikslas yra surinkti visą svarbią informaciją apie realų pasaulį (probleminę sritį) – tai pirmiausia duomenys apie procesus, jų struktūrą ir duomenų kaitą. Veiklos taisyklės apima visus tris paminėtus dalykus, todėl jas galima laikyti centrine sistemų analizės dalimi ir sistemų analizės procesą pritaikyti prie veiklos taisyklių.

Daugelis veiklos procesų vykdomi, atsižvelgiant į taisykles, kurios apsprendžia tam tikrus veiksmus arba riboja galimų veiksmų aibę. Veiklos taisyklės vaidina didelį vaidmenį veiklos objektų architektūroje ir komponentų projektavimo metoduose – naujoje objektiškai orientuotų metodų kryptyje. Vienas iš labiausiai paplitęs objektiškai orientuotų metodų yra UML.

IS sistemų projektavimas UML kalba yra lengvai suprantamas tiek vartotojui, tiek programuotojui. Bet UML kalboje ne visos diagramos yra skirtos biznio procesams modeliuoti. Todėl yra svarbu, jei įmanoma, praplėsti tuos modelius ir pritaikyti biznio procesų modeliavimui.

ĮVADAS

Šiuolaikiniame besikeičiančiame pasaulyje veiklos sėkmė priklauso nuo sugebėjimo reaguoti į rinkos pokyčius. Dauguma veiklos procesų yra automatizuoti, todėl reikalingos technologijos, įgalinančios informacines sistemas efektyviai pritaikyti prie kintančių veiklos poreikių. Šiuo metu veiklos pokyčiai yra gana dažni ir daro didelį poveikį informacinėms sistemoms. Pasikeitus organizacijos veiklos taisyklėms, informacinėje sistemoje reikia atlikti tokius veiksmus [11]:

- Surasti, kurioje sistemos vietoje yra aprašyta pasikeitusi veiklos taisyklė;
- Modifikuoti procedūrą, kurioje reikia realizuoti pakeitimus;
- Išanalizuoti galimas pakeitimo pasekmes likusiai programai;
- Atlikti pakeistos procedūros ar net visos sistemos kompiliavimą bei testavimą.

Šis procesas yra lėtas, varginantis bei neefektyvus. Todėl šiandien IS srityje jaučiamas didelis susidomėjimas veiklos taisyklių automatizavimu, kuris tampa vienu iš programinės įrangos lankstumo ir pakartotino panaudojimo šaltinių. Tai aktuali sritis, pereinant prie komponentinio IS kūrimo ir elektroninio verslo. Veiklos taisyklių požiūriu galima išskirti dvi informacinių sistemų kūrimo kryptis [11]:

- IS kūrimas, neišskiriant veiklos taisyklių kaip atskiro komponento;
- IS kūrimas, išskiriant veiklos taisyklių komponentus, kuriuos galima įjungti į IS architektūrą.

Egzistuoja daug modeliavimo kalbų ir technikų, kuriuos galima naudoti veiklos taisyklėms (VT) modeliuoti. Modeliuojant verslo sistemas VT požiūriu, siekiama sukurti formalų VT modelį, kad jį būtų galima panaudoti modeliuojant IS. Modeliuojant IS ir programų sistemas taisyklių pagrindu, siekiama to pačio tikslo – sukurti pakankamai formalų taisyklių modelį, kurį būtų galima panaudoti modeliuojant ir kuriant programų sistemas. Be to, taisyklių modelis turi būti kuriamas taip, kad modelyje būtų tik taisyklės, kurios nedalomos į smulkesnes neprarandant prasmės. Literatūroje taisykles siūloma modeliuoti jau egzistuojančiomis kalbomis ir jose naudojamomis diagramomis arba siūlomi egzistuojančių kalbų išplėtimai, pritaikyti taisyklėms modeliuoti. Kuriamos ir naujos VT modeliavimui specializuotos kalbos.

Šiuo metu populiariausia objektinio projektavimo kalba yra UML, kurios antroji versija yra labiau pritaikyta verslo procesams modeliuoti [7], todėl šiame darbe VT analizė remiasi UML metodologija.

Norint modeliuoti VT reikia išskirti jų struktūrą ir klasifikaciją. Žinomiausi šios srities tyrėjai yra Business Rules Group (BRG), kuri ir pasiūlė VT struktūrą ir klasifikaciją [2]. Paprasčiausias VT struktūrinimo metodas yra šablonai [18]. Taip sustrūktūrintas taisykles jau galima modeliuoti struktūriniais modeliais.

Kadangi šio darbo tikslas yra pritaikyti VT modeliuoti objektiškai, tai bus atlikta UML kalbos analizė. Vienas iš galimų VT modeliavimo metodų yra OCL (Obejct Constrain Language) kalba [7, 10], kuria galima aprašyti ir sudėtingas VT. Bet kaip teigiama [15, 16] šaltiniuose tokiu atveju veiklą patogiau aprašyti operacijų ir su jomis susijusių invariantų rinkiniu. Tai yra savotiška VT klasifikacija, kurią patogiau modeliuoti OCL kalba, kuri yra lengvai suprantama ir vidutiniam modeliuotojui. Pagal tokį modeliavimo būdą VT modelyje prisegama kaip pastaba arba visai paslepiama [17]. Tai nėra labai priimtina, todėl taip pat nagrinėjama UML praplėtimo galimybė. Šaltinyje [19] apžvelgta kaip būtų galima pridėti VT stereotipą, o jei leidžia ir įrankio galimybės, tai papildyti UML meta modelį VT klase [19, 23].

Atliekant VT analizę, taip pat svarbu yra nustatyti kaip tas taisykles ir kur būtų galima saugoti. Šaltinyje [13] apibrėžiama VT saugyklos sąvoka ir jos savybės. Rosas [21] pasiūlė VT palaikymo architektūrą, kurioje įvedama taisyklių procesoriaus sąvoka. Jis reikalingas taisyklių valdymui ir panaudojimui.

Analizės dalyje bus apžvelgta kas tai yra veiklos taisyklės, kokios charakteristikos joms yra taikomos, kaip atliekamas jų formalizavimas ir kokios yra išskiriamos jų klasifikacijos. Bus apžvelgtos VT modeliavimo galimybės UML ir OCL kalbomis. Remiantis analize bus pasiūlytas sprendimas VT modeliavimui UML kalba konkrečiam sistemų projektavimo įrankiui.

1. VEIKLOS TAISYKLIŲ ANALIZĖS METODŲ APŽVALGA

1.1. OBJEKTIŠKAI ORIENTUOTAS POŽIŪRIS

Analizei buvo pasirinkta objektinė taikomųjų informacinių sistemų (IS) kūrimo metodologija, paremta objektiniu realaus pasaulio objektų modeliavimu ir to modelio panaudojimu nuo programavimo kalbos nepriklausomam projektavimui.

Objektinio metodo skirtumai nuo funkcinės metodologijos [8].

Objektinis metodas skiriasi nuo funkcinės metodologijos (Yourdon ir DeMarco), kurioje pagrindinis dėmesys skiriamas sistemos funkcijų specifikavimui. Toks požiūris gali atrodyti pats tiesiausias kelias realizuoti tikslui, bet sukurta sistema gali būti nepatvari. Jei keičiasi poreikiai, sistema, pagrįsta funkcine dekompozicija, gali pareikalauti esminių pertvarkymų.

Objektinis požiūris pirmiausia sutelkia dėmesį probleminės srities objektų identifikavimui, po to sutelkia aplink juos funkcijas. Nors tai gali atrodyti netiesioginis kelias, objektiniai programiniai produktai geriau elgiasi, keičiantis reikalavimams, kadangi jis labiau pagrįstas pačios probleminės srities struktūra, negu besikeičiančiais atskirų uždavinių reikalavimais.

1990 metų pradžioje išsivystė Objektinis modeliavimas ir projektavimas. Žemiau (1 lentelė) pateikti pagrindiniai autoriai, kurie sukūrė OO požiūrį ir metodus.

1 lentelė. Pagrindiniai autoriai, kurių kūryba ir bendradarbiavimas sukūrė OO požiūrį ir jo metodus

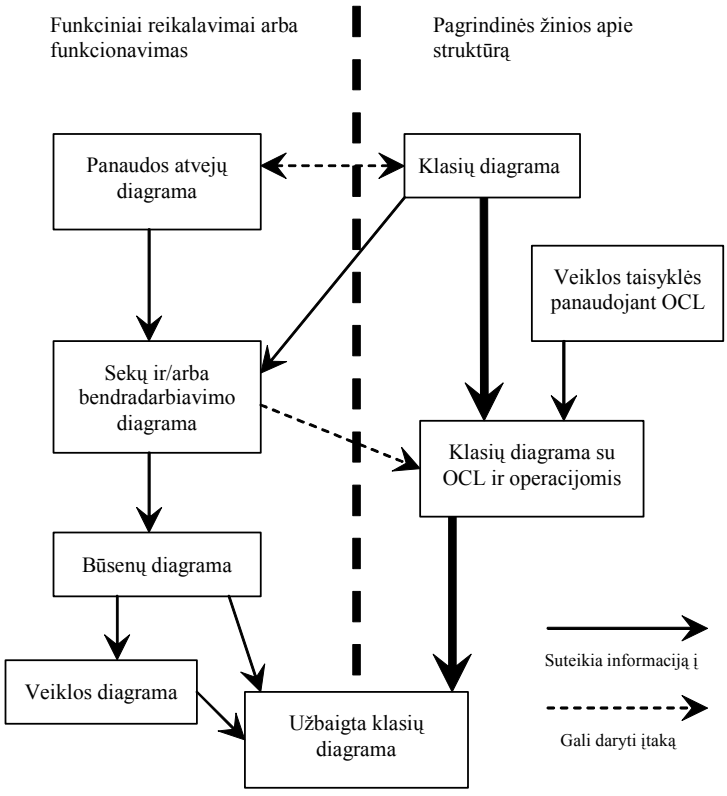
Grady Booch	Object oriented design with application, the Benjamin/Cummings Publ. Comp., 1991
Jim Rumbaugh	Object oriented modeling and design, Prentice Hall International, Inc. 1991, p. 490.
Ivar Jacobson	OOSE - Object oriented software engineering (Use case diagram)
P. Coad, E.Yourdan	Object oriented analysis, Yourdon press, 1991
Donald G. Firesmith	Object-Oriented requirements analysis and logical design. John Wiley & Sons, Inc. , 1993
Peter J.Robinson	Hierarchical Object-Oriented Design. Prentice Hall International (UK) Ltd., 1992. 238 p.
Sally Shlaer, Stephen J.Mellor	Object-Oriented Systems Analysis. Modeling the World in Data. Yourdon Press PTR Prentice Hall Inc., 1988. 144 p

Ian Graham	Object Oriented Methods. Second edition. Addison_Wesley Publishing Company Inc.,England, 1994. 473 p.
	Pagrindinės OO modeliavimo (analizės ir projektavimo) kalbos
UML	Unified modeling language, 1995
OML	Open modeling language

Veiklos taisyklių modeliavimui buvo pasirinkta UML kalba.

UML kalba universali koncepcinio modeliavimo ir objektinio projektavimo kalba, kuri nesiejama nei su viena kokia nors konkrečia objektinio projektavimo metodika. Iš tiesų tai rinkinys tarpusavyje susietų kalbų, kiekviena, iš kurių grindžiama kitu koncepcinio modeliavimo formalizmu. Kiekviena iš kalbų vartojama kito tipo diagramoms sudaryti [6].

Paveiksle (1 pav.) parodyti ryšiai tarp įvairių diagramų. Stora rodyklė viduryje rodo, kad klasių diagrama yra pagrindinis modelis sistemos vystyme. Paveikslas parodo skirtumus tarp sistemos struktūrinių ir elgsenos pusių. Struktūrinė pusė yra parodyta klasių diagramoje, kuri su kiekvienu žingsniu yra vis labiau detalizuojama. Elgsenos pusė yra parodyta kitose diagramose, išskyrus komponentų ir realizavimo (deployment) diagramas. Paveiksle abiejų pusių galai yra integruoti į klasių diagramą, kuri tiesiogiai susijusi su objektiškai orientuota programavimo kalba.



1 pav. Ryšiai tarp UML diagramų

UML atveria standartinius kelius aprašyti konceptualinius dalykus, biznio procesus ir sistemos funkciją.

UML yra kalba, bet nėra metodas, nes neapibrėžia standartinio proceso kaip reikia kurti informacijos sistemą. UML pagrindu yra siūloma daug metodų ir metodologijų.

UML technologijos privalumai:

- Supaprastėja komunikacija, visi kalba ta pačia kalba, iššvaistoma mažiau laiko.
- Reikalavimai lengviau apibrėžiami ir dokumentuojami, mažiaus „pamirštų“ vietų.
- Vartotojai įtraukiami į programos kūrimą nuo pat pradžių, mažiau perdarymų pabaigoje.
- Priemonė išsaugoti sukauptas žinias firmoje, net jei žmonės ją palieka.
- Sutaupo laiko susipažįstant su jau sukurtomis sistemomis.

1.2. OBJEKTŲ APRIBOJIMO KALBA

Objektų apribojimo kalba (Object Constraint Language) yra formali teksto specifikavimo kalba. Ji leidžia glaustą, tikslią apribojimų notaciją atskiruose UML modeliuose. OCL glaustumas yra tai, kad UML modelių elementai automatiškai priklauso sistemos tipui, kuria naudojasi OCL apribojimai. Todėl nėra būtina turėti savarankišką sistemos tipo apibrėžimą. OCL, kaip specifikavimo kalbą, yra labai lengva išmokti tiems, kurie yra susipažinę su programavimo kalbomis.

Objektų apribojimų kalba skirta biznio taisyklėms modeliuoti, patikslinti objektinius modelius. OCL savyje apjungia išraiškų kalba, modeliavimo kalbai ir formaliai kalbai būdingus bruožus, bei savybes [7].

Pridedant teksto dalis prie UML diagramų, galima sukurti pilną modeliuojamos sistemos specifikaciją. OCL užtikrina, kad projektuojamas modelis yra suderintas, tiksliai apibrėžtas ir pilnai atvaizduojamas programinės įrangos lygmenyje.

OCL buvo kuriama kaip dalykinio modeliavimo kalba IBM draudimo sektoriuje [2].

OCL vartotojai – tai programinės įrangos projektuotojai, naudojantys objektines technologijas.

OCL nėra programavimo kalba, dėl to neįmanoma aprašyti programos logikos ir kontrolės OCL priemonėmis. Neįmanoma aktyvuoti procesą ar operacijas per OCL. OCL pirmiausiai yra modeliavimo kalba, todėl ne viskas, kas OCL aprašyta, gali būti tiesiogiai atliekama.

OCL gali būti panaudota skirtingais tikslais:

- Apibrėžiant pastovias klases ir tipus klasių modelyje;
- Apibrėžiant pastovų tipą stereotipams;
- Aprašant procesų ir metodų būkles prieš ir po vykdymo;
- Apibrėžiant apsaugą;
- Aprašant procesų ryšius/apribojimus.

2. VEIKLOS TAISYKLIŲ MODELIAVIMO UML KALBA ANALIZĖ IR JOS REZULTATAI

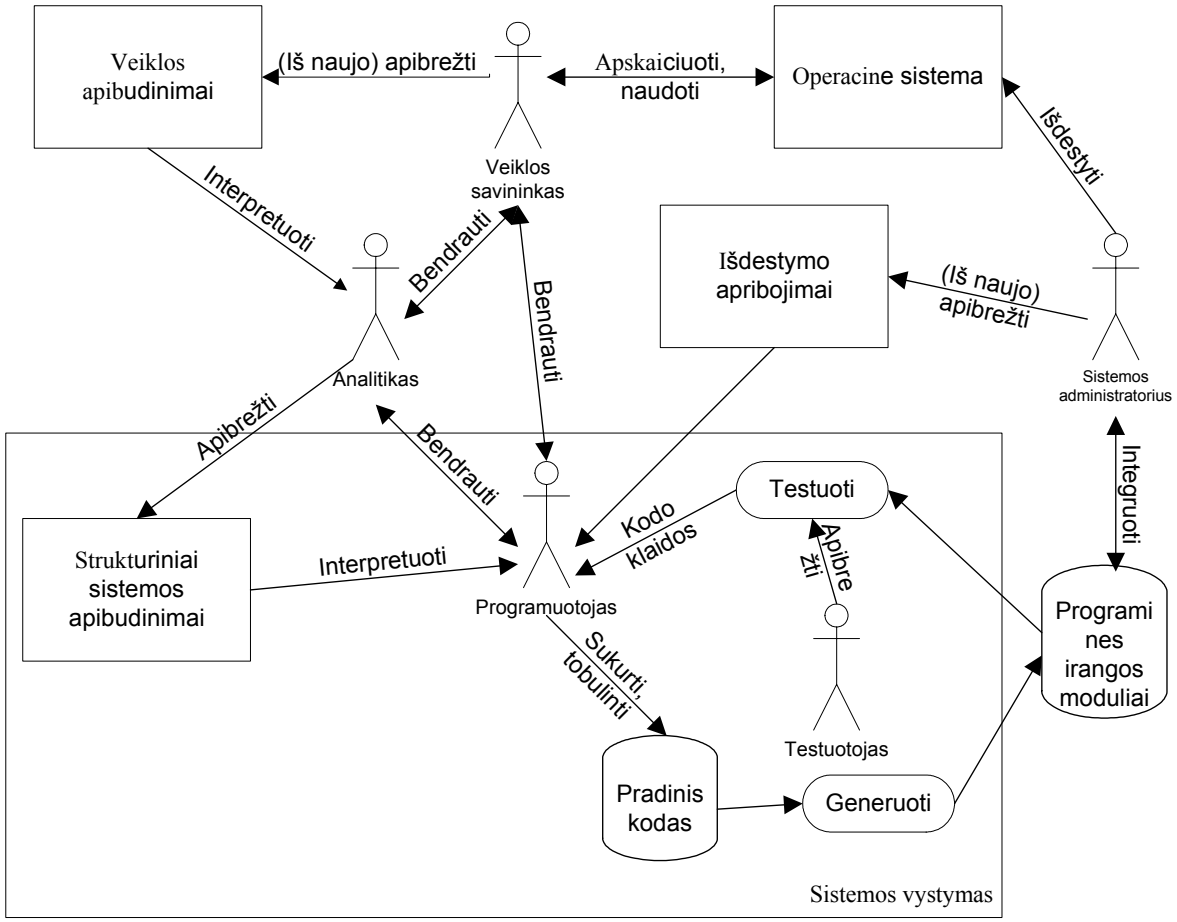
2.1. ESAMOS SITUACIJOS SRITYJE ANALIZĖ

Informacijos sistemų (IS) projektavimas naudojant vien tik grafinius duomenų atvaizdavimo modelius turi aiškių trūkumų [4]. Duomenų modeliai didžiąja dalimi tik aprašo duomenų struktūras. Čia beveik nieko nepasakoma kaip tie duomenys gali ar privalo būti panaudoti. Šiuos trūkumus bandoma išspręsti tradiciniais IS projektavimo metodais su veiklos taisyklėmis.

Pirmieji moksliniai straipsniai apie veiklos taisykles (VT) pradėti publikuoti apie 1990-uosius metus. Tuomet buvo pastebėta, kad, siekiant pilnai aprašyti organizaciją, tradicinių į procesus ir duomenis orientuotų analizės metodų nebeužtenka, o programuotojams paliekama per daug laisvės improvizacijai, programiniu kodu užpildant neišanalizuotus organizacijos aspektus.

Veiklos taisyklės dabar naudojamos tik labai nedaugelyje kompiuterizuoto IS projektavimo (CASE) įrankių. Šie įrankiai dažniausiai apsiriboja paprasčiausiomis struktūrinėmis taisyklėmis duomenų modelių struktūroms aprašyti (tokios taisyklės pateikiamos kaip grafinės modelių notacijos dalis). Jei naudojamos ir kitų tipų veiklos taisyklės (VT), jos paprastai saugomos tik kaip paprastas tekstas modelio objektų komentaro laukeliuose.

Norint suprasti veiklos taisyklių svarbą reikia sužinoti iš ko sudaryta sistema. Paveiksle (2 pav.) parodytas sistemos vystymo eskizas, tokios struktūros, kuri įgyvendinta daug kur šiais laikais.



2 pav. Sistemos vystymas, esant VT

2.2. LITERATŪROS ŠALTINIUOSE PATEIKTŲ VEIKLOS TAISYKLIŲ APŽVALGOS LYGINAMOJI ANALIZĖ

2.2.1. VEIKLOS TAISYKLĖS

Veiklos taisyklė (VT) yra teiginys, kuris apibrėžia ar apriboja tam tikrus veiklos aspektus. Veiklos taisykle nusakoma tam tikros veiklos struktūra, kontroliuojama arba įtakojama tam tikra veikla.

Kai veiklos taisyklė nusakoma veiklos žmogaus, tai ji dažniausiai įvardijama dviprasmiškai ir negriežtai. Dažnai veiklos taisykles galima išskaidyti į detalesnius veiklos taisyklės atvejus. Išskaidant veiklos taisykles iki savo elementariausios formos, jos dalomos tol, kol apsiribojama viena išbaigta mintimi. Veiklos taisyklėmis nemodeliuojama sukurtos programinės įrangos sistemos valdymo logika, tipiškai randama programos logikoje. Elementari veiklos taisyklė, parašyta deklaratyviu būdu, naudojant standartinę natūralios kalbos gramatiką, kurią veiklos žmonės lengvai supranta, nėra dviprasmiška.

Apskritai kalbant veiklos taisyklės yra apribojimai: jos apibrėžia sąlygas, kurios turi būti išpildytos specifikuotuose situacijose. Kartais vadinamos invariantais, veiklos taisyklės nėra procesų ar apribojimų apibrėžimai. Greičiau jie apibrėžia sąlygas prie kurių procesai yra vykdomi ar naujas sąlygas, kurios išliks po to, kai procesas bus įvykdytas.

Kitaip sakant, veiklos taisyklės nusako „kas“ turi būti faktai labiau nei „kaip“ tai turi įvykti. Taisyklių aibė, kuri nusako prieš- ir po- sąlygas, gali elgtis kaip proceso specifikacija, kurioje nenusakomas mechanizmas, kurio pagalba prieš-sąlygos yra transformuojamos į po-sąlygas.

Veiklos taisyklių sakiniai veiklos modelyje apibrėžia norimos veiklos logiką. Jie apibūdina dalyko būseną, kurioje veikla nori egzistuoti. Jei sakiniai yra išreikšti kaip loginės funkcijos, tuomet veiklos taisyklės visada turi grąžinti teigiamą loginę reikšmę, kitaip jos bus nenaudingos ir prieštaraus logikos apibrėžimams. Iš loginės perspektyvos veiklos taisyklėse nėra išimčių, čia yra tik taisyklės. Veiklos taisyklių numatoma išimtis paprastai yra kita veiklos taisyklė.

Veiklos taisyklė yra nepriklausoma nuo projektavimo modelio ar techninės platformos. Taigi, veiklos taisyklės yra [9]:

- Deklaratyvios;
- Elementarios;
- Išreikštos natūralia kalba;
- Savarankiškos, nepriklausomos loginės struktūros;
- Orientuotos į veiklą, o ne į technologiją;
- Priklausančios veiklai, o ne technologijai.

2.2.2. VEIKLOS TAISYKLIŲ KLASIFIKAVIMAS

Nėra vieningos nuomonės įvairiais klausimais, susijusiais su taisyklių identifikavimu, atvaizdavimu, jų integravimu IS kūrimo procese, diegimu taikomosiose programose ir kt. Nėra vieningo veiklos taisyklių apibrėžimo, notacijos bei jų klasifikacijos.

Organizacija „Business Rule Group“ (dar žinoma kaip „GUIDE Business Rule Project“) pasiūlė tokį veiklos taisyklių apibrėžimą: „Veiklos taisyklė – tai teiginys, kuris apibrėžia arba sąlygoja tam tikrą veiklos aspektą“ [9]. Žemiau pateikta (2 lentelė) veiklos taisyklių klasifikacija iš įvairių šaltinių.

2 lentelė. Veiklos taisyklių klasifikacija [16]

Šaltinis	Taisyklių klasifikacija
Business Rules Group (Guide Business Rules Project)	Struktūrinės taisyklės (terminai, faktai), veiksmo taisyklės (integralumo apribojimai, sąlygos, autorizacija), išvestys (skaičiavimai, loginės išvados)
Ronald Ross, Database Research Group	Terminai, faktai (ryšių tipai, subtipai, atributai), taisyklės (skaičiavimai, atmetimo taisyklės, projekcijos)
Tom Romeo, IBM	Struktūrinės taisyklės (ryšiai, domenai, kardinalumas), elgesio taisyklės (pre-sąlygos, post-sąlygos, išvestys)
Margaret Thorpe, Tangram	Apibrėžimai, integralumo taisyklės, bendrieji deklaratyvūs apribojimai, procedūriniai apribojimai, išvestys
Barbara von Halle, Knowledge Partners	Apibrėžimai, faktai, apribojimai, išvestys, loginės išvados
Usoft Corporation	Apribojimai, išvestys, elgesio taisyklės, atvaizdavimo
Brightware	Veikla, strategija, darbų sekos, sprendimų euristika
Vision Software	Validavimo taisyklės, išvestys, ryšiai, sąlygos veiksmai

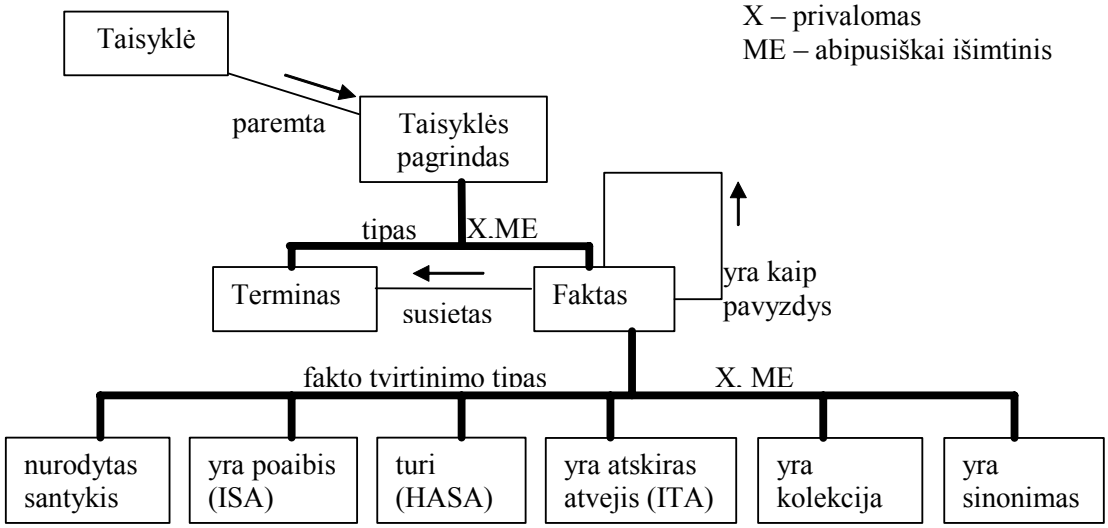
Kiekviena veiklos taisyklių srityje dirbanti organizacija, nepriklausomi mokslininkai kuria savas veiklos taisyklių klasifikacijas, priklausomai nuo iš keltų darbo tikslų, probleminės srities specifikos.

Viena dažniausiai naudojamų veiklos taisyklių klasifikacijų yra sudaryta organizacijos „Business Rules Group“ (BRG). Pagal šią klasifikaciją taisyklė gali:

- Apibrėžti terminą
- Sujungti terminus į faktus
- Pateikti skaičiavimų rezultatus
- Patikrinti sąlygas naujo fakto sukūrimui
- Patikrinti sąlygas veiksmui inicijuoti.

Trys paskutiniosios kategorijos apibrėžia tikrąsias veiklos taisykles, o terminai ir faktai turėtų būti interpretuojami kaip struktūrinės taisyklės, kurių didžiąją dalį galima atvaizduoti ir tradicinėmis duomenų modelius grafinėmis notacijomis.

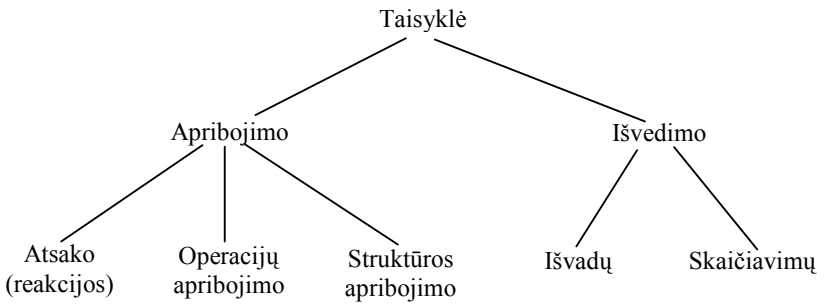
Paveiksle (3 pav.) parodoma veiklos taisyklių terminų ir faktų sąsaja [12].



3 pav. Veiklos taisyklės struktūra (terminai ir faktai)

Nepaisant sudėtingų veiklos taisyklių klasifikacijų, nemažą dalį jų galima aprašyti daugeliui suprantamais *JEI ...[IR ...] TUOMET ...*arba *KAI ...[IR] TUOMET ...* formatais. Tokiu formatu užrašytomis taisyklėmis galima įvertinti objektų savybes ir iššaukti atitinkamus veiksmus. Veiksmai gali modifikuoti veiklos objektų savybes, sukurti naujus veiklos objektus, iššaukti metodus arba kitų taisyklių vykdymą [1].

Norint integruoti veiklos taisykles Organizacijos veiklos modelyje tikslinga pateikti ir savo taisyklių klasifikaciją bei sudėtį. Žemiau pateikta klasifikacija (4 pav.) [14], atspindinti projektuotojo požiūrį į taisykles.



4 pav. Taisyklių klasifikacija projektuotojo požiūriu

Atsako (reakcijos) taisyklės (Stimulus and Responce Rules) apriboja elgesį, nusako sąlygas, prie kurių galima vykdyti operaciją. Pvz.: *KAI skaitytojas užsako bibliotekos knygą, JEIGU yra bibliotekos knygos kopija, TUOMET išduoti skaitytojui bibliotekos knygos kopiją.* Atsako taisyklės priklausomos nuo konteksto. Taisyklės JEIGU sąlyga tikrinama tik

tada, KAI įvyksta tam tikro tipo įvykis. Tokios taisyklės sutinkamos, pvz., darbų sekų ir būsenų perėjimo diagramose. Kartais jos vadinamos ECA (Event – Condition - Action) taisyklėmis [14, 17].

Operacijų apribojimo taisyklės (Operation Constraint Rules) nurodo, kokios sąlygos turi būti tenkinamos prieš operaciją ir po jos, kad būtų užtikrintas vykdymo teisingumas. Tokie apribojimai yra gyvybiškai svarbūs operacijos vykdymui ir nepriklauso nuo operacijos sužadinimo konteksto. Į šias taisykles galima žiūrėti kaip į kontraktą, siejantį operaciją su jos iškvietėjais. Tai galima užrašyti taip: „Jei iškviesi operaciją, kai įvykdyta prieš-sąlyga (precondition), gausi galutinę būseną, kurioje bus patenkinta po-sąlyga (postcondition)“ [17]. Operacijos prieš-sąlygos išreiškia apribojimus, prie kurių operacija atliekama teisingai [14]. Pvz.: *Pasiūlyti tarnautojui vadovo pareigas TIK TUOMET JEIGU tarnautojas nėra vadovas.* Operacijos po-sąlygos garantuoja rezultatus. Kai operacija įvykdyta, turi gautis tam tikra būsena. Pvz.: *Pasiūlyti tarnautojui vadovo pareigas TEISINGAI ATLIKTA TIK TUOMET JEIGU tarnautojas yra vadovas* [17].

Struktūros apribojimo taisyklės (Structure Constraint Rules) aprašo objektų ir jų sąryšių sąlygas, kurių negalima pažeisti. Taisyklė gali apriboti atributo reikšmes: *VISADA GALIOJA, KAD tarnautojo alga neturi būti didesnė už vadovo algą;* gali apriboti riboti objekto tipo egzempliorius: *VISADA GALIOJA, KAD teisėjų skaičius nebūtų didesnis už 5;*gali apriboti ryšio kardinalumą: *VISADA GALIOJA, KAD naujas vartotojas neturėtų daugiau kaip 7 užsakymus*. Struktūros apribojimo taisyklės nedaro nuorodų į operacijas, nes jos turi būti tenkinamos bet kuriomis veikimo aplinkybėmis, nesvarbu, kad pasikeitė objekto būsena [14, 17].

Išvadų taisyklės (Infences) aprašo, kokias išvadas galima padaryti iš tam tikrų faktų. Tokios taisyklės susiję su ekspertinėmis sistemomis. Pvz.: *JEIGU daugiakampis turi perimetrą, TUOMET kvadratas turi perimetrą* [14, 17].

Skaičiavimų taisyklės (Mathematical Computations) apibrėžia algoritmus, kuriuos vykdant galima gauti rezultatus. Jos išreiškiamos formulėmis. Pvz.: *Bendra produkto kaina APSKAIČIUOJAMA TAIP: produkto kaina*(1+mokesčių procentas/100)* [14, 17].

Skirtingos taisyklės yra modeliuojamos skirtingose diagramose. Vieno tipo taisyklė gali būti modeliuojama keliuose modeliuose, bet ne visi modeliai tinka biznio procesams modeliuoti. Žemiau pateikta (3 lentelė) lentelė, kurioje parodyta kokiais UML modeliais aukščiau apibrėžtos taisyklės yra modeliuojamos [12, 23].

3 lentelė.Veiklos taisyklių klasifikacija UML modeliuose

Veiklos taisyklės tipas	Išraiška	UML diagrama
Struktūrinės (Structural Assertions): • Terminai • Faktai	 *Yra, Turi, Privalo turėti, Negali turėti*	 Biznio Objektų diagramos Biznio Objektų diagramos
Apribojimo taisyklės (Constraint Rules): • Atsako (reakcijos) taisyklės • Struktūros apribojimo taisyklės • Operacijų apribojimo taisyklės	 *Kai/ Jeigu/ Tuomet* *Visada galioja kad* *Tik tuomet jeigu*	 Būsenų, Veiklos diagramos Biznio Objektų, Klasių diagramos Veiklos diagramos
Išvedimo taisyklės (Derivations): • Skaičiavimų taisyklės • Išvadų taisyklės	 *Apskaičiuojama taip* *Jeigu Tuomet*	 Veiklos diagramos, Klasių diagramos (metodai) Veiklos diagramos

2.2.3. VEIKLOS TAISYKLIŲ CHARAKTERISTIKOS

Veiklos analitiko darbas yra specifikuoti eilę aiškių sakinių apie esminę veiklos logiką. Aiškumo išryškinimas yra lemiamas. Veiklos taisyklės formuluotės turi būti tokios formos, kad veiklos savininkas galėtų tiesiogiai priimti kaip galiojančią ar atmesti kaip negaliojančią. Jei formuluotė primena techninius posakius, tai yra nesėkmės pripažinimas. Nėra geras bruožas skųstis, kad veiklos žmonės nesupranta predikatų skaičiavimo arba yra pakankamai nutolęs nuo technologinės perspektyvos. Faktas, kad jie jau vykdo biznį nenaudodami nieko daugiau nei paprastą šnekamąją kalbą yra tvirtas įrodymas, kad specialios notacijos, techninės kalbos ir t.t. nėra svarbūs.

Savo turiniu veiklos taisyklės gali pasirodyti neįspūdingos. Jos gali parodyti kažką iš veiklos kas atrodo banalu, akivaizdu ir kuo beveik neverta rūpintis. Taip ir turi būti. Jei viskas atrodo akivaizdu, tai išspręsta sudėtingiausia problema – padaryti veiklos logiką suprantamą.

Aukšto lygio veiklos taisyklės galėtų būti suklasifikuotos vienu ar daugiau požymiais:

- Sumažinant ar minimizuojant veiklos rizikos poveikį.

- Gerinant klientų aptarnavimą.
- Sudarant geriausią organizacijos išteklių panaudojimą.
- Kontroliuojant ar valdant darbo srautus.

Kaip ten bebūtų, visa tai žinant nebūtinai gali padėti identifikuojant pradines taisykles. Paskutiniame detalizavimo lygyje taisyklės yra dažniausiai susijusios su skirtingais veiklos aspektais. Lentelėje apibendrinti keletas labiausiai bendrų aspektų (4 lentelė) [22].

4 lentelė. Keletas taisyklių panaudojimo atvejų

Aspektas	**Pavyzdžiai**
Informacijos nuoseklumas	Datos/ terminai, adresų režimai ir taip toliau, kas turėtų būti dažnai svarstoma kokiais tai būdais organizacijoje.
Esybių ryšiai	Ryšiai turi būti vykdomi tarp esybių tiesiogiai įtakojant veiklą, galbūt esant tam tikroms sąlygoms. Pavyzdžiui: partija turi turėti bent vieną kontaktinį numerį.
Situacijų identifikacija	Bendrų veiklos situacijų atpažinimas leidžia atlikti normuotus, numatomus ir gerai valdomus atsakus.
Duomenų integralumas	Numatytos reikšmės, skaičiavimo algoritmai, reikšmių teisingumo kontrolė ir kitos duomenų teisingumo laidavimo priemonės.

Veiklos taisyklės turėtų nusakyti tik tai sąlygas, kurias reikia įvykdyti apibrėžtoje būsenoje. Papildoma informacija VT yra nepageidaujama ir ji turėtų būti paslėpta kažkur kitur, nes tokios informacijos dubliavimas pačioje taisyklėje didina klaidų tikimybę.

Konkreti veiklos taisyklė turi apibrėžti kas turi būti faktais ir ji neturi įsakinėti:

- Kas iškvietė taisyklę. (Tai dažniausiai apibrėžta panaudojimo atvejyje ar proceso apibrėžime.)
- Kada taisyklė yra vykdoma. (Tai dažniausiai apibrėžta veiklos įvykyje, panaudojimo atvejyje ar proceso apibrėžime.)
- Kur taisyklė vykdoma. (Tai bus apibrėžiama projektavime.)
- Kaip turi būti įvykdomos taisyklės. (Tai bus apibrėžiama projektavime.)

2.2.4. VEIKLOS TAISYKLIŲ IŠRAIŠKOS LYGIAI

Iki šiol buvo kalbėta tik apie taisykles kaip viena paprastą daiktą. Iš tikrųjų yra įmanoma įsivaizduoti mažiausiai tris taisyklės išraiškos lygius. Toliau pateikti pavyzdžiai

parodo kaip taisyklė gali atrodyti kiekviename lygyje (tai nereiškia, kad šie pavyzdžiai atitinka tam tikrą kalbą ar notaciją).

1. *Neformalus*. Tai šnekamosios kalbos teiginiai su ribotu diapazonu šablonų. Pavyzdžiui: Kreditinės sąskaitos užsakovas turi būti ne jaunesnis kaip 18 metų.

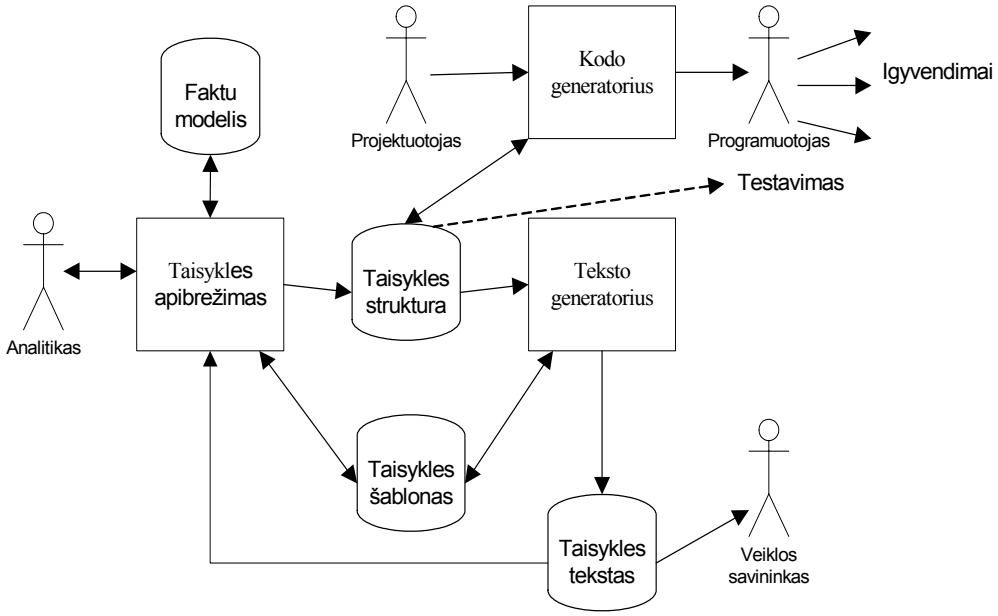
2. *Techninis*. Jis sujungia struktūrinių duomenų informaciją, operatorius ir apribotą natūralią kalbą. Pavyzdžiui:

KreditoSąskaita

Kliento.amžius >= 18

3. *Formalus*. Šio lygio sakiniai labiau atitinka sintaksę su matematinėmis savybėmis. Pavyzdžiui: {X, Y, (klientas X) (KreditoSąskaita Y) (Savininkas X Y)}➔ (ge (amžius X) 18)

Formalizuotų savybių struktūros ieškojimas yra labiausiai pageidaujamas, bet daugelis žmonių biznio lygyje būtų labiau laimingi su šnekamosiomis neformaliomis savybėmis. Vienas tokios problemos sprendimas yra per tinkamo įrankio palaikymą. Žemiau esančiame paveiksle (5 pav.) parodytas eskizas kaip būtų galima kurti formalių taisyklių struktūrą išlaikant lengvus šnekamosios kalbos teiginius. Šiuo atveju analitiko darbas yra manipuliuoti iš anksto apibrėžtų struktūrinių vienetų aibe. Įvairios struktūrinės formos gali būti naudojamos kaip bazė ekvivalenčių tekstinių vaizdų generavimui; jos nėra redaguojamos tiesiogiai. Šiuo atveju pateikta struktūra su kuria galima ne tik dirbti, bet ir įmanoma pagalvoti apie kodo generavimą [22].



5 pav. Kontroliuojamas taisyklės apibrėžimas

2.2.5. VEIKLOS TAISYKLIŲ FORMALIZAVIMAS

Pasak Barbaros von Halle, visame sistemos kūrimo procese veiklos taisyklės verslo žmonėms yra labiausiai suprantamas dalykas. Iš vartotojo surenkamos taisyklės turi būti aprašomos tokiu formatu, kad vėliau jas būtų galima dokumentuoti ir automatizuoti.

Taisyklės turi būti:

- deklaratyvios - taisyklės neturi būti apibrėžtos procedūriškai, naudojant aukšto lygio procedūrines programavimo kalbas. Šiuo atveju nėra svarbu, kaip taisyklė bus įgyvendinta;
- tikslios - taisyklė turi būti interpretuojama vienareikšmiškai. Jei taisyklė gali būti suprasta daugiareikšmiškai, ji turi būti suformuluota kitaip;
- atominės - taisyklė turi išlaikyti vieną ir tik vieną išbaigtą mintį. Bandant suskaldyti atominę taisyklę į keletą dalių bus prarandama informacija;
- neperteklinės - taisyklių rinkinys negali turėti taisyklių, teigiančių tą patį.
- VT taip pat gali būti kompozicinės, t.y. būti sudarytos iš kitų taisyklių.

Šiuos reikalavimus bandoma užtikrinti pseudo-formaliomis kalbomis, kuriose naudojamos natūralios kalbos išraiškos, sudėtos į tam tikros struktūros sakinius - šablonus. Veiklos taisyklės turi būti užrašytos verslo atstovams suprantama kalba, nes būtent šie žmonės naudosis veiklos taisyklių valdymo sistemomis. Be to:

- šablonais užrašytos veiklos taisyklės yra dalinai formalizuotos,
- naudojant šablonus yra išlaikoma vienoda to paties tipo taisyklių struktūra,
- taisyklių šablonai laikomi pseudo-kodu - tai svarbu programuotojams.

Šablonų naudojimas yra patogus būdas taisyklėms iš vartotojų surinkti ir validuoti, tačiau gilesnei jų analizei reikėtų naudoti deklaratyvias formalias kalbas, pavyzdžiui, Prolog, Z, Predikatų logiką, OCL, Alloy.

Šablonais užrašytos taisyklės transformuojamos į formalias pasirinktos kalbos išraiškas. Formaliam taisyklių užrašymui pasirinkta OCL (Object Constraint Language) [4] dėl tokių priežasčių:

- Informacijos sistemų projektavimo metu naudojami grafiniai UML (Unified Modeling Language) modeliai, o OCL yra UML standarto dalis. Pagrindinė OCL paskirtis yra formaliomis išraiškomis papildyti UML grafinius modelius (pvz., klasių modelį). OCL idealiai tinka struktūrinėms taisyklėms užrašyti, tačiau šia kalba galima užrašyti ir kitų tipų taisykles.

- Yra sukurta keletas OCL sintaksės tikrinimo įrankių bei interpretatorių (pvz., IBM OCL parser [10], Dresden OCL Toolkit [5]). Kai kurie įrankiai yra atviro kodo ir moksliniais tikslais juos galima modifikuoti.
- OCL yra gana lengvai įsisavinama ir suprantama kalba.

Šablonuose naudojami formalumai yra šie:

- Lenktiniai skliaustai ( ) aptveria terminų grupę.
- Laužtiniai skliaustai [ ] aptveria laisvai parenkamus terminus.
- Vertikalūs barjerai | skiria alternatyvius terminus.
- Kampuoti skliaustai < > aptveria specialius terminus, kurie apibrėžti lentelėje (5 lentelė).

Simbolių nesimato taisyklėje, jie yra tik šablono apibrėžime.

5 lentelė. Kintami šablonų elementai [22]

Elementas	**Reikšmė**
<det>	Determinantas objektui;
<objektas>	Atpažįstama veiklos esybė, tokio kaip matomos veiklos objektas faktų modelyje, rolės pavadinimas ar objekto savybė. Esybė gali būti patikslinta kitais vaizduojamaisiais elementais, tokiais kaip jo gyvavimas tam tikroje būsenoje, tam kad specifikuotų taisyklės pritaikomumą su pakankamu tikslumu.
<charakteristika>	Veiklos elgesys, kuris turi įvykti ar sąryšis, kuris turi būti sukurtas.
<faktas>	Sąryšis faktų modelyje tarp atpažįstamų faktų ir apibrėžtų apribojimų. Sąryšis gali būti patikslintas kitais vaizduojamaisiais elementais, tam kad specifikuotų taisyklės pritaikomumą su pakankamu tikslumu.
<faktų-sąrašas>	Sąrašas <faktas> elementų.
<m>, <n>	Skaitmeniniai parametrai.
<rezultatas>	Bet kokia reikšmė, nebūtinai skaitinė, kuri turi prasmę veiklai. Rezultatas nebūtinai turi būti biznio objekto atributo reikšmė.
<algoritmas>	Specialus metodo apibrėžimas, kuris turi būti panaudotas specialiai reikšmei įgyti. Paprastai išreikštas naudojant kombinaciją atpažįstamų terminų reikšmių kartu su naudingais apribojimais faktų modelyje.

5 lentelė. Kintami šablonų elementai [22] (tęsinys)

Elementas	**Reikšmė**
<klasifikacija>	Termino faktų modelyje apibrėžimas. Tai dažniausiai apibrėžia arba atributo reikšmę pavadintą „būseną“ arba objektų poaibį egzistuojančioje klasėje.
<išvardytas-sąrašas>	Išvardytų reikšmių sąrašas. *Open* išvardijimas nurodo, kad sąrašas gali būti modifikuojamas remiantis ateities reikalavimais; pavyzdžiui, statuso reikšmių sąrašas objektas kaip žinomas dabar. *Closed* išvardijimas nurodo, kad pakeitimai sąrašui yra nenumatyti; pavyzdžiui, savaitės dienos.

Žemiau (6 lentelė) pateikti keli VT šablonų pavyzdžiai [18].

6 lentelė. VT šablonų pavyzdžiai

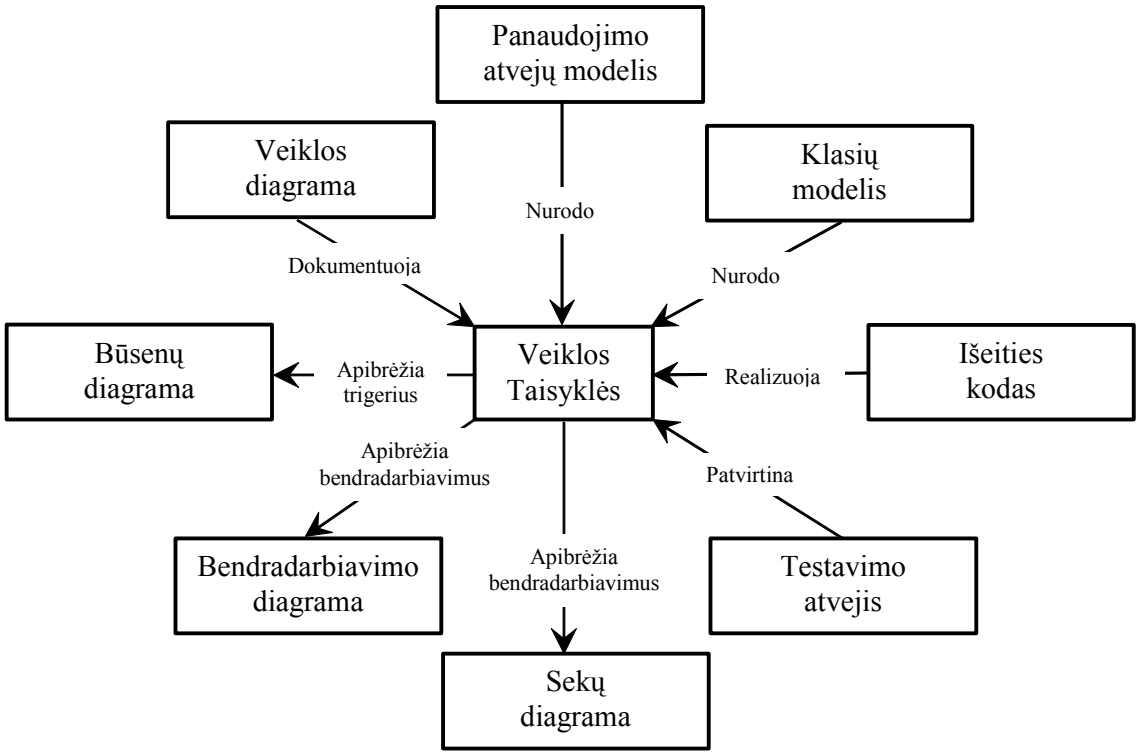
VT tipas	**Šablonas**
Terminas	<Term[1]> *yra apibrėžtas kaip* <tekstinis apibrėžimas>
Faktas	<Term_1> *gali būti* <*veiksmažodis*> <Term_2> <Term_1> *turi savybę* <Term_2>
Išvestis	<Term> *apskaičiuojams kaip* <formulė>
Apribojimas	<Term_2> *turi turėti* <mažiausiai, daugiausiai, tiksliai *n* iš> <Term_1> <Term_1> *turi* <lyginant su> {<Term_2>, Termino reikšmė, konstanta} <Term_1> *turi būti sąraše* <sąrašas>
Veiksmo įgalinimas	*JEI* <taisyklės sakinys> *[IR* <taisyklės sakinys> *IR* <taisyklės sakinys> *IR* <taisyklės sakinys>…]
Loginė išvada	*JEI* <taisyklės sakinys> *[IR* <taisyklės sakinys> *IR* <taisyklės sakinys> *IR* <taisyklės sakinys>…]

2.2.6. VEIKLOS TAISYKLIŲ MODELIAVIMAS UML KALBA

UML projektuotojams pateikia aibę notacijų ir semantikų informacinių sistemų statiniams, dinaminiams ir elgesio aspektams modeliuoti. Pavyzdžiui UML palaiko penkias notacijas sistemos dinamikai modeliuoti. Dvi iš jų yra veiklos ir būsenų diagramos. Naudojant veiklos diagramą galima modeliuoti darbų sekas, kurios pakeičia sudėtinių veiklos objektų, tokių kaip draudimo ieškiniai, banko sąskaitos, statusą. Naudojant būsenų

[1] Term = Terminas;

diagramą galima pavaizduoti kaip darbai sudėtiniame darbų sekų modelyje keičia atskirų atvejų, draudimo ieškinių ar bankų sąskaitų, būseną. Šios kelių dimensijų panaudojimo perspektyvos leidžia apibūdinti sistemą iš skirtingų požiūrių ir padeda atvaizduoti šiuolaikinių sistemų sudėtingas savybes [19]. Žemiau pateiktoje diagramoje parodoma sąsaja tarp VT ir UML modelių (6 pav.).



6 pav. Ryšiai tarp VT ir UML modelių

Negalima teigti, kad UML neleidžia modeliuoti VT. Yra daug būdų pavaizduoti VT UML kalboje [19, 23]:

1. Grafiškai, naudojant UML modelių semantiką - asociaciją ir apibendrinimą objektų modeliuose.

2. Grafiškai, naudojant praplėtimą, vadinamą apribojimu (constraint). Pavyzdžiui apribojimai leidžia tiksliau specifikuoti struktūrines veiklos taisykles.

3. Grafiškai, naudojant pastabą (note), susietą su kitu modelio elementu, pavyzdžiui, asociacija ar įvykiu.

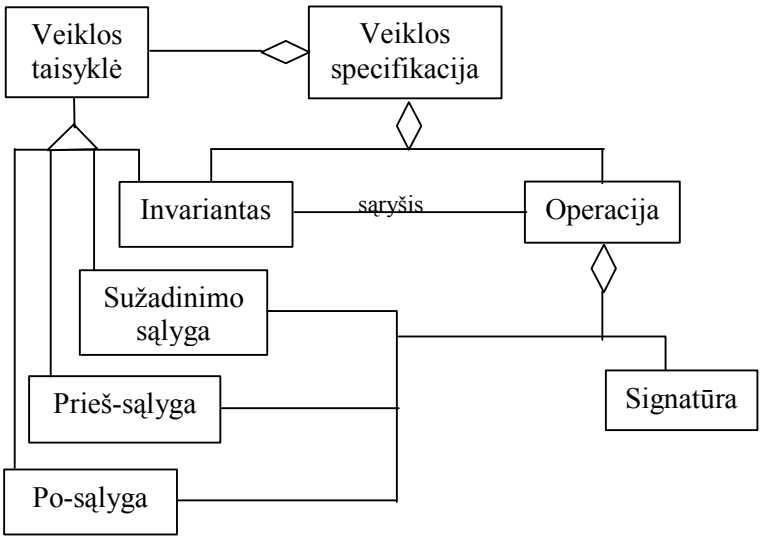
4. Naudojant predikatų logiką objektų apribojimo kalboje (OCL).

5. Apibrėžiant veiklos taisykles kaip kitų artefaktų savybes. Pavyzdžiui, apsaugos sąlygos perėjimuose būsenų modeliuose.

Problema yra ta, kad UML nespecifikuoja VT kaip atskiro elemento modelių kūrimo bloke – unikalios modeliavimo elemento klasės.

2.2.7. VEIKLOS TAISYKLIŲ MODELIAVIMAS OCL

Organizacijos veiklą galima pilnai specifikuoti taisyklėmis. Iš vienos pusės, veiklą galima specifikuoti kaip veiklos taisyklių rinkinį, iš kitos pusės, kaip operacijų ir su jomis susijusių invariantų rinkinį. Operacijos modelį sudaro jos signatūra, prieš-sąlygų, po-sąlygų ir, galbūt, kitokių taisyklių rinkinys (7 pav.) [15, 16].

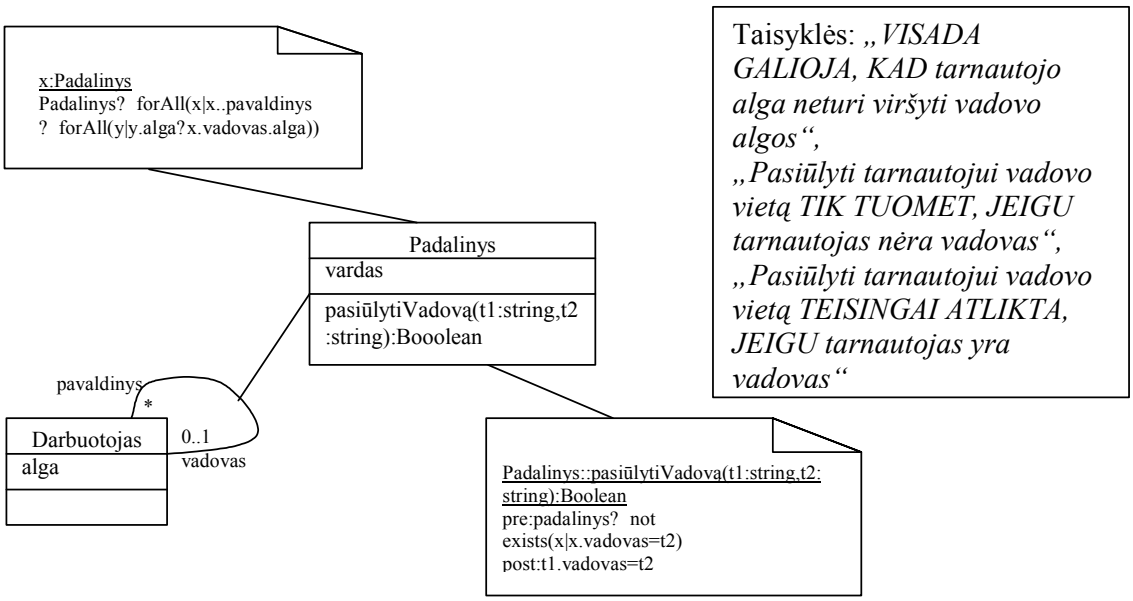


7 pav. Veikos specifikavimas taisyklėmis (UML notacija)

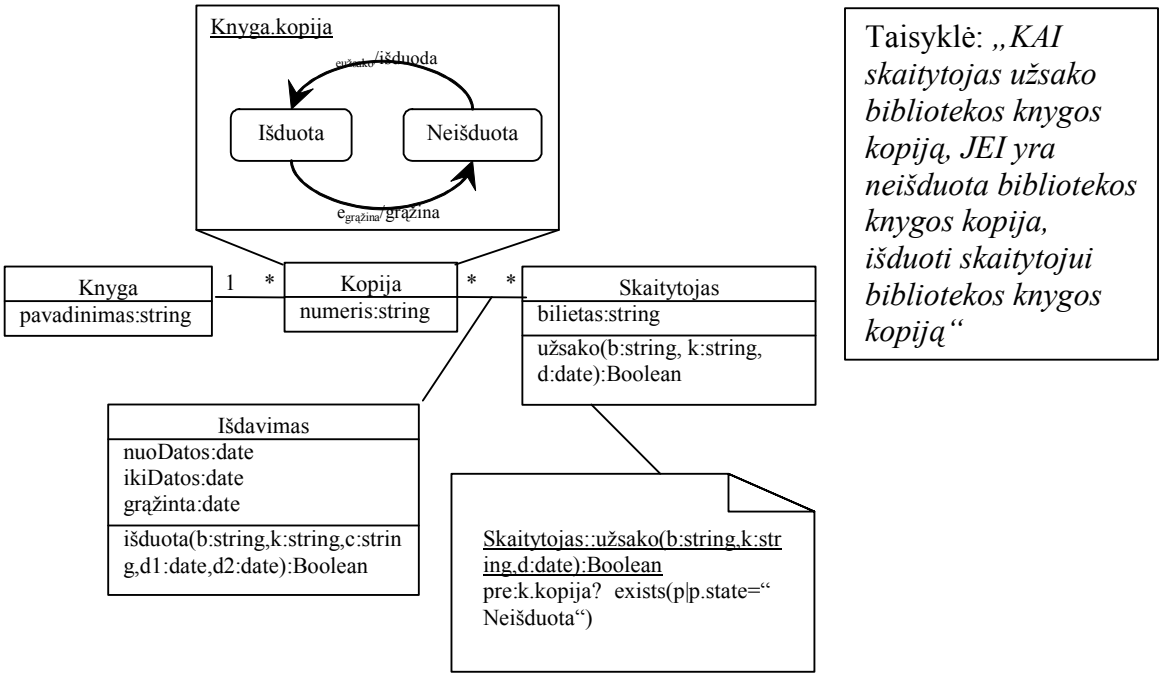
Skaičiavimo sąlygos veiklos modelyje neaprašomos, kadangi veiklos modelis turi atsakyti į klausimą „ką?“, nedetalizuojant „kaip“. Toks principas taikomas dėl to, kad veiklos modelis, kaip ir kompiuterizavimo projektas, turi būti nepriklausomas nuo realizacijos. Tačiau skaičiavimų ar išvedimo modelis gali atsirasti vėlesnėse, realizavimo stadijose. Pateiktąjį modelį galima atvaizduoti bet kuria formalia modeliavimo kalba.

UML taisyklės vaizduojamos kaip pastabos, pradžioje aprašomos natūralia kalba, po to verčiamos į OCL. Pastaba grafiškai vaizduojama specialia piktograma, kuri susiejama su kitais diagramų simboliais.

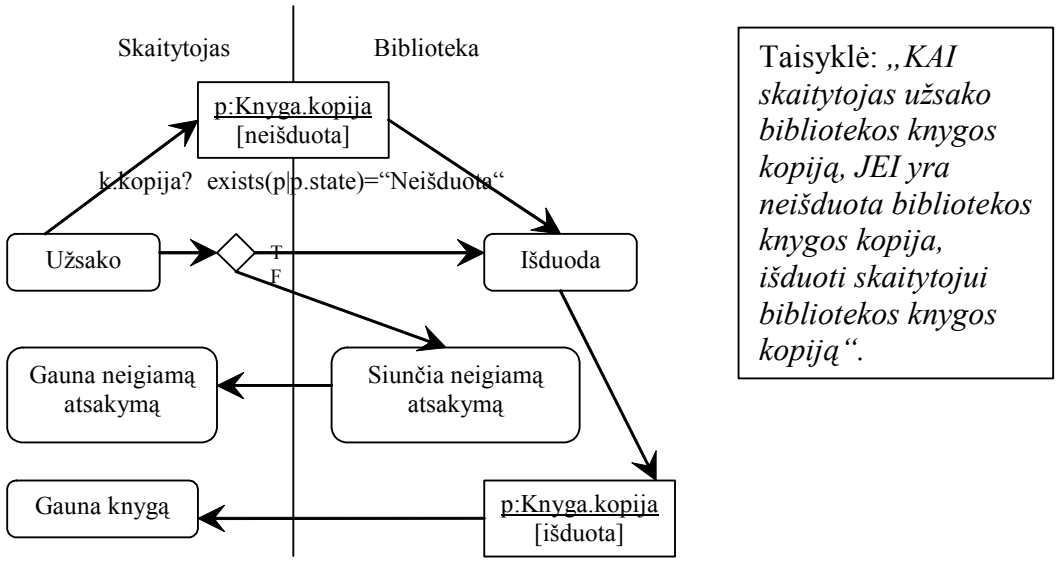
Modeliuojant veiklos taisykles, OCL galima panaudoti invariantų specifikavimui, prieš ir po sąlygų specifikavimui, navigavimui. UML apibrėžime OCL panaudota UML semantikos aprašymui. OCL papildo UML, įgalindama modeliuoti tiksliai. Be to, ji nėra labai sudėtinga, prieinama vidutiniam modeliuotojui. OCL išraiškomis galima papildyti klasių (8 pav.), būsenų (9 pav.), veiklos (10 pav.) ir kitas UML diagramas [17].



8 pav. UML klasių diagrama [17]



9 pav. UML klasių ir būsenų diagrama [17]



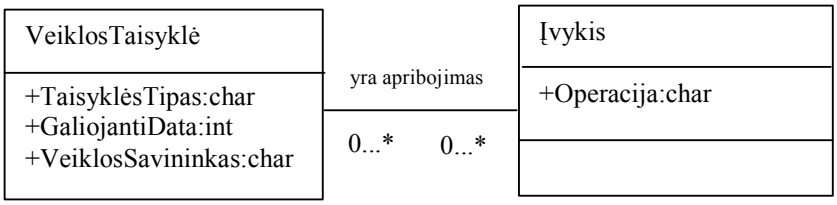
10 pav. UML veiklos diagrama [17]

2.2.8. UML MODIFIKAVIMAS VEIKLOS TAISYKLĖMS MODELIUOTI

UML nespecifikuoja VT kaip atskiro modelio elemento, todėl tai nėra visiškai priimtinas VT modeliavimo būdas. Kad UML notacija būtų tinkama VT modeliavimui ir valdymui, reikia įtraukti atskirą modeliavimo elementą, pavadintą *VeiklosTaisyklė*. Tuo tikslu reikia praplėsti UML meta modelį pridedant VT klasę su jos savybėmis. Tada reikia specifikuoti asociacijas su kitais modeliavimo elementais.

UML siūlo tris praplėtimo mechanizmus. Tai stereotipai, prijungtos reikšmės ir apribojimai.

Stereotipai leidžia sukurti naują UML modelio kūrimo bloką. Stereotipai yra suskirstyti į keturias kategorijas: procesus, šaltinius ir taisykles, tikslus bei įvairialypius. Tikslų kategorija taip pat turi nedidelę apribojimų aibę, skirtą tikslų hierarchijų modeliavimui. Žemiau yra pateiktas pavyzdys (11 pav.), kuriame yra parodyta, kaip galima pridėti naują metodo stereotipą, pavadintą *VeiklosTaisyklė* [19].



11 pav. Atskiras elementas VeiklosTaisyklė

UML apibrėžia ir įtraukia daug naudingų prijungtų reikšmių, bet Eriksson – Penker veiklos praplėtimai siūlo aibę naujų prijungtinių reikšmių apibūdinti veiklos procesui. Šios

reikšmės yra: tikslo, paskirties, dokumentavimo, proceso savininko, proceso autoriaus, prioriteto, rizikos, galimybės, laiko ir kainos [20].

Tolesniam šio proceso vykdymui reikia veiklos taisyklių saugyklos, kurioje yra saugomos ir apdorojamos veiklos taisyklės. Gerai sudaryta veiklos taisyklių saugykla leidžia praplėsti veiklos taisyklių modeliavimo galimybes UML'e.

2.2.9. VEIKLOS TAISYKLIŲ SAUGYKLA

Egzistuojantys informacinių sistemų specifikavimo ir projektavimo modeliai blogai išreiškia dalykinėje srityje galiojančias taisykles, jos yra padrikai paskirstytos po visą sistemą. Kadangi UML neužtikrina kelių projektuotojų sukurtų VT specifikacijų darnos, tai gali vesti prie prieštaravimo skirtinguose modeliuose ar sistemos specifikacijos fragmentuose. Norint išvengti tokių prieštaravimų reikalinga VT saugykla – duomenų bazė, kurioje užfiksuojamos, saugomos ir valdomos VT.

Taikant veiklos taisyklių nepriklausomumo sluoksnio koncepciją, galima išspręsti šią problemą, nes taisyklių identifikavimas ir automatizavimas yra išreiškiamas formaliais metodais, o pačios taisyklės saugomos atskirai nuo kitų sistemos komponentų.

Egzistuoja keletas veiklos taisyklėmis pagrįstų informacinių sistemų architektūros modelių, tačiau saugyklos vaidmuo visais atvejais yra panašus.

Veiklos taisyklių saugykla – tai autonomiška tam tikru būdu aprašytų taisyklių bazė.

Veiklos taisyklių saugykla yra centrinė vieta, kur vadybininkai, analitikai ir programinės įrangos kūrėjai gali apibrėžti, dalytis ir parengti kompanijos veiklos taisykles. Veiklos taisyklių saugykla suteikia technologinę galimybę paskirstymo pritaikomumą, vystymą, išdėstymą ir valdymą.

Veiklos taisyklių saugyklų privalumai:

- Leidžia lengvai modifikuoti veiklos taisykles;
- Išvengiama taisyklių dubliavimo ir prieštaravimų;
- Leidžia priėjimą prie saugyklos taisyklių nustatymui ir pakeitimui;
- Leidžia sistemos realizavimo elementams surasti kiekvieną veiklos taisyklę taip, kad taisyklės suradimas ir pakeitimas gali būti padarytas greitai, atsakant į veiklos iniciatyvos pasikeitimą;
- Leidžia spausdinti modelių ir veiklos taisyklių apibrėžimus, todėl tarpininkai gali jas peržiūrėti.

VT saugykla yra skirta vadybininkams, analitikams ir programuotojams jas apibrėžti, bendrai naudotis jų specifikacija ir parengti kompanijos VT bazę. VT saugykla suteikia technologines galimybes jas paskirstyti pritaikant įvairiems funkciniams uždaviniams, vystyti jų aprašą bei valdyti centralizuotu būdu [13].

Reikalavimai idealiai VT saugyklai:

1. Viena integruota duomenų bazė, skirta UML modelių ir VT saugojimui, valdymui ir pakartotiniam panaudojimui.

2. Papildomas meta modelis.

3. Ataskaitų ir užklausų redaktorius.

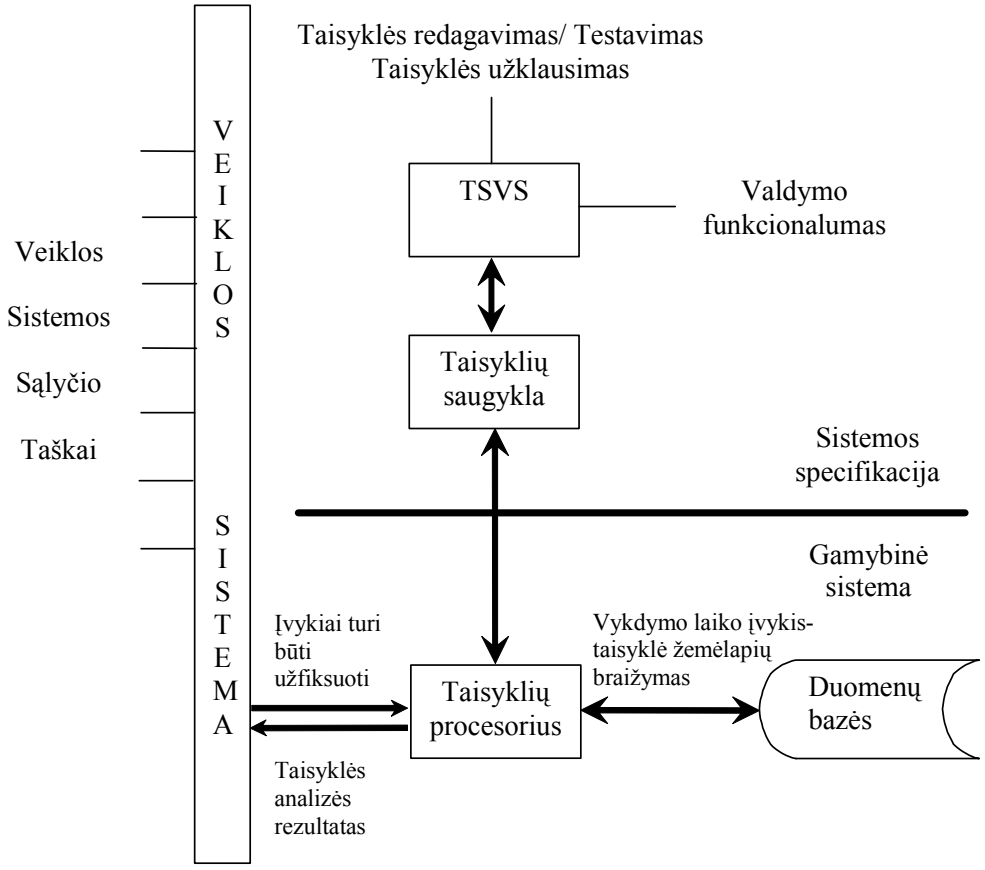
4. Publikavimo Internete galimybės.

5. Duomenų valdymo funkcijos veiklos taisyklių ir modelių integralumui ir saugumui.

6. Galimybė bendrauti (integracijos požiūriu) su kitais projektavimo įrankiais, pavyzdžiui, skirtais veiklos taisyklių manipuliavimui ir testavimui.

2.2.10. VEIKLOS TAISYKLIŲ PROCESORIUS

Veiklos taisyklių tiesioginis palaikymas informacinėse sistemose vis labiau tampa įprastas. Paveiksle (12 pav.) yra parodyta Roso pasiūlyta VT palaikymo architektūra, kuri skirta vykdyti VT naudojant taisyklių procesorių (engine), kuris tiesiogiai pagrįstas VT semantika. VT procesorius specialiai skirtas atmetimo (rejectors) ir išvadų (inferences) taisyklių įgyvendinimui nors procesorius gali valdyti ir kitokių rūšių taisykles, tokias kaip procesų trigeriai ar procedūros [21].



12 pav. Veiklos taisyklių vykdymo platformos architektūra

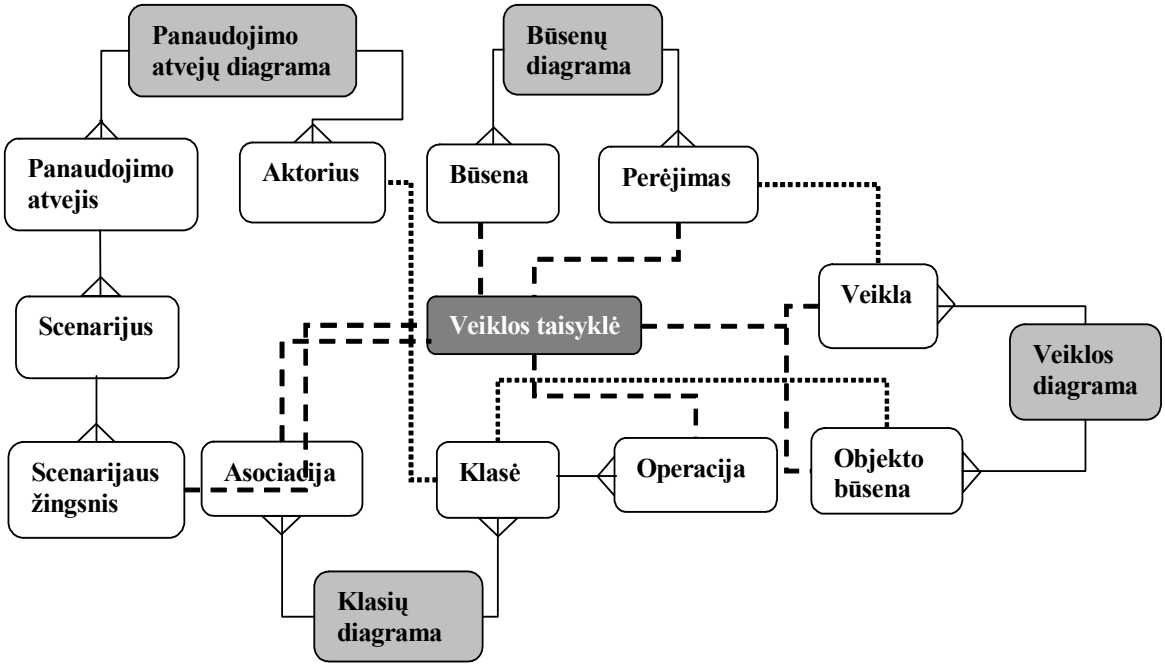
- Taisyklių saugyklos valdymo sistema (TSVS) siūlo tokius funkcionalumus kaip redagavimas (dažniausiai tai atlieka veiklos vartotojas), ratifikavimas ir užklausų pateikimas. Be to ji siūlo tokį valdymo funkcionalumą kaip taisyklės registravimą/išregistravimą, taisyklės aprobavimą ir pritaikomumą [21].
- Taisyklių saugykla, kuri dažnai yra nurodoma kaip taisyklių bazė, talpina veiklos taisykles ir su jomis susijusius meta duomenis, įskaitant taisyklių palaikymo istoriją ir taisyklių valdymo priemones [21].
- Veiklos sistemos yra „ploni“ klientai siūlantys veiklos funkcionalumą darbuotojams, pirkėjams ir bet kam, kad prisijungtų prie organizacijos internetinio puslapio. Visi funkciniai taškai, per kuriuos pasaulis sąveikauja su organizacijos veiklos sistema, gali būti pavadinti „Veiklos sistemos sąlyčio taškai“ [21].
- Duomenų bazė pateikia bet kurį veiklos faktų modelio atskirą atvejį, kuris gali registruoti įvykius atsiradusius veiklos sistemose. Taisyklių procesorius iš tikrųjų veikia kaip filtras, per kurį visi įvykiai turi praeiti. Patvirtinti įvykiai gali būti veiklos žinių būsenos pasikeitimo rezultatu [21].

- Taisyklių procesorius tikrina visus įvykius vykstančius veiklos sistemose. Kai procesorius aptinka žinių būsenos pasikeitimo įvykį, kuris turi būti registruojamas, procesorius naudoja atitinkamą vykdymo laiko įvykis-taisyklės žemėlapio braižymą nustatyti kurios taisyklės gali būti pažeistos įvykio. (Apskritai kiekviena taisyklė gali būti pažeista mažiausiai dviejų įvykių) [21].

Taisyklių procesorius braižo žemėlapį nuo įvykių iki taisyklių dalykiškai. Taisyklių procesorius nustato taisykles prieš naują „siekiamą“ įvykio būseną. Jei taisyklė vykdo atmetimą ir turi būti pažeista, taisyklių procesorius uždraudžia vykdymą ir vartotojui, kuris bandė inicijuoti įvykį, parodoma pažeista taisyklė. Taisyklė yra parodoma tik tuo atveju, jei vartotojas yra įgaliotas pamatyti taisyklę ir gali ją suprasti. Jei taisyklė yra išvadų taisyklė (kuri savaime yra apribojimas), procesorius užtikrina, kad išvada nebūtų sulaužoma, nes ji turi būti įvykdoma ir pripažinta teisinga kaip specifikuota taisyklė. Jei taisyklė yra trigeris, tuomet apibrėžtas procesas yra inicijuojamas procesoriaus [21].

2.2.11. META MODELIO PAPILDYMAS VEIKLOS TAISYKLIŲ KLASE

Daugumos CASE įrankių specifikacijų saugyklos palaiko UML meta modelį. Jei CASE paketo saugyklos meta modelis yra išplečiamas, tuomet galima sustiprinti įrankio UML palaikymą, pritaikant UML įrankio meta modelį tokioms veiklos taisyklių modeliavimo galimybėms, kokių reikia. Galima pridėti klasę *VeiklosTaisyklė*, specifikuoti jos savybes ir tuomet nustatyti asociacijas su kitais UML modeliavimo elementais. Paveiksle (13 pav.) pavaizduotas UML meta modelio fragmentas su pridėta VT esybe ir jos ryšiais (punktyrinės linijos). Reikia pažymėti, kad čia parodyti ne visi ryšiai tarp UML modelio artefaktų (taškinės linijos), kad diagrama nepavirstų į ryšių raizginį [19, 23].



13 pav.UML meta modelis papildytas esybe VeiklosTaisyklė

2.3. VEIKLOS TAISYKLIŲ MODELIAVIMO UML KALBA ANALIZĖS REZULTATAI IR IŠVADOS

1. Analizės dalyje apžvelgiami VT klasifikavimo būdai, pastebėti klasifikavimo principų panašumai ir skirtumai. Taip pat susipažinta su įvairių autorių ir šaltinių siūloma VT klasifikacija. Analizuota literatūra apie VT formalizavimo metodus ir jų išraiškos lygius. Klasifikavimo etape išskirta VT klasifikacija projektuotojo požiūriu ir šiai klasifikacijai pritaikyti šablonai.
2. Buvo analizuojama kokie UML modeliai tinka biznio procesams modeliuoti ir pagal kokią taisyklių klasifikaciją galima įtraukti VT į šiuos modelius. Pasirinktai projektuotojo požiūriu klasifikacijai buvo nustatyti UML modeliai ir VT modeliavimo notacijos. Apžvelgtos UML meta modelio praplėtimo galimybės VT modeliuoti. Pagal pasirinkto modeliavimo įrankio galimybes nustatytas UML meta modelio išplėtimas stereotipo klase.
3. Biznio procesai ir veikla yra modeliuojami efektyviai kai VT yra saugomos struktūrizuotai ir autonomiškai. Todėl VT specifikacijai saugoti siūloma sukurti autonomišką VT saugyklą. Taip pat literatūroje apžvelgtas tokios saugyklos galimas architektūros modelis.

3. VEIKLOS TAISYKLIŲ MODELIAVIMO UML KALBA VERTINIMAS IR PRAPLĖTIMO VARIANTO PARINKIMAS

UML yra populiari informacinių sistemų (IS) modeliavimo ir projektavimo kalba. UML pagrindu yra siūloma daug metodų ir metodologijų. Bet ne visi UML modeliai tinka veiklos taisyklėms modeliuoti. Veiklos taisyklė yra formuluotė, kuri apibrėžia ar suvaržo kai kurias veiklos kryptis. Ji skirta tvirtinti veiklos struktūrą, ją reguliuoti ar veikti veiklos funkcionavimą. UML turi 12 diagramų, iš kurių 7 tinka biznio procesams modeliuoti. Biznio procesai negali egzistuoti be apribojimų – veiklos taisyklių. Šio projekto tikslas yra praplėsti UML modelius, kad jie būtų tinkami veiklos taisyklėms modeliuoti, taip kad sumodeliuotos veiklos taisyklės paveiktų objekto elgseną.

Iki šiol veiklos taisyklės prie modelių pridedamos papildomu tekstu, kuris nepaveikia modelio elgsenos ir neįtakoja jo būsenos. Kadangi kiekviena veiklą reguliuoja arba apriboja tam tikros taisyklės, todėl svarbu jas išskirti ir pavaizduoti projektuojant bet kokią biznio sistemą. UML į savo notaciją įtraukia objektų apribojimo kalbą (OCL), bet ji yra paslėpta modelio specifikacijose ir taisyklių nematome, jei jų neprijungiame kaip atskiro papildomo teksto. Modeliuojant veiklą yra svarbu matyti tas taisykles ir jei taisyklė pasikeičia, jas operatyviai pakeisti. Tai įmanoma tuomet, kai taisykles įmanoma modeliuoti atskiru elementu ir reguliuoti, kad ta pati taisyklė nebūtų panaudota kelis kartus skirtingų projektavimo specialistų. VT modeliavimas atskiru modelio elementu leidžia sukontroliuoti taisyklių panaudojimą, keisti jas greitai ir operatyviai joms pasikeitus, bei įtakoti jomis tam tikrą veiklą.

Norint praplėsti UML modelius reikia nagrinėti priemones, kuriomis galima modeliuoti biznio procesus ir kurios turi atviras modelių saugyklas (repository). MagicDraw UML notacijoje pasakyta, kad jo saugykla yra atvira ir ją galima praplėsti OCL kalbos pagalba, ar sukurti naują modelio elementą, stereotipo pagalba.

Kompiuterizuojamos sistemos varianto pasirinkimą lėmė šie kriterijai: pilnas UML palaikymas, diagramų eksportas, platformų palaikymas, kodo generavimas, duomenų modeliavimas, tiesioginės ir atvirkštinis projektavimas, repozitoriaus palaikymas, paketo atvirumas pildymams ir veiklos taisyklių modeliavimas. Buvo lyginami šie CASE įrankiai: Rational Rose 2000, MagicDraw, Visio 2002 Enterprise, Ameos, Visual Paradigm (7 lentelė).

7 lentelė. CASE įrankų palyginimas

CASE įrankis / Kriterijus	Rational Rose 2000	Magic-Draw	Visio 2002 Enterprise	Ameos	Visual Paradigm for UML
Pilnas UML palaikymas	+	+	+	+	+
Diagramų eksportas	+	+	+	+	+
Platformų palaikymas	C++, Visual C, Java, VBasic	C++, Java	Visual C, Java	C, C++, Java	C++, Java
Kodo generavimas	+	+	+	+	+
Duomenų modeliavimas	+	-	+	-	-
Tiesioginis ir atvirkštinis projektavimas	+	+	+	+	+
Repozitoriaus palaikymas	+	+	+	+	+
Kodo atvirumas	+	+	+	+	-
Veiklos taisyklių modeliavimas	+ (OCL kalba, sintaksė netikrinama)	+ (OCL kalba, sintaksė tikrinama)	galima sukurti savo figūrą, bet specialios formos nėra	-	-

Pagal lentelės duomenis daugiausiai privalumų turi Rational Rose 2000, bet šio paketo galimybės yra plačiai išnagrinėtos, o šio projekto tikslas yra pabandyti pritaikyti veiklos taisyklių modeliavimą kitiems populiariems projektavimo įrankiams. Antras pagal populiarumą CASE įrankis yra MagicDraw, bet jo trūkumas prieš Visio yra tas, kad MagicDraw dirba Java VM platformoje, o Visio Windows platformoje. Ameos iš Aonix įrankis dar nėra prieinamas viešam naudojimui.

Projektui įvykdyti buvo pasirinkta MagicDraw UML iš No Magic. Šio paketo oficialus platintojas Lietuvoje yra UAB “Baltijos Programinė Įranga”. Lentelėje (8 lentelė) yra pateikti šio paketo privalumai.

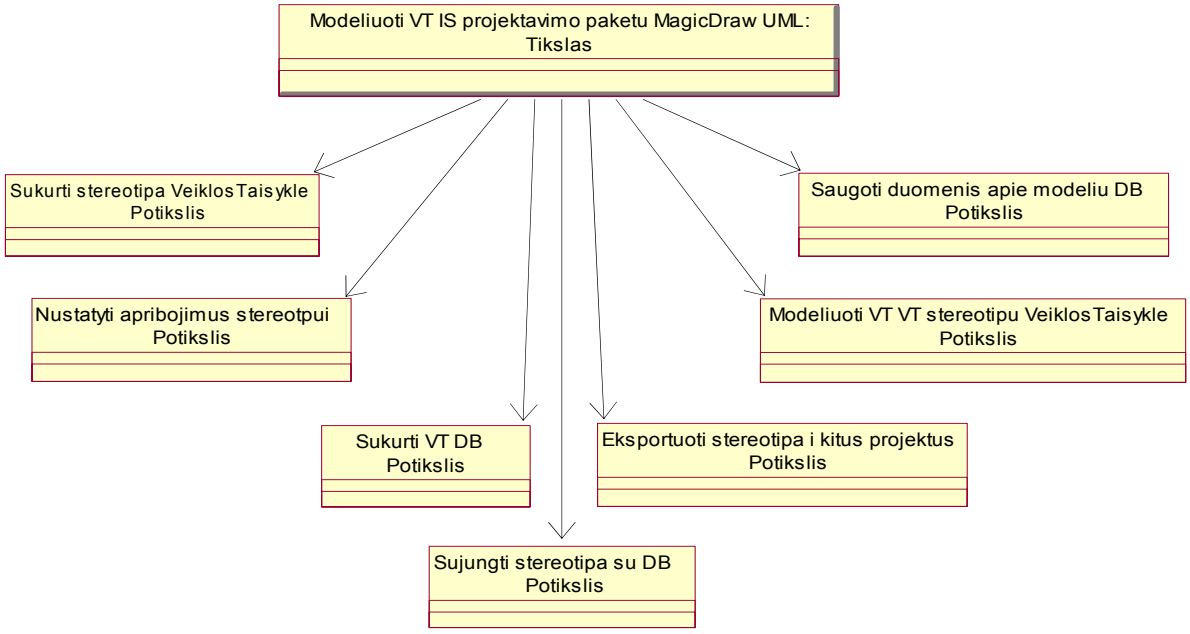
8 lentelė. Programinio paketo MagicDraw privalumai

<table>
<tr>
<td>MagicDraw yra efektyvus UML modeliavimo įrankis, skirtas:
• IT architektams
• Programuotojams
• Kokybės užtikrinimo inžinieriams
• Programinės įrangos analitikams
• Programinės įrangos dokumentacijos kūrėjams
• Verslo analitikams</td>
<td>UML Modeliavimas
<u>Struktūrinės diagramos:</u>
• Klasių;
• Objektų;
• Komponentų;
• Realizavimo (deployment).
<u>Elgsenos diagramos:</u>
• Panaudojimo atvejų;
• Sekų;
• Bendradarbiavimo;
• Veiklos;
• Būsenų.
<u>Modelio tvarkymo diagramos:</u>
• Paketų;
• Posistemių;
• Modelio.</td>
</tr>
<tr>
<td colspan="2">MagicDraw kodo inžinerijos įrankiai veikia su šiomis technologijomis:
• Java: Išeities kodas, Binarinis kodas, EJB 2.0
• C#
• C++
• CORBA IDL
• DB modeliavimas (DDL): Oracle, DB2, MS SQL Server, Sybase, Pointbase, MySQL, PostgreSQL, Pervasive SQL
• .NET (CIL)
• WSDL
• XML Schema</td>
</tr>
</table>

4. SISTEMŲ PROJEKTAVIMO ĮRANKIO MAGICDRAW MODELIAVIMO GALIMYBIŲ IŠPLĖTIMO PROJEKTO MODELIS

4.1. PROJEKTO TIKSLAS

Realizavus šį projektą būtų galima modeliuoti VT. Kiekvienas, dirbantis su paketu ar programuojantis sistemą pagal modelius, matytų veiklos apribojimus, sąlygas procesams ir t.t., kurie paprastai nespecifikuojami UML kalba, kaip matomi elementai. Pagal tokius modelius sukurtą sistemą lengva modifikuoti ir pritaikyti naujoms veiklos taisyklėms. Žemiau pateiktas sistemos tikslų modelis (14 pav.).



14 pav. Sistemos tikslų modelis

Projekto sritis – projektas turės įtakos sistemų modeliuotojams, analitikams ir programuotojams.

Veiklos galimybės ir problemos

Problemos – sunku rasti ir vieningai panaudoti veiklos taisykles, jei jos prie modelio prijungtos kaip paprastas tekstas, ir be to toks jų aprašymas sistemos įgyvendinimui įtakos neturi.

Galimybės –pagerinti sistemos modeliavimą naudojant VT, pagreitinti VT atnaujinimą modeliuose.

Veiklos sritis

Vidiniai dalyviai – analizės specialistas, projektuotojas ir programuotojas.

Išoriniai dalyviai – veiklos savininkas, analizės specialistui išsakantis veiklos pakitimus.

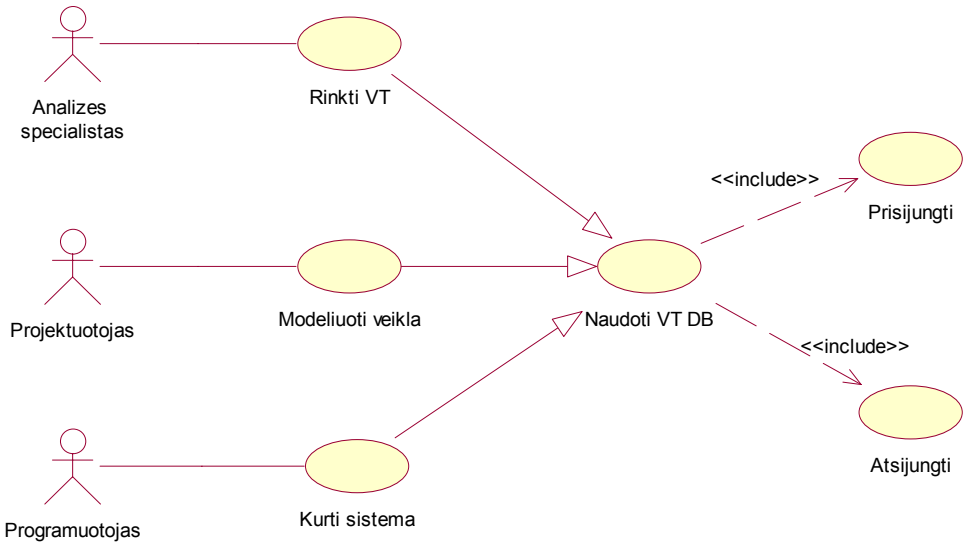
4.2. TOBULINAMO PROJEKTAVIMŲ ĮRANKIO REIKALAVIMŲ MODELIS

4.2.1. FUNKCINIAI REIKALAVIMAI ĮRANKIO PATOBULINIMUI

Sistemos funkciniai reikalavimai yra orientuoti į vartotoją, jo pageidavimų realizavimą. Vartotojo poreikius geriausiai parodo panaudojimo atvejų diagramos. Vartotojų panaudojimo atvejų modelyje aktoriais yra šie asmenys:

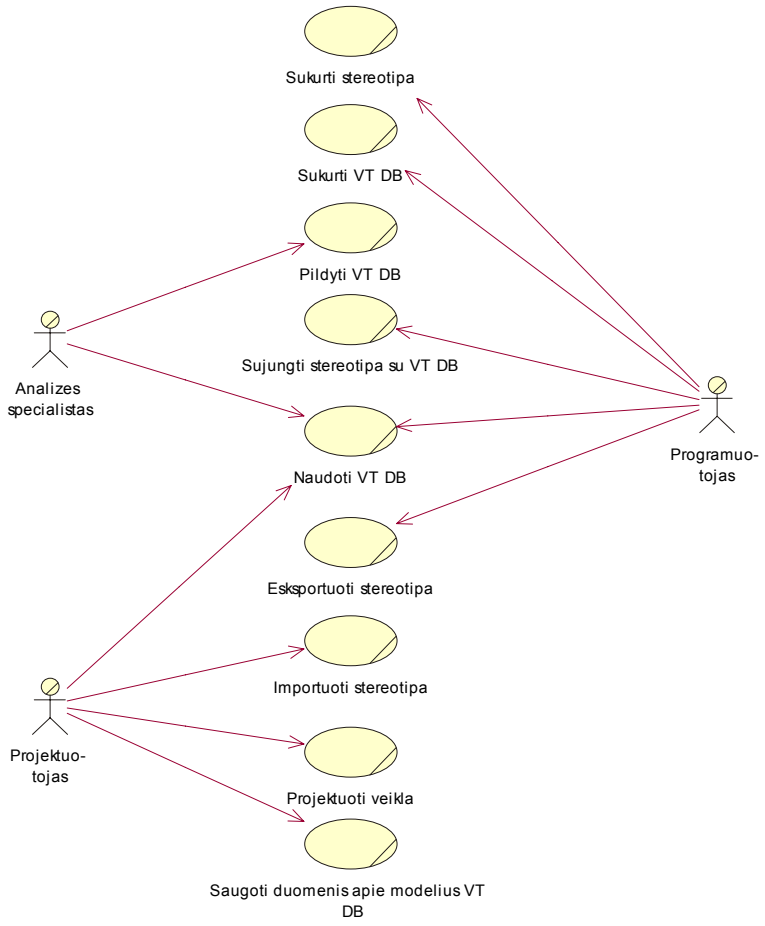
- analizės specialistas – bendrauja su veiklos savininku ir išsiaiškina veiklą ribojančias taisykles. Jis tas VT struktūrina ir pildo VT duomenų bazę (DB). Tik analizės specialistas DB gali keisti VT, kad nebūtų išgadinta VT sandara.
- programuotojas – sukuria veiklos taisyklių duomenų baze (jos loginę struktūrą). Sistemų projektavimo įrankyje MagicDraw UML sukuria stereotipą <<VeiklosTaisykle>>. Stereotipui nustato struktūros apribojimus ir ODBC draiverių pagalba sujungia sukurtą DB su stereotipu <<VeiklosTaisykle>>. Sukurtą stereotipą programuotojas eksportuoja t.y. užsaugo XML formatu. Stereotipą bus galima naudoti bet kokiame projekte, bet jį reiks importuoti.
- projektuotojas – importuoja į projektą stereotipą <<VeilosTaisykle>>. Projektuoja sistemą, įtraukdamas į ją reikalingas taisykles. Taisyklių išrinkimui naudoja VT duomenų bazę. Duomenų bazėje išsaugo duomenis apie modelius, kuriuose buvo panaudotos konkrečios VT. Projektuotojas naujų VT nekuria.

Pagrindinės sistemos vartotojų veiklos pavaizduotos vartotojų panaudojimo atvejų diagramoje (15 pav.).



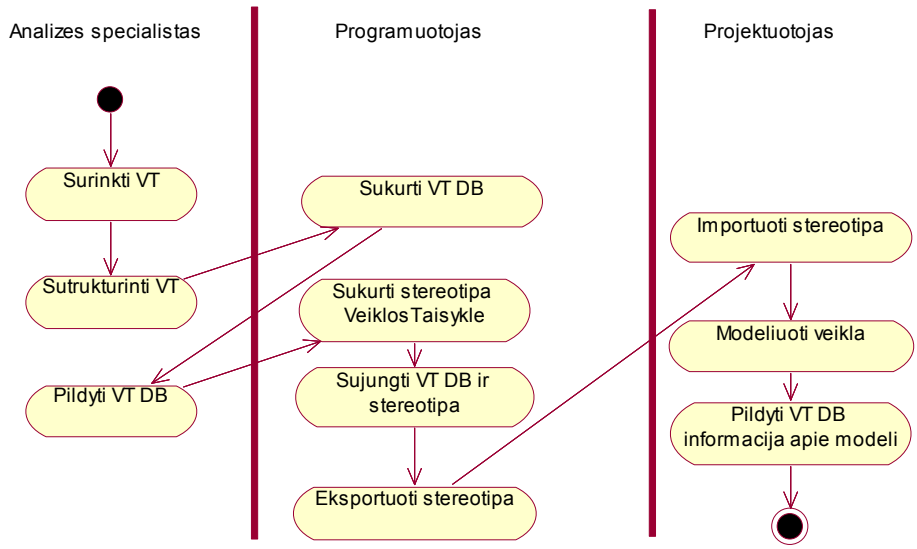
15 pav. Vartotojų panaudojimo atvejų diagrama

Vartotojų panaudojimo atvejų modelis atskleidžia sistemos vartotojų reikalavimus. Pagal šiuos reikalavimus galima detalizuoti veiklos panaudojimo atvejų diagramas. Veiklos panaudojimo atvejų diagrama atspindi ne kompiuterinės sistemos panaudojimo atvejus, bet analizuojamoje organizacijoje vykdomus veiklos procesus (16 pav.).



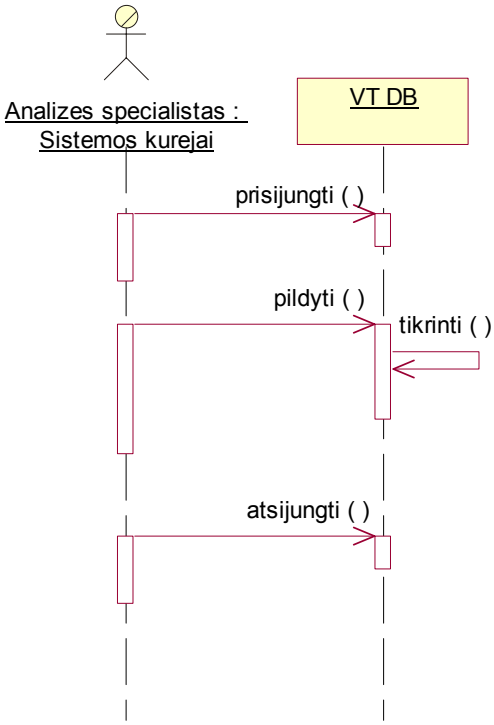
16 pav. Veiklos panaudojimo atvejų modelis

Pagal veiklos panaudojimo atvejų modelį sudaroma veiklos procesų diagrama, kuri parodo kokie procesai vykdomi ir kokia tvarka įgyvendinamas projektas (17 pav.).

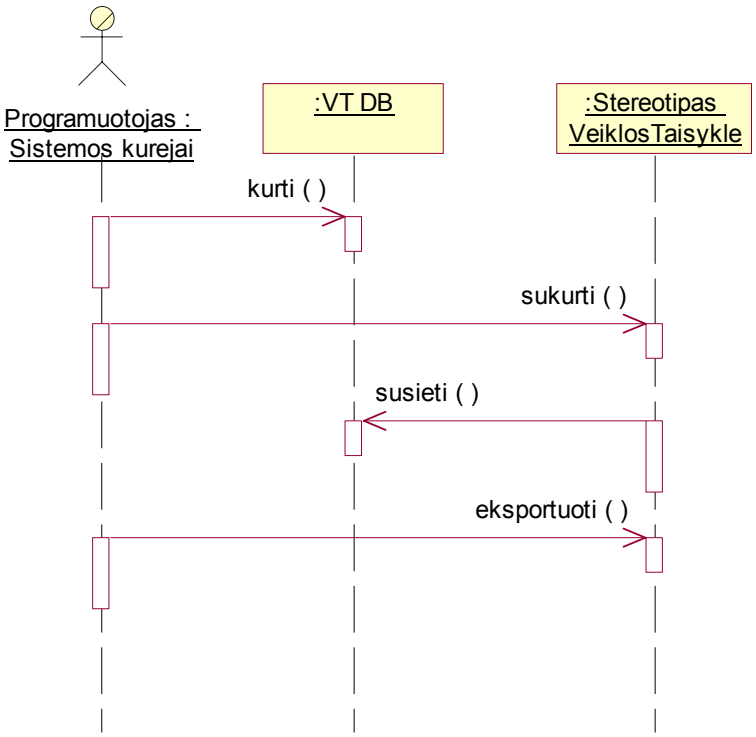


17 pav. Veiklos procesų diagrama

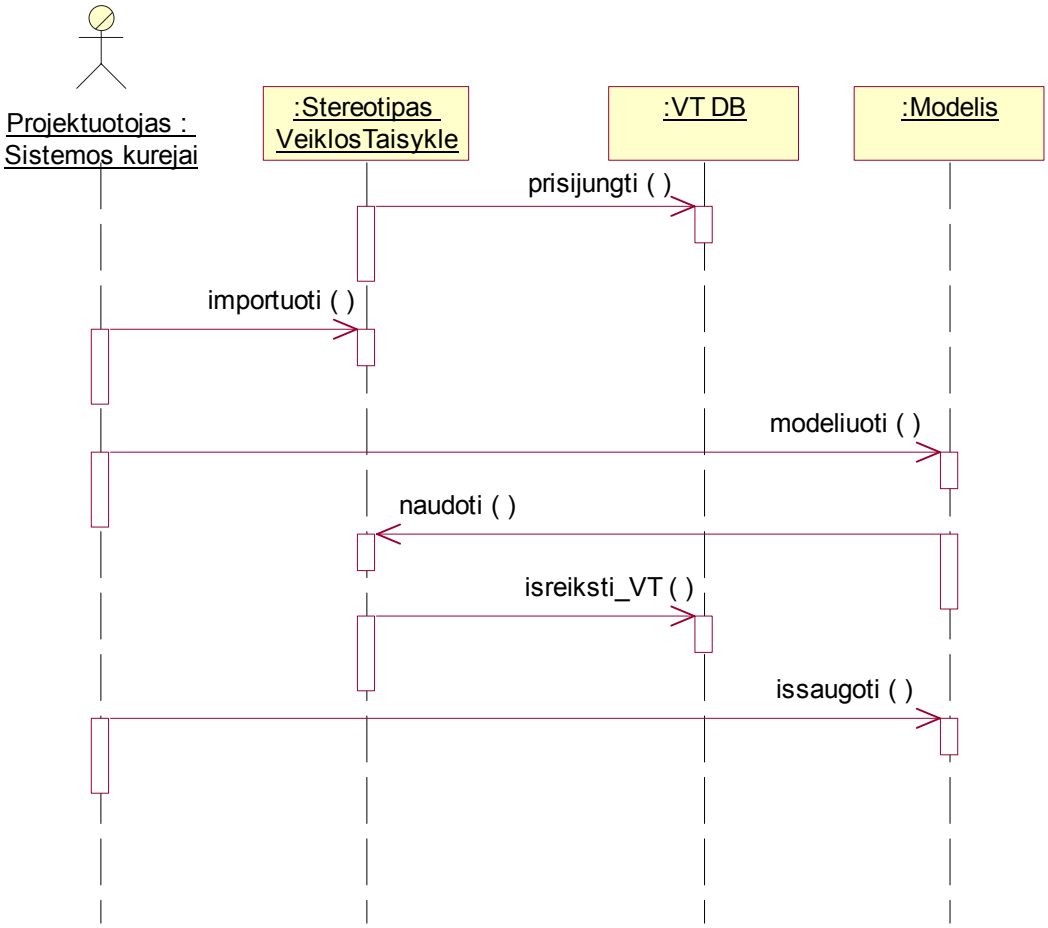
Iš veiklos procesų diagramos detaliau galima būtų išskirti analizės specialisto, programuotojo ir projektuotojo veiksmus. Sekų diagramos geriausiai iliustruoja atliekamus vartotojo veiksmus (18, 19, 20 pav.).



18 pav. Vartotojo „Analizės specialistas“ sekų diagrama



19 pav. Vartotojo „Programuotojas“ sekų diagrama



20 pav. Vartotojo „Projektuotojas“ sekų diagrama

Kadangi nėra kuriamas visiškai naujas projektas, todėl neprojektuojamas vartotojo sąsajos modelis. Vartotojo sąsaja išlieka nepakitusi, nes projektavimo įrankyje Magic Draw yra įterpiamas tik naujas stereotipas, kuris vartotojo sąsajos nepakeičia.

4.2.2. NEFUNKCINIAI REIKALAVIMAI ĮRANKIO PATOBULINIMUI

Nefunkciniai reikalavimai apima tokias sritis kaip: funkcionalumą, patikimumą, patogumą, efektyvumą, perkeliamumą ir kitas savybes.

Reikalavimai įtakojantys funkcionalumą sprendžia tokias problemas kaip tinkamumas sprendžiamiems uždaviniams, rezultato tikslumas, sąveiką su kitomis sistemomis ir kt.

Projektuojamos sistemos uždavinys yra veiklos taisyklių modeliavimas UML kalba. VT galima modeliuoti naudojant apribojimų kalbą ar kitas UML kalbos notacijas, bet tai neišsprendžia pagrindinės problemos – siekiama ne tik VT matyti tiesiogiai modelyje, bet ir

paveikti veiklos logiką, o ne paslėpti VT programos kode ar modelio elemento savybėse. Nuo sistemos tinkamumo sprendžiamiems uždaviniams priklauso ir rezultatų tikslumas. Jei sistema suskaidyta į modulius, kuriuos modeliuoja skirtingi projektuotojai, tai kuriamos sistemos tikslumas priklauso nuo to, kaip projektuotojai bendrauja. Jei sistemą modeliuoja vienas projektuotojas, tai problemų neiškyla, bet kai sistemos modeliavimas paskirstomas tarp dviejų ir daugiau projektuotojų, tai jau atsiranda painiava ir nuo to arba suprastėja kuriamos sistemos kokybė arba atsiranda didelės laiko sąnaudos. Todėl yra užsibrėžta pagerinti sistemų projektavimo įrankio MagicDraw UML funkcionalumą, o tiksliau pritaikomumą sprendžiamiems uždaviniams ir kokybiškesnės sistemos kūrimą.

Kitas svarbus faktorius yra kuriamos sistemos sąveika su kitomis sistemomis. Sistema turi privalumų tuomet, kai ją galima importuoti į kitas sistemas arba eksportuoti kitus modelius ir standartus. Magic Draw gali bendrauti su didele dalimi sistemų, kurios palaiko UML 1.4, CORBA IDL, WSDL ir XML, todėl šios jos savybės gerinti nereikia ir tai nėra labai aktualu.

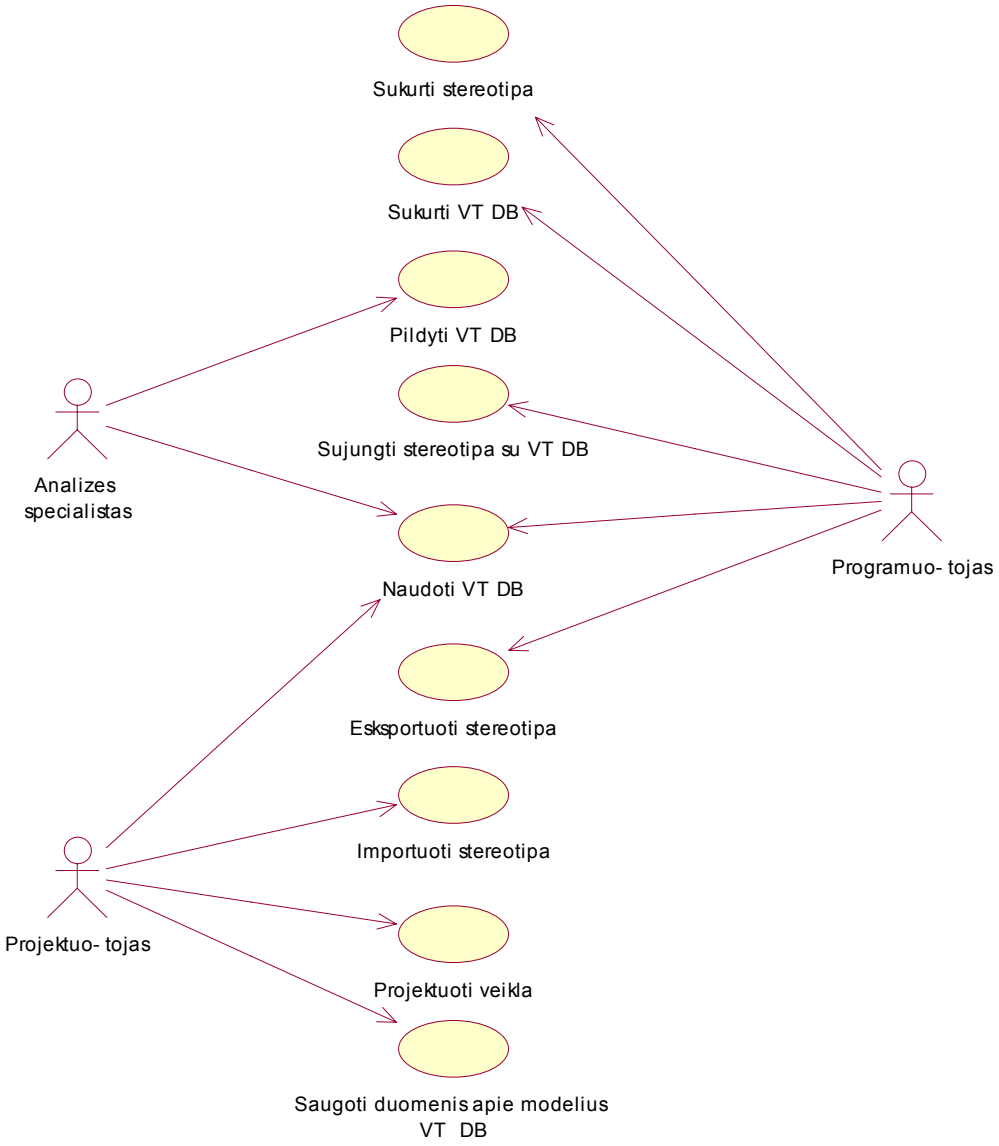
Patikimumas taip pat yra svarbi sistemos savybė. Patikimumui priskiriamas sistemos užbaigtumas t.y. klaidų dažnumas. Jei sistema daro daug klaidų, tai ji yra nepatikima. Paketas MagicDraw nėra iš pagrindų keičiamas, todėl jo patikimumas nesikeičia. Šiuo atveju patikimumas priklauso nuo duomenų bazės sąsajos su stereotipu <<*Veiklostaisykle*>> ir efektyvaus duomenų atnaujinimo DB. Jei duomenys atnaujinami neefektyviai, tai nuo to gali nukentėti ir sistemos patikimumas.

Nefunkciniams reikalavimams priskiriamas ir vartotojo servisas. Vartotojui labai svarbus sistemos paprastumas, išmokstamumas, vykdymo savybės ir sąsajos patrauklumas. Paketo MagicDraw vartotojo sąsaja nebus keičiama iš pagrindų, nes tai nėra įtraukta į tikslus. Todėl reikalavimai vartotojo sąsajai nėra keliami ir jie lieka tokie, kokius yra sumanę tikrieji įrankio kūrėjai.

Kiekviena sistema yra patraukli ir priimtina, jei ji dirba efektyviai. Efektyvumu laikomas greitas programos vykdymas ir nedidelis išteklių naudojimas. Šiuo atveju siekiama pagerinti metodą, kuriuo modeliuojamos veiklos taisyklės. Todėl tobulinamos sistemos tikslas yra automatinis taisyklių išrinkimas iš VT DB ir automatinis jų įkėlimas į modelį.

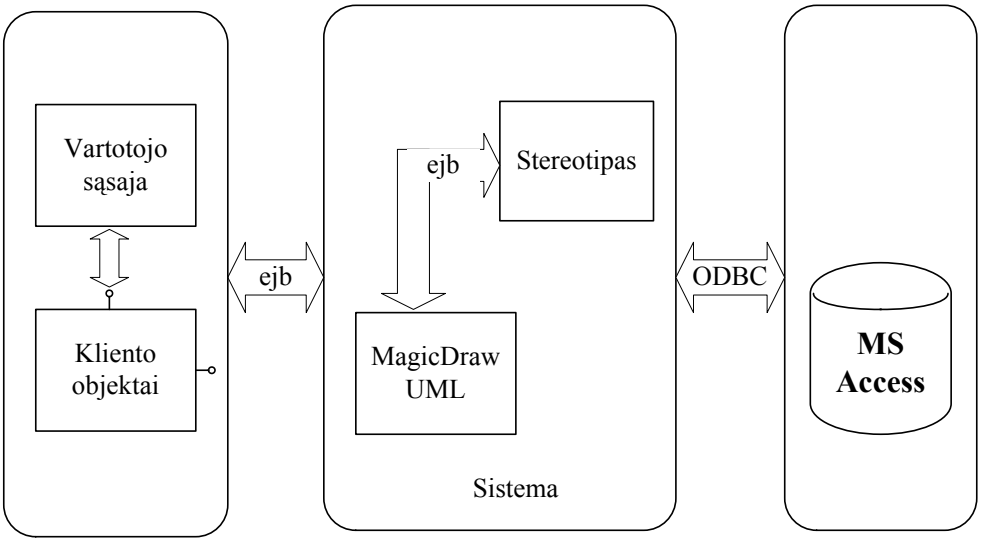
4.3. ĮRANKIO MAGICDRAW TOBULINIMO PROJEKTO MODELIS

Sistemos projekto modelis parodo sistemos veiksmų seką sprendžiant uždavinius, atliekant tam reikalingus veiksmus. Sistemos veiksmus geriausiai iliustruoja sistemos panaudojimo atvejų diagrama (21 pav.).



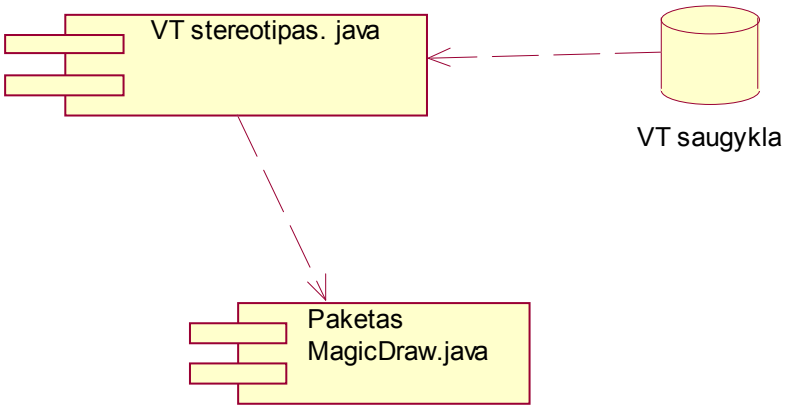
21 pav. Sistemos panaudojimo atvejų diagrama

Žemiau pateiktas galimas sistemos architektūros statinės struktūros modelis, kuris apima visus posistemius bei lygius: atvaizdavimo (vartotojo interfeiso) lygį, taikomųjų programų lygį, duomenų bazės lygį, bei veiklos taisyklių specifikavimo lygį (20 pav.).



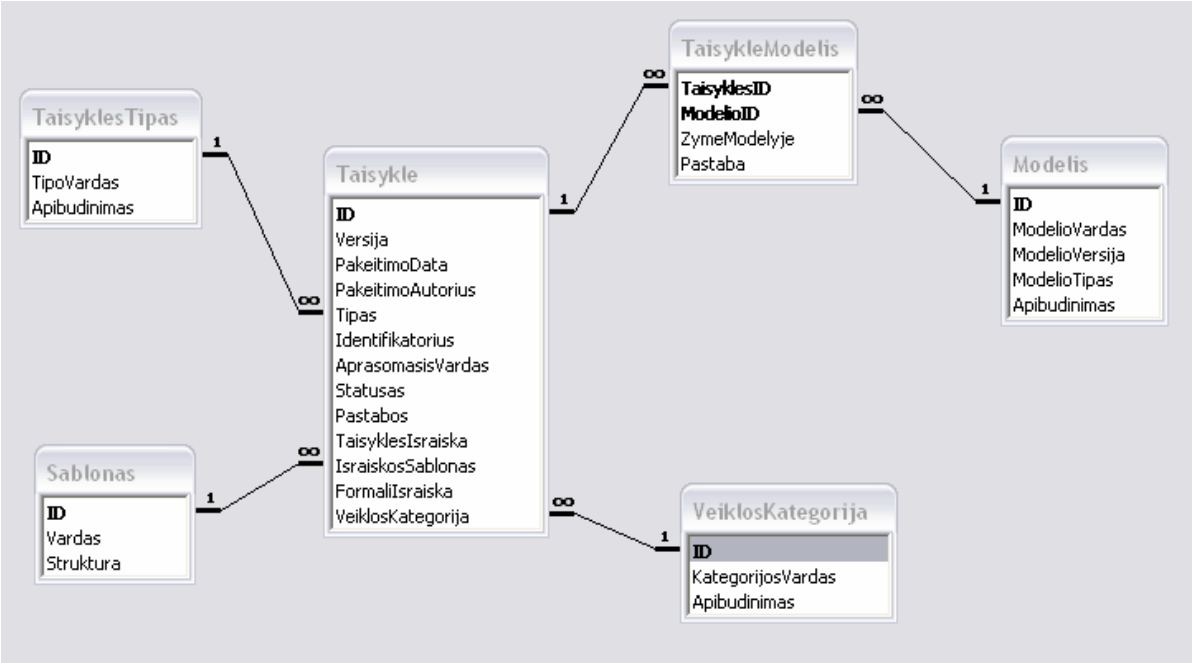
20 pav. Planuojamas sistemos architektūros modelis

Sistemos komponentų modelis (21 pav.) parodo iš kokių komponentų susideda sistema, ir kokie komponentai sąveikauja tarpusavyje.



21 pav. Komponentų modelis

VT saugykla kuriama MS Access duomenų bazių kūrimo programa. Žemiau pateiktas duomenų bazės modelis (22 pav.).



22 pav. Duomenų bazės loginė struktūra

Šią duomenų bazės loginę struktūrą apibūdina tokios specifikacijos (9, 10, 11, 12, 13, 14 lentelės).

9 lentelė. Duomenų lentelės „Taisykle“ specifikacija

	Lauko pavadinimas	**Duomenų tipas**	**Apibūdinimas**
key	ID	AutoNumbet	Taisyklės identifikatorius
	Versija	Text	Versijos numeris, atitinkantis bet kokį formatą
	PakeitimoData	Date/Time	Taisyklės modifikavimo data
	PakeitimoAutorius	Text	Atsakingo asmens vardas
	Tipas	Number	Sąsaja su taisyklės tipu
	Identifikatorius	Text	Taisyklės identifikatorius, nepriklausomas nuo aplinkos
	AprasomasisVardas	Text	Taisyklės tekstinis apibūdinimas
	Statusas	Text	Taisyklės statuso identifikatorius
	Pastabos	Memo	Pastabos
	TaisyklesIsraiska	Memo	VT išraiška XML‘u
	IsraiskosSablonas	Number	Sąsaja su šablonu
	FormaliIsraiska	Memo	Formalus taisyklės aprašymas
	VeiklosKategorija	Number	Sąsaja su veiklos kategorija

10 lentelė. Duomenų lentelės „TaisykleModelis“ specifikacija

	Lauko pavadinimas	**Duomenų tipas**	**Apibūdinimas**
Key	TaisyklesID	Number	Sąsaja su taisykle
key	ModelioID	Number	Sąsaja su modeliu
	ZymeModelyje	Text	Rodyklė į taisyklės buvimo vietą modelyje
	Pastaba	Memo	Pastabos

11 lentelė. Duomenų lentelės „Modelis“ specifikacija

	Lauko pavadinimas	**Duomenų tipas**	**Apibūdinimas**
Key	ID	AutoNumber	Modelio identifikatorius
	ModelioVardas	Text	Tekstinis modelio vardas
	ModelioVersija	Text	Modelio versija
	ModelioTipas	Text	Modelio tipas
	Apibudinimas	Memo	Modelio apibūdinimas

12 lentelė. Duomenų lentelės „VeiklosKategorija“ specifikacija

	Lauko pavadinimas	**Duomenų tipas**	**Apibūdinimas**
Key	ID	AutoNumber	Kategorijos identifikatorius
	KategorijosVardas	Text	Tekstinis kategorijos vardas
	Apibūdinimas	Memo	Apibūdinimas

13 lentelė. Duomenų lentelės „Sablonas“ specifikacija

	Lauko pavadinimas	**Duomenų tipas**	**Apibūdinimas**
Key	ID	AutoNumber	Šablono identifikatorius
	Vardas	Text	Tekstinis šablono pavadinimas
	Struktura	Memo	Šablono struktūra XML‘u

14 lentelė. Duomenų lentelės „TaisyklesTipas“ specifikacija

	Lauko pavadinimas	**Duomenų tipas**	**Apibūdinimas**
Key	ID	AutoNumber	Tipo identifikatorius
	TipoVardas	Text	Tekstinis tipo vardas
	Apibūdinimas	Memo	Tipo apibūdinimas

4.4. PROJEKTO DALIES IŠVADOS

1. Reikalavimų modelyje buvo apibrėžti tobulinamos sistemos projektavimo tikslai, kokiems vartotojams tokia sistema reikalinga ir kas ja naudosis. Reikalavimų modelis išreikštas per sistemos vartotojų ir veiklos panaudojimo atvejų diagramas. Pagal veiklos panaudojimo atvejų modelį sudaryta veiklos procesų diagrama. Taip pat pavaizduotos sistemos vartotojų sekų diagramos. Buvo apibrėžti tobulinamos sistemos nefunkciniai reikalavimai ir išskirti sekantys: tinkamumas specifikuotiems uždaviniams spręsti, rezultatų tikslumas, duomenų atnaujinimo kriterijai bei greitas programos vykdymas.
2. Projektavimo modelyje sudarytas sistemos panaudojimo atvejų diagrama, pristatytas tobulinamos sistemos architektūros modelis, bei pasiūlyta loginė duomenų bazės struktūra.
3. Remiantis šia darbo dalimi buvo sukurtas sistemos patobulinimas, pasinaudojus UML praplėtimo mechanizmu ir galimybe susieti tą mechanizmą su pasiūlytos struktūros duomenų baze.

5. KURIAMO MODELIO APRAŠAS, SĄVEIKA SU EGZISTUOJANČIA SISTEMA IR REZULTATAI

5.1. KURIAMŲ ELEMENTŲ APRAŠAS IR INTEGRAVIMO PRINCIPAI

Sistemą sudaro duomenų bazė, sistemų projektavimo įrankis ir manipuliavimo mechanizmas, kuris sujungia duomenų bazę su sistemų projektavimo įrankiu.

Duomenų bazėje saugoma informacija apie taisykles, jų struktūrą ir panaudojimą UML modeliuose.

Veiklos taisyklės iš vartotojų išgaunamos tekstine forma, kuri yra netinkama sistemų kūrėjams ir programuotojams. Todėl analitikai VT analizuoja ir suveda jas į struktūrinę formą. Šiame darbe yra naudojami VT šablonai. VT ir šablonai yra užrašomi taip pat XML'u, nes tai šiuo metu efektyviausia duomenų integracijos kalba, kuri nepriklausoma nuo sistemos platformos.

Atliekant VT analizę buvo apžvelgtos įvairios VT klasifikacijos ir šio projekto įgyvendinimui buvo pasirinkta VT klasifikacija, kuri sudaryta remiantis projektuotojo požiūriu. Ši klasifikacija buvo pasirinkta todėl, kad projektuojant sistemą jau turi būti atlikta veiklos analizė ir išskirtos veiklai būdingos taisyklės, kurios galėtų būti sustruktūrintos taip, kad jas lengva būtų įtraukti UML modelį kaip atskirą jo elementą, o ne paslėptą atributo savybę, kuri bus matoma tik sistemos programavimo etape.

Projektuotojo požiūriu taisyklės yra skirstomos į apribojimo ir išvedimo taisykles.

Apribojimo taisyklės yra sakiniai, kurie išreiškia sąlygas, kurios turi būti patvirtintos arba paneigtos. Jos apibrėžia ryšio kardinalumą ir pasirinktinumą, priklausomybę tarp atributo reikšmių. Ryšio kardinalumas ir pasirinktinumas yra ryšio objektų savybės.

Apribojimo taisyklės dar skiriamos į atsako (reakcijos), operacijų apribojimo ir struktūros apribojimo veiklos taisykles.

Atsako (reakcijos) taisyklės apriboja elgesį, nusako sąlygas, prie kurių galima vykdyti operaciją. Tokios taisyklės sutinkamos darbų sekų ir būsenų perėjimo diagramose.

Atsako VT šablonas: *KAI* <taisyklės tekstas>

JEIGU <taisyklės tekstas>

TUOMET <taisyklės tekstas>

Operacijų apribojimo taisyklės nurodo, kokios sąlygos turi būti tenkinamos prieš operaciją ir po jos, kad būtų užtikrintas vykdymo teisingumas. Operacijos prieš-sąlygos

išreiškia apribojimus, prie kurių operacija atliekama teisingai. Operacijos po-sąlygos garantuoja rezultatus.

Operacijų apribojimo VT prieš-sąlygos šablonas: <taisyklės tekstas> *TIK TUOMET JEIGU* <taisyklės tekstas>.

Operacijų apribojimo VT po-sąlygos šablonas: <taisyklės tekstas> *TEISINGAI ATLIKTA TIK TUOMET JEIGU* <taisyklės tekstas>.

Struktūros apribojimo taisyklės aprašo objektų ir jų sąryšių sąlygas, kurių negalima pažeisti. Taisyklė gali apriboti atributo reikšmes, objekto tipo egzempliorius ar ryšio kardinalumą.

Struktūros apribojimo VT šablonas: *VISADA GALIOJA KAD* <taisyklės tekstas>.

Išvedimo taisyklės taip pat turi savo skirstymą ir šiame darbe naudojami vienas iš galimų jų šablonų.

Išvadų taisyklės, tai sąlygų aibė, kuri sukuria naują terminą. Visi taisyklėje naudojami terminai turi būti apibrėžti iš anksto. Išvadų taisyklės siejamos su esybėmis, o ne su atributais.

Išvadų VT šablonas: *JEI* <taisyklės sakinys> [*IR* <taisyklės sakinys>
IR <taisyklės sakinys> ...]
TUOMET <išvados pateikimas>.

Skaičiavimų taisyklės tai algoritmai, kurie parodo kaip gauti naują terminą. Algoritme naudojami terminai turi būti apibrėžti iš anksto ir tai dažniausiai skaitinės reikšmės. Skaičiavimo taisyklės apibrėžiamos kaip metodai klasių diagramose.

Skaičiavimų VT šablonas: <Term[2]> *APSKAIČIUOJAMAS TAIP* <formulė>.

Visas šitas taisykles pavaizduosiu per naujo vartotojo registravimo pašto sistemoje pavyzdį.

Registruojant naują pašto sistemos vartotoją, jis gali būti registruojamas tokia tvarka ir tikrinamos tokios taisyklės:

1. susikurti vartotojo vardą;
   1.1. įvesti vartotojo vardą;
   1.2. tikrinti vartotojo vardą vartotojų duomenų bazėje

Taisyklė: *KAI* registruojamas vartotojas *JEIGU* tokio vartotojo nėra duomenų bazėje *TUOMET* jam pripažįstamas šis vartotojo vardas.

[2] Term = Terminas

2. susikurti slaptažodį;

2.1. įvedamas slaptažodis;

2.2. patvirtinamas slaptažodis;

2.3. tikrinama ar slaptažodžiai sutampa

Taisyklė: *JEIGU* slaptažodis pakartotinai suvedus sutapo *TUOMET* pereiti į kitą registravimosi etapą;

3. tvirtinti registraciją

3.1. tvirtinama registracija;

3.2. tikrinama ar vartotojas užregistruotas

Taisyklė: vartotoją sistemoje užregistruoti *TIK TUOMET JEIGU* visos operacijos buvo atliktos korektiškai;

4. prisijungti prie sistemos,

4.1. jungiamasi prie sistemos,

4.2. leidžiama pasikeisti vartotojo slaptažodį,

Taisyklė: VISADA GALIOJA, KAD vartotojas gali pakeisti savo slaptažodį

4.3. vartotojui parodoma jo pašto dėžutėje esančios laisvos vietos kiekis procentais.

Taisyklė: laisvos vietos vartotojo pašto dėžutėje procentinis kiekis *APSKAIČIUOJAMAS TAIP*: (vartotojo nepanaudotas megabaitų kiekis/vartotojo informacijai kaupti skirtų megabaitų kiekis)* 100 procentų.

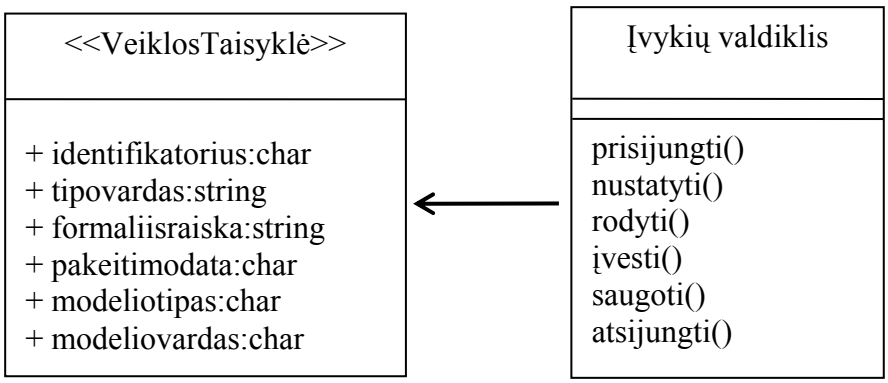
Šiame pavyzdyje pavaizduotos šakos kurios grąžina tik teigiamą reikšmę, neigiamos atšakos bus pavaizduotos veiklos proceso modelyje.

Visą šitą aukščiau pateiktą pavyzdį galima aprašyti OCL kalba. Tokiu atveju taisyklės nebus matomos modelyje ir nebus galima operatyviai jų keisti be profesionalaus projektuotojo pagalbos, jei VT veiklos vykdymo eigoje pasikeistų. Norint išspręsti šią problemą, buvo siūloma VT modeliuoti atskiru modelio elementu.

Sistemų projektavimo įrankis MagicDraw leidžia išplėsti modeliavimo galimybes naudojant stereotipą. Stereotipas yra UML modelio praplėtimo mechanizmas, kuris apibrėžia naują ir labiau specializuotą modelio elementą, pagrystą jau egzistuojančiu elementu. Stereotipas gali būti paremtas visų tipų elementais: klasėmis, paketais, komponentais ir pastabomis taip pat ryšiais – asociacijomis, generalizacija ir priklausomybėmis. Stereotipinis elementas įgyja visas pagrindinio elemento savybes ir gali turėti įprastinius apribojimus. Tokiu atveju stereotipai laikomi kaip atskiri modelio elementai.

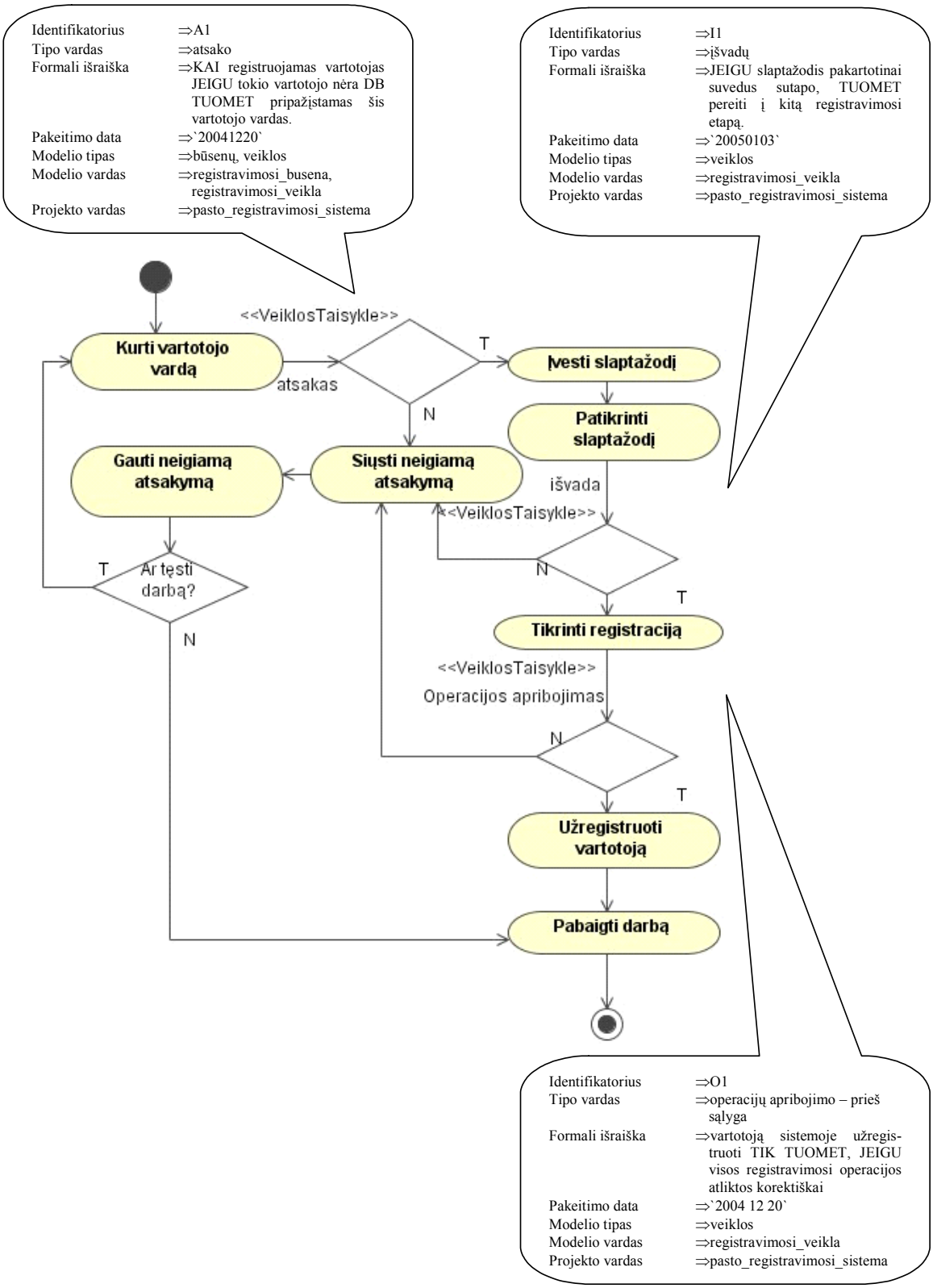
Savo darbe sukūriau stereotipą <<VeiklosTaisykle>> kuris paveldėjo klasės ModelElement visas savybes. Šiuo elementu galima modeliuoti daugelį elementų, bet reikia sudėti apribojimus, kad jo nebūtų galima modeliavimo metu panaudoti klaidingai.

Stereotipas yra sujungtas su VT duomenų baze per ODBC draiverius, nes tai skirtingų programavimo kalbų programiniai komponentai. Paketui MagicDraw pasiūlyta tokia stereotipo struktūra (23 pav.).



23 pav. Praplėtimo elementas <<VeiklosTaisyklė

Importavus į projektą veiklos taisyklė, anksčiau apibrėžtą pavyzdį (naujo vartotojo registravimo pašto sistemoje) galima būtų modeliuoti veiklos procesų diagrama, kuri parodo kaip yra užregistruojamas vartotojas sistemoje ir kokią informaciją galima gauti apie VT (24 pav.).

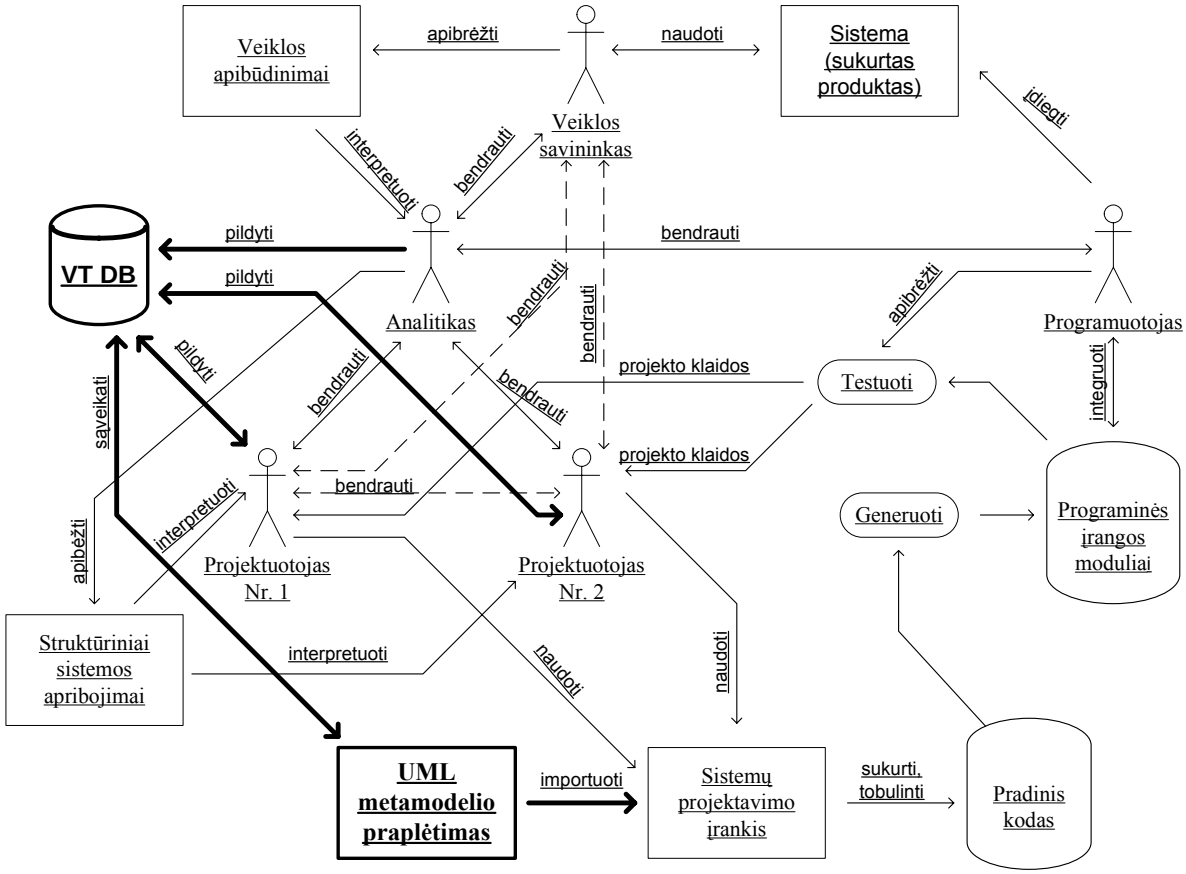


24 pav. Naujo vartotojo registravimo pašto sistemoje veiklos procesų diagrama.

5.2. IŠPLĖSTO ĮRANKIO NAUDOJIMO CHARAKTERISTIKA

Tradiciniai informacijos sistemos projektavimo metodai, didėjant operatyviam sistemos projektavimo poreikiui, nėra pakankamai efektyvūs. Šias problemas padeda spręsti sąlygiškai naujas informacijos sistemų projektavimo būdas pagrįstas veiklą reguliuojančių taisyklių. Kadangi VT taip pat egzistuoja dinaminėje aplinkoje, tai ir ji pati keičiasi reaguodama į aplinką. Dinaminiams veiklos aspektams modeliuoti šiuo metu priimtiniausia objektinė projektavimo kalba. Viena iš populiariausių objektinio požiūrio metodologijų yra UML kalba. Kadangi ji yra nepriklausoma nuo projektavimo ar programavimo platformos, tai ji naudojama įvairių struktūrų sistemų projektavimui. Bet UML savo notacijoje turi mažai būdų verslo procesams reguliuoti įskaitant ir veiklos taisykles. Tradiciškai VT UML kalboje modeliuojamos OCL kalba, kuri paslepia VT modelio notaciją ir tuo pačiu net nepaveikia sistemos logikos. Taisyklės kodo generavimo metu yra įsiuvamos į kodą ir pasikeitus VT reikia pergeneruoti ne tik programos kodą bet ir surasti kurioje vietoje tos taisyklės buvo sumodeliuotos. Tai sudėtingas procesas, ypač jei įvyko sistemos projektuotojų kaita, kuriems reikia daug laiko tam, kad įsigilinti į jau pradėta kurti sistemą. Tai veiklos savininkui sukelia nepatogumus, nes netgi turėdamas savo firmoje kad ir vidutinį projektuotoją turi kreiptis į aukštos kvalifikacijos specialistus ir laukti kol sistema bus sutvarkyta pagal pakitusius poreikius.

Esant dinaminiams rinkos poreikiams, bei aukštam kokybės ir laiko santykiui einama prie komandinio darbo, kai projektuojama sistema yra išskaidoma į modulius ir skiriama modeliuoti keliems projektuotojams. Projektuotojai gali dirbti net ne viename mieste. Tai sukelia VT suderinamumo problemą. Tam kad taisyklės nesidubliuotų bei neprieštarautų viena kitai, projektuotojai turi bendrauti ne tik su analitiku, veiklos sąvininku bet tarpusavyje. Kokybiškos informacijos tikrinimas užima daug laiko, ir norint atlikti darbą laiku, nukenčia sistemos kokybė ir patikimumas. Žemiau parodyta (25 pav.) kaip sistemos kūrėjai bendrauja tarpusavyje, kai esant tokiai situacijai – būtų sukurtas kokybiškas produktas. Ištisinėmis bei punktyrinėmis plonomis linijomis pavaizduota kaip bendrauja sistemos kūrėjai nesant patobulintam programos kūrimo įrankiui MagicDraw.



25 pav. Sistemos tobulinimo schema

Patobulinus sistemą atsiranda pastorintos ištisinės linijos, kurios rodo ryšius tarp naujų elementų, o tuo tarpu ryšiai pažymėti punktyrinėmis linijomis dingsta, nes atsiradus struktūrintai VT duomenų bazei nebelieka būtinybės bendrauti tiesiogiai projektuotojams tarpusavyje, nes apie VT panaudojimą projektuose ir modeliuose galima sužinoti VT DB, kuri yra dinamiškai atnaujinama.

Esant tik dviems projektuotojams, pasinaikina trys ryšiai, jei tai būtų septyni projektuotojai, tai tokiu atveju pasinaikina dvidešimt aštuoni ryšiai. O atsiranda tik dešimt naujų ryšių. Todėl sistema tampa ne tik paprastesne, bet ir žymiai efektyvesne, nes vien bendraujant projektuotojams per VT DB yra taupomas laikas. Esant tokiai darbo metodikai, suprantama yra taupomas laikas, kurį galima skirti sistemos modeliavimui, o ne VT prieštaravimo vienos kitai, ar nesuderinamumo tarpusavyje tikrinimui. Tokiu būdu įmanoma efektyviai sukurti kokybiškas, patikimas sistemas.

5.3. PATOBULINTOS SISTEMOS KOKYBĖS PARAMETRŲ ĮVERTINIMAS IR SISTEMOS TOLIMESNĖS PLĖTOJIMO GALIMYBĖS

Reikalavimų modelyje buvo išskirti šie kokybės parametrai: tinkamumas sprendžiamiems uždaviniams, rezultatų tikslumas, duomenų atnaujinimas ir greitas programų vykdymas.

UML yra notacija kuri nepriklauso nuo jokios platformos, todėl kiekvienas vartotojas ją gali pritaikyti savo poreikiams pasinaudodamas UML išplėtimo galimybe. Bet ne visi CASE įrankiai palaikantys UML metodologiją yra tokie atviri kaip pati UML. MagicDraw UML yra vienas iš tų įrankių, kurie nesuteikia galimybės imų modelyje buvo išskirti šie kokybės parametrai: tinkamumas sprendžiamiems uždaviniams, rezultatų tikslumas, duomenų atnaujinimas ir greitas programų vykdymas.

UML yra notacija kuri nepriklauso nuo jokios platformos, todėl kiekvienas vartotojas ją gali pritaikyti savo poreikiams pasinaudodamas UML išplėtimo galimybe. Bet ne visi CASE įrankiai palaikantys UML metodologiją yra tokie atviri kaip pati UML. MagicDraw UML yra vienas iš tų įrankių, kurie nesuteikia galimybės įtraukti visiškai naują modelio elemento klasę. Šiame darbe naudojamas išplėtimo mechanizmas – stereotipas, kurio visiškai pakako iškeltiems uždaviniams spręsti. Todėl galima teigti, kad funkcionalumas yra dalinai patenkintas.

Rezultatų tikslumas, šiuo atveju, priklauso nuo tikslaus stereotipo reikšmių apribojimo. Tai yra programuotojo kompetencijoje. Šiame darbe nebuvo išvengta klaidų, todėl negalima teigti, kad rezultatai yra tikslūs.

Duomenų atnaujinimas iš DB realaus laiko metu nevyksta, todėl aprėžtas patikimumo kriterijus buvo neįvykdytas, nes tai brangs sistemos realizavimo variantas.

Įvedus stereotipą <<VeiklosTaisykle>> VT modeliavimas pagreitėjo ir palengvėjo, todėl buvo įvykdytas programos efektyvumo kriterijus.

Sistemos plėtojimui galima pasiūlyti kitokią DB struktūrą, kurioje VT būtų saugomas ne šablonais. Išskirti tobulesnę VT klasifikaciją atsižvelgiant į UML modelius.

5.4. SISTEMOS PATOBULINIMO REALIZACIJOS IŠVADOS

1. Darbe buvo pasiūlytas metodas, kaip galima praplėsti MagicDraw modeliavimo galimybes. Metodas paremtas VT taisyklių klasifikacija, jų formalizavimo metodu ir galimybe VT modeliuoti tam tikrais modeliais.

Kadangi sistemų projektavimo įrankis MagicDraw UML išplėtimo galimybes, buvo sukurtas stereotipas veiklos taisyklėms modeliuoti. Stereotipas susietas su VT duomenų baze, kurioje VT struktūrinimui panaudoti šablonai. Pateiktas pavyzdys, kuris parodo kaip galima taisykles modeliuoti veiklos procesų diagrama.

2. Pateikta schema, kuri atspindi kokią įtaką daro naujųjų elementų įvedimas į sistemų kūrimo etapą. Įvedus VT DB ir UML praplėtimo mechanizmą, atsiranda galimybė tiksliau, efektyviau ir kokybiškiau projektuoti sistemą, išvengiant papildomų laiko sąnaudų, reikalingų dubliuojančių bei prieštaraujančių taisyklių paieškai.

IŠVADOS

1. Darbe atlikta VT klasifikavimo būdų daugiaaspektinė analizė, atskleidžiant naudojamų klasifikavimo principų panašumus ir skirtumus. Susipažinta su įvairių autorių ir įvairių šaltinių siūloma veiklos taisyklių klasifikacija. Analizuota literatūra apie veiklos taisyklių formalizavimo metodus – šablonus ir tų metodų formalizavimo kalbas.
2. Apžvelgta OCL kalba, kuri tinka veiklos taisyklių formalizavimui bei yra sudėtinė UML notacijos dalis. Nors OCL nėra programavimo kalba, bet jos pagalba grafiškai galima įtraukti veiklos taisykles į UML modelius. Darbo pradžioje visas dėmesys buvo kreipiamas į šį modeliavimo būdą, bet šio metodo buvo atsisakyta, nes VT paslepiamos programos kode, prarandant galimybę greitai ir efektyviai modeliuoti veiklos pasikeitimus.
3. Buvo analizuojama kokie UML modeliai tinka biznio procesams modeliuoti ir pagal kokią taisyklių klasifikaciją galima įtraukti veiklos taisykles į šiuos modelius. Išskirta VT klasifikacija projektuotojo požiūriu ir šiai klasifikacijai pritaikyti šablonai. Remiantis klasifikacija projektuotojo požiūriu nustatyti UML modeliai, šiai klasifikacijai projektuoti.
4. Apžvelgtos UML meta modelio praplėtimo galimybės VT modeliuoti. Išskirtos UML metamodelio praplėtimo savybės ir pagal pasirinkto modeliavimo įrankio galimybes pasirinktas UML meta modelio išplėtimas stereotipo klase.
5. VT specifikacijai saugoti siūloma sukurti autonomišką VT saugyklą.
6. Nagrinėtos literatūroje pateiktos galimos sistemos architektūros, bei pristatytas tobulinamos sistemos architektūros modelis, bei pasiūlyta loginė duomenų bazės struktūra saugoti VT.
7. Analizėje pasirinkti metodai realizuoti sistemų projektavimo įrankiu MagicDraw. Remiantis analize pasiūlytas metodas kaip galima praplėsti MagicDraw modeliavimo galimybes. Metodas paremtas VT taisyklių klasifikacija, jų formalizavimo metodu ir galimybe VT modeliuoti konkrečiais modeliais – biznio objektų, būsenų, veiklos, sekų ir klasių diagramomis. Kadangi sistemų projektavimo įrankis MagicDraw leidžia plėsti išplėtimo galimybes, buvo sukurtas Java kalba stereotipas veiklos taisyklėms modeliuoti. Stereotipas susietas su VT duomenų baze,

kurioje VT struktūrinimui panaudoti šablonai. VT duomenų bazė sukurta MS Access duomenų bazių kūrimo įrankiu ir su stereotipu sujungta per ODBC draiverius.

8. Darbe pateikta schema, iliustruojanti pridėtų naujų komponentų įtaką sistemų kūrimo etapui. Įvedus VT DB ir UML praplėtimo mechanizmą, atsiranda galimybė tiksliau, greičiau ir kokybiškiau projektuoti sistemą, išvengiant papildomų laiko sąnaudų, reikalingų dubliuojančių bei prieštaraujančių taisyklių paieškai. Visos taisyklės patalpintos vienoje vietoje, tai paspartina ir supaprastina jų modeliavimą, nes taikoma vientisa struktūra ir notacija, bei tokiu būdu kokybiškai kuriama sistema.

NAUDOTA LITERATŪRA

[1] About Blaze Advisor Rule Services, http://www.kpiusa.com/brbook/Blaze.htm (žr. 2003.09.30)

[2] Business Rules Group, Business Rules Manifesto vs. 1.2, Edited by Ronald G. Ross, 2003, http://www.businessrulesgroup.org/brmanifesto.htm (žr. 2003.09.30)

[3] Hay D. C. What Data Models can't do, http://www.essentialstrategies.com/documents/brules.pdf (žr. 2003.09.30)

[4] Object Constraint Language Specification, http://www-3.ibm.com/software/ad/library/standards/ocl.html (žr. 2003.11.10)

[5] OCL Compiler, http://dresden-ocl.sourceforge.net/index.html (žr. 2003.11.10)

[6] OMG Unified Modeling Language Specifikation vs. 1.5, 2003.

[7] Response to the UML 2.0 OCL RfP Revised Submission, vs. 1.6, 2003

[8] Rumbaugh J., Blaha M., Premerlani W., .Eddy F., Lorensen W. Object-Oriented Modeling and Design. Prentice Hall, Englewood Cliffs, 1991.

[9] The Business Rules Group, Final Report, ver. 1.3, http://www.businessrulesgroup.org/first paper/BRG-whatisBR 3ed.pdf (žr. 2003.09.30)

[10] The Object Constraint Language, http://www-3.ibm.com/software/ad/library/standards/ocl.html (žr. 2003.11.10)

[11] Petrulytė R., Nemuraitė L. Veiklos taisyklių valdymas informacinėse sistemose. Informacinės ir telekomunikacinės technologijos: VII tarpuniversitetinė magistrantų ir doktorantų konferencija. Kaunas: Technologija, 2002: pp. 113-116.

[12] Ross R. G. Business – Oriented Business Rule Analysis and Management, http://www.omg.org/docs/ad/02-07-07.pdf (žr. 2004.01.23)

[13] UI Tax and Benefits Systems Modernization – Business Rules. Information Technology Support Center.http://www.itsc.state.md.us/PDF/M-1-1%20rev091401.pdf (žr. 2004.01.23)

[14] Martin J., Odell M. Rules. Object – Oriented Methods: a Foundation. Prentice-Hall, 1995, 412 p.

[15] Gottesdiener E. Business Rules show Power, Promise.- www.addtmag.com/pub/mar97/softeng.html (žr. 2003.10.09)

[16] Gudas S., Skersys T. Veiklos taisyklių integravimo kompiuterizuotoje IS inžinerijoje būdas. Informacinės Technologijos‘2003, Kaunas, Technologija, 2003: pp. VI-41-45.

[17] Ažubalis K., Nemuraitė L. Veiklos taisyklių modeliavimas objektiškai orientuotuose metoduose. Informacinės technologijos’99, Kaunas, Technologija, 1999: pp. 171 – 178.

[18] Reeder J. Templates for Capturing Business Rules. Business Rules Community - http://www.brcommunity.com/p-b056.php (žr. 2004.05.26)

[19] Haggerty N. Modeling Business Rules Using the UML and CASE, http://www.brcommunity.com/p-b016.php (žr. 2004.04.20)

[20] Erisson H. E., Penker M. Business Modeling with UML: Business Patterns at Work. John Wiley & Sons, 2000, 260 p.

[21] Wilson D. Business Rules, Platforms, and Inferencing. Business Rules Community - http://www.brcommunity.com/p-b169.php (žr. 2004.04.20)

[22] Morgan T. Business Rules and Information Systems. Addison Wesley, 2003

[23] Blažytė I., Butleris R. Veiklos taisyklių modeliavimo UML kalbos pagrindu analizė. Informacinės technologijos verslui – 2004. Kaunas, Technologija, 2004: pp. 37-43.

THE RESEARCH OF BUSINESS RULES MODELING ON THE BASIS OF THE LANGUAGE UML (UNIFIED MODELING LANGUAGE)

SUMMARY

The main aim of the system's analysis is to find the most important information about real world (problematic sphere); and, firstly, that will be data about processes, their structure and data shift. Business rules combines all these three things that were mentioned above, that is why they can be regarded as the central part in the system's analysis, and the system's analysis can be applied to business rules.

Business rule is a proposition that defines and limits some aspects of the business. Certain business structure is defined by the business rule; additionally, certain business is controlled or influenced by the business rule too. Moreover, business rules do not depend on projecting model or technical platform. Objective application informational system creation methodology was chosen for the analysis; and it was based on objective modelling of the real world objects and the use of that model for the projecting that does not depend on certain programming language. UML language was chosen for business rules modelling because this language is universal of conception modelling and it is an objective programming language.

Despite of all advantages of UML, it does not specify business rules as a separate model's unit. In UML the rules are depicted as comment; in the beginning they are written by the help of natural speech then they are made into OCL (object constrain language). OCL is not a programming language that is why the logics and control of the program cannot be written by the means of OCL. So, the business rules of OCL are hidden in the code of the program and they are not seen for the system's user. In this way, when the business rules are changed so the program code should be changed too. The task of this research is to find out if it is possible to project business rules as single units by the help of UML.

In this work the structure of business rules was analyzed, some classifications and characteristics were made, and some levels of aspects were defined. Besides, the means of formalization of business rules were introduced, UML language was overviewed and its expansion opportunities too. After business rules analysis, the classification of projector attitudes were made; and for this classification modelling were set certain projecting models; and the model of realization was suggested by the help of system's designing tool MagicDraw UML.

VILNIAUS UNIVERSITETAS
KAUNO HUMANITARINIS FAKULTETAS

INFORMACINĖS TECHNOLOGIJOS VERSLUI – 2004

Konferencijos pranešimų medžiaga

Information Technologies for Business – 2004

Proceedings of the Annual Conference

Kaunas * Technologija * 2004

TURINYS

VEIKLOS TAISYKLIŲ MODELIAVIMO UML KALBOS PAGRINDU ANALIZĖ

Inga Blažytė[1], Rimantas Butleris[1,2]

[1]Kauno technologijos universitetas, [2]Vilniaus universitetas, Kauno humanitarinis fakultetas

Darbe atliekama veiklos taisyklių apžvalga ir pateikiama jų klasifikacija. Nagrinėjama UML kalba, bandant nustatyti kokie modeliai yra tinkami veiklos taisyklėms atvaizduoti. Išskiriama OCL kalba, kurios pagalba atliekamas veiklos taisyklių formalizuotas specifikavimas. Pagal veiklos taisyklių klasifikaciją ir UML kalbos tyrimo rezultatus siūlomas veiklos taisyklių konceptualus modelis, kurį numatoma realizuoti sistemų projektavimo įrankiu MagicDraw.

1. Įvadas

Pagrindinis sistemų analizės tikslas yra surinkti pakankamai informacijos apie realų pasaulį (probleminę sritį) – tai pirmiausia duomenys apie procesus, jų struktūrą ir duomenų kaitą. Veiklos taisyklės (VT) apima visus tris paminėtus dalykus, todėl jas galima laikyti centriniu objektu sistemų analizės kontekste ir sistemų analizės procesą pritaikyti prie veiklos taisyklių sklandžios specifikavimo eigos.

Daugelis veiklos procesų vykdomi, atsižvelgiant į taisykles, kurios apsprendžia tam tikrus veiksmus arba riboja galimų veiksmų aibę. Veiklos taisyklės vaidina didelį vaidmenį veiklos objektų architektūroje ir komponentų projektavimo metoduose – naujoje objektiškai orientuotų metodų kryptyje. Vienas iš labiausiai paplitusių objektiškai orientuotų metodų yra UML (Unified Modeling Language).

Informacinių sistemų (IS) projektavimas UML kalba yra lengvai suprantamas tiek vartotojui, tiek programuotojui. Bet UML kalboje ne visos diagramos yra skirtos veiklos procesams modeliuoti. Todėl yra svarbu, jei įmanoma, praplėsti tuos modelius ir pritaikyti veiklos procesams modeliuoti.

Pirmieji moksliniai straipsniai apie VT pradėti publikuoti apie 1990-uosius metus. Tuomet buvo pastebėta, kad, siekiant pilnai aprašyti organizaciją, tradicinių į procesus ir duomenis orientuotų analizės metodų nebeužtenka, o programuotojams paliekama per daug laisvės improvizacijai, programiniu kodu užpildant neišanalizuotus organizacijos aspektus.

VT dabar naudojamos tik nedaugelyje kompiuterizuoto IS projektavimo (CASE) įrankių. Šie įrankiai dažniausiai apsiriboja paprasčiausiomis struktūrinėmis taisyklėmis duomenų modelių struktūroms aprašyti (tokios taisyklės pateikiamos kaip grafinės modelių notacijos dalis). Jei naudojamos ir kitų tipų VT, jos paprastai saugomos tik kaip paprastas tekstas modelio objektų komentaro laukeliuose.

2. Veiklos taisyklės

VT yra teiginys, kuris apibrėžia ar apriboja tam tikrus veiklos aspektus. VT nusakoma tam tikros veiklos struktūra, kontroliuojama arba įtakojama tam tikra veikla.

Kai VT nusakoma veiklos dalyvio, tai ji dažniausiai įvardijama dviprasmiškai ir negriežtai. Dažnai VT galima išskaidyti į detalesnius VT atvejus. Išskaidant VT iki savo elementariausios formos, jos dalomos tol, kol apsiribojama viena išbaigta mintimi. VT nemodeliuojama sukurtos programinės įrangos sistemos valdymo logika, tipiškai randama programos logikoje. Elementari VT, parašyta deklaratyviu būdu, naudojant standartinę natūralios kalbos gramatiką, kurią veiklos dalyviai lengvai supranta, nėra dviprasmiška.

VT yra nepriklausoma nuo projektavimo modelio ar techninės platformos. Taigi, veiklos taisyklės yra [1]: 1. Deklaratyvios; 2. Elementarios; 3. Išreikštos natūralia kalba; 4. Savarankiškos, nepriklausomos loginės struktūros; 5. Orientuotos į veiklą, o ne į technologiją; 6. Priklausančios veiklai, o ne technologijai.

3. Veiklos taisyklių klasifikavimas

Nėra vieningos nuomonės įvairiais klausimais, susijusiais su taisyklių identifikavimu, atvaizdavimu, jų integravimu IS kūrimo procese, diegimu taikomosiose programose ir kt. Nėra vieningo VT apibrėžimo, notacijos bei jų klasifikacijos.

Organizacija „Business Rule Group" (dar žinoma kaip „GUIDE Business Rule Project") pasiūlė tokį veiklos taisyklių apibrėžimą: „Veiklos taisyklė – tai teiginys, kuris apibrėžia arba sąlygoja tam tikrą veiklos aspektą" [1]. 1 lentelėje pateikta VT klasifikacija įvairių šaltinių pagrindu.

1 lentelė. Veiklos taisyklių klasifikacija [3]

Šaltinis	Taisyklių klasifikacija
Business Rules Group (Guide Business Rules Project) [1, 3]	Struktūrinės taisyklės (terminai, faktai), veiksmo taisyklės (integralumo apribojimai, sąlygos, autorizacija), išvestys (skaičiavimai, loginės išvados)
Ronald Ross, Database Research Group [2, 3]	Terminai, faktai (ryšių tipai, subtipai, atributai), taisyklės (skaičiavimai, atmetimo taisyklės, projekcijos)
Tom Romeo, IBM [3]	Struktūrinės taisyklės (ryšiai, domenai, kardinalumas), elgesio taisyklės (prieš-sąlygos, po-sąlygos, išvestys)
Margaret Thorpe, Tangram [3]	Apibrėžimai, integralumo taisyklės, bendrieji deklaratyvūs apribojimai, procedūriniai apribojimai, išvestys
Barbara von Halle, Knowledge Partners [4]	Apibrėžimai, faktai, apribojimai, išvestys, loginės išvados
Usoft Corporation [5]	Apribojimai, išvestys, elgesio taisyklės, atvaizdavimo taisyklės
Brightware [3]	Veikla, strategija, darbų sekos, sprendimų euristika
Vision Software [6]	Validavimo taisyklės, išvestys, ryšiai, sąlygos veiksmai

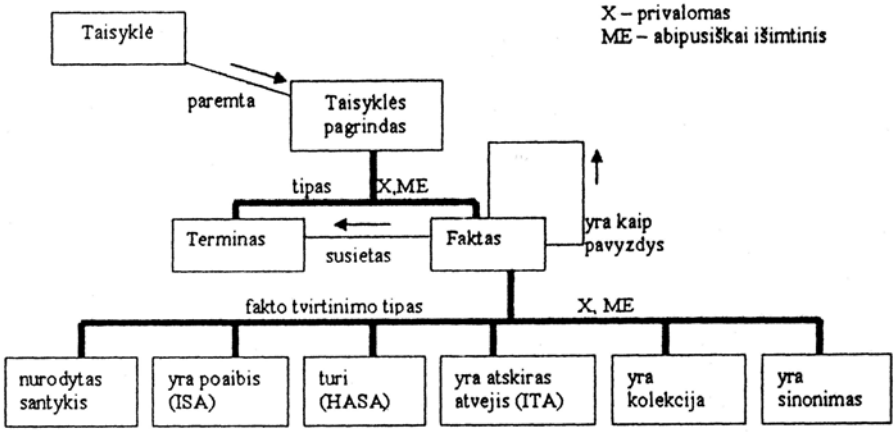
Daugelis VT srityje dirbančių organizacijų ir mokslininkų kuria savitas VT klasifikacijas, priklausomai nuo iškeltų darbo tikslų, probleminės srities specifikos.

Viena dažniausiai naudojamų VT klasifikacijų yra sudaryta organizacijos „Business Rules Group" (BRG). Pagal šią klasifikaciją taisyklė gali [1]:

1. Apibrėžti terminą; 2. Sujungti terminus į faktus (*faktai atspindi terminų tarpusavio priklausomybes, kas statinės struktūros modelyje vaizduojama ryšiu tarp esybių)*; 3. Pateikti skaičiavimų rezultatus; 4. Patikrinti sąlygas naujo fakto sukūrimui; 5. Patikrinti sąlygas veiksmui inicijuoti.

Trys paskutiniosios kategorijos apibrėžia tikrąsias VT, o terminai ir faktai turėtų būti interpretuojami kaip struktūrinės taisyklės, kurių didžiąją dalį galima atvaizduoti ir tradicinėmis duomenų modelių grafinėmis notacijomis.

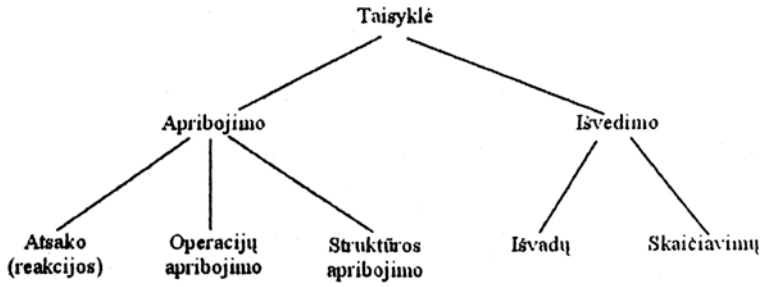
Paveiksle (1 pav.) parodoma veiklos taisyklių terminų ir faktų sąsaja [7].



1 pav. Veiklos taisyklės struktūra (terminai ir faktai)

Nepaisant veiklos taisyklių klasifikacijos įvairovės, nemaža jų dalis gali būti aprašoma klasikiniais produkcinių taisyklių (JEI – TAI) formatais, leidžiančiais modeliuoti priežastinius-pasekminius santykius. Tokiu formatu užrašytomis taisyklėmis galima įvertinti objektų savybes ir iššaukti atitinkamus veiksmus. Veiksmai gali modifikuoti veiklos objektų savybes, sukurti naujus veiklos objektus, iššaukti metodus arba kitų taisyklių vykdymą [8].

Norint integruoti veiklos taisykles organizacijos veiklos modelyje, tikslinga pateikti jam adekvačią taisyklių klasifikaciją. Žemiau pateikta klasifikacija (2 pav.) atitinka sistemos analitiko bei projektuotojo požiūrį, užtikrinantį VT integravimą į sistemą [9].



2 pav. Taisyklių klasifikacija kompiuterizuotos sistemos kūrėjų požiūriu

Atsako (reakcijos) taisyklės (Stimulus and Responce Rules) apriboja elgesį, nusako sąlygas, prie kurių galima vykdyti operaciją. Atsako taisyklės priklausomos nuo konteksto. Taisyklės JEIGU sąlyga tikrinama tik tada, KAI įvyksta tam tikro tipo įvykis. Tokios taisyklės sutinkamos, pvz., darbų sekų ir būsenų perėjimo diagramose. Kartais jos vadinamos ECA (Event – Condition - Action) taisyklėmis [9, 12].

Operacijų apribojimo taisyklės (Operation Constraint Rules) nurodo, kokios sąlygos turi būti tenkinamos prieš operaciją ir po jos, kad būtų užtikrintas vykdymo teisingumas. Tokie apribojimai svarbūs operacijos vykdymui ir nepriklauso nuo operacijos sužadinimo konteksto. Į šias taisykles galima žiūrėti kaip į kontraktą, siejantį operaciją su jos iškvietėjais. Tai galima užrašyti taip: „Jei iškviesi operaciją, kai įvykdyta prieš-sąlyga (precondition), gausi galutinę būseną, kurioje bus patenkinta po-sąlyga (postcondition)"[12]. Operacijos prieš-sąlygos išreiškia apribojimus, prie kurių operacija atliekama teisingai [9].

Struktūros apribojimo taisyklės (Structure Constraint Rules) aprašo objektų ir jų sąryšių sąlygas, kurių negalima pažeisti. Taisyklė gali apriboti atributo reikšmes, objekto tipo egzempliorius, ryšio kardinalumą. Struktūros apribojimo taisyklės nedaro nuorodų į operacijas, nes jos turi būti tenkinamos bet kuriomis veikimo aplinkybėmis [9, 12].

Išvadų taisyklės (Inferences) aprašo, kokias išvadas galima padaryti iš tam tikrų faktų. Tokios taisyklės susiję su ekspertinėmis sistemomis, naudojančiomis produkcinių taisyklių žinių atvaizdavimo modelius [9,12].

Skaičiavimų taisyklės (Mathematical Computations) apibrėžia veiksmus ir algoritmus, kuriuos vykdant galima gauti rezultatus. Jos išreiškiamos formulėmis [9, 12].

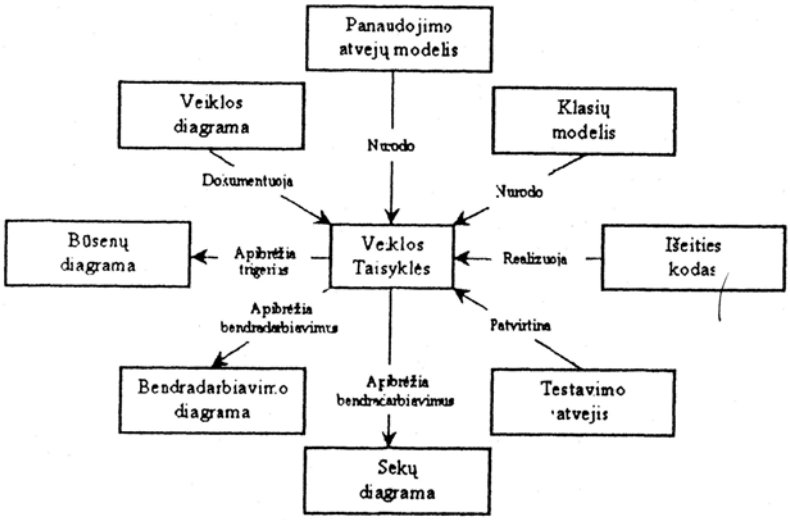
Skirtingos taisyklės yra modeliuojamos skirtingose diagramose. Vieno tipo taisyklė gali būti modeliuojama keliuose modeliuose, bet ne visi modeliai tinka veiklos procesams modeliuoti. Žemiau pateikta 3 lentelė, kurioje parodyta kokiais UML modeliais aukščiau apibrėžtos taisyklės yra modeliuojamos [7].

3 lentelė. Veiklos taisyklių klasifikacija UML modeliuose

Veiklos taisyklės tipas	Išraiška	UML diagrama
Struktūrinės (Structural Assertions): - Terminai - Faktai	 *Is, Has, Must have, Must not have*	 Veiklos Objektų diagramos Veiklos Objektų diagramos
Apribojimo taisyklės (Constraint Rules): - Atsako (reakcijos) taisyklės - Struktūros apribojimo taisyklės - Operacijų apribojimo taisyklės	 *When/ If/ Then* *It must always hold that* *Only If*	 Būsenų, Veiklos diagramos Veiklos Objektų, Klasių diagramos Veiklos diagramos
Išvedimo taisyklės (Derivations): - Skaičiavimų taisyklės - Išvadų taisyklės	 *Is Computed As* *If and Only If*	 Veiklos diagramos, Klasių diagramos (metodai) Veiklos diagramos

4. Veiklos taisyklių modeliavimas UML kalba

UML projektuotojams pateikia aibę notacijų ir semantikų informacinių sistemų statiniams, dinaminiams ir elgesio aspektams modeliuoti. Pavyzdžiui UML palaiko penkias notacijas sistemos dinamikai modeliuoti. Dvi iš jų yra veiklos ir būsenų diagramos. Naudojant veiklos diagramą galima modeliuoti darbų sekas, kurios pakeičia sudėtinių veiklos objektų, tokių kaip draudimo ieškiniai, banko sąskaitos, statusą. Naudojant būsenų diagramą galima pavaizduoti kaip darbai sudėtiniame darbų sekų modelyje keičia atskirų atvejų, draudimo ieškinių ar bankų sąskaitų, būseną. Šios kelių dimensijų panaudojimo perspektyvos leidžia apibūdinti sistemą iš skirtingų požiūrių ir padeda atvaizduoti šiuolaikinių sistemų sudėtingas savybes [10]. Žemiau pateiktoje diagramoje parodoma sąsaja tarp VT ir UML modelių (3 pav.).



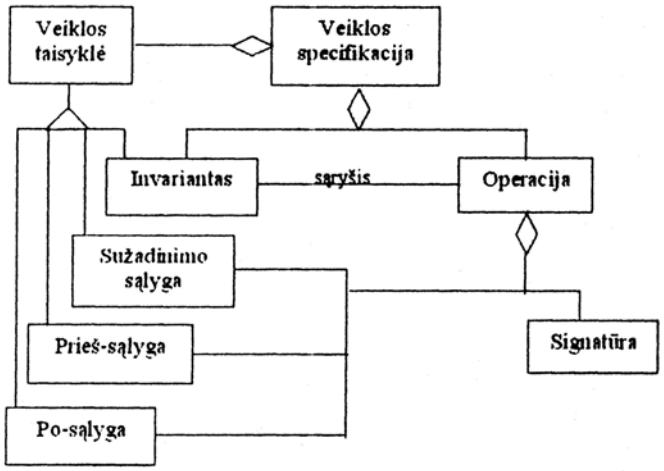
3 pav. Ryšiai tarp VT ir UML modelių

Negalima teigti, kad UML neleidžia modeliuoti VT. Yra daug būdų pavaizduoti VT UML kalboje [10]: 1. Grafiškai, naudojant UML modelių semantiką - asociaciją ir apibendrinimą objektų modeliuose. 2. Grafiškai, naudojant praplėtimą, vadinamą apribojimu (constraint). Pavyzdžiui apribojimai leidžia tiksliau specifikuoti struktūrines veiklos taisykles. 3. Grafiškai, naudojant pastabą (note), susietą su kitu modelio elementu, pavyzdžiui, asociacija ar įvykiu. 4. Naudojant predikatų logiką objektų apribojimo kalboje (OCL). 5. Apibrėžiant veiklos taisykles kaip kitų artefaktų savybes. Pavyzdžiui, apsaugos sąlygos perėjimuose būsenų modeliuose.

Problema yra ta, kad UML nespecifikuoja VT kaip atskiro elemento modelių kūrimo bloke – unikalios modeliavimo elemento klasės.

5. Veiklos taisyklių modeliavimas objektų apribojimo kalba (OCL)

Organizacijos veiklą galima pilnai specifikuoti taisyklėmis. Iš vienos pusės, veiklą galima specifikuoti kaip veiklos taisyklių rinkinį, iš kitos pusės, kaip operacijų ir su jomis susijusių invariantų rinkinį. Operacijos modelį sudaro jos signatūra, prieš-sąlygų, po-sąlygų ir, galbūt, kitokių taisyklių rinkinys (4 pav.) [11].



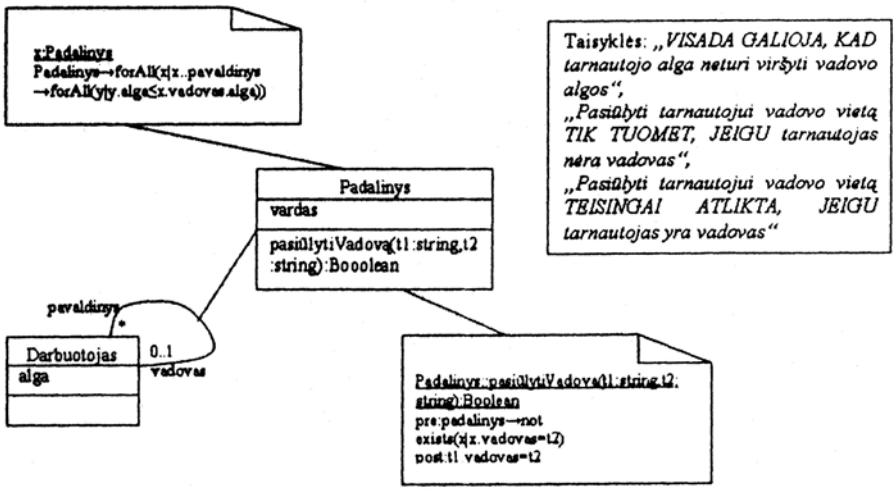
4 pav. Veikos specifikavimas taisyklėmis

Skaičiavimo sąlygos veiklos modelyje neaprašomos, kadangi veiklos modelis turi atsakyti į klausimą „ką?“, o ne „kaip“. Toks principas taikomas dėl to, kad veiklos modelis, kaip ir kompiuterizavimo projektas, turi būti nepriklausomas nuo realizacijos. Tačiau skaičiavimų ar išvedimo modelis gali atsirasti vėlesnėse, realizavimo stadijose. Pateiktąjį modelį galima atvaizduoti bet kuria formalia modeliavimo kalba.

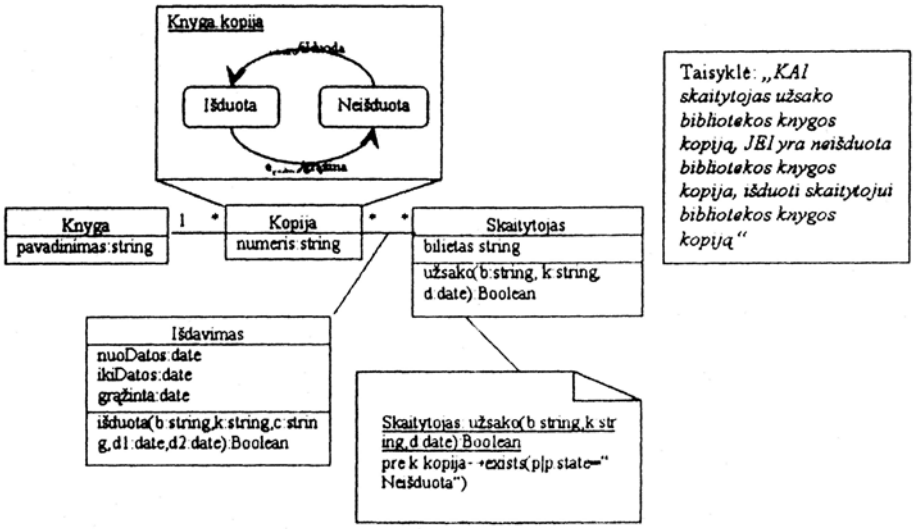
UML taisyklės vaizduojamos kaip pastabos, pradžioje aprašomos natūralia kalba, po to verčiamos į OCL. Pastaba grafiškai vaizduojama specialia piktograma, kuri susiejama su kitais diagramų simboliais.

Modeliuojant veiklos taisykles, OCL galima panaudoti invariantų specifikavimui, prieš ir po sąlygų specifikavimui, navigavimui. UML apibrėžime OCL panaudota UML semantikai aprašyti. OCL papildo UML,

įgalindama modeliuoti tiksliai. Be to, ji nėra labai sudėtinga, prieinama vidutiniam modeliuotojui. OCL išraiškomis galima papildyti klasių (5 pav.), būsenų (6 pav.) ir kitas UML diagramas [12].



5 pav. UML klasių diagrama [12]



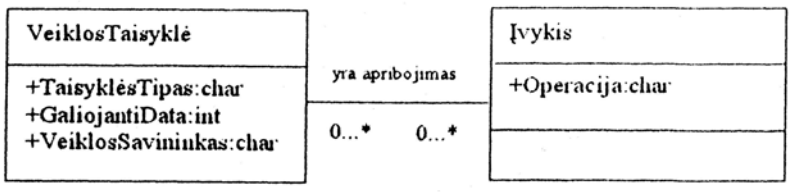
6 pav. UML klasių ir būsenų diagrama[12]

6. UML modifikavimas veiklos taisyklėms modeliuoti

UML nespecifikuoja VT kaip atskiro modelio elemento, todėl tai nėra geras VT modeliavimo būdas. Kad UML notacija būtų tinkama VT modeliavimui ir valdymui, reikia įtraukti atskirą modeliavimo elementą, pavadintą *VeiklosTaisyklė*. Tuo tikslu reikia praplėsti UML meta modelį pridedant VT klasę su jos savybėmis. Tada reikia specifikuoti asociacijas su kitais modeliavimo elementais.

UML siūlo tris praplėtimo mechanizmus. Tai stereotipai, prijungtos reikšmės ir apribojimai.

Stereotipai leidžia sukurti naują UML modelio kūrimo bloką. Žemiau yra pateiktas pavyzdys (7 pav.), kuriame yra parodyta, kaip galima pridėti naują metodo stereotipą, pavadintą *VeiklosTaisyklė* [10].



7 pav. Atskiras elementas VeiklosTaisyklė

Tolesniam šio proceso vykdymui reikia veiklos taisyklių saugyklos, kurioje yra saugomos ir apdorojamos veiklos taisyklės. Gerai sudaryta veiklos taisyklių saugykla leidžia praplėsti veiklos taisyklių modeliavimo galimybes UML'e.

7. Veiklos taisyklių saugykla

Kadangi UML neužtikrina kelių projektuotojų sukurtų VT specifikacijų darnos, tai gali vesti prie prieštaravimo skirtinguose modeliuose ar sistemos specifikacijos fragmentuose. Norint išvengti tokių prieštaravimų reikalinga VT saugykla – duomenų bazė, kurioje užfiksuojamos, saugomos ir valdomos VT. VT saugykla – tai autonomiška tam tikru būdu aprašytų taisyklių bazė.

VT saugykla yra skirta vadybininkams, analitikams ir programuotojams jas apibrėžti, bendrai naudotis jų specifikacija ir parengti kompanijos VT bazę. VT saugykla suteikia technologines galimybes jas paskirstyti pritaikant įvairiems funkciniams uždaviniams, vystyti jų aprašą bei valdyti centralizuotu būdu [13].

Veiklos taisyklių saugyklų privalumai:

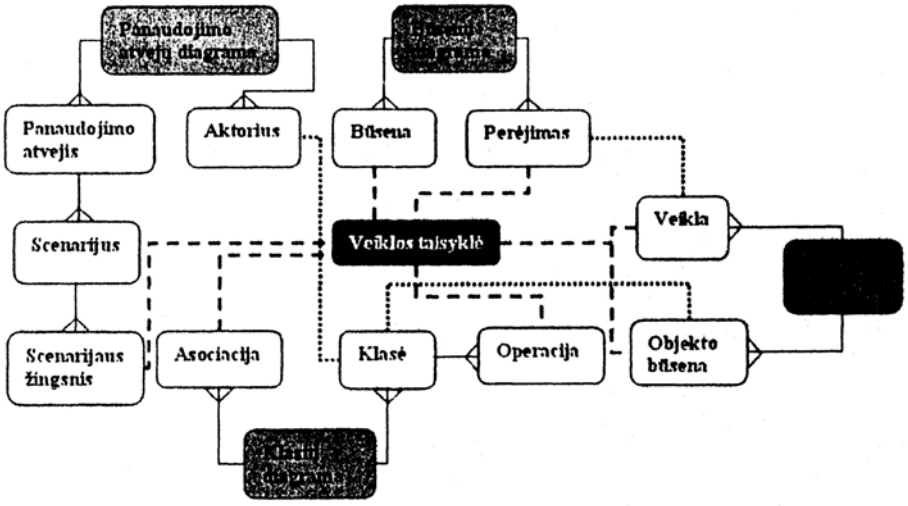
1. Leidžia lengvai modifikuoti VT; 2. Išvengiama taisyklių dubliavimo ir prieštaravimų; 3. Leidžia prieigą prie saugyklos taisyklėms specifikuoti ir pakeisti; 4. Leidžia sistemos realizavimo elementams greitai surasti bei pakeisti kiekvieną VT, atsakant į veiklos iniciatyvos pasikeitimą; 5. Leidžia spausdinti modelių ir VT apibrėžimus, todėl dalykinės srities ekspertai gali jas peržiūrėti.

Reikalavimai idealiai VT saugyklai:

1. Viena integruota duomenų bazė, skirta UML modelių ir VT saugojimui, valdymui ir pakartotiniam panaudojimui. 2. Papildomas meta modelis. 3. Ataskaitų ir užklausų redaktorius. 4. Publikavimo Internete galimybės. 5. Duomenų valdymo funkcijos veiklos taisyklių ir modelių integralumui ir saugumui. 6. Galimybė bendrauti (integracijos požiūriu) su kitais projektavimo įrankiais, pavyzdžiui, skirtais veiklos taisyklių manipuliavimui ir testavimui.

8. Meta modelio papildymas veiklos taisyklių klase

Daugumos CASE įrankių specifikacijų saugyklos palaiko UML meta modelį. Jei CASE paketo saugyklos meta modelis yra išplečiamas, tuomet galima sustiprinti įrankio UML palaikymą, pritaikant UML įrankio meta modelį tokioms veiklos taisyklių modeliavimo galimybėms, kokių reikia. Galima pridėti klasę *VeiklosTaisyklė*, specifikuoti jos savybes ir tuomet nustatyti asociacijas su kitais UML modeliavimo elementais. Paveiksle (8 pav.) pavaizduotas UML meta modelio fragmentas su pridėta VT esybe ir jos ryšiais (punktyrinės linijos). Reikia pažymėti, kad čia parodyti ne visi ryšiai tarp UML modelio artefaktų (taškinės linijos), kad diagrama nepavirstų į ryšių raizginį [10].



8 pav. UML meta modelis papildytas esybe VeiklosTaisyklė

9. Išvados

Veiklos taisyklės yra neatsiejama veiklos procesų dalis, todėl joms yra skiriamas nemažas dėmesys kuriant taikomąsias sistemas bei jų automatizuoto projektavimo priemones. Darbe atlikta VT klasifikavimo būdų daugiaaspektinė analizė, atskleidžiant naudojamų klasifikavimo principų panašumus ir skirtumus. Išnagrinėtos UML galimybės modeliuoti VT objektų apribojimo kalbos OCL pagrindu. Siekiant VT modeliuoti autonomiškai numatoma praplėsti UML kalbos meta modelį VT klase. VT specifikacijai saugoti siūloma sukurti autonomišką VT saugyklą. UML kalbos praplėtimui VT stereotipu numatoma realizuoti sistemų projektavimo įrankio MagicDraw pagrindu.

10. Literatūra

1. The Business Rules Group, Final Report, ver. 1.3, http://www.businessrulesgroup.org/firstpaper/BRG-whatisBR 3ed.pdf (žr. 2003.09.30)
2. Hay, D. C. What Data Models Can't Do, http://www.essentialstrategies.com/documents/brules.pdf (žr. 2003.09.30)
3. Gudas, S., Skersys, T. Veiklos taisyklių integravimo kompiuterizuotoje IS inžinerijoje būdas. Informacinės Technologijos'2003, Kaunas, Technologija, 2003: pp. VI-41-45.
4. About Blaze Advisor Rule Services, http://www.kpiusa.com/brbook/Blaze.htm (žr. 2003.09.30)
5. Business Rules Today: A KPI Position Paper, http://www.kpiusa.com/ReadingRoom/BusinessRulesToday.htm (žr. 2003.09.30)
6. Business Rules - Bizagi, http://www.visionsoftware.biz/resources/BusinessRules.htm (žr. 2003.09.30)
7. Ross, R. G. Business – Oriented Business Rule Analysis and Management, http://www.omg.org/docs/ad/02-07-07.pdf (žr. 2004.01.23)
8. About Blaze Advisor Rule Services, http://www.kpiusa.com/brbook/Blaze.htm (žr. 2003.09.30)
9. Martin, J., Odell, M. Rules. Object – Oriented Methods: a Foundation. Prentice-Hall, 1995, 412 p.
10. Haggerty, N. Modeling Business Rules Using the UML and CASE, http://www.brcommunity.com/p-b016.php (žr. 2004.04.20)
11. Gottesdiener, E. Business Rules show Power, Promise.- http://www.addtmag.com/pub/mar97/softeng.html (žr. 2003.10.09)
12. Ažubalis, K., Nemuraitė, L. Veiklos taisyklių modeliavimas objektiškai orientuotuose metoduose. Informacinės technologijos'99, Kaunas, Technologija, 1999: pp. 171 – 178.
13. UI Tax and Benefits Systems Modernization – Business Rules. Information Technology Support Center, http://www.itsc.state.md.us/PDF/M-1-1%20rev091401.pdf (žr. 2004.01.23)

Analysis of Business Rules Modelling using UML

In this work business rules are reviewed and the classification of them is presented from different sources. In this paper UML language is researched in the case to find out what models are appropriate to represent business rules. OCL language is segregated and the help of it makes the formalisation of business rules. Conceptual model of business rules is suggested according to the classification of business rules and the results of UML language researching; and also the realisation of this conceptual model will be done by the CASE tool MagicDraw.