KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Agnė Ručinskaitė

Veiklos taisyklių specifikavimo šablonais metodika ir jų manipuliavimo tyrimas

Magistro tezės

Darbo vadovas: prof. dr. Rimantas Butleris
Konsultantė: dokt. Lina Tutkutė

Kaunas 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Agnė Ručinskaitė

Veiklos taisyklių specifikavimo šablonais metodika ir jų manipuliavimo tyrimas

Magistro tezės

Vadovas:

Prof.dr. Rimantas Butleris

2009-05-25

Konsultantė:

dokt. Lina Tutkutė

Recenzentė:

Doc. Dr. Regina Misevičienė

2009-05-25

Atliko:

IFM-3/2 gr. stud.

Agnė Ručinskaitė

2009-05-25

Kaunas 2009

Turinys

Lentelės

Paveikslėliai

SUMMARY

Templates based Business Rules Specification Methods and Manipulation Research

In this work there is analyzed business rules` conception, ways of their classification and possibilities to use them in order to realize different needs, how business rules are specified and how you can manipulate them. The main aim of this work is to explore rules Blaze Advisor, to analyze business rules specification principles, by using objectively oriented SRL language and opportunities to use them on the ground of inner advisor components designed for business rules realization. After the detailed analysis it was noticed that there are defects, which don`t let effectively model business rules and narrow the access enterprises representatives to them, trying to eliminate the defects by suggested methods, which would present a more flexible way to create business rules and guide, which would ensure the possibility for business representatives themselves to participate in the realization and creation stage without IT specialists immediate help. Realizing the system by the suggested method, there is a possibility for business representatives to guide business processes, by changing business rules` permissible parts, in this way all the program`s and its realization logic and acceptable decisions. It`s easy to conform business processes to the changeable conditions, which in one or another way make influence to the enterprise`s activity by reducing the enterprise`s costs.

1. ĮVADAS

Šiuolaikinės verslo įmonės yra priverstos greitai reaguoti prie besikeičiančių aplinkos sąlygų, t.y. adaptuoti savo vykdomus veiklos procesus bei keisti jų parametrus, pasikeitus klientų poreikiams, ekonominei situacijai ir pan. [21]. Veiklos taisyklės yra priemonė, leidžianti efektyviau kontroliuoti veiklos procesus. Veiklos taisyklė (VT) – tai loginis teiginys, nusakantis kaip turi būti elgiamasi, t.y. kokių veiksmų turi būti imtasi, konkrečioje situacijoje. VT egzistuoja jau seniai, tačiau jos nebuvo aiškiai išreikštos ir laisvai prieinamos, t.y. jos buvo patalpintos programiniame kode, duomenų bazių trigeriuose, įmonių strateginiuose/gamybiniuose planuose, įvairiuose reglamentuose, darbdavių patirtyje. Su jomis dirbti galėjo tik IT ekspertai, taigi jų modifikavimas, besikeičiant aplinkos sąlygoms, buvo beveik neįmanomas. Taigi veiklos taisyklių paskirtis yra sudaryti, modifikuoti bei automatizuotai administruoti įmonės valdymo taisykles, apimančias klientų aptarnavimą, produktų gamybą, vidinę infrastruktūrą, finansus ir t.t. [22]. Tokiu būdu suformuojama automatizuota veiklos taisyklių valdymo sistema (VTVS). Joje VT tiek loginiame, tiek fiziniame lygmenyje yra atskirtos nuo programinio kodo, tačiau jomis naudojasi duomenų saugyklos, vartotojo sąsaja, programos [23]. VT sistemos tikslas yra užtikrinti greitą verslo procesų adaptavimąsi prie pasikeitusių aplinkos sąlygų, išvengiant sistemų darbo pertraukimų.

Pirmajame skyriuje pateikta veiklos taisyklių samprata. Išnagrinėti galimi veiklos taisyklių tipai ir struktūra. Atlikta lyginamoji veiklos taisyklių valdymo sistemų analizė, kurios pasirinktos pagal veiklos taisyklių apdorojimo tipologiją. Pateiktos VTVS architektūros bei integravimo schema į IS. Apžvelgti VT išvedimų technologijos būdai sprendimų priėmime. VT taikymo metodika programų kūrimo evoliucijos proceso metu. Pateiktas detalesnis VT vaidmens aprašas kiekvieno etapo metu.

Po atliktos analizės pasirinkta dalykinė sritis, kuria remiantis buvo sukurtas studijų administravimo sistemos prototipas, kuris atskleidė kaip galima kurti sistemas, naudojant atominius veiklos vienetus – veiklos taisykles. Iliustracinis pavyzdys pateiktas iš dėstytojų, finansininkių, sekretorių pozicijų. Naudojant skirtingus VTVS elementus, realizuoti skirtingi studijų procesai. Nustatyti trūkumai, neleidžiantys efektyviai modeliuoti veiklos taisyklių, kurie apriboja dalykinės srities atstovų priėjimą prie jų. Siekiant pašalinti šiuos trūkumus pateiktas pasiūlymas kurti veiklos taisykles remiantis šablonais. Aprašytas VT apdorojimo procesas VTVS vykdymo metu.

Atskleista veiklos taisyklių specifikavimo sintaksės architektūra, remiantis VTVS. Bendri principai VTVS elementų identifikavimui, kad būtų palengvintas VT ir jų elementų skaitomumas projekto rėmuose.

Pateiktas algoritmas, kuris aprašo kokius veiksmus reikia atlikti, norint sukurti VT sudedamuosius šablonus, kurie sudaro pagrindinio, vykdymui parengto, šablono turinį. Aprašyta šablono struktūra bei pateiktas detalizuotas formavimo algoritmas. Atliktas eksperimentas, kurio metu buvo sudarytas VT modeliavimo šablonas remiantis pateikta metodika.

Pastebėta, kad VT redagavimas, kūrimas ir šalinimas per RMA, dalykinės srities atstovams pateikia VT patogesniu būdu, nes jie mato tik redagavimo aplinką. Be detalių ir realizavimo principų, reikalingų VTVS. Valdant veiklos taisykles, suskurtas šablonais, dalykinės srities atstovui svarbiausios veiklos žinios, kurias intuityviai gali pildyti pagal sukurtą šabloną.

Darbo tikslas: atskleisti veiklos taisyklių specifikavimo ir manipuliavimo galimybes, remiantis šablonais.

Tyrimo objektas: veiklos taisyklės ir jų panaudojimas informacinėms sistemoms kurti.

Iškelti uždaviniai:

- Pasirinkti ir išanalizuoti dalykinės srities procesus.
- Pagrindinius procesus transformuoti į veiklos taisykles.
- Veiklos taisykles specifikuoti VTVS elementais.
- Išsiaiškinti VTVS elementų privalumus ir trūkumus.
- Pateikti pasiūlymą, kuris pagerina VT specifikavimo ir manipuliavimo procesą.

2. VEIKLOS TAISYKLIŲ SPECIFIKAVIMO BŪDŲ ANALIZĖ

2.1. Veiklos taisyklių sąvoka

Veiklos taisyklės vaidina svarbų vaidmenį ne tik versle, bet ir sistemų kūrimo procese. Dažniausiai jos užrašomos natūralia kalba, kad būtų lengvai suprantamos [14]. Priklausomai nuo požiūrio į veiklos taisykles, jos traktuojamos skirtingai. Didelė dalis žmonių besidominčių veiklos taisyklėmis, tiek mokslininkai, tiek mėgėjai skirtingai traktuoja veiklos taisyklių apibrėžimus. IT specialisto požiūriu veiklos taisyklė yra:

- VT – teiginys, kuris apibrėžia arba apriboja tam tikrais apsektais verslą. Teiginiai patvirtina veiklos struktūrą, juos galima valdyti ir taip atlikti pakeitimus esamai veiklos struktūrai.
- VT – teiginys, naudojamas kaip organizacijos pagrindas, plėtojantis jos veiklą. Jos yra kaip pagrindas, kuriuo remiantis suvokiama veikla [14].
- VT - taisyklė, kuri vykdo veiklos teisingumą [5].

Iš duomenų bazės perspektyvos – veiklos taisyklės tam tikrais būdais reaguoja į įvykius, kurie keičiasi bendrame taikomosios programos kontekste. Veiklos taisyklėmis modeliuojamas įvykių atoveiksmis, kuris veikia duomenų bazės turinį, sutrumpinant taikomųjų programų reakcijos laiką tiems įvykiams.

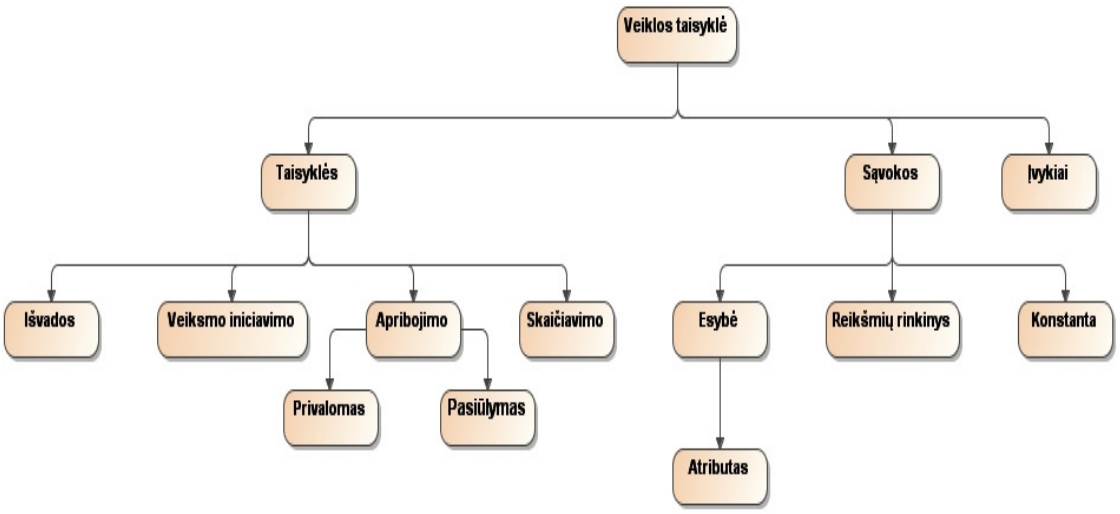
Atvirkštinės inžinerijos kontekste veiklos taisyklė suprantama kaip reikalavimų išreiškimas sąlygomis arba manipuliavimas duomenimis, išreikštais sąvokomis.

Iš reikalavimų perspektyvos veiklos taisyklė yra teiginiai, nusakantys organizacijos veiklos būdus. Jie atspindi strategiją, procedūras ar kitus apribojimus. Paprastai daugelis reikalavimų yra klasifikuojami į kategorijas, kuriuose atsispindi veiklos taisyklės, išreiškiančios, pavyzdžiui, skaičiuojamuosius reikalavimus. Kurie apibrėžia arba įtakoja veiklos plėtotę. Veiklos taisyklė yra reikalavimai, kylantys iš įmonės veiklos tikslų. Veiklos taisyklės – organizacijos politikos transformacija į sąlygos ir veiksmo detalizavimą. Tokiu būdu įgyvendinama organizacijos strategija.

2.2. Veiklos taisyklių struktūrizavimas

Nėra nusistovėjusios standartinės veiklos taisyklių klasifikavimo schemos. Bendru atveju veiklos taisyklės apima sąvokas, įvykius, taisykles. (1 pav.). *Sąvokos* - daiktavardžių sintaksinis darinys, kuris gali būti konstanta, tam tikras reikšmių rinkinys arba esybė ar jos atributas. *Esybė – dėstytojas, atributas – dėstytojo ID, reikšmė – lytis, reikšmių rinkinys – darbo dienos (P, A, T ,K, Pe, Š, S). Įvykiai* yra teiginiai, jungiantys sąvokas. Jais išreiškiamas konkretus faktas, apimantis sąvokos dalis. (*Dėstytojas gali dirbti šeštadienį). Faktas* yra ryšys

tarp esybių, o atributo ir esybės – asociacija. Sąvokos ir įvykiai intuityviai apibrėžiami prieš taisykles. Jie sudaro pamatą loginiam duomenų modelio, DB ir veiklos objektinio modelio kūrimui. Taisyklė – deklaratyvus teiginys, kuriuo pateikiama nauja informacija ar sprendimas apie atliekamą tam tikrą veiksmą. Taisyklės logika yra naudojama kaip tam tikra įvestis, kuri savo ruožtu pateikia išvestį: tam tikrą naują informaciją arba veiksmą.



1 pav. *Veiklos taisyklių sandaros koncepcija*

Įvykis – teiginys, apjungiantis sąlygas į tam tikrą pastebėjimą, apibūdinantį veiklą.

Apribojimai gali būti privalomieji (*mandatory),* nurodomieji (*guidlines). Privalomasis apribojimas* – užbaigtas teiginys, išreiškiantis besąlyginį įvykį, kuris privalo būti teisingas arba neteisingas (true/false) veiklos įvykiui. *Nurodomasis apribojimas* – užbaigtas teiginys, kuris įspėjas, kad tam tikras įvykis turėtų būti teisingas arba neteisingas (true/ false). Nurodomasis apribojimas tiktai įspėja, bet neapsprendžia tam tikrus atvejus. Paliekamas laisvas sprendimo priėmimas. Sistemos kūrimo metu, tokie apribojimai paprastai transformuojami į įspėjimo pranešimus. Kiekvienu atveju naudojami skirtingi raktažodžiai, sustiprinantys apribojimus.

Veiksmo iniciavimas – užbaigtas teiginys, kuris tikrina sąlygas. Jei sąlyga yra tenkinama, inicializuojamas kitas veiklos įvykis, pranešimas. Bendrame kontekste privalomieji apribojimai sustabdo tam tikrą įvykį, o veiksmo iniciavimas jį pradeda.

Skaičiavimo bei išvados taisyklės suformuoja naują informaciją iš turimosios.

Skaičiavimo taisyklė – užbaigtas teiginys, kuris apibrėžia algoritmą, kuris nurodo kaip gaunama tam tikros sąvokos reikšmė. Skaičiavimo taisyklės vykdymo rezultatas yra sukurta nauja dalinė informacija. Nauja informacija yra kaip nauja reikšmė atributui. Veiklos taisyklių

kūrimo eigoje, ši informacija naudojama loginiame duomenų modelyje, DB projektavime bei objektiniame modelyje.

Išvados taisyklė – užbaigtas teiginys, kuris tikrina sąlygas. Jei sąlygos tenkinamos nustatoma tam tikro teiginio tiesa kitam naujam įvykiui.

Visos veiklos taisyklės remiasi žiniomis. Sąvokos apibrėžia duomenų bendrą koncepciją ir detalizaciją, įvykiai nustato asociacijas tarp duomenų, apribojimai ir nurodymai testuoja duomenų reikšmes, skaičiavimai pateikia duomenų reikšmes, išvados yra kaip duomenų pabaigos žyma, veiksmo iniciavimas nustato duomenų reikšmių prioritetus inicijuojant veiksmą.

2.3. Veiklos taisyklių struktūros forma

Veiklos taisyklių tipų yra ne vienas. Paprasčiausios (produkcinės) taisyklės forma apibrėžiamai taip:

Jei sąlyga **tada** veiksmas

IF condition **THEN** action

Operatorius else tiksliai nusako alternatyvaus veiksmo atlikimą jei galimi objektai netenkina taisyklės sąlygos.

IF condition **THEN** actions.	**Jei** sąlyga **tada** veiksmas
ELSE alternate_actions	**toliau** alternatyvus veiksmas

Taisyklė yra teiginys, jungiantis veiksmų rinkinius su sąlygos rinkiniais. Taisyklės būsena išreiškia vieną ar kelis testuojamus tikslus. Kalbant apie veiklos taisykles, nuo jų neatsiejama sąlygos sąvoka. Sąlygos yra taisyklių prielaidos. Sąlygos rašomos patikrinti objektų reikšmes, kintamumą, taisyklių rinkinio parametrus ar kombinacijas [7]. Veiklos taisyklės turi būti susietos su objektais ir įvykiais. Taip pat atskirtos nuo veiklos taisyklių saugyklos bei valdomos atskiriant nuo taikomosios programos kodo [22].

Ne tik taisyklių apibrėžimas traktuojamas skirtingai, bet ir išskiriamos skirtingais aspektais taisyklių rūšys [16].

Veiklos taisyklės specifikacija yra tiksli ir detali, kuri kaip nepriklausomas komponentas bendrauja vienas su kitu visoje sistemoje. Nuo to laiko veiklos taisyklės charakterizuoja elgseną kasdieninėje veikloje. Jos yra išreiškiamos kaip sąlyginiai reiškiniai pagal tam tikras formules. Veiklos taisyklė susideda iš dviejų dalių : sąlygos ir veiksmo. Kai sąlyga yra aptinkama, veiksmas yra sužadinamas. Detalesnė sintaksė, kuri yra naudojama veiklos taisyklėm specifikuoti:

Objekto sintaksinis darinys + sąlyga + veiksmas

Paprasta veiklos taisyklių sintaksė gali būti išreikšta keliais metodais:

- IF + sintaksinis darinys+ THEN + sintaksinis darinys +ELSE + sintaksinis darinys.
- WHEN + sintaksinis darinys + THEN sintaksinis darinys.
- Objekto sintaksinis darinys + veiksmo sintaksinis darinys + WHEN + sąlyga [4].

Veiklos taisyklės gali būti apibrėžiamos ir klasifikuojamos skirtingais būdais. Veiklos taisyklės yra atominė dalis visos veiklos logikos, specifikuojamos deklaratyviai, kurios tikslas kontroliuoti, vadovauti gerinti elgseną. Taisyklės kuriamos, tikslų užtikrinimui, kasdieniniam produktyvumui didinti veiklos procese. Veiklos taisyklių specifikavimas grafiniais būdais.

2.3.1. Specifikavimas informacijos srautų diagramomis

Informacijos srautų diagramos (DFD) tiksliai apibrėžia informacijos srautus tarp proceso ir išorinio įtaiso arba duomenų įrašymo. Čia neapibrėžiamas veiksmų eiliškumas. Informacijos srautai kartais, bet ne visada, sutampa su įvykiais. Yra informacijos srautai kurie netiesiogiai išplaukia iš įvykio, kuomet jis įvyksta. Taip pat yra laiko ir kontrolės įvykiai, kurie negali būti apibrėžiami ir pristatomi kaip informacijos srautai.

DFD neleidžia išreikšti sąlygos, sinchronizuojančios duomenų srautus, atsižvelgiant į proceso vykdymą. Pavyzdžiui:

ON įterpimas vykdant iš eilės

DO parašyti patvirtinimo raidę;

keisti skilties būseną „patvirtintas“

Šis pateiktas pavyzdys gali būti modeliuojamas pagal DFD, nes ši veiksmo taisyklė neturi sąlygos dalies. Įvykis šiuo atveju numanomas, laikinas ryšys tarp paleidžiamo įvykio ir veiksmo yra praleistas.

Elementarūs procesai DFD yra paprastai vaizduojame per mini specifikacijas, kurios gali apimti veiklos taisykles. Vienas iš būdų yra jų specifikavimas sprendimų lentelėmis arba medžiais, kurie apibrėžia sąlygas ir veiksmus, bet formaliai nepalaiko įvykio komponento [1, 10]

2.3.2. Specifikavimas Merise pagrindu

Merisės pasiūlytas modelis sutrumpintai apibrėžiamas CPM (konceptualus vykdymo modelis), kuris palaiko įvykių iniciavimą, operacijas, įvykių rezultatyvumą. Sinchronizavimas derinasi su sąlygos komponentu, esančiu ECA mechanizme [20]. Kiekviena operacija gali nukreipti į skirtingus įvykius, pavyzdžiui, „operacija įvykdyta sėkmingai“ ar „operacija nutraukta“. CPM modelyje įvykis yra kaip atributų laikmena.

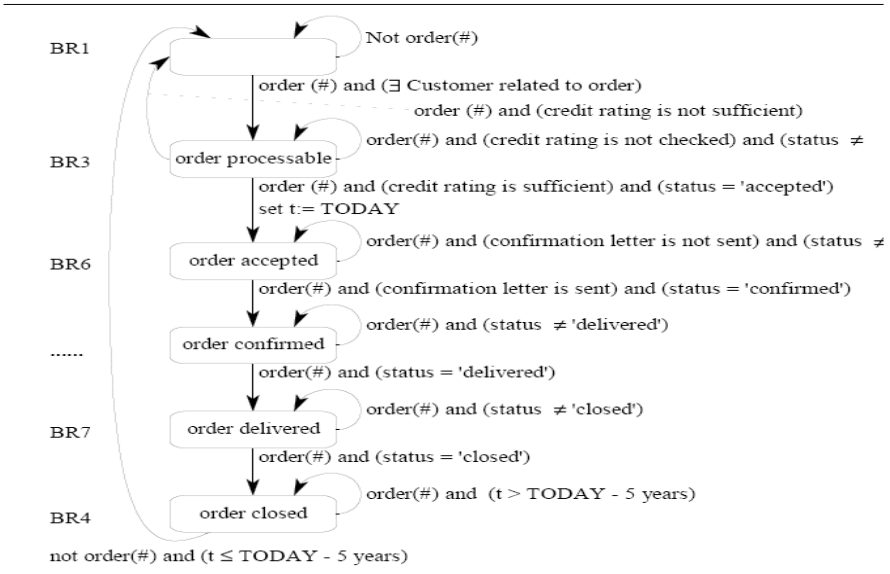
CPM koncentruojasi į dinaminį modeliavimą ir priklausomybes tarp atskirų taisyklių.

Čia naudojamos dvi grafinės konstrukcijos. Elipsė vaizduojanti trigerį, pranešimą ar įvykį atitinkančias būsenas. Stačiakampis – vaizduojantis sąlygą [10].

2.3.3. Specifikavimas State-Transition modeliais, perėjimų grafais ir būsenų diagramomis

Būsenų perėjimų diagramos (STD) paremtos grafais, kurių viršūnės simbolizuoja įvykius (pristatomas kaip apskritimas), o rodyklės simbolizuoja perėjimą. Jei kelios rodyklės išeina iš viršūnės, tuomet tiktai vienas iš perėjimų gali būti įvykdomas, priklausomai nuo jų pažymėjimo.

Perėjimų grafai, kuriais paremtas STD kūrimas, yra sudaryti iš laiko taisyklių ir naudojami apibūdinti specialių objektų gyvavimo ciklą su apribojimais. Viršūnės pristato tam tikras objektų būsenas, rodyklės išreiškia perėjimų būsenas iš vieno galimo objekto būsenos į kitą. STD neaiškiai pristato veiklos taisyklių komponentų simbolius. Perėjimai žymimi priklausomai pagal sąlygas ir veiksmus. Šis specifikavimo būdas labiau tinka atvaizduoti priklausomybes tarp taisyklių [10]. Būsenų perėjimo diagramos pavyzdys pateikiamas (2 pav.)



2 pav. *Būsenų perėjimo (State – Transition) modelis [10].*

2.3.4. ER – RM modelis

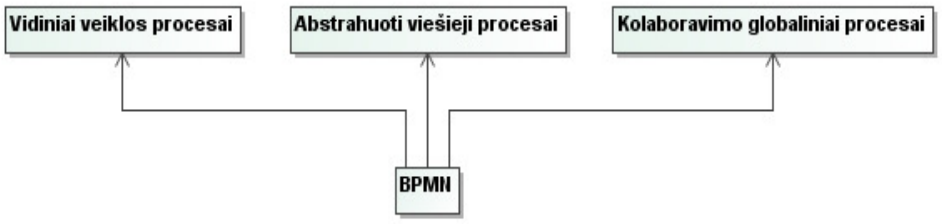
Šiame modelyje pristatomos situacijos – veiksmo taisyklės kontroliuojančios objektų būsenas, sąveiką ir jų atributus. Taisyklė bus teisinga kai įvyks konkretus įvykis ir sąlyga bus patenkinta.

Taisyklės išreiškiamos ER – R diagramomis kaip lygiagretainiai. Jos pažymimos charakteringais ir savitais vardais, nes veiklos turinys nėra pateikiamas diagramoje. Lygiagretainis, su trimis aktualiais dariniais, jungiamas rodykle. Rodyklės, įeinančios į lygiagretainius, simbolizuoja įvedimą arba trigerių įvykius. Išeinančios rodyklės simbolizuoja išvestį arba įvykio rezultatą. Rodyklėms yra įvedami simboliniai žymėjimai kaip: įvedimas (i), atnaujinimas (u), šalinimas (d). Punktyrinės linijos naudojamos norint pavaizduoti, jog reikalingi papildomi duomenys taisyklei [10].

Naudojant šią notaciją, svarbu nurodyti ryšius tarp taisyklių bei su jomis susijusius objektus.

2.3.5. Veiklos procesų diagrama(BPD)

Veiklos procesų diagramas tikslinga naudoti projektuojant ir valdant dalykinės srities veiklos procesus. Šių tipų diagramos remiasi veiklos procesų modeliavimo notacija (BPMN). Kitaip tariant, veiklos procesus galima suvokti per grafinę išraišką. Pateikiami procesų tipai (3 pav.).



3 pav. *Procesų tipai, kurie gali būti aprašomi veiklos procesų valdymo notacija*

Vidiniai veiklos procesus turi kiekviena organizacija, kurie bendru atveju gali būti traktuojami kaip duomenų srautai. *Abstrahuoti viešieji procesai* apima vidinių procesų bei kitų procesų arba veiklos dalyvių sąveiką. Juose nurodoma vidinių procesų sąveika su kitomis išorinėmis veiklomis. *Kolaboravimo vidiniai procesai* vaizduoja sąveiką tarp dviejų ar daugiau veiklos objektų. Sąveikavimas yra kaip seka veiklų, kurios nurodo pranešimų pasikeitimų šablonus tarp įtrauktų objektų.

Šia specifikacija atvaizduoja duomenų srauto (pranešimais) bei artefaktų ir veiklos asociacija. Tačiau tai nėra tas pats, kas informacijos srauto diagramos (DFD).

Diagramose naudojami šie baziniai elementai:

- Srauto objektai
- Sujungiamieji objektai
- Swimlines

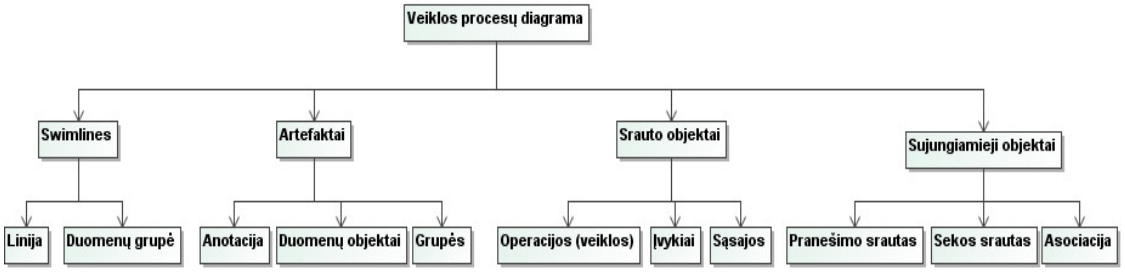
- Artefaktai

Srauto objektai yra pagrindiniai grafiniai elementai, apibrėžiantys veiklos proceso elgseną. Tai yra įvykiai (žymimi apskritimu), veiklos operacijos (žymimi stačiakampiu apvalintais kampais), sąsajos (žymimos rombu). Srauto objektus galima sujungti trijų tipų sujungimo objektais. *Sekos srautu*, *pranešimo srautu*, *asociacija*. Jos atitinkamai žymimos skirtingo tipo rodyklėmis, kurios nurodo srauto kryptį tarp objektų.

Grupuojama dviem būdais: duomenų grupe (pools), linija. Duomenų grupe pažymimas proceso dalyvis. Arba – grafinis konteineris, jungiantis rinkinį užduočių iš kitų duomenų grupių. Paprastai šio elemento vaizdavimas tikslingas B2B situacijose. Linija dalija duomenų srautą. Taip yra kategorizuojamos tam tikros užduotys. Toks žymėjimas palengvina bendro diagramos konteksto skaitomumą.

Artefaktai atskleidžia papildomą informaciją apie procesus. Artefaktai apima duomenų objektus, grupes bei anotaciją. Duomenų objektas yra laikomas artefaktu dėl to, kad jis neturi tiesioginės įtakos procesų sekos srautui ar pranešimo srautui. Tačiau artefaktai suteikia informaciją apie tai, kokios užduotys turi būti atliktos ir/arba ką iš jų galima gauti, jas atlikus. Grupė naudojama tos pačios kategorijos užduočių grupavimui. Grupė identifikuojama kategorijos pavadinimu. Kategorijos gali būti naudojamos dokumentacijos arba analizės tikslais. Todėl grupė yra vienas iš būdų, kuriuo objektų kategorijos gali būti vizualiai pateikiamos diagramoje.

BPMN lygmenyje galima atvaizduoti ne tik veiklos procesus, bet ir programinės įrangos veikimą. Grafinė notacija palengvina veiklos funkcionalumo efektyvumo suvokimą bei veiklos transakcijas. Jos nagrinėjamos organizacijos viduje arba tarp kelių organizacijų. Pagrindiniai veiklos procesų diagramos elementai pateikiami 4 pav.

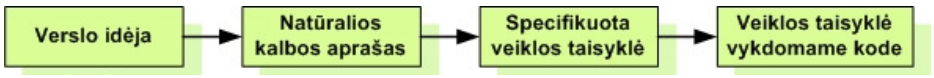


4 pav. *Procesų tipai, kurie gali būti aprašomi veiklos procesų valdymo notacija*

2.4. Veiklos taisyklių apžvalga remiantis MDA

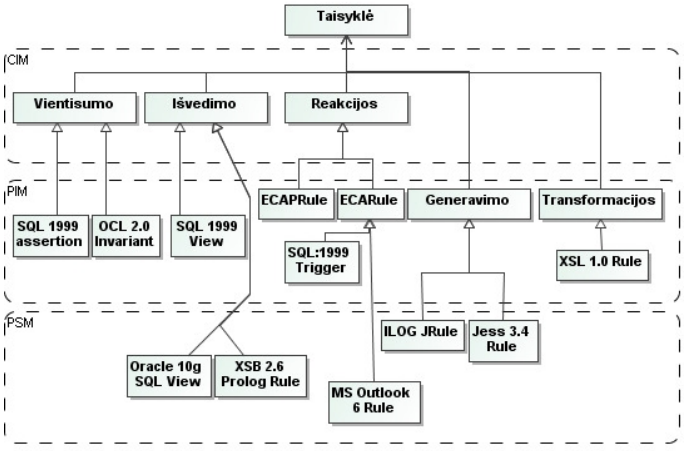
Veiklos taisyklė gali būti keturių skirtingų abstrakcijos lygių (5 pav.) [9], kiekvienas iš jų skirtas skirtingai darbo grupei dirbančiai su VT [7]. Pirmajame lygyje veiklos taisyklę

suformuoja įmonės atstovai, išreikšdami idėją, kad tam tikras įmonės vykdomas procesas turėtų būti kontroliuojamas. Antrame lygyje VT idėja užfiksuojama neformaliu VT aprašu, nusakančiu jos pagrindinę paskirtį, tačiau nedetalizuojant jos veikimo. Šiame lygmenyje siekiama VT poreikį išgryninti, t.y. nors ir neformaliai veiklos taisyklė yra interpretuojama kaip nedalomas vienetas susijęs su konkrečiais verslo procesais. Trečiame lygyje veiklos taisyklė yra modeliuojama pasirinkus taisyklių specifikavimo kalbą. Ši užduotis yra skirta IT ekspertui (analitikui/projektuotojui). VT specifikacija atsako į klausimą „ką veiklos taisyklė turi daryti“, bet ne „kaip“.



5 pav. *Veiklos taisyklių abstrakcijos lygiai*

Šiame lygyje veiklos taisyklė įgyja visas deklaratyviai apibrėžtai taisyklei reikalingas savybes, tokias kaip užbaigtumas, tikslumas, autonomiškumas, deklaratyvumas, aktualumas. Idealiu atveju, tinkamai sumodeliavus veiklos taisykles, remiantis MDA (angl. *Model Driven Architecture, Modeliais grindžiama architektūra*) principu, iš jų galima sugeneruoti programinį kodą. MDA išskiria tris fundamentaliuosius modeliavimo lygius: CIM (*computation independent modeling, Nepriklausomas skaičiavimų modeliavimas)* , PIM (*platform independent modeling, nuo platformos nepriklausomas modeliavimas)* ir PSM (*platform specific modeling, konkrečios platformos modeliavimas).*



6 pav. *Taisyklių koncepcija trijuose skirtinguose abstrakcijos lygiuose: CIM, PIM ir PSM*

CIM lygyje esančios taisyklės yra teiginiai, išreiškiantys verslo idėją deklaratyviu būdu, paprastai naudojant natūralią kalbą ar tam tikrą vizualizavimo kalbą. PIM lygmenyje, taisyklės yra teiginiai, išreikšti tam tikru formalizmu, kuris gali būti tiesiogiai atvaizduojamas

į programinę įrangą. Taisyklių kalbų pavyzdžiai, atitinkantis šį MDA lygmenį pateikiami 6 pav. PSM lygmenyje taisyklės yra teiginiai, išreikšti tam tikros vykdymo aplinkos kalba.

Vientisumo taisyklės, gali būti apibrėžiamos kaip apribojimai, kurie gali būti paremti predikatų logika. Jos išreiškia teiginius, kurių privaloma laikytis evoliucionuojant sistemai. Išvedimo taisyklės turi sąlygos bei išvados dalis. Reakcijos taisyklės susideda iš privalomo paleisties įvykio sąvokos, papildomos sąlygos ir paleisties veiksmo sąlygos arba po – sąlygos. Reakcijos taisyklių sąlyga yra tokia pat kaip ir išvedimo taisyklių. Išskiriama dviejų tipų reakcijos taisyklės *ECA* (*Event – Condition – Action)* bei ECAP (*Event – Condition – Action – Post-condition).* Generavimo taisyklės susideda iš sąlygų ir veiksmo. Jų evoliucionavimas ir naudojimas sparčiai išaugo atsiradus ekspertinėms sistemos apie 1980 metus.

2.5. Veiklos taisyklių technologija

Dažniausiai programose, kuriuose disponuojama veiklos taisyklėmis, naudojami algoritmai, kurie remiasi Rete algoritmo pagrindu. Šį algoritmą atrado Charles Forgy.

Čia taisyklės traktuojamos kaip atskiri duomenys, šis požiūris leidžia atlikti taisyklių konfigūraciją, nekeičiant programos kodo.

Rete algoritmas gali apdoroti taisykles tik tada , kai jos apibrėžtos tiksliau aprašytos formaliai, griežta sintakse.

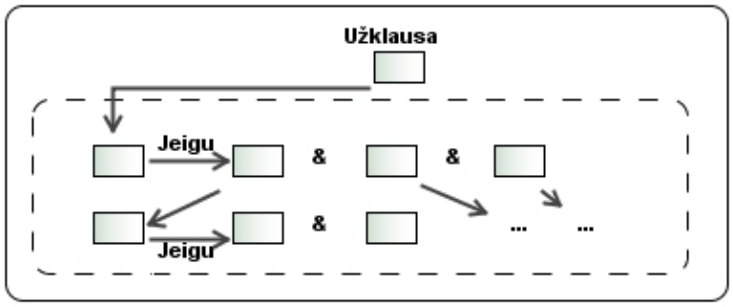
2.6. Veiklos taisyklių sprendimų išvedimo būdai

2.6.1. Atgalinis susiejimas

Veiklos taisyklės sąveikauja tarpusavyje keliais būdais, t.y., naudojamas atgalinis susiejimas (backward changing) ir tiesioginis susiejimas.

Atgalinio susiejimo atveju sistema gauna užklausą. Suaktyvinamos tik reikiamos taisyklės, kurios tenkina sąlygą (7 pav.).

- Užklausa yra prijungiama prie išvados, kuri susieja taisyklės reikšmę su objektu ir užklausos reikšme.
- Tada taisyklių variklis stengiasi pateikti išvadą, pateiktai taisyklei.
- Pirmiausia taisyklių variklis pažiūri ar yra faktas, su kuriuo sąlyga siejasi.
- Jei ne, tada peržiūri vėl taisykles, kurios gali užbaigti sąlygą.
- Antra taisyklės išvada jungiama su pirmosios taisyklės sąlyga. Susieja reikšmę su objektu.
- Variklis bando nustatyti abi sąlygas antrajai taisyklei ir t.t.
- Pateikiamas sėkmingas arba nesėkmingas užklausos sprendimas.

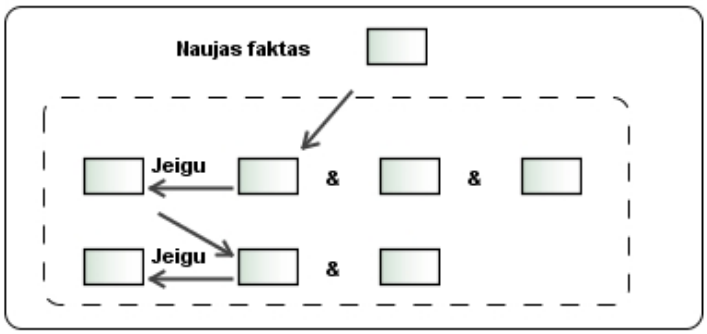


7 pav. Atgalinis susiejimas

2.6.2. Tiesioginis susiejimas

Tiesioginis susiejimas pradedamas įterpiant naują faktą (8 pav.). Sistema bando surasti taisykles, kurių sąlyga siejasi su nauju faktu. Nustato naują sąlygą ir pateikia taisyklės išvadą kaip naują faktą. Ši strategija naudojama tada, kai nėra nurodyto tikslo.

- Faktas susilieja su sąlygomis.
- Taisyklių variklis bando nustatyti taisyklės pirmąją sąlygą ieškodamas fakto, kuris siejasi su taisykle.
- Jei faktas yra randamas, taisyklės sąlyga yra įterpiama kaip naujas faktas. Šis įterpimas iš naujo aktyvuoja tiesinį sujungimą [13, 17].



8 pav. *Tiesioginis susiejimas*

2.7. Veiklos taisyklių valdymo sistemos (BRMS)

Veiklos taisyklės praktikoje naudojamos gan seniai. Kiekvienos veiklos srities kūrimas, evoliucionavimas, egzistavimas paremtas tai veiklos sričiai būdingomis taisyklėmis. Jomis remiamasi priimant tam tikrus sprendimus.

Pastaruoju metu visame pasaulyje veiklos taisyklių naudojimo paklausa sparčiai auga. Jos naudojamos ne tik programos kūrimo etapuose, bet ir priimant svarbius sprendimus, bankininkystėje, biržoje ar kitose srityse.

Veiklos taisyklės naudojamos taisyklių valdymo sistemose (business rules management system). Nepriklausomai nuo organizacijos veiklos šios sistemos įgalina sumažinti išlaidų kaštus susijusius su kuriama programa, ar sukurtos programos tobulinimu.

2.7.1. Veiklos taisyklių valdymo sistemų klasifikavimas, pagal juose apdorojamų taisyklių tipus

Rinkai yra pateikiama nemažai programinių sprendimų, paremtų veiklos taisyklėmis, kurios iš esmės skiriasi tik paskirtimi bei kaina. Vienas iš pagrindinių sistemų klasifikavimo kriterijų būtų skirtingos tipologijos taisyklės, kurias valdo sistema.

- Paprastos veiklos taisyklės (Simple Business Rules) arba produkcinės veiklos taisyklės – tai taisyklės, kurios išreiškiamos standartizuoto žodyno forma. Šios grupės taisyklėmis manipuliuoja tokie produktai kaip JRules [6], Blaze Advisor [7] , QuickRules [5] ir t.t
- Dirbtinio intelekto taisyklės (Artficial intelligence rules) – taisyklės, kurios įtakoja algoritmus DI ir duomenų gavybos sistemose. Šias taisykles apdoroja tokios sistemos kaip IBM kompanijos DB2 [8] ir kt.
- Įvykių apibendrinimo taisyklės (Event correliation rules) – taisyklės, naudojamos įvykių apibendrinimui.
- Taisyklės orientuotos į duomenis (Data – Centric rules) – taisyklės, kurių pagrindinė paskirtis – duomenų paieška ir atnaujinimas. Apribojimai kontroliuoja kaip duomenys yra transformuojami ir kas juos gali pasiekti. Sintaksės, semantikos ir konteksto įgyvendinimo pagalba užtikrinamas duomenų vientisumas. Šiai grupei būtų galima priskirti Versata produkciją [4].

FairIsaac kampanijos pateikta BRM sistema Blaze Advisor yra orientuota į paprastąsias veiklos taisykles, todėl tikslinga panagrinėti sistemas, kurios turi analogišką kryptingumą. Šios kampanijos produktas yra laikomas lyderiu rinkoje, jį vejasi tik ILOG grupė pateikusi rinkai JRules BRM sistemą.

Blaze Advisor paketas šiomis dienomis naudojamas bankuose, draudimo sektoriuje, apdirbamosios pramonės įmonėse, telekomunikacijų kompanijose taip pat ir vyriausybiniuose organuose bei daugelyje kitų sektorių.

Didžiausia internetinės bankininkystės kompanija Egg, esanti Amerikoje, teigia, jog jos vaidmuo viso pasaulio rinkoje išaugo tada, kai pradėjo naudoti Blaze Advisor, kuris užtikrina greitą ir efektyvų duomenų perdavimą. Kompanija pateikia tokius skaičius:

- Įvairūs persiuntimai išaugo nuo 9% iki 35 %.
- Pajamos padidėjo apie 1.5 %.

- Nemokių klientų, kuriem išduodamos paskolos, sumažėjo 1 %. Sistema padeda tinkamai atrinkti klientus.

Egg lyderiai atliko skaičiavimus, kurie parodė, jog pasirinkta technologija ir sprendimų priėmimo būdas, leis bankui sutaupyti 5 milijonus dolerių per metus.[1]

19 Banque Populaire iš 23 mažmeninės bankininkystės atstovų teigia pasiekę šių rezultatų:

- Pavyko sutrumpinti projekte programavimo laiką 30%, lyginant su standartiniu programavimu.
- Numatoma sutrumpinti laiką, kurio reikia programuojant atnaujinimus, apie 50% - 75%.

Veiklos taisyklių valdymo sistema Blaze Advisor InfoWorld skelbtame konkurse „2007 Technology of the Year Awards“ laimėjo pirmąją vietą, kaip novatoriškiausias veiklos taisyklių valdymo produktas, kuris pralenkė JRules produktą, sparčiai bandantį pasyvyti Fair Isaac kompanijos BRMS produktą. Per 2006 metus testavimo centro darbuotojai testavo įvairaus pobūdžio plačiai paplitusias pasaulyje programas. Jų buvo daugiau nei 230 iš jų apdovanotos 41 techninės ir programinės įrangos produktai, kurie savo produktų klasėje pateikė vartotojam geriausius sprendimus.

2006 metų rugsėjo mėnesį IDC ataskaitoje buvo prognozuojamas Fair Isaac kompanijos lyderiavimas, kaip paklausiausios BRMS programinės įrangos, visame pasaulyje per 2006-2010 metus. Augantis šios kompanijos biudžetas tvirtina, jog prognozės buvo teisingos.[2]

2.8. Veiklos taisyklės sistemų kūrimo procese

Veiklos taisyklės gali būti taikomos programos kūrime. Veiklos taisyklių metodologija, neįveda didelių pakeitimų tradiciniame programos kūrimo procese.Tačiau patobulina ir praplečia pagrindinius principus. Veiklos taisyklės yra tiesiogiai programuojamos remiantis veiklos taisyklėms būdinga if...then...else išraiška. Tačiau čia susiduriama su keliomis problemomis. Pagrindinės iš jų būtų šios:

- Sunku programos kode išdėstyti veiklos taisykles taip, kad jos būtų interpretuojamos korektiškai.
- Sunku suprasti veiklos taisyklių vertimą, kai jis atliekamas programiniame kode.

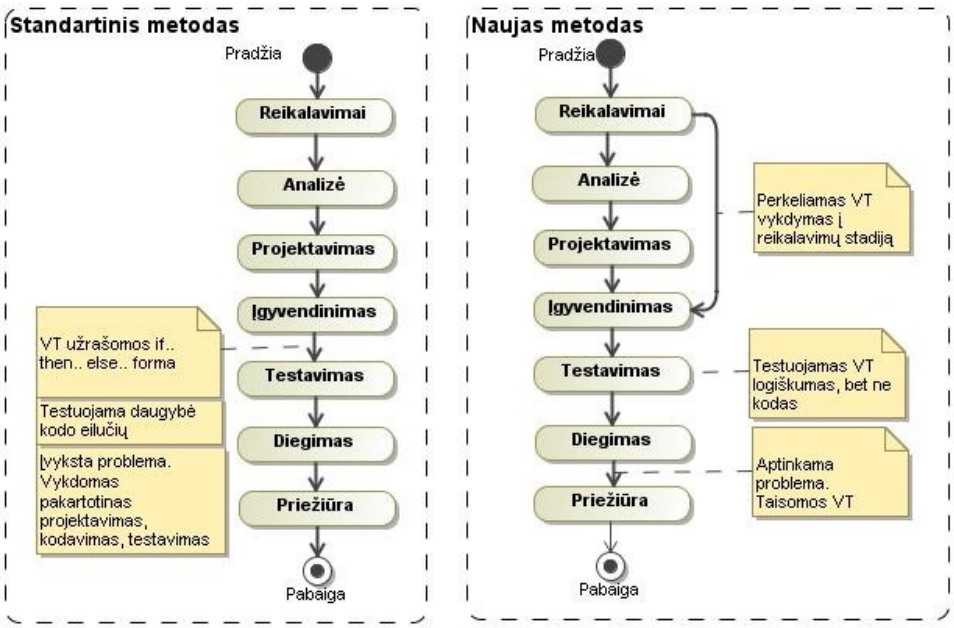
Šie teiginiai tampa esminiais, kada susiduriama su programos tobulinimu, keitimu ir t.t., visa tai pradedama įgyvendinti kai keičiasi tam tikros veiklos strategija, planai apribojimai ir panašiai [6].

Todėl tikslinga veiklos taisykles naudoti sekant šiems etapams:

- Veiklos analitikas apibūdina veiklos taisykles.
- Kūrėjas konstruoja taisyklių rinkinius paprastu formatu.
- Atskiriamos veiklos taisyklės nuo realizavimo etapo.
- Veiklos taisyklės apibrėžiamos programos kūrimo reikalavimų etape kaip atskiras komponentas.

Visa tai iliustruojama 9 pav.

Kalbant apie veiklos taisyklių plėtotę pirmiausia reikėtų išsiaiškinti ir suprasti kokia informacija turi būti užfiksuota. Norint atlikti įvertinimą, jei taikomojoje programoje įvyksta veiklos pažeidimas, programuotojams reikia nustatyti rinkinį įvykių, kurie galėjo pažeisti taisyklę. Visa tai apibrėžiama kaip potencialių pažeidimų įvykių aibė (PVE). Tuomet programuotojas turi susumuoti informaciją apie tai, kokių veiksmų bus imamasi, jei aptinkamas taisyklės pažeidimas. Arba pati programa turi pabandyti atitaisyti įvykusį pažeidimą (pvz., prašant įvesti vartotojui kitą teisingą reikšmę, arba grąžinti vartotoją į prieš tai įvykusį veiksmą).



9 pav. *Veiklos taisyklės programų kūrimo etape.*

Kiekvienos taisyklės PVE galima nustatyti apsaugos sąlygą, kuri nustatytų ar tikrai įvykis pažeidė taisyklę. Nustatant tokius apribojimus galima išvengti būsimų pažeidimų, kurie gali įtakoti galutinį rezultatą.

Kiekviena veiklos taisyklė gali būti laikoma kaip trilypė aibė S{a, b, c}.

Aibės nariai atitinka šias reikšmes:

- a – taisyklės PVE.
- b – apsaugos sąlyga susijusi su įvykiu.
- c – veiksmas, kurio imamasi užkertant kelią pažeidimams arbą taisant pažeidimus tam tikroje situacijoje.

Norint išvesti naują taisyklę t‘ iš senos taisyklės t, reikia atlikti šiuos veiksmus:

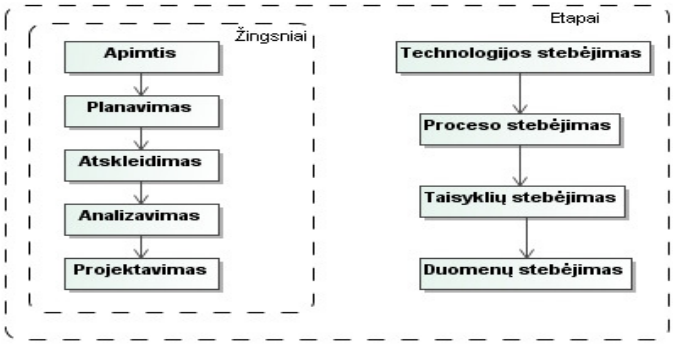
1. Pašalinti iš kodo vykdomas apsaugas ir veiksmus įvykių, kurie išreiškiami S(t), bet ne S(t‘).
2. Kodo fragmentų įterpimas, įvedant apsaugą ir įvykių veiksmus, kurie yra pristatomi S(t‘) bet ne S(t). Tai apima visos programos fragmentų identifikavimą, kad įvykiai būtų įvykdyti, būtina apsaugos ir veiksmo formuluotės gali būti įtraukiamos į programą.
3. Kodo fragmentų modifikavimas, kad įvykdytume apsaugas ir įvykių veiksmus, kurie yra pristatomi ir S(t), ir S(t‘). Kaip ir pirmajame atvejy, reikia pirmiausia identifikuoti visus programos fragmentus, kad realizuotume apsaugas ir veiksmus abiem įvykiams, tiek senajam tiek naujajam įvykiui. Tuomet ištriname visus elementus, kurie nėra bendrai naudojami pagal naują apsaugą ar veiksmą, ir realizuojame bet kurį trūkstamą elementą.

Programuotojai, turėtų atlikti pagrindines užduotis:

- Suprasti ir išanalizuoti veiklos taisyklę, tam, kad nustatyti kokie įvykiai gali pažeisti, ir kokie apsaugojimo būdai ir veiksmai yra reikalingi. Nustatyti ir papildyti kodo fragmentus, kad būtų įvykdyti įvykiai, apsaugos ir veiksmai.
- Suvokti kiekvienos kodo eilutės vaidmenį bendrajame programos kontekste ir taip nuspręsti, kurie kodo elementai turi būti įtraukti arba ištrinti iš programos. [3, 18, 21]

Veiklos taisyklės taip pat naudojamos sistemų reikalavimų specifikavimo procese. D. Ch. McDermid siūlo naudoti veiklos taisyklių diagramas (BRD) [1].

2.9. Veiklos taisyklių metodikos etapai



10 pav. *Veiklos taisyklių metodikos etapai*

Pagal [24] šaltinį.

2.9.1. Veiklos taisyklių metodikos etapų detalizavimas

11 pav. *VT metodikos etapai, kurie atliekami sistemos kūrimo metu*

Technologijos stebėjimas apima sistemos kūrimo technologijos pasirinkimą. Jos pritaikymą užsakovo reikalavimams registruoti bei realizuoti.

Proceso stebėjimas orientuotas į aktorių sąveikos išskyrimą. Žmonių ir esamos sistemos, jei tokia yra, tarpusavio ryšių (veiksmų) nustatymą. Proceso stebėjimo metu sudaromas sąrašas visų veiklos įvykių, kurie turės funkcionuoti būsimojoje sistemoje (kiekvienas veiklos įvykis susiejamas su veiklos procesais), kokie sprendimai priimami įvykus įvykiui, proceso metu. Paprastai nuo įvykių procesų supratimo pereinama prie sprendimų atskleidimo prieš įvykstant tam tikram įvykiui, tokiu būdu pereinama nuo proceso stebėjimo prie taisyklių stebėjimo.

Taisyklių stebėjimas identifikuoja taisyklių rinkinį. Taisyklės yra kaip nepriklausomi loginiai komponentai, atskirti nuo programinio kodo bei duomenų. Taisyklės formuojamos taip, kad atspindėtų organizacijos viduje ar išorėje atliekamus veiksmus ir sprendimus, kurie atsispindės sistemoje. Taisyklės pateikiamos organizacijos atstovams. Kad jie jas įvertintų, nustatydami jų logiškumą, teisingumą bei kitus matus. Remiantis pateiktų taisyklių specifikacija, veiklos ekspertas pateikia galimus situacijos sprendimo variantus įprastoms ar nenumatytoms situacijoms spręsti, kurias iššaukia vienoks ar kitoks veiklos įvykis. Taisyklių stebėjimo pirmasis žingsnis – nustatyti taisykles prieš veiklos įvykį arba prieš sprendimo priėmimą tam įvykiui. Tuomet analizuojamos priklausomybės tarp taisyklių. Taisyklių stebėjimas, kaip ir visi kiti stebėjimai, plačiai paplitę visuose metodikos etapuose. Pavyzdžiui, apimties etape, nustatomas bendrosios veiklos konteksto taisyklės, kurios išplaukia iš organizacijos tikslų ir strategijos. Atskleidimo etape, nusprendžiama pagal kokias kategorijas taisyklės bus grupuojamos į logines grupes. Analizavimo etapo metu atliekamas veiklos taisyklių optimizavimas iki minimalaus, tačiau logiškai išpildyto, taisyklių rinkinio. Tuomet analizuojama taisyklių priklausomybė.

Duomenų stebėjimo metu suformuojamas duomenų modelis bei galima duomenų bazė. Kai atskleidžiamos taisyklės, jų kiekvienas žodis ar išraiška turi atsispindėti duomenų modelyje ir DB. Sekant šio etapo žingsniams, faktai ir sąvokos apjungiamos į vieningą duomenų struktūrą, naudojant architektūrinę duomenų analizę ir projektavimo principus.

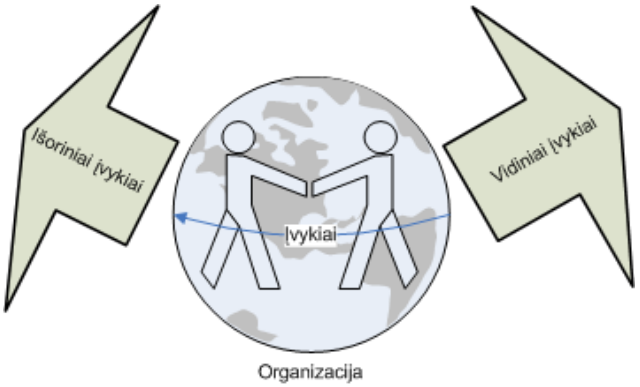
Tačiau gaunama tik tam tikra struktūra su minimaliu rinkiniu taisyklių, kuris pasipildo evoliucionuojant veiklos procesams arba keičiantis veiklos logikai. Tokiu atveju, pakeisti sistemą greitai ir esant minimalioms sąnaudomis yra gan sudėtinga. Keitimo darbus atliekantys programuotojai turi įsigilinti ne tik į sistemos logiką, bet ir į jos programinį kodą. Visa tai reikalauja didelių laiko sąnaudų bei kaštų.

Veiklos taisyklių metodikos etapai pateikiami 10 pav., jų detalizacija 11 pav.

2.9.2. Veiklos taisyklių metodikos žingsnių detalizavimas

2.9.2.1. Apimties etapas

Apimtis yra procesas, kurio metu fiksuojami bendriausi reikalavimai ir ribos naujai arba tobulinamai sistemai. Jo metu identifikuojami veiklos įvykiai, susiejant su planuojamos sistemos realizacija, bei duomenų srautai esantys konkrečioje organizacijoje.



12 pav. *Organizacijos įvykių klasifikacija*

Veiklos įvykiai gali būti dviejų tipų: *išoriniai* ir *vidiniai* (12 pav.). Išoriniai įvykiai ateina iš globalios aplinkos, kitaip tariant išplaukia ne iš sistemos konteksto. *Laikini įvykiai*, tokie įvykiai, kurie įvyksta iš anksto nustatytu laiku ar metu. Pavyzdžiui, atsiskaitymo pažymys turi būti įrašytas kiekvieno laboratorinio darbo pabaigoje. Įvykiai identifikuojami vieninga forma (13 pav.). Pavyzdžiui, sekretorė registruoja studentą. Aktorius yra asmuo, kuris sąveikauja su sistema. Išskiriami kelių rūšių vartotojai. *Tiesioginis vartotojas* yra asmuo, kuris naudojasi sistemos funkcijomis, ar teikiama sistemos išvestimi kaip darbo atsakomybės dalimi. Tiesioginiu vartotoju gali būti bet kuris asmuo, kuris turi priėjimo teisę prie organizacijos sistemos funkcijų. *Dalyvis* yra asmuo, kurio indėlis reikalingas viso projekto metu. Pavyzdžiui srities tikslų ekspertas, kuris padeda atskleisti projekto apimtį. *Išoriniai stakeholder* yra asmenys ar organizacija, kuri domisi planuojama sistema arba kažkokiu būdu įtakoja jos dizainą ir operacijas.

Kiekvienam įvykiui apibrėžiamas įvykio – atsako procesas. Kiekvienam įvykiui numatoma kokią informaciją turi pateikti sistema. Pavyzdžiui, įvykis dėstytojas rašo įvertinimą dešimtbalėje sistemoje. Tuomet įvykio – atsako procesas gali būti vykdyti įvertinimą dešimtbalėje sistemoje. Įvykio atsako proceso aprašas susideda iš veiksmažodžio/daiktavardžio arba daiktavardžio išraiškos. Įvykius ir įvykio – atsako procesus galima registruoti lentelėje, kuri palengvina skaitomumą. Nurodant veiklos įvykį, įeinamą informaciją, įvykio – atsako procesus, išeinamą informaciją.

13 pav. *Įvykio registravimo forma*

Projekto apimties nustatymo etape fiksuojami bendriausi naujos ar praplečiamos savo funkcionalumu, sistemos reikalavimai ir ribos. Pagrindinis tikslas – identifikuoti veiklos įvykius, susiejant juos su planuojamos sistemos pateikimu. Toliau siekiama atskleisti duomenų mainus organizacijos viduje bei išorėje. Apžvelgus organizacijos politiką, strategiją, tikslus bei misiją, galima nustatyti potencialias veiklos taisykles, kurios vėlesnių etapų metu, bus transformuotos į realiai veikiančią sistemą.

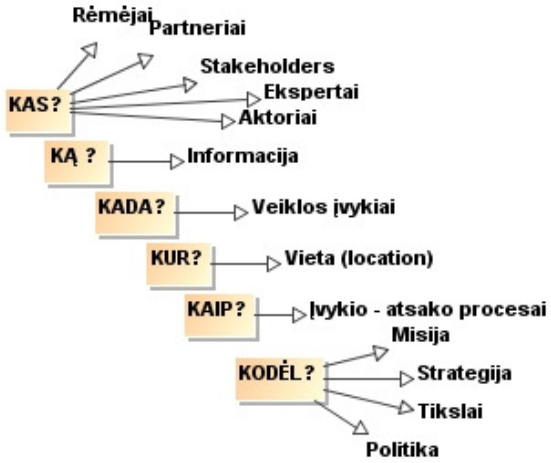
Išsiaiškinama kokiu tikslu, pateikiamas užsakymas, kodėl organizacija nori valdyti taisykles. Iš sistemos kūrėjų pozicijų, šio etapo metu, svarstoma kokia metodologija, saugyklos veiklos taisyklėms saugoti ir apdoroti, bus pasirenkama. Sprendimas priimamas remiantis pirminiais duomenimis apie organizaciją ir jos veiklą.

Šiame etape nustatomi reikalavimai apimantys:

- tinkamumą naudoti (*usability)*
- našumą *(performance)*
- prieinamumą (*availability)*
- talpumą *(capacity)* – kokį duomenų kiekį sistema apdoros, kiek žmonių naudosis sistema, kokia vartotojų dislokacija, kokį skaičių transakcijų vienu metu turi vykdyti sistema)
- Saugumą (*security)*

Be to dar nustatomos rizikos susijusios su techniniais bei verslo aspektais. Numatoma jų mažinimo strategija.

Apskritai apimties etape yra glaudžiai bendradarbiaujama ir dirbama su atsakingais asmenimis, veiklos šalininkais, veiklos ir technikos specialistais. Pagrindinis tikslas išsiaiškinti atsakymus į pagrindinius klausimus (14 pav.).



14 pav. *Apimties etapo pagrindiniai klausimai, į kuriuos reikia rasti atsakymus*

2.9.2.2. Planavimo etapas

Apimties nustatymo etapą keičia planavimo etapas. Tai tarsi paskutinė dalis prieš tai buvusio etapo. Čia sukuriamas projekto planas. Kaip bus kuriama sistema. Projekto plane išryškinami šie aspektai:

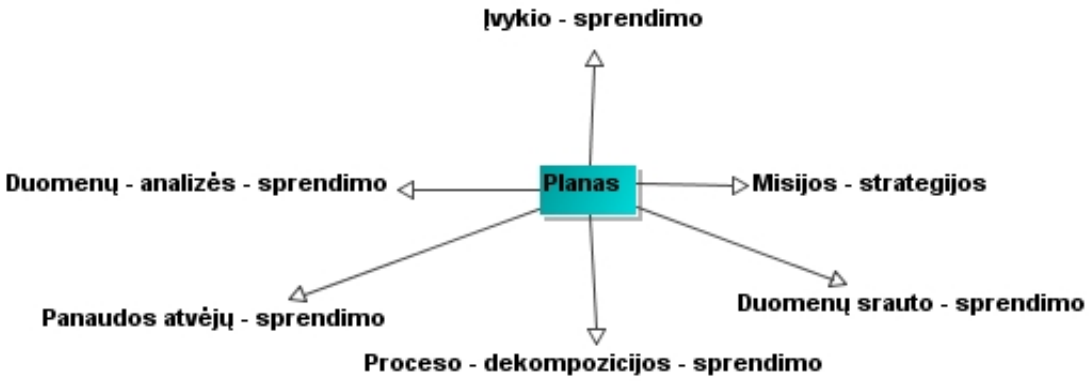
- Veiklos taisyklių atskyrimas nuo atskleidimo, analizes, projektavimo ir pateikimo etapų. Veiklos taisyklės pradedamos formuoti jau reikalavimų stadijoje.
- Taisyklių eigos protokolavimas nuo veiklos ištakų iki sistemos įgyvendinimo.
- Taisyklėms suteikiama konkreti forma. Taisyklės prieinamos per duomenų saugyklą.
- Taisyklių keitimams atlikti, pasirenkama taisyklėmis paremta technologija. Pavyzdžiui, veiklos taisyklių valdymo sistemos.

Į projektą įtraukiami taisyklių/veiklos analitikas, taisyklių/veiklos projektuotojas, taisyklių/veiklos realizuotojas, integruotojas. Analitikas atsakingas už taisyklių fiksavimą pokalbio metu, tarp užsakovo ir realizuotojo organizacijų. Dokumentų analizės ar esamo sistemos programinio kodo analizavimo metu. Projektuotojas apibrėžia kur taisyklės bus realizuojamos, kuriamos sistemos bendroje architektūroje. Realizuotojas transformuoja veiklos taisykles į vykdymo kodą, administruoja DB.

2.9.2.3. Atradimo etapas

Atradimo etape atskleidžiami detalūs reikalavimai, tačiau nepateikiama technologija, kaip bus įgyvendinti užfiksuoti reikalavimai. Naujų reikalavimų atskleidimas yra iteracinis procesas. Siekiant sukurti sistemą, kurioje galima įvesti naujus reikalavimus, šis etapas yra labai svarbus. Atradimo metu *planas,* tai aprašas, kurio laikomasi nustatant veiklos taisykles. Taisyklių nustatymo – valdymo – prognozavimo įrankiai (planas) gali būti kelių tipų (15 pav.). Planas pradedamas konkretaus veiklos įvykio aprašymu, pavyzdžiui, studento vidurkis lemia stipendijos dydį. Toliau fiksuojami sprendimai, kurie suteikia informacijos būsimoms taisyklėms. Pavyzdžiui, ar studentas yra žinomas? Ar vidurkis žinomas? Tuomet taisyklės apsprendžia įvykius. Pavyzdžiui, jei studento asmens kodas atitinka nustatytą formatą, tada studentas yra žinomas. Jei vidurkis yra daugiau už 0, tuomet vidurkis yra žinomas. Jei vidurkis yra žinomas ir studentas yra žinomas, tuomet jis gali/negali gauti stipendiją.

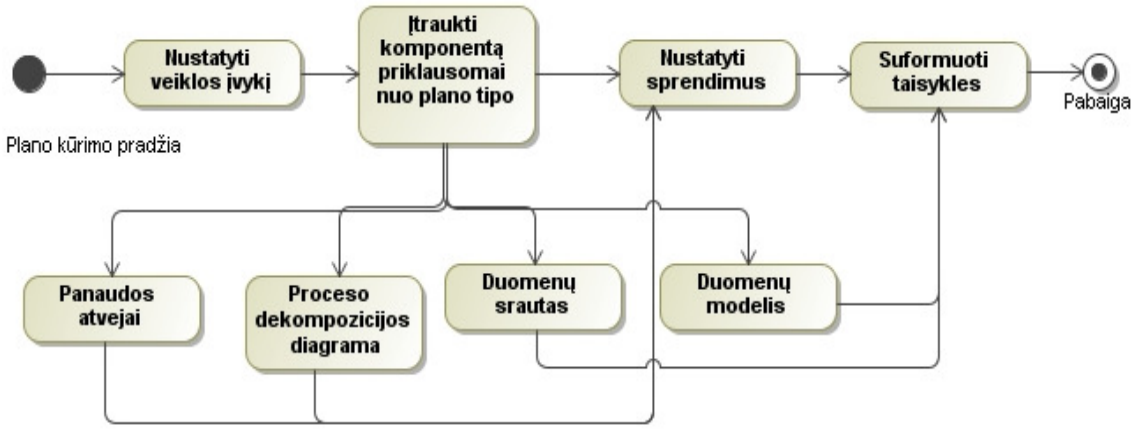
Priklausomai nuo pasirinkto plano tipo. Tarp veiklos įvykio ir sprendimo, įterpiamas konkrečiam planui priklausantis komponentas. Plano formavimo diagrama pateikiama 16 pav.



15 pav. *Plano tipai*

Pagrindinis tikslas išsiaiškinti pagrindinius reikalavimus. Dokumentuoti sistemos proceso aspektus taip, kad būtų įtraukiami šie punktai:

- Charakteringosios detalės prieš kiekvieną veiklos procesą.
- Sprendimai atliekami prieš veiklos įvykių sąveiką. Tai reiškia, jog užfiksuojamos detalizuotos taisyklės prieš priimant kokį nors sprendimą, proceso atžvilgiu.
- Remtis informacija priimant sprendimus.
- Žinios sukuriamos remiantis sprendimais.
- Paprastas realus arba tariamas įvykių scenarijus, ištestuoti proceso pilnumą.



16 pav. *Plano formavimo diagrama*

2.9.2.4. Analizės etapas

Analizės etapo metu nustatomi taisyklių prieštaringumai ir pertekliškumas. Šis etapas apima žingsnius, kurių metu pateikiama taisyklių priklausomybės seka. Nustatoma, kurie sprendimai ir esančios taisyklės asistuos organizacijai per būsimosios sistemos priimamus sprendimus, ar daromas išvadas. Pagrindinė šio etapo idėja – veiklos įvykio srautų strategijos atskleidimas. Kai sukurtoje sistemoje bus apdorojamos taisyklės, jos remsis tam tikra, šių etapų metu, nustatyta daline informacija ir pateiks naują dalinę informaciją, kuri vadinama *žiniomis (knowledge)*. Tokiu būdu pateikiamas rūpimo klausimo sprendimas.

Šis etapas taip pat apima taisyklių analizę ir jų grupavimą į aukštesnės abstrakcijos taisyklių rinkinius, kurie tam tikru aspektu apibendrina esamas taisykles. Sukuriamas taisyklėmis praplėstas loginis duomenų modelis. Jei organizacijoje egzistuoja jau sukurta sistema ir atliekamas jos tobulinimas arba kuriama nauja sistema, tai šiame etape iš esamo kodo nustatomos ir sudaromos egzistuojančios taisyklės. Kurios yra paslėptos po atitinkamu programiniu kodu.

Galiausiai visa informacija susijusi su taisyklėmis, pateikiama veiklos auditorijai. Kurie atlieka tobulinimą ir optimizavimą veiklos atžvilgiu. Vienas iš veiklos taisyklių metodikos privalumų tas, kad tinkamai sumodeliuotas taisykles, galima keisti ir vėlesnėse stadijose, pasitelkiant į pagalbą veiklos taisyklių valdymo sistemą. Tačiau nesvarbu kuriame etape evoliucionuoja sistemos kūrimo procesas, veiklos taisyklės turi išlikti neprieštaringos. Formuojamas duomenų modelis, kuriame nurodomi *sąvokos* ir *faktai.*

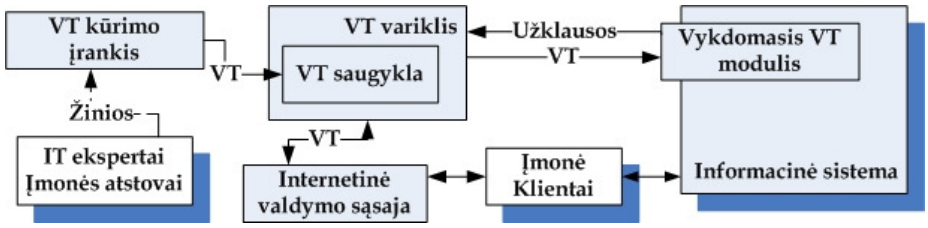
Apskritai šiame etape analizuojami duomenys, taisyklės ir procesai. Nuo konceptualaus jų modelio pereinama prie loginio modelio.

2.9.2.5. Projektavimo etapas

Projektavimo etapo metu klasifikuojamos taisyklės pagal tipus. Tam, kad jas būtų galima realizuoti. Pagrindinis tikslas transformuoti duomenis, taisykles ir atliktą proceso analizę į sistemos realų projektą. Kuris funkcionuoja ir kurį galima keisti. Taisyklių projektavimas suvokiamas kaip loginio taisyklių modelio transformacija į fizinį modelį. Loginis modelis susideda iš šablono, išreikšto natūralios kalbos sąlygomis ir faktais. Šiame etape pasirenkama veiklos taisyklių valdymo sistema. Nagrinėjamu atveju naudojama servisais – orientuotas FairIsaac kompanijos produktas Blaze Advisor.

Kaip pateikiama 2.7.1 skyriuje veiklos taisyklių valdymo įrankių yra platus pasirinkimas. Norint tinkamai įgyvendinti veiklos taisyklių principus bei atskleisti organizacijos procesus, per veiklos taisyklių tipus, tikslinga pasirinkti tinkamą taisyklių valdymo įrankį. Remiantis analize sistema turi įgalinti modeliuoti įvairaus sudėtingumo taisykles, kurias būtų galima ištestuoti bei integruoti į kuriamą IS. Integracija vyksta patogios vartotojo sąsajos pagalba. Kalba, kuria modeliuojamos įvairaus sudėtingumo taisyklės, turi būti artima natūraliajai kalbai. Šis aspektas palengvina veiklos taisyklių transformaciją iš reikalavimų etape užfiksuotų taisyklių į programinį kodą, kuriuo aprašomos veiklos taisyklės. Esant vienodos tipologijos, nesudėtingoms taisyklėms, galima naudoti veiklos taisyklių modeliavimo komponentus, tokius kaip sprendimų lentelės, sprendimų medžiai, taškų modeliai ir panašiai. Kadangi veiklos taisyklės remiasi dalykinės srities įvairiais vykstančiais procesais, kurie dažnai kinta, todėl veiklos taisyklių valdymas turi užtikrinti parametrų, pačių taisyklių išdėstymo, aktyvavimo/deaktyvavimo keitimo galimybę. Keitimus gali atlikti dalykinės srities atstovai, jeigu keitimo funkcija realizuota per internetinę sąsają. Principinė veiklos taisyklių valdymo įrankio schema pateikiama 17 pav. Nepriklausomai nuo dalykinės srities, IT ekspertai ir organizacijos atstovai bendradarbiauja tarpusavyje. IT specialistai identifikuoja reikalavimus bei kitus svarbius aspektus, aprašytus ankstesniuose skyriuose, kuriuos gauna iš įmonės atstovų. Tuomet veiklos taisyklės yra modeliuojamos VT kūrimo įrankyje. Iš jų vykdoma transformaciją į vykdymo kodą (paprastai pasirinktai taisyklių valdymo sistemai būdinga programavimo kalba) bei saugoma taisyklių saugykloje. Kuri yra autonomiškas elementas. Saugyklą gali pasiekti ir kitos duomenų saugyklos, kurios naudojasi čia talpinamomis taisyklėmis, kaip papildomais duomenimis, formuojant naujas veiklos taisykles. Taip pat saugyklą galima integruoti į taikomąsias programas. Veiklos taisyklių variklis apdoroja VT, kurios per vykdomąjį modelį integruojamos į informacinę sistemą. Vartotojui dirbant su sistema, tam tikru momentu kreipiamasi į taisyklių variklį, kuris atrenka reikiamas taisykles, kurios glūdi veiklos taisyklių saugykloje ir per vykdomąjį modelį grąžina

sistemai. Organizacijos atstovai gali taisykles modifikuoti internetinės valdymo sąsajos pagalba, jei tokia yra sukurta



17 pav. *Veiklos taisyklių integracijos IS principinė diagrama*

2.9.2.6. Pateikimo etapas

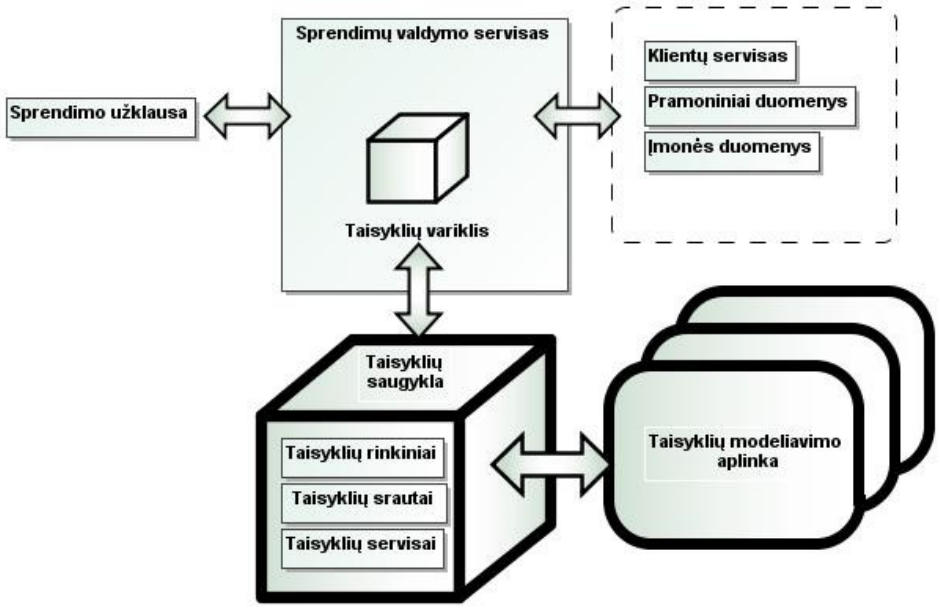
Apmokomi vartotojai kaip dirbti su sukurta sistema bei valdyti bei kurti veiklos taisykles.

2.10. VTVS lyginamoji analizė

2.10.1. Veiklos taisyklių architektūra pagal Blaze Advisor

Veiklos taisyklių valdymo sistema apibrėžia veiklos sprendimų logiką kaip taisykles, šios taisyklės yra įrašomos į taisyklių saugyklą. Naudojant VTVS aplinką, individualios taisyklės yra sugrupuojamos į taisyklių rinkinius. Taisyklių rinkiniai gali būti įkombinuojami į procesų sprendimų išvedimą.

Taisyklių srautų, rinkinių ir individualių taisyklių kombinacija yra naudojama taisyklių variklyje. Čia, pasitelkus į pagalbą spendimų logiką, yra apdorojamas konkretus veiklos procesas. Taisyklių servisai gali būti sukombinuoti ir įvykdomi sprendimų valdymo serviso viduje. VTV sistemos taisyklių variklis naudojamas sprendimų valdymui. Jis per taisyklių valdymo servisą apdoroja tam tikrą įvestį iš operatyvinės sistemos aplinkos. Gali būti apdorojami duomenys iš skirtingų duomenų šaltinių. Grąžinamas optimalus sprendimas į kliento taikomąją programą. BA VT sąveikos principas pateikiamas 18 pav.



18 pav. *Blaze Advisor veiklos taisyklių sąveikos principas.*

BA architektūros metodika, lyginant su tradicine architektūra, užtikrina lengvesnį priėjimą prie duomenų. Teoriškai tai reiškia, kad taip yra lengviau taikomąsias programas pritaikyti pagal asmeninius poreikius. Pritaikyti taikomąsias programas individualiam vartotojui bei jas plėtoti. Tradicinėje architektūroje, taikomosios programos vidinė logika, keičiama per vartotojo sąsają, tiesiogiai perduodant arba gaunant duomenis. Šiuo atveju duomenų sąveika pereina per taisyklių variklį ir saugyklą.

Servisų vidaus mechanizmas gali būti lengvai keičiamas nekeičiant jo interfeiso. Tai leidžia greitai įvykdyti pakeitimus, priklausomai nuo to kaip pasikeičia kampanijos veikla. Nereikia taisyti žemesnio lygio kodo. Tai labai aktualu toms organizacijoms, kuriuose esančios taisyklės dažnai keičiasi.

2.10.2.Veiklos taisyklių architektūra pagal JRules

JRules saugykla išskaidyta į du taisyklių saugyklų rinkinius. Vienas iš jų skirtas verslo atstovams, kitas kūrėjams, sinchronizuojamas per taisyklių vykdymo serverį (19 pav.).

- Taisyklių komandos serveris:

Veiklos atstovai valdo ir taiso taisykles naršyklėje. Kiekvienam suteikiamas skirtingas atsakomybės lygis, leidžiantis kontroliuoti priėjimus prie taisyklių. Tokiu būdu kontroliuojama kiekvieno vartotojo įtaka. Kiekvienas prisijungęs turi savo teisę su skirtingais vaidmenimis. Pavyzdžiui, asmuo, kuris gali kurti modelius, programuotojas, žinių inžinierius, verslo analitikas ir pan.

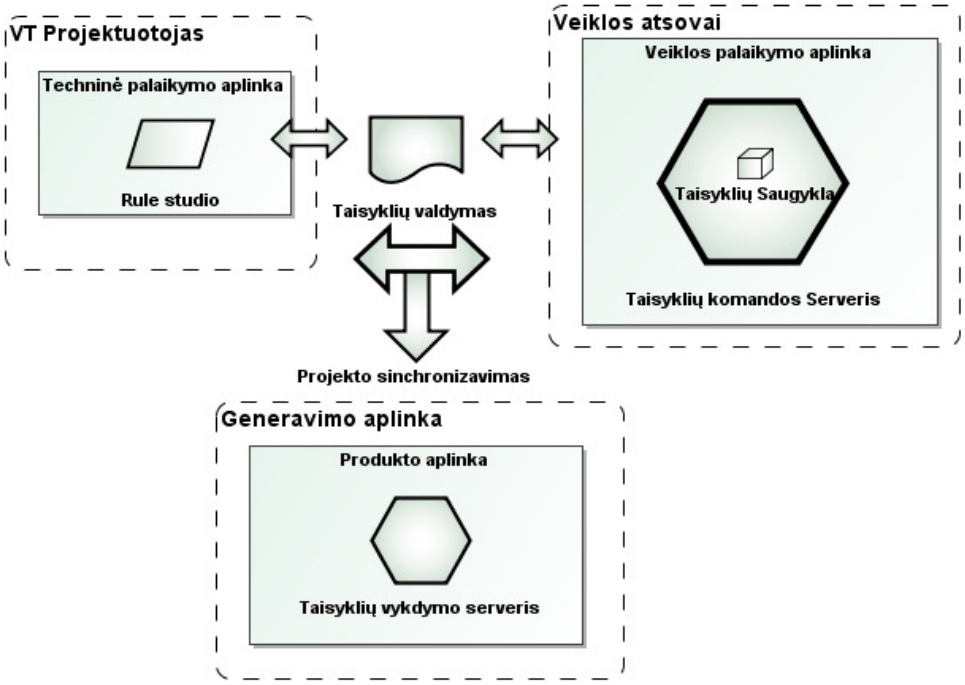
- Rule Studio

Aplinka skirta VT kūrėjams pasiekti VT.Tai konstravimo aplinka naudojama taisyklių taikymo kūrime. Ji leidžia integruoti Java kodo taisymą ir derinimą bei pačias taisykles. Rule Studio palaiko taisyklių rinkinių išdėstymą ir derinimą taisyklių vykdymo serveryje. Per Taisyklių Komandos Serverio kūrėjai bendradarbiauja su veiklos taisyklių autoriais.

- Taisyklių Vykdymo Serveris

Sistemos administratorius gali veiklos taisyklių vykdymą išdėstyti, tikrinti ir valdyti per Taisyklių Vykdymo Serverio administravimo internetinę aplinką.

Jis sujungtas su Rules Studio ir Taisyklių komandos Serveriu, kuris leidžia veiklos taisykles pasiekti tiek kūrėjam, tiek verslo atstovam.



19 pav. *JRules veiklos taisyklių sąveikos principas*

2.10.3.Kalbos naudojamos veiklos taisyklėms apibrėžti

2.10.3.1. Struktūriškai apibrėžta taisyklių kalba (Structured rules language) pagal Blaze Advisor

Blaze Advisor paketas naudoja SRL kalbą, kuri yra artima natūraliai kalbai. SRL kalbos naudojimą ir suvokimą verslo žmonės gali įsisavinti pakankamai lengvai.

Pagal kūrėjus SRL yra objektiškai orientuota kalba. Šios sintaksės pagalba galima kurti individualias taisykles ir taisyklių rinkinius, kurie yra lengvai suprantami tiems žmonėms, kurie mažai susidūrę su programavimu ir jo specifika. Integruota kūrimo aplinka

(IDE), vadinama Blaze Advisor Builder, apima pilną rinkinį redaktorių kuris supaprastina SRL objektų kūrimą ir sintaksės generavimą.

Blaze advisor paketas leidžia aprašyti taisykles keliais būdais, naudojant:

- natūralios kalbos metodą anglų kalba (SRL).
- sprendimų metaforas (sprendimų lenteles, sprendimų medžius bei taškų modelį.
- RMA (Rule maintence aplication).

Veiklos taisyklių kūrėjai gali naudoji Blaze Advisor su Java ir COM papildomai įsidiegiant įrankius, kurie pateiktų taisykles kaip web servisus (Java Beans, COM+ ir .NET). Tai metodas, kuris supaprastina integraciją su egzistuojančiomis taikomosiomis programomis, ir naujų įrankių panaudojimą. BA programa turi įrankius, kurie padeda valdyti ir manipuliuoti taisyklėmis, taisyklių saugyklą, kuri naudojama talpinti taisykles ir bendrai jas naudoti, XML pagalba. Taisykles galima taip pat išsaugoti duomenų bazėje ar į LDAP katalogą.

XML formatas naudojamas suderinant su SRL. Egzistuojančių taisyklių rinkiniai gali būti konvertuojami iš ir į XML.

Pagrindiniai požymiai.

SRL yra kalba susieta su XML vaizdavimu. SRL gali būti naudojama charakterizuoti šias kategorijas:

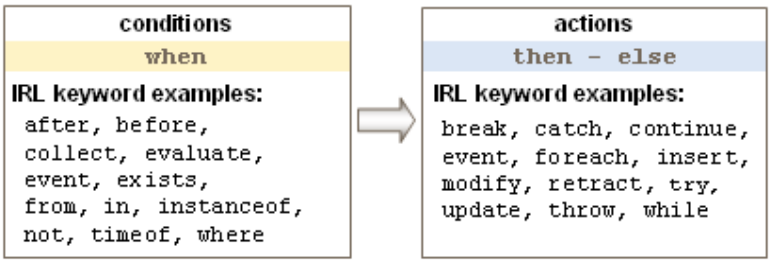
- Įvykių-sąlygos-vyksmo taisykles, veikiančias taisyklių serviso aplinkoje arba lokaliai taisyklių rinkinyje.
- Taisyklių pateikimas taisyklių rinkiniais .
- Turi funkcijas (t.y. procedūriniai algoritmai)(kartais gali būti susietos su biblioteka).
- Būtini: klasės, objektai, sąrašai ir kintamieji.

Blaze Advisor pateikia taisyklių srautą kaip WYSIWYG (tai ką matote atitinka tai ką gausite) taisyklių rinkinio mechanizmą.

JRules sistemoje naudojamos dvi kalbos konstrukcijos BAL ir IRL (Busines Action Language ir ILOG rule language). Pastaroji skirta techninių taisyklių specifikavimui. Technines taisykle JRules kūrėjai sieja su sąlygomis ir veiksmais. Kitaip tariant techninės taisykles sudaro sąlygos dalis ir veiksmo dalis.

Sąlygos dalis prasideda junginiu when, susiejančiu kintamuosius su objektų ir atributų reikšmėmis. Nurodo atributų reikšmių kriterijus. When veikia tarsi objektų filtravimo mechanizmas.

Veiksmo dalis, prasidedanti junginiu then, veiksmas yra įvykdomas, jei įvykdoma taisyklė. Taip pat čia esti ir kita dalis, kuri yra nebūtina konstrukcijoje, susiejanti su else išraiška. Ji vykdoma tada, kai atlikta užklausa gauna neigiamą atsakymą apibrėžtai sąlygai (20 pav.).



20 pav. *Veiklos taisyklių dalys jRules*

BAL kalba aprašomos įvairios konstrukcijos kurios naudojamos:

- Grupuoti taisyklių rinkinius.
- Išvedimų atvaizdavimui, spausdinimui.
- Egzistuojančių sąlygų rašymui.
- Skaičiuojamųjų sąlygų aprašymui.
- Sąlygų apribojimams kaip: where, in, from.

Atsižvelgiant į kalbų paskirtį būtų galima traktuoti, jog IRL skirta gramatikos specifikavimui, o BAL – atliekamų operacijų specifikavimui. JRules produktas paremtas Java kalbos pagrindu.

2.10.3.2. ILOG ir Fair kampanijų sukurtų kalbų palyginimas

Iš pateikto pavyzdžio galima matyti kokia sintakse aprašomos veiklos taisyklės(Lentelė Nr.1).

Jei vykdomas užsakymas viršija 100 LTL taikoma 3% nuolaida, jei užsakymas didesnis nei 100 LTL, bet mažesnis nei 1000 LTL nuolaida 6%, didesniems užsakymams taikoma 10% nuolaida.

Lentelė Nr. 1 VTVS kalbų sintaksė

Aprašymas	Blaze Advisor	JRules
Standartinis veiklos taisyklių konstravimas	`Rule bestDiscount is` **`if`**… **`then`**…	`Rule cheapPurchases{` **`when...then...`**`}`
Sąlygos dalis:	`If order < 100 then...` `If order>=100 and order<1000 then...` `If order>=1000 then...`	`When{IlrContext() from?context; Evaluate(orderValue<100);}then…` … `When{IlrContext() from?context; Evaluate(orderValue<=100&& orderValue<1000);}then…` … `When{IlrContext() from?context; Evaluate(orderValue>=1000);}then…`

Aprašymas	Blaze Advisor	JRules

Veiksmo dalis	If…then{
	Discount = 3
	}
	
	If…then{
	Discount = 6
	}
	If…then{
	Discount = 10
	}
		When{...}then{
		Discount = 3.0;
		updateContext();
		}
		When{...}then{
		Discount = 6.0;
		updateContext();
		}
		When{...}then{
		Discount =10.0;
		updateContext();}

2.11. Išvados

1. Atlikus veiklos taisyklių galimų struktūrų ir jų sudarymo būdų analizę, nustatyta, kad grafiniu būdu modeliuoti veiklos taisyklės gali tiek IT ekspertai, tiek verslo atstovai. Vis dėlto, šiuo metu egzistuojančios VT grafinio modeliavimo priemonės nėra lanksčios ir neužtikrina greito ir efektyvaus veiklos taisyklių sukūrimo būdo. Pagrindinės priežastys yra tos, kad netgi sudarant VT grafinėmis priemonėmis, verslo atstovai turi turėti programavimo žinių, tokiu būdu yra sulėtinamas jų darbas ar visiškai sustabdomas. Taip pat, kuriant panašios struktūros taisykles, jas reikia sukurti kiekvieną atskirai, dėl šios priežasties kūrimo laikas didėja priklausomai nuo taisyklių kiekio.

2. Išnagrinėta veiklos taisyklių formavimo etapų integracija informacinių sistemų kūrimo etapuose. Nustatyta informacija reikalinga veiklos taisyklės kūrimo inicijavimui bei sudarytas VT formavimo planas, tokiu būdu užtikrinamas nuoseklus jų realizavimas, atitikimas iškeltiems reikalavimams bei korektiškas vykdymas veiklos taisyklių valdymo sistemoje.

3. Atlikus egzistuojančių veiklos taisyklių valdymo sistemų analizę (Blaze Advisor, JRules), atliktas jų palyginimas, nustatyti esami privalumai ir trūkumai. VTVS naudojamos produkcinės taisyklės, kurių struktūra yra lengvai suprantama, jos dėka galima sudaryti įvairaus sudėtingumo VT. Nustatyti pagrindiniai trūkumai yra šie: nėra galimybių kurti daugkartinio panaudojimo veiklos taisykles, VT kūrime verslo atstovas negali savarankiškai kurti VT, vykdomų VT taisyklių koregavimas atliekamas kūrimo aplinkoje, t.y. jį atlieka IT ekspertas ar verslo atstovas, jei turi IT įgūdžių.

3. VEIKLOS TAISYKLIŲ SPECIFIKAVIMAS VTVS IR VARTOTOJŲ ROLĖS

3.1. Programų sistemos funkcijos

Sistemos modeliui realizuoti, buvo pasirinkta studijų administravimo dalykinė sritis. Veiklos taisyklėms realizuoti, pasirinktas Blaze Advisor veiklos taisyklių valdymo įrankis. Pagrindinis tikslas buvo atskleisti veiklos taisyklių specifikavimo, naudojimo ir modeliavimo principus. Taip iliustruojamas Blaze Advisor paketo galimybių koncepcinis modelis, kuris parodo, kaip galima sukurti sistemą veiklos taisyklių pagrindu.

Veiklos taisyklės, susijusios su studijų procesu, atskleistos per šiuos komponentus:

- Taisyklių srautai (ruleflows) - grafiškai pateikia vykdymo žingsnių seką sprendimų procese, naudojant įvykius, ciklus, šakas ir užduotis. Kiekviena užduotis taisyklių sraute apibrėžiama taisyklių rinkiniu, sprendimų lentele, funkcija arba šalutiniu taisyklių srautu.
- Sprendimų lentelės (rulesets) – tai peržvalgų lentelės, kur eilutės ir stulpeliai simbolizuoja skirtingas būsenas. Rezultato pateikimas ar gražinamos duomenų reikšmės yra apibrėžtos per jų sankirtą. Sprendimų lentelės duomenys gali būti importuojami iš išorinių šaltinių.
- Sprendimų medžiai – grafiškai vaizduoja sekas. Priklausomai nuo veiksmui pateikiamų sąlygų. Medžiai projektuojami derinant duomenų statistinę analizę.
- Taškų modeliai/įverčių modeliai (Scorecards) – specialios formos lentelės nagrinėjančios skirtingas objektų ypatybes ar charakteristikas, nustato įtaką reikšmių pagrindu. Įtakos kriterijai pateikiami remiantis kriterijų analize.
- Funkcijos – atliekančios algoritminį ir procedūrinį apdorojimą.

  Veiklos taisyklės saugomos taisyklių saugykloje, jos atskirtos nuo programinio kodo.

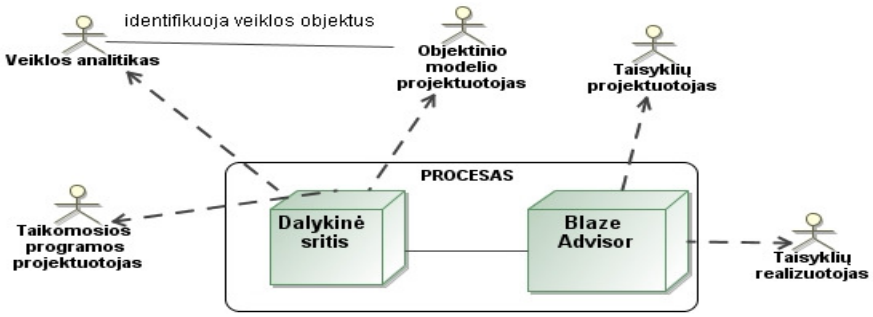
3.2. Sistemos vartotojai bei kūrėjai ir jų atsakomybės projekto kūrimo metu

Išskiriami kelių kategorijų vartotojai:

Žmonės, kurie nori susipažinti su veiklos taisyklių naudojimo galimybėmis Blaze Advisor paketo pagalba, bei suvokti kokius sprendimus galima priimti, kokius rezultatus galima gauti, naudojant veiklos taisykles. Kaip veiklos taisyklių metodu kurti sistemas.

Akademinė visuomenė. Šiuo atveju Kauno Technologijų Universiteto darbuotojai bei studentai.

Projektuojant veiklos taisykles, veiklos taisyklių valdymo sistema, idealiu atveju remiamasi kelių specialistų suformuotomis ir pateiktomis žiniomis bei išvadomis. Projekto realizavimo dalyviai pateikiami (21 pav.)



21 pav. *Projekto realizavimo dalyviai*

Veiklos analitikas atsakingas už dalykinės srities išsamų proceso apibrėžimą. Taisyklių sąrašo formavimą natūralia kalba. Taisyklės išreiškiamos aiškiai, pilnai ir glaustai. Nurodant faktus, strategijas ir procedūras, kurios reikalingos taisyklių įgyvendinimui. Jis nustato kuomet informacija yra prieštaringa, nepilna, neteisinga ir nereikšminga, formuoja veiklos strategijų aprašą. Pagrindinis tikslas surinkti aktualią informaciją iš įvairių šaltinių.

Veiklos analitikas bendradarbiauja su objektinio modelio architektu. Jiems aktualus veiklos objektų identifikavimo procesas, tačiau iš skirtingų aspektų. Iš veiklos analitiko perspektyvos, veiklos objektų rinkinys pilnai ir tiksliai apibūdina duomenų reikalavimus. Objektinio modelio architektui aktualu, kokius duomenis perduoti taisyklių servisui, ir kokią informaciją gaus iš serviso po apdorojimo. Pastarasis atsakingas už sistemos praplečiamumą, saugumą, palaikymą.

Objektinio modelio architektas identifikuoja veiklos objektus, kuriais remsis veiklos taisyklės. Atvaizduoja veiklos taisyklių elementų reikalavimus atitinkamiems duomenų šaltiniams.

Veiklos taisyklių projektuotojas atsakingas už projekto vystymą naudojant Blaze Advisor įrankį. Tai apima objektinio modelio vykdymo srauto importavimą ar projektavimą, taisyklių rinkinių ir funkcijų apibrėžimą.

Taisyklių realizuotojas atsakingas už detalizuotų taisyklių rinkinių rašymą bei Blaze Advisor programinio kodo konstrukcijų specifikaciją, įgyvendinimą, naudojant struktūrizuotą taisyklių kalbą (SRL Structure rule language). Jis tiesiogiai bendrauja su veiklos analitiku, tobulinant veiklos taisykles. Veiklos taisyklių realizavimas konkrečioje platformoje, remiantis

natūralia kalba apibrėžtomis taisyklėmis, gali sukelti įvairių pasikeitimų taisyklių atžvilgiu. Logiškumo, korektiškumo ir pan.

3.3. Veiklos taisyklių specifikavimo bazinės sintaksės architektūra

Šioje dalyje pateikiama bazinė veiklos taisyklių specifikavimo sintaksė. Aprašytos klasės, metodai, tipai naudojami tolesnio pavyzdžio „Studijų administravimas“ kūrimo procese.

Priklausomai nuo naudojamų komponentų skirtingai iškviečiami servisai, kurias VT sąveikauja su taisyklių saugykla. Kai naudojami taisyklių rinkiniai, sprendimų lentelės, sprendimų medžiai arba įverčių modeliai, iškviečiamas funkcinis objektas. Jis apibrėžtas taisyklių servise, sukonfigūruotas kaip serviso įėjimo taškas. Naudojant taisyklių srauto komponentą, naudojamas įvykio perdavimo (event post) serviso tipas – suaktyvinamas kai tam tikri įvykiai yra fiksuojami.

Resursai: Business Libraray – valdoma veiklos taisyklių kūrėjo per taisyklių palaikymo TP. Talpina klases. Technical library – techninės taisyklės, sukuriamos techninio vartotojo per IDE, šablonai, veiklos taisyklių tiekėjai (providers), SRL objektai.

Trijų tipų taisyklių serveriai (*stateless* – vykdomos taisyklės gražinamos vartotojui sinchroniškai. ASS(Asynchronous stateful server) – vartotojas siunčia užklausą taisyklių serveriui ir tęsia procesą nelaukdamas grąžinamo rezultato. SSS(Synchronous Stateful Server) – vartotojas pats gali nutraukti sesiją. Jų naudojimas priklauso nuo pasirinkto BA komponento.

Pateikiama bendra koncepcija, naudojama aprašant veiklos taisykles visuose komponentuose, jų klasių metoduose. Naudojama:

- Taisyklių rinkiniai
- Taisyklės
- Klasės
- Objektai
- Kintamieji
- Šablonai
- Funkcijos
- Įvykiai

Bendra koncepcija, naudojama BA projekte:

[Object_Model] ... [Rulesets] ... [Event_Rules] ... [Functions] ...

Skaičiavimams ir modifikavimams atlikti veiklos taisyklės naudoja duomenų reikšmes, kurios vadinamos *veiklos objektais.* Kiekvienas veiklos objektas apibrėžiamas klasės rinkiniu. Veiklos objektai, kuriuos veiklos taisyklės apdoroja, yra apibrėžtos klasės kopijos. Taisyklių servisai sudaro galimybę sąveikauti taikomosios programos duomenims, kurie perduodami iš kitų sistemų, tam yra naudojamas *Veiklos taisyklių objektinis modelis (BOM, Lentelė 14 prieduose).* Jis gali būti trijų tipų: Java veiklos objektinis modelis, DB veiklos objektinis modelis, XML veiklos objektinis modelis. BOM įgalina duomenų atvaizdavimą. Kai duomenys yra perduodami taisyklių servisui (paleidimo metu), atvaizdavimas ir konvertavimas į sistemą atliekamas automatiškai.

Jei nenaudojami kiti išoriniai duomenų šaltiniai, tuomet klasės apibrėžiamos tiesiogiai programoje. Klasė nurodo visus atributus, kuriuos apima objektai, taip pat kiekvieno atributo tipą. Sukūrus klasę ar ją importavus, jos turinio pagalba sukuriamas objektas, šablonas ar kintamieji, kurie naudoja pasirinktos klasės atributus. Šių elementų sintaksė pateikiama *14 lentelėje*. Kiekviena klasė išvedama iš tėvinės klasės, aukščiausios abstrakcijos klasė yra objekto klasė. Nauja sukuriama klasė identifikuojama *a KlasėsVardas*, nurodant tėvinės klasės vardą su raktažodžiu *an* (SRL sintaksės atveju).

Priskyrimo teiginyje arba metodo iškvietime, taisyklių kompiliatorius interpretuoja atributus ar aprašytus metodus, įtraukdamas objektus, kuriem priskirtos jau konkrečios reikšmės. Raktažodis *it* naudojamas iškviesti metodus objektams ir klasėms. Kiekvienu atveju metodas ar funkcija privalo grąžinti atitinkamo tipo reikšmę. Jei naudojamas daugiau nei vienas priskyrimo teiginys, visas priskyrimo deklaravimas apskliaudžiamas {}.

Aprašant sąrašą, kuris yra rinkinys su jam priklausančiomis konkrečiomis reikšmėmis, kurias jis gali įgyti esant apibrėžtoms sąlygoms. Sąrašo reikšmės gali būti naudojamos tiesiai taisyklėje, tiksliau taisyklės sąlygos dalyje. Rinkinių gali būti daug, tačiau juose esantys elementai turi būti unikalūs visos programos rėmuose.

Jei atsiranda poreikis, įvertinti visus prieinamus objektus, kurie yra vieno tipo, naudojant taisykles, tam naudojamas *šablonas (pattern)*. Šablonas apimą konkrečią klasę arba sąsają, kuris yra naudojamas pačioje taisyklėje Taip nurodoma taisyklių varikliu, jog reikia įvertinti kiekvieną objektą, kuris atitinka šabloną. Kuomet taisyklių variklis aptinka šabloną konkrečioje taisyklėje, jis traktuoja jį kaip sutrumpintos pateikimo formos notaciją. Šablonas gali būti globalus, prieinamas viso projekto ribose, arba lokalus, galiojantis tik konkrečiam taisyklių rinkiniui.

Taisyklės yra grupuojamas į taisyklių rinkinius, kurie turi loginę prasmę, jame esančių taisyklių atžvilgiu. Pasirenkant skirtingus taisyklių rinkinių vykdymo metodus, pagerinamas

taisyklių apdorojimo našumas. Išskiriami keli vykdymo būdai: optimizuojamosios išvados, standartinis, nuoseklus, kompiliacinis.

Optimizuojamos išvados atveju naudojamas Rete (III) algoritmas, išvadoms pateikti. Jis naudojamas tik taisyklių rinkiniams ir sprendimų išraiškoms. Standartiniu atveju naudojamas Rete algoritmas. Abu algoritmai naudojami dėl to, kad apdoroja didelį skaičių taisyklių, bei didelį kiekį duomenų.

Pasitaiko tokių situacijų, kai išauga objektų skaičius tam tikram rinkiniui taisyklių, taip pat, kai didelis skaičius taisyklių turi būti apdorotas per fiksuotą laiko tarpą ir Rete neužtikrina sklandų taisyklių keičiamumą. Tokiu atveju naudojamas nuoseklusis arba kompiliacinis apdorojimo būdai. Nuosekliuoju atveju, taisyklės yra vertinamos nuosekliai, nustatytų taisyklių prioritetu, naudojant nuoseklųjį algoritmą. Kiekviena taisyklės sąlyga yra įvertinama kiekvienai taisyklei.

Kompiliacinis apdorojimo būdas panašus į nuoseklųjį, tačiau lyginant su pastaruoju būdu, kompiliacinis padidina projekto našumą, nes yra sugeneruojamas Java - baitinis kodas, todėl duomenys apdorojami greičiau. Jei taisyklė remiasi ta pačia sąlyga, naudojama prieš tai įvertintos taisyklės, tuomet tos taisyklės sąlyga yra iš naujo įvertinama.

Taisyklių rinkinio *(Lentelė 15 prieduose)* pavadinimas identifikuojamas unikaliu pavadinimu. Aprašytų parametrų sąrašas naudojamas apibrėžti bet kokių reikšmių tipus ir pavadinimus, kurios naudojamos taisyklių rinkinyje. Šios dalies aprašo nebelieka jei taisyklių rinkinys neturi jokių parametrų. Egzistuojant taisyklių rinkinio turinio apibrėžimui, juo apibrėžiamos taisyklės, šablonai, kintamieji ar objektai yra lokalūs taisyklių rinkinio atžvilgiu. Todėl nėra pasiekiami už rinkinio ribų, kitiems taisyklių rinkiniams. Taisyklių rinkinio aprašą sudaro jame apibrėžto objekto, šablono, kintamojo, taisyklės, atributo/objekto/išorinio įvykio taisyklės aprašai. Taisyklių rinkiniuose esančioms taisyklėms nurodomas jų galiojimo laikas, t.y. kuriuo metu taisyklių variklis apdoros taisykles. Apribojimai yra dviejų tipų – laiko ir datos atžvilgiu. Paprastai taisyklės vykdomos nuosekliai, priklausomai kaip jos parašytos taisyklių rinkinyje, vykdymo tvarką galima keisti nurodant kiekvienos taisyklės prioritetus.

Išraiškomis vykdomi įvairūs skaičiavimai. Kuriuose naudojama viena ar daugiau priminių tipų reikšmių *(Lentelė 14 prieduose).* Taisyklėse išraiškos glūdi veiksmo arba sąlygos dalyje. Taisyklių variklis išraiškos rezultatą naudoja sąlygos ar veiksmo užbaigimui, toje išraiškoje kur taisyklė egzistuoja. Palyginimo išraiškos paprastai komponuojamos sąlygos dalyje, lyginant atributų, skaitines, logines bei string tipo reikšmes. Palyginimo operatoriai yra standartiniai, kurie egzistuoja bet kurioje programavimo kalboje. Jei išraiškoje yra

daugiau nei vienas operandas, visi operandai turi būti suderinami su pirmuoju operandu, t.y. to paties reikšmės tipo. Skaitinė išraiška pateikia skaitinį rezultatą *(Lentelė 16 prieduose).*

Įvertinimo išraiškoje *(Lentelė 16 prieduose)* grąžinama true arba false, kuomet apdorojama taisyklė. Ji naudojama ir funkcijose. Įvertinimo sąlygomis aprašomas loginis darinys, kuris tikrina ar egzistuoja tam tikras skaičius objektų, kurie priklauso atitinkamai klasei ar elementui, išraiškoje kuri yra apibrėžta. Taisyklės sąlygos ar funkcijos teiginiai įvertinami naudojant kiekinius raktažodžius, kurie pateikiami *(Lentelė 16)* . N žymi teigiamą sveikąjį skaičių, kuris gali būti lygus nuliui arba didesnis už jį. Kiekvienoje išraiškoje, išskyrus su raktažodžių every, nurodomas sveikasis skaičius, uždedantis atitinkamus apribojimus išraiškai. Satisfy dalis pažymi predikatą, kuris aprašomas logine išraiška Such that dalis apima loginės išraiškos apribojimus, kuriuo uždedami apribojimai apibrėžtam predikatui. Jei šios dalies nėra, taisyklių variklis tikrina predikatą su visais klasės objektais arba rinkinio elementais.

Palaikomos klasės, kurios sutiekia taisyklių specifikavimui lankstumo, yra kelių kategorijų, kurios pateikiamos *(Lentelė 19 prieduose).*Pasitaiko tokių veiklos taisyklių, kuriuose svarbų vaidmenį vaidina data, laikas, trukmė. Tam naudojamos klasės, kurių metodai palengviną darbą su šiomis struktūromis. Pavyzdžiui, norime sužinoti kokia buvo data 2008 metais, aštunto mėnesio, trečios savaitės pirmadienis. Naudojama funkcija *monthWeekDay(),* kuri konvertuoja šį aprašą į datos formatą atitinkamai *August 10, 2008*. Funkcija *yearWeekDay()* pateikia konkrečią datą, kai žinomi metai, ir kelinta metų savaitė.

Datos formatas gali būti kelių tipų: *pilnasis* (savaitės diena, Mėnuo diena, metai), *ilgasis* (mėnuo diena, metai), *sutrumpintas* analogiškas ilgajam, tačiau mėnesio pavadinimas inicializuojamas trimis simboliais, kurie atitinka konkretaus mėnesio inicialus, *trumpasis* (mm/dd/yy) data išreiškiama skaičiais nurodytu formatu.

Išskiriami trys laiko formatai: *pilnasis, tarpinis, trumpasis.* Pirmojo forma (valanda: minutės: sekundės AM/PM GMT – laikas), antrojo (valanda: minutės: sekundės AM/PM), trečiojo (valanda: minutės AM/PM).

Money tipo naudojimas taisyklėse, leidžia atlikti įvairius skaičiavimus ir palyginimus naudojant pinigines išraiškas, kurios gali būti apibrėžiamos bet kokia valiuta. Taip pat atlikti kelių skirtingų valiutų konversiją.

Veiklos taisyklių teisingumas patikrinamas, naudojant assert klasės palaikomus metodus. Kurie naudojami palyginti laukiamą ir gautas reikšmes. Šios išraiškos naudojamos testavimo įskiepyje brModule *(Lentelė 19 prieduose).*

Taškų modeliai ir priežasčių kodai, kurie priklauso taškų modeliams, turi klases pateiktas *(Lentelė 19 prieduose).* NdReasonCode klasė apima rinkinį atributų, kurie palaiko priežasčių kodus. Jie pateikia paaiškinimą, kodėl po taisyklės vykdymo buvo gautas tam tikras skaičius taškų. Klasė NdScoreCharacteristic pateikia rinkinį atributų, kuriuos palaiko kiekviena charakteristika taškų modelyje, o klasė NdScoreModelReasonCalculationOptions – atributus, kurie palaiko priežasčių kodo skaičiavimo nustatymus taškų modelyje.

Priklausomai nuo to, koks taškų modelio būdas taikomas, gaunami skirtingi apskaičiuoti rezultatai. Galima skaičiuoti pagal vartotojo nustatytus pradinius taškus arba pagal maksimalųjį tašką. Jei tas pats sprendimų kodas priskiriamas kelioms charakteristikoms, skaičiuojama pagal dubliavimo metodą.

3.4. Bendri principai BA elementų identifikavimui

Kadangi veiklos taisyklės aprašomos struktūrine taisyklių kalba (SRL), kuri yra artima natūraliajai kalbai, reikia laikytis tam tikrų taisyklių ir jų elementų specifikavimo principų.

Specifikuojant objektą, kiekvienam iš jų reikia priskirti unikalų identifikatorių, kurį sudaro vienas ar daugiau simbolių. Simboliais gali būti didžiosios ir mažosios raidės. Jos traktuojamos kaip skirtingi simboliai, didžioji ir mažoji raidė nėra identiškos. Pabraukimas „_“ gali būti naudojamas, jei identifikatorius yra sudėtinis, t.y. jį sudaro daugiau nei vienas žodis. Neleidžiami tarpai ar kiti simboliai.

Kintamieji identifikuojami raktiniu daiktavardiniu žodžiu. Esant sudėtiniam kintamojo pavadinimui, pirmoji kiekvieno žodžio raidė rašoma didžioji. Taip, bendrajame kontekste, atskiriamas taisyklės ir kintamojo pavadinimai.

Konstantoms specifikuoti taip pat naudojamas daiktavardis su visomis didžiosiomis raidėmis. Jei konstantą sudaro keli sudėtiniai žodžiai, jie sujungiami „_“ simboliu tam, jog būtų galima suvokti koks konstantos tikslas sistemos kontekste.

Funkcijų identifikavimui galioja anksčiau išvardinti apribojimai, tačiau identifikavimo raktažodžiu tikslingiausia pasirinkti veiksmažodį. Funkcija atspindi veiksmus, todėl tikslingiausia jį identifikuoti šiuo sintaksiniu dariniu.

Identifikuojant pačias veiklos taisykles, reikia vengti ilgų jų pavadinimų. Pirmoji žodžio raidė turėtų būti didžioji, tai pagerintų skaitomumą bendrajame kontekste, kodo dalyje. Tačiau taisyklės identifikacijai galima naudoti ir sutrumpinimus, kurie turėtų būti visuotinai priimtini ir suprantami, simbolizuotų vieną ar kitą prasmę.

Nesilaikant šių pagrindinių principų, esant dideliam skaičiui taisyklių, sunku jas valdyti tiek programiniame lygmenyje tiek viso projekto rėmuose.

3.5. Veiklos taisyklių realizavimo komponentai Blaze Advisor pakete.

Veiklos taisyklių realizavimo komponentų pagalba veiklos taisyklių realizavimas ir palaikymas užtikrinamas netechniniam sistemos vartotojui, t.y. asmenims, kurie neturi programavimo žinių, o geba nagrinėti tik dalykinę sritį ir joje egzistuojančius veiklos procesus. Paprastai tai būna atitinkamos dalykinės srities dalyviai, kurie kontroliuoja, kuria vykstančius veiklos procesus tam, kad būtų pateikiami tam tikri sprendimai, priklausomai nuo jų pačių suformuotų sąlygų.

Kiekviename komponente naudojamos sąlygos ir veiksmo dalys, tačiau jų realizacija ir galutinio rezultato išvedimo metodai yra skirtingi. Komponentų naudojimas yra tikslingas, kuomet manipuliuojama dideliu skaičių veiklos taisyklių, kurios viena su kita siejasi analogiška struktūros forma (logine prasme).

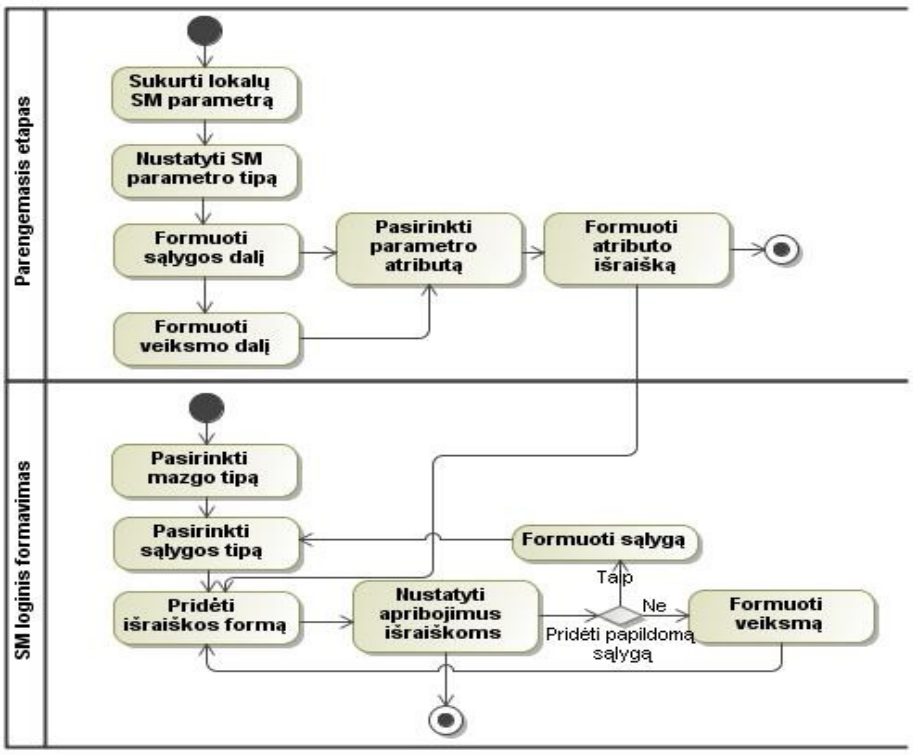
Sprendimų medis

Vienas iš būdų veiklos taisyklėms atvaizduoti ir susistemintai pateikti vartotojui yra *sprendimų medžiai.* Formuojami gerai žinoma medžio struktūra. Medžio šakojimas pradedamas nuo tėvinės sąlygos, kurią naudoja iš jos išeinančios šakos su papildomomis sąlygomis. Sprendimų medžio kūrimo eigos procesas pateikiamas 27 pav. Kiekvienos šakos pabaigoje, nepriklausomai kiek ta šaka turi sąlygų, pateikiamas veiksmo aprašas. Realizuojamas grafine viršūne, vaizduojančia trikampio formos figūrą. Sprendimų medžio duomenų reikšmių pildymo teisingumo tikrinimas yra vykdomas kiekviename mazge. Tai sumažina klaidų atsiradimo tikimybę programos vykdymo metu. Prieš naudojant šį komponentą, yra svarbu apibrėžti, kokio tipo rezultato laukiama po vykdymo. Grąžinami tipai gali būti trijų tipų: priminiai duomenų tipai, rinkinys, sudėtinis tipas (objektas arba masyvas). Perduodamos reikšmės, kuriomis remiantis sprendimų medis pateiks rezultatus, apibrėžiami kaip parametrų reikšmės arba šablonai.

Kiekvienas sprendimų medis turi turėti bent vieną sąlygos šaką, kuri apima vieną ar daugiau išraiškų, naudojančių tuos pačius parametrus, arba parametrų atributus, arba šablonus. Kiekviena išraiškos forma turi skirtingus apribojimus reikšmėms.

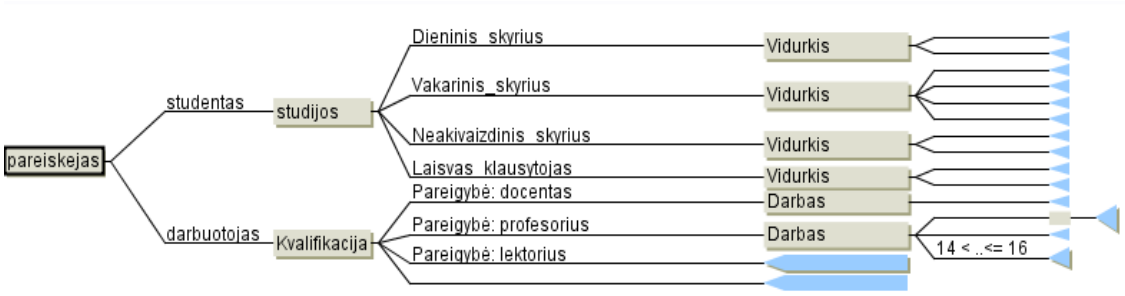
Parametras identifikuojamas informatyviu pavadinimu. Tipo parinkimas priklauso nuo to, kokius atributus norima naudoti sprendimų išvedime. Sąlygos formavimui naudojami sukurto parametro tipo atributai. Kiekvienas atributas, esantis sąlygos arba veiksmo dalyse, identifikuojamas konkrečia logine išraiška, kurioje nurodomi apribojimai tipui. Paprastai esama tokia forma *theParametras.atributas <operatorius> reikšmė.* Kiekvienam atributui galima nustatyti kelias išraiškas. Jos leidžia keisti operatorių tipus bei redaguoti jam priskirtas

reikšmes. Išraiškų nustatymas pradedamas dar parengiamajame etape, kuomet yra nurodomi sąlygos arba veiksmo atributai, kurie bus naudojame sprendimų medžio loginiame formavime. Sprendimų medžio loginis formavimas yra sąlygos mazgų seka, kurių kiekviena užbaigiama veiksmo mazgu. Kaip buvo minėta ankščiau, sprendimų medis turi turėti bent vieną sąlygos mazgą. Sąlygos mazgo tipo pasirinkimo variacijos leidžiamos dar parengiamajame etape, priskirtų atributų rėmuose. Vienas sąlygos mazgas gali turėti tik vieną sąlygos tipą, su jam priskirtomis išraiškomis. Esant papildomam apribojimui, kuriama papildoma medžio šaka, kuri išeina iš prieš tai buvusio sąlygos mazgo kaip alternatyva tam mazgui. Išraiškose nurodomi konkretūs apribojimai sąlygai, su realiomis reikšmėmis arba reikšmių intervalais. Kiekviena medžio šaka, su savo identifikuotais mazgais, yra sąlygų seka, kuri užbaigiama veiksmo aprašu.



27 pav. *Sprendimų medžio kūrimo eigos procesas*

28 pav. pateiktas pavyzdys, kuriame realizuotas paskolos suteikimas, naudojant sprendimų medį. Paskolos pareiškėjas gali būti studentas arba dėstytojas. Jeigu pareiškėjas studentas, tuomet vertinimo kriterijai yra studijų tipas bei vidurkis. Studijos ir vidurkis yra sąlygos mazgai. Nustatomas paskolos tipas, kurį gali gauti pareiškėjas, jeigu atitinka apibrėžtas sąlygas. Paskolos tipas bei jos dydis yra veiksmo mazgai. Analogiškai vertinamo dėstytojo duomenys. Sąlygos parametrai yra išdirbtas laikas bei pajamų dydis.



28 pav. *Paskolos dydžio nustatymas naudojant sprendimų medį*

Sprendimų medžio sukūrimas yra taisyklių rinkinių sukūrimas, kuriuose egzistuoja vieninga parametrų identifikavimo sistema. Tai yra, sukūrus mazgą „pareiškėjas“, yra sukuriami du taisyklių rinkiniai: *_nNodeCond* ir *_nNode*. Čia n yra sveikasis skaičius, identifikuojantis atitinkamus taisyklių rinkinius unikaliu pavadinimu. Pirmasis rinkinys apibrėžia mazgo sąlygą. Antrajame yra sąlygos, apimančios prieš tai buvusio mazgo sąlygą ir esamo mazgo sąlyga (_nNodeCond). Antrajame rinkinyje, then taisyklės dalyje, yra specifikuojama sprendimų medžio šakos veiksmo dalis. Kitaip tariant sprendimo medžiuose egzistuojančios taisyklės yra paveldimos. Tai reiškia, jog taisyklės sąlygos gali remtis kitos taisyklės sąlygų prielaidomis, kurios yra toje pačioje šakoje.

Sprendimų lentelė

Sprendimų lentelė yra grafinis veiklos taisyklių pateikimo būdas, kuriame veiklos taisyklių sąlygos ir veiksmo dalys išdėstomos lentele. Patogu naudoti, kai reikia valdyti didelį kiekį veiklos taisyklių. Lentelės struktūra pagal pateikimą yra trijų tipų. *Eilutėmis grįsta lentelė, stulpeliais grįsta* ir *sudėtinė lentelė,* kurioje naudojamos pirmos dvi rūšys. Kad ir koks būtų lentelės tipas, veiklos taisyklės į saugyklą talpinamos kuriant vienodos struktūros taisyklių rinkinius. Kūrimo eigos procesas analogiškas sprendimų medžio kūrimo procesui. Kiekviena lentelė turi turėti bent po vieną sąlygos langelio grupę. Tačiau esminis skirtumas tarp sprendimų lentelės ir sprendimų medžio yra veiklos taisyklių rinkinio turinio generavimas. Čia viena lentelės eilutė atitinka vieną taisyklę, sukurtajame taisyklių rinkinyje. Taisyklėje talpinami visi sąlygos ir veiksmo aprašymai. Taisyklių grafinį įrašą atitinkantį kodą, galima atrasti tik pagal taisyklei sugeneruotą pavadinimą, kuriame atsispindi veiklos taisyklės numeris. Pirmoji taisyklė laikoma nuline. Taisyklės yra generuojamos tokia tvarka, kokia jos apibrėžtos lentelėje.

Taisyklių srautas

Taisyklių srauto komponentas vizualiai pateikia viso arba dalies projekto vykdymo seką. Skirtingais atvejais, priklausomai nuo veiklos taisyklių turinio, paskirties, kiekio naudojami skirtingi komponentai, kurie per taisyklių srauto komponentą apjungiami į visumą rezultatui pateikti. Taisyklių srautas identifikuojamas pradžios ir pabaigos mazgais, o tarp jų formuojami veiklos procesai, kurie apibrėžiami užduoties mazgais. Jais gali būti sprendimų medžiai, lentelės, taisyklių rinkiniai, taškų modeliai ar funkcijos. Išskiriamos galimos srauto kontroliavimo variacijos, naudojant skaidymą, paralelinį - sąlyginį skaidymą (šakos, kuriuose apibrėžiamos užduotys, vykdomos paraleliai. Joms naudojamos sąlyginiai apribojimai.) ir paralelinį besąlyginį skaidymą. Įvykio laukimo mazgas įgalina proceso sustabdymą tol, kol bus gaunama norima reikšmė procesui pratęsti.

Taškų modelis (įverčių modelis)

Taškų modelio komponentas (kaip ir sprendimų lentelė ar medis) pateikia grafiškai veiklos taisykles. Tačiau nuo minėtų komponentų turi esminį skirtumą. Tai daugiau matematinių formulių pateikimas, skirtas prognozuoti laukiamiems rezultatams ateityje, remiantis veiklos dalyvių pateiktomis žiniomis dabartyje. Žinomi tam tirki faktai yra transformuojami į taškų modelio *charakteristikas*, kurios gali būti skaitinės reikšmės, loginiai taip ir ne, klasė ar alternatyvų sąrašas. Charakteristikos traktuojamos kaip kintamieji. Kiekviena charakteristika turi vieną ar daugiau sąlygų. Kurioms identifikuoti naudojamas pavadinimas, sąlygos forma, balų kiekis. Pastarasis priskiriamas, kai tenkinama atitinkama sąlyga bei kategorija (*Rason code).*

Įverčių modelyje yra vertinami vienas ar keli kintamieji. Pagal tam tikras sąlygas įvertinus kintamojo reikšmę yra priskiriamas atitinkamas balų skaičius. Įvertinus kelis kintamuosius, galutinis rezultatas yra visų įvertinimų balų suma.

Kartais vertinant kintamąjį yra patogiau jį įvertinti ne balų skaičiumi, o kitu, žmogui labiau suprantamu, kriterijumi. Kriterijai apibrėžiami *kategorijomis*, kurios apibrėžiamos *NdReasonCode* klase. Jos atributai pateikiami *Lentelė Nr. 19 prieduose.* Įverčių modelio charakteristikos sąlygos forma priklauso nuo kintamoji tipo. Bendru atveju, nepriklausomai nuo to, koks yra tipas, struktūrą sudaro *charakteristika, operatorius, reikšmė.*

Taigi kitas šio komponento skirtumas tas, kad čia nustatomi atributai ir reikšmės taisyklių sąlygoms. Taisyklių veiksmas yra automatiškai sugeneruojamas, naudojant įvestus taškus taškų skiltyje. O sprendimų lentelių ar medžio atveju, reikšmės apibrėžiamos tiek sąlygos tiek veiksmo dalims.

Taškų modelyje atliekamas automatinis validavimas:

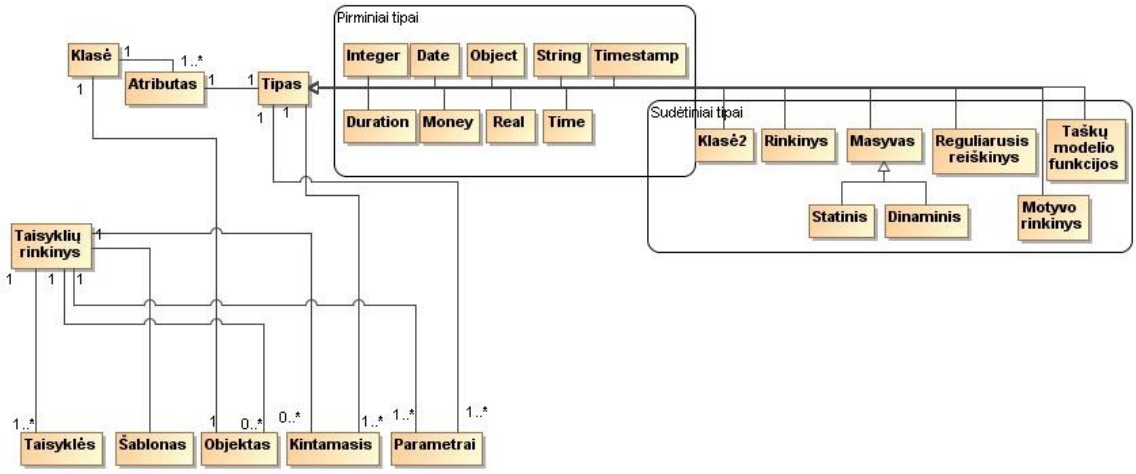
- Įvestiems duomenų tipams
- Persidengiančioms reikšmėms
- Besidubliuojančioms išraiškoms
- Apribojimo reikšmių nuoseklumą (Žemesnė apribojimo reikšmė yra didesnė negu aukštesnio apribojimo reikšmė).

Nepriklausomai nuo to, koks skaičiavimo būdas, galutiniam rezultatui gauti yra grąžinamas objektas, kurio tipas yra *NdScoreModelReturnInfo*. Šio objekto atributai saugo informaciją apie įverčių modelį. Objekto atributai pateikiami *Lentelė 14 prieduose*.

Atlikus analizę ir realizavus veiklos taisykles Blaze Advisor palaikomais komponentais pastebėta, jog jais galima realizuoti veiklos taisykles pusiau automatizuotu būdu. Kuomet naudojamas automatinis kodo generavimas iš komponentuose esančių duomenų. Juose galima keisti atitinkamas duomenų reikšmės, tačiau keisti komponento struktūros ar net visos jo vykdymo logikos negalima. Tam reikia lankstesnio metodo veiklos taisyklėms realizuoti. Suteikiančio joms betarpišką manipuliavimo galimybę, kuomet galima keisti ne tik duomenų reikšmes, bet ir užfiksuoto proceso logiką. Tokiu būdu palengvinti priėjimą prie veiklos taisyklių neprogramiame lygmenyje, o įmonės atstovams, kurie dirba su taisyklėmis Sekančiame skyriuje pateikiama šablonas ir jo metodika, kuriuo pagalba suformuotas veiklos taisykles gali administruoti tiek IT ekspertas, tiek įmonės atstovas.

4. VEIKLOS TAISYKLIŲ SPECIFIKAVIMO ŠABLONAIS METODIKA

4.1. Vidiniai VTVS elementai bei jų kardinalumai



29 pav. *VTVS elementų kardinalumai*

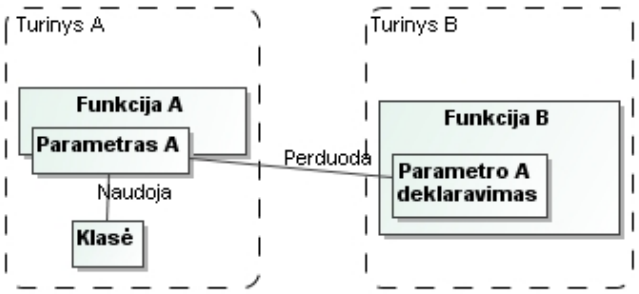
Pagrindinis veiklos taisyklių elementas yra klasė. Jos vaidmuo tolimesnėje kūrimo eigoje neatsiejamas nuo bendrojo konteksto (29 pav.). Klasė atspindi veiklos procesą, kurio turinys aprašomas veiklos taisyklėse. Klasėje yra bent vienas atributas su tik tam atributui būdingu tipu. Tipai skirstomi į sudėtinius ir pirminius. Sudėtiniai tipai yra išvedami iš pirminių. Hierarchiniu požiūriu kuriamoji klasė išvedama iš pagrindinės VTVS objektinės klasės. Jei sukurtosios klasės tipas yra kita klasė, tarp jų egzistuoja paveldėjimas, kurio metu paveldimos atributų reikšmės. Objektas yra fundamentalus klasės atvaizdavimas.

Objektas vykdymo atžvilgiu yra *statinis* arba *dinaminis*. Prieinamumo atžvilgiu išskiriami *lokalūs* ir *globalūs objektai*. *Globalus objektas* yra sukuriamas taisyklių rinkinio galiojimo srityje arba funkcijoje. Todėl tik juose esantys teiginiai ir išraiškos gali pasiekti ir pakeisti objekto reikšmes.

Statinis objektas sukuriamas taisyklių rinkinyje, kuris paveldi tik vienos, jam priskirtos, klasės atributus. Dinaminiai objektai yra sukuriami vykdymo metu, jų sukūrimo iniciavimas egzistuoja programiniame kode, t.y. sukuriamos duomenų struktūros, kurios taisyklių vykdymo metu sukuria laikiną objektą ir priskiria konkrečias nurodyto tipo reikšmes. Pastarosios įvedamos kaip parametrai taisyklių atžvilgiu. Nepriklausomai nuo to kokio tipo yra objektas, jo turinį sudaro atributai. Išankstinis objekto inicializavimas, leidžia priskirti atributams konkrečias reikšmes, kurios perduodamos vykdymo metu. Vykdymui pasibaigus objektas yra sunaikinamas. Lokalūs objektai, kurių galiojimo sritį apibrėžia taisyklių rinkinys ir įvairios jos formos, galioja tol, kol yra grąžinamas konkretus atributas. O globalus objektas galioja tol, kol baigiama veikos taisyklių vykdymo sesija. Šį objektą galima

pasiekti visoms išraiškoms, teiginiams, taisyklėms, taisyklių rinkiniams ar funkcijoms, kurios sukurtos konkretaus projekto ribose.

Parametrai yra argumentai arba reikšmės, kurios perduodamos taisyklių rinkiniams, funkcijoms ar kitiems sprendimų išvedimo elementams, kad būtų galima atlikti apibrėžtus veiksmus. Parametras deklaruojamas kuomet sukuriama funkcija arba taisyklių rinkinys, o parametro perdavimas įvyksta kuomet iškviečiama funkcija arba taisyklių rinkinys.



30 pav. *Parametro vaidmuo bendrajame VTVS kontekste*

Sukuriamas parametras_A funkcijai_A (30 pav.). Parametras_A naudoja klasę, kitaip tariant pasiekia klasės atributus, kurie aprašomi funkcijoje_A, atitinkamam loginiam turiniui pasiekti. Parametro perdavimas funkcijai_B atliekamas kuomet tos funkcijos turinyje iškviečiama funkcija_A. Nurodant tos funkcijos lokalų parametrą_A ir jo tipą. Šiuo atveju parametro tipas yra klasė. Funkcijos turinys šiai daliai sudaromas iš vykdymo iniciavimo, kuris gali būti execute arba apply (priklausomai ar tai yra funkcija ar taisyklių rinkinys), funkcijos pavadinimo, parametrų susiejimo formos. Pastaroji inicializuojama raktažodžiu with:

execute *Funkcija_A* ***with*** *{ parametras_A =* ***an*** *klasė}*

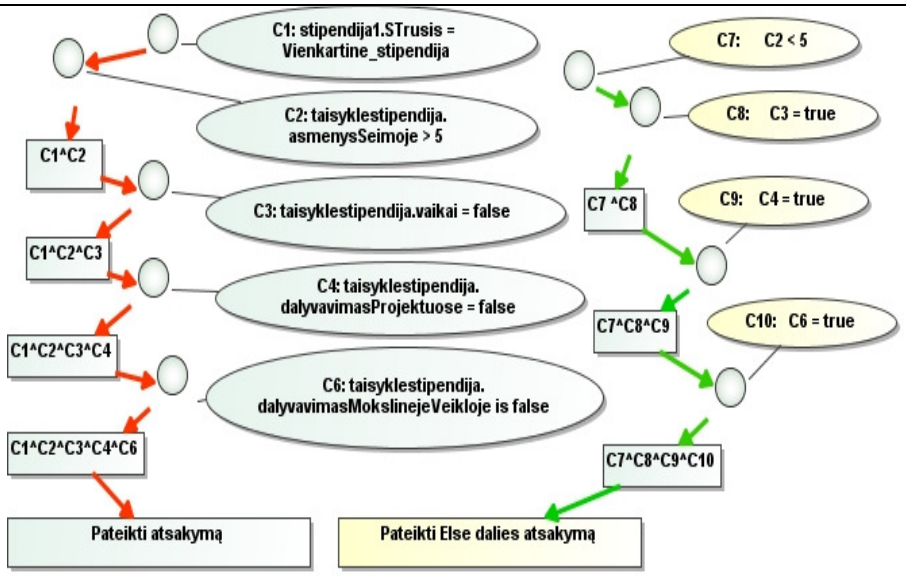
Kintamojo sukūrimas (viso projekto galiojimo srityje) taisyklių vykdymo metu, leidžia keisti jo reikšmes, kuomet jis priskiriamas teiginyje arba iškviečiamas funkcijoje. Galimas kintamojo inicializavimas konkrečia reikšme, kuri priklauso nuo kintamojo tipo. Globalaus kintamojo gyvavimo trukmė projekte prasideda jo aprašymo metu ir baigiasi kuomet taisyklių variklis baigia projekto turinio apdorojimą. Lokalaus kintamojo gyvavimo ciklas baigiasi kuomet yra baigiamas vykdyti SRL kodas, kuriame tas kintamasis apibrėžtas. Todėl lokalus objektas labiausiai tinka saugoti laikinąsias reikšmes, kuomet yra vykdomas SRL kodas.

Taisyklėse sukuriamas šablonas tam, kad taisyklių variklis įvertintų kiekvieną objektą, kuris atitinka apibrėžtą šabloną. Kitaip sakant, taisyklių variklis traktuoja šabloną per taško notaciją, kuomet įvertina kiekvieną objektą pagal aprašytą taisyklę ar kelias taisykles, kurios apjungtos taisyklių rinkiniu. Taisyklių rinkinio vykdymo metu šablonas esti dinaminis. Taisyklės yra įvertinamos iš naujo.

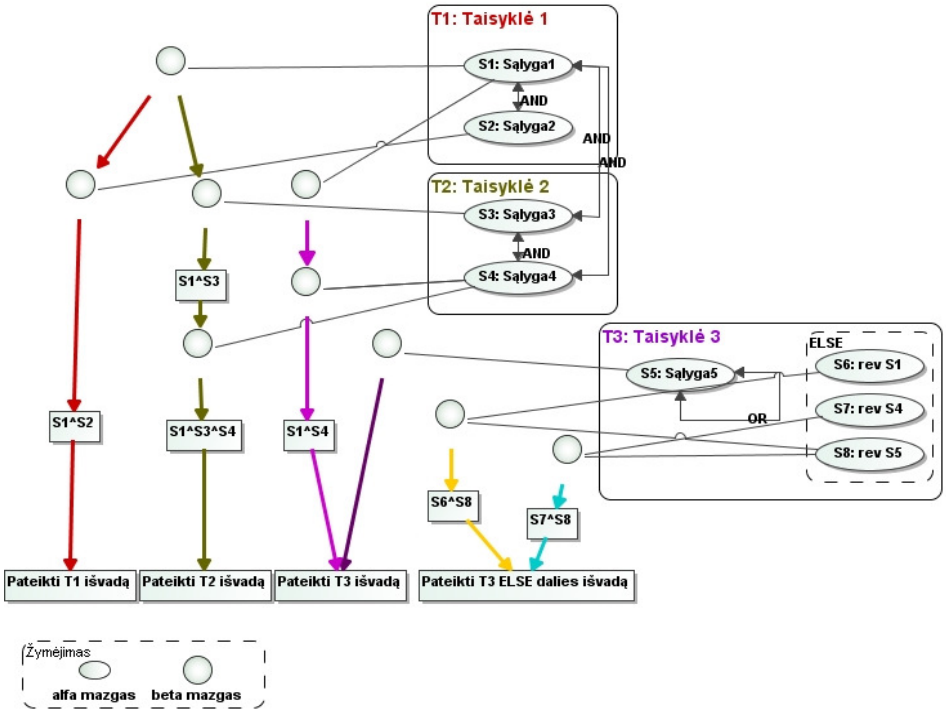
4.2. Veiklos taisyklių apdorojimo procesas vykdymo metu

Kaip buvo minėta anksčiau, VTVS (Blaze Advisor) taisyklių variklio apdorojimo procesas paremtas Rete algoritmu. Taisyklių viduje stebimi pasikeitimai duomenų atžvilgiu, kurie aprašomi naudojant 4.1 skyriuje pateiktais komponentais. Esant dideliam skaičiui taisyklių, kurios bendrai naudoja tas pačias sąlygas arba egzistuoja taisyklės, kurių suaktyvinimas priklauso nuo kitų taisyklių vykdymo sekos, duomenimis paremtas apdorojimo būdas, įgalina efektyvų veiklos taisyklių valdymą ir apdorojimą. Apdorojimas ir suaktyvinimas paremtas tinklo koncepcija. Taisyklių servisas yra suaktyvinamas veiklos taisyklių apdorojimo metu. Taisyklių sąlygos transformuojamos į tinklo mazgus, kurie sudaro seką, priklausančią nuo veiklos taisyklės logikos. Kuomet taisyklės sąlyga yra tenkinama, ji dedama į pagrindinį operatorių sąrašą, kuris turi būti suaktyvintas ir apdorotas. Kiekviena taisyklės sąlyga tinkle yra *α –mazgas*, o kiekviena sąlygų konjunkcija yra *β – mazgas*. Sąlygų konjunkcija – sąlygos, sujungtos „and“ operatoriumi. Kiekvienos taisyklės sąlygos sujungiamos į grandines, naudojant mazgus. Disjunkcija – sąlygos sujungtos loginiu „ar“. Ji sudaro atskiras grandines tinkle. „Else“ taisyklių dalis traktuojama kaip papildomos taisyklės, mazgai. Yra generuojami kaip kiekvienos loginės išraiškos, esančios sąlygos dalyje, neigimas. Pateikiamas realus pavyzdys *(Lentelė 2),* ir jo vykdymo iliustracija.

Pateikiama konceptuali veiklos taisyklių vykdymo schema (31 pav.), naudojant Rete algoritmą. Kiekvienos taisyklės sąlygos dalyje apibrėžiama bent viena sąlyga. Esant kelioms sąlygoms, jos jungiamos loginiais „ir“ „arba“ operatoriais, nuo kurių priklauso, kaip veiklos taisyklės bus vykdomos. Pateiktoje diagramoje pirmąją taisyklę sudaro dvi sąlygos sujungtos loginiu „ir“. T2 taisyklės pirmoji taisyklės sąlyga analogiška T1 sąlygai S1. Antroji sąlyga S3, trečioji – S4. T3 taisyklė, priešingai nei T2 ir T3, turi alternatyvaus veiksmo inicializuojamo „else“ dalį. T3 sąlygos dalis sudaryta iš S1, S4 sąlygų, kurias jungia loginis „ir“ bei S5, kuri prijungiama kaip alternatyvi sąlyga operatoriumi „arba“. Ši dalis apdorojama atskirai nuo prieš tai minėtų. Paprastai operatoriaus „else“ galiojimo srityje apibrėžiamas alternatyvus veiksmas. Vykdomas jei netenkinamos prieš tai apibrėžtos sąlygos. Šios dalies turinys išvadoms pateikti, analizuojamas kaip sąlygų neigimas.

Vykdymo procesas	

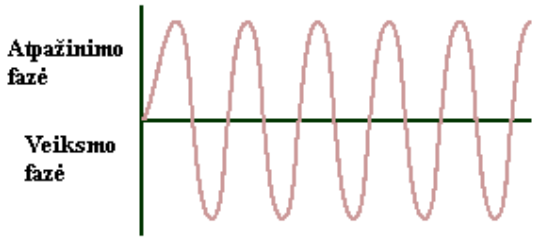
Kodas	*if*
	(stipendija1.STrusis is Stipendijos_rusys.Vienkartine_stipendija
	and(taisyklestipendija.asmenysSeimoje > 5)
	and(taisyklestipendija.vaikai = false)
	and(taisyklestipendija.dalyvavimasProjektuose = false)
	and(taisyklestipendija.dalyvavimasMokslinejeVeikloje = false)
	)
	then {STPtaisyklesParametras.spausdinimostringas = "veikia".
	}
	else
	print("NEGALIMA PRETENDUOTI I Stipendijos_rusys.Vienkartine_stipendija STIPENDIJA.Neatitinka pasirinktos stipendijos gavimo kriteriju").



31 pav. *Bendra koncepcinė veiklos taisyklių vykdymo schema*

Taisyklių apdorojime išskiriamos du etapai, tai *atpažinimo* ir *veiksmo*. Atpažinimo etape kiekvienas tinklo mazgas yra tikrinamas ar jis tenkina sąlygą (arba kelias sąlygas). Kai tos pačios sąlygos mazgai yra naudojami skirtinguose grandinėse, tai reiškia, jog ta pati sąlyga galioja keliuose taisyklėse. Šitokio apdorojimo būdo privalumas yra tas, kad duomenys, atitinkantys konkrečius mazgus, atpažinimo etape, tikrinami tik vieną kartą. Jei visi mazgai, esantys grandinėje yra tenkinami, tuomet taisyklė talpinama į vykdymui ruošiamų išraiškų sąrašą, kuris apdorojamas po kiekvieno atpažinimo etapo. Aukščiausią prioritetą turinčios taisyklės yra suaktyvinamos ir visi veiksmo teiginiai yra įvykdomi. Atpažinimo etapą keičia veiksmo etapas. Alfa mazgai yra paveikiami duomenų, veiksmo etapo metu.

Veiksmo etapo rezultatas – paveikti mazgai, kurių poveikį įtakojo įvesti duomenys. Tinkle alfa mazgai perduoda signalą iš eilės beta mazgams. Jog jais remiasi pateikiant atitinkamus veiksmo rezultatus. Taisyklių variklis tęsia ciklą tol, kol nebelieka veiklos taisyklių. Ciklas praeina atpažinimo ir veiksmo etapus. Grafiškai butų galima pavaizduoti ciklą taip, kaip pateikta 33 pav.



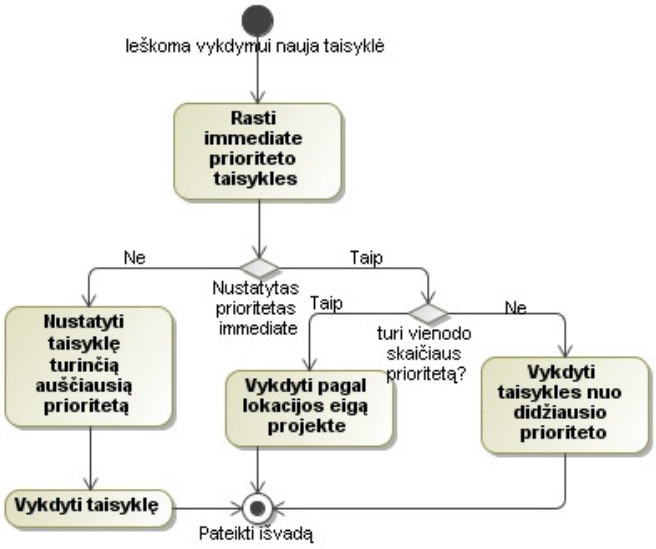
32 pav. *Veiklos taisyklių apdorojimo ciklo etapai*

Taisyklių variklis tikrina taisyklių prioritetus. Jie suteikiami veiklos taisyklių apdorojimo eiliškumui keisti. Aukščiausią prioritetą turinčios taisyklės, talpinamos į vykdymui ruošiamų išraiškų sąrašą. Aukščiausią prioritetą turi *immediate* taisyklės, kurių apdorojimas atliekamas prieš bet kurį prioritetą. Esant kelioms taisyklėms, turinčioms šį aukščiausios abstrakcijos prioritetą, jos visos apdorojamos prieš kitą atpažinimo etapo sužadinimą tokia eilės tvarka, kokia buvo realizuotos projekte. Tai reiškia, jog taisyklių variklis apdoroja kelias taisykles tame pačiame veiksmo etape.

Prioritetas yra skaičius, veiksmo etapo metu nurodantis, taisyklių apdorojimo eiliškumą bendrojoje veiklos taisyklių eigoje. Prioritetas gali būti teigiamas ar neigiamas sveikasis skaičius. Taisyklių vykdymo eiga, prioritetų atžvilgiu, pateikiama 33 pav.

34 pav pateiktas taisyklių variklio elgsenos ciklas, kurio metu taisyklės išrenkamos iš vykdymui paruošto taisyklių sąrašo. Veiklos taisyklių eiliškumas turi įtakos bendriems

sistemos konteksto rezultatams. Prioritetų suteikimo mechanizmas, įgalina veiklos taisyklių apdorojimo eigos valdymo kontroliavimą.



33 pav. *Veiklos taisyklių atranka vykdymui, taisyklėse turinčiose prioritetus*

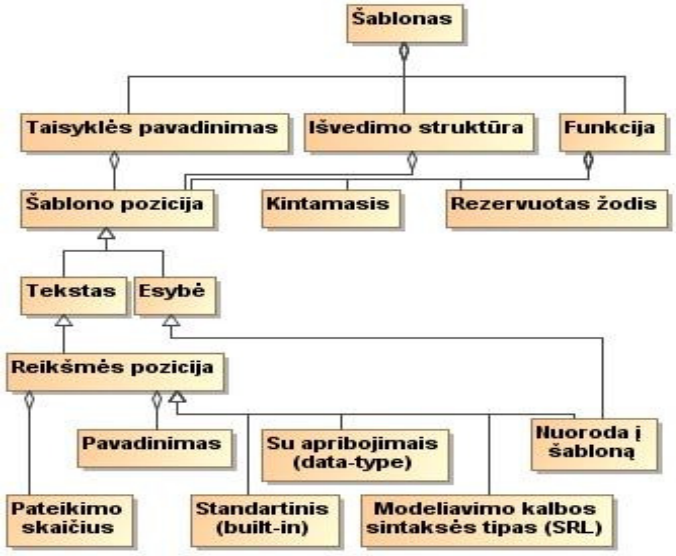
34 pav. *Taisyklių variklio elgsena per apdorojimo procesą*

4.3. Veiklos taisyklių specifikavimas, naudojant šablonus

4.3.1. Veiklos taisyklių šablono struktūra

Remiantis atlikta analize padaryta išvada, jog veiklos taisyklių kūrimas naudojant VTVS komponentus yra pakankamai lankstus būdas specifikuoti veiklos taisykles grafiniais būdais. Kuomet aiškiai apibrėžtos taisyklės sąlygos ir laukiami rezultatai. Tačiau toks VT specifikavimo būdas nėra lankstus, todėl kitos struktūros VT reikia kurti tiesiai per VTVS įrankį. Tam reikalingi specialistai, kurie turėtu ne tik dalykinės srities žinių, tačiau išmanytų ir programavimo subtilybes. Siekiant sumažinti atotrūkį tarp veiklos atstovo ir VTVS programinio realizavimo, siūloma veiklos taisykles specifikuoti kruopščiai sumodeliuotais šablonais.

Pasirinktas VTVS modeliavimo įrankis turi internetinę sąsają (RMA – *Rule maintenance application)*, kuri įgalina veiklos analitikams kurti, redaguoti veiklos taisykles intuityviu būdu, negalvojant kaip tos veiklos taisyklės turi būti sumodeliuotos, kad būtų galima gauti atitinkamas išvadas.



35 pav. *Veiklos taisyklių formavimo šablono struktūra*

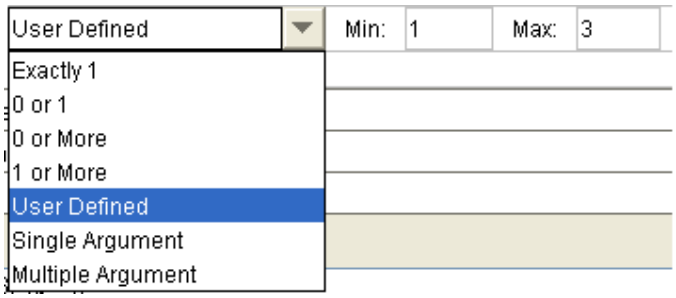
35 pav. pateikiama šablono formavimo struktūra, kurioje atsispindi jos elementai. Pagrindiniai šablono elementai yra *Taisyklės pavadinimas*, *Išvedimo struktūra* ir *Funkcija*. *Taisyklės pavadinimas* yra skirtas identifikuoti taisyklę. Kuriant taisyklę, jis gali būti sugeneruojamas automatiškai, jei jame nurodomas kintamasis arba jai suteikta galimybė analitikui pavadinimą keisti. *Išvedimo struktūra* yra sudaryta iš statinio teksto, kurį suformuoja analitikas, ir iš dinaminių elementų – *Šablono pozicijų*. *Šablono pozicija* gali būti dviejų tipų: *Tekstas* ir *Esybė*. Tipas priklauso nuo *Reikšmės pozicijos*. *Reikšmės pozicija* identifikuojama pavadinimu, pateikimo skaičiumi bei tipu. Tipą sukurti gali VT projektuotojas, pasirinkdamas standartinius tipus (*built – in provider)*, tipus su papildomais apribojimais (*Data Type provider)*, konkrečios kalbos sintaksės tipus (*SRL provider*), tipu gali būti nuoroda į kitą šabloną.

- *Standartiniai tipai:* and-or, boolean, date, duration, integer, localized time, localized timestamp, money, real, string, systemproperty, time, timestamp, typevalues. Jie yra naudojami nurodant reikšmės pozicijos tipus, kai nereikia papildomų apribojimų. Pavyzdžiui, veiklos analitikas formuoja taisyklę per RMA, profesoriaus pareigybėms užimti. Dalyje parašyti vadovėliai, reikšmė gali būti tik integer tipo, naudojant integer standartinį tipą, užtikrinamas reikšmės teisingumo įvedimas.
- *Tipai su papildomais apribojimais:* yra tų pačių tipų kaip ir standartiniai tipai, tačiau juose galima nustatyti papildomus apribojimus, priklausomai nuo nustatyto tipo.
- *SRL tipai:* And-Or, klasės atributų, klasių, sąrašo reikšmių, sąrašo, funkcijų, įvardintų objektų atributų, projektų, atributų tipų, taisyklių rinkinių, SRL atributų, SRL atributų

tipo, Operacijų tipų, Tipų reikšmių. Jie leidžia veiklos analitikui per RMA pasirinkti iš SRL sukurtų provedrių reikšmių. Pavyzdžiui pasirinkti klasę iš visų klasės sąrašų, atributus, rinkinius, taisyklių rinkinius ar funkcijas.

- *Pagalbiniai tipai:* asosiacijos, duomenų bazės, failo, bendrasis, sankirtos, valdymo atributų pavadinimų, valdymo atributų tipo, valdymo atributų reikšmių, tipo nuorodos, sistemos atributų, šablonų sąrašo, šablono nuorodos, reikšmių sąrašo tipai. Naudojami sukurti laukus, kurie veiklos analitikui, naudojant RMA leidžia pasirinkti tam tikrą reikšmę iš nurodyto reikšmių sąrašo. Šias reikšmes galima perduoti per DB, reikšmių sąrašą ar nuorodas.

Viena reikšmės pozicija gali turėti tik vieną tipą. Nustačius leistiną konkrečios reikšmės pozicijos pateikimo kiekį (angl. *count)*, formuojamas šablono struktūros dydis. Pozicijos pateikimo kiekio variacijos pateikiamos 36 pav, Lentelė Nr.3 pateikiama detalizacija. Kai tam tikra reikšmė yra įvedama reikšmės pozicijai, tipas tikrina ar ta reikšmė atitinka teisingą semantiką.



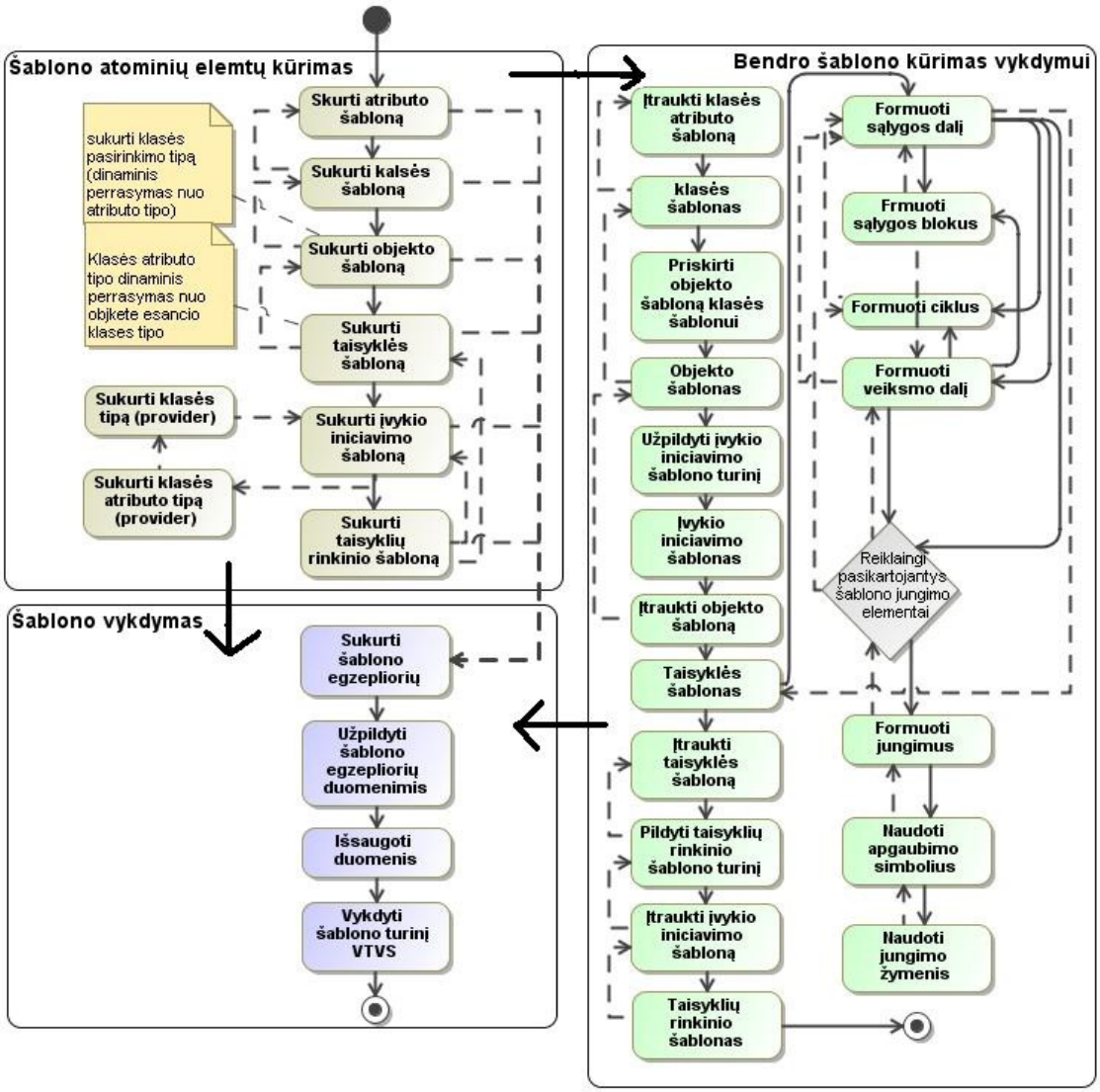
36 pav. *Reikšmės pozicijos pateikimo kiekio variacijos*

Lentelė Nr. 3 *Reikšmės pozicijos kiekio pateikimo įtaka reikšmės pozicijai*

Jei naudojama...	**Įtaka reikšmės pozicijai**
exactly 1	reikšmės pozicija vienu metu gali turėti tik vieną reikšmę
0 or 1	Gali turėti 1 arba neturėti jokios reikšmės
0 or More	Gali turėti be galo daug arba nei vienos reikšmės
1 or More	Turi vieną arba daugiau reikšmių
0 or More-No Duplicates	Pateikia visas nustatyto sąrašo reikšmes. Jei pridedama daugiau nei viena, neleidžiamas reikšmių dubliavimas. Galima taikyti tik toms reikšmių pozicijoms, kurios apibrėžia sąrašo tipą su atitinkamomis reikšmėmis
1 or More-No Duplicates	Reikšmės pozicija privalo pateikti bent vieną sąrašo reikšmę arba daugiau. Panašu į *0 or More-No Duplicates*
User Defined	Reikšmės pozicija turi galiojimo sritį, kuri apibrėžta maksimaliu ir minimaliu sveikuoju skaičiumi.
Single Argument	Pateikia šablono argumentą su vienintele reikšme
Multiple Argument	Pateikia šablono argumentą su sudėtinėmis reikšmėmis

Sukurtos reikšmės pozicijos į bendrą šablono kontekstą integruojamos per *Teksto šablono poziciją* (*text placeholder.)* Viena teksto šablono pozicija gali apibrėžti tik vieną reikšmės poziciją. Jeigu reikšmės pozicija yra kitas šablonas, tuomet šablono funkcijoje galima naudoti *Esybės šablono poziciją* (*entity placeholder)*. Tuomet funkcijos turinys bus šablonas, kuris nurodytas esybių šablono pozicijoje. Funkcijos turinyje, priklausomai nuo šablono tipo, gali būti kintamieji, identifikuoti statiniais ar dinaminiais pavadinimais, rezervuoti žodžiai, naudojami funkcijai realizuoti (standartiniai tam tikros veiklos taisyklių specifikavimo kalbos sintaksiniai dariniai – *if, then, equal* ir t.t.).

4.3.2. Veiklos taisyklių šablono sudarymo algoritmas



37 pav. *Veiklos taisyklių specifikuotų šablonais kūrimo algoritmas*

Veiklos taisyklių kūrimo algoritmas (37 pav.) susideda iš trijų dalių. Tai šablono atominių elementų sukūrimo, bendro šablono vykdymui kūrimo ir sukurto šablono vykdymo. Kiekviena dalis sąveikauja tarpusavyje. Realizavus šablono atominių elementų kūrimo dalį,

pereinama prie bendro šablono kūrimo vykdymui, kuriame naudojamas pirmosios dalies bendrasis modeliavimo rezultatas. Antroji dalis naudojama trečiajai daliai (šablono vykdymas). Norint sukurti atskirus atominių elementų šablonus vykdymui pereinama iš pirmosios dalies prie trečiosios dalies (šablono vykdymas).

1. *Šablono atominių elementų kūrimas:*
   1.1. Sukuriamas atributo šablonas.
   1.2. Sukuriamas klasės šablonas. Kad sukurti pilną klasės šabloną, reikalingas sukurtas atributo šablonas, kuris sudaro klasės šablono turinį.
   1.3. Sukuriamas objekto šablonas. Šį šabloną sudaro klasės šablonas, o ją atributo šablonas.
   1.4. Sukuriamas įvykio iniciavimo šablonas.
      1.4.1. Sukuriamas klasės tipas.
      1.4.2. Sukuriamas atributo tipas.
   1.5. Sukuriamas taisyklių rinkinio šablonas. Jame naudojamas taisyklių šablonas, su jam priklausančiais kitais šablonais bei įvykio iniciavimo šablonas.

Nuo 1.1 iki 1.5 galima sukurti kiekvienam atskirą šablono egzempliorių, tuomet pereinama į šablono vykdymo algoritmo dalį, kuriame atliekami šie veiksmai:

2. *Šablono vykdymas*
   2.1. Sukuriamas šablono egzempliorius.
   2.2. Užpildomas sukurtas egzempliorius duomenimis.
   2.3. Išsaugomi duomenys saugykloje.
   2.4. Vykdomas, šablone sudarytas turinys, veiklos taisyklių valdymo sistemoje.

Vienam šablonui sukurti, kurį būtų galima vykdyti iš vieno egzemplioriaus, atliekami veiksmai pagal bendrąjį šablono kūrimo vykdymui algoritmą.

3. *Bendras šablono kūrimas vykdymui*
   3.1. Į klasės šabloną įtraukiamas atributo šablonas
   3.2. Į objekto šabloną įtraukiamas klasės šabloną. Automatiškai įtraukti klasės atributo šabloną.
   3.3. Sukūrus įvykio iniciavimo šabloną, užpildomas jo turinys
      3.3.1. naudojant klasės tipą
      3.3.2. klasės atributų tipą
      3.3.3. sudaromas klasės ir atributų dinaminis perrašymas.
   3.4. Taisyklių šablone formuojamos dalys:
      3.4.1. Veiksmo dalis. Esant poreikiui naudojama:
         3.4.1.1. Suformuoti ciklai

3.4.1.2. Suformuoti sąlygos blokai. Juose gali būti ir 3.4.1.1, 3.4.1.3, 3.4.1.4

Formuojant šablono loginį turinį tam tikri pasikartojantys elementai formuojami per junginius. Nurodant:

3.4.1.3. Apgaubimo simbolius

3.4.1.4. Jungimo žymenis. Sujungia elementus nustatytu simboliu, kurie yra tarp nurodytų apgaubimo simbolių.

3.4.2. Sąlygos dalis

3.4.2.1. Jei reikalingi pasikartojantys jungimo elementai atliekami veiksmai 3.4.1.2, 3.4.1.3, 3.4.1.4.

3.5. Taisyklių rinkinio šablonas formuojamas atlikus 3.4, 3.3, 3.2, 3.1 žingsnius

3.5.1. Įtraukiamas taisyklės šablonas.

3.5.2. Remiantis taisyklių šablono sudedamosiomis dalimis, pildomas taisyklių rinkinio šablono turinys.

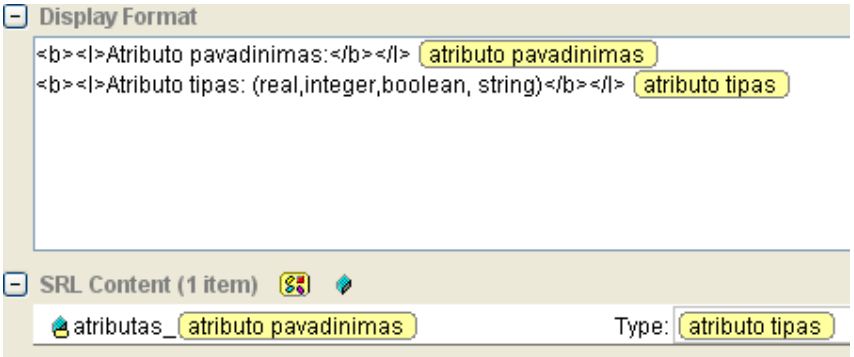
3.5.3. Įtraukiamas įvykio iniciavimo šablonas.

3.5.4. Įtraukiamas objekto šablonas.

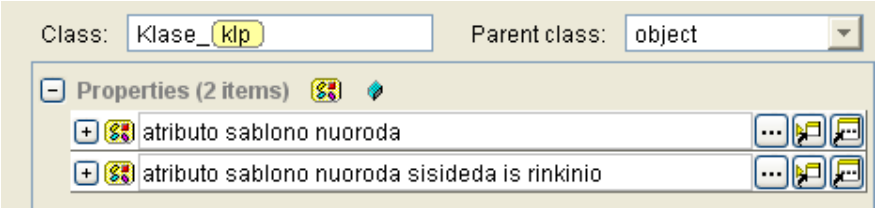
Atlikus visus 3 dalies žingsnius pereinama prie 2 dalies *šablono vykdymas*. Tačiau čia 2.1, 2.2, 2.3, 2.4 žingsniai atliekami tik vienam taisyklių rinkinio šablonui.

4.3.3. Veiklos taisyklių šablono sudarymo procesas

Siekiant suformuoti veiklos taisyklių rinkinio šabloną, pirmiausiai turi būti sukurti VT rinkinį sudarančių elementų šablonai (35 pav.). Pirmas kuriamas, klasę sudarančių, atributų šablonas (38 pav.). Atributo tipas sukuriamas per reikšmės poziciją. Vienas atributas gali turėti tik vieną iš apibrėžtų tipų. Sukurtus atributus galima redaguoti per internetinę vartotojo sąsają (*RMA*). Sukurtas atributų šablonas yra naudojamas klasių šablone (39 pav.). Atributą sudaro šablono pozicijos, identifikuojančios unikalų atributo pavadinimą bei jo tipą. SRL kalbos turinio dinamiškumas, taip pat formuojamas šablono pozicijomis.



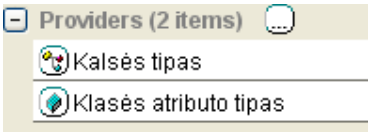
38 pav. *Atributo šablonas*



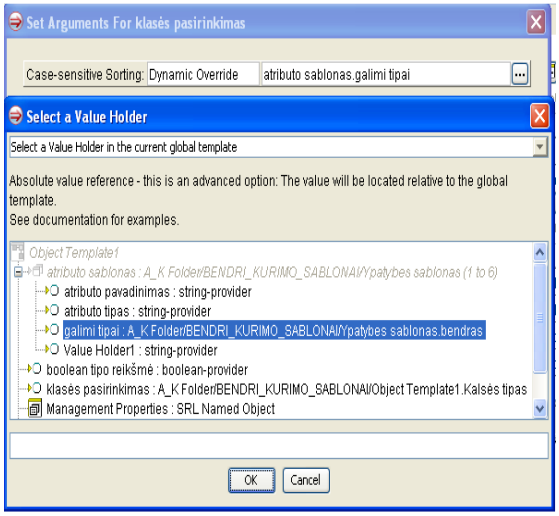
39 pav. *Klasės šablonas*

Sukuriamos reikšmės pozicijos, kurios yra nuorodos į atributų šablonus (38 pav.), taigi atributo šablonas į klasės funkciją įtraukiamas per esybių šablono poziciją (39 pav.). Klasės tipas gali būti objektas, kita klasė, arba rinkinys. Pastaruoju atveju klasės funkcijoje nurodoma esybių šablono pozicija, kuri remiasi rinkinio šablonu.

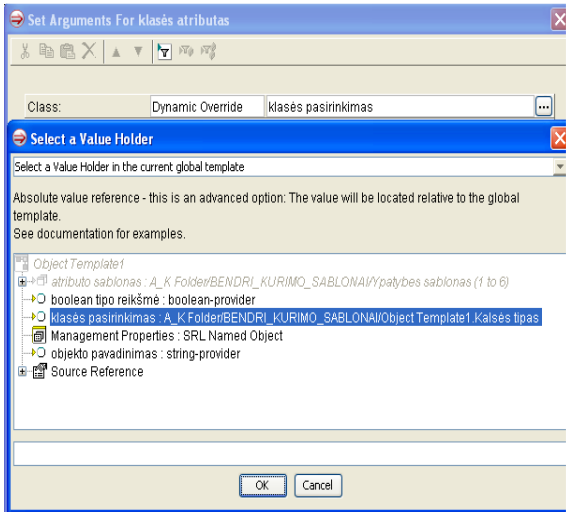
Objekto šablonas sukuriamas tam, kad būtų galima tam tikrai klasei sukurti objektą. Jo tikslas yra saugoti, vykdymo metu, gautas laikinas reikšmes. Objektai gali būti lokalūs arba globalūs. Globalūs objektai yra pasiekiami viso projekto rėmuose, o lokalus objektas saugo tik tos taisyklių grupės reikšmes, kuriai jis yra priskirtas. Klasės tipas nurodomas reikšmės pozicijoje. Kadangi objektas priskiriamas konkrečiai klasei, taigi jis apima ir konkrečios klasės atributus. Pakeitus klasės atributus, keičiasi ir objekto atributai, t.y. atliekamas dinaminis klasės atributų perrašymas. Perrašymas realizuojamas lokalių tipų objekto šablone sukūrimu. Kaip minėta objektą apima klasės šablonas, kurio funkcijos turinyje egzistuoja klasės atributai. Dėl šios priežasties būtinas atributo tipo sukūrimas ir klasės tipo sukūrimas (40 pav.). Dinaminis perrašymas realizuojamas per reikšmės poziciją, kuri pirmiausia apima atributo šabloną. Pastarasis reikalingas reikšmėms pasiekti ir perduoti kuriamo objekto turiniui. Objekto sukūrimas internetinėje sąsajoje yra objekto pavadinimo realizavimas konkrečiai klasei. Šablono vidinės logikos atžvilgiu tai kur kas sudėtingesnis procesas. Klasės pasirinkimo metu į objekto šablono turinį automatiškai perduodami atitinkamos klasės atributai, bei jų tipai. Klasės perrašymas vykdomas kartu su jai priklausančiais atributais (41 pav., 42 pav.).



40 pav. *Klasės ir jos atributų lokalūs tipai*

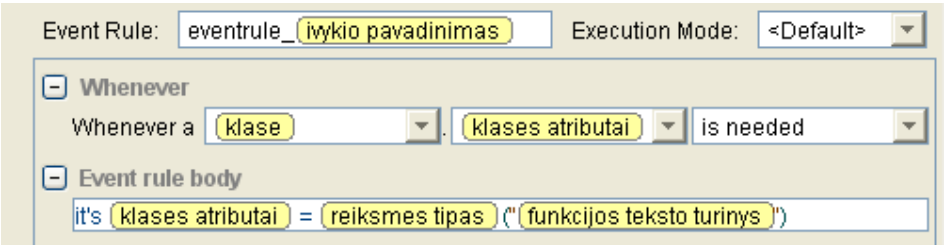


41 pav. *Klasės dinaminis perrašymas*



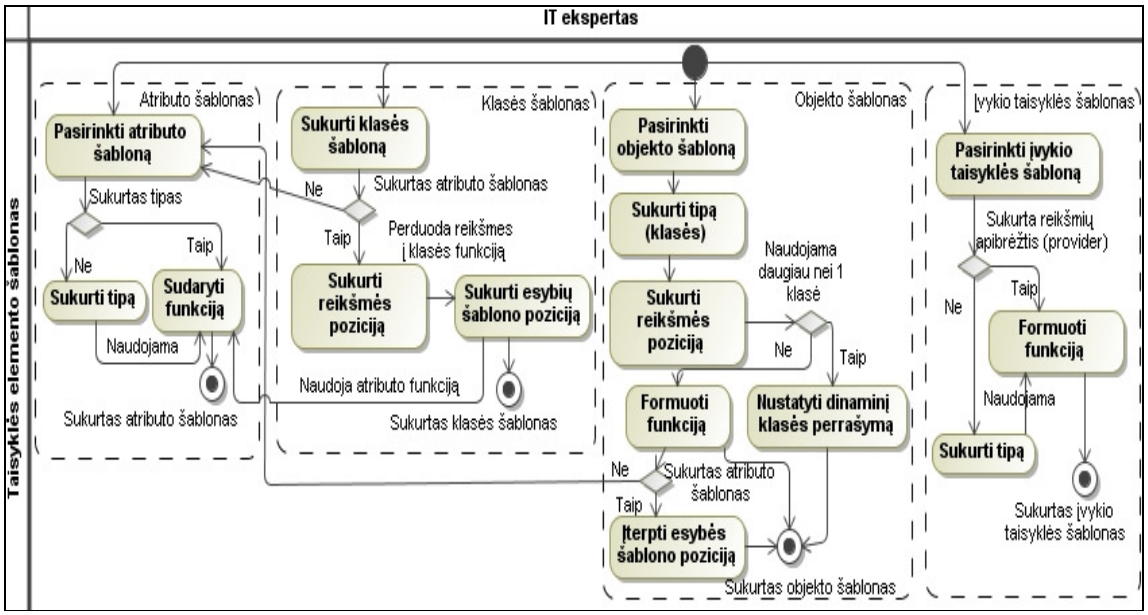
42 pav. *Klasės atributų perrašymas, priklausantis nuo pasirinktos klasės*

Paskutinis elementas, reikalingas taisyklių rinkinio sukūrimui yra įvykio taisyklės šablono sudarymas (43 pav.). Įvykio taisyklė inicijuoja taisyklių rinkinio vykdymą. Įvykio taisyklės gali būti objekto atributo reikšmės pasikeitimas, perdavimas taisyklių varikliui, objekto inicijavimas, sunaikinimas. Įvykio taisyklės šablone aprašyti įvykiai yra iškviečiami automatiškai, kai aptinkamas konkretus įvykis. Įvykio taisyklė gali būti trijų lygmenų : atributo, objekto ir išorinė. Bendru atveju įvykio taisyklės iniciavimo šablono funkciją sudaro klasė, vienas atributas, rezervuotas žodis, įvykio taisyklės turinys.



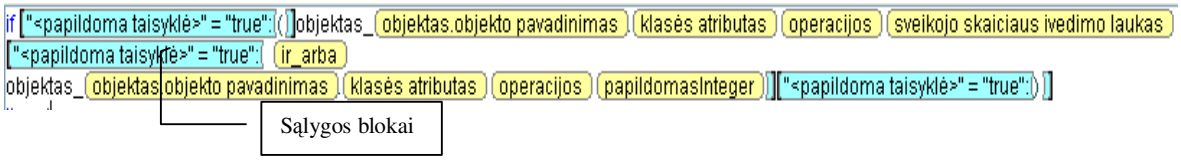
43 pav. *Įvykio taisyklės šablonas*

Viena įvykio taisyklė gali inicijuoti tik vienos klasės vieną atributą. Todėl yra reikalingas taip pat dinaminis klasės ir jai priklausančių atributų perrašymas, kuris veiktų ne tik taisyklės iniciavimo, bet ir turinio dalyje. Reikšmės tipas, įvykio taisyklei, perduodamas per sąrašo tipą, kurio elementus sudaro prompt funkcijos. Jos objektui perduoda vieną iš galimų reikšmių tipų (boolean, string, integer, real, enumeration).



44 pav. *Taisyklės elementų šablonų formavimo procesas*

Suformavus atitinkamų elementų šablonus (44 pav.), yra sudaromas veiklos taisyklės šablonas. Taisyklės šablonas apima 44 pav. sudarytus šablonus, kurie įtraukiami per reikšmės pozicijas, nurodant konkrečius tipus arba šablonus. Taisyklės funkcijoje naudojami šie elementai: sąlygos blokai, teksto šablono pozicijos, ciklo konstrukcijos bei rezervuotų žodžių kombinacijos.

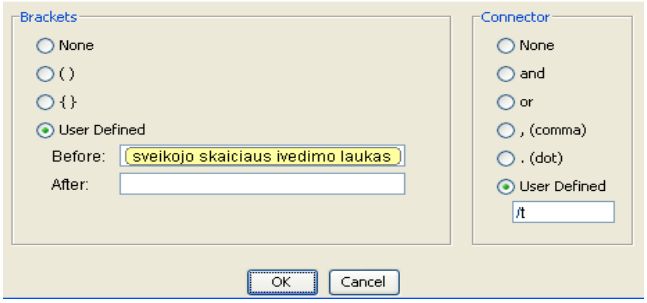


45 pav. *Taisyklės šablono turinio sąlygos dalis*

45 pav. mėlynos spalvos sąlygos blokai naudojami taisyklės turinio papildomai konstrukcijai kurti, kuomet taisyklės sąlygos dalį sudaro daugiau nei viena sąlyga. Kuri gali būti alternatyva arba papildomas apribojimas. Taisyklės turinio pildymas praplečiamas internetinėje sąsajoje. Automatiškai generuojamas papildomas kodas ir perduodamas į veiklos taisyklių saugyklą.

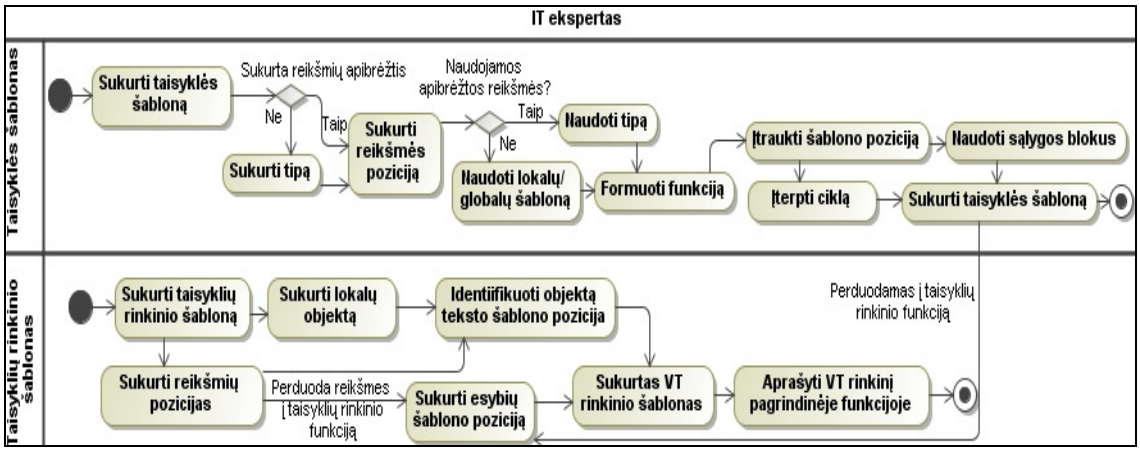
Sudarant sprendimų bloką, sąlygos dalis (*if* konstrukcija) nurodoma naudojant statines reikšmes arba teksto šablono poziciją. Ciklo konstrukcija (46 pav.) leidžia suformuoti dinaminius ciklus, kuriuose nurodomas veiksmas prieš ir po ciklo, ciklo dalių sujungimo veiksmas (47 pav.).

46 pav. *Ciklo konstrukcijos naudojimas atsakymo spausdinimo formavime*



47 pav. *Ciklo konstrukcijos formavimas*

Sukūrus taisyklės šabloną, sudaromas veiklos taisyklių rinkinio šablonas pagal tuos pačius principus (49 pav.).



48 pav. *Taisyklės ir taisyklių rinkinio šablonų formavimo procesas*

Pagrindinė taisyklių rinkinio šablono funkcija yra grupuoti veiklos taisykles. Grupavimas gali būti atliekamas pagal taisyklių funkcionalumą, sudėtingumą ar kitus parametrus. Taisyklių rinkinio funkcija apima esybių šablono pozicijas (taisyklės šablono, objekto šablono, taisyklių rinkinio šablono). Taisyklių rinkinio šablone grupuojami *Blaze Advisor* elementai. Grupavimas pagerina taisyklių variklio našumą.

4.3.4. Veiklos taisyklių kūrimas naudojant šabloną

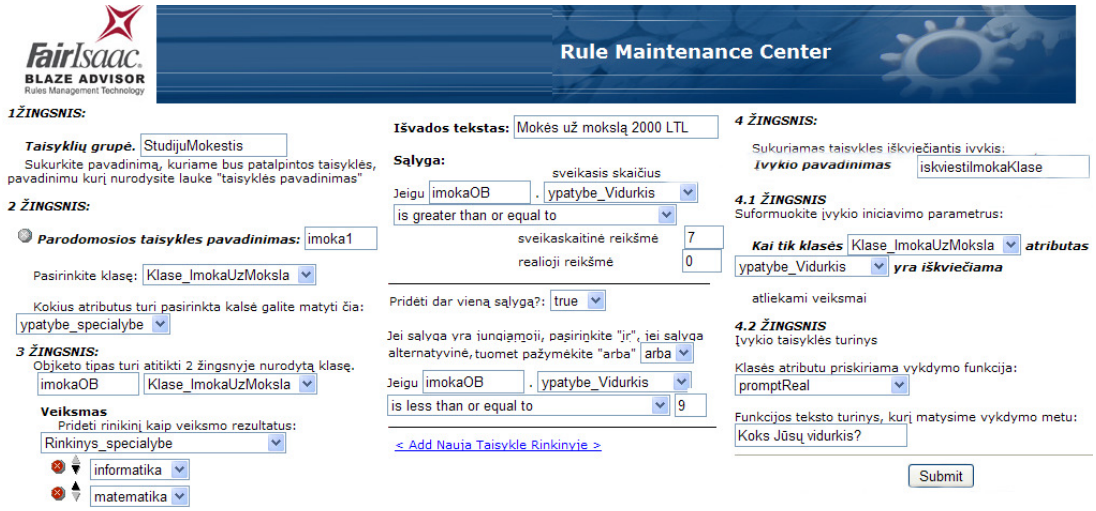
Sukūrus reikiamą šabloną, kiekvienam šablonui sukuriamas šablono egzempliorius, kuris užpildomas konkrečiais duomenimis. Duomenys įvedami per internetinę vartotojo sąsają (RMA), taigi duomenis suvesti gali ir įmonės darbuotojas. Taisyklių variklis VT sprendimo išvedimui naudoja taisyklės egzempliorių. Jo turinį galima valdyti per sprendimų blokus, t.y. jie nurodo VT varikliui kurią kodo dalį apdoroti, esant tam tikroms sąlygoms. Pagal suformuotą šabloną, (49 pav.) pateikiamas sukurtas veiklos taisyklių rinkinio egzempliorius, kurį sudaro taisyklių rinkinio pavadinimas, taisyklės pavadinimas, priskirta klasė su atributais,

objektai, taisyklės. 51 pav. pateikta VTVS kūrimo aplinka, kuri akivaizdžiai įrodo, kad šablonais suformuotas VT kūrimas yra paprastesnis.

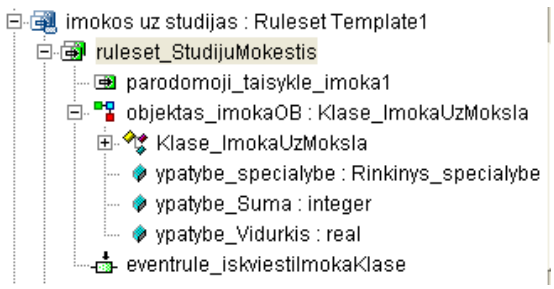
Sukruti elementai per internetinę vartotojo sąsają transformuojami į atskirų komponentų programinį kodą. Egzemplioriaus turinys, kuris automatiškai sukuriamas taisyklių saugykloje, pateikiamas 50 pav. Taisyklių rinkinyje esančios taisyklės kodas, bei tą taisyklių rinkinį inicijuojanti įvykio taisyklė, laikinąsias reikšmes, vykdymo metu, saugantis objektas pateikiami 52 pav.

Visi VT grupės elementai gali būti modifikuojami, įtraukiami/pašalinami. Tokiu būdu įmonės atstovas pats gali kontroliuoti, kurios veiklos taisyklės turi būti aktyvios, kokie turi būti jų parametrai, esant poreikiui juos adaptuoti prie pasikeitusių sąlygų.

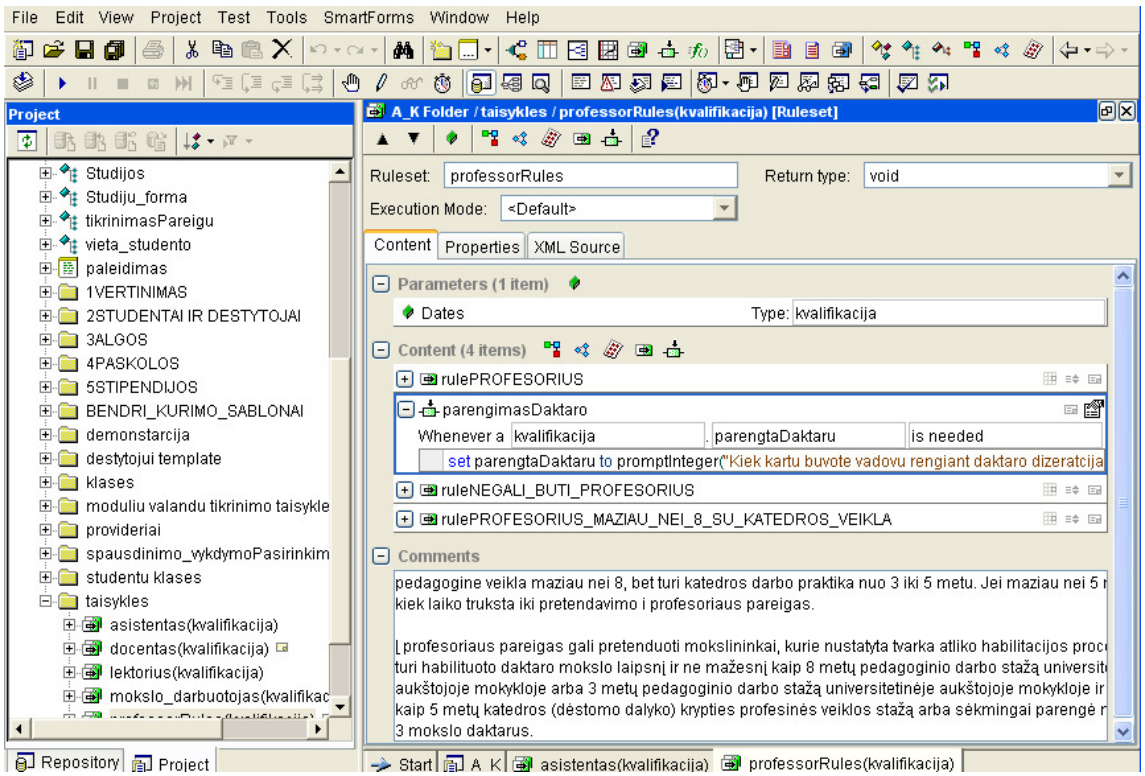
Egzempliorių kūrimas apsaugo veiklos taisykles ir jų komponentus, nuo bet kokių modifikavimo metu įtakojamų klaidų. Redagavimo metu taisomas ne šablonas, bet konkretus egzempliorius, kuris yra užpildytas duomenimis ir naudoja tik šablono struktūrą. Taigi vartotojai šablono struktūros paveikti negali. Net jei iš saugyklos pašalina visus egzempliorius.



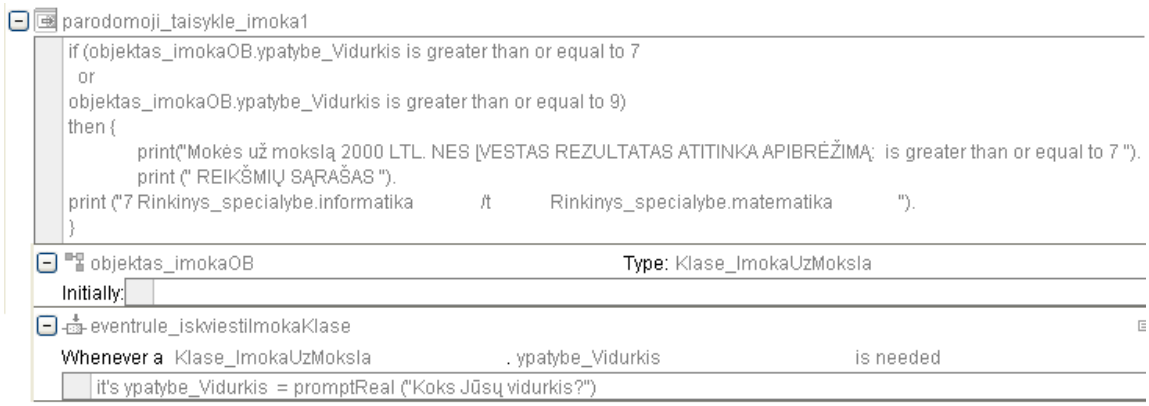
49 pav. *Taisyklių valdymo aplinka RMA*



50 pav. *Pavyzdinio egzemplioriaus turinys taisyklių saugykloje*



51 pav. *VTVS kūrimo aplinka veiklos taisyklėms ir jų komponentams*

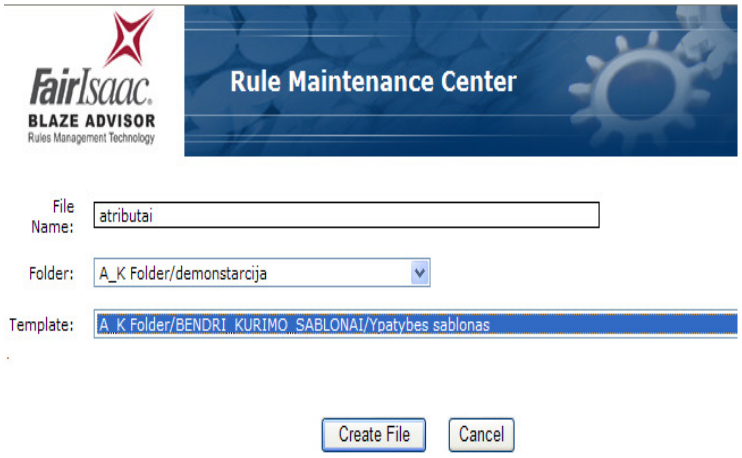


52 pav. *Taisyklės, sukurtos šablonu turinys*

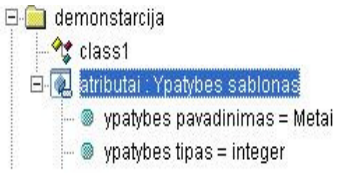
Egzempliorius kuriamas atskiriems šablono elementams arba visam taisyklių rinkinio šablonui. Egzemplioriaus kūrimo procesas pateikiamas 53 pav. 54 pav. pateikiamas internetinės sąsajos aplinka egzempliorių kūrimui. Veiklos taisyklių valdymo sistemoje, siekiant hierarchinio projekto išdėstymo, palengvinančio projekto turinio suprantamumą, sukuriami aplankai. Jų turinį galima pasiekti per internetinę sąsają tik tuomet, kai jame egzistuoja veiklos taisyklių valdymo sistemos bent vienas elementas. 55 pav. pateikiamas VTVS projekte esančio aplanko turinys, kuris buvo sukurtas per RMA. Sukurti atskiri egzemplioriai į bendriausią egzemplioriaus turinį įtraukiami per egzempliorių pasirinkimo

tipą, kuris leidžia į esamo šablono turinį įtraukti norimą sukurtą šablono egzempliorių. VTVS ir internetinė sąsaja yra susiejama betarpiškai, t.y. sukurto egzemplioriaus turinio užpildytus duomenis galima keisti ir VTV sistemoje bei atvirkščiai. Sukūrus šabloną VTVS galima atlikta jos redagavimą per RMA. Siekiant apsaugoti konkrečiame egzemplioriuje esančius duomenis, suteikiamos priėjimo prie šablono teisės.

53 pav. Egzemplioriaus kūrimo per internetinę sąsają procesas



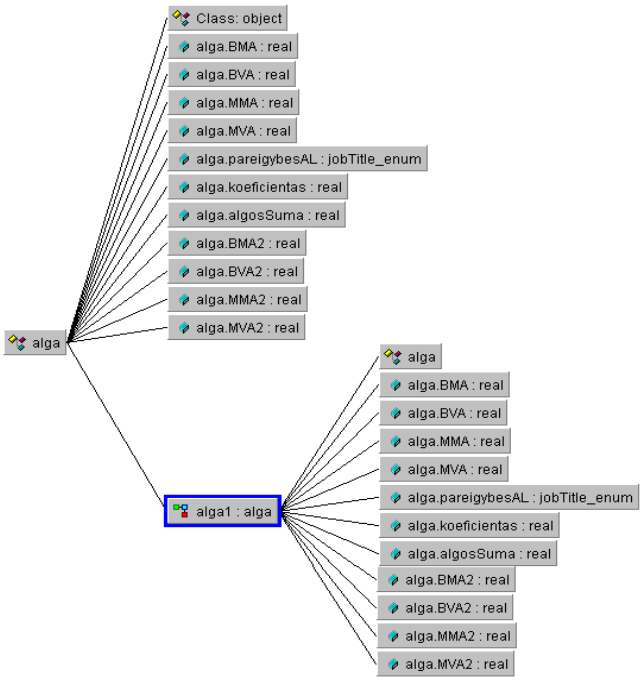
54 pav. Egzemplioriaus kūrimas naudojantis RMA.



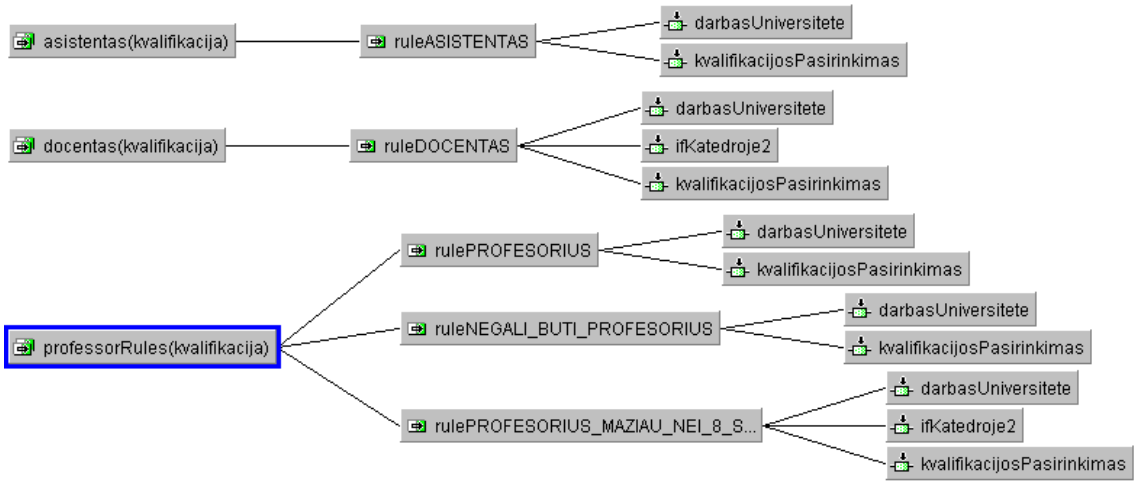
55 pav. Egzemplioriaus turinys VTV sistemoje

Objektas, paveldi klasės atributus. Sukurto ir užpildyto duomenimis objekto šablono klasės paveldėjimas grafiškai pateikiamas 56 pav.

Paveiksle (57 pav.) pateikiami taisyklių rinkiniai, su juose esančiomis taisyklėmis bei tas taisykles iškviečiančiomis įvykio taisyklėmis. Kiekviena įvykio taisyklė iškviečia skirtingus klasės atributus. Vykdymo metu vertinami vartotojo įvedami duomenys ir pateikiamas atitinkamas sprendimas. Kiekviena įvykio taisyklė klasės atributą iškviečia per prompt funkciją ir pateikia konkretaus atributo įvedimo laukus vartotojui. Tuomet nuskaito įvestas reikšmės rezultato pateikimui ir perduoda jas į taisyklų rinkinių taisykles.

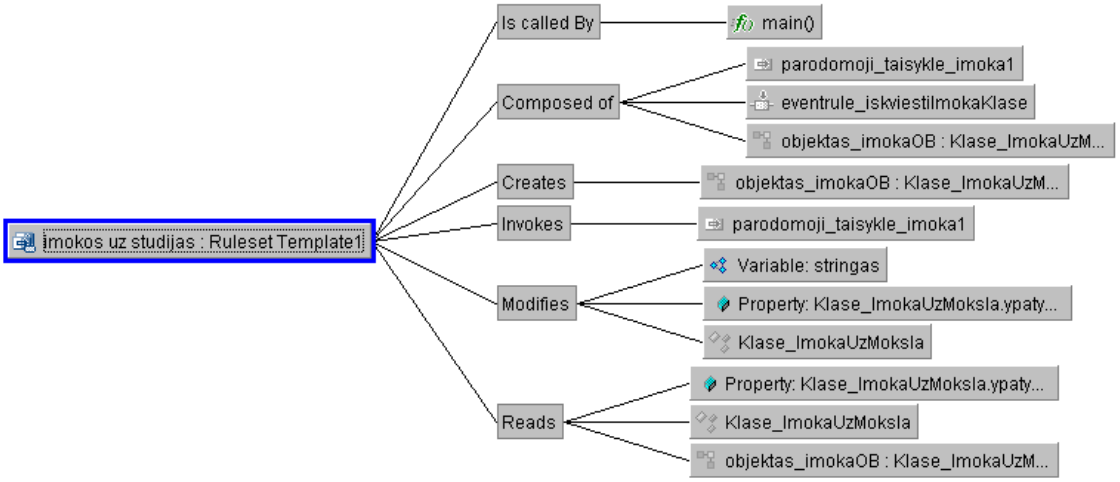


56 pav. *Objekto sukūrimas klasei*



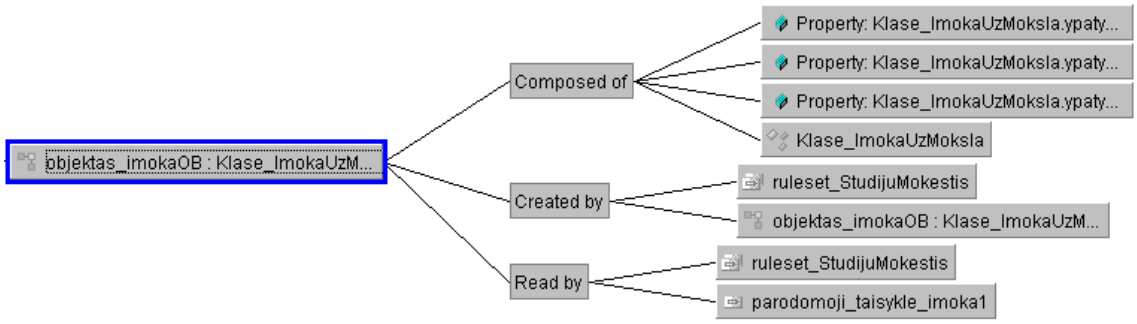
57 pav. *Taisyklių rinkiniai, jų taisyklės ir taisykles inicijuojančios įvykio taisyklės*

Egzemplioriuje buvo sukurtas *įmokos už studijas* taisyklių rinkinys (58 pav.), kurį sudaro taisyklė identifikuota *parodomoji taisyklė įmoka1*, įvykio iniciavimo taisyklė *eventrule_iškviesti įmoką klasė* ir objektas inicializuotas pavadinimu *objektas_imokaOB*. Vykdymui sukurtas taisyklių rinkinys iškviečiamas per *main* funkciją. Vykdymo metu yra modifikuojamos atributų ir kintamojo reikšmės, kurios nuskaitomos ir perduodamos vykdymui per sukurtą objektą. Taisyklių rinkinys, vykdymo metu, iškviečia jame sukurtą veiklos taisyklę *parodomoji taisyklė įmoka1*. Taisyklių rinkinio egzemplioriaus veiksmai ir sandara pateikiama 58 pav.



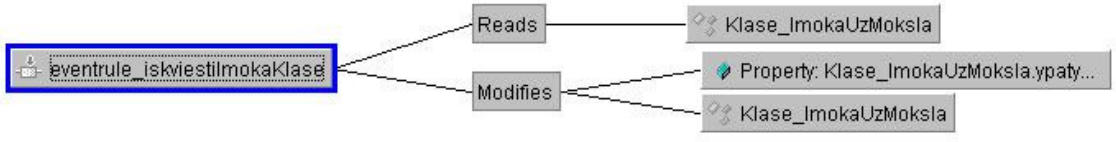
58 pav. *Taisyklių rinkinio šablono turinys veiksmai.*

Objektą sudaro klasė *klasė_įmoka už mokslą,* ir trys sukurti klasės atributai. Sukurtas objektas yra taisyklių rinkinyje *Studijų mokestis.* Nuskaitomas taisyklės *parodomoji taisyklė įmoka1*, esančios taisyklių rinkinyje (59 pav.)

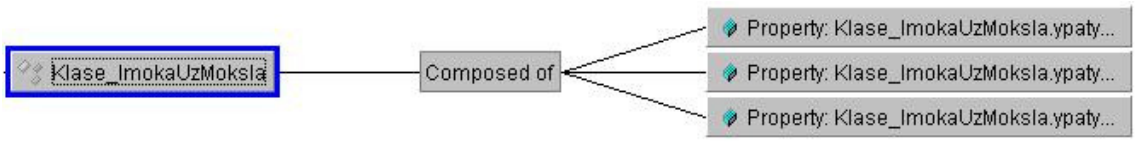


59 pav. *Sukurto objekto šablone sudėtis ir vaidmuo bendrajame šablono kontekste*

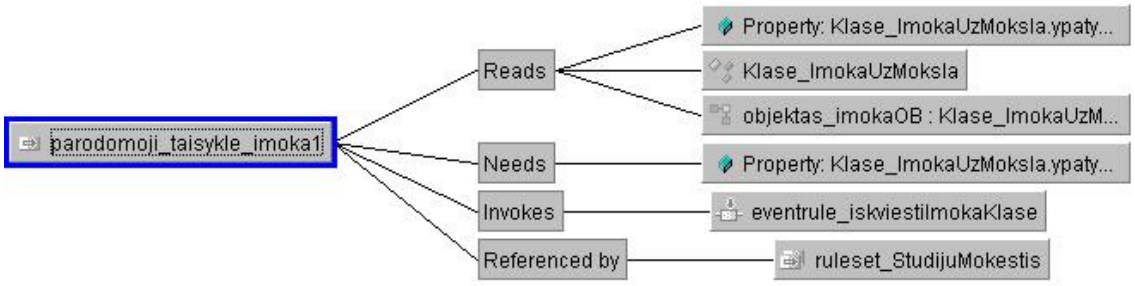
Įvykio iniciavimo taisyklei (60 pav.) e*ventrule_iškviesti įmoka klasę* perduodama *klase_įmoka už mokslą*, kuri modifikuoja šios klasės parametrą *klasė_įmoka už mokslą.atributas_Vidurkis*. Kitaip sakant, šis klasės atributas perduodamas kaip vykdymo funkcijos parametras, kuriam įvykio taisyklė turi gauti reikšmę ir perduoti, taisyklių rinkinyje esančiai taisyklei per jos objektą.



60 pav. *Sukurtos įvykio iniciavimo taisyklės vaidmuo šablone*



61 pav. *Šablone sukurtos klasės sudėtis*



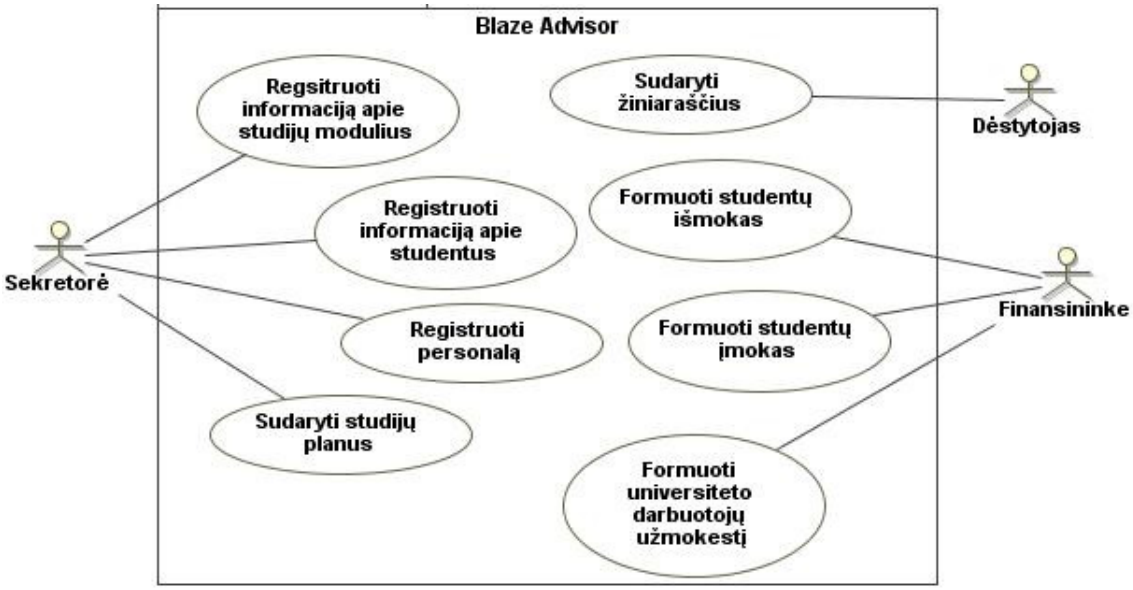
62 pav. *Šablone sukurtos taisyklės vaidmuo šablono kontekste*

Klasę *įmoka už mokslą* sudaro trys atributai: *specialybė, suma, vidurkis* (61 pav.). *Parodomiji_taisyklė įmoka1* klasės atributą *vidurkis* perduoda taisyklių varikliui per juos apibrėžiantį objektą, kuriam perduodamas konkretus atributas *vidurkis*. Kad taisyklių variklis galėtų apdoroti apibrėžtą veiklos taisyklę, reikia konkrečios atributo *vidurkis* reikšmės, kuri inicijuojama *eventrule_iškviesti įmoka klasę*. Veiklos taisyklė yra apibrėžta taisyklių rinkinyje *ruleset_studijų mokestis* (62 pav.).

5. VEIKLOS TAISYKLIŲ REALIZAVIMO TYRIMAS

5.1. Visų elementų realizavimo vertinimas

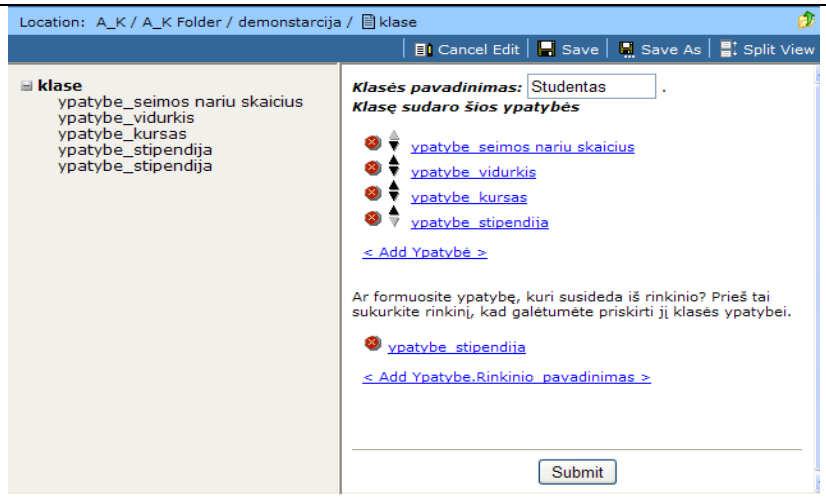
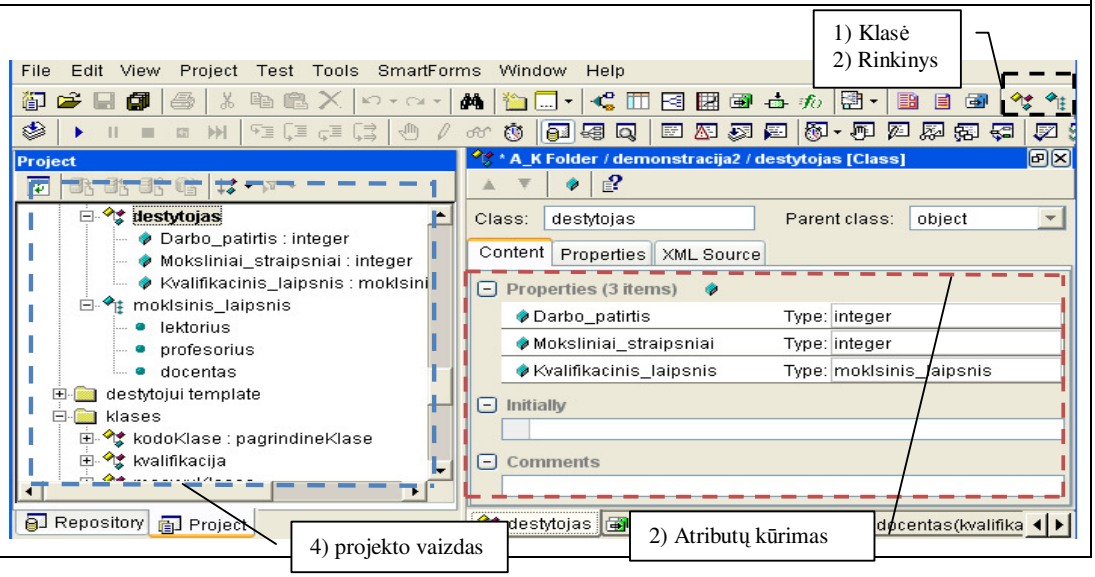
Tyrimas atliktas, realizuojant tokio pat sudėtingumo veiklos taisykles, naudojant RMA aplinką ir BA įrankio aplinką. Dalykinė sritis – studijų organizavimas. Pateikiama PA diagrama (63 pav.) Elementų kūrimas atliktas nuosekliai. Stebėta galimybė, kiekvieno kuriamo komponento atžvilgiu, kurti, išsaugoti, modifikuoti veiklos taisykles. Rezultatai pateikiami lentelėse.



63 pav. *Sistemos „Studijų administravimas“ prototipo PA diagrama*

Sukuriama klasė ir jos atributai:

Lentelė Nr. 4 Klasės ir atributų kūrimas

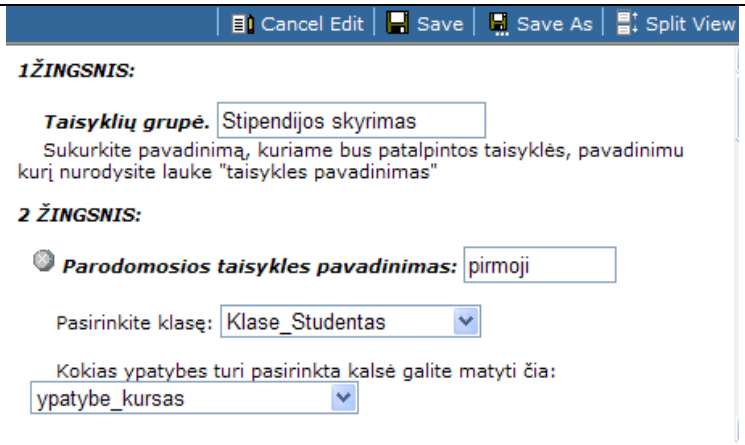
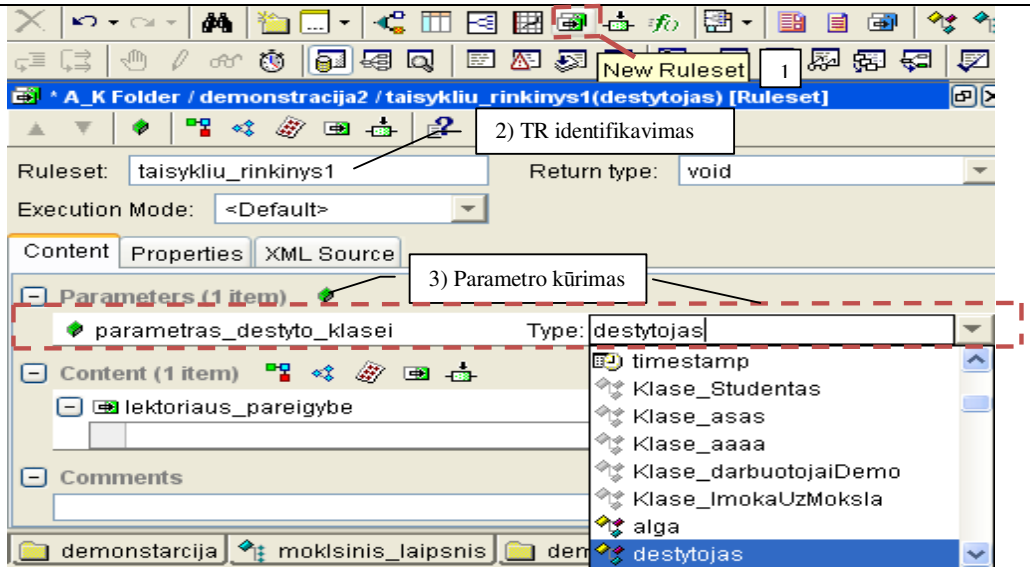
RMA	
BLAZE ADVISOR	

Lentelė Nr. 5 *Klasės ir atributų kūrimas Vertinimas*

Atstovai	***RMA aplinka***			***BA aplinka***		
	Kūrimas	***Redagavimas***	***Šalinimas***	***Kūrimas***	***Redagavimas***	***Šalinimas***
IT specialistas	*100 %*	*100 %*	*100 %*	*100 %*	*100 %*	*100 %*
Veiklos atstovas	*100 %*	*100 %*	*100 %*	*30%*	*40 %*	*50 %*

IT specialistas abiejuose aplinkose visus veiksmus atlikti gali vienodai. Veiklos atstovas BA įrankio aplinkoje sukurti klasę gali tik tuomet, jeigu yra informuotas apie VT kūrimo pirmąjį žingsnį. Surasti egzistuojančią klasę gali tik žinant kokiu simboliu identifikuojama klasė. Redagavimą ir šalinimą gali atlikti intuityviai.

Lentelė Nr. 6 *Taisyklių rinkinio ir taisyklės kūrimas*

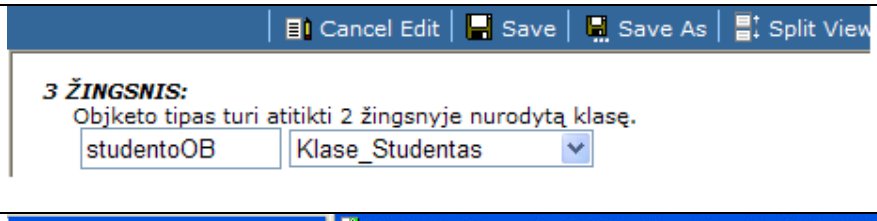
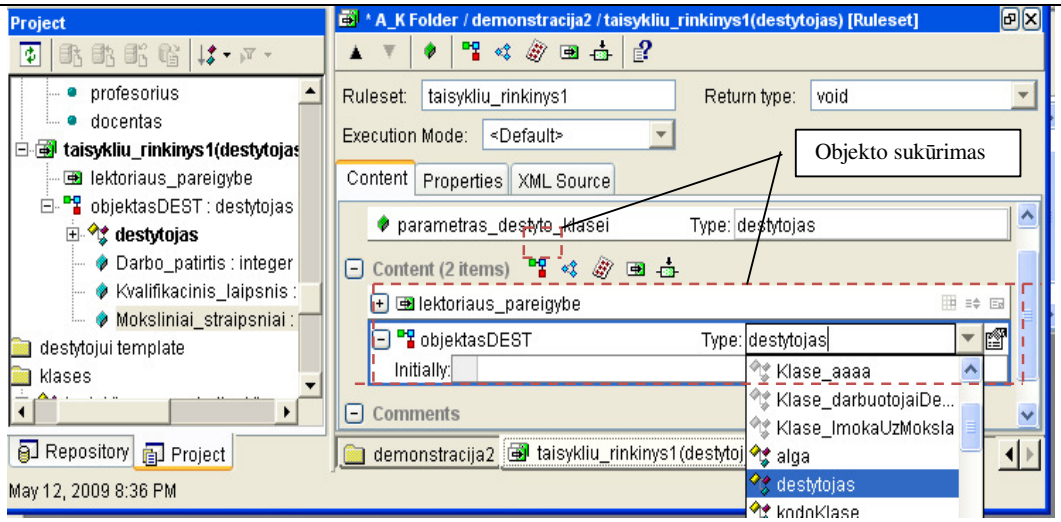
RMA	
BLAZE ADVISOR	

Lentelė Nr. 7 *Taisyklių rinkinio ir taisyklės kūrimas vertinimas*

Atstovai	***RMA aplinka***			***BA aplinka***		
	Kūrimas	***Redagavimas***	***Šalinimas***	***Kūrimas***	***Redagavimas***	***Šalinimas***
IT specialistas	*100 %*	*100 %*	*100 %*	*100 %*	*100 %*	*100 %*
Veiklos atstovas	*100 %*	*100 %*	*100 %*	*10%*	*20 %*	*20 %*

RMA aplinkoje veiklos atstovui pateikiami žingsniais, kuriais jis turi vadovautis, pildant leidžiamus laukus. Taisyklės rinkinį identifikuoja pavadinimu, sukuria taisyklę su norimu pavadinimu. Pasirinkus klasę , mato kokius atributus ji turi ir iš kurių galės formuoti taisyklę. IT vadovas atlieka tuos pačius žingsnius BA įrankyje. Šiame atsiranda papildomas žingsnis – lokalaus parametro sukūrimas. Šį veiksmą RMA aplinkoje atlieka šablonas. Parametro sukūrimo veiksmas nėra pateikiamas veiklos atstovui.

Lentelė Nr. 8 *Objekto kūrimas*

RMA	
BLAZE ADVISOR	

Lentelė Nr. 9 *Objekto kūrimas vertinimas*

Atstovai	***RMA aplinka***			***BA aplinka***		
	Kūrimas	***Redagavimas***	***Šalinimas***	***Kūrimas***	***Redagavimas***	***Šalinimas***
IT specialistas	*100 %*	*100 %*	*100 %*	*100 %*	*100 %*	*100 %*
Veiklos atstovas	*90%*	*100 %*	*10 %*	*30%*	*40 %*	*50 %*

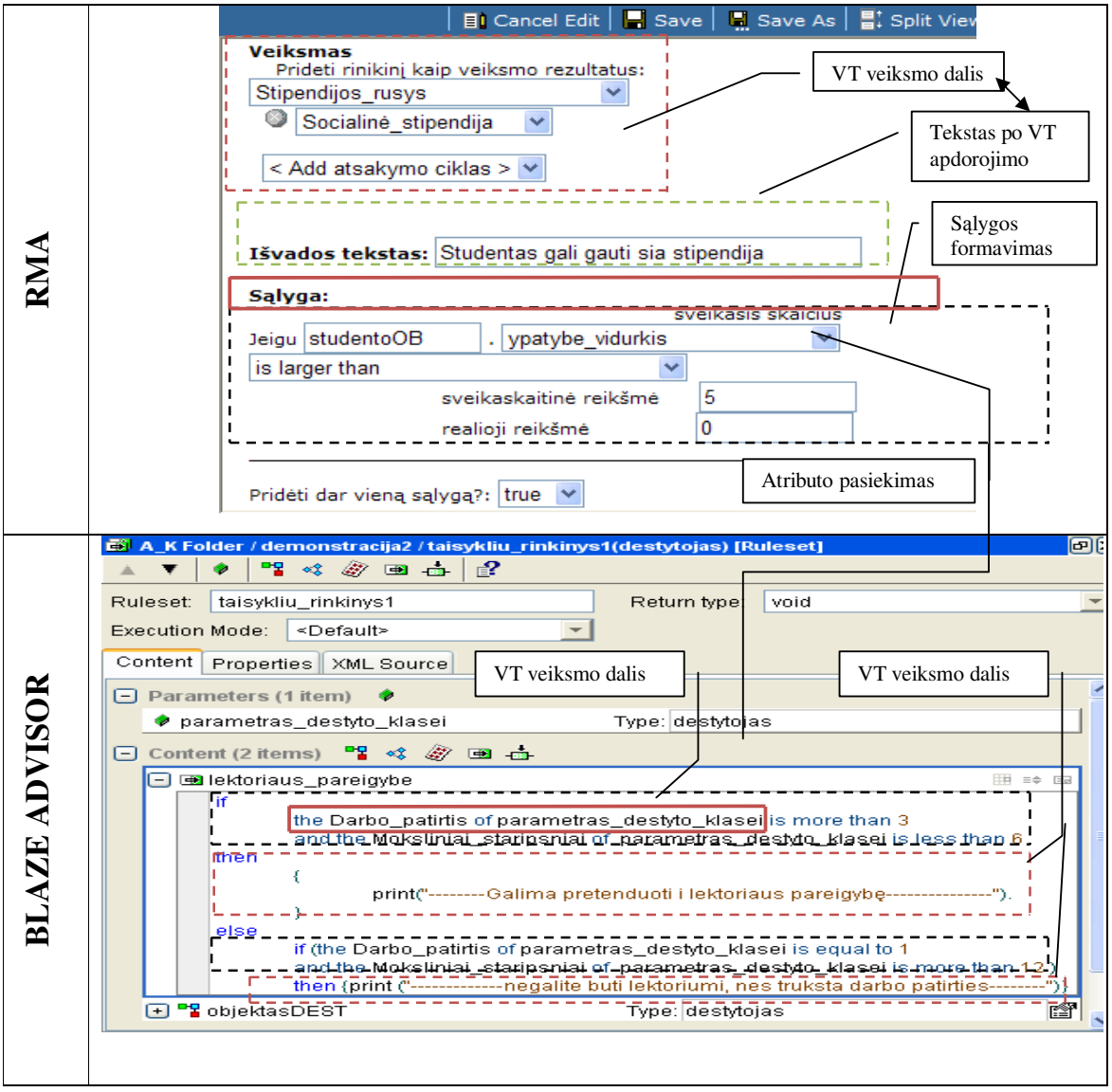
Veiklos atstovas BA įrankyje objektą sukurti gali intuityviai, suradęs objekto žymenį. Tačiau didžiausia tikimybė, jog nepasirinks reikiamos klasės. RMA šablone atlikęs tuos pačius veiksmus, jis informuotas, jog šiame žingsnyje klasė turi būti ekvivalenti 2 žingsnyje nurodytai klasei. Galimi trukdžiai: kuriant objektą, veiklos atstovas gali nenurodyti reikiamos klasės. Redaguoja tik objekto pavadinimą. Objekto pašalinti negali, nes jis reikalingas perduodant klasės atributų reikšmes vykdymui.

Lentelė Nr. 10 *VT turinio kūrimas vertinimas*

Atstovai	***RMA aplinka***			***BA aplinka***		
	Kūrimas	***Redagavimas***	***Šalinimas***	***Kūrimas***	***Redagavimas***	***Šalinimas***
IT specialistas	*100 %*	*100 %*	*100 %*	*100 %*	*100 %*	*100 %*
Veiklos atstovas	*90 %*	*100 %*	*0 %*	*10%*	*40 %*	*50 %*

Veiklos atstovas taisyklės sąlygos aprašyme turi nurodyti tik tą atributą, kuriam bus formuojama sąlyga. Pasirinkto atributo reikšmė, įvesta vykdymo metus, lyginama su veiklos atstovo nustatyta sąlygos apibrėžtimi. Joje nurodomas operatorius ir rėžis. Šioje dalyje objekto pavadinimas identiškas 3 žingsnyje sukurtam objektui (Lentelė 8). Pakeitus objekto pavadinimą bet kuriame jo lauke, pakeitimas atliekamas automatiškai ir kitame. Todėl suklysti vartotojas negali. Rinkinys naudojamas išvados dalyje. Sukurtas rinkinys identifikuotas statiniu tekstu *Rinkinys_vartotojo sukurtas pavadinimas*. Pasirinkus reikiamą rinkinį automatiškai pateikiami rinkinio elementai.

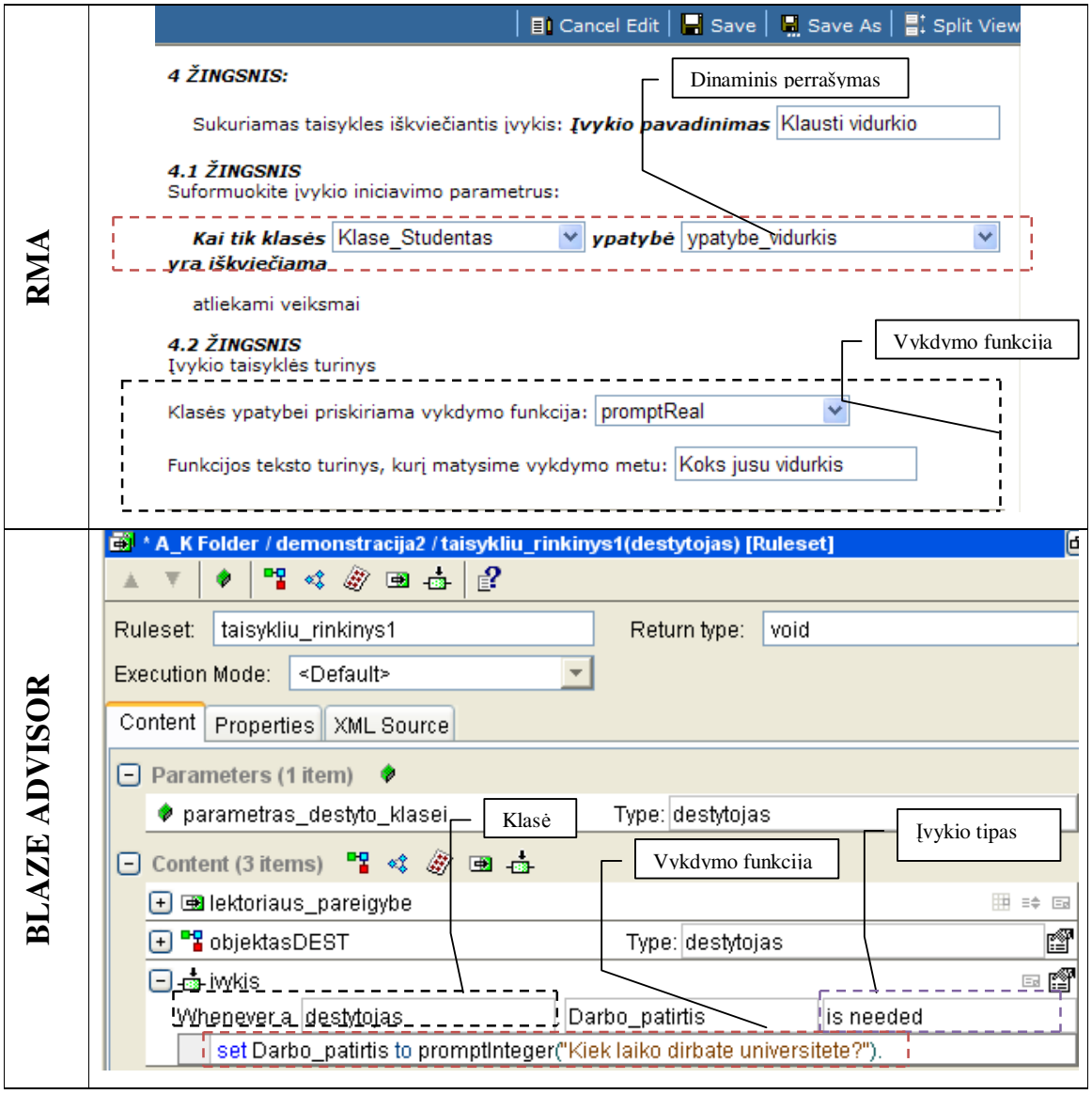
Lentelė Nr. 11 *VT turinio kūrimas*



Pateikiamos jo reikšmės, kurias įtraukia prie išvados teksto. Veiklos atstovas prideda papildomą sąlygos apribojimą, kurią prie sukurtosios prijungia žodžiais *ir, arba.* Sąlyga kuriama analogiškai. IT specialistas RMA aplinkoje sąlygos operatorius ir rėžius keičia

rankiniu būdu. Pakeitus parametro ar objekto klasę, reikia perrašyti ir atributų pavadinimus kode. Galimi trukdžiai: vartotojas gali pasirinkti ne tą rinkinį, kuris apibrėžtas klasės atribute. Šalinimas sudedamųjų dalių negalimas. Vartotojas įspėjamas apie neleistiną tuščią lauką, jis turi būti užpildytas.

Lentelė Nr. 12 *Įvykio iniciavimo kūrimas*



Lentelė Nr. 13 *Įvykio iniciavimo kūrimas vertinimas*

Atstovai	*RMA aplinka*			*BA aplinka*		
	Kūrimas	*Redagavimas*	*Šalinimas*	*Kūrimas*	*Redagavimas*	*Šalinimas*
IT specialistas	*100 %*	*100 %*	*100 %*	*100 %*	*100 %*	*100 %*
Veiklos atstovas	*98 %*	*100 %*	*90 %*	*10%*	*40 %*	*50 %*

IT atstovas RMA aplinkoje papildomai nurodo įvykio tipą. Vykdymo funkcija aprašoma nurodant, operaciją, atributą vykdymo funkciją su jos parametrais. Veiklos atstovui 4.1 žingsnyje klasė automatiškai pateikiama tokia pati kaip ir objekto. Dinamiškai keičiasi ir klasės atributai. Galimi trukdžiai: veiklos atstovas pasirenka vykdymo funkciją, neatitinkančią atributo tipo. Gali palikti tuščias reikšmes, leidžiamuose laukuose. Palikti tuščių laukų nerekomenduojama, nes jas teks užpildyti po projekto sukompiliavimo, kuomet pateikiamas pranešimas apie tuščius laukus.

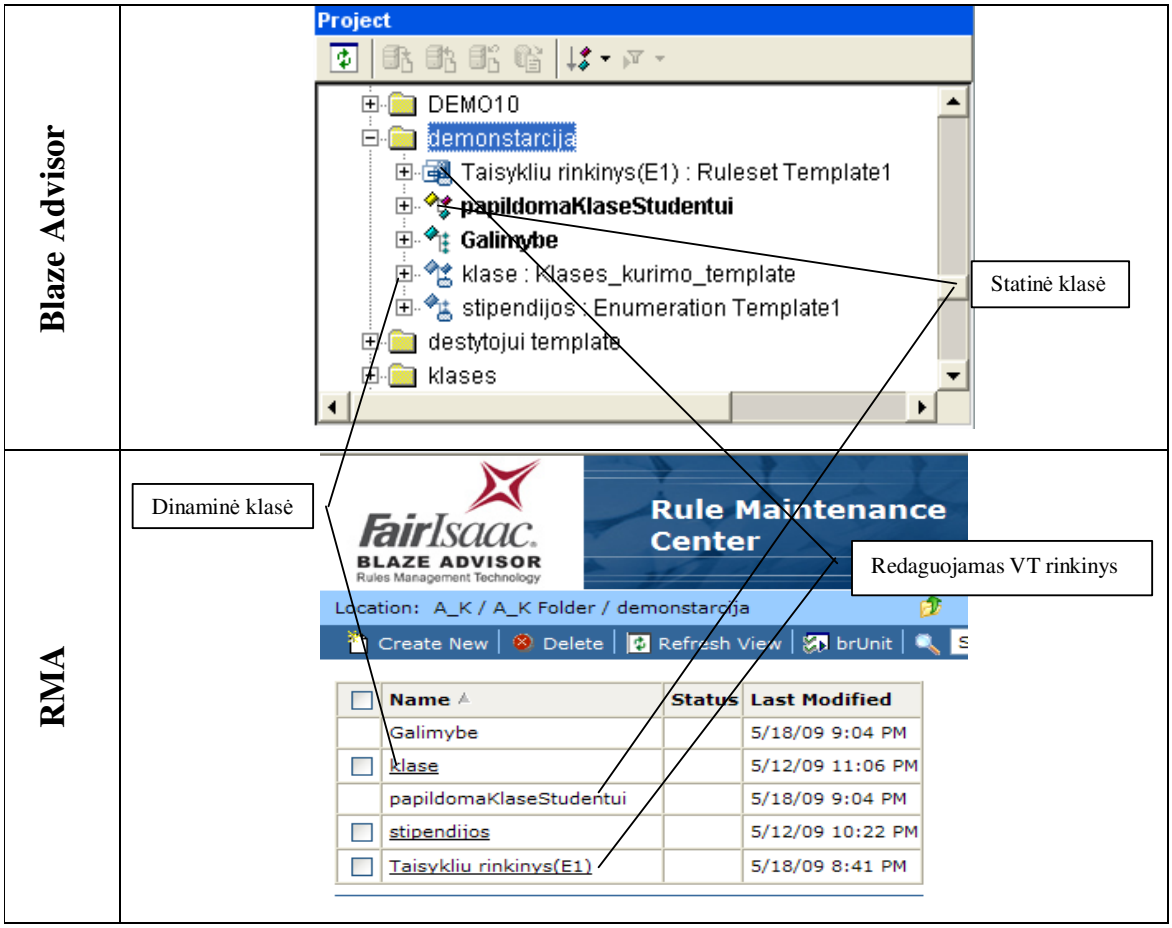
5.2. Sukurto šablono, dinaminės klasės keitimo statine klase, galimybių vertinimas

IT specialistas, VTVS papildė statine klase. Stebima dalykinės srities atstovo (toliau vartotojo) galimybė panaudoti statinę klasę, taisyklės formavimui. Vartotojas turi susikūręs taisyklių rinkinį, dinaminę klasę, rinkinį (Lentelė Nr. 14).

Taisyklių rinkinį, keičiant statine klase, kuri sukurta VTVS, šalinami elementų unikalūs pavadinimai, kurie pakeičiami naujais (Lentelė Nr. 15). Jei pavadinimai nėra keičiami, sukūrus dar vieną taisyklių rinkinį su ta pačia statine klase, vykdymo metu pranešama apie dubliuojamą VT rinkinį. Tačiau esant tik vienam taisyklių rinkiniui, su statine klase, pavadinimų galima nekeisti, nes jie neįtakoja vykdymo. Dinaminę klasę pakeitus statine pakeičiami klasės atributai. Tai įtakoja privalomus tolimesnius statinius pakeitimus. Svarbiausias iš jų yra objekto klasės parinkimas, sąlygos ir VT įvykio iniciavimo dalyse atributų keitimas.

Atliktas eksperimentas parodė, jog VT vykdymui įtakos neturi tik funkcijos tekstas, kuris pateikiamas vartotojui. Nekeičiant elementų identifikavimo pavadinimų, vykdymas gali būti įtakojamas, kuomet sukuriamas daugiau nei vienas jau egzistuojantis elementas. Tokiu atveju, vartotojas bus įspėtas apie besidubliuojančius elementus arba neidentifikuotus elementus. Didžiausią įtaką VT vykdymui turi naudojamų pavienių atributų neatitikimas pasirinktos klasės atributams.

Lentelė Nr. 16 *Dinaminės klasės keitimas statine VT rinkinyje*



Lentelė Nr. 17 *Dinaminės klasės keitimas statine VT rinkinyje vertinimas*

	Dinamiškai keičiama	**Šalinama**	**Statiškai keičiama**	**Galima nekeisti**
Taisyklių rinkinys	-	Pavadinimas	Pavadinimas	Pavadinimas
Taisyklės	Klasės atributai	Pavadinimas	Pavadinimas	Pavadinimas
Objektas	-	Pavadinimas	Klasė	Pavadinimas
Veiksmo dalis	Rinkinio elementai	-	Rinkinys	-
Išvados tekstas	-	-	-	Jei išvados tekstas tinka formuojamam
Sąlygos dalis	Objektas	Reikšmė	Operatorius, atributas, reikšmė	Operatorius, reikšmė, objektas
Įvykio iniciavimo taisyklė	-	-	Klasė, atributas, pavadinimas, vykdymo funkcija	Pavadinimas, funkcijos tekstas

6. IŠVADOS

1. Atlikus veiklos taisyklių galimų struktūrų ir jų sudarymo būdų analizę, nustatyta, kad grafiniu būdu modeliuoti veiklos taisyklės gali tiek IT ekspertai, tiek verslo atstovai. Vis dėlto, šiuo metu egzistuojančios VT grafinio modeliavimo priemonės nėra lanksčios ir neužtikrina greito ir efektyvaus veiklos taisyklių sukūrimo būdo. Pagrindinės priežastys yra tos, kad netgi sudarant VT grafinėmis priemonėmis, verslo atstovai turi turėti programavimo žinių, tokiu būdu yra sulėtinamas jų darbas ar visiškai sustabdomas. Taip pat, kuriant panašios struktūros taisykles, jas reikia sukurti kiekvieną atskirai, dėl šios priežasties kūrimo laikas didėja priklausomai nuo taisyklių kiekio.

2. Išnagrinėta veiklos taisyklių formavimo etapų integracija informacinių sistemų kūrimo etapuose. Nustatyta informacija reikalinga veiklos taisyklės kūrimo inicijavimui bei sudarytas VT formavimo planas, tokiu būdu užtikrinamas nuoseklus jų realizavimas, atitikimas iškeltiems reikalavimams bei korektiškas vykdymas veiklos taisyklių valdymo sistemoje.

3. Atlikus egzistuojančių veiklos taisyklių valdymo sistemų analizę (Blaze Advisor, JRules), atliktas jų palyginimas, nustatyti esami privalumai ir trūkumai. VTVS naudojamos produkcinės taisyklės, kurių struktūra yra lengvai suprantama, jos dėka galima sudaryti įvairaus sudėtingumo VT. Nustatyti pagrindiniai trūkumai yra šie: nėra galimybių kurti daugkartinio panaudojimo veiklos taisykles, VT kūrime verslo atstovas negali savarankiškai kurti VT, vykdomų VT taisyklių koregavimas atliekamas kūrimo aplinkoje, t.y. jį atlieka IT ekspertas ar verslo atstovas, jei turi IT įgūdžių.

4. Darbe sudaryta šablonais grindžiama veiklos taisyklių sudarymo metodika. Ja remiantis, sudaromas veiklos taisyklės šablonas, kurį naudojant suformuojamos reikiamos veiklos taisyklės. Šablono naudojimas užtikrina efektyvų daugkartinį tos pačios struktūros veiklos taisyklių sukūrimą, kadangi jos sudaromos užpildant to paties šablono egzempliorius duomenimis (nurodomos sąlygos, išvados). VTVS leidžia administruoti šablonų pagrindu sukurtas VT per internetinę sąsaja, tokiu būdu verslo atstovas vienas gali koordinuoti VT.

5. Pagrindiniai metodo ypatumai yra šie:

- Sudaryta šablonų, skirtų veiklos taisyklių kūrimui, bei veiklos taisyklių struktūra. Joje nurodomi pagrindiniai elementai, reikalingi veiklos taisyklių valdymo lankstumui pasiekti bei taisyklės struktūros pilnumui ir teisingumui užtikrinti.
- Suformuotas algoritmas leidžia nuosekliai sudaryti šabloną nuo atominių elementų sukūrimo iki šablono pateikimo VT kūrimui. Tokiu būdu užtikrinamas korektiškas veiklos

taisyklių vykdymas VTVS. Algoritme identifikuoti visi galimi veiklos taisyklių elementai, dėl to galima sudaryti įvairaus sudėtingumo veiklos taisyklių šablonus.

– Sudarytas algoritmas, skirtas veiklos taisyklių kūrimui, pagal sudarytus šablonus, atsižvelgiant į VTVS funkcionalumą bei VT vykdymą ir administravimą. Šis algoritmas skirtas įvairaus skirtingumo VT kurti, tokiu būdu užtikrinamas korektiškas VT vykdymas.

6. Siūlomos metodikos efektyvumui nustatyti, atliktas tyrimas, kurio metu buvo analizuojami studijų administravimą valdančios veiklos taisyklės. Pagal pasiūlytą metodiką buvo sudaryti šablonai, kurių egzempliorius užpildžius buvo gautos VT, reikalingos valdyti šiuos procesus. Taip pat buvo tiesiogiai kuriamos (programuojamos) tiems patiems procesams valdyti veiklos taisykles. Išanalizavus tyrimo rezultatus nustatytą, kad :

- Kuriant programuojamas VT, manipuliuojama klasėmis ir jų atributais, naudojami sudėtingi SRL sintaksiniai dariniai, kurių sudėtingumas priklauso nuo VT sudėtingumo. VT kūrimo šablonais metu, veiklos atstovas praleidžia VT turinio programavimo dalį. Čia jis pildo suprojektuotą šablono turinį konkrečiais duomenimis, tokiu būdu netiesiogiai keičiama VT turinio programinė dalis. Todėl, veiklos taisyklių kūrimo procese IT specialisto vaidmuo sumažėja 30%, veiklos taisyklių realizavimo tikimybė veiklos atstovo atžvilgiu siekia beveik 100%. 80% lieka IT specialistui šablono struktūros keitimui.
- Programuojamų VT redagavimas atliekamas programiniame kode, todėl veiklos atstovas be IT specialisto pagalbos to atlikti negali, išskyrus tuos atvejus, kuomet atliekamas globalių klasių ir kitų elementų identifikavimo projekte, bet ne programiniame kode, redagavimas. Šablone leidžiamas konkrečių laukų redagavimas, kuris gali būti reikšmės šalinimas arba elemento pakeitimas. Tokiu būdu veiklos atstovas netiesiogiai atlieka programinio kodo parametrų redagavimą. Tai leidžia sukurti vykdomas VT, kurios atitinka SRL semantiką.
- Programuojamoje VT aplinkoje,VT elementų šalinimas atliekamas programiškai, siekiant išlaisvinti atmintį. Globalių ar lokalių projekto elementų šalinimas sukelia pažeidimus VT programiniame kode. Tokiu būdu pažeidžiami elementų pasiekiamumo aprašai, neleidžiantys vykdyti VT. Šablone užtikrinama galimybė, kad veiklos atstovas pašalindamas šablono egzempliorių su užpildytais duomenimis, nepažeis VT programinio kodo. Pakartotinis egzemplioriaus kūrimas ir užpildymas garantuoja VT teisingumą vykdymui.

7. Tyrimo medžiagos pagrindu parašytas straipsnis „Veiklos tasyklių, grindžiamų šablonais, valdymas informacinėse sistemose”, kuris buvo pristatytas Tarpuniversitetinėje magistrantų ir doktorantų mokslinėje konferencijoje.

7. LITERATŪRA

[1] **D'Hondt M, Gybels K, Jonckers V.** Seamless Integration of Rule-Based Knowledge and Object-Oriented Functionality with Linguistic Symbiosis. *ACM simpoziumas tema kompiuterių panaudojimas.* [Interaktyvus] 2004, žiūrėta [2009.05.12]. Prieiga per internetą: <http://portal.acm.org/citation.cfm?id=968168> .

[2] **McDermid D.C.v** The development of Business Rules Diagram. *Daktaro disertacija*. Curtin University of Technology, Austrija, 1998.

[3] **Lin L.et al.** Facilitating the Implementation and Evolution of Business Rules. *Programų priežiūra*: *tarptautinės konferencijos medžiaga* (ICSM'05). 2005, p. 609- 612.

[4] **Xu Y, Yang H, Amin I.** Business Rule Based Program Transformation for CRM System Evolution. Information Reuse and Integration: tarptautinė konferencijos medžiaga. USA, 2006 , p. 244-247.

[5] **OMG.** Semantics of Business Vocabulary and Business Rules (SBVR). *Object management group*. 2006, žiūrėta [2009.05.12]. Prieiga per internetą: <http://www.omg.org/docs/dtc/06-03-02.pdf >.

[6] **Ali. S, Soh.B, Torabi.T** Using Software Engineering Principle to Develop Reusable Business Rules. *Information and Comunication Technologies*: *tarptautinės konferencijos pranešimų medžiaga.* Australija, 2005, p. 276-283.

[7] **Prado Leite J.C.S, Leaonardi M.C.** Business Rules as Organizational Policies. *Ninth International Workshop on Volume*, 1998, p. 68 – 76.

[8] **The Business Rules Group.** Veiklos taisyklių manifestas: *taisyklių nepriklausomumo principai* [interaktyvus]. 2003, žiūrėta [2009.05.12]. Prieiga per internetą: <http://businessrulesgroup.org/brmanifesto/BRManifestLithuanian(v1.0).pdf >.

[9] **Herbst. H et al**. The specification of business rules: a comparison of selected methodologies. *IFIP Working Group 8.1 Conference CRIS 94*. Maastrich, 1994, p. 29-46.

[10] **Steinke G, Nickolette C.** Business rules as the basis of an organization's information system. *Volume 103*, 2003, Nr.1, p. 52-63.

[11] **Tabet S. et al.** OMG Production Rule Representation - Context and Current Status. *OMG*, *RuleLanguages for Interoperability* [interaktyvus] W3C, 2005, žiūrėta [2009.05.12]. Prieiga per internetą: <http://www.w3.org/2004/12/rules-ws/paper/53>

[12] **Antoniou G., Arief M.** Executable declarative business rules and their use in electronic commerce. *ACM symposium on Applied computing*. Madrid, 2002, p. 6-10.

[13] **Charif. A, Mezini M.** Hybrid web service composition: business processes meet business rules. *2nd international conference on Service oriented computing*. New York, 2004, p. 30-38.

[14] **D'Hont M., Vivane J.** Hybrid aspects for weaving object-oriented functionality and rule-based knowledge. *3rd international conference on Aspect-oriented software development.* Lancaster, 2004, p. 132-140.

[15] **Zsifkov N., Campeanu R.** Business rules domains and business rules modeling. *International symposium on Information and communication technologies.* Las Vegas, 2004, p. 172-177.

[16] **D'Hondt M., Gybels K. Jonckers V.** Seamless integration of rule-based knowledge and object-oriented functionality with linguistic symbiosis. *ACM symposium on Applied computing*. Nicosia, 2004, p. 1328-1335.

[17] **Pankowski T.** Active rules for business rules specification and control. [interaktyvus]. 1997, žiūrėta [2009.05.12]. Prieiga per internetą: <http://citeseer.ist.psu.edu/pankowski97active.html>

[18] **Ali S., Soh B., Torabi T.** Using Software Engineering Principles to Develop Reusable Business Rules. *Information and Communication Technologies: tarptautinė konferencija*. 2005, p. 276-283.

[19] **Motiejūnas L., Butleris R.** Business Rules manipulation model. *Information technology and control / - ISSN 1392-124X. - Kaunas. Vol. 36, No. 3, p. 295-301., 2007*

[20] **Taylor J.** Achieving decision consistency across the SOA-based enterprise. *Using business rules management system in an SOA*: *white paper*. 2005, p. 3-4.

[21] **Ross R**. Principles of the Business Rule Approach. Addison Wesley, ISBN: 0-201-78893-4, 2003.

[22] **Goedertier S., Vanthienen, J.** Rule-based Business Process Modeling and Execution. *Proceedings of the International IEEE EDOC Workshop on Vocabularies, Ontologies and Rules for The Enterprise,* 2005, p. 67-74.

[23] **Loucopoulos P., Kadir W. M. N. W**. BROOD: Business Rules-driven Object Oriented Design. *Journal of Database Management*, Vol. 19, issue 1, 2008, 41–73.

[24] **Von Halle B**. *Business rules applied – business better systems using the business rules approach.* Niujorkas, 2002.

8. TERMINŲ IR SANTRUMPŲ ŽODYNAS

VT – veiklos taisyklė.

DFD – informacijos srautų diagrama.

CPM – konceptualus vykdymo modelis. Pasiūlytas Merisės.

BRD – veiklos taisyklių diagramos. Technika, naudojama sistemų kūrimo, reikalavimų specifikavimo procese.

STD – būsenų perėjimų diagramos, naudojamos State-Transition modelyje veiklos taisyklių specifikavimui.

ER - RM – veiklos taisyklių grafinis specifikavimo metodas.

PVE – potencialių įvykių pažeidimų aibė.

VTVS – veiklos taisyklių valdymo sistema

Blaze Advisor – programinis sprendimas, veiklos taisyklių valdymo sistema.

JRules – ILOG programinis sprendimas, veiklos taisyklių valdymo sistema

WYSIWYG – akronimas pasakymui What You See Is What You Get (angl. tai ką matote atitinka tai ką gausite), kuris naudojamas kalbant apie kompiuterines programas, norint apibūdinti kokią nors programinę įrangą, kurios turinio išvaizda rengimo metu yra labai panaši į tos programinės įrangos pateikiamo galutinio produkto išvaizdą

BAL – (business action language) veiklos taisyklių aprašymo kalba pagal ILOG.

IRL – (ILOG rules language) skirta JRules gramatikos specifikavimui.

SRL – (structured rule language) kalba, naudojama veiklos taisyklėms užrašyti.

Rete – greitas algoritmas, pažymintis problemą. Daugiau žr.[1]

Rule server – išdėsto taisykles serveryje ar kitose Java aplinkose.

9. PRIEDAI

Lentelė Nr. 18 *Objektinio modelio elementai*

	Apima	Sintaksė
Object_ Model	Klasės apibrėžimas	**a**[**n**] *klasėsVardas* **is a**[**n**] *tėvinės_klasėsVardas* [**with** atributų aprašymo sąrašai] [inicializuotas teiginys]**.**
	Objekto apibrėžimas	*objketoVardas* **is a[n]** *klasėsVardas* [**with** ypatybių aprašymo sąrašas] [inicializuotas teiginys]**.** **arba** *objektoVardas* **is a**[**n**] **array of** *klasėsVardas* | *sąveikosVardas* | tradicinisKintTipas **to** *klasėsVardas* | *sąveikosVardas* | tradicinisKintTipas [**with** atributų aprašymo sąrašas] [**initially {** **it.append(***reikšmė***)** | **it.insert(***reikšmė***),** **it.append(***reikšmė***)**| **it.insert(***reikšmė***),** ... **}].** **arba** *objketoVardas* **is a**[**n**] **fixed array of** k*lasėsVardas* | *sąveikosVardas* | tradicinisKintTipas [**with** ypatybių aprašymo sąrašas] [**initially {** **it[***raktas***] =** *reikšmė***,** **it[***raktas***] =** *reikšmė***,** ... **}].**
	Atributų apibrėžimo rinkinys	**{** Atributo apibrėžimas [**,** Atributo apibrėžimas] ... **}**
	Atributo apibrėžimas	**a**[**n**] *atributoVardas* **: a**[**n**] tradicinisKintTipas **a**[**n**] *atributoVardas***: a**[**n**] *rinkinioVardas* **arba** **a**[**n**] *rinkinioVardas* **a**[**n**] *atributoVardas***: some** k*lasėsVardas* | *sąveikosVardas* **arba** **some** k*lasėsVardas* | *sąveikosVardas* **a**[**n**] *atributoVarda***: some association from** k*lasėsVardas* |

	Apima	Sintaksė
		sąveikosVardas | tradicinisKintTipas **to** k*lasėsVardas* | *sąveikosVardas* | tradicinisKintTipas **a**[**n**] *atributoVardas***: some** [**fixed**] **array of** k*lasėsVardas* | *sąveikosVardas* | tradicinisKintTipas
	Inicializuotas teiginys	**initially {** Priskyrimo_Teiginys [**,** Priskyrimo_Teiginys]... **}** **arba** **initially** iškviesti_metodą| įvykdyti_funkciją| sukurti_teiginį | priskirti teiginį
	Sąrašo apibrėžimas	**a**[**n**] sąrašoVardas **is one of {** *elementoVardas***,** *elementoVardas* [**,** *elementoVardas*] ... **}.**
	Šablono_Aprašymas	*šablonoVardas* **is any** *klasėsVardas* | *sąveikosVardas* [**in** *rinkinioVardas*] [**such that** *atributoVardas* lyginamieji_operatoriai pirminė_išraiška [**and** | **or** *atributoVardas* lyginamieji_operatoriai pirminė_išraiška]]**.**
	Kintamojo aprašymas	*kintamojoVardas* **is a**[**n**] pirminis_tipas [**initially true** | **false** | **unavailable** | **null** | **unknown** | *value*]**.** *kintamojoVardas* **is a**[**n**] *rinkinioVardas* [**initially** *rinkinioElementoVardas*]**.** ***arba*** *kintamojoVardas* **is some** *klasesVardas* | *rinkinioVardas* [**initially** *objektoIšraiška*]**.** *kintamojoVardas* **is some association from** *klasėsVardas* | *sąveikosVardas* | Pirminis_Tipas **to** *klasėsVardas* | *sąveikosVardas* | Pirminis_Tipas [**initially** *objektoIšraiška*]**.** *kintamojoVardas* **is some** [**fixed**] **array of** *klasėsVardas* | *sąveikosVardas* | Pirminis_Tipas [**initially** *objektoIšraiška*]**.**
	Pirminis_Tipas	**boolean** | **integer** | **real** | **string** | **timestamp** | **date** | **time** | **duration** | **money**

Lentelė Nr. 19 *Taisyklių rinkinių komponentai*

	Apima	**Sintaksė**
Taisyklių rinkinys (Ruleset)	Vykdymo_metodas	[optimized inference |inference | sequential| compiled sequential] Taisyklių_rinkinio_apibrėžimas
	Taisyklių_rinkio_apibrėžimas	**ruleset** *Taisyklių_rinkinio_Vardas* [**for {** Aprašytų_parametrųs_sąrašas **}**] [**returning a**[**n**]Pirminis_Tipas|*klasėsVardas*| *sąveikosVardas* | *sąrašas*] **is {** Taisyklių_rinkinio_Turinio_Apibrėžimas **}**
	Rulesets_Turinio_Apibrėžimas	[Objekto_aprašas] [Šablono_aprašas] [Kintamojo_aprašas] Taisyklės_aprašas [Taisyklės_aprašas] ... [Atributo_Įvykio_taisyklės_aprašas| Objekto_Įvykio_taisyklės_aprašas | Išorinės_Įvykio_taisyklės_aprašas]

Lentelė Nr. 20 *Išraiškų komponentai*

	Apima	**Sintaksė**
Išraiškos	Pirminės_Išraiškos	Boolean_Išraiška | Palyginimo_Išraiška | Skaitinė_Išraiška | Įvertinimo_Išraiška | Laiko_Išraiška
	Boolean_Išraiška	**true | false** Palyginimo_Išraiška **not** Boolean_Išraiška Boolean_Išraiška **and** Boolean_Išraiška Boolean_Išraiška **or** Boolean_Išraiška
	Skaitinė_Išraiška	Atributo_reikšmė | *value* Skaitinė_Išraiška Skaitinis_operatorius Skaitinė_Išraiška Išraiškos apskaičiuoja skaitines reikšmes. Naudojami skaitiniai operatoriai atlikti skaitines operacijas atitinkamose skaitinėse reikšmėse ar išraiškose.
	Įvertinimo_Išraiška	**at least** * ***n*** *klasėsVardas* [**such that** Boolean_Išraiška] **satisfy** Boolean_Išraiška **arba** **at least*** ***n*** *elemntoTipoVardas* **in** *rinkinioVardas* [**such that** Boolean_Išraiška] **satisfy** Boolean_Išraiška

	Apima	Sintaksė
		* gali būti ir šie identifikatoriai : **at most, every, exactly, zero** Grąžinamas loginis rezultatas. such that dalis nėra būtina ji pažymi apribojimą. Naudojama taisyklių sąlygose, funkcijose.

Lentelė Nr. 21 *Funkcijos apibrėžimas*

Funkcijos	Apima	Sintaksė
	Funkcijos_Apibrėžimas	**function** *funkcijosVardas* [**for** **{** Parametrų_Apprašymas_Sąrašas **}**] [**returning** **a**[**n**] Pirminis_Tipas|*klasėsVardas* |sąveikosVardas |*rinkinioVardas*] **is {** Teiginių_Blokas **}**
	Parametrų_Aprašymas	*parametroVardas* **:** **a**[**n**] Pirminis tipas | *klasėsVardas* | *sąveikosVardas* | **some association from** *klasėsVardas* **to** *klasėsVardas* | **some** [**fixed**] **array of** *klasėsVardas* | Pirminis tipas

Lentelė Nr. 22 *String tipo komandos*

String tipo raktai	Raktas	Aprašymas
	starts with	Tikrinamos dvi eilutės. Ar pirmoji prasideda su antrąja eilute
	does not start with	Tikrinamos dvi eilutės. Ar pirmoji neprasideda su antrąja eilute
	ends with	Tikrinamos dvi eilutės. Ar pirmoji baigiasi su antrąja eilute
	does not end with	Tikrinamos dvi eilutės. Ar pirmoji nesibaigia su antrąja eilute
	is blank	Tikrinama ar eilutė tuščia
	is not blank	Tikrinama ar eilutė nėra tuščia
	contains text	Tikrinamos eilutės pažiūrint ar pirmoji turi įvykius aptinkamus antrojoje.

String tipo raktai	Raktas	Aprašymas
	is contained in text	Tikrinamos eilutės pažiūrint ar pirmoji įvyksta antrojoje
	contains match	Tikrinama ar įprasta išraiška valdoma eilutėje.
	does not contain match	Tikrinamas išraiškos sutapimas su eilute.
	exactly matches	Tikrinama ar paprasta išraiška pilnai apibrėžia eilutę.
	does not exactly match	Tikrinama ar paprasta išraiška nepilnai apibrėžt apibrėžia eilutę.

Lentelė Nr. 23 *Integruotos klasės*

Palaikomos klasės	Klasių kategorijos	Tipai, metodai, savybės
NdCalendar	Paprastų duomenų tipų reikšmės	Naudojama aprašant datos, laiko trukmės reikšmes. *date(), time(), timestamp(),currentDate(), currentTime(), currentTimestamp(),* - grąžina einamąją datą/laiką/pilną formatą. *monthWeekDay()* – grąžina konkrečią datą, kuri apibrėžiama nurodant metus, mėnesį metuose, savaitės dieną mėnesyje, dieną savaitėje. *yearWeekDay()* – grąžina nustatytų metų savaitės dienos konkrečią datą. *indexedMonthWeekDay(), indexedYearWeekDay()* tipo metodai (generuojantys naujas reikšmes, skaičiavimu, palyginimų, reikšmių priėjimų metodai) *isLeap()* – tikrina ar metai yra keliamieji, *month()* – mėnesį grąžina sveikuoju skaičiumi, *monthLength( )* – grąžina dienų skaičių nurodyto mėnesio, *monthName( ), shortMonthName(), shortWeekDayName()* – sutrumpinami pavadinimai, *weekDay()* - grąžina savaitės dieną, *weekDayName()*-

Palaikomos klasės	Klasių kategorijos	Tipai, metodai, savybės
NdDate NdTime NdTimestamp NdDuration		einamąją datą Date tipo metodai (skaičiavimų, palyginimų metodai) *add()* – grąžina laiko argumentą praplėstą nurodyta data, *equals()* – grąžina loginę reikšmę, tikrinama ar objektai turi tą pačią datą, *stampAt(), subtract()* – iš nurodytos datos atimama trukmė, *subtractInDays()* – grąžina skirtumą tarp datos reikšmės ir argumento, išreikšto dienų skaičiumi Time tipo metodai (skaičiavimų, palyginimų metodai) *add(),equals(),inDaylightSavings()* – laiko zona, *stripTimeZone(),subtract()* – analogiška kaip laiką atimti iš trukmės, Timestamp tipo metodai (skaičiavimų, palyginimų metodai) *add(), equals(), inDaylightSavings(), stripTimeZone(), subtract(), subtractInDays(), subtractInMilliseconds(), subtractInMonths()*. Metodai analogiški NdTime, tačiau čia operuojama su Timestamp. Duration tipo metodai (skaičiavimų, palyginimų metodai) *add(), compare()* – lygina pradinė trukmę su argumento reikšme, *equals()* – grąžina loginę reikšmę, tikrindmai ar argumentai yra tos pačios trukmės, *invert()* – konvertuoja trukmės skaičių į priešingą, *multiply()* - grąžina reikšmę, kuri gaunama trukmę padauginant iš nurodytos argumento reikšmės, *subtract()* – atima trukmes.
NdCurrencyManager	Pinigų reikšmės	*Savybės:* formatMode – nurodomas išvesties formatas, roundingMode –

Palaikomos klasės	Klasių kategorijos	Tipai, metodai, savybės
NdCurrency		kokiu tikslumu apvalinama, defaultMappingCurrency *Metodai:* *find(), lookup()* - grąžina valiutą, kuri atitinką argumentą, *register()* – grąžina pakeistos valiutos argumento reikšmę. *Savybės:* decimalPrecision – kokiu tikslumu rodyti valiutos reikšmę, name – ISO valiutos kodas *Metodai:* žiūrėti NdCurrencyManager
NdMoneyAmount		Savybės: currency, decimalPrecision, Value – konvertuoja sumą i real tipą. *Metodai:* *add()* – atlieka sumavimo funkciją, *convertTo(), divide()* – dalijimo funkcija, *equals()* – garzina loginę reikšmę, tikrinant ar argumentai yra tos pačios sumos ir valiutos, *invert(), multiply()* - kelia laipsniu, *preciseAt()* – grąžina reikšmę nurodytu tikslumu, *substract()* – atimtis.
Assert	brUnit teiginiai	*Metodai:* *assert().assert* – grąžina void, *assert().check* – grąžina loginę reikšmę, *assert().fail* – grąžina void, *assert().assertEquals, assert().checkEquals, assert.fail()*
NdReasonCode	Taškų modeliai ir motyvų kodas	*Savybės:* Code (string tipo), message (string tipo) – pranešimas, kuris suietas u priežasties kodu, name (string tipo) – priežasties kodo pavadinimas, rank (integer) – rangas susiejamas su priežasties kodu.
NdScoreCharacteristic		*Savybės:* baselineScore (real) – pradinis taškas, binLabel (string) – aprašas priežasties, dėl kurios sumažinami taškai, characteristicName (string), maxBinScore (real) – maksimalus taškų skaičius, jei leidžiama išraiška nėra įvedama, partialScore (real) – daliniai taškai,

Palaikomos klasės	Klasių kategorijos	Tipai, metodai, savybės
NdScoreModelReason_ _CalculationOptions		reasonCode (NdReasonCode) – priežasties kodas grąžinamas charakteristikai. topReasons - Reason Code List sąrašo elementai, kurie buvo gauti pritaikius įverčių modelį. Šį objektų masyvą galima gauti iškvietus NdScoreModelReturnInfo objekto metodą calculateReasons(). unexpectedCount - Kiekis charakteristikų, kuriose žymės „*unexpec–ted*“ reikšmė nustatyta i *true*. *Metodai:* *dedupeMethod()* – suskaidomas dublikuotas priežasties kodas, *includeOnlyPositiveDistances()* – nustatoma loginė reiklšmė, pagal kurią skaičiuojamos teigiamos arba neigiamos atstumo reikšmės , *numberOfReasons()* – nurodomas priežasties skaičius, *sortOrder()* – kodų klasifikavimas pagal distanciją, *allowExpressions()* – rašomi teiginiai, pagal kuriuos apskaičiuojami daliniai taškai bin.
NdScoreModelInfo		*Savybės:* characteristicCount – taškų modelio charakteristikų sveikojo skaičiaus grąžinimas (Įvertinamų kriterijų kiekis įverčių modelyje), score – taškų modelio išvestis (Balų suma, gauta atlikus įvertinimą), scoredCharacteristics – masyvas charakteristikų, kuriame pateikiamos detalės apie charakteristikas (Objektai aprašantys sukurtų įvertinimo kriterijų charakteristikas: pavadinimą, maksimalų įvertinimą ir t.t.) scoreModelName – taškų modelio pavadinimas. *Metodai*: *addReason (ndReasonCode, real)* – naudojama pridėti priežasties kodą ir rangą ar distanciją prie taškų ir priežasčių skaičiavimo taškų modeliui, *calculateReasons* – metodas vietoj įvesties ima (NdScoreModelReasonCalculationOptions) ir grąžina fiksuotą NdReasonCode masyvą , *calculateReasons()* – spausdina masyvo NdReasonCode taškų modelio

Palaikomos klasės	Klasių kategorijos	Tipai, metodai, savybės
		priežastis.
regularExpression	Reguliarusis teiginys	*Metodai:* *Match()* – string tipo eilutėje ieško tiksliai atitinkančius nurodytus simbolius ar jų seką, jei simbolių eilutė atitinką nurodytą, tuomet ji išvedama į ekraną priešingu atveju grąžinama nulinė reikšmė. *substitute()* – naudojamas pakeisti kiekvieną match dalies vietą pirmąjį simbolių eilutės argumentą antruoju to paties tipo argumentu , *split()* – išskaido simbolių eilutę, gražindamas tik tuos simbolius, kurie yra prieš ir po regularExpression.