



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ SISTEMŲ KATEDRA

Martynas Remeika

**TAISYKLĖMIS GRINDŽIAMŲ DINAMINIŲ VERSLO PROCESŲ
MODELIAVIMO IR SIMULIACIJOS BŪDAS**

**METHOD FOR MODELING AND SIMULATION OF RULE BASED
DYNAMIC BUSINESS PROCESSES**

Baigiamasis magistro darbas

Informacinių sistemų programų inžinerijos studijų programa, valstybinis kodas 621E15002

Informacinių sistemų specializacija

Informatikos inžinerijos mokslo kryptis

Vilnius, 2014



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ SISTEMŲ KATEDRA

TVIRTINU
Katedros vedėjas

(Parašas)

(Vardas, pavardė)

Martynas Remeika

**TAISYKLĖMIS GRINDŽIAMŲ DINAMINIŲ VERSLO PROCESŲ
MODELIAVIMO IR SIMULIACIJOS BŪDAS**

**METHOD FOR MODELING AND SIMULATION OF RULE BASED
DYNAMIC BUSINESS PROCESSES**

Baigiamasis magistro darbas

Informacinių sistemų programų inžinerijos studijų programa, valstybinis kodas 621E15002

Informacinių sistemų specializacija

Informatikos inžinerijos mokslo kryptis

Vadovas prof. dr. O. Vasilecas (Moksl. laipsnis, vardas pavardė) (Parašas) (Data)

Konsultantas T. Savickas (Moksl. laipsnis, vardas pavardė) (Parašas) (Data)

Konsultantas R. Grašys (Moksl. laipsnis, vardas pavardė) (Parašas) (Data)

Lietuvių kalbos konsultantas doc. dr. A. Kaulakienė (Moksl. laipsnis, vardas pavardė) (Parašas) (Data)

Vilnius, 2014

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ SISTEMŲ KATEDRA

Informatikos inžinerija studijų kryptis

Informacinių sistemų programų inžinerija studijų programa,

valstybinis kodas 621E15002

Informacinių sistemų specializacija

TVIRTINU

Katedros vedėjas

(Parašas)

(Vardas, pavardė)

(Data)

**BAIGIAMOJO MAGISTRO DARBO
UŽDUOTIS**

.....Nr.

Vilnius

Studentui Martynui Remeikai

Baigiamojo darbo tema: „Taisyklėmis grindžiamų dinaminių verslo procesų modeliavimo ir simuliacijos būdas“ patvirtinta 201...m. d. dekanas potvarkiu Nr.

Baigiamojo darbo užbaigimo terminas 2014 m. d.

BAIGIAMOJO DARBO UŽDUOTIS:

1. Atlikti literatūros ir susijusių sąvokų analizę.
2. Ištirti simuliacijos būdus ir egzistuojančias problemas.
3. Pasiūlyti problemų sprendimo būdus ir juos pagrįsti.
4. Realizuoti pasiūlytą metodą ir sukurti simuliacijos įrankio prototipą.
5. Atlikti eksperimentinį tyrimą ir pagrįsti rezultatus.

Baigiamojo darbo rengimo konsultantai:

doc. dr. Angelė Kaulakienė

(Moksl. laipsnis/pedag.vardas, vardas, pavardė)

Titas Savickas, Remigijus Grašys

Vadovas
(Parašas)

prof. dr. Olegas Vasilecas
(Moksl. laipsnis/pedag.vardas, vardas, pavardė)

Užduotį gavau

.....
(Parašas)

Martynas Remeika

.....
(Data)

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INFORMACINIŲ SISTEMŲ KATEDRA

Informatikos inžinerijos studijų kryptis

Informacinių sistemų programų inžinerijos studijų programa, valstybinis kodas 621E15002

Informacinių sistemų specializacija

PAŽYMA
APIE BAIGIAMĄJĮ MAGISTRO DARBĄ

.....Nr.

Vilnius

Studentas Martynas Remeika

Studento studijų svertinis įvertinimų vidurkis.....balo.

Baigiamojo darbo tema: Taisyklėmis grindžiamų dinaminių verslo procesų modeliavimo ir simuliacijos būdas

Baigiamasis darbas peržiūrėtas ir studentui Martynui Remeikai leidžiama ginti šį baigiamąjį darbą magistro laipsnio suteikimo komisijoje.

Katedros vedėjas

.....
(Parašas)

.....
(Moksl. laipsnis/pedag. vardas, vardas, pavardė)

VADOVO ATSILIEPIMAS
APIE BAIGIAMĄJĮ MAGISTRO DARBĄ

2014.06.04

Baigiamajame darbe nagrinėjamas taisyklėmis grindžiamų dinaminių verslo procesų modeliavimas ir simuliacija. Analizuojami susiję moksliniai darbai, verslo taisyklių valdymo sistemos, verslo procesų simuliacijos būdai ir priemonės bei analizuojamoje srityje egzistuojančios problemos. Nustatyta, kad esami verslo procesų modeliavimo ir simuliacijos būdai nėra tinkamai pritaikyti dinaminių verslo procesų simuliacijai. Darbe pasiūlytas problemos sprendimą, kuris išplečia simuliacijos būdą, papildydamas jį galimybe vykdyti taisykles dinamiškai. Pasiūlytas metodas realizuotas autoriaus sukurtame simuliacijos įrankio prototipe. Atliktas prototipo eksperimentinis tyrimas bei pateiktos išvados ir pasiūlymai tolimesniems tyrimams.

Autorius išnagrinėjo aktualią informatikos inžinerijos problemą, atliko prasmingą darbą ir sukūrė naują lanksčių ir adaptyvių verslo procesų simuliacijos būdą. Atliktas eksperimentas parodė, kad metodas gali būti įgyvendintas ir panaudotas įmonių verslo procesų simuliacijai bei tobulinimui.

Siūlau darbą vertinti puikiai (10).

Vadovas

.....
(Parašas)

prof. dr. Olegas Vasilecas
(Moksl. laipsnis/pedag. vardas, vardas, pavardė)

Vilniaus Gedimino technikos universitetas
Fundamentinių mokslų fakultetas
Informacinių sistemų katedra

ISBN ISSN
Egz. sk.
Data-.....-.....

Antrosios pakopos studijų **Informacinių sistemų programų inžinerijos** programos magistro baigiamasis darbas

Pavadinimas **Taisyklėmis grindžiamų dinaminių verslo procesų modeliavimo ir simuliacijos būdas**
Autorius **Martynas Remeika**
Vadovas **prof. dr. Olegas Vasilecas**

Kalba: lietuvių

Anotacija

Šiame darbe nagrinėjamas taisyklėmis grindžiamų dinaminių verslo procesų modeliavimas ir simuliacija. Verslo procesai ir informacinės sistemos yra neatsiejamos verslo sistemų dalys. Sudėtingėjant informacinėms sistemoms, jų pritaikymas prie pasikeitusių verslo poreikių darosi sudėtingas ir brangus procesas. Kintant poreikiams ir verslo aplinkai, verslo procesų vykdymo metu reikia juos keisti ir pritaikyti. Verslo procesai, kurie keičiami vykdymo metu, yra dinaminiai. Todėl yra svarbu informacinių sistemų analizės fazėje turėti galimybę atlikti dinaminių verslo procesų simuliaciją ir taip įvertinti verslo procesų pasikeitimų poveikį.

Darbe analizuojami dinaminiai verslo procesai, taisyklėmis grindžiamų verslo procesų simuliacijos įrankių ir verslo taisyklių valdymo sistemų pritaikymo galimybės dinaminių verslo procesų analizei atliekant simuliaciją. Pasiūlytas metodas, leidžiantis dinamiškai keisti taisykles simuliacijos vykdymo metu. Remiantis pasiūlytu metodu, suprojektuota dinaminių verslo procesų simuliacijos įrankio architektūra ir sukurtas jos prototipas. Juo remiantis atliktas eksperimentinis tyrimas, kuris parodo įprastos ir dinaminės simuliacijos skirtumus bei privalumus.

Darbo apimtis – 85 p. teksto be priedų, 38 pav., 5 lent., 41 bibliografinis šaltinis. Atskirai pateikiami darbo priedai.

Prasminiai žodžiai: Verslo procesas, verslo taisyklė, dinaminis verslo procesas, modeliavimas, simuliacija, dinaminių verslo procesų simuliacija.

Vilnius Gediminas Technical University
Faculty of Fundamental Sciences
Department of Information Systems

ISBN ISSN
Copies No.
Date-.....-.....

Master Degree Studies **Information Systems Software Engineering** study programme Master Graduation Thesis

Title **Method for modeling and simulation of rule based dynamic business processes**
Author **Martynas Remeika**
Academic supervisor **Prof Dr Olegas Vasilecas**

Thesis language: Lithuanian

Annotation

This paper deals with dynamic rule-based business process modeling and simulation. Business processes and information systems are integral parts of business entities. Increasing complexity of information systems makes their adaptation to changing business needs a complex and expensive process. As business needs and environment changes, business processes also have to be changed and adapted accordingly. Processes that change during their execution are called dynamic. It is important to have a possibility of dynamic business process simulation while evaluating information systems, to assess the effect of changes in business processes.

This work deals with dynamic business processes, rule-based business process simulation tools, business rules management systems, and possibilities of their application to analyze business processes using simulation. A method is proposed that allows dynamic changes in business rules during the simulation. Based on the proposed method a dynamic business process simulation tool was designed, and a prototype was developed. A simulation experiment was carried out using this tool, and the experiment shows differences between dynamic and regular simulation methods, as well as their advantages.

Thesis consists of: 85 pages of text without appendixes, 38 pictures, 5 tables and 41 references. Appendixes are supplied separately.

Keywords: Business process, business rule, dynamic business process, modeling, simulation, simulation of dynamic business process.

Vilniaus Gedimino technikos universiteto
egzaminų, sesijų ir baigiamųjų darbų rengimo
bei gynimo organizavimo tvarkos aprašo 2011-
2012 m. m.

1 priedas

(Baigiamojo darbo sąžiningumo deklaracijos forma)

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Martynas Remeika, 20121362

(Studento vardas ir pavardė, studento pažymėjimo Nr.)

Fundamentinių mokslų fakultetas

(Fakultetas)

Informacinių sistemų programų inžinerija, ISIfm-12

(Studijų programa, akademinė grupė)

BAIGIAMOJO DARBO (PROJEKTO) SĄŽININGUMO DEKLARACIJA

2014 m. gegužės 31 d.

Patvirtinu, kad mano baigiamasis darbas tema „Taisyklėmis grindžiamų dinaminių verslo procesų modeliavimo ir simuliacijos būdas“ patvirtintas 2013 m. spalio 28 d. dekanų potvarkiu Nr. 370fm, yra savarankiškai parašytas. Šiame darbe pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose.

Prenkant ir įvertinant medžiagą bei rengiant baigiamąjį darbą, mane konsultavo mokslininkai ir specialistai: Titas Savickas, Remigijus Grašys. Mano darbo vadovas prof. dr. Olegas Vasilecas.

Kitų asmenų indėlio į parengtą baigiamąjį darbą nėra. Jokių įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs (-usi).

(Parašas)

Martynas Remeika

(Vardas ir pavardė)

TURINYS

PAVEIKSLŲ SĄRAŠAS	12
LENTELIŲ SĄRAŠAS	14
SANTRUMPOS	15
ĮVADAS	16
Tiriamoji problema	16
Tyrimo objektas	16
Darbo tikslas	16
Uždaviniai	16
Temos naujumas	17
Temos aktualumas	17
Tyrimo metodika	17
Mokslinio darbo vertė	18
Darbo struktūra	18
1. LITERATŪROS ANALIZĖ	19
1.1. Verslo procesai	19
1.1.1. Verslo proceso apibrėžimas	19
1.2. Verslo taisyklės	21
1.2.1. Verslo taisyklės apibrėžimas	21
1.2.2. VT klasifikavimas	21
1.2.3. Įvykis – sąlyga – veiksmas tipo taisyklės (ECA)	23
1.3. Verslo procesų modeliavimas	23
1.3.1. Modeliavimo sąvoka	23
1.3.2. Metamodeliai	24
1.3.3. VP modeliavimas	25
1.3.4. VP modeliavimo kalbos	25
1.3.5. VP modeliavimo požūriai	27
1.4. Verslo procesų simuliacija	27

1.4.1.	Simuliacijos apibrėžimas	27
1.4.2.	Modeliavimo ir simuliacijos procesas	28
1.5.	Dinaminiai verslo procesai	29
1.5.1.	Dinaminis VP valdymas	30
1.6.	Apibendrinimas.....	31
2.	VERSLO TAISYKLIŲ VALDYMO SISTEMŲ IR SIMULIACIJOS ĮRANKIŲ ANALIZĖ	32
2.1.	Taisyklėmis grindžiamų VP simuliacijos įrankių analizė.....	32
2.1.1.	IBM Websphere Business Modeler Advanced.....	32
2.1.2.	Simprocess.....	33
2.1.3.	Simul8.....	33
2.1.4.	AccuProcess Modeler	34
2.1.5.	Palyginimas	34
2.1.6.	Apibendrinimas	36
2.2.	BRMS architektūros analizė	36
2.2.1.	BRMS apibrėžimas.....	36
2.2.2.	BRMS architektūra	36
2.2.3.	Išvadų gavimo metodai.....	39
2.2.4.	Apibendrinimas	40
2.3.	BRMS analizė	40
2.3.1.	Drools	41
2.3.2.	OpenL Tablets	42
2.3.3.	OpenRules	43
2.3.4.	IBM ILog JRules	44
2.3.5.	BRMS galimybių palyginimas	45
2.3.6.	BRMS apibendrinimas	46
2.4.	Apibendrinimas.....	47
3.	TAISYKLĖMIS GRINDŽIAMŲ DINAMINIŲ VERSLO PROCESŲ SIMULIACIJOS METODAS.....	48

3.1.	Dabartinė situacija	48
3.1.1.	Įprastos ir dinaminės simuliacijos skirtumai	49
3.2.	Siūlomas metodas	50
3.2.1.	Naudojami terminai ir apibrėžimai.....	50
3.2.2.	Galimi problemos sprendimo būdai	51
3.2.3.	Siūlomo metodo aprašymas.....	51
3.2.4.	Siūlomo metodo nauda	54
3.3.	Dinaminių verslo procesų simuliacijos įrankio prototipo architektūra.....	55
3.3.1.	Simuliacijos įrankio architektūra.....	55
3.3.2.	Taisyklių mašinos architektūra.....	56
3.3.3.	Įvykių valdymo mašinos architektūra	57
3.3.4.	Simuliacijos valdymo komponento architektūra	58
3.3.5.	Grafinės vartotojo sąsajos architektūra.....	59
3.4.	Apibendrinimas.....	59
4.	SIMULIACIJOS ĮRANKIO PROTOTIPO REALIZACIJOS APRAŠYMAS.....	61
4.1.	Kūrimo priemonės	61
4.2.	Įrankio realizacija	61
4.2.1.	Paketų diagrama	61
4.2.2.	Taisyklių sintaksė, komandos.....	62
4.2.3.	Įvykių generavimo taisyklės.....	64
4.3.	Realizacijos aprašymas	65
4.3.1.	Taisyklių mašina.....	65
4.3.2.	Simuliacijos valdymo komponentas.....	71
4.3.3.	Įvykių valdymo mašina	73
4.4.	Vartotojo sąsaja.....	74
4.5.	Apibendrinimas.....	74
5.	EKSPERIMENTINIS TYRIMAS	75
5.1.	Verslo proceso modelis.....	75

5.2.	Simuliacinis modelis.....	75
5.3.	Eksperimentinis tyrimas	76
5.3.1.	Įvykiai simuliacijai	76
5.3.2.	ECA taisyklės simuliacijai.....	76
5.3.3.	Eksperimento eiga	77
5.3.4.	Simuliacijos rezultatų vertinimo kriterijai	77
5.3.5.	Simuliacijos pradiniai duomenys.....	77
5.4.	Eksperimento rezultatai	77
5.5.	Apibendrinimas.....	79
IŠVADOS.....		81
SIMULIACIJOS ĮRANKIO FUNKCIONALUMO DIDINIMO GALIMYBĖS IR TYRIMŲ KRYPTYS.....		82
LITERATŪRA		83
PRIEDAI		86

PAVEIKSLŲ SĄRAŠAS

1 pav. Bendra verslo proceso schema.....	21
2 pav. Metamodelio, modelio ir realaus pasaulio tarpusavio sąryšis	24
3 pav. Įvykiais valdomo verslo proceso modelio pavyzdys	26
4 pav. BRMS architektūra	37
5 pav. Drools žemo lygio taisyklės pavyzdys	41
6 pav. OpenL Tablets taisyklių užrašymo pavyzdys	42
7 pav. Taisyklių užrašymas naudojant lentelę	43
8 pav. Kodo fragmentas, parodantis OpenRules naudojimo būdą	44
9 pav. ILog JRules BRMS naudojama naturalia kalba užrašytos taisyklės ir veiksmo pavyzdys	45
10 pav. ILog JRules BRMS naudojama žemo lygio taisyklės ir veiksmo pavyzdys	45
11 pav. Simuliacijos schema Nr. 1	48
12 pav. Simuliacijos schema Nr. 2	49
13 pav. Statinė simuliacija.....	50
14 pav. Dinaminė simuliacija	50
15 pav. Siūlomas simuliacijos būdas	52
16 pav. Dinaminio verslo proceso taisyklių vykdymo algoritmas	53
17 pav. Veiklos “Surasti taisyklę vykdymui naudojant taisykles” detalus algoritmas.....	53
18 pav. Taisyklėmis grindžiamų dinaminių VP simuliacijos įrankio prototipo architektūra	55
19 pav. Taisyklių mašinos architektūra	56
20 pav. Įvykių valdymo mašinos architektūra	57
21 pav. Simuliacijos valdymo komponento architektūra	58
22 pav. Grafinės vartotojo sąsajos komponento architektūra.....	59
23 pav. Simuliacijos įrankio paketų diagrama.....	61
24 pav. Kolizijų sprendimo realizacijos klasių diagrama.....	66
25 pav. Valdymo komponentų klasių diagrama	67
26 pav. Vykdyto komponento klasių diagrama	68
27 pav. Transliatoriaus komponento klasių diagrama	69
28 pav. Sintaksės tinkamumo vertinimo komponento klasių diagrama	70
29 pav. Darbinės atminties klasių diagrama	70
30 pav. Taisyklių saugyklos klasių diagrama.....	71
31 pav. Simuliacijos valdymo komponentų klasių diagrama	72
32 pav. Simuliacijos rezultatų duomenų apdorojimo komponento klasių diagrama.....	73
33 pav. Įvykių generatoriaus komponento klasių diagrama	73

34 pav. Įvykių saugyklos komponento klasių diagrama	74
35 pav. Akcijų biržos prekybos verslo proceso modelis	75
36 pav. Akcijų biržos prekybos simuliacinis modelis	76
37 pav. Simuliacinis VP modelis su pakeistomis taisyklėmis	78
38 pav. VP modelis su įterpta veikla	79

LENTELIŲ SĄRAŠAS

1 lentelė. Verslo procesų apibrėžimai pagal autorius	19
2 lentelė. Verslo procesų klasifikavimas pagal grupes.....	22
3 lentelė. Taisyklėmis grindžiamų verslo procesų simuliacijos įrankių palyginimas	35
4 lentelė. BRMS palyginimas	46
5 lentelė. Simuliacijos rezultatai.....	78

SANTRUMPOS

BPM – verslo procesų modeliavimas (angl. *Business Process Modelling*)

BPM – verslo procesų valdymas (angl. *Business Process Management*)

BPMN – verslo procesų modeliavimo notacija (angl. *Business Process Modelling Notation*)

BRMS – verslo taisyklių valdymo sistema (angl. *Business Rule Management System*)

DB – duomenų bazė (angl. *Database*)

ECA – įvykis-sąlyga-veiksmas tipo taisyklė (angl. *Event-Condition-Action*)

EPC – įvykiais paremta procesų grandinė (angl. *Event-driven process chain*)

IT – informacinė technologija (angl. *Information Technology*)

PI – programinė įranga (angl. *Software*)

PS – programų sistema (angl. *Software System*)

UML – unifikuota modeliavimo kalba (angl. *Unified Modeling Language*)

VP – verslo procesas (angl. *Business Process*)

VT – verslo taisyklė (angl. *Business Rule*)

IVADAS

Tiriamoji problema

Vis labiau populiarėjant šiuolaikinėms technologijoms, kiekviena organizacija privalo taikytis prie naujų rinkos sąlygų ir tobulėti, todėl organizacijos savo veikloje naudoja įvairias programų sistemas, projektuoja informacines sistemas, kurios vykdo organizacijos numatytas funkcijas. Projektuojant įvairias, prie specifinių organizacijos poreikių pritaikytas informacines sistemas, analizuojant yra kuriami įvairūs verslo procesų modeliai. Modeliams yra įvedami verslo procesų veiklų apribojimai – verslo taisyklės ir atliekami modelių efektyvumo tikrinimo procesai – modelių simuliacija. Galiausiai verslo procesai yra automatizuojami kuriant programų sistemas. Tačiau verslo aplinka, sąlygos ir poreikiai nuolat kinta ir dažnai keičiantis reikalavimams pastebima, jog sunku įprastas informacines ir programų sistemas pritaikyti prie dažnai besikeičiančių verslo poreikių ir tokių sistemų priežiūra bei pritaikymas prie naujų poreikių yra brangus. Taip pat kintant poreikiams, verslo procesų vykdymo metu reikia juos keisti ir pritaikyti, nes kitu atveju, verslo procesų nutraukimas, jo keitimas ir paleidimas iš naujo yra neefektyvu ir kainuoja brangiai. Vienas iš pastaruoju metu vis labiau populiarėjančių būdų palengvinti informacinių ir programų sistemų pritaikymą prie dažnai kintančių verslo poreikių – atskirti verslo taisykles nuo programų sistemų ir valdyti verslo taisykles atskirai. Verslo taisyklėms atskirti nuo programų sistemų yra naudojamos verslo taisyklių valdymo sistemos. Tačiau verslo taisyklių valdymo sistemų pritaikymas naudojimui šiuolaikinėse programų sistemose yra ribotas, nes jos neleidžia vykdymo metu keisti verslo taisyklių bei lemti programų sistemų procesus. Iš to matyti, jog taisyklėmis grindžiamų dinaminių verslo procesų simuliacijos proceso atlikimo galimybės, panaudojant verslo taisyklių valdymo sistemas taip pat yra ribotas: nėra būdo, kuri leistų simuliacijos vykdymo metu keisti taisykles ir dinamiškai konstruoti vykdomą verslo procesą. Todėl nesant dinaminių verslo procesų simuliacijos būdai, yra sunku ištirti kokį poveikį turės organizacijoje vykdomų verslo procesų pakeitimai.

Tyrimo objektas

Tyrimo objektas – taisyklėmis grindžiamų dinaminių verslo procesų modeliavimas ir simuliacija.

Darbo tikslas

Pasiūlyti dinaminių verslo procesų simuliacijos būdą.

Uždaviniai

1. Atlikti su verslo procesais, verslo taisyklėmis, verslo procesų modeliavimu ir simuliacija susijusių mokslinių darbų analizę.

2. Atlikti verslo procesų simuliacijos priemonių ir verslo taisyklių valdymo sistemų analizę, ištirti priemonių galimybes atlikti dinaminių verslo procesų simuliaciją, simuliacijos būdus bei egzistuojančias problemas.
3. Remiantis analizės rezultatais, pasiūlyti metodą dinaminių verslo procesų simuliacijai.
4. Realizuoti pasiūlytą metodą ir sukurti simuliacijos įrankio prototipą.
5. Atlikti simuliacijos eksperimentinį tyrimą.
6. Suformuluoti išvadas ir pateikti tolimesnes mokslinių tyrimų kryptis.

Temos naujumas

Taisyklėmis grindžiamų verslo procesų modeliavimo ir simuliacijos tema nagrinėjama jau ne vieną dešimtmetį ir šiuo metu rinkoje yra keletas prieinamų taisyklėmis grindžiamų verslo procesų simuliacijos įrankių ir verslo taisyklių valdymo sistemų variantų. Tačiau verslo taisyklėmis grindžiamų dinaminių verslo procesų simuliacija yra nauja, nemažai susidomėjimo kelianti tema, nes ši sritis nėra iki galo ištyrinėta, joje trūksta bendro supratimo ir joje vis dar yra nemažai neatsakytų klausimų ir neišspręstų problemų.

Temos aktualumas

Darbe nagrinėjama problema šiandien yra ypač aktuali dėl technologijų paplitimo, konkurencijos didėjimo ir nuolat kintančios verslo aplinkos bei poreikių. Prie nuolat kintančių poreikių norint išlikti konkurencingiems rinkoje reikia gebėti prisitaikyti minimaliomis išlaidomis ir greitai (Martin et al. 2009). Šiuo metu rinkoje yra nemažai taisyklėmis grindžiamų verslo procesų simuliacijos įrankių ir verslo taisyklių valdymo sistemų, tačiau yra poreikis jas tobulinti ir plėsti jų galimybes, nes įrankiai nėra pritaikyti simuliuoti dinامينius verslo procesus. Todėl yra poreikis išplėsti priemonių galimybes taip, kad jas būtų palyginti nesudėtinga taikyti ir geriau jos prisitaikytų prie dinamiškų verslo poreikių. Todėl taisyklėmis grindžiamų dinaminių verslo procesų modeliavimas ir simuliacija yra aktuali tema mokslinių tyrimų aplinkoje, nes šioje srityje yra neišspręstų problemų.

Tyrimo metodika

Analitinėje darbo dalyje analizuojama mokslinė literatūra, kuri susijusi su verslo procesais, verslo taisyklėmis ir dinaminiais verslo procesais bei jų valdymu. Taip pat analizuojami rinkoje esančios verslo taisyklių valdymo sistemos bei plačiai naudojamų taisyklėmis grindžiamų verslo procesų modeliavimo ir simuliacijos įrankiai, naudojami būdai bei problemos. Įvertinus tyrimo rezultatus, siūlomas metodas leidžiantis vykdyti taisyklėmis grindžiamų dinaminių verslo procesų simuliaciją. Siūlomas metodas įgyvendinamas praktiškai ir atliekamas eksperimentinis tyrimas su pasirinktu verslo procesu, kad galima būtų ištirti metodo galimybes ir įvertinti privalumus.

Mokslinio darbo vertė

Nagrinėjama tema yra aktuali ir nauja dėl savo problemos pobūdžio, todėl šio darbo sukurti rezultatai, įrankiai, atlikti eksperimentai ir padarytos išvados padės ateities kartoms gilintis į problemą ir toliau plėtoti šio darbo metu iškeltas idėjas bei sukurtą simuliacijos prototipą. Ja remiantis, ateityje bus galima atlikti daugiau eksperimentų, atsakyti vis dar neatsakytus klausimus ir galbūt sukurti masiniam naudojimui skirtą dinaminių verslo procesų simuliacijos priemonę.

Darbo struktūra

Darbą sudaro įvadas, penki skyriai, išvados ir pasiūlymai:

Įvade iškeliama problema, formuluojami darbo tikslai ir uždaviniai, aptariama aktuali darbo tematika ir pristatoma darbo struktūra.

Pirmame skyriuje yra išdėstoma susijusių mokslinių darbų literatūros analizė, analizuojamos dalykinės srities sąvokos.

Antrame skyriuje nagrinėjami taisyklėmis grindžiamų verslo procesų simuliacijos įrankiai ir verslo taisyklių valdymo sistemos, jų veikimo metodas ir architektūra, pritaikymo galimybės atlikti dinaminių verslo procesų simuliaciją.

Trečiame skyriuje pateikiamas metodas, skirtas išplėsti taisyklių mašinų galimybes suteikiant galimybę vykdymo metu dinamiškai keisti verslo taisykles ir simuliacijos įrankio prototipo architektūra, leidžiantis vykdyti dinaminių verslo procesų simuliaciją.

Ketvirtame skyriuje aprašomas realizuotas simuliacijos įrankio prototipas.

Penktame skyriuje yra aprašomas eksperimentinis tyrimas, kuris atliekamas panaudojus sukurtą simuliacijos įrankio prototipą su pasirinktu eksperimentiniu verslo proceso modeliu. Pateikiami atlikto eksperimento rezultatai.

Išvadose ir pasiūlymuose pateikiami darbo išvados bei sukurto simuliacijos įrankio prototipo tobulinimo galimybės ir tolimesnių tyrimų kryptys, kurie gali būti vykdomi remiantis šio darbo rezultatais.

1. LITERATŪROS ANALIZĖ

Šio skyriaus tikslas yra išanalizuoti ir pateikti susijusių mokslinių darbų, sąvokų, metodų analizę, surinkti ir pateikti pagrindines nagrinėjamos dalykinės srities sąvokas bei teorinę medžiagą, kuri reikalinga nagrinėjant tiriamą problemą. Šiame skyriuje analizuojami ir pateikiami nagrinėjamos dalykinės srities – verslo procesų, verslo taisyklių bei verslo procesų modeliavimo ir simuliacijos sąvokos, apibrėžimai, analizuojami bruožai ir idėjos. Iš surinktų sąvokų bei paaiškinimų galima bus apibrėžti visiems priimtinus nagrinėjamos temos terminus ir išvengti dviprasmybių ar neteisingų interpretacijų šiame darbe.

1.1. Verslo procesai

1.1.1. Verslo proceso apibrėžimas

Tarptautinių terminų žodyne (Tarptautinių žodžių žodynas 2014) žodis „procesas“ apibrėžiamas taip: „eiga, vyksmas, raida, būvių kaita, veiksmas“, „priežastiniais ryšiais susijusių, vienas po kito vykstančių pokyčių seka“, „sistemos būsenų kaita“. Remiantis literatūros šaltiniais (Paul 1998; Kontrimas 2005) įvairių autorių verslo procesų (toliau – VP) apibrėžimai pateikiami 1 lentelėje.

1 lentelė. Verslo procesų apibrėžimai pagal autorius

Autorius, metai	Apibrėžimas
Davenport and Short, 1990	Logiškai susijusių uždavinių rinkinys, kuris vykdomas, kad galima būtų pasiekti apibrėžtus verslo tikslus.
Omrani, 1992	Veiksmų ciklas, kuris paimtas kartu siekia verslo tikslo.
Hammer and Champy, 1993	Veiklų rinkinys, kuris kartu generuoja naudingą rezultatą klientams.
Ferrie, 1995	Iš žinomo veiklos pradžios taško galimų apibrėžti veiklų rinkinys.
Saxena, 1996	Susijusių darbų rinkinys, kuris charakterizuojamas specifinių įvesčių (angl. <i>input</i>) ir įvestinės reikšmės, kurios generuoja specifinį rezultatą.

R. Kontrimas, V. Bulbenkienė, 2005	Komplektas logiškai susijusių užduočių, kurios vykdomos, kad pasiektų apibrėžtus verslo tikslus.
------------------------------------	--

Iš pateiktos lentelės galime matyti, kad VP apibrėžimų yra daug, tačiau jų esmė yra panaši – tai ryšiai (sąryšis) tarp ėjimų ir išėjimų, kur įėjimai yra transformuoti į išėjimus, naudojant veiklos scenarijų, kai reikšmė yra pridedama įėjime (Kontrimas 2005). Todėl VP galima suvokti kaip įeitį, kurią procesas transformuoja į išvestį (rezultatą), kuris įgyvendina verslo tikslus.

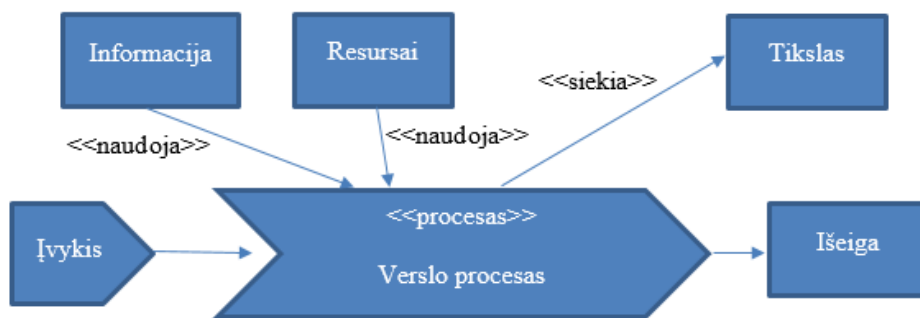
VP turi du svarbius bruožus:

1. Procesai turi klientus, t. y. procesai apibrėžia verslo tikslus, siekiamus rezultatus ir jų gavėjai yra klientai. Klientai gali būti vidiniai (pati organizacija) ir išoriniai (išoriniai organizacijos klientai).
2. Procesai pritaikyti organizacijai ir jie vykdomi jos viduje. Procesai yra nepriklausomos nuo formalios organizacinės struktūros.

Remiantis šaltiniu (Eriksson 1999), verslo procesas:

1. Turi tikslą;
2. Turi įeitį (angl. *input*);
3. Turi išeigą (angl. *output*);
4. Naudoja resursus;
5. Turi veiklas, kurios vykdomos reikalinga tvarka;
6. Gali įgyvendinti ne vieną organizacijos tikslą;
7. Sukuria vertę klientui.

Šaltinis (Eriksson 1999) pateikia (1 pav.) verslo bendrą VP schemą, kuri parodo aukščiau aprašytą VP sudėtį ir jų tarpusavio sąryšį. Pagal schemą, VP inicijuoja įvykis, jai vykdyti yra reikalinga informacija ir resursai. VP tikslas – tai tikslas, kurią įgyvendina procesas, o išeiga – rezultatas, kuris yra naudingas klientui.



1 pav. Bendra verslo proceso schema

VP pavyzdžių praktikoje gali būti labai įvairių, pavyzdžiui: vartojimo kredito suteikimas, ataskaitos paruošimas, priėmimas į universitetą, akcijų biržos prekyba, užsakymo vykdymas ir t. t..

1.2. Verslo taisyklės

1.2.1. Verslo taisyklės apibrėžimas

VP veiklą ir jo funkcionavimo būdą reguliuoja verslo taisyklės (toliau – VT). Įvairiose mokslinės literatūros šaltiniuose galima rasti skirtingus VT apibrėžimus. Pavyzdžiui: VT yra pažodžiui užkoduotos verslo praktikos žinios, o IT požiūriu – „VT - tai nedaloma daugkartinio naudojimo verslo logikos dalis“ (Hay 1993).

VT yra teiginys, kuris apibrėžia arba riboja tam tikrą verslo aspektą. Jis yra skirtas ginti verslo struktūrą arba kontroliuoti ir lemti verslo elgesį. VT, kuriuos susijusios su VP, yra atominės (Ross 2003; Mircea 2009) ir jos negali būti skaidomos į smulkesnes dalis.

Į taisyklės galima žiūrėti iš kelių požiūrio taškų: verslo, informacinių ir programų sistemų (Valatkaite et al. 2005). Iš verslo sistemos požiūriu – VT susijusi su apribojimais, taikomi organizacijos žmonėms, procesams ir lemia jų elgesį (pavyzdžiui: nuo rūkymo apribojimų darbo metu, iki procedūrų užpildant sąskaitą-faktūrą). Iš informacinių sistemų požiūriu – VT susijusi su faktais, kurie apskaitomi kaip duomenų apribojimai. Apribojimai gali būti, pavyzdžiui, - kokie duomenys gali arba negali būti įvedami į informacinę sistemą. Iš programų sistemų požiūriu - VT yra transformuojamos į žemiausią lygmenį – programos kodą. Tai gali būti duomenų bazių trigeriai, įvairūs programos apribojimai ar sprendimų kelių nustatymo taisyklės. Todėl projektuojant programų sistemas, jos reikalavimai ir taisyklės yra iš aukščiausio lygmens – verslo lygmens yra transformuojamos ir detalizuojamos iki žemiausio – programų sistemų lygmens.

1.2.2. VT klasifikavimas

Analizuojant VT klasifikavimą į grupes, pastebėta jog nėra vieningo nuomonės VT klasifikavijos klausimais. Šaltinyje (Halle 2002), remiantis įvairių autorių pateikiamais VT

klasifikacijų būdais, sudaryta VT klasifikacijos lentelė (2 lentelė). Tačiau galima pastebėti, jog dažniausiai yra remiamasi ir naudojama *Business Rules Group* organizacijos pateikta taisyklių klasifikavimo schema.

2 lentelė. Verslo procesų klasifikavimas pagal grupes

Autorius	Klasifikavimas į
Business Rules Group (2000)	• Išvestis (skaičiavimai, išvados) • Struktūrinis teiginys (terminai, faktai) • Veiksmo teiginys (integralumo apribojimai, sąlygos, autorizacija)
Ross (2001)	• Faktus • Terminus • Apribojimus • Kilmę • Išvadą • Laiką (angl. <i>timing</i>) • Seką • Euristicą
C. J. Date (2000)	• Apribojimus • Būsenos apribojimus • Perėjimų apribojimus • Išvedimą • Apskaičiavimą • Išvadą
USoft Inc.	• Apribojimo • Elgsenos • Išvedimo • Atvaizdavimo • Nurodymų

1.2.3. Įvykis – sąlyga – veiksmas tipo taisyklės (ECA)

ECA (angl. *ECA – Event – Condition – Action*, toliau - ECA) taisyklių modelį galima suprasti taip – kai įvyksta *įvykis*, jeigu *sąlyga* yra tenkinama, tai atliekamas *veiksmas* (Vasilecas et al. 2008). Trumpai tokią taisyklę galima užrašyti: „JEI Įvykis IR Sąlyga TAI Veiksmas“. ECA taisyklės struktūra susideda iš trijų dalių:

- Įvykis, kuris nurodo įvykį, kuriam esant aktyvuojama taisyklė.
- Sąlyga, tai įvykio atsitikimo metu vertinama užklausa.
- Veiksmas yra vykdomas, kai įvykis įvyksta ir sąlyga yra tenkinama.

Taisyklėmis grindžiamų VP modeliavime ir simuliacijoje gali būti naudojamos ECA tipo taisyklės (Herbst 1997). VP galima apibrėžti kaip baigtinis ECA tipo taisyklių rinkinys. Tokiu būdu sumodeliuotas VP yra vykdomas tokiu būdu: kai įvyksta įvykis, iš taisyklių rinkinio yra išrenkamas taisyklių poaibis, kuris reaguoja į taisyklėje aprašytą įvykį. Kiekviena šio poaibio taisyklės sąlyga yra tikrinama (gali būti tikrinama tiek pagal įvykio parametrus, tiek pagal kitus) ir tokiu būdu yra sudaromas kitas taisyklės sąlygą tenkinančių taisyklių poaibis. Iš šio poaibio gali būti išrenkama viena taisyklė (naudojant konfliktų sprendimo būdą), kurio veiksmas vykdomas. Taip pat gali būti vykdomi visos sąlygas tenkinančių taisyklių poaibio taisyklių veiksmai. Veiksmų vykdymo rezultatas gali būti sugeneruotas kitas įvykis į kurį gali reaguoti kitos taisyklės. Tokiu būdu gali būti konstruojama taisyklių seka, kuri atitinka VP.

ECA tipo taisyklės yra aktualūs ne tik VP kontekste, tačiau ir aktyviuose duomenų bazėse. ECA tipo taisyklės naudoti yra naudinga, kadangi verslo reikalavimai gali būti užrašomi neformaliu taisyklių rinkiniu, todėl juos transformavus į formalių ECA tipo taisyklių rinkinį tokiu būdu yra sumodeliuojamas vykdomas VP modelis, kuris gali būti naudojamas simuliacijoje. Iš to išplaukia terminai „taisyklėmis grindžiamas verslo proceso modelis“ ir „taisyklėmis grindžiamų verslo procesų simuliacija“. Šio darbo kontekste toliau yra naudojamos ECA tipo taisyklės.

1.3. Verslo procesų modeliavimas

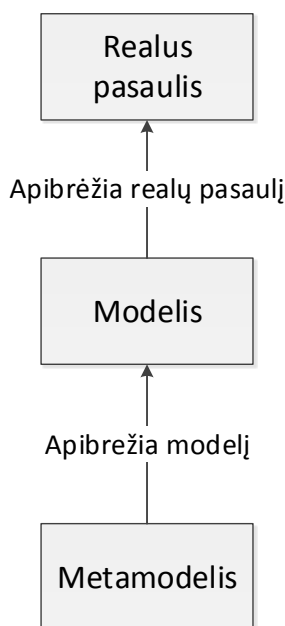
1.3.1. Modeliavimo sąvoka

Modelis - realaus reiškinių, proceso, struktūros, sistemos ar produkto abstrakcija, kuriamas spręsti dalykinės srities problemai (Gigch 1991; Maria 1997). Modelyje paliekamos tik tos savybės, kurios svarbios sprendžiamai problemai. Modelis gali būti tiek fizinis (pavyzdžiui, - pastato), tiek virtualus (pavyzdžiui, - verslo proceso, matematinis modelis). Modeliams kurti yra naudojamos modeliavimo priemonės.

Modeliavimas yra abstrakčių ir konceptualių, grafinių arba matematinių modelių generavimo procesas (Gigch 1991). Kitaip modeliavimas – modelio gaminimo procesas, kuris reprezentuoja konkrečios sistemos arba konstrukcijos veikimą ir sudėtį. Modeliavimo rezultatas yra modelis – abstraktus, supaprastintas realaus pasaulio pavyzdys. Vienas iš pagrindinių modeliavimo tikslų – padėti analitikams suprojektuoti sistemą, nustatyti galimas problemas ir atlikti eksperimentus - simuliaciją (atsakyti į klausimą „kas būtų, jeigu būtų?“). Modeliai turi būti kaip galima labiau supaprastinti ir sumažinti, nes dideli modeliai yra sunkiai suprantami ir sudėtinga su jais atlikti eksperimentus. Geras modelio pavyzdys turi būti tarp realistiškumo ir paprastumo.

1.3.2. Metamodeliai

Kad apibrėžti modelių formalumą, elementus iš kurių gali susidėti modeliai naudojama metamodelio ir metamodeliavimo sąvokos. Metamodelis – modeliavimo kalbos modelis, kuriame apibrėžtos esminės kalbos savybės. Metamodelis – darnus išreikštinių artefaktų ir taisyklių apibrėžimų rinkinys, kuris nusako kokie elementai gali būti modelyje, kaip jie gali būti tarpusavyje siejami ir kokie tų elementų tarpusavio sąryšiai (Aharoni et al. 2008). Metamodeliavimas – tai veikla, kurios vienas iš rezultatų yra metamodelio sukūrimas. Metamodeliavimas yra artimas modeliavimo procesui (Metamodels and metamodeling languages 2014). Metamodelis gali būti laikomas kaip išreikštinis aprašymas (dariniai ir taisyklės) kaip yra sukurtas tam tikros srities modelis. Tai yra formali specifikacija iš konkrečios dalykinės srities notacijų. Schemoje (2 pav.) parodytas autoriaus (Gigch 1991) pateikiamas metamodelio, modelio ir realaus pasaulio tarpusavio sąryšis. Žemiausiame lygmenyje yra metamodelis, kuris apibrėžia kaip gali būti sudaromi modeliai, o sukurtas modelis apibrėžia realų pasaulį.



2 pav. Metamodelio, modelio ir realaus pasaulio tarpusavio sąryšis

1.3.3. VP modeliavimas

VP modeliavimas (angl. BPM - *Business Process Modeling*) sistemų inžinerijoje yra veikla, kurios tikslas yra suteikti VP aiškumo, kurią galima panaudoti kaip įrankį tiek analizuojant, projektuojant VP, tiek kaip įrankį bendravimui tarp verslo atstovų, analitikų ir programuotojų (Aguilar-Saven 2003). Šios sąvokos nereikėtų painioti su VP valdymo sąvoka (angl. BPM – *Business Process Management*), kurios trumpinys irgi yra BPM. VP modeliavimą paprastai atlieka verslo analitikai, kurie siekia pagerinti VP efektyvumą ir kokybę. Naudojant šiuolaikinius VP modeliavimo ir simuliacijos įrankius, galima pakankamai efektyviai nustatyti VP problemas ir jas analizuojant bandyti išspręsti ir pagerinti VP. Tačiau nors ir turime pakankamai VP modeliavimo įrankių, ši sritis nėra tiek išstobulinta, kad VP modeliavimas ir jų simuliacija išspręstų visas verslo problemas ir padarytų jį labai efektyvų. Todėl ši sritis vis dar yra nemažai kelianti susidomėjimo ne tik verslo analitikams, bet ir mokslininkams.

1.3.4. VP modeliavimo kalbos

VP modeliavimo kalbų ir būdų yra daug, todėl šiame skyriuje pateikiami tik populiariausi, dažnai praktikoje naudojami ir šio darbo kontekste aktualios VP modeliavimo kalbos (*BPMN*, *EPC*). Tačiau be paminėtų, egzistuoja ir kitos VP modeliavimo kalbos (Aguilar-Saven 2003), tokios kaip: *IDEF0* (funkcinio modeliavimo metodas), *IDEF3* (procesų aprašymo metodas), *BPEL* (VP vykdymo kalba), *Petri* tinklai, *UML* (angl. *Unified Modelling Language*).

1.3.4.1. BPMN kalba

BPMN (angl. *Business Process Modeling Notation*) – tai VP modeliavimo standartas, kurį parengė BPML. Vienas iš svarbiausių šio VP modeliavimo kalbos kūrimo tikslų – sukurti verslo modeliavimo kalbą taip, kad jis būtų lengvai suprantamas suinteresuotiems šalims (White 2004). Todėl ši VP modeliavimo kalba puikiai tinkama kaip priemonė susikalbant tarp įvairių verslo atstovų – vadovų, verslo analitikų ir programuotojų. Todėl ši kalba yra aukšto lygio kalba, tačiau yra galimybė panaudojant *BPMN* modelius transformuoti į *BPEL4WS* kalbą, kuri leidžia vykdyti VP. *BPEL* – VP vykdymo kalba (angl. *Business Process Execution Language*). Ši kalba yra žiniatinklio paslaugų WS-BPEL (angl. *Web Service Business Process Execution Language*) trumpinys.

1.3.4.2. EPC kalba

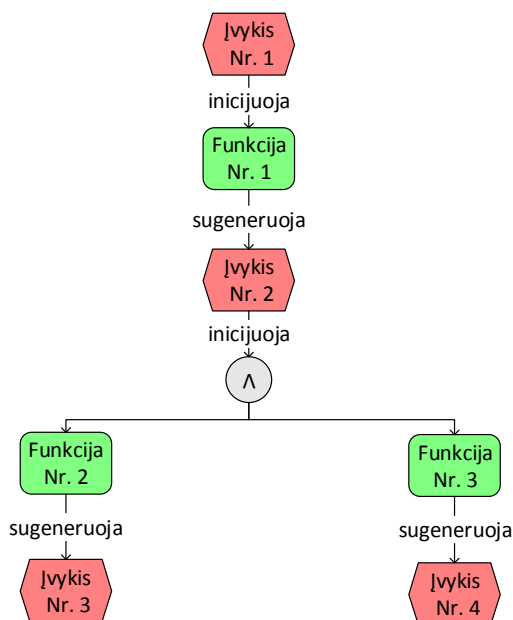
Įvykiais valdomų procesų grandinės (EPC – *Event-driven Process Chain*), tai struktūrinių diagramų tipas, naudojamas įvykiais valdomų VP modeliavimui (Nuttgens 1998). Šis modeliavimo būdas buvo sukurtas, projektuojant ARIS VP modeliavimo įrankį 1992 metais. Pagrindinė EPC idėja

yra tai, jog VP susideda iš funkcijų kurie įgyvendina VP veiklas. Veiklos yra vykdomos paeiliui ir kiekvieną veiklą inicijuoja veiklos inicijavimo įvykis, o veikla savo ruožtu gali sugeneruoti (iššaukti) kitus įvykius į kuriuos reaguoja kitos veiklos. Tokiu būdu, grindžiant įvykiais yra konstruojamas vykdomų veiklų grandinė, kuri įgyvendina VP. Toks modeliavimo principas yra panašus į anksčiau aprašytą ECA tipo taisyklėmis modeliuojamą VP, kadangi tiek ECA taisyklę, tiek EPC kalbos funkciją inicijuoja su jais susieti įvykiai, o funkcija savo ruožtu gali sugeneruoti kitus įvykius.

Įvykiais grindžiamų procesų modelyje yra galimi keletas įvykių ir veiklų sąveikos būdų, juose galioja jungimo operacijos „And“, „Or“, „Xor“, kurie atitinkamai matematiškai reiškia „ir“, „arba“, „atvirkštinis arba“ sąlygas. Tai reiškia, jog įvykiai procesams inicijuoti gali būti tarpusavyje jungiami, pavyzdžiui:

- Kai įvyksta įvykis Nr. 1 ir Nr. 2 (įvykių konjunkcija);
- Kai įvyksta įvykis Nr. 1 arba Nr. 2 (įvykių disjunkcija);
- Kai įvyksta bet kuris įvykis (atvirkštinis arba);
- Veikla gali sugeneruoti įvykį Nr. 1 arba įvykį Nr. 2;
- Veikla gali sugeneruoti įvykį Nr. 1 ir įvykį Nr. 2;
- Kiti sudėtiniai įvykių jungimo būdai, naudojant aukščiau paminėtus būdus.

3 pav. (Nuttgens 1998) pavaizduotas įvykiais valdomo proceso modelio pavyzdys, kai funkcijos Nr. 1 sugeneruotas įvykis Nr. 2 lygiagrečiai inicijuoja funkcijas Nr. 2 ir Nr. 3.



3 pav. Įvykiais valdomo verslo proceso modelio pavyzdys

1.3.5. VP modeliavimo požiūriai

Šaltiniai (Lu et al. 2007; Bauer 2009) pateikia du skirtingus požiūrius į VP modeliavimą – grindžiant grafiniais modeliais (angl. *graph-based*) ir grindžiant taisyklėmis (angl. *rule-based*).

Grindžiant grafiniais modeliais VP modeliavimas yra paremti grafų teorijos mokslu ir grafiniais modeliais. Modeliuojant VP grindžiant grafiniais modeliais yra naudojamos tokios modeliavimo kalbos kaip *Petri* tinklai, *FlowMake*, *YAWL* ir kitos. Prie grafiniais modeliais grindžiamų procesų modeliavimo požiūrio taip pat galima priskirti tokias anksčiau minėtas grafines notacijas modeliavimo kalbas kaip *BPMN* ar *EPC*. Tokiu būdu sukurtų grafinių VP modelių paskirtis yra suteikti galimybę aiškiai juos suprasti įvairiems verslo atstovams. Tačiau dauguma grafiniu būdu sukurti modeliai yra abstraktūs ir jie nėra tinkami vykdymui arba simuliacijai. Norint vykdyti grafinius modelius, juos reikia papildyti vykdymui reikalinga informacija arba transformuoti į vykdomąjį modelį. Grafiniai VP modeliai šio darbo kontekste yra naudojamas kaip priemonė pavaizduoti ECA tipo taisyklių rinkiniu užrašytą VP modelį.

Grindžiant taisyklėmis procesų modeliavime yra naudojamos ECA tipo taisyklės. Tai nėra grafinis modelių vaizdavimo būdas. Taisyklėmis grindžiami VP modeliavimas palyginti su grafiniais modeliais grindžiamu modeliavimu turi privalumą dėl to, jog formaliomis taisyklėmis užrašyti modeliai gali būti panaudojami tiek vykdymui, tiek simuliacijoje. Taip pat jie gali būti pakartotinai panaudojami modeliuojant kitus VP. Kad vykdyti arba simuliuoti tokias taisykles yra naudojama VT valdymo sistema (angl. BRMS – *Business Rule Management System*) arba taisyklių mašina (angl. *Rule Engine*). Tačiau taisyklėmis grindžiamų VP modeliai yra sunkiai suprantami, todėl juos ne visada gali suprasti verslo atstovai. Sukurti taisyklėmis grindžiamą modelį, kuris būtų tinkamas ir vykdymui, gali tik specialistai, turintys žinių kaip tos taisyklės turi būti užrašomos (kiekviena BRMS ar taisyklių mašina gali turėti savo taisyklių užrašymo kalbą).

1.4. Verslo procesų simuliacija

1.4.1. Simuliacijos apibrėžimas

Modelių vertinimo metodas – simuliacija (Maria 1997). Turint konkretų modelį, reikia nustatyti jo teisingumą ir efektyvumą. Tuo tikslu atliekami eksperimentai – modelių simuliacija. Prieš atliekant modelių simuliaciją, yra apibrėžiami modelių simuliacijos rezultatų vertinimo kriterijai, pavyzdžiui – kaina, laikas, klaidų skaičius ir kiti parametrai. Įvertinus modelio simuliacijos rezultatus, galima nustatyti modelio trūkumus ir jį patobulinus siekti geresnių simuliacijos rezultatų. Tačiau reikia turėti omenyje, kad beveik neįmanoma sukurti modelį ir jį simuliuoti taip, kad jis atitiktų realų pasaulį. Tokiu atveju reikėtų į modelį įtraukti visus realiame pasaulyje modelio procesus galinčius

lemti veiksmus, bet tai padaryti būtų labai sudėtinga. Modelių simuliacija, kuri yra artima realiam pasauliui, gali pakankamai tiksliai suteikti reikiamą informaciją.

Modeliuojant VP, siekiant ištirti VP efektyvumą ir pasirinkti geriausius sprendimus būtina ją iš visų pusių analizuoti. Todėl yra atliekama VP simuliacija, įvairūs pakeitimai modeliuose ir taisyklėse siekiant įvertinti jų pasekmes. Toks pakeitimų įvertinimo būdas yra žymiai pigesnis negu juos vertinant atliekant realius sprendimus. Todėl buvo sukurtos įvairios VP simuliacijos priemonės, kurios leidžia modeliuoti, atkartoti procesus ir padeda suvokti sprendimų poveikį.

Šaltinis (Tinsley et al.) pateikia nuolat didėjančio susidomėjimo simuliacija priežastis:

- Naudoti modelius ir jų atlikti jų simuliaciją yra pigiau ir saugiau, negu tai bandyti su galutiniais produktais arba jų prototipais. Kuriant į realų pasaulį panašius modelius ir jas simuliuojant, sutaupoma lėšų, nes iš anksto galima numatyti galimas problemas ir jas spręsti iš anksto, negu tai darant vėliau. Pavyzdžiui: vienas iš galingiausių pasaulio superkompiuterių, simuliuoja galimus uragano susidarymo vietas, jų maršrutus ir numatomą poveikį žmonėms bei žalai. Tokiu atveju remiantis simuliacijos rezultatais, galima nustatyti uragano poveikį ir sumažinti jo žalą iš anksto perspėjant gyventojus.
- Simuliacija gali būti realistiškesnė ir tikslesnė negu tradiciniai bandymai, nes jie leidžia pasirinkti laisvą konfigūraciją - aplinkos, vykdymo parametrai, netikėtos situacijos, kurias negalima nustatyti ar imituoti atliekant tradicinius bandymus.
- Simuliacija gali būti vykdoma žymiai greičiau, negu yra realus laikas. Todėl simuliuojant modelį, simuliaciją galima kartoti šimtus, tūkstančius ir milijonus kartų su skirtingais parametrais, norint surasti efektyviausią variantą. Realame pasaulyje bandymų skaičius yra ribotas.

1.4.2. Modeliavimo ir simuliacijos procesas

Modeliavimo ir simuliacijos procesas yra iteracinis, todėl šių procesų ciklas, priklausomai nuo siekiamų modelių ir jų simuliacijos rezultatų, procesai periodiškai kartojami kol pasiekiamas užsibrėžtas efektyvumas. Modeliavimo ir simuliacijos procesas (Maria 1997):

1. Problemos identifikavimas;
2. Problemos formulavimas;
3. Realios pasaulio duomenų ir procesų surinkimas;
4. Modeliavimo kalbos ir metodikos pasirinkimas;
5. Modelio formulavimas ir kūrimas;
6. Modelio logikos tikrinimas (ar jis atitinka realų pasaulį ir VP);

7. Modelio dokumentavimas ateities naudojimui;
8. Vertinimo kriterijų pasirinkimas;
9. Modelio simuliacija;
10. Simuliacijos rezultatų interpretavimas ir vertinimas;
11. Tolimesnis modelio tobulinimas (grįžtama į 1 punktą).

1.5. Dinaminiai verslo procesai

Analizuojant su dinaminių VP valdymu susijusią literatūrą, pastebėta, jog terminas „dinaminis verslo procesas“ yra pakankamai naujas terminas ir nėra dažnai vartojamas. Kalbant apie VP, kurie gali prisitaikyti prie konteksto, dažnai vartojami tokie apibūdinimai kaip: lankstus, prisitaikantis, kintantis, adaptyvus VP (Eijndhoven et al. 2008; Lu et al. 2007). Šie autoriai VP lankstumą vertina kaip įverčių matą, kurie nusako VP prisitaikymo prie konteksto įvertį trimis kriterijais – laikas (kaip greitai VP gali prisitaikyti), kaina (kiek tai kainuoja), paprastumas (kiek sudėtinga pritaikyti). Remiantis šiais kriterijais galima įvertinti, kurie VP yra daugiau lankstūs ir mažiau lankstūs ir jais remiantis didinti VP lankstumą gerinant paminėtus kriterijus.

Šaltinis (Zeng et al. 2002) nagrinėja pilnai automatizuojamus VP, kuriuos automatizuojant programų sistemose yra sudaromos vykdomų darbų seka (angl. *workflow*) ir iškelė dinaminių darbų sekos sudarymo poreikį, kai apibrėžta darbų seka yra keičiama darbų sekos vykdymo metu. Tokio poreikio priežastys yra tai, kad keičiantis verslo poreikiams, programų sistemose taip pat reikia keisti vykdomų darbų eigos seką. Autorius siūlo dinamiškai generuoti darbų sekos modelį naudojant taisykles. Šiame šaltinyje autoriai išskiria VP į du tipus: statines ir dinamines. Statiniai VP yra iš anksto apibrėžti procesai ir jie vykdomo eigoje nėra keičiami. Dinaminiai VP – tai procesai, kurie gali būti keičiami vykdymo metu ir jo vykdymo seka yra konstruojama dinamiškai. Dinaminių VP privalumai prieš statinius:

- Automatizuojant VP nebūtina numatyti iš anksto ir žinoti visų įmanomų variantų, kuriuos turi įgyvendinti VP ir pakanka koncentruotis į VP tikslą. Paaiškėjus naujiems poreikiams ar pasikeitus reikalavimams, VP gali būti keičiamas vykdymo metu.
- Pakeitus vykdomus VP, juos nereikia paleisti iš naujo, todėl taupomas laikas.
- Dinaminių VP pritaikymas prie poreikių yra greitesnis.

Terminas „dinaminis verslo procesų valdymas“ yra jau prigijęs ir dažnai vartojamas. Šiuo atveju žodis dinaminis reiškia nuolat kintančių VP valdymą ir jų pritaikymą prie konteksto (detaliau žr. 1.5.1). Dinaminių VP požiūriu, dinaminis procesas yra tas, kuris yra konstruojamas dinamiškai ir jį galima keisti vykdymo metu. Apibendrinant, VP pritaikymą prie konteksto galima apibrėžti dviem požiūriais:

- Tai lankstus procesas, kuriame viskas yra numatyta iš anksto ir jo vykdymas kinta priklausomai nuo konteksto (pavyzdžiui: nuo aplinkos temperatūros, palūkanų normos). Toks procesas vykdymo metu geba automatiškai prisitaikyti pagal esamą kontekstą ir pats nusprendžia kaip geriausia keisti vykdymo eigą, kad būtų pasiektas optimalus rezultatas. Pavyzdžiui – konkurentams pasiūlius prekės nuolaidą, proceso vykdymas nukreipiamas kita, iš anksto apibrėžta šaka, kuri klientams taip pat pasiūlo nuolaidą. Tokiu būdu prisitaikius prie konteksto yra išvengiamas klientų sumažėjimas dėl konkurentų veiksmų.
- Tai dinaminis procesas, kurį vykdymo metu keičia išoriniai veiksniai. Tokį procesą gali keisti žmogus arba kita automatizuota sistema (taip pat gali būti ir kitas procesas). Proceso dinamiškumas gali būti tiek būsenos keitimas jos vykdymo metu, tiek jos struktūros keitimas. Proceso keitimas vykdymo metu gali pasireikšti tuo, kad joje gali būti įterpiamos naujos veiklos, būti panaikinti, pasikeisti veiklos prasmė arba išsišakojimų sąlygos. Toks procesas turi pranašumą prieš anksčiau minėtą lankstų procesą, kadangi ne visada esamuose VP viskas gali būti apibrėžiama iš anksto. Galimybė vykdymo metu keisti VP ypač reikalingi nenumatytiems konteksto pokyčiams, pavyzdžiui – pasikeitus konkurentų pasiūlymams, pasikeitus įstatymams ir pan., nes jau vykdomų VP nutraukimas kainuoja brangiai ir žymiai efektyviau yra vykdomą VP pritaikyti prie konteksto.

1.5.1. Dinaminis VP valdymas

Dinaminis VP valdymas (angl. *dynamic Business Process Management, dynamic BPM*) yra sprendimai, kurie leidžia greitai reaguoti į nuolat kintančias verslo sąlygas ir patenkinti individualius poreikius verslui arba klientams laiku (Martin et al. 2009). Kintančios verslo sąlygos gali būti skirstomos į vidines (kurios kinta organizacijos viduje) ir išorines (kurios kinta už organizacijos ribų). Trumpai apibūdžiant, dinaminis VP valdymas - tai valdymo metodas, skirtas tam, kad VP greitai prisitaikytų prie kintančių verslo poreikių, konteksto arba būtų greitai pritaikytas. Tokių VP valdyme yra projektuojami ir naudojami tokie procesai, kurie gali būti lengvai pakeičiami vienas kitu ir būti pritaikomi prie aplinkos sąlygų sparčiai. Dinaminis VP valdymas yra naudojamas organizacijose, kuriose reikia išlaikyti ir padidinti VP efektyvumą sparčiai kintančioje, chaotiškoje verslo aplinkoje (pavyzdžiui – akcijų birža, spekuliacijos akcijomis, valiutų rinka, didelė rinkos konkurencija). Kaip vienas iš pavyzdžių dinaminis VP gali būti individualūs mokėjimų plano pasiūlymai mobiliojo ryšio klientams pagal jų poreikius ir galimybes, greitas kainų politikos keitimas ir prisitaikymas konkurentams pasiūlius geresnį mokėjimo planą (Martin et al. 2009), labai didelio greičio prekyboje akcijomis (angl. *High-frequency trading*). Šaltinis (Dynamic BPM: From simple to sophisticated 2011) pateikia du dinaminių VP valdymo lygius:

- Žemo lygio – tai dinaminių VP valdymas, kai yra nustatyta riba tarp automatinio lankstumo, kai procesas prisitaiko prie konteksto ir rankinio – kai procesą koreguoja žmogus.
- Aukšto lygio – tai procesai, kurie turi dinamines konfigūracijos galimybes, taip pat galimybes procesuose pakeisti proceso žingsnių tvarką arba pereiti prie alternatyvaus kelio.

Taip pat labai svarbu, kad dinaminiam VP valdyme gali dalyvauti ir kiti procesai, kurie gali lemti ir valdyti pagrindinius procesus ir juos keisti. Ši sritis yra ypač aktuali nagrinėjant tokias sritis kaip: prognozavimo modeliai, simuliacijos modeliai, automatinė aplinkos analizė ir stebėjimas. Tai yra palyginus nauja ir mažai ištirtinėta sritis, kadangi šiuo metu vis mažai arba visai nėra priemonių, kurie leistų pilną apimtį simuliuoti dinamines verslo sistemas ir VP. Be to, taip pat šiuo metu nėra išbaigtų įrankių ir priemonių, kurias integruojant į informacines sistemas leistų įgyvendinti, įdiegti ir palaikyti pilnai dinamines ir lanksčias informacines sistemas.

1.6. Apibendrinimas

Atlikus literatūros ir dalykinės srities sąvokų analizę, pastebėta, kad dinaminių VP ir jų valdymo tematika yra nauja ir nemažai susidomėjimo kelianti mokslinių tyrimų kryptis. Modeliuojant VP, siekiant įvertinti jo efektyvumą yra atliekama jų simuliacija ir cikliška kartojant modeliavimo-simuliacijos-rezultatų analizės procesą yra sukuriamas efektyvus VP modelis, kuris yra vykdomas informacinėje sistemoje. Vykdamas procesus ir pasikeitus kontekstui yra reikalinga prie jo pritaikyti VP. Procesai, kurie keičiami procesų vykdymo metu yra dinaminiai. Kad įvertinti dinaminį VP, reikia turėti galimybę atlikti dinaminių VP simuliaciją, kai simuliacijos metu, vykdamas procesą, procesas yra keičiamas ir taip išbandyti įvairius procesų keitimo variantus bei atsakyti į klausimą kokią poveikį organizacijoje turės vykdomų procesų pakeitimai ir išrinkti geresnį pakeitimų variantą.

Išanalizavus modeliavimo būdus, dinaminių VP modeliavimui ir simuliacijai pasirinktas ECA tipo taisyklėmis grindžiamas VP modeliavimo būdas ir šio darbo kontekste toliau nagrinėjami tik taisyklėmis grindžiami VP. VT gali būti transformuojamos į ECA tipo taisyklėmis grindžiamą VP modelį ir tokiu būdu taisyklėmis konstruojamas VP modelis. Toks būdas pasirinktas todėl, kad VT iš prigimties yra dinamiškos, nes jos priklauso nuo konteksto ir jas yra lengva pakartotinai panaudoti ir pritaikyti prie dažnai besikeičiančio konteksto. Taip pat ECA tipo taisyklės naudoti yra naudinga, nes jos gali būti panaudojamos vykdymui arba atliekant simuliaciją.

2. VERSLO TAISYKLIŲ VALDYMO SISTEMŲ IR SIMULIACIJOS ĮRANKIŲ ANALIZĖ

Taisyklėmis grindžiamų VP vykdyme ir jų simuliacijoje pagrindinis komponentas yra VT valdymo sistema (angl. *BRMS – Business Rule Management System*). Šiame skyriuje analizuojami BRMS architektūra ir jos veikimo principai, BRMS priemonių dinaminį VP simuliacijai taikymo galimybės bei taisyklėmis grindžiamų VP simuliacijos priemonių galimybės simuliuoti dinaminį VP. Gauti analizės rezultatai leis išsiaiškinti kokios yra problemos ir galimybės realizuoti dinaminį VP simuliacijos priemonę panaudojant esamus komponentus.

2.1. Taisyklėmis grindžiamų VP simuliacijos įrankių analizė

Išnagrinėjus šiuo metu rinkoje esančių VP modeliavimo įrankių rinką, pastebėta, jog tokių įrankių pasiūla yra didelė. Remiantis šio darbo tematika, aktualūs yra tie įrankiai, kurie leidžia simuliuoti VP grindžiant taisyklėmis. Todėl susiaurinus šių įrankių spektrą pagal poreikius, išskiriami 4 didesni, daugiau funkcionalumo turintys įrankiai (*IBM Websphere Business Modeler Advanced*, *Simprocess*, *Simul8* ir *AcuProcess Modeler*), kurių galimybės analizuojamos detaliau.

2.1.1. IBM Websphere Business Modeler Advanced

IBM korporacija verslo analitikams siūlo dvi komercines VP modeliavimo įrankių versijas: *IBM Websphere Business Modeler Basic* ir *IBM Websphere Business Modeler Advanced*. Pirmoji versija siūlo sumažintą funkcionalumo ir galimybių rinkinį ir tokiu būdu sumažinta produkto kaina. Antroji *Advanced* įrankio versija pateikia išplėstą prieš tai pateikto *Basic* įrankio versijos funkcionalumą ir galimybes (*WebShere Business Modeler v.7.0 2014*). Ši PĮ leidžia verslo analitikams modeliuoti ir analizuoti VP standartais pagrįstoje vartotojo sąsajos aplinkoje. Šio įrankio savybės:

- Galimybė verslo analitikams projektuoti, modeliuoti ir dokumentuoti VP, siekti nuolatinio jų tobulinimo.
- Integruota IBM Websphere KPI biblioteka, kuri supaprastina VP modeliavimą. KPI biblioteka suteikia galimybę vertinti VP panaudojant daugiau nei 800 skirtingų indikatorių (KPI – angl. *Key Performance Index*).
- Galimybė verslo analitikams modeliuojant dirbti kartu ir prie vieno modelio ir dalintis rezultatais.
- Palaiko BPMN modeliavimo kalbą, simuliacijos ataskaitų generavimą.
- Integracija su kitais IBM BPM (angl. *Business Process Management*) įrankiais, vieninga vartotojo sąsaja.

- Integracija su Microsoft™ Visio programa, galimybė eksportuoti modelius į kitus dokumentų formatus. Taip pat yra galimybė naudoti modelių saugyklą.
- Tam tikru lygmeniu palaikomas dinamiškumas – galima pasirinkti procesų vykdymo šaką vykdymo metu.

Ši priemonė tam tikru lygmeniu suteikia galimybę vykdyti simuliaciją dinamiškai – galima pasirinkti procesų vykdymo šaką vykdymo metu, tačiau galimybės pilnai simuliuoti dinامينius VP nėra.

2.1.2. Simprocess

Simprocess – modeliavimo priemonė, kuri leidžia kombinuoti VP žemėlapių sudarymą, įvairių VP dalių simuliaciją. Šio įrankio gamintojai teigia, jog „priemonė leidžia simuliuoti VP esant komplikotai verslo logikai“ (SimProcess User’s Manual 2012). Taip pat ji leidžia analizuoti įvairias „kas-jeigu“ verslo scenarijus. Priemonės galimybės:

- Hierarchinis procesų žemėlapių sudarymas;
- Įvykiais valdoma simuliacija;
- Į objektinį programavimą orientuotas modeliavimas;
- Procesų vykdymo animacija;
- Microsoft Access PĮ paremta ataskaitų ir grafikų sudarymas;
- VP vertinimas KPI kriterijais.

Simprocess neturi galimybės simuliuoti dinامينius VP.

2.1.3. Simul8

Simul8 modeliavimo ir simuliacijos priemonė pateikiama keliomis skirtingomis versijomis: *Simul8 Education*, *Simul8 Basic*, *Simul8 Professional*, *Simul8 Team*. Skirtingos versijos suteikia skirtingą funkcionalumą – nuo *basic* versijos VP simuliacijos ir modeliavimo galimybės, toliau sekant *professional* versijai – simuliacijos ataskaitų, VP vertinimas remiantis KPI vertinimo sistema, iki *team* versijos, kuri leidžia dirbti prie kelių modelių keliems analitikams nepriklausomai nuo buvimo vietos (Simul8 Help center 2014).

Žiūrint iš dinامينių VP simuliacijos panaudojimo perspektyvos, ši priemonė leidžia apibrėžti dinامينius parametrus, kurie vykdymo gali kisti pagal apibrėžtas taisykles (pavyzdžiui, simuliuojant prekybos procesą, galima apibrėžti parametą „klientų skaičius“, kuris vykdant simuliaciją didėja kiekvieną dieną tam tikru apibrėžtu dydžiu. Tačiau tai nėra dinامينio verslo proceso simuliacija, todėl šis įrankis yra netinkamas tolimesnei šio darbo kontekste nagrinėjamai temai.

2.1.4. AccuProcess Modeler

AccuProcess Modeler yra grafinė, pakankamai nesudėtinga VP modeliavimo priemonė, kuri padeda verslo žmonėms projektuoti, dokumentuoti, simuliuoti ir tobulinti VP. Kaip teigia šio įrankio gamintojai, AccuProcess yra „modeliavimo PĮ suprojektuota kaip galima paprasčiau ir suprantamiau verslo analitikams ir verslo žmonėms naudoti“ ir „AccuProcess paprasta naudoti kaip Microsoft™ Word ar PowerPoint“ (AccuProcess Business Process Modeler 2014). AccuProcess Modeler priemonė, palyginti su kitais, pasižymi galimybėmis:

- Verslo procesų žemėlapių sudarymas;
- Automatinis modelių dokumentacijų generavimas (angl. *self-documentation*).

AccuProcess Modeler neturi galimybės simuliuoti dinامينius VP.

2.1.5. Palyginimas

Vertinant rinkoje šiuo metu esančius VP ir jų simuliacijos įrankius, remiamasi autoriaus (Grašys 2011) išanalizuotais ir pateikiamais išplėtais vertinimo kriterijais. Jo siūlomas įrankių vertinimo metodas yra suskirstytas į kategorijas pagal PĮ galimybes (aparatinės ir programinės įrangos savybės, modeliavimo galimybės, imitavimo galimybės, įvesties ir išvesties galimybės, darbas su VT galimybės). Kadangi nagrinėjame PĮ, kuri susijusi su taisyklės grindžiamų VP simuliacija, mums daugiausiai aktualus yra paskutinė autoriaus siūloma vertinimo kategorija, susijusi su VT, todėl įrankių vertinime remiamasi šios kategorijos vertinimo kriterijais, kurie yra suskirstyti į 3 dalis pagal vertinimo aspektus (Grafiniai aspektai, taisyklių struktūros ir versijų valdymo aspektai, taisyklių kokybės aspektai) pridedant dar vieną papildomą vertinimo aspektą – dinaminės simuliacijos galimybės.

Atliktas PĮ vertinimas pagal pasirinktus vertinimo kriterijus, pateikiamas lentelėje. Vertinamas atliekamas 3 vertinimo lygiais:

- (+) – PĮ pagal kriterijų pilnai palaiko vertinimo kriterijaus funkcionalumą.
- (+/-) – PĮ pagal kriterijų funkcionalumą palaiko iš dalies.
- (-) – PĮ pagal kriterijų funkcionalumą nepalaiko.

3 lentelė. Taisyklėmis grindžiamų verslo procesų simuliacijos įrankių palyginimas

	Vertinimo kriterijus	IBM Websphere	Simprocess	Simul8	Acuprocess
Grafiniai aspektai	Verslo taisyklių integravimas į verslo procesų modelį	+	+	+	+
	Verslo taisyklių įtraukimas į proceso modelį naudojant sprendimų lenteles	+	+	+	-
	Modelio generavimas pagal verslo taisyklių išraiškas	-	-	-	-
	Pakartotinis taisyklių ir jų šablonų panaudojimas	+	+	-	-
	Taisyklių ir šablonų panaudojimas skirtinguose projektuose	+	-	-	-
	Taisyklių modeliavimo politika	+	+	-	-
	Trigeriais iškviečiamos taisyklės	+	+	+	-
Taisyklių struktūros ir versijos	Hierarchinė taisyklių struktūra:	-	-	-	-
	Taisyklių versijos	+	+	+	-
	Padinių modelio duomenų keitimas taisyklėmis imitavimo metu	+	+	+	+
Taisyklių kokybės aspektai	Klaidų taisyklėse tikrinimas	+	+	+	+
	Verslo taisyklių palyginimas	+	-	-	-
	Automatinis grafinės dokumentacijos generavimas	-	-	-	-
	Vykdomų/vykdytų taisyklių panaudojimo analizė	+	-	-	-
Dinaminės simuliacijos galimybės	Dinaminių verslo procesų simuliacija	+/-	-	+/-	-
	Taisyklių keitimas sustabdžius verslo procesų simuliaciją	-	-	-	-

2.1.6. Apibendrinimas

Apibendrinant taisyklėmis grindžiamų VP simuliacijos įrankių analizę, galima teigti kad nors tokių įrankių pasirinkimas yra nedidelis, bet funkcionalumo atžvilgiu jie suteikia galimybę modeliuoti, analizuoti ir simuliuoti VP pakankamai plačiu spektru. Tačiau dinaminių VP simuliacija yra vis dar nauja sritis ir pateiktuose įrankiuose kol kas tik iš dalies galima atlikti dinaminių VP tyrimą – pavyzdžiui, nukreipiant jos vykdymą kita šaka ir pan. Nei vienas iš nagrinėtų įrankių nesuteikia galimybės vykdymo metu sustabdyti VP vykdymą, į jį pridėti arba išimti veiklas, keisti taisykles iš esmės, tokiu būdu ištiriant pasikeitusių taisyklių poveikį. Todėl įvertinus įrankių galimybes, nuspręsta šio darbo kontekste jas toliau nenaudoti.

2.2. BRMS architektūros analizė

2.2.1. BRMS apibrėžimas

BRMS – verslo taisyklių valdymo sistema. Literatūros analizės metu buvo nustatyta, jog BRMS yra priemonė atskirti verslo taisyklės nuo programos logikos ir ši priemonė yra naudojama taisyklių valdymui bei vykdymui. Viena iš didžiausių pasaulyje PĮ kūrėjų – IBM korporacija, BRMS naudojimą apibrėžia kaip būdą valdyti organizacijos verslo politiką – ją apibrėžti, valdyti ir palaikyti atskiriant VT nuo PS kodo (What is Business Rules Management? 2014). Tokiu būdu, atskiriant VT nuo PS ir panaudojant BRMS įrankius jiems valdyti, BRMS leidžia verslo ekspertams apibrėžti ir palaikyti sprendimus, kurie gali lemti tiek VP, tiek PS elgesį ir taip pagreitinti VP bei PS pritaikymą prie naujų reikalavimų.

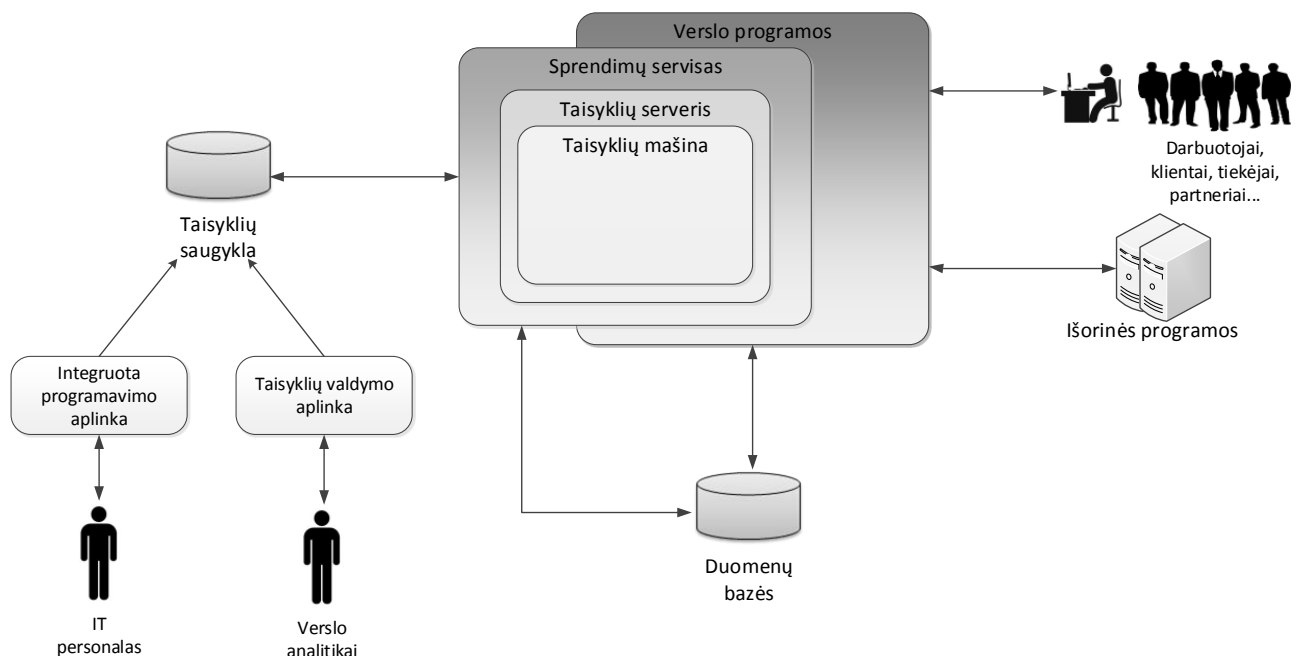
Svarbu paminėti, jog BRMS nėra viena priemonė arba komponentas, tai yra komponentų ir priemonių rinkinys – sistema, susidedanti iš mažiausiai tokių (Ritchie et al. 2011) dalių:

1. Verslo ekspertų įrankiai. Tai įrankiai skirti darbui su VT valdymo sistema, kurie leidžia įvesti naujas, keisti, stebėti, kontroliuoti taisykles.
2. Taisyklių bazė. Taisyklių bazėje saugomi visos taisyklės, kurios yra atskirtos nuo PS kodo. Taisyklių bazės vaidmenį gali atlikti įprastos duomenų bazės.
3. Taisyklių vykdymo ir kontrolės komponentas (taisyklių mašina). Tai komponentas, kuris priima PS užklausas, jas apdoroja ir vykdo logiką, kuri aprašyta taisyklėmis taisyklių bazėje.

2.2.2. BRMS architektūra

Išanalizavus įvairių BRMS gamintojų architektūrą, pastebėta, jog nors kiekvienas gamintojas jos architektūrą pateikia skirtingai, tačiau pagrindiniai komponentai yra tie patys ir pagrindinės idėjos yra panašios. Straipsnis (Duda 2010) pateikia BRMS architektūrą (4 pav.). Šios architektūros idėja

yra ta, kad PĮ logika yra atskirta nuo taisyklių logikos ir sprendimams priimti yra naudojama sprendimų žiniatinklio paslauga (angl. *web-service*), per kurią PĮ į BRMS siunčia ir gauna užklausas. Sprendimų žiniatinklio paslauga suteikia prieigą prie taisyklių serverio, kurioje veikia taisyklių mašina. Taisyklių mašina naudoja taisyklių saugykloje esančias taisykles, kuriais remiantis gali duoti atsakymą į gautas užklausas per sprendimų žiniatinklio paslaugą. Taisyklių saugykloje esančias taisykles valdo verslo analitikai panaudojant taisyklių valdymo aplinką – tai verslo analitiko darbo priemonės su taisyklių saugykla. IT personalas – programuotojai, naudojant jiems skirtus įrankius – integruotą programavimo aplinką, kurdami PĮ jie atskiria nuo programos logikos taisykles, jas rašo formalia taisyklių kalba ir talpina į taisyklių saugyklą. Taip pat yra svarbu tai, jog tiek taisyklių serveris, tiek taikomosios, verslo programos gali pasiekti tą pačią duomenų bazę (toliau – DB). Tai reiškia, jog tiek PĮ gali naudoti ir keisti DB, tiek taisyklėse aprašytuose veiksmuose galima imti esančius DB duomenis arba juos keisti.



4 pav. BRMS architektūra

Autorius (Graham 2007) teigia, jog žinių bazė (taisyklės) gali susidėti iš trijų žinių tipų:

1. Žinios apie objektus - tai žinios apie tai, kokia forma gali būti saugomi duomenys ir objektai (pavyzdžiui, duomenų bazės modelis yra duomenų modelio žinios). Žinios apie objektus gali saugomos XML Schema, semantiniiais tinklais arba duomenų modeliu;
2. Procedūros – tai žinios, kurios nusako veiksmą, t. y. ką reikia daryti. Procedūros gali būti pateikiamos įvairia forma, pavyzdžiui, užrašant JAVA programavimo kalbos kodu, Excel formatu, išreiškiant formalia verslo kalba, taisyklėmis;
3. Tarpusavio ryšių žinios – tai žinios apie tai kaip procedūros gali būti siejamos tarpusavyje.

Taip pat, autorius (Graham 2007) pateikia dalis ir komponentus iš kurių susideda BRMS pagrindas:

Taisyklės. Tai formalizuoti taisyklių sakiniai, kurie dažniausiai išreiškiami forma *IF X THEN Y*, t. y. jei X tenkinama, vadinasi vykdome Y. Vienuose BRMS, taisyklės gali būti aukšto lygio, kitose – žemo. Aukšto lygio taisyklės, tai taisyklės, kurios užrašomos pakankamai natūralia, suprantama kalba, pavyzdžiui:

IF Klientas yra lojalus ir kliento išlaidos yra didesnės nei 1000 litų
THEN Pritaikyti 10% nuolaidą.

Ta pati taisyklė užrašyta žemo lygio taisyklių kalba gali atrodyti taip:

IF Client.Status == "Lojalus" && Client.Value > 1000
THEN Client.Total = Client.Value – (Client.Value * 0.1)

Taisyklių šablonai. Tai taisyklių šablonai, kurie leidžia sukurti taisykles ir jas vėliau užpildyti. Taisyklių šablonai leidžia sukurti iš dalies apibrėžtas taisykles, kuriuose yra rezervuotos vietos, kuriuose reikia įrašyti trūkstamus duomenis (pvz. – skaičius). Taisyklių šablonai supaprastina taisyklių kūrimą verslo analitikams, kadangi dėl šablonų jiems nereikia žinoti visos taisyklių užrašymo formalumo.

Taisyklių sintaksės tinkamumo vertintojas. Sintaksės tinkamumo vertintojas turi gebėti patikrinti užrašytų taisyklių atitikimą taisyklių užrašymo kalbai, jeigu įmanoma – parodyti klaidas, kurios buvo padarytos užrašant taisyklę. BRMS neturi leisti į taisyklių saugyklą įvesti sintaksiškai neteisingų taisyklių.

Procedūros ir algoritmai. Nebūtinai BRMS komponentas, tačiau kai kurie jų gamintojai tai pateikia. Tai komponentas, kuris leidžia sudaryti procedūras, kurias kviečiant su tam tikrais parametrais yra gaunamas reikalingas atsakymas. Procedūros realizacijos lygmenyje yra aprašytas taisyklių rinkinys ir jo vykdymo seka, kurias vykdant yra gaunamas reikiamas rezultatas.

Taisyklių vykdymo eigos nustatymas. Nebūtinai BRMS komponentas, tačiau kai kurie jų gamintojai leidžia sudaryti, pavyzdžiui, užduočių vykdymo eigą naudojant taisykles.

Išvadų generatorius. Išvadų generatorius, kitaip gali būti vadinamas išvedima mašina (angl. *inference engine*) turi naudojant taisykles surasti sprendimą ir išvadą (atsakymą). Išvados gauti yra naudojami išvadų gavimo algoritmai ir metodai (plačiau - 2.2.3 skyriuje). BRMS kontekste išvadų generatorius yra vienas iš svarbiausių komponentų.

Taip pat yra svarbu, jog BRMS turėtų užtikrinti svarbią savybę – taisyklių neprieštarumą. Taisyklės, esančios taisyklių saugykloje neturi prieštarauti viena kitai, jose neturi būti begalinių ciklų. Tačiau tai yra sudėtingas uždavinys ir ši problema yra nagrinėjama ne vienerius metus, todėl šio darbo kontekste tai yra nenagrinėjama.

2.2.3. Išvadų gavimo metodai

Kadangi BRMS pagrindinė paskirtis – gauti išvadas (atsakymus) į pateiktas užklausas. Todėl išvadoms gauti yra naudojami išvadų gavimo metodai, kurie yra realizuojami pagrindiniame BRMS komponente – taisyklių mašinoje. Išvadoms gauti yra svarbu pritaikyti turimus duomenis ir iš jų gauti žinias. Išvadų gavimo metodų, kurie gali būti naujais taisyklių mašinose yra keletas: tiesioginio išvedimo (angl. *forward-chaining*), Rete tinklas, Fuzzy logika (Smaizys et al. 2009), tačiau populiariausi ir šiuo metu naudojami yra tiesioginio išvedimo ir Rete tinklo sudarymo metodai. Taisyklių mašinoje gali būti naudojamas ir atbulinio (angl. *backward-chaining*) algoritmas, kurį tikslinga naudoti taisyklių derinimui ir vykdymo sekos atsekamumui (Graham 2007).

2.2.3.1. Tiesioginio išvedimo algoritmas

Tiesioginio išvedimo algoritmas remiasi jau turimais duomenimis ir taikant taisyklių bazėje esančias taisykles, išveda naujus faktus (duomenis). Tiesioginio išvedimo procesas gali būti tęsiamas iki tol, kol yra pasiekiamas tikslas (tikslas yra reikiamas faktas), kol visos taisyklės patikrintos ir nebeįmanoma gauti daugiau faktų. (Kuter et al 2004; Smaizys et al. 2009; Graham 2007; Čyras 2008). Tiesioginio išvedimo algoritmas taip pat yra naudojamas ekspertinėse sistemose, dirbtinio intelekto sistemose. Kad geriau suprasti tiesioginio išvedimo algoritmo principą, pateikiamas tiesioginio išvedimo algoritmo pavyzdys su automobilio gamintojo pasirinkimu taisyklių rinkiniu:

Taisyklės:

1. *JEI* pinigų suma mažesnė už 50 000, *TAI* ekonominė klasė
2. *JEI* pinigų suma yra tarp 50 000 ir 100 000, *TAI* vidutinė klasė
3. *JEI* pinigų suma yra didesnė nei 100 000, *TAI* aukšta klasė
4. *JEI* ekonominė klasė, *TAI* opel
5. *JEI* vidutinė klasė, *TAI* volkswagen
6. *JEI* aukšta klasė, *TAI* mercedes-benz

Tikslas: automobilių gamintojas

Išvedimas: Tarkime turime pradinį faktą – pinigų sumą 65 000. Šį faktą atitinka 2 taisyklė ir pritaikius ją, faktų bazėje turime naują faktą – vidutinė klasė. Pritaikius 5 taisyklę pasiekiame tikslą

– tinkami automobilio gamintojai yra *volkswagen*. Šio išvedimo kelias yra 2, 5 – tai taisyklių seka, kurios buvo taikomos siekiant tikslo.

Tiesioginis išvedimas gali būti tęsiamas iki tol kol tikslas yra pasiekiamas arba iki tol, kol nebėra taisyklių, kurias galima taikyti. Todėl žiūrint iš taisyklėmis grindžiamų VP vykdymo požiūrio, VP gali būti konstruojamas kaip taisyklių seka, kuri buvo gauta siekiant tikslo naudojant tiesioginio išvedimo algoritmą.

2.2.3.2. Rete algoritmas

Rete algoritmas yra efektyvus šablonų atitikimo algoritmas, naudojamas taisyklių mašinose sprendimams ir išvadoms gauti (Wright 2003; Graham 2007). Šis algoritmas remiasi standartine medžio struktūra ir yra taikomas kai taisyklių ir faktų bazė labai išsiplečia. Iš taisyklių faktų bei teiginių yra sukuriamas medžio egzempliorius kuriame faktai išdėstomi ir sujungiami taip, kad pagal taisyklių prielaidų šablonus, būtų vykdoma kuo efektyvesnė peržiūra/paieška, t. y. medis atsimena surastus faktų derinius ir taip sutaupo paieškos laiko (kai tų derinių dažnai ieškoma). Atsiradus naujam faktui medis minimaliai pakoreguojamas. Pagrindinis šio algoritmo trūkumas – didelės atminties sąnaudos. Todėl taikant šį algoritmą ir turint sukauptą didelę taisyklių bei faktų bazę tikėtinos atminties trūkumo problemos. Laikui bėgant, buvo sukurta keletas *Rete* algoritmo patobulinimų (*Rete**), kurie pasiekė dar geresnių efektyvumo rezultatų. Šis algoritmas plačiai taikomas ir naudojamas tokiose taisyklių mašinų bei ekspertinių sistemų realizacijose kaip *Tibco Business Events*, *Drools*, *JRules*, *Blaze Advisor*, *BizTalk Rules Engine* ir *Soar*.

2.2.4. Apibendrinimas

Apibendrinus, BRMS pagrindas ir branduolys yra taisyklių mašina, kuris geba vykdyti taisykles ir gauti išvadas. Išvadų gavimui yra naudojami išvadų gavimo metodai ir taisyklių mašinų realizacijai yra naudojami du populiariausi algoritmai – tiesioginio išvedimo ir *Rete*. Išvedimo rezultatas yra išvada ir taisyklių seka, kuri buvo taikoma išvadai gauti.

Todėl išsiaiškinus, jog taisyklių vykdymui yra naudojamos BRMS ir norint toliau nagrinėti taisyklėmis grindžiamų dinaminių VP simuliaciją, reikia išsiaiškinti kokios yra rinkoje esančių BRMS galimybės.

2.3. BRMS analizė

Šiuo metu rinkoje egzistuoja keletas BRMS. Vieni iš jų yra laisvai prieinami naudojimui arba jie leidžia išbandyti jų demonstracinę versiją (*Drools*, *OpenL Tables*, *IBM ILog JRules*), kiti yra komerciniai, kurie prieinami tik komercinėms organizacijoms (*InRule*, *Versata BRMS*, *HaleyRules*,

FICO® Blaze Advisor® Business Rules Management System). Todėl šiame skyriuje pateikiama tik prieinamų naudojimui BRMS analizė, kurios tikslas yra išsiaiškinti BRMS galimybes, taisyklių užrašymo kalbą bei galimybes juos panaudoti taisyklėmis grindžiamų dinaminių VP simuliacijos tyrimui.

2.3.1. Drools

Drools – atviro kodo BRMS, kurio kūrimo projektas buvo pradėtas 2001 metais. Drools rinkinys susideda iš tokių komponentų, kaip:

- Drools Guvnor – taisyklių valdymas;
- Drools Expert – taisyklių mašina;
- jBPM 6 – VP modeliavimas grindžiant taisyklėmis ir modelių vykdymas. Palaiko BPMN 2 notaciją;
- Drools Fusion – įvykių valdymas.

Drools taisyklių mašinos darbas paremtas tiesioginio išvedimo arba *Rete*. Drools skirta naudoti su *Java* kalba kuriama PĮ. Taisyklės šioje BRMS gali būti užrašomos tiek aukšto lygio kalba (panaudojant Drools Guvnor vartotojo sąsają, taisyklių šablonus arba grafinį taisyklių kūrimo įrankį), tiek žemo lygio (5 pav.) kalba. Žemo lygio kalba užrašytos taisyklės iš dalies paremta *Java* programavimo kalbos kodu (Drools Documentation 2014).

```
rule "Calculate discount"
when
  p: ShoppingCart( Sum > 1000 )
then
  p.Discount = p.Sum * 0.1;
  p.Total = p.Sum - p.Discount;

  System.out.println(p.UserName);
  System.out.println(p.Total);
end
```

5 pav. Drools žemo lygio taisyklės pavyzdys

Pagal pateiktą pavyzdį, Drools taisyklės struktūra yra tokio tipo:

```
rule "pavadinimas"
when
  salyga
then
  veiksmas
end
```

Įdomu yra tai, jog dėl plataus Drools galimybių spektro, ją galima nesunkiai panaudoti realizuojant PĮ atskirti reikšmingą programos kodą arba jos logiką į Drools taisykles ir centralizuotai jas valdant sumažinti PĮ palaikymo kaštus ir suteikti PĮ daugiau lankstumo. Taip pat yra įdomu tai,

jog Drools taisyklės veiksmuose gali būti naudojamas *Java* programavimo kalbos kodas, todėl jame gali būti sudėta dalis programos logikos. Veiksme aprašytas kodas gali palaikyti abipusį ryšį su ja naudojančia PĮ, todėl veiksmuose galima naudoti pasiekiamus realizuotoje PĮ esančias klases, objektų egzempliorius ir vykdyti jų metodus (pavyzdžiui, - atidaryti reikiamą vartotojui duomenų įvedimo formą). Taip pat Drools išplečia taikymo galimybes, suteikiant galimybę tiek ją naudojančiai PĮ, tiek taisyklės veiksmuose kreiptis į Drools taisyklių mašinos darbinę atmintį, į ją patalpinti, išimti arba keisti faktus ir jų atributus. Todėl veiksmuose galima naudoti tokias komandas darbui su darbine atmintimi: *insert()*, *modify()*, *retract()* ir t. t.

Drools taisyklių mašiną galima panaudoti taisyklėmis grindžiamų VP simuliacijai tokiu būdu: taisyklės veiksmuose galima aprašyti veiksmus, kurie atitinka VP veiklas, panaudojant taisyklių sąlygas galima konstruoti verslo procesų sąlyginius išsišakojimus. Paleidus vykdyti tokiu būdu sukonstruotas taisykles, įvykdytų veiksmų grandinė atitiktų VP veiklų grandinę, kuri savo ruožtu yra proceso egzempliorius. Tačiau Drools neturi galimybės vykdymo metu keisti VT dėl taisyklių rinkinio kompiliavimo į nedalomą vykdomąją taisyklių rinkinį, kurį paleidus vykdyti nebegalima keisti.

2.3.2. OpenL Tablets

OpenL Tablets gamintojai teigia, jog joje yra naudojamos „verslui draugiškos taisyklės“. Tai dar viena atviro kodo BRMS. Ši sistema susideda iš kelių dalių: taisyklių mašinos, interneto puslapių sąsaja paremta VT valdymo aplinka, SOA žiniatinklio paslaugos, taisyklių saugyklos ir programavimo aplinkos (OpenL Tablets – Developer Guide 2014). Ši BRMS skirta naudoti *Java* kalbomis realizuotoms programoms. Taisyklių užrašymo kalba yra natūrali, struktūrizuota naudojant paprastas lenteles saugant *MS Excel* dokumentuose (6 pav.), todėl jos lengvai suprantamos tiek verslo atstovams, verslo analitikams, tiek programuotojams. Taisyklėms užrašyti gali būti naudojamos *Java* kalbos išraiškos ir funkcijos. Ši BRMS naudoja nuoseklų išvedimą, todėl rezultatui pasiekti aprašomame taisyklių rinkinyje turi būti apibrėžta jų vykdymo seka.

Rules void hello1(int hour)			
Rule	C1	C2	A1
	min <= hour	hour <= max	System.out.println(greeting + ", World!")
	int min	int max	String greeting
Rule	From	To	Greeting
R10	0	11	Good Morning
R20	12	17	Good Afternoon
R30	18	21	Good Evening
R40	22	23	Good Night

6 pav. OpenL Tablets taisyklių užrašymo pavyzdys

OpenL Tablets neturi galimybės atlikti taisyklių keitimo veiksmo taisyklių rinkinio vykdymo metu, todėl jos panaudojimas taisyklėmis grindžiamų dinaminių VP simuliacijai yra negalimas.

2.3.3. OpenRules

OpenRules yra atviro kodo BRMS, išleista 2003 m. Ši BRMS yra orientuota į verslo analitikus, suteikiant jiems lengvai kurti, testuoti, palaikyti verslo logiką naudojant *Microsoft™ Excel* arba *Google Docs* dokumentus (Open Souce Business Decision Management System 2013). Ši BRMS yra pakankamai gerai žinoma visame pasaulyje ir yra plačiai naudojama didelių korporacijų, bankų ir kitų institucijų. Ją galima integruoti tiek *Java*, tiek *C#* ir *.NET* programavimo aplinkose. OpenRules BRMS siūlo tokius komponentus:

- Rule Repository – taisyklių valdymui ir saugojimui;
- Rule Engine – taisyklių mašina;
- Rule Learner – taisyklių išgavimas;
- Rule Solver – taisyklių derinimas ir optimizavimas;
- Finite State Machines – įvykių valdymas;
- Rule Dialog – vartotojo sąsaja taisyklėms įvesti.

OpenRules taisyklės yra užrašomos lentelių pavidalu. 7 pav. pateikiamas OpenRules taisyklės užrašytos lentelėje pavyzdys, kuri pasako kreipimosi į asmenį būdą, panaudojant asmens lyties, šeimyninės padėties ir gimimo datos informaciją. Pateiktoje lentelėje, kiekviena eilutė nusako *Or* ryšį, o Condition stulpeliai – *And* ryšį. Conclusion stulpelis nurodo atsakymą – išvadą. Toks taisyklių užrašymo būdas yra gana paprastas ir lengvai perprantamas verslo analitikams, kadangi panaudojus taisyklių šablonus jiems užtenka tik užpildyti reikiamas reikšmes į lentelę ir patikrinti lentelės sintaksės teisingumą OpenRules taisyklių sintaksės tinkamumo vertintoju. Tokiu būdu šioje BRMS yra realizuotas 2.2.2 skyriuje apie BRMS architektūrą aprašytas vienas iš BRMS komponentų – taisyklių šablonai.

DecisionTable Define Salutation							
Condition		Condition		Condition		Conclusion	
Gender		Marital Status		Date of Birth		Salutation	
Is	Male			<	January 1, 2007	Is	Mr.
Is	Female	Is	Married			Is	Mrs.
Is	Female	Is	Single			Is	Ms.
				>=	January 1, 2007	Is	Little

7 pav. Taisyklių užrašymas naudojant lentelę

Programuotojų požiūriu, OpenRules integruoti į kuriamą PĮ yra gana paprasta. 8 pav. pateikiamas kodo fragmentas, parodantis, koku būdu panaudojus 7 pav. pateiktą taisyklę, galima gauti reikiamą atsakymą – šiuo atveju užtenka programos kode nurodyti failo, kuriame yra lentelė, taisyklės pavadinimą ir parametrus – lytį, šeimyninę padėtį ir gimimo metus. Įvykdžius metodą *execute()*, gaunamas reikiamas sprendimas.

```
String fileName = "file:rules/main/Decision.xls";
Decision decision = new Decision("DetermineCustomerGreeting", fileName);
Customer customer = new Customer();
customer.setName("Robinson");
customer.setGender("Female");
customer.setMaritalStatus("Married");
DateFormat df = DateFormat.getDateInstance(DateFormat.SHORT);
customer.setDob(df.parse("2/15/1990"));
decision.put("customer", customer);
Response response = new Response();
decision.put("response", response);
decision.saveRunLog(true);
decision.execute();
decision.log("Decision: " + decision.getOutput());
decision.printSavedRunLog("results.txt");
```

8 pav. Kodo fragmentas, parodantis OpenRules naudojimo būdą

Žiūrint į OpenRules iš taikymo dinaminių VP simuliacijai ir dinaminių taisyklių palaikymo perspektyvų, OpenRules skelbiasi, jog palaiko dinamines taisykles, tačiau jų pakeitimus galima atlikti tik prieš taisyklių rinkinio vykdymą, o vykdymo metu atlikti taisyklių pakeitimai nelems viso rezultato. Vadinasi šios BRMS panaudojimas taisyklėmis grindžiamų dinaminių VP simuliacijoje galimas tik tuo atveju, jeigu simuliuoti dinaminis VP tik simuliacijos rėmuose, bet ne proceso egzemplioriaus vykdymo metu.

2.3.4. IBM ILog JRules

Tai IBM korporacijos nupirkta iš kitų gamintojų ir vystomas komercinis BRMS produktas, siūlantis labai didelių galimybių spektrą įvairiais atžvilgiais: tiek iš verslo analitikų ir vadovų pusės, suteikiant jiems priemonės taisyklių valdymui, optimizavimui ir priežiūrai, tiek iš programuotojų pusės, suteikiant jiems priemonės integruoti PĮ su IBM ILog JRules BRMS paketu naudojant įvairias programavimo kalbas (palaiko tiek *Java*, tiek *C#* ir *.NET*). Panaudojant žiniatinklių paslaugomis grindžiamus sprendimus, priklausomybė nuo programavimo kalbos išnyksta. ILog JRules yra labai didelis PĮ paketas, kurį įdiegti, prižiūrėti reikia ne tik nemažai resursų, bet ir žinių. Vieni iš svarbesnių *ILog JRules* BRMS paketo suteikiamų galimybių – taisyklių versijų valdymo sistema, galimybė atlikti taisyklių testavimą ir simuliaciją prieš jų perkėlimą į darbinę aplinką.

Remiantis šaltiniu (IBM WebSphere ILOG JRules Version 7.0 2014), *ILOG JRules* yra lanksti BRMS, leidžianti užrašyti taisykles keliais būdais. Pavyzdžiui, viena iš jų – verslo veiksmų kalba

(angl. *BAL – Business Action Language*). Tokiu būdu apibrėžtus verslo žodyną (pavyzdžiui, kas yra pirkinių krepšelis, jo vertė ir pan.) galima užrašyti VT ir veiksmus, kuri yra labai artima natūraliai žmogaus kalbai (9 pav.). Todėl tokios taisyklės yra aiškios ir lengvai suprantamos verslo analitikams. Kitas būdas užrašyti taisykles – žemo lygio kalba, kurioje gali būti naudojami tiek programos kodo fragmentai, tiek žemo lygio operacijos atliekant tiesioginius veiksmus su darbine atmintimi (principas panašus kaip ir Drools BRMS atveju).

```
if
  the value of the shopping cart is more than $100
then
  apply a 15% discount on the shopping cart
else
  apply a 5% discount on the shopping cart
```

9 pav. ILog JRules BRMS naudojama natūralia kalba užrašytos taisyklės ir veiksmo pavyzdys

Ta pati pav. 9 pateikta taisyklė ir veiksmas užrašyta žemo lygio kalba, kurią IBM vadina ILog taisyklių kalba (angl. *IRL – Ilog Rule Language*):

```
rule rule_1 {
  when {
    shoppingCart: shoppingCart.Value > 100
  }
  then {
    modify shoppingCart { shoppingCart.Total = shoppingCart.Value -
      (shoppingCart.Value * 0.1); }
  }
  else {
    modify shoppingCart { shoppingCart.Total = shoppingCart.Value -
      (shoppingCart.Value * 0.05); }
  }
}
```

10 pav. ILog JRules BRMS naudojama žemo lygio taisyklės ir veiksmo pavyzdys

ILog JRules gamintojai skelbiasi, jog palaiko dinamines taisykles, jų keitimą vykdymo metu, tačiau tai gali būti atliekamas tik prieš pasirinkto taisyklių rinkinio vykdymą. Taip pat ši BRMS leidžia dinamiškai keisti iš anksto apibrėžtus parametrus, kurie gali būti naudojami taisyklėse. Yra galimybė, kaip teigia gamintojai apibrėžti taisyklių užduotis (angl. *Rule Tasks*), kuriuose nurodžius tam tikrus datos ir laiko režius ir jiems atitinkamai priskyrus taisyklių rinkinius, atitinkamu nurodytu laiku bus pasirenkamos taisyklių rinkiniai vykdymui. Tačiau ši priemonė nėra tinkama panaudoti dinaminių verslo procesų simuliacijai.

2.3.5. BRMS galimybių palyginimas

Rinkoje esančių BRMS mašinų palyginimui naudojami aukščiau nagrinėti BRMS, kurių palyginimų lentelė pateikiama lentelėje žemiau. Įrankių palyginimui naudojami tokie kriterijai kaip: dinaminių taisyklių palaikymas, pritaikymo galimybės panaudojant taisyklėmis grindžiamų

dinaminių VP simuliacijai, naudojamas algoritmas ir t. t. (detaliau žr. 4 lentelę). Tokie kriterijai yra aktualūs, norint panaudoti įrankius kuriant taisyklėmis grindžiamų dinaminių VP simuliaciją.

4 lentelė. BRMS palyginimas

Įrankis	Dinaminės taisyklės	Dokumentacija	Galimybės panaudoti simuliacijai	Atviro kodo	Naudojamas algoritmas	Suderinamumas su keleta kalbų	Komercinis	Aukšto lygio taisyklės
Drools	-	Išsami	+	+	Tiesioginis išvedimas, Rete	-	-	+/-
OpenL Tablets	-	Trūksta informacijos apie naudojimą programose	-	+	Optimizuotas nuoseklusis algoritmas (angl. <i>Optimized Sequential Algorithm</i>)	-	-	+
OpenRules	+/-	Išsami	-	+	Optimized sequential algorithm, Rete	+	-	+
IBM ILog	+/-	Išsami	+	-	Tiesioginis išvedimas	+	+	+

2.3.6. BRMS apibendrinimas

Vienas iš BRMS tikslų yra pateikiant įvairius įrankius suteikti galimybę su ja dirbti ne tik programuotojams, tačiau ir su programavimu nesusijusiais asmenims – verslo analitikams, verslo atstovams ir ekspertams. Priešingai nei *OpenL Tablets* ir *OpenRules*, *Drools* ir *IBM ILog* BRMS

suteikia galimybę užrašyti VT žemo lygio kalba, todėl naudojant šias BRMS kuriant didelės apimties sistemas turi lankstesnę PĮ pritaikymo prie konteksto galimybę. Žemo lygio kalba užrašytos taisyklės yra labai panašios į programos kodą (*Drools* leidžia naudoti *Java* kalba užrašytą kodą taisyklių veiksmuose), todėl atskyrus esminę programos ir verslo logiką į taisykles ir jų valdymui panaudojant centralizuotą taisyklių valdymo įrankius, galima sumažinti PĮ palaikymo išlaidas. Taip pat, apibendrinant nagrinėtas BRMS galima teigti, jog juose galima naudoti dviejų tipų taisykles:

- Taisyklės, kurios pateikia atsakymą (rezultatą), bet neatlieka veiksmo. Pavyzdžiui, - nuolaidos pritaikymo taisyklės atveju, uždavus klausimą BRMS „kokią nuolaidą reikia pritaikyti klientui“ su parametru išlaidų suma, BRMS panaudojus nuolaidų taikymo taisyklę grąžina atsakymą, jog „reikia taikyti 15% nuolaidą“. Toks metodas yra naudojamas *OpenRules*, *IBM ILog JRules* BRMS.
- Taisyklės, kurios atlieka veiksmą, t. y. aukščiau paminėtu atveju, BRMS rezultatas bus ne atsakymas „reikia taikyti 15% nuolaidą“, o suma su pritaikyta nuolaida, kurią reikia mokėti klientui. Šis metodas yra plačiau naudojamas dėl didesnių galimybių, kadangi į taisyklių veiksmus galima atskirti nemažai programos kodo ir kontroliuoti ją atskirai nuo PĮ.

Šio darbo kontekste, nei vienas iš analizuotų BRMS nesuteikia galimybės dinamiškai konstruoti vykdomų taisyklių seką ir lanksčiai bei dinamiškai keisti taisykles vykdymo metu, todėl jų panaudojimas dinaminių VP simuliacijai nėra galimas.

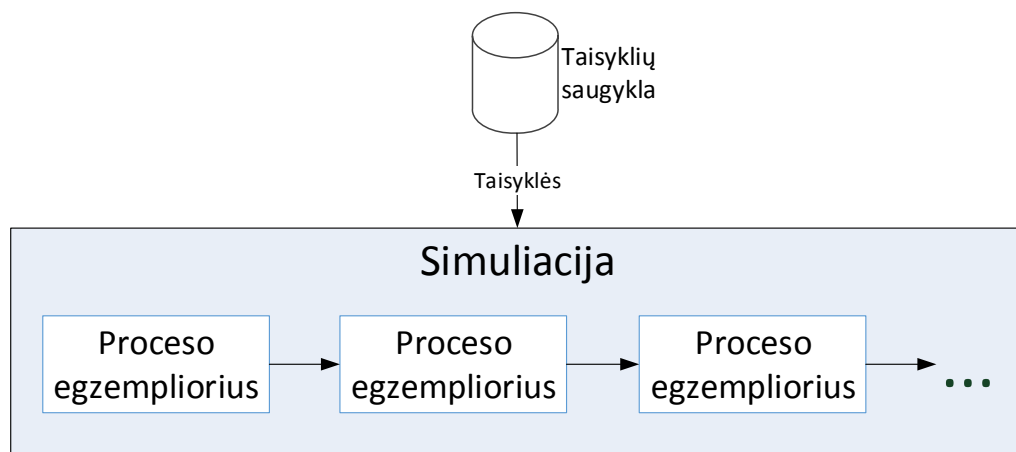
2.4. Apibendrinimas

Apibendrinant atliktų tiek taisyklėmis grindžiamų simuliacijos įrankių, tiek BRMS analizę, nustatyta kad šių priemonių gamintojai vis dar mažai dėmesio skiria dinaminių verslo procesų simuliacijos tyrimui ir analizuotų įrankių panaudojimas dinaminių VP simuliacijos tyrimui yra nepritaikyti. Todėl yra reikalingas metodas, kuris leistų dinamiškai vykdyti VP taisykles. Taip pat yra reikalinga priemonė, kurioje įgyvendinus metodą, būtų galima simuliuoti dinامينius VP.

3. TAISYKLĖMIS GRINDŽIAMŲ DINAMINIŲ VERSLO PROCESŲ SIMULIACIJOS METODAS

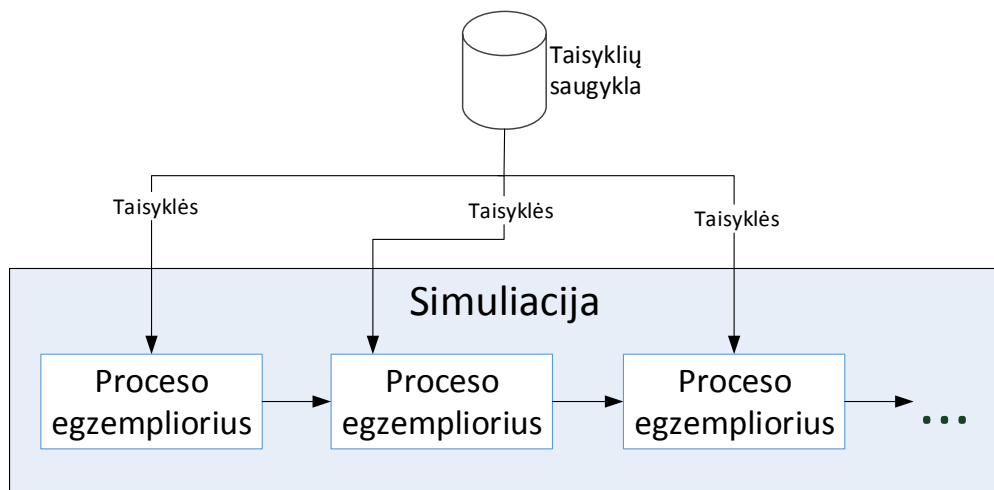
3.1. Dabartinė situacija

Atlikus taisyklėmis grindžiamų VP modeliavimo ir simuliacijos įrankių analizę, pastebėta nors ir įrankiai turi pakankamai daug funkcionalumo ir galimybių, tačiau jie neleidžia iš esmės simuliuoti modelio apribojimus – VT pakeitimus dinamiškai simuliacijos vykdymo metu. Taip pat atlikus BRMS analizę, taip pat buvo nustatyta jog jų panaudojimo galimybės taisyklėmis grindžiamų dinaminių VP simuliacijos priemonės realizacijai irgi yra ribotas. Tyrimo metu paaiškėjo, jog galima išskirti du naudojamus simuliacijos būdus, kurie yra grindžiami taisyklėmis. Pirma schema (11 pav.) parodo, jog visos simuliacijos metu yra naudojama taisyklių būsena, kuri buvo simuliacijos inicijavimo metu ir ta būsena yra naudojama visos simuliacijos metu. Schemoje pavaizduoti procesų egzemplioriai parodo, kad simuliacijos metų procesas yra vykdomas vienas po kito daug kartų. Toks simuliacijos būdas yra naudojamas analizuotuose modeliavimo ir simuliacijos įrankiuose.



11 pav. Simuliacijos schema Nr. 1

Antras būdas (12 pav.) parodo, jog simuliacijos metu, kiekvieno proceso egzemplioriaus simuliacijai yra naudojama naujausia taisyklių būsena. Tai reiškia, kad kai simuliacijos metu simuliuojant vieną proceso egzempliorių pakeitus taisyklės, pakeistos taisyklės atsispindės tik sekančio proceso egzemplioriaus simuliacijoje. Tokį simuliacijos būdą galima realizuoti panaudojus analizuotus BRMS.

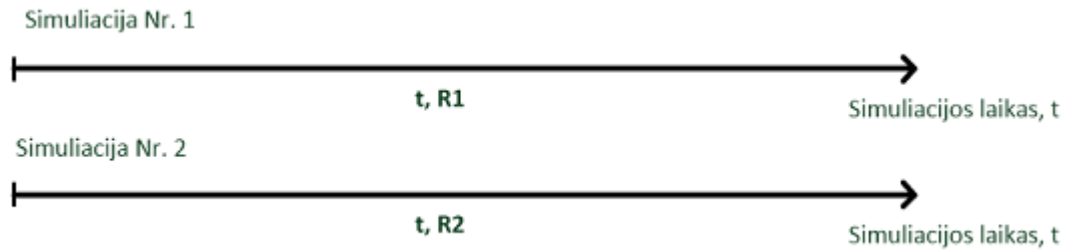


12 pav. Simuliacijos schema Nr. 2

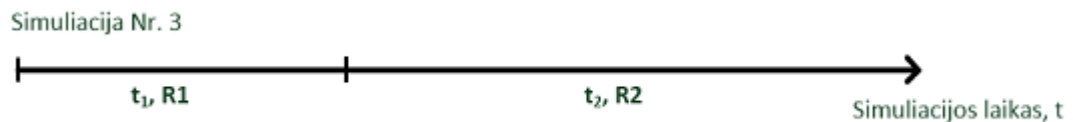
Naudojant aprašytus simuliacijos būdus, net ir sustabdžius VP simuliaciją, VT pakeitimai nelems simuliuojamo VP, kadangi pradėjus simuliaciją, VP vykdymas yra jau iš anksto apibrėžtas ir vykdant jo simuliaciją jis negali būti keičiamas. Nors kai kurių komercinių simuliacijos įrankių analizė (*Simprocess*, *Simul8*, *AcuProcess Modeler*) parodė, jog šių įrankių gamintojai neatskleidžia įrankio architektūros, jos veikimo principų, todėl nėra aišku kokie simuliacijos algoritmai yra naudojami, tačiau ištyrus jų veikimą galima nustatyti naudojamą simuliacijos būdą.

3.1.1. Įprastos ir dinaminės simuliacijos skirtumai

13 ir 14 paveiksluose yra pateikiami VP simuliacijos vykdymo skirtumai, kai VP yra simuliuojamas du kartus – pirmą su taisyklių rinkiniu R1, antrą – su pakeistu taisyklių rinkiniu R2. Turint galimybę simuliuoti VP dinamiškai, jo simuliacija laiko atžvilgiu (taip pat ir vieno proceso simuliacijos rėmuose) turėtų atrodyti taip, kaip pavaizduota 14 pav.: t_1 laiko trukmės simuliacija vykdoma su taisyklių rinkiniu R1, o likusį t_2 laiką – taisyklių rinkiniu R2. Bendras simuliacijos vykdymo laikas – t . Todėl pagrindiniai skirtumai tarp įprastos simuliacijos ir dinaminės yra, kad atliekant dinaminę simuliaciją pakanka vykdyti simuliaciją vieną kartą, ją galima lemti ir keisti taisyklių rinkinį tiek kartų kiek reikia ir gauti kitokį simuliacijos rezultatą, negu vykdant dvi simuliacijas atskirai su skirtingais taisyklių rinkiniais. Tokiu atveju galimybė dinamiškai simuliuoti VP, galima ištirti taisyklių pasikeitimų poveikį VP vykdymo eigoje, o tai yra svarbu šiuolaikiniame verslo aplinkoje dažnai keičiantis kontekstui, nes iš esmės versle dauguma procesų yra dinaminiai. Taip pat tai yra svarbu dėl to, nes dinaminės simuliacijos metu teoriškai galim gauti visiškai kitokius rezultatus, negu simuliuojant kiekvieną taisyklių rinkinį atskirai.



13 pav. Statinė simuliacija



14 pav. Dinaminė simuliacija

3.2. Siūlomas metodas

3.2.1. Naudojami terminai ir apibrėžimai

Siekiant suteikti aiškumo, šiame skyriuje apibrėžiami terminai, kurie yra aktualūs tiek kalbant apie siūlomą metodą, tiek apie jo realizaciją.

1. Programų sistemų lygmenyje kalbant apie VP veiklas, veiklos yra vadinamos veiksmais. VP veiklą gali atitikti vienas arba keli veiksmi.
2. Verslo sistemos požiūriu, VP yra veiklų seka, o programų sistemos požiūriu – veiksmų seka.
3. Kalbant apie taisyklę, turima omenyje ECA tipo taisyklę, kuri susideda iš įvykio, sąlygos ir veiksmo. ECA tipo taisyklė yra nedalomas vienetas (neskaidomas į mažesnius ECA tipo taisyklių vienetus). Vykdam ECA taisyklių rinkinį, taisyklių vykdymo seka atitinka VP, kadangi ECA taisyklių veiksmi atitinka 1-2 punktuose apibrėžtas VP veiklas.
4. Taisyklės veiksmas yra vykdomas, kai yra aktyvuojama ECA tipo taisyklė. Tai yra, kai įvyksta įvykis ir yra tenkinama taisyklėje apibrėžta sąlyga. Toliau šiame darbe yra naudojamas terminas „taisyklės vykdymas“, kuris reiškia aktyvuotos taisyklės, kurios sąlyga yra tenkinama, veiksmo vykdymas.
5. Kalbant apie VP simuliaciją, reikia skirti du skirtingus VP modelius – VP modelis skirtas vaizdavimui ir VP modelis skirtas simuliacijai. Dažnai vaizduojamas VP modelis nėra tinkamas simuliacijai vykdyti, todėl jis turi būti transformuojamas į simuliacinį VP modelį. Šiuo atveju, BPMN VP modelis gali būti transformuojamas į ECA taisyklėmis paremtą modelį, kuris tinkamas tiek vykdymui, tiek simuliacijai.

3.2.2. Galimi problemos sprendimo būdai

Kaip parodė atlikta BRMS analizė, pagrindinė problemos, dėl kurių jas negalima panaudoti dinaminei VP simuliacijai – prieš vykdymą iš anksto remiantis naujausia taisyklių būsena yra kompiliuojamas taisyklių rinkinys arba sudaromas galimų ėjimų medis, kurie yra nedalomi. Todėl kad spręsti šią problemą yra reikalinga atsisakyti tokio principo ir generuoti vykdymo seką dinamiškai.

Kadangi yra svarbu atsiskirti nuo nedalomų taisyklių rinkinių ir vykdymo metu konstruoti vykdomų taisyklių seką dinamiškai, todėl žinant šiuo metu naujausiais esamas programavimo technologijas, nustatyta kad tai gali būti realizuojama:

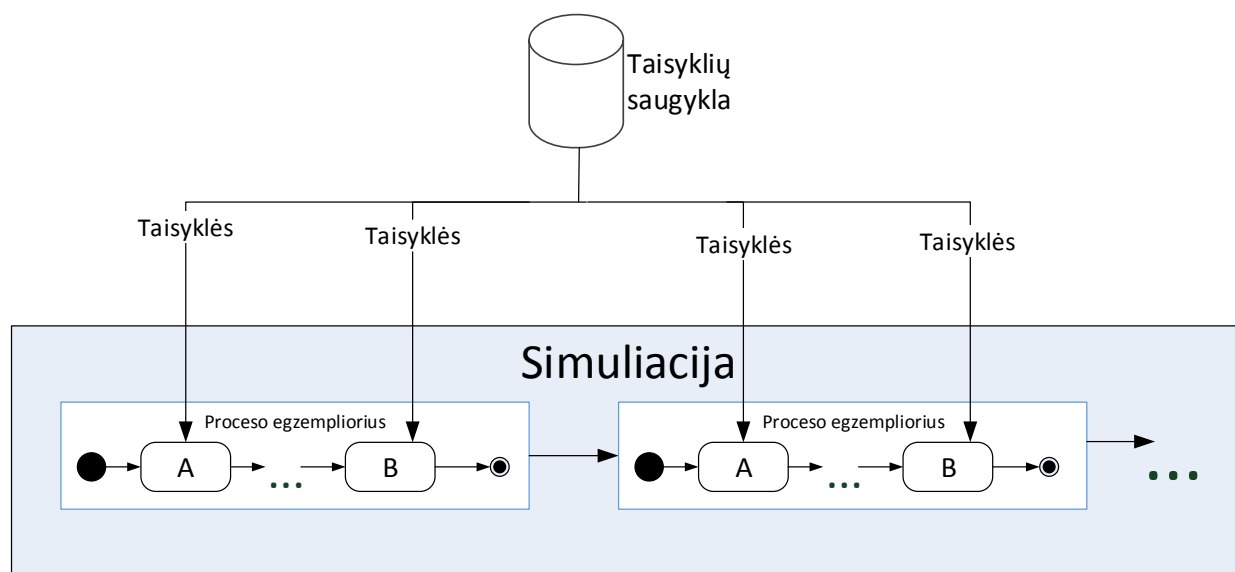
1. Naudojant interpretuojamas kalbas, kai prieš kiekvieną taisyklės vykdymui suradimą vykdymui yra interpretuojama naujausia taisyklių būsena.
2. Naudojant į atskirus komponentus kompiliuojamas taisykles, kurios būtų kai prieš kiekvieną taisyklės vykdymui suradimą yra remiamasi naujausia taisyklių būsena, o taisyklės kurios buvo pridėtos arba pakeistos būtų paruošiamos vykdymui – kompiliuojamos ir saugomos taisyklių mašinos atmintyje.

Dinaminiam VT valdymo mašinai įgyvendinti realizuoti pasirenkamas 2 realizacijos būdas. Tokio būdo privalumas – sistemai realizuoti naudojant vienas nuo kito nepriklausomus komponentus galima išgauti didelį sistemos lankstumą, nes dėl komponentų nepriklausomumo juos galima keisti kitais ir jų vykdymo seką konstruoti dinamiškai. Taip pat kompiliuojami ir vykdomi programiniai komponentai yra žymiai greitesni lyginant su interpretuojamomis kalbomis.

3.2.3. Siūlomo metodo aprašymas

Šio darbo kontekste, kaip dinaminių VP modeliavimo būdas yra pasirinktas literatūros analizėje aprašytas taisyklėmis grindžiamų VP modeliavimo požiūris, kai VP modelis aprašomas naudojant ECA tipo taisykles (žr. skyrių 1.3.5). Taisyklėms vykdyti yra naudojama taisyklių mašina, kuri yra pagrindinis BRMS komponentas. Siekiant suteikti galimybę simuliuoti VP dinamiškai, kad galima būtų konstruoti VP dinamiškai, taisyklių mašinos galimybės turi būti išplėstos realizuojant siūlomą metodą. Tai reiškia, jog simuliuojant VP dinamiškai, sustabdžius simuliaciją VP vykdymo metu esant taisyklėje X, turi būti galimybė įvesti naują taisyklę Y, kuri bus vykdoma sekanti arba to proceso egzemplioriaus simuliacijos eigoje jai dar nepasibaigus. Lygiai taip pat, turi būti galimybė simuliuojant VP esant taisyklėje X, sustabdžius simuliaciją pakeisti A taisyklę taip, kad sekanti būtų vykdoma ne A, o B taisyklė (pasirinktas alternatyvus kelias).

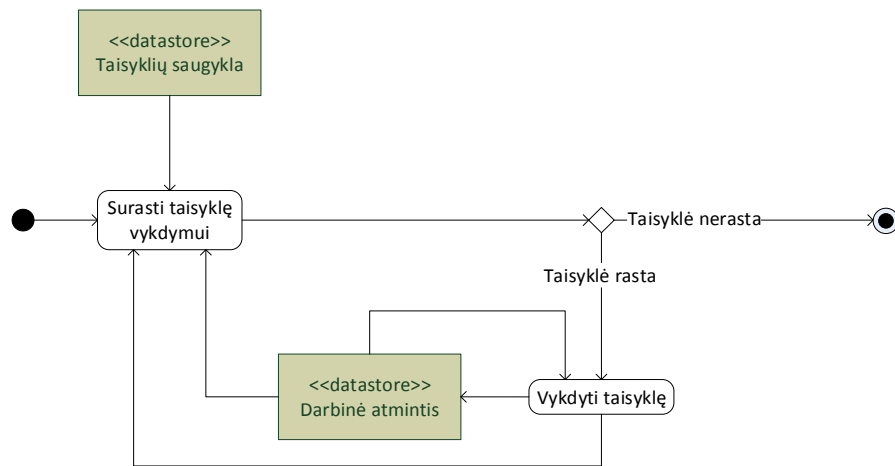
Remiantis jau anksčiau pateiktais simuliacijos schemomis, siūlomas naujas simuliacijos būdas (15 pav.), kai proceso egzemplioriaus simuliacijos metu, prieš kiekvienos taisyklės vykdymui nustatymo procesą, naudojama naujausia taisyklių būsena esanti taisyklių saugykloje.



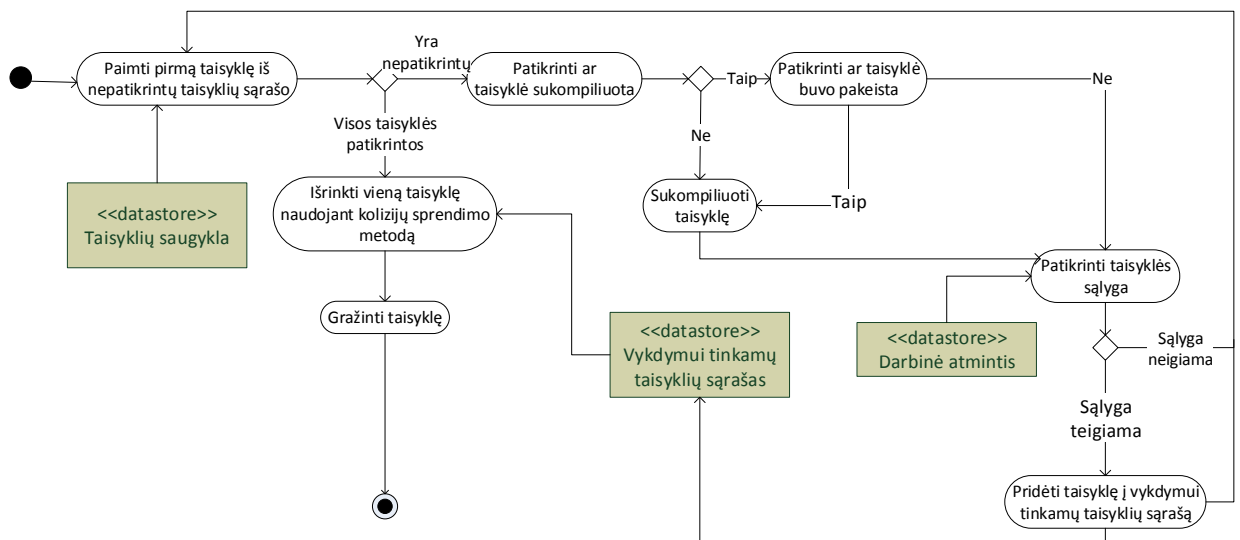
15 pav. Siūlomas simuliacijos būdas

Taisyklių mašina turi būti realizuota vykdomųjų taisyklių komponentų pagrindu, kur ECA taisyklės yra transformuojamos į programos kodą, kompiliuojamos į atskirą vykdomąjį komponentą ir saugomas taisyklių mašinos atmintyje iki tol kol taisyklė nebus pakeista. Tokiu būdu realizuojant vykdomaisiais taisyklių komponentais paremtą taisyklių mašiną, galima išgauti didelio našumo ir efektyvumo sistemą, kuri geba dideliu greičiu vykdyti taisykles ir nesant jų tarpusavio priklausomybei, dėl to galima išgauti didelio lankstumo galimybes vykdymo metu keisti, pridėti arba išimti taisykles.

Dinaminio VP vykdymą (vieno proceso vykdymo rėmuose) pavaizduotas UML veiklų diagrama 16 pav. Šios proceso esminė vieta yra veikla “Surasti taisyklę vykdymui”, kurios detalus algoritmas pateiktas UML veiklų diagramoje 17 pav.



16 pav. Dinaminio verslo proceso taisyklių vykdymo algoritmas



17 pav. Veiklos „Surasti taisyklę vykdymui naudojant taisykles“ detalus algoritmas

Apibendrinant šiuos du aukščiau pateiktus UML veiklos diagramas, bendras dinaminis proceso konstravimo algoritmas yra:

1. Paimti pirmą taisyklę iš nepatikrintų taisyklių sąrašo.
2. Jei taisyklė nesukompiliuota (nėra vykdomojo taisyklės modulio) – sukompiliuoti.
3. Jei taisyklė sukompiliuota, patikrinti ar taisyklė palyginus su esama sukompiliuota versija nėra pakeista. Jei taip – sukompiliuoti pakeistą taisyklę.
4. Patikrinti taisyklėje aprašytą sąlygą. Taisyklės sąlygos patikrinimui yra naudojama taisyklių valdymo mašinos darbinė atmintis, kuriame yra saugomi faktai. Jeigu sąlyga yra teigiama, taisyklė yra pridėjama į vykdymui tinkamų taisyklių sąrašą.
5. Kartoti 1-4 žingsnius, kol yra nepatikrintų taisyklių.

6. Patikrinus visas taisykles, naudojant kolizijų sprendimo algoritmą išrinkti vieną taisyklę, kuri grąžinama vykdymui. Jeigu tokios taisyklės nėra (vykdymui tinkamų taisyklių sąrašas yra tuščias), proceso vykdymas baigiamas.

7. Jeigu yra išrinkta taisyklė – vykdyti ją. Taisyklės vykdymui yra naudojama taisyklių mašinos darbinė atmintis ir jos vykdymo metu gali būti tiek naudojami, tiek keičiami arba pridedami darbinėje atmintyje esantys faktai. Po taisyklės vykdymo, vėl kartoti 1-7 žingsnius iki tol kol nebebus nustatoma taisyklė vykdymui 6 žingsnyje.

3.2.4. Siūlomo metodo nauda

Igyvendinus pasiūlytą metodą taisyklių mašinoje, bus galima:

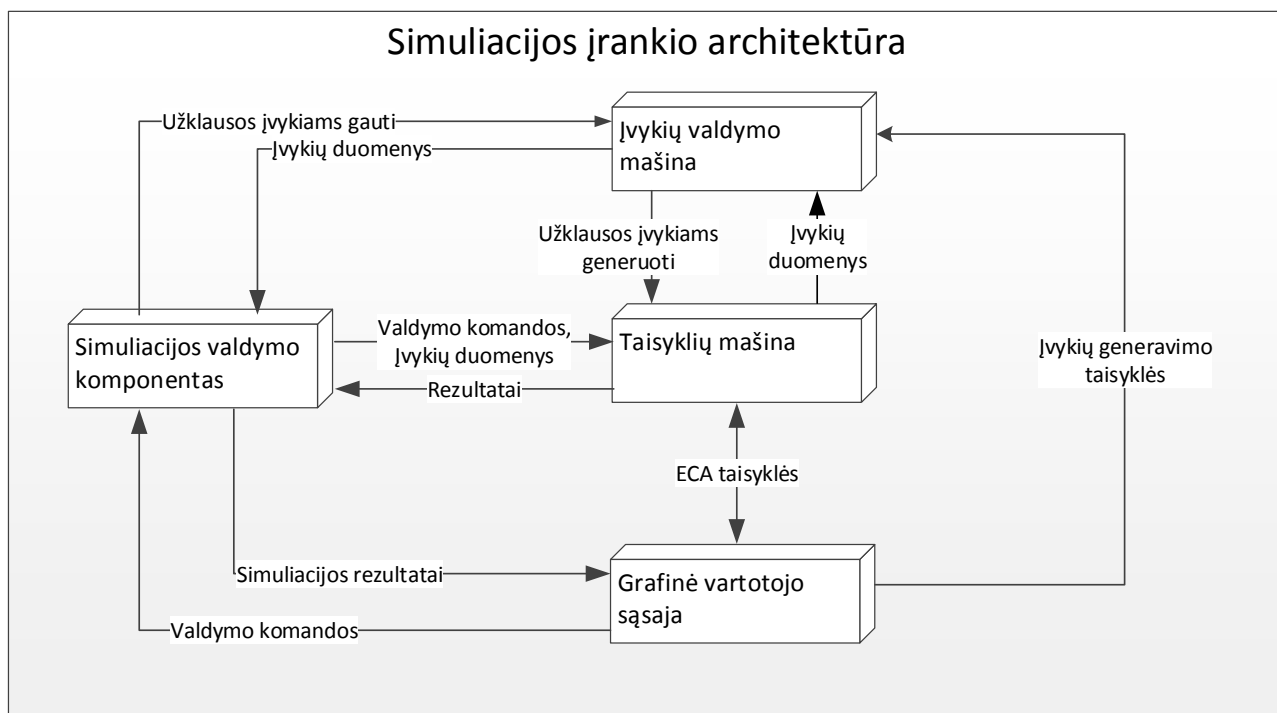
- Integruoti į kuriamą PĮ ir atskirti taisykles nuo programos logikos. Kadangi taisyklės galės bus keičiamos vykdymo metu, vadinasi bus galima lemti programos veikimą, keisti jos funkcionalumą vykdymo metu.
- Panaudoti taisyklėmis grindžiamų dinaminių VP simuliacijai ir tyrimui. Kadangi vienas iš pagrindinių dinaminių VP simuliacijos komponentų yra taisyklių mašina, kuri leidžia keisti VT vykdymo metu (taip pat pačioms taisyklėms keisti kitas taisykles simuliacijos vykdymo metu), bus suteikta galimybė proceso vykdymo metu keisti pačio proceso eigseną ne kaskart paleidžiant simuliaciją iš naujo, o bet kuriame proceso taške keisti proceso eigą: vykdymo šaką, proceso struktūrą, būseną. Kadangi panaudojant tokią mašiną galima realizuoti tokias VT, kurios keistų kitas VT, atsiras galimybė tyrinėti ir eksperimentuoti tokius uždavinius kaip automatiškai prisitaikančių VP, save optimizuojantys VP.
- Kiti moksliniai tyrimai ir eksperimentai. Galimybė kurti sistemas paremta moduliais, kurie vieni nuo kitų nepriklauso ir jie gali būti prijungiami, keičiami arba atjungiami nuo sistemos tiesiog jos veikimo metu.

Panaudojus siūlomą metodą ir įgyvendinus taisyklių mašiną, turėtų būti pademonstruotas aukšto lygio simuliacijos dinamiškumo lygis, kurio, kaip parodė įrankių analizė 2 skyriuje, šiuo metu neturi įgyvendinusios nei viena rinkoje esančios taisyklėmis grindžiamų VP simuliacijos įrankiai. Turint galimybę lanksčiai ir dinamiškai simuliacijos metu simuliuoti procesą ir taisykles, vienas iš privalumų – nepertraukiama simuliacija (nenutrūkstantis simuliacijos procesas), o jo vykdymas lemiamas ir konstruojamas dinamiškai. Tai yra naudinga dėl to, nes realiame pasaulyje procesai iš prigimties yra dinamiški ir jie nėra pertraukiami – jie yra palaipsniui keičiami ir tobulinami. Todėl įgyvendinus tokią galimybę VP simuliacijos įrankiuose, būtų galima simuliuoti konteksto pokyčius, kurie lemia taisyklių pokyčius dinamiškai.

3.3. Dinaminių verslo procesų simuliacijos įrankio prototipo architektūra

Kadangi esamų priemonių panaudojimas tobulinimui yra sudėtingas ir painus, be to ne visos jos yra atviro kodo, todėl nuspręsta kurti naują simuliacijos įrankio protipą. Simuliacijos įrankio pagrindas turi būti taisyklių mašina, kuri leidžia dinamiškai konstruoti vykdomų taisyklių seką. Kiti papildomi komponentai yra skirti simuliacijai atlikti ir valdyti. Tolimesniuose skyriuose yra pateikiami suprojektuota simuliacijos įrankio prototipo architektūra su esančių vidinių bei išorinių komponentų sąryšiais.

3.3.1. Simuliacijos įrankio architektūra



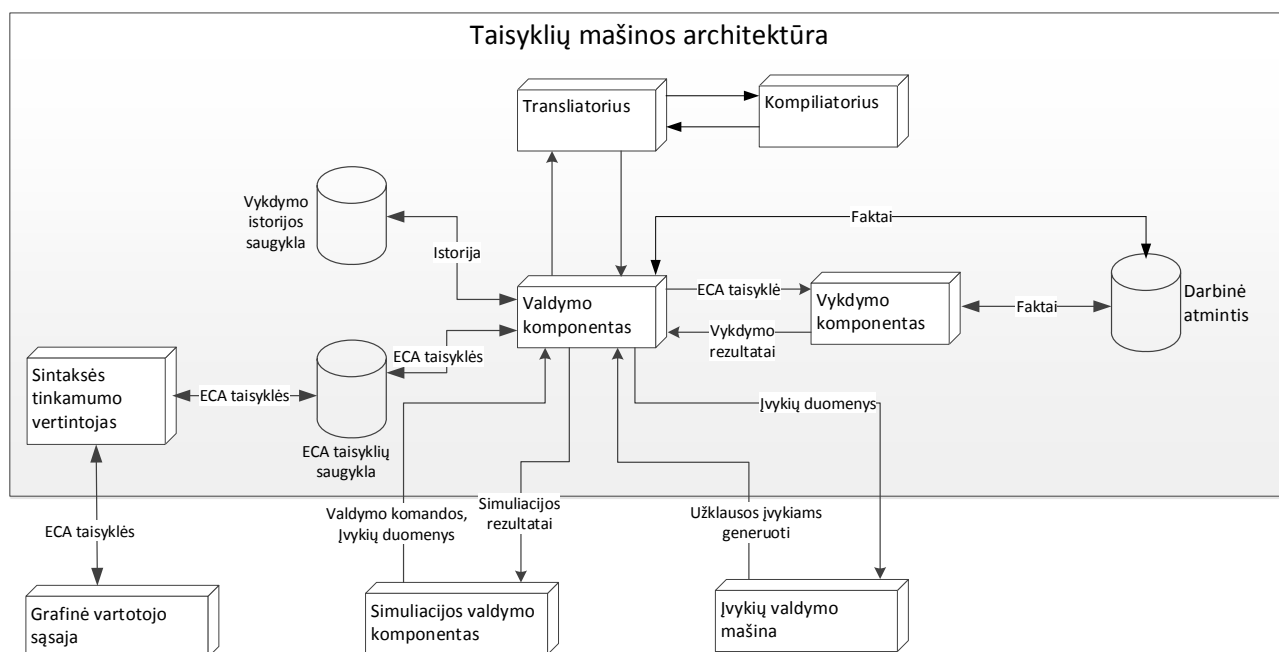
18 pav. Taisyklėmis grindžiamų dinaminių VP simuliacijos įrankio prototipo architektūra

Simuliacijos įrankio prototipas turi susidėti iš tokių komponentų: (18 pav.):

- Įvykių valdymo mašinos. Įvykių valdymo mašina turi suteikti galimybę generuoti ir valdyti įvykius. Tai įvykiai, kurie inicijuoja procesą, todėl juos čia galima traktuoti kaip VP inicijavimo trigerius. Įvykiai turėtų būti generuojami iš grafinės vartotojo sąsajos panaudojant įvykių generavimo taisykles. Įvykių valdymo mašina panaudojus įvestas įvykių generavimo taisykles ir taisyklių mašinos komponentą, gali sugeneruoti įvykius simuliacijai.
- Simuliacijos valdymo komponento. Simuliacijos valdymo komponentas turi suteikti galimybę kontroliuoti simuliacijos procesą, jos greitį, ją sustabdyti. Šis komponentas turi aptikus įvykį patalpinti įvykio faktą į taisyklių mašiną ir ją įjungti taip pradedant proceso simuliaciją. Simuliacija kontroliuojama panaudojus grafinę vartotojo sąsają.

- Taisyklių mašinos. Tai svarbiausias simuliacijos įrankio komponentas, kuriame yra realizuojamas skyriuje 3.2.3 aprašytas algoritmas.
- Grafinės vartotojo sąsajos. Grafinė vartotojo sąsaja turi leisti valdyti visą simuliacijos procesą pasiunčiant valdymo signalus į simuliacijos valdymo komponentą. Nusiuntus simuliacijos stabdymo komandą, panaudojant grafinės vartotojo sąsajos pateikiamomis taisyklių valdymo galimybėmis, gali būti taisyklių mašinoje keičiamos, įvedamos naujos arba šalinamos senos taisyklės ir pratęsus simuliaciją nuo sustabdyto taško stebėti tų taisyklių pasikeitimų poveikį simuliacijos eigai ir rezultatui.

3.3.2. Taisyklių mašinos architektūra



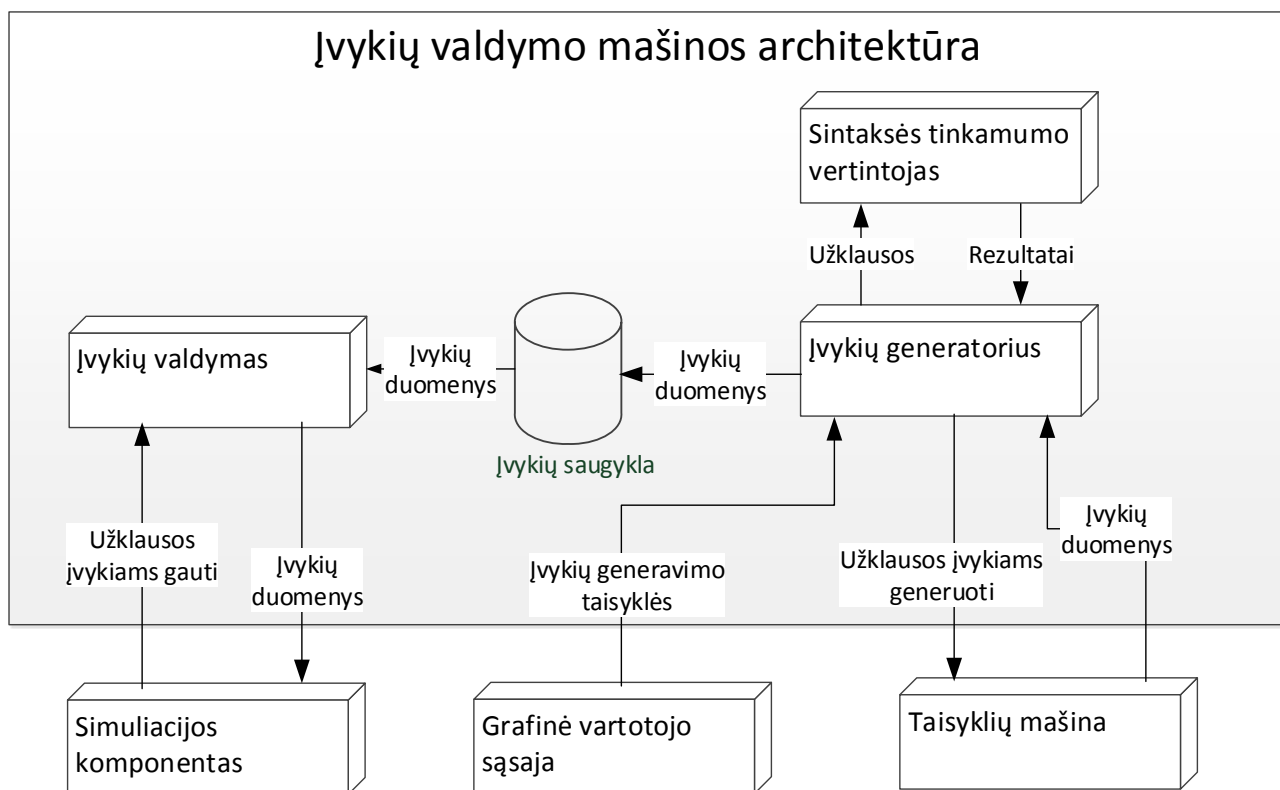
19 pav. Taisyklių mašinos architektūra

Svarbiausias simuliacijos įrankio komponento vidinėje jos architektūroje turi būti tokie komponentai (19 pav.):

- ECA taisyklių saugykla – saugykla, kurioje saugomos ECA tipo taisyklės.
- Darbinė atmintis – pagrindinė taisyklių valdymo mašinos atmintis, kurioje gali būti saugomi tiek taisyklių mašinos sukurti faktai, tiek ją naudojančios programinės įrangos faktai. Faktai – programos objektų egzemplioriai.
- Vykdymo istorijos saugykla – žurnalas, kurioje fiksuojami visa su taisyklių vykdymu susijusi informacija siekiant vėliau panaudoti taisyklių derinimui.

- Sintaksės tinkamumo vertintojas - taisyklių sintaksės tikrinimo komponentas, kuris turi suteikti galimybę patikrinti ar taisyklės sintaksė yra teisinga. Į ECA taisyklių saugyklą galima įvesti tik sintaksiškai teisingas taisykles.
- Transliatorius – VT, kurios užrašytos aukšto lygio kalba (priedas A.1.) transliavimas į programos kalbą (priedas A.2.), kurią galima sukompiliuoti.
- Kompiliatorius – sutransliuotos taisyklės iš aukšto lygio kalbos į programos kalbą pavertimas į vykdomąjį modulį. Sukompiliuoti galima tik sutransliuotą taisyklę (priedas A.2.).
- Valdymo komponentas – pagrindinis taisyklių valdymo mašinos branduolys, kurio paskirtis valdyti ir vykdyti procesą.
- Vykdymo komponentas – taisyklių vykdymas, kuris gali vykdyti taisyklės sąlygą (patikrinti ar taisyklės sąlyga tenkinama naudojant darbinę atmintį) ir vykdyti veiksmus (taip pat panaudojant darbinę atmintį). Komponentas gali vykdyti tik sukompiliuotas taisykles. Veiksmų vykdymo rezultatai gali būti talpinami darbinėje atmintyje.

3.3.3. Įvykių valdymo mašinos architektūra



20 pav. Įvykių valdymo mašinos architektūra

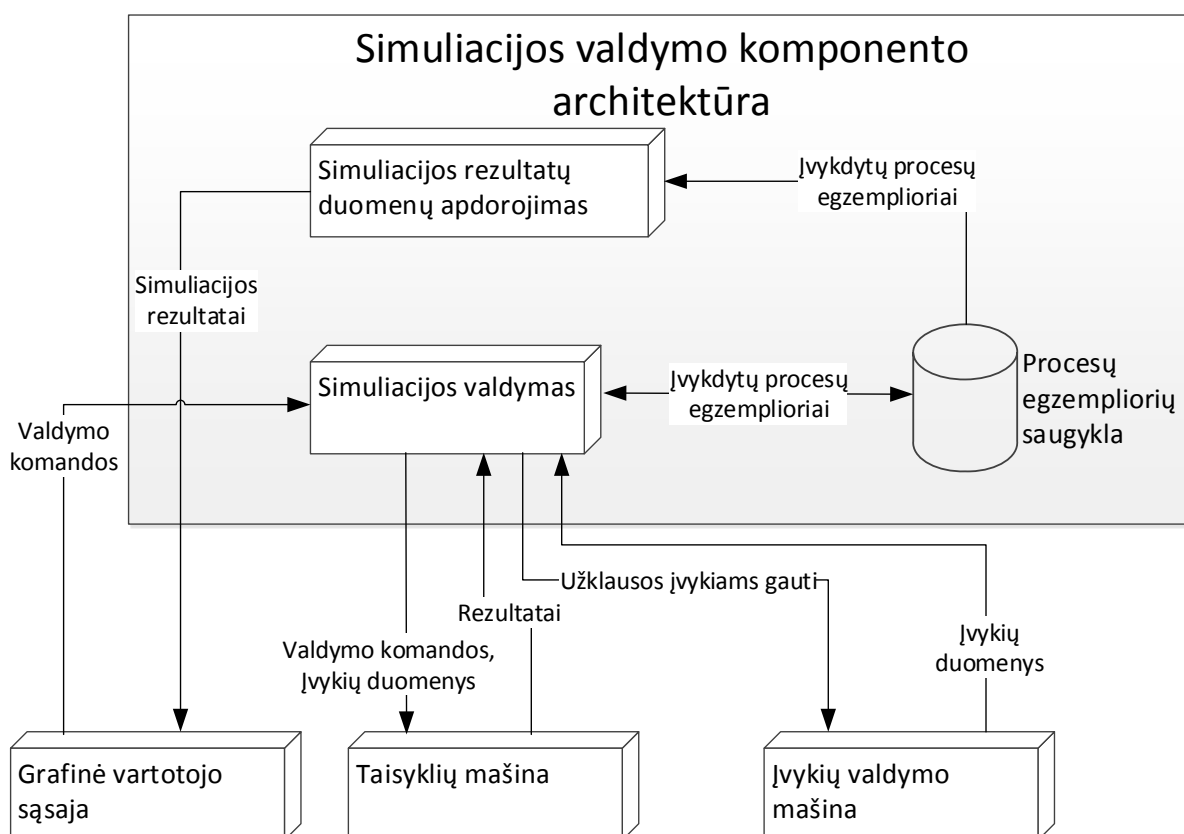
Įvykių valdymo mašinos vidiniai komponentai (20 pav.):

- Įvykių generatorius – šio komponento paskirtis remiantis vartotojo įvykių generavimo taisyklėmis sugeneruoti įvykius, kurie naudojami kaip trigeriai VP inicijuoti. Įvykių

generatorius turi palaikyti atsitiktinių skaičių generavimą, tokiu būdu sugeneruojant didelį skaičių įvykių egzempliorių su skirtingais atributų reikšmėmis.

- Sintaksės tinkamumo vertintojas – vartotojo įvedamų įvykiams generuoti taisyklių sintaksės tikrinimo komponentas.
- Įvykių saugykla – šioje bazėje patalpinami įvykių generatoriaus sugeneruojami įvykiai.
- Įvykių valdymas – įvykių valdymo komponentas, kuris turi suteikti galimybę paimti įvykius iš įvykių duomenų bazės ir jas pateikti simuliacijos valdymo komponentui.

3.3.4. Simuliacijos valdymo komponento architektūra



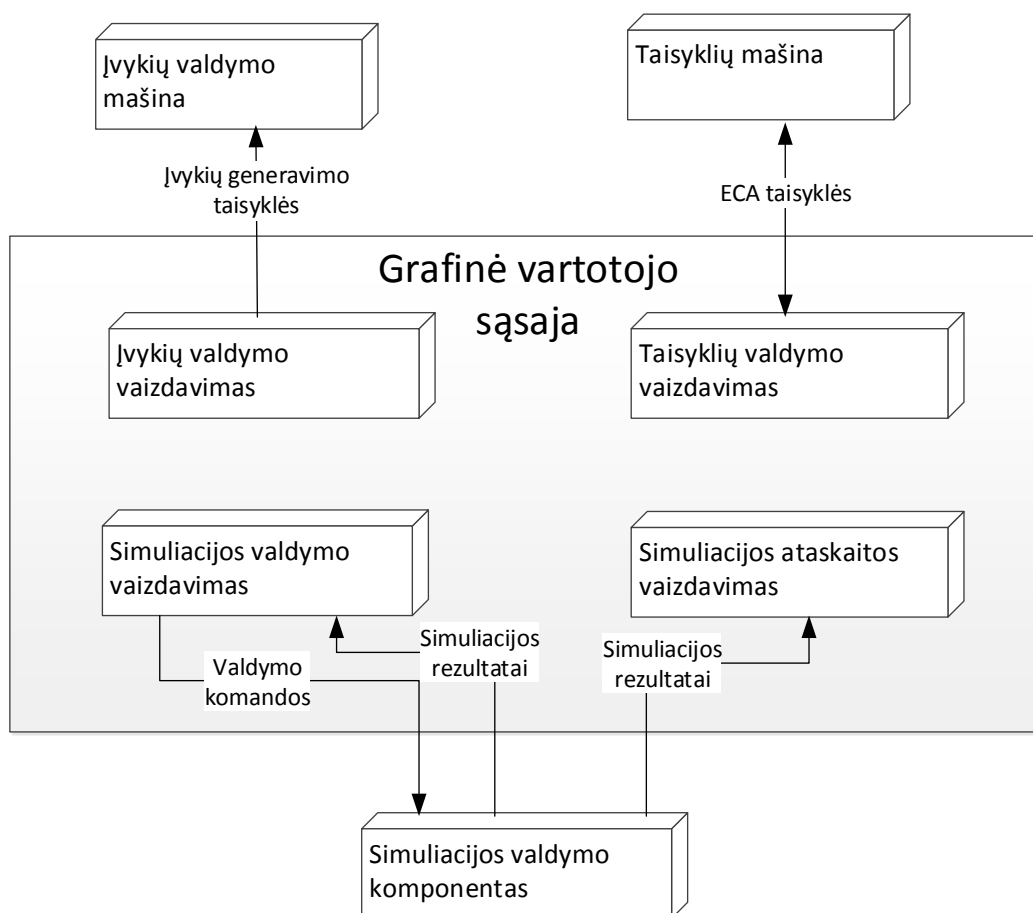
21 pav. Simuliacijos valdymo komponento architektūra

Simuliacijos valdymo komponentas yra antras pagal svarbumą komponentas, kurio pagrindinė paskirtis – valdyti simuliaciją (21 pav.)

- Simuliacijos valdymas – komponento paskirtis yra valdyti simuliaciją suteikiant galimybę kontroliuoti jos vykdymo greitį, ją sustabdyti ir pratęsti neprarandant simuliacijos būsenos ir duomenų. Taip pat šis komponentas savyje turi turėti simuliacijos laikrodį, kuris yra atskirtas nuo realaus pasaulio laiko.

- Simuliacijos rezultatų duomenų apdorojimas – panaudojus procesų egzempliorių duomenų bazėje sukauptą informaciją, šis komponentas turi gebėti apdoroti duomenis ir pateikti simuliacijos duomenų statistiką.
- Procesų egzempliorių bazė - vykdant simuliaciją, čia patalpinami kiekvieno vykdomo proceso egzemplioriai ir su jų vykdymo rezultatai.

3.3.5. Grafinės vartotojo sąsajos architektūra



22 pav. Grafinės vartotojo sąsajos komponento architektūra

Grafinės vartotojo sąsajos komponento paskirtis – suteikti galimybę vartotojui patogiai grafiniu būdu naudotis simuliacijos įrankio prototipu. Todėl šis komponentas susideda iš 4 mažesnių komponentų (22 pav.) – įvykių valdymo, simuliacijos valdymo, taisyklių valdymo ir simuliacijos ataskaitos vaizdavimų, kurie atitinkamai leidžia valdyti ir stebėti darbą. Jie tarpusavyje nėra susieti, nes visi šie grafinės sąsajos dalys yra nepriklausomi.

3.4. Apibendrinimas

Pasiūlytas metodas taisyklių mašinos funkcionalumo išplėtimui leidžia dinamiškai konstruoti vykdomų taisyklių seką. Pasiūlyta metodą realizuojanti simuliacijos įrankio architektūra turi būti

įgyvendinta praktiškai realizuojant simuliacijos įrankio prototipą, kuria remiantis bus atliekamas eksperimentinis tyrimas.

4. SIMULIACIJOS ĮRANKIO PROTOTIPO REALIZACIJOS APRAŠYMAS

4.1. Kūrimo priemonės

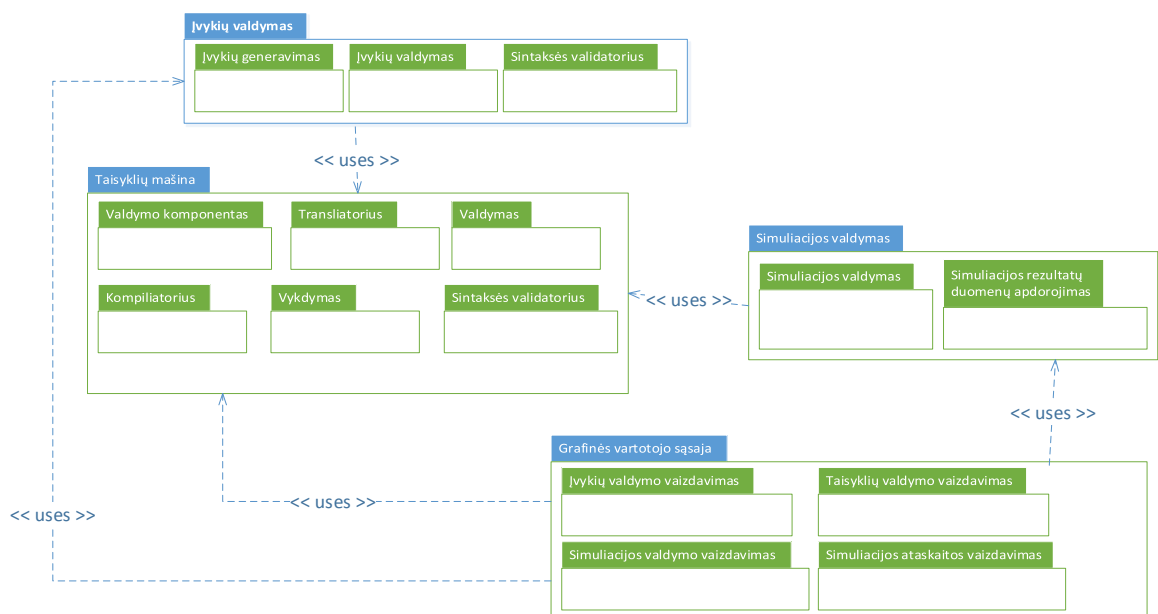
Simuliacijos įrankio prototipas buvo realizuojamas naudojant Microsoft™ sukurtas priemones: *Visual Studio 2012*, *C#* programavimo kalbą ir *.NET 4.5* versijos programavimo karkasą. Realizacijos metu buvo naudoti *Windows Server 2012* serveris ir *Team Foundation Server 2012* versijų kontrolės sistema programos kodo versijų valdymui. Vartotojo sąsajai kurti buvo panaudotas *Windows Presentation Foundation* komponentai, kurie yra standartiniai *.NET* karkaso paketai. Šie komponentai leidžia projektuoti ir programuoti grafinę vartotojo sąsają *Windows* operacinei sistemai.

Tokio technologijų ir įrankių pasirinkimo priežastys:

- *C#* programavimo kalba ir *.NET* karkasas yra viena iš plačiausių naudojamų ir geriausiai išvystytų technologijų komercinei PĮ kurti.
- Labai aukšto lygio programavimo kalba ir *.NET* karkaso pateikiami įvairių paruoštų komponentų naudojimui leidžia efektyviai ir greitai realizuoti įvairius sprendimus.
- Palyginus su kitomis panašiomis kalbomis ir programavimo technologijomis (*Java*, *Python*), *C#* programavimo kalba ir *.NET* platforma realizuota PĮ yra spartesnė vykdymo atžvilgiu.
- *Windows Presentation Foundation* komponentų pateikiamos galimybės yra vienos iš plačiausių lyginant su kitais esančiais komponentais (*Qt*, *Windows Forms* ir pan.).

4.2. Įrankio realizacija

4.2.1. Paketų diagrama



23 pav. Simuliacijos įrankio paketų diagrama

Remiantis 3 skyriuje pateikta simuliacijos įrankio architektūra, realizacijos lygmenyje buvo sukurti paketai, kurių viduje yra realizuoti 3.3 skyriuje aprašyti komponentai (23 pav.):

- Grafinė vartotojo sąsaja – vartotojui patogi ir informatyvi sąsaja, kuri suteikia galimybę vartotojui valdyti ir generuoti įvykius, įvesti bei keisti taisykles, valdyti simuliaciją ir peržiūrėti jos vykdymo duomenis ir statistiką.
- Taisyklių mašina – simuliacijos įrankio branduolys, kuris atsakingas už ECA taisyklių tikrinimą, valdymą ir vykdymą ir suteikiantis galimybę joje vykdymo metu keisti taisykles. Jos viduje, valdymo komponente yra realizuotas 16 pav. ir 17 pav. pateikiamas metodas.
- Simuliacijos valdymas – atsakingas už simuliacijos valdymą (įskaitant simuliacijos laiką), leidžiantis vykdymo metu pagreitinti, sulėtinti simuliaciją arba ją sustabdyti tam tikrame simuliacijos vykdymo būsenoje. Taip pat šiame pakete realizuota galimybė vykdyti simuliaciją pažingsniui, kuri leidžia iš esmės tirti simuliacijos eigą.
- Įvykių valdymas – atsitiktinių įvykių generavimo mašina, kuri atsakinga už įvykių generavimą naudojant įvykių generavimo taisykles. Sugeneruoti įvykiai naudojami atliekant simuliaciją, kaip procesų inicijavimo trigeriai.

4.2.2. Taisyklių sintaksė, komandos

Taisyklių mašinos realizacijos metu reikėjo pasirinkti taisyklių užrašymo struktūrą, kalbą ir komandas, kurios būtų skirtos darbui su darbine atmintimi. Todėl kai kurios užrašymo idėjos ir naudojamų komandų pavadinimai bei principai yra pasirinkti remiantis BRMS analizės metu analizuoto Drools BRMS idėjomis.

4.2.2.1. Taisyklių struktūra

Kiekviena taisyklė susideda iš tokių duomenų:

- Identifikatorius – unikalus kiekvienai taisyklei, kuri yra taisyklių bazėje;
- Pavadinimas – taisyklės pavadinimas;
- Prioritetas – taisyklės prioritetas (naudojamas sprendžiant konfliktus pagal pasirinktą konfliktų sprendimo būdą);
- Sąlyga – sąlyga, kurią patenkinus yra vykdomas veiksmas;
- Veiksmas – veiksmas, kuris yra vykdomas patenkinus sąlygą.

4.2.2.2. Taisyklių sintaksė

Taisyklės užrašymo pavyzdys pateikiamas priede A.1. Naudojama taisyklių užrašymo kalba yra panaši į *Visual Basic* programavimo kalbą su tam tikromis išlygomis:

- Faktų kintamieji darbinėje atmintyje yra pasiekiami naudojant išraišką *\$pavadinimas*, iš čia *pavadinimas* – fakto pavadinimas. Tokiu būdu galima pasiekti faktą ir jo atributus, pavyzdžiui, *\$AkcijuPirkimas.PirktiVienetu* gali grąžinti skaičių 100.
- Aktualūs simuliacijai duomenys stebėjimui pasiekiami naudojant išraišką *#pavadinimas*, iš čia *pavadinimas* – aktualaus duomens kintamojo pavadinimas. Tokiu būdu galima pasiekti arba sukurti naują aktualų stebėjimo kintamąjį, pavyzdžiui, *#Bankas* gali grąžinti skaičių 150 arba *#Bankas = 200* reiškia *Bankas* aktualaus stebėjimo duomens kintamojo priskyrimas reikšme 200.

Faktų ir aktualių stebėjimo duomenų pavadinimai turi būti išreiškiami naudojant lotynų kalbos raidėmis nenaudojant tarpų.

Faktai ir aktualūs simuliacijai duomenys yra saugomi darbinėje atmintyje, tačiau aktualūs simuliacijai duomenys, priešingai nei įprasti faktai, po kiekvieno proceso egzemplioriaus įvykdymo nėra šalinami. Todėl aktualūs simuliacijai duomenys simuliacijos įrankio prototipe yra naudojami kaip procesų simuliacijos rezultatų vertinimo kriterijai

4.2.2.3. Galimos komandos taisyklėse

Taisyklėse turi būti galimybė dirbti su darbine atmintimi, todėl žemiau yra apibrėžiamos komandos, kurios užrašant taisyklę gali būti naudojamos darbui su darbine atmintimi:

- *CREATE(pavadinimas)* – fakto sukūrimas ir patalpinimas darbinėje atmintyje su nurodytu pavadinimu, pavyzdžiui, *CREATE("AkcijuPirkimas")*.
- *REMOVE(pavadinimas)* – fakto pašalinimas iš darbinės atminties pagal nurodytą fakto pavadinimą, pavyzdžiui, *REMOVE("AkcijuPirkimas")*.
- *MEM(pavadinimas)* – gauti faktą iš darbinės atminties pagal jo pavadinimą, pavyzdžiui, *MEM("AkcijuPirkimas")* grąžins faktą *AkcijuPirkimas*.
- *MEM(pavadinimas, faktas)* – pakeisti faktą su nurodytu pavadinimu kitu kito fakto objektu, pavyzdžiui, *MEM("Akcijupardavimas", MEM("AkcijuPirkimas"))*.
- *POINTS(pavadinimas)* – gauti simuliacijos aktualių stebėjimo duomenų pagal jo pavadinimą, pavyzdžiui, *POINTS("Bankas")* gali grąžinti skaičių 150.
- *POINTS(pavadinimas, reikšmė)* – pakeisti aktualų stebėjimo duomenį pagal nurodytą pavadinimą nurodyta reikšme, pavyzdžiui, *POINTS("Bankas", 200)*.
- *Facts.Exists(pavadinimas)* – patikrinti ar faktas pagal nurodytą pavadinimą egzistuoja darbinėje atmintyje. Komandos rezultatas *true* arba *false*. Pavyzdžiui, *Facts.Exists("AkcijuPirkimas")* grąžins reikšmę *true*, jei toks faktas egzistuoja darbinėje atmintyje.

- `Points.Exists(pavadinimas)` – patikrinti ar aktualus stebėjimo duomuo pagal nurodytą pavadinimą egzistuoja atmintyje. Komandos rezultatas *true* arba *false*. Pavyzdžiui, `Points.Exists("Bankas")` grąžins reikšmę *true*, jei toks duomuo egzistuoja.

4.2.3. Įvykių generavimo taisyklės

Įvykių generavimo taisyklių paskirtis – sugeneruoti įvairius įvykius, kurie naudojami simuliacijoje kaip trigeriai, kurie gali inicijuoti procesą. Generuojami įvykiai gali būti bet kokie, priklauso nuo įvykių generavimo metu naudojamos generavimo taisyklės aprašo. Kiekvienas sugeneruotas įvykis gali su savimi turėti įvairius, su įvykiu susietus duomenis. Todėl naudojant įvykių generavimo komponentą, galima sugeneruoti tokios struktūros įvykius, pavyzdžiui:

Įvykio pavadinimas: *AkcijuKainosPokytis*

Įvykio duomenys: *ImonesPavadinimas = "ImoneA"*

AkcijuKaina = 17

Data: 2014-03-02

Laikas: 14:45

4.2.3.1. Įvykių generavimo duomenys

Įvykiams generuoti reikalingi duomenys:

- Įvykio pavadinimas;
- Generavimo taisyklė (detaliau žr. 4.2.3.2);
- Datos režis, kurie gali būti priskirti sugeneruotiems įvykiams, pavyzdžiui, nuo 2014-01-01 iki 2014-04-01;
- Laiko režis, kurie gali būti priskirti sugeneruotiems įvykiams, pavyzdžiui, nuo 8:00 iki 17:00 val.;
- Generuojamų įvykių kiekis.

4.2.3.2. Įvykių generavimo taisyklių sintaksė

Įvykių generavimo taisyklėse turi būti aprašytos naudojant priskyrimo sakinius, kurie tam tikriems atributams priskiriamos tam tikros reikšmės (pavyzdys pateiktas priede B). Todėl įvykių generavimo taisyklėse yra galimos tokios išraiškos, kurios vienas nuo kitos yra atskiriamos nauja eilute:

- *Atributas = reikšmė* (pavyzdžiui, *Kaina = 15*);
- *Atributas = "Žodis"* (pavyzdžiui, *Adresas = "Saulėtekio al. 15, Vilnius"*);

- *Atributas = Komanda* (pavyzdžiui, *Tikimybė = RANDOM(80.0, 85.5)*, plačiau apie komandas žr. skyrių 4.2.3.3).

Atributo pavadinimas gali būti išreiškiamas tik lotyniškais raidėmis nenaudojant tarpų.

4.2.3.3. Įvykių generavimo galimos komandos

Norint suteikti galimybę sugeneruoti atsitiktinius įvykius su atsitiktiniais duomenimis, yra reikalingas atsitiktinių skaičių generatorius. Todėl įvykių generavimo taisyklėje gali būti naudojama žemiau aprašyta atsitiktinių skaičių generavimo komanda:

- *RANDOM(reikšmė_nuo, reikšmė_iki)* – atsitiktinių skaičių generavimas nurodyto intervalo režyje. Komandos parametrai *reikšmė_nuo* ir *reikšmė_iki* nurodo intervalą, kuriame galima sugeneruoti atsitiktinį skaičių. Pavyzdžiui, įvykdžius *RANDOM(10, 50)*, pirmą kartą vykdant gali grąžinti skaičių 14, antrą – 21 ir t.t.

4.3. Realizacijos aprašymas

Šiame skyriuje yra pateikiami realizuotų komponentų aprašymai. Realizacijos lygmenyje realizuotų klasių kiekis yra didelis, todėl siekiant suteikti aiškumo apie realizuotus simuliacijos įrankio prototipo komponentus, šiame skyriuje pateikiami tik esminių komponentų realizacijos aprašymai su pagrindinėmis klasių diagramomis. Šis aprašymas parodo esminių komponentų vidinę struktūrą, veikimo principą ir realizacijos pagrindimą, todėl tai gali padėti ateitiems kartoms tobulinti sukurto prototipą.

4.3.1. Taisyklių mašina

Taisyklių mašina buvo realizuota grupinio darbo metu. Realizacijos metu buvo remiamasi universalumo ir plečiamumo idėja tuo tikslu, jog ateityje šios mašinos funkcionalumas galėtų būti plečiamas ir naudojamas ne tik šio simuliacijos įrankio prototipo kontekste, bet ir integruojant į PĮ kaip priemonę atskirti nuo jos taisyklės arba reikšmingą kodą. Todėl realizuojant komponentą buvo remiamasi objektinio projektavimo šablonais (angl. *design patterns*) ir objektinio programavimo paradigmos principais, kai paveldint tiek abstrakčias, tiek įprastas klases arba realizuojant interfeisus galima plėsti funkcionalumą nepažeidžiant esamo. Taip pat toks principas leidžia plėsti funkcionalumą, nežinant kitų esančių komponentų veikimo principų.

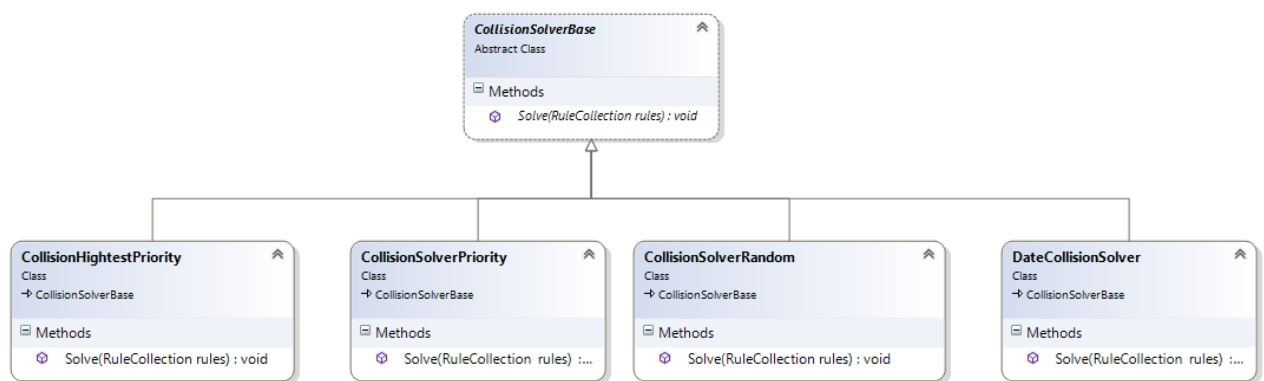
4.3.1.1. Valdymo komponentas

Valdymo komponentas yra pagrindinis sukurto taisyklių mašinos komponentas. Šis komponentas, apjungiant visus kitus taisyklių mašinai priklausančius komponentus, geba valdyti visą

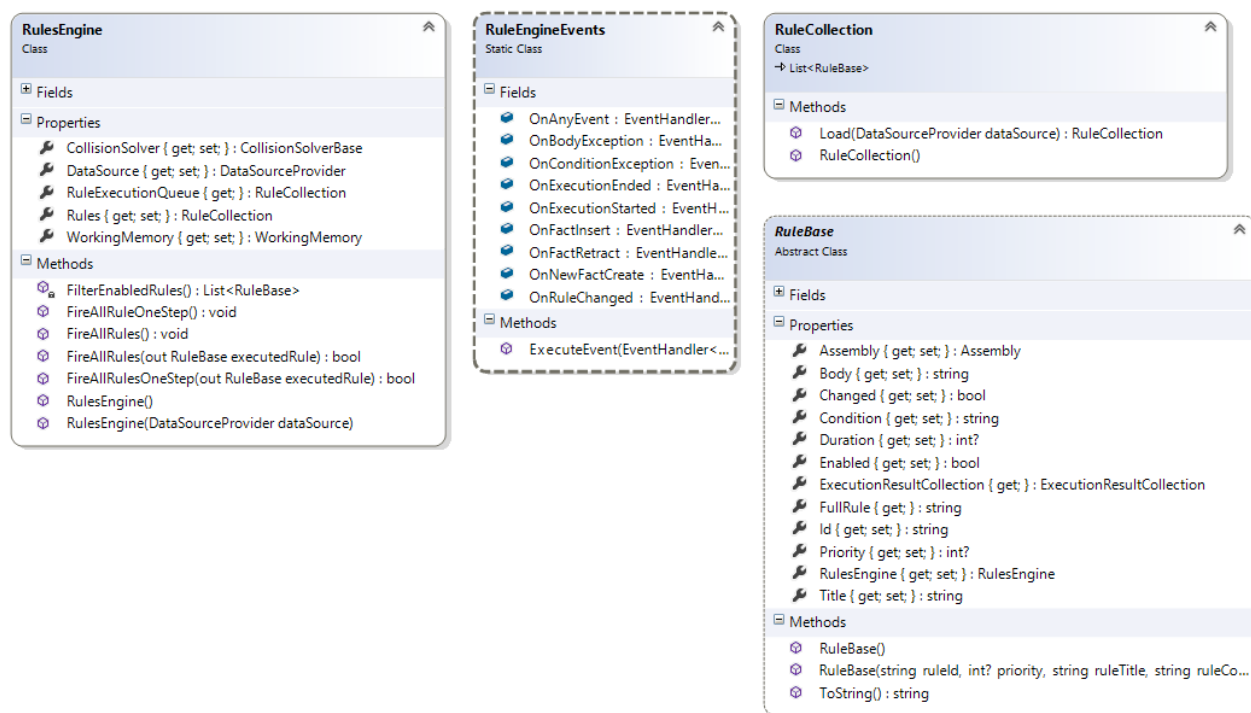
darbą ir vykdyti taisykles. Valdymo komponentas susideda iš dviejų dalių – kolizijų sprendimo (klasių diagrama pateikta 24 pav.) mechanizmo ir darbo valdymo komponentų (25 pav.). Kolizijų sprendimas yra realizuotas taip, kad būtų galima panaudoti skirtingus kolizijų sprendimo būdus, panaudojant kolizijų sprendimo abstrakčios klasės realizaciją:

- CollisionHightestPriority – kolizijos sprendžiamos pasirenkant taisyklę su aukščiausiu prioritetu;
- CollisionSolverPriority – kolizijos sprendžiamos sudarant surūšiuotą taisyklių sąrašą prioriteto didėjimo tvarka;
- CollisionSolverRandom – kolizijos sprendžiamos sudėliojant taisyklių tvarką atsitiktine tvarka;
- DateCollisionSolver – kolizijos sprendžiamos išrenkant naujausią taisyklę.

Žemiau pateikiamos klasių diagramos parodo realizuotų komponentų vidinę struktūrą su esančiais metodais.



24 pav. Kolizijų sprendimo realizacijos klasių diagrama



25 pav. Valdymo komponentų klasių diagrama

Valdymo komponente yra realizuotas įvykių aptikimo ir apdorojimo mechanizmas. Reaguojant į įvykius, jas galima apdoroti dvejopai: į sistemą įvesti į įvykius reaguojančias taisykles, kurias patenkinus bus įvykdomos pajungiamomis funkcijomis iš ją naudojančių PI. Tai leidžia realizuoti klaidų aptikimo, derinimo, nenumatytų atvejų apdorojimo mechanizmą ir dar labiau išplėsti taisyklių mašinos lankstumą. Taip pat valdymo komponentas leidžia vykdyti taisykles ir taisyklių rinkinį pažingsniui (*RulesEngine* klasės metodas *FireAllRulesOneStep*).

Realizuoti įvykių tipai gali būti dvejopi: į taisyklių mašinos darbą reaguojantys įvykiai ir į paties taisyklių vykdymą orientuoti įvykiai. Mašinos darbo metu sugeneruojami įvykiai ir realizuota galimybė jas aptinkant apdoroti:

- *OnConditionException* – įvyksta tada, kai nepavyksta patikrinti taisyklės sąlygos arba ji yra tokia, jog galima jos įvykdyti (pavyzdžiui, darbinėje atmintyje nėra reikiamo fakto).
- *OnBodyException* – įvyksta, kai taisyklės kūne aptinkama kritinė klaida (pavyzdžiui, dalyba iš nulio).
- *OnRuleChanged* – įvyksta, kai pakeičiama kuri nors taisyklė.
- *OnAnyEvent* – įvyksta, įvykus bet kokiai kritinei klaidai, dėl kurios negalima įvykdyti taisyklės.

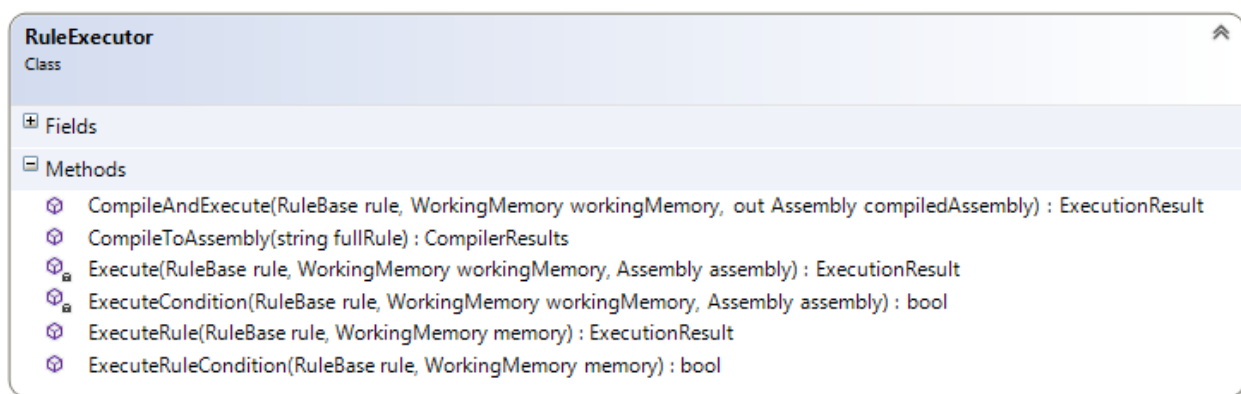
Į taisyklių vykdymą orientuoti įvykiai ir galimybė jas aptinkant apdoroti:

- *OnNewFactCreate* – įvyksta, kai kuri nors taisyklė sukuria naują faktą darbinėje atmintyje;

- *OnFactInsert* – įvyksta, kai naujas faktas yra pridedamas į darbinę atmintį arba taisyklių mašiną naudojančioji PĮ (šiuo atveju – simuliacijos komponentas) prideda naują faktą.
- *OnFactRetract* – kai esamas faktas išimamas iš darbinės atminties.

4.3.1.2. Vykdymo ir kompiliatoriaus komponentai

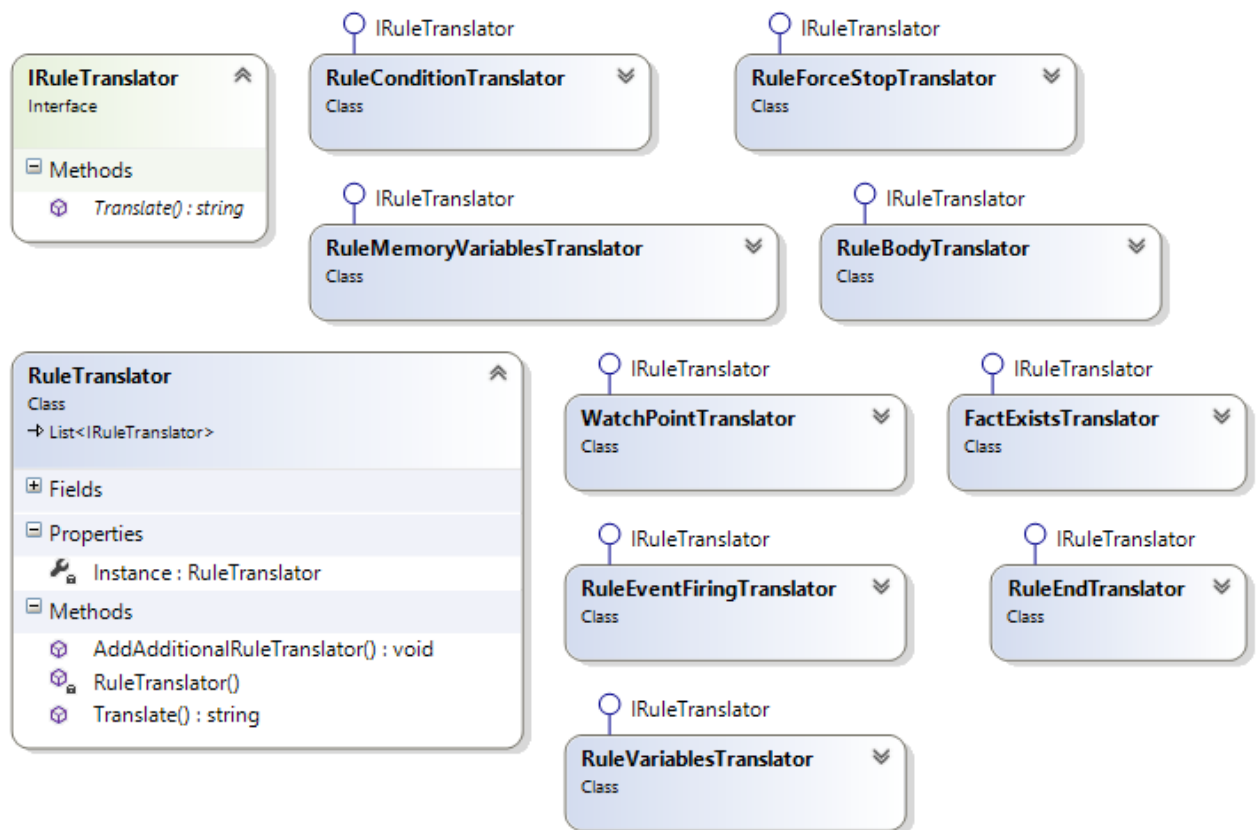
3.3.2 skyriuje pateikiamoje taisyklių mašinos architektūros diagramoje kompiliatorius yra išskirtas kaip atskiras komponentas, tačiau realizacijos lygmenyje kompiliatoriaus vaidmenį atlieka *Visual Basic* kalbos kompiliatorius. Tačiau pabrėžiant taisyklių mašinos veikimo principą tai yra svarbi dalis. Kompiliatorius šios mašinos kontekste yra skirtas kompiliuoti sutransliuotos taisyklės kodą (apie transliatorių žr. detaliau skyrių 4.3.1.3). Sukompiliuota į vykdomąjį komponentą, taisyklės sąlyga gali būti patikrinta naudojant darbinę atmintį. Taip pat jeigu patikrinus visų taisyklių sąlygas yra išrenkama viena taisyklė, gali būti vykdomas su taisykle susietas veiksmas. Šio komponento vidinė struktūra su metodais pateikiamas 26 pav.



26 pav. Vykdymo komponento klasių diagrama

4.3.1.3. Transliatorius

Šio komponento paskirtis – paversti aukšto lygio kalba užrašytą taisyklę (priedas A.1.) paversti į žemo lygio kompiliuojamą ir vykdomą programos kodą (priedas A.2.). Aukšto lygio kalba užrašyta VT – tai taisyklė, kuri savo sintakse yra panaši į Microsoft™ Visual Basic programavimo kalbą (žr. skyrių 4.2.2).



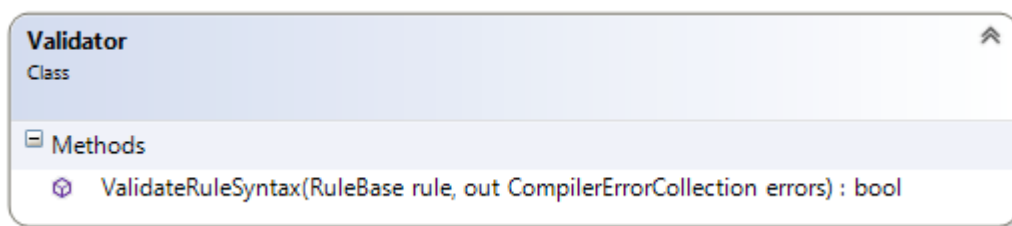
27 pav. Transliatoriaus komponento klasių diagrama

Pagal pateikiamą klasių diagramą (27 pav.) galime matyti, jog aukšto lygio taisyklės transliacija į programos kodą yra išskaidyta pagal atskirą logiką į klases – pavyzdžiui: taisyklės sąlygos transliatorius, taisyklės veiksmo transliatorius, taisyklės darbinės atminties transliatorius, taisyklės įvykių apdorojimo valdymo transliatorius. Tokiu būdu buvo realizuotas lengvai ir greitai plečiamas sprendimas, kai atsiradus poreikiui šio modulio veikimą galima išplėsti įvedant papildomas klases realizuojant transliatoriaus interfeisą.

4.3.1.4. Sintaksės tinkamumo vertintojas

Tai atskiras komponentas, kuris nekeičiant darbinės atminties būsenos ir nelemiant pačios taisyklių mašinos darbo, galintis patikrinti taisyklių mašinoje įvedamų arba keičiamų taisyklių struktūrą ir sintaksę. O tai yra svarbu, nes taisyklių mašina turi užtikrinti, kad į ją nebus įvestos taisyklės, kurias neįmanoma įvykdyti. Taisyklių sintaksės tikrinimo metu, aukšto lygio VT yra sutransliuojama į kompiliuojamą kalbą, kuri yra sukompiliuojama ir bandoma vykdyti. Tačiau ji vykdoma izoliuotoje aplinkoje ir esamos darbinės atminties nelemia. Šis komponentas yra aktualus vartotojui, kuris dirba su sistema, todėl jis taip pat geba pasakyti kurioje taisyklės eilutėse yra klaidos ir tuo pačiu padeda su ja dirbančiam vartotojui išspręsti taisyklės sintaksės problemas.

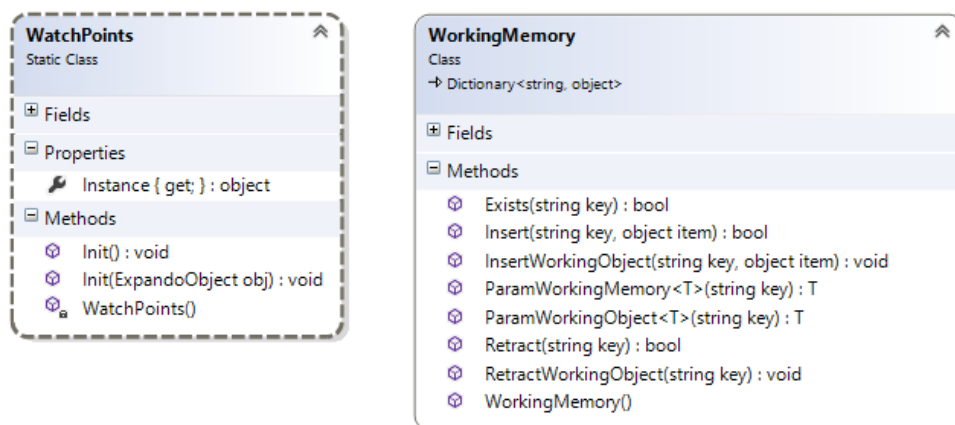
Svarbu pabrėžti tai, jog šis komponentas negali patikrinti prieštaringumą ar taisyklės logiką – tai yra visiškai atskira mokslinių tyrinėjimų sritis ir šio darbo kontekste ši problema nėra nagrinėjama.



28 pav. Sintaksės tinkamumo vertinimo komponento klasių diagrama

4.3.1.5. Darbinė atmintis

Realizacijos lygmenyje (29 pav.), darbinė atmintis – tai programos klasių egzempliorių sąrašas, kurių tipai yra neapibrėžti. Dėl to darbinėje atmintyje gali būti saugomi bet kokio tipo faktai. Realizuotas sprendimas palaiko abipusį ryšį tarp taisyklių mašinos ir ją naudojančios PĮ, todėl tiek PĮ, tiek taisyklių mašina gali įterpti ir keisti darbinės atminties faktus. Kadangi dažnai yra poreikis turėti galimybę tiek integruojant taisyklių mašiną su PĮ, tiek panaudojant jį simuliacijoje palaikyti daugiaprocesoriškumą, todėl darbinė atmintis palaiko naudojamų faktų resursų sinchronizaciją. Todėl tokiu būdu yra užtikrinama, jog vienam procesui taisyklės vykdymo metu naudojant tam tikrą darbinės atminties faktą, faktas nebus pakeistas kito proceso ir dėl to nebus iššaukta resursų konflikto problema.

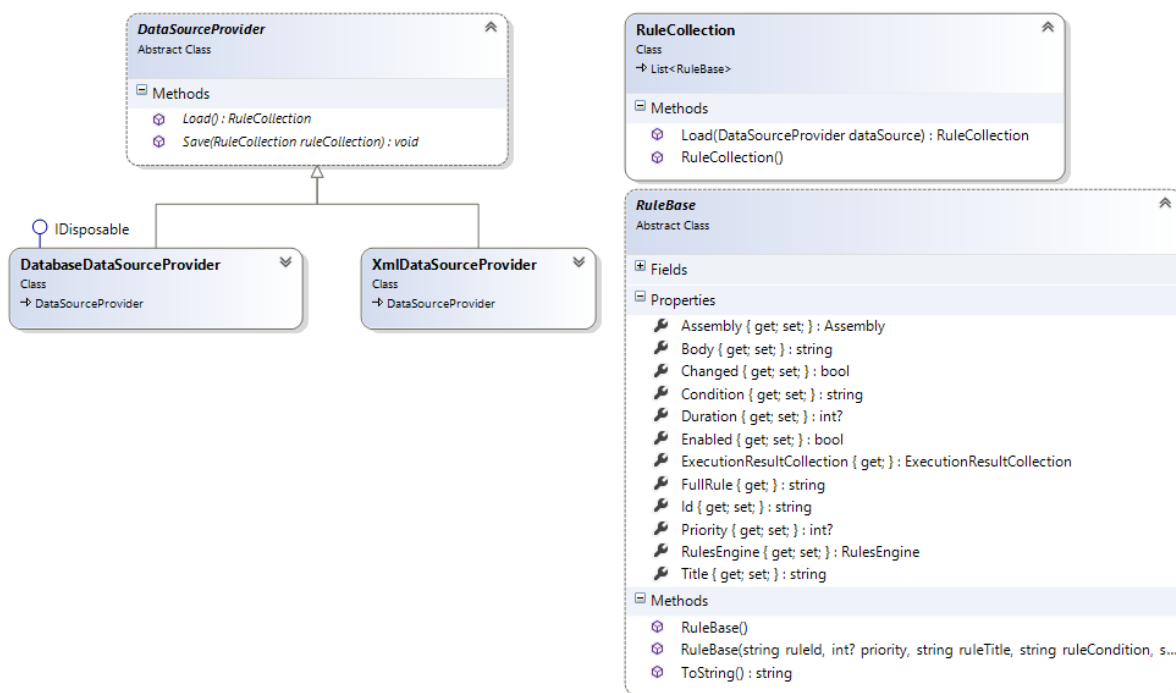


29 pav. Darbinės atminties klasių diagrama

4.3.1.6. Taisyklių saugykla

Taisyklių saugyklos realizacijos metu buvo numatyta, jog taisyklių saugykla gali būti bet kokio tipo saugykla. Kitą taisyklių saugyklą galima prijungti realizuojant abstrakčią klasę *DataSourceProvider* ir tokiu būdu taisyklių mašina gali prisirišti prie bet kokio tipo išorinės saugyklos (tai gali būti duomenų bazė, žiniatinklio paslauga ir pan.). Todėl panaudojant pasirinktą

taisyklių šaltinio realizaciją (klasių diagramoje 30 pav. žymimas kaip *DatabaseDataSourceProvider* ir *XmlDataSourceProvider*), taisyklės yra pakraunamos į *RuleCollection* sąrašą atmintyje, kurioje saugomi *RuleBase* tipo taisyklių objektų egzemplioriai. Simuliacijos įrankio prototipe yra naudojama XML (angl. *Extensible Markup Language*) kalbos taisyklių saugykla. Saugomos taisyklės XML formatu pavyzdys pateiktas priede A.3.



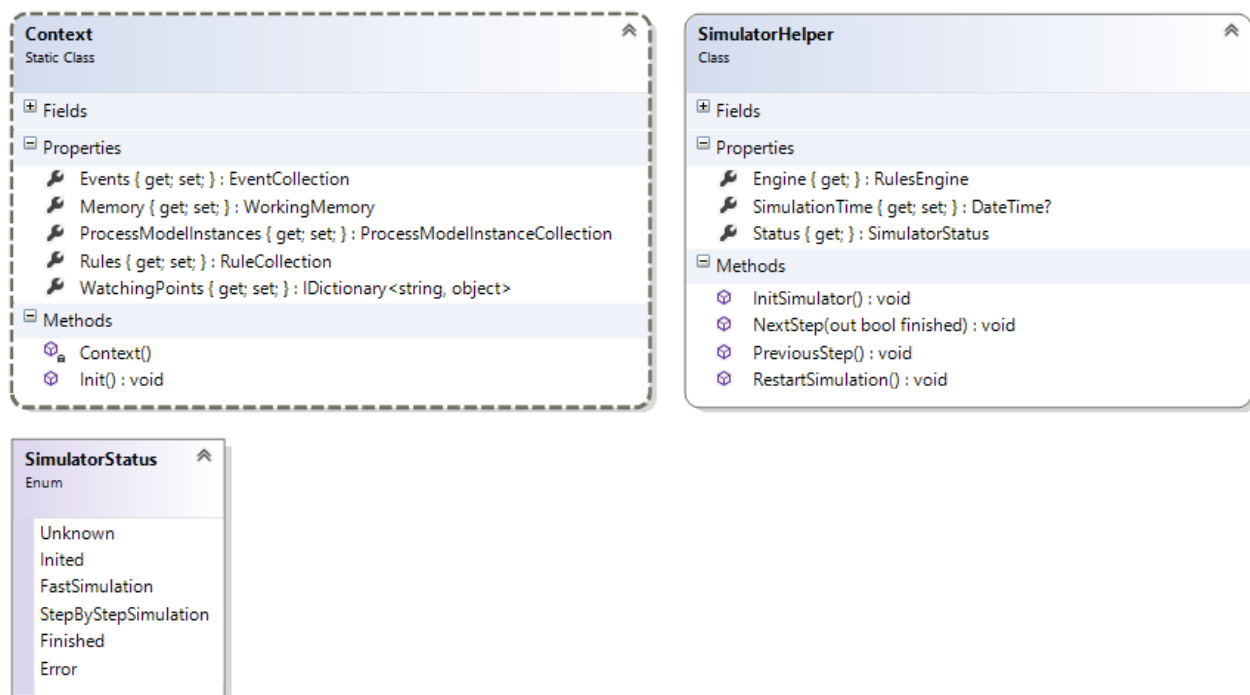
30 pav. Taisyklių saugyklos klasių diagrama

4.3.2. Simuliacijos valdymo komponentas

Simuliacijos valdymo komponento paskirtis valdyti simuliaciją ir saugoti simuliacijos vykdymo metu kiekvieno įvykdyto proceso egzemplioriaus aktualius rezultatus.

4.3.2.1. Simuliacijos valdymas

Tai simuliacijos valdymo komponento branduolys. Simuliacijos valdymo komponento pagrindas – simuliacijos laikas ir simuliacijos vykdymo eigos kontrolė. *Context* klasė realizuoja simuliacijos kontekstą (31 pav.), kurioje saugomi visa simuliacijai reikalinga informacija. *SimulatorHelper* realizacija suteikia galimybę vykdyti simuliaciją pažingsniui, valdyti, stebėti simuliacijos būseną ir laiką.



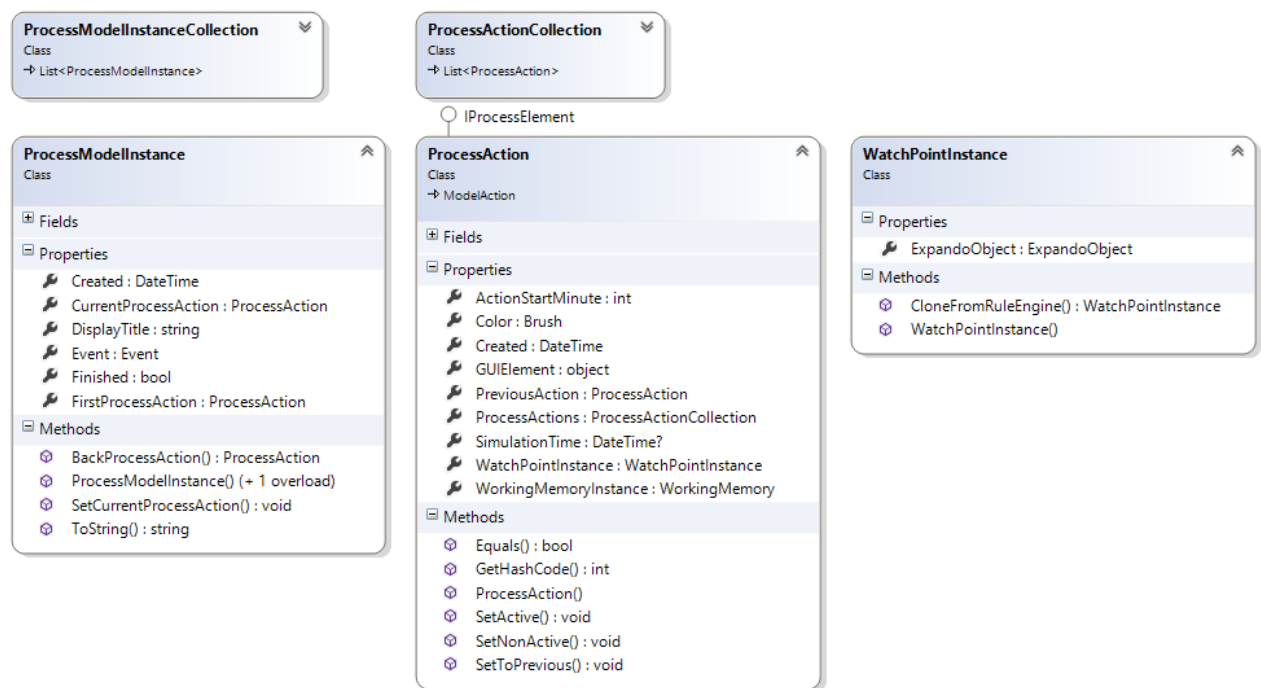
31 pav. Simuliacijos valdymo komponentų klasių diagrama

4.3.2.2. Procesų egzempliorių saugykla ir simuliacijos rezultatų duomenų apdorojimo komponentai

Procesų simuliacijos metu yra reikalinga fiksuoti kiekvieno proceso egzemplioriaus vykdymo rezultatą. Šio simuliacijos įrankio prototipo kontekste, aktuali procesų vykdymo rezultatai yra laikomi stebėjimo taškų duomenys, kuri yra realizuota taisyklių mašinoje. Todėl realizacijos lygmenyje (32 pav.), siekiant simuliaciją vykdančiam vartotojui suteikti kuo daugiau informacijos apie simuliaciją, kiekvienas proceso egzempliorius (realizacijos klasė *ProcessModelInstance*) fiksuoja informaciją apie:

- Proceso egzempliorių inicijavusio įvykio informacija;
- Įvykdytų veiksmus ir jų grandinę;
- Po kiekvieno veiksmo įvykdymo aktualių simuliacijos stebėjimo taškų būseną;
- Kiekvieno įvykdyto veiksmo darbinės atminties kopija.

Aktualių stebėjimo taškų informacija yra naudojama analizuojant simuliacijos rezultatus ir remiantis jai pateikiami simuliacijos ataskaitos diagramos.



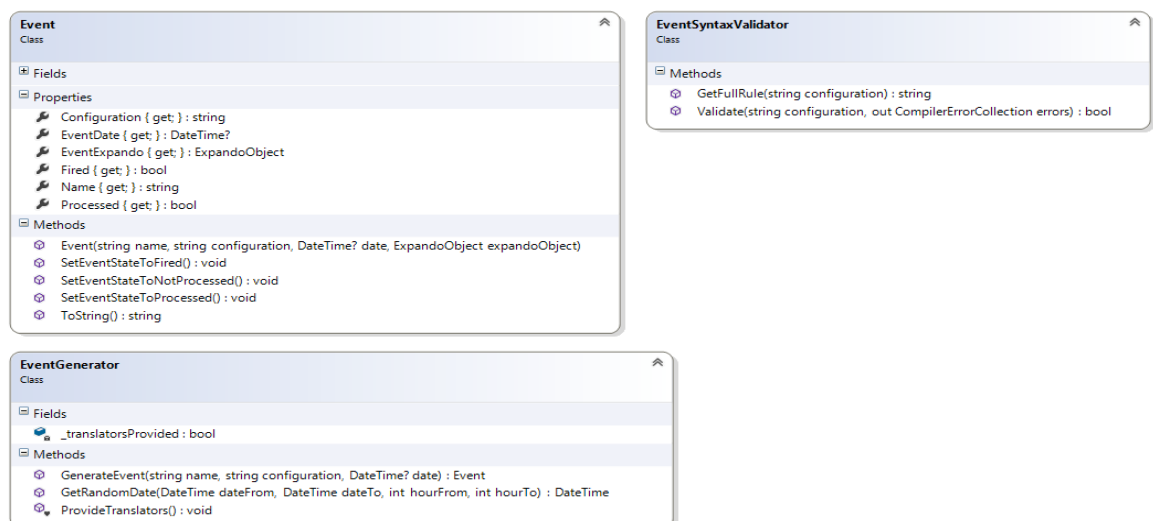
32 pav. Simuliacijos rezultatų duomenų apdorojimo komponento klasių diagrama

4.3.3. Įvykių valdymo mašina

Įvykių valdymo mašinos paskirtis suteikti galimybę generuoti įvairaus tipo įvykius, kurie savyje gali laikyti generavimo metu generavimo taisyklėse nurodytą informaciją. Todėl kiekvienas įvykis turi tam tikrą informaciją, kuri gali būti apdorojama simuliacijos metu.

4.3.3.1. Įvykių generatorius

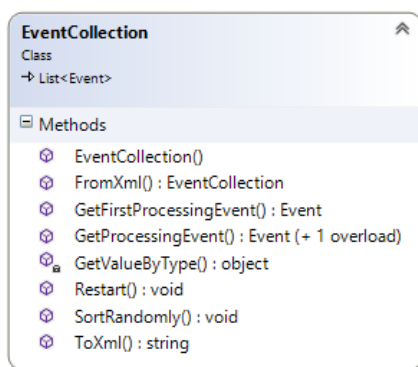
Įvykių generatoriaus paskirtis generuoti įvykius pagal nurodytas įvykių generavimo taisykles (33 pav.). Įvykių generavimo taisyklės Įvykiams generuoti yra naujama taisyklių mašina.



33 pav. Įvykių generatoriaus komponento klasių diagrama

4.3.3.2. Įvykių saugykla

Sugeneruoti įvykiai saugomi įvykių saugykloje. Įvykių saugykloje esantys įvykių duomenys gali būti pateikiami simuliacijos valdymo komponentui pagal jo užklausimą. Taip saugykla leidžia išsaugoti įvykius į failą tuo tikslu, kad ateityje būtų galima juos vėl užkrauti į simuliacijos prototipą ir atlikti simuliaciją negeneruojant įvykių iš naujo.



34 pav. Įvykių saugyklos komponento klasių diagrama

4.4. Vartotojo sąsaja

Siekiant supaprastinti simuliacijos įrankio naudojimą, vartotojo sąsaja padalinta į 4 dalis:

- Įvykių valdymo (priedas D.1.);
- Taisyklių valdymo – (priedas D.2.);
- Simuliacijos valdymo (priedas D.3.);
- Simuliacijos rezultatų peržiūros (priedas D.4.).

Grafinė vartotojo sąsaja buvo realizuota naudojant *Windows Presentation Foundation* karkasu ir MVC (angl. *Model-View-Controller*), MVVM (angl. *Model View – View-Model*) programavimo principais, kurie leidžia supaprastintai valdyti ir pateikti vartotojo sąsają. Grafinės vartotojo sąsajos komponentas detaliau nėra aprašomas, nes tai nėra esminė simuliacijos įrankio dalis, o tik informacijos ir darbo su įrankiu pateikimo būdas.

4.5. Apibendrinimas

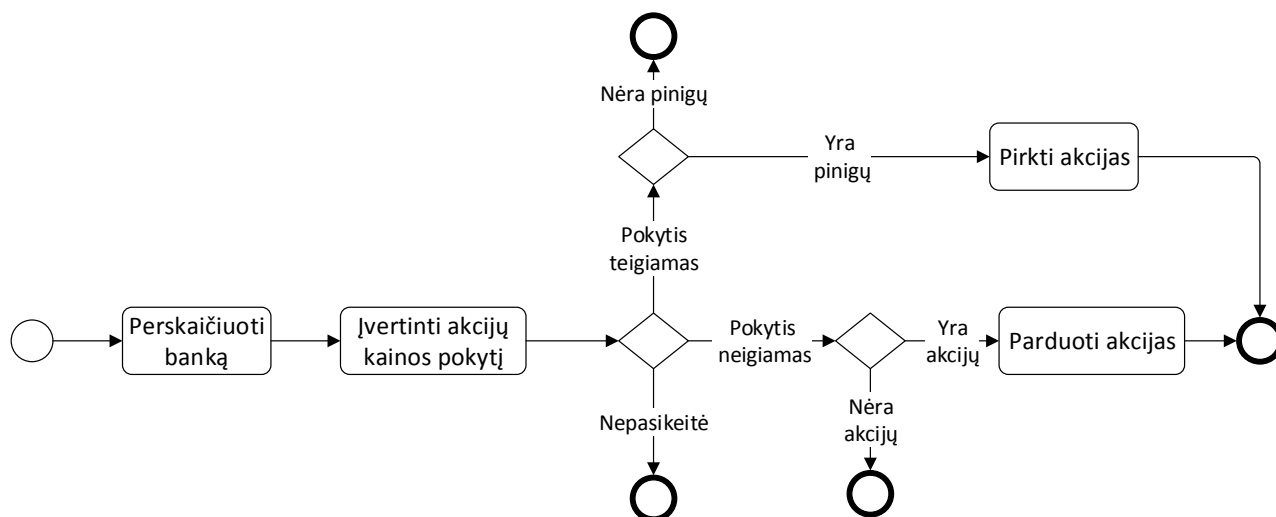
Siūlomo metodo ir simuliacijos įrankio prototipo realizacija parodė, jog panaudojant šiuolaikinės priemonės ir programavimo technologijas galima realizuoti universalią priemonę, leidžianti simuliuoti įprastas ir dinamines VP. Kadangi realizavus taisyklių mašiną, nebuvo susidurta su problemomis realizuojant simuliacijos įrankio prototipą, taisyklių mašina gali būti panaudojama kaip priemonė atskirti verslo taisykles nuo programos logikos ir ją naudojant dinamiškai lemti programos veikimą.

5. EKSPERIMENTINIS TYRIMAS

Eksperimentinio tyrimo tikslas – išsiaiškinti skirtumus ir privalumus tarp taisyklėmis grindžiamų VP simuliacijos, kai vykdymo metu taisyklės yra statinės ir dinaminės ir pagrįsti metodo naudingumą. Todėl eksperimento metu atliekami tyrimai vykdant simuliaciją nuo pradžios iki galo su fiksuotu taisyklių rinkiniu ir dinaminės simuliacijos, kai vykdant simuliaciją taisyklės ir jų rinkiniai yra keičiami jos vykdymo metu.

5.1. Verslo proceso modelis

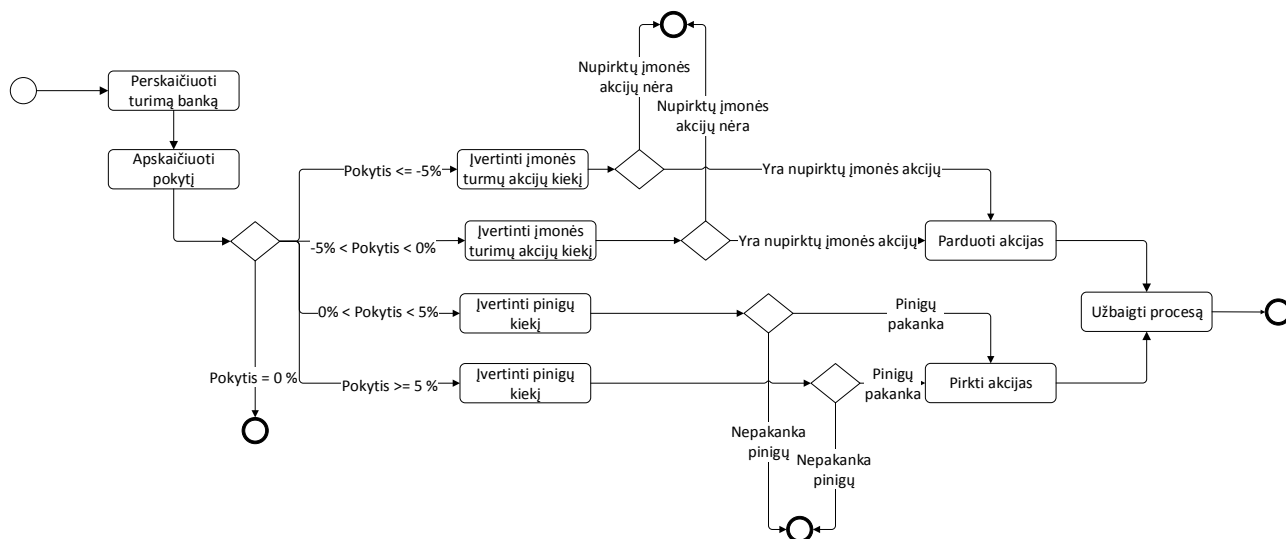
Simuliacijai pasirinktas akcijų biržos prekybos akcijomis simuliacijos procesas (35 pav.), kai turint fiksuotą pradinę banką atsižvelgiant į taisykles pagal įvykius (įvykiai šiuo atžvilgiu yra įmonių akcijų vertės pokyčiai), siekiant gauti kuo didesnę pelną, akcijos yra perkamos arba parduodamos.



35 pav. Akcijų biržos prekybos verslo proceso modelis

5.2. Simuliacinis modelis

VP modelis yra transformuojamas į simuliacinį modelį, kuris yra aprašytas taisyklėmis (priedas C.1.). Simuliacinio modelio veiklos atitinka priede pateiktas taisykles. Transformuojant VP modelį į simuliacinį, VP modelis yra išplečiamas ir papildomas simulacijai reikalingais duomenimis ir informacija (36 pav.).



36 pav. Akcijų biržos prekybos simuliacinis modelis

5.3. Eksperimentinis tyrimas

5.3.1. Įvykiai simuliacijai

Simuliuojant VP, įvykiai, kurie inicijuoja VP, yra akcijų biržoje įmonių akcijų vertės pokyčiai. Akcijų biržos simuliacijoje naudojami 4 skirtingų įmonių akcijų pokyčio įvykiai, pavadinti atitinkamai ĮmonėA, ĮmonėB, ĮmonėC, ĮmonėD. Naudojami akcijų vertės kitimų įvykiai nėra atitinkantys realaus pasaulio šių įmonių akcijų vertės pokyčiams akcijų biržose – tai tik simuliacijai sugeneruoti duomenys.

Įvykiai simuliacijai sugeneruojami naudojant sukurtą simuliacijos priemonę ir naudojant priede B aprašytą įvykių generavimo taisyklę. Sugeneruoti simuliacijos įvykiai, kurie buvo panaudoti simuliacijai vykdyti yra išsaugoti įvykių faile, kuris pateiktas kompaktiniame diske (failas Experiment-Files\Events.NeutronEvents).

5.3.2. ECA taisyklės simuliacijai

Simuliacijos eksperimentas atliekamas su trimis taisyklių rinkiniais:

- R1 – pradinis taisyklių rinkinys (priedas C.1.).
- R2 – pakeistas taisyklių rinkinys, kurio taisyklės esančios pradiname R1 rinkinyje yra pakeistos R2 rinkiniu (priedas C.2.). Šiame taisyklių rinkinyje taisyklių skaičius nėra keičiamas, joje yra pakeisti tik taisyklių rinkinio R1 taisyklių turinys – sąlygos ir konstantos.
- R3 – pakeistas taisyklių rinkinys, kurio taisyklės esančios pradiname R1 rinkinyje yra pakeistos R3 rinkiniu (priedas C.3.). Šiame taisyklių rinkinyje yra pakeisti esamų R1 rinkinio taisyklių logika, ir pridėtos naujos taisyklės.

Taisyklių rinkinių failai taip pat pateiktas kompaktiniame diske (aplankalas Experiment-Files).

5.3.3. Eksperimento eiga

Simuliacijos vykdymo eiga aprašomi žingsniais, po kiekvieno fiksuojant jos rezultatus:

- Įprastos verslo proceso simuliacijos:
 1. Atliekama simuliacija naudojant taisyklių rinkinį R1.
 2. Atliekama simuliacija naudojant taisyklių rinkinį R2.
 3. Atliekama simuliacija naudojant taisyklių rinkinį R3.
- Dinaminės verslo proceso simuliacijos:
 4. Atliekama simuliacija naudojant taisyklių rinkinį R1, jos vykdymo eigoje sustabdome simuliaciją ir taisyklių rinkinį R1 keičiame R2 rinkiniu.
 5. Atliekama simuliacija naudojant taisyklių rinkinį R1, jos vykdymo eigoje sustabdome simuliaciją ir taisyklių rinkinį R1 keičiame R3 rinkiniu.

5.3.4. Simuliacijos rezultatų vertinimo kriterijai

Natūralu, jog vykdant akcijų biržos veiklos simuliaciją, kai atsižvelgiant į vertės pokyčius yra perkamos arba parduodamos akcijos, svarbiausias kriterijus yra gautas pelnas ir bendra turimų akcijų vertė. Vertinant simuliuojamą procesą, pagrindiniai kriterijai yra:

- Turimų grynąjų pinigų kiekis;
- Turimų akcijų vertė;
- Bankas - (turimų grynų pinigų kiekis sudėjus turimų akcijų vertę);
- Turimų nupirktų kiekvienos įmonių akcijų skaičius.

5.3.5. Simuliacijos pradiniai duomenys

Simuliacijos pradiniai duomenys įvedami panaudos taisyklę „Inicijuoti simuliaciją“ (priedas C.1.1.). Pradinis grynąjų pinigų kiekis yra 10 000 vienetų neapibrėžtos valiutos (valiuta šiame simuliacijos kontekste neturi prasmės).

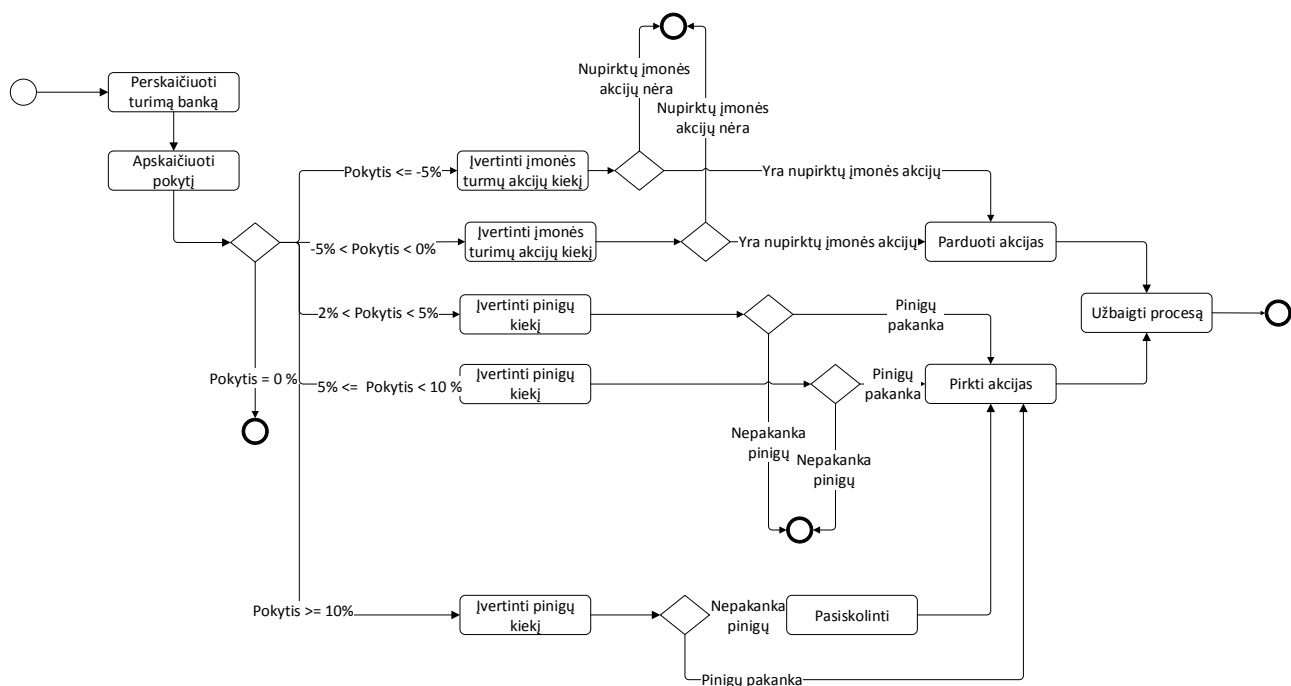
5.4. Eksperimento rezultatai

Žemiau pateikiamas eksperimentinio tyrimo rezultatų lentelė. Geltonai išskirta eilutė parodo pagrindinį simuliacijos vertinimo kriterijaus – banko rezultatą.

5 lentelė. Simuliacijos rezultatai

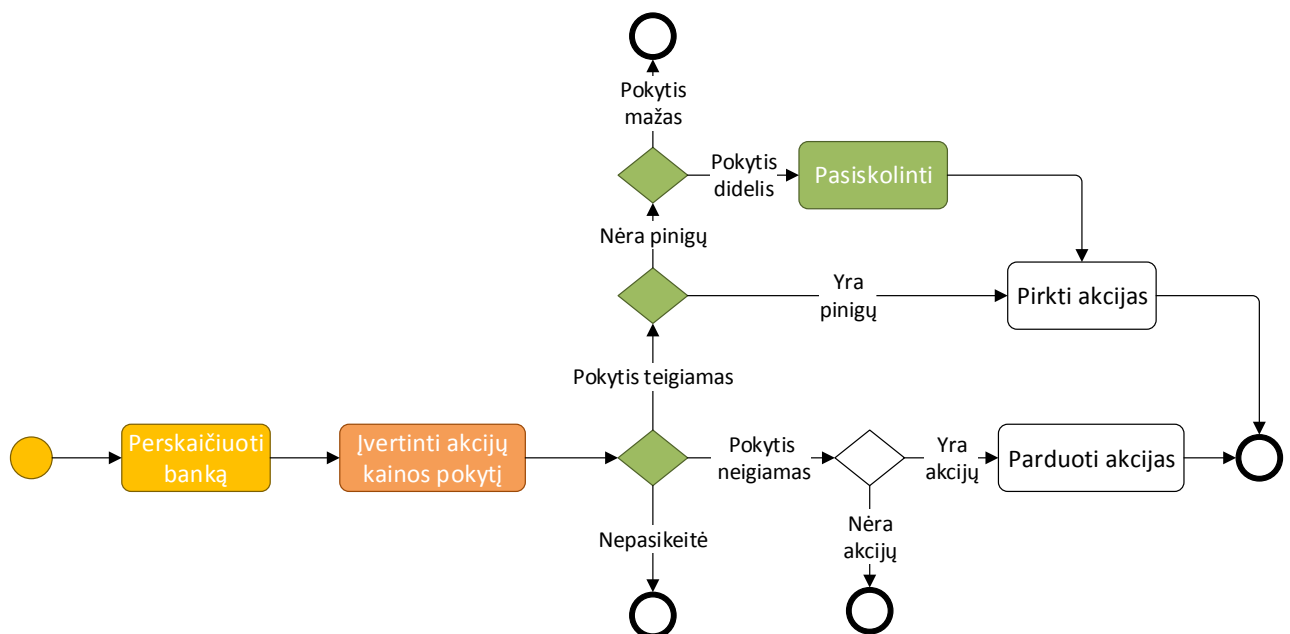
	Įprasta simuliacija Nr. 1	Įprasta simuliacija Nr. 2	Įprasta simuliacija Nr. 3	Dinaminė simuliacija Nr. 4	Dinaminė simuliacija Nr. 5
Grynieji pinigai	262	64	12519	228	14701
Akcijų vertė	10465	11725	226595	11725	225900
Bankas	10727	11789	26364	11953	27151
A įmon. akcijos	0	100	1150	100	1200
B įmon. akcijos	200	200	3400	200	3500
C įmon. akcijos	100	50	3950	50	3900
D įmon. akcijos	150	100	212750	100	213450

Kadangi atliekant proceso simuliaciją galima stebėti jos vykdymą ir duomenų kitimo eigą, kontroliuoti simuliacijos vykdymo greitį arba vykdyti ją pažingsniui, pastebėjus neįprastai didelį pokytį (esant veikloje apskaičiuoti pokytį) ir nepakankant pinigų kiekiams pirkti akcijų – sustabdžius simuliaciją įvedame naują taisyklę, kuri esant dideliems pokyčiams atlieka skolinimosi veiksmą (lentelėje pateikiami duomenys yra jau atskaičiavus skolą) ir pakeičiamos esamos. Simuliacinis modelis po taisyklių pakeitimo pateiktas 37 pav.



37 pav. Simuliacinis VP modelis su pakeistomis taisyklėmis

Tokiu būdu skolinantis galima pasiekti geresnį rezultatą, negu atliekant įprastą simuliaciją 38 pav. pavaizduotas VP modelis ir jo vykdymo kelias, kai vykdant simuliaciją Nr. 5, pakeitus taisyklių rinkinį buvo įterpta nauja taisyklė, kuri savo ruožtu pakeitė VP modelį – įvesta nauja veikla. Oranžine spalva pažymėti elementai parodo vykdymo eigą iki vykdymo sustabdymo. Simuliacija buvo sustabdyta tokiu momentu, kai vykdant simuliaciją pažingsniui buvo pastebėtas didelis, teigiamas akcijų kainos pokytis ir nebuvo grynųjų pinigų akcijoms pirkti. Sustabdžius simuliaciją veikloje „Įvertinti akcijų kainos pokytį“ (pažymėta raudonai) buvo pakoreguotos esamos ir įvestos naujos taisyklės. Po šio veiksmo, pratęsus simuliaciją, jo eiga pažymėta žaliai. Tokiu būdu neturint pinigų akcijoms pirkti, bet žinant kad jų nupirkimas bus naudingas bendram rezultatui, buvo pasiektas didesnis pelnas nupirkus brangstančias akcijas. Todėl galime priėti prie išvados, kad galimybė simuliacijos vykdymo metu, sustabdžius vykdomą procesą ir pakeitus taisyklių būseną, leido pasiekti geresnį rezultatą.



38 pav. VP modelis su įterpta veikla

5.5. Apibendrinimas

Esant nesudėtingiems VP simuliacijos modeliams ir žinant įvykių eigą, dinaminės simuliacijos rezultatus galima nesunkiai nuspėti. Tačiau atliktas eksperimentas parodo, jog šio darbo metu sukurtas simuliacijos prototipas leidžia simuliacijos vykdymo metu dinamiškai konstruoti simuliuojamą procesą ir prototipas suteikia galimybę sustabdyti simuliaciją vykdant proceso egzempliorių ir keisti taisyklių būseną. Tokiu būdu keičiant taisykles, galima ištirti kokia įtaką turės taisyklių būsenos pasikeitimai simuliacijos rezultatams.

Apibendrinant eksperimentinio tyrimo rezultatus, galima teigti, jog dinaminės simuliacijos privalumai yra:

- Dinaminė simuliacija lyginant su įprasta, leidžia gauti skirtingus simuliacijos rezultatus, todėl tokių rezultatų analizė gali padėti ištirti atvejus, kurias sudėtinga ištirti atliekant įprastą simuliaciją.
- Dinaminė simuliacija leidžia taupyti laiką, nes vietoj kelių taisyklių rinkinių atskirų simuliacijų, galima atlikti vieną simuliaciją.
- Dinaminė simuliacija leidžia priartėti prie siekiamo rezultato, nes simuliacijos metu stebint simuliacijos eigą pastebėjus rezultatui nenaudingus nuokrypius, jas galima ištaisyti vykdymo metu keičiant taisykles. Tokiu būdu galima įvertinti periodus, kuriuose reikia naudoti skirtingus taisyklių rinkinius siekiant efektyvesnio procesų vykdymo.
- Žiūrint iš praktinės pusės, kadangi realiame pasaulyje VP gali būti vykdomi ilgą laiko tarpą ir jų vykdymo kaina gali būti didelė, keičiantis poreikiams ir reikalavimams reikia keisti patį vykdomąjį VP, nes jo vykdymą nutraukti, jį pakeisti ir paleisti iš naujo nėra tikslinga dėl jo kainos. Dėl to, turint metodą ir priemonę atlikti tokių VP simuliaciją, galima ištirti kokį poveikį turės vykdomų procesų pakeitimai realiame pasaulyje ir išrinkti geresnį variantą.

Taip pat atliekant eksperimentinį tyrimą buvo pastebėta, jog naudojant sukurtą simuliacijos prototipą, yra galimybė taisyklėse aprašyti veiksmus, kurie keistų kitas taisykles. Tokiu būdu būtų galima tyrinėti VP optimizavimo uždavinius, kai priklausomai nuo rezultatų automatiškai būtų keičiami kitų taisyklių parametrai ir tie pakeitimai iš karto atsispindėtų vykdant simuliaciją. Todėl tai galėtų būti nauja tyrimų sritis.

IŠVADOS

1. Atlikus susijusių mokslinių darbų literatūros analizę, nustatyta, kad dinaminio verslo proceso veiklą vykdymo seka yra konstruojama verslo proceso vykdymo metu. Verslo procesai gali būti modeliuojami dviem būdais - grindžiant grafiniais modeliais ir grindžiant taisyklėmis. Modeliuojant taisyklėmis grindžiamus dinامينius verslo procesus, verslo procesą galima apibrėžti kaip dinaminį ECA tipo taisyklių rinkinį.
2. Atlikus taisyklėmis grindžiamų verslo procesų modeliavimo ir simuliacijos įrankių analizę, nustatyta, jog įrankių gamintojai vis dar mažai dėmesio skiria dinaminų verslo procesų simuliacijos funkcionalumui. Kad atskirti verslo taisykles nuo programų sistemų arba simuliuoti taisyklėmis grindžiamus verslo procesų modelius, yra naudojamos verslo taisyklių valdymo sistemos. Tačiau jos nėra pritaikytos dinamiškai konstruoti vykdomus procesus ir atlikti dinaminų verslo procesų simuliaciją.
3. Remiantis verslo taisyklių valdymo sistemų analizės rezultatais, buvo išskirti du naudojami simuliacijos būdai, kurie neleidžia vykdyti verslo procesus dinamiškai. Todėl remiantis išanalizuotais būdais, buvo pasiūlytas metodas, kuris išplečia esamus būdus ir leidžia dinamiškai vykdymo metu keisti verslo taisykles, kurie lemia vykdomus verslo procesus. Pasiūlytas metodas verslo procesų vykdymo metu kiekvienos veiklos suradimui, kurią reikia įvykdyti, naudoja naujausią taisyklių būseną. Todėl vykdymo metu verslo procesas yra konstruojamas dinamiškai atsižvelgiant į naujausią taisyklių būseną taisyklių saugykloje.
4. Remiantis pasiūlyta dinaminų verslo procesų simuliacijos įrankio prototipo architektūra, buvo realizuota taisyklių mašina ir simuliacijos įrankio prototipas. Realizuota taisyklių mašina yra tinkama panaudoti kaip priemonę atskirti verslo taisykles nuo programų sistemų. Simuliacijos įrankio prototipo realizacijos metu nustatyta, kad šiuolaikinių programavimo technologijų ir priemonių naudojimas leidžia sukurti universalią priemonę taisyklėmis grindžiamų dinaminų verslo procesų simuliacijai atlikti.
5. Naudojant sukurtą dinaminų verslo procesų simuliacijos įrankio prototipą, atliktas eksperimentinis tyrimas parodė dinamiškai konstruojamų verslo procesų privalumus. Dinaminė simuliacija palyginti su įprasta, gali pateikti skirtingus rezultatus ir tokių rezultatų analizė leidžia ištirti ir įvertinti kokį poveikį turės organizacijoje vykdomiems procesams taisyklių būsenos pasikeitimai ir taip išrinkti geresnį taisyklių keitimo variantą. Taip pat dinaminė simuliacija leidžia priartėti prie siekiamų rezultatų ir sutaupyti laiko simuliacijų eksperimentų skaičių atžvilgiu.

SIMULIACIJOS ĮRANKIO FUNKCIONALUMO DIDINIMO GALIMYBĖS IR TYRIMŲ KRYPTYS

Tyrimo metu ir simuliacijos įrankio realizacijos metu pastebėtos kitos tyrimų kryptys ir galimybės išplėsti įrankio funkcionalumą:

- Įrankio grafinės vartotojo sąsajos tobulinimas, pritaikymas patogesniai naudojimui ir lengvesniai vartotojo supratimui;
- Taisyklių skaidymas į grupes ir jų naudojimas pagal paskirtį, pavyzdžiui – sisteminės, verslo, palaikymo ir kita;
- Įrankio simuliacijos optimizacija esant dideliame įvykių kiekiui;
- Ataskaitų generavimo ir pateikimo galimybių išplėtimas pagal įvairius kriterijus;
- Taisyklių kalbos ir sintaksės pakėlimas iš žemesnio lygio kalbos į aukštesnę ir taip supaprastinant taisyklių supratimą naudotojams (pavyzdžiui, įvedant žodyną);
- Naujų saugyklų prijungimas prie taisyklių mašinos (duomenų bazė);
- Verslo procesų modeliavimo galimybės įdiegimas integruojant Microsoft™ Visio įrankį ir jo integracija su simuliacijos varikliu;
- Verslo taisyklių vykdymo atsekamumas ir derinimas;
- Taisyklių versijų valdymo galimybės įdiegimas.

Galimos tolesnės magistro darbo kryptys, remiantis šio darbo tyrimu ir rezultatais:

- Verslo taisyklių neprieštaringumo užtikrinimo tyrimas;
- Dinaminės verslo procesų optimizacijos tyrimas simuliacijos vykdymo metu;
- Taisyklėmis grindžiamų dinaminių verslo procesų modeliavimas.

LITERATŪRA

1. AcuProcess Business Process Modeler [interaktyvus] 2014. [žiūrėta 2014-03-27]. Prieiga per internetą: <http://www.accuprocess.com/documentation/Process_Modeler_DataSheet.pdf>
2. Aguilar-Saven, R. S. 2003. *Business process modelling: Review and framework*, International Journal of Production Economics, Vol. 90 (2), Sweden, p. 129-137.
3. Aharoni, A.; Reinhartz-Berger, I. 2008. *A Domain Engineering Approach for Situational Method Engineering*, ER'08 Proceedings of the 27th International Conference on Conceptual Modeling, Springer-Vegal, Berlin, p. 455-458.
4. Bauer, E. 2009. *The Business Rule Approach*, University of Paderborn, Germany, p. 2-10.
5. Čyras, V. 2008. *Dirbtinis intelektas. Paskaitų konspektas*, Vilniaus universitetas, Matematikos ir informatikos fakultetas, Vilnius, p. 115-117.
6. Drools Documentation [interaktyvus] 2014. [žiūrėta 2014-04-03]. Prieiga per internetą: <<http://docs.jboss.org/drools/release/6.0.1.Final/drools-docs/html/index.html>>
7. Duda, J.; Stawowy, A. 2010. *A possibility of Business Rules application in production planning*, Archives of Foundry Engineering, Vol. 10 (2), Katowice, p. 28-30.
8. Dynamic BPM: From simple to sophisticated [interaktyvus] 2011. [žiūrėta 2014-04-04]. Prieiga per internetą: <http://www.ebizq.net/topics/social_bpm/features/13197.html>
9. Eijndhoven, T.; Iacob, M.; Ponisio, M. 2008. *Achieving business process flexibility with business rules*, 12th International IEEE Enterprise Distributed Object Computing Conference, Auckland, p. 95-97.
10. Eriksson, H.; Penker, M. 1999. *Business Modeling with UML: Business Patterns at work*, John Wiley & Sons, Canada, p. 78
11. Gigch, J. P. 1991. *System design modeling and metamodeling*, Plenum Press, New York, p. 225
12. Graham, I. 2007. *Business Rules Management and Service Oriented Architecture. A pattern language*, John Wiley & Sons, England, p. 101-107.
13. Grašys, R. 2011. *Verslo procesų imitavimas naudojant procesų modeliavimo priemones*, Vilniaus Gedimino Technikos Universitetas, Vilnius, p. 47-49.
14. Halle, B. 2002. *Business Rules Aplied. Building Better Systems Using the Business Rules Approach*, John Wiley & Sons, New York, p. 32-40.
15. Hay, D.; Anderson, K. 2000. *Defining Business Rules – What Are They Really?*, Business Rule Group, p. 4-6.
16. Herbst, H. 1997. *Business Rule-Oriented Conceptual Modeling*, Institute of Information Systems, University of Bern, Bern, p. 173-176.

17. IBM WebSphere ILOG JRules Version 7.0 information center [interaktyvus] 2014. [žiūrėta 2014-04-10]. Prieiga per internetą:
<<http://pic.dhe.ibm.com/infocenter/brjrules/v7r0/index.jsp>>
18. Kontrimas, R.; Bulbenkienė, V. 2005. *Verslo procesų modeliai*, Tarptautinė mokslinė – metodinė KONFERENCIJA, Regioninės problemos: Ekonomika, Vadyba, Technologijos, Vakarų Lietuvos verslo kolegija, Klaipėda, p. 107-109.
19. Kuter, U.; Nau, D. 2004. *Forward-Chaining Planning in Nondeterministic Domains*, Nineteenth National Conference on Artificial Intelligence (AAAI-04), San Jose, California, p. 513-518.
20. Lu, R.; Sadiq, S. 2007. *A Survey of Comparative Business Process Modelling Approaches*, 10th International Conference, Poznan, p. 84-90.
21. Maria, A. 1997. *Introduction to modelling and simulation*, Proceedings of the 1997 Winter Simulation Conference, State University of New York, Binghamton, p. 7-11.
22. Martin, A., Steele, M.; Rettinger, J., et al. 2009. *Dynamic Business Process Management (BPM) from IBM*, IBM Corporation, Somers, NY, p. 2-12.
23. Metamodels and metamodeling languages [interaktyvus] 2014 [žiūrėta 2014-04-06] Prieiga per internetą: <http://users.jyu.fi/~jpt/doc/thesis/ime-3_3.html>
24. Mircea, M.; Andreescu, A. 2009. *Using business rules in business intelligence*, Journal of applied quantitative methods Vol. 4, No. 3, 2009, Romania, p. 384-385.
25. Nuttgens, M.; Feld, T.; Zimmermann, V. 1998. *Business Process Modelling with EPC and UML - Transformation or Integration?*, The Unified Modelling Language – Technical Aspects and Applications Workshop, Physica-Verlag HD, Heidelberg, p. 252-255.
26. Open Souce Business Decision Management System [interaktyvus] 2013. [žiūrėta 2014-04-02]. Prieiga per internetą: <<http://openrules.com/pdf/OpenRulesGettingStarted.pdf>>
27. OpenL Tablets – Developer Guide [interaktyvus] 2014. [žiūrėta 2014-04-03]. Prieiga per internetą: <<http://openl-tablets.sourceforge.net/docs/openl-tablets/5.12.0/OpenL%20Tablets%20-%20Developer%20Guide/index.html>>
28. Paul, J. R.; Hlupic, V.; Giaglis, M. G. 1998. *Simulation modelling of business processes*, Proceedings of the 3rd UK Academy of Information Systems Conference, Oxford, p. 313-316.
29. Ritchie, A.; Balakrishnan, S.; Clark, D., et al. 2011. *Patterns: Integrating WebSphere ILOG JRules with IBM Software*, IBM Redbooks, International Technical Support Organization, Poughkeepsie, NY, p. 16-17.
30. SimProcess User's Manual [interaktyvus] 2012. [žiūrėta 2014-03-27]. Prieiga per internetą: <http://simprocess.com/SP_Documentation/SP49/SPUser.pdf>
31. Simul8 Help center [interaktyvus] 2014. [žiūrėta 2014-03-26]. Prieiga per internetą: <<http://simul8.com/support/help/doku.php>>

32. Smaizys A.; Vasilecas, O. 2009. *Business Rules Based Agile ERP Systems Development*, Journal Informatica, Vol. 20 (3), Amsterdam, p. 439-444.
33. Tarptautinių žodžių žodynas [interaktyvus] 2014. [žiūrėta 2014-03-17]. Prieiga per internetą: <http://www.ukc.ktu.lt/zodynas>
34. Tinsley, J.; Belytschko, T.; Fish, J., et al. 2006. *Revolutionizing Engineering Science through Simulation*, Report of the NSF Blue Ribbon Panel on Simulation-Based Engineering Science, National Science Foundation, Arlington, VA, p. 25-29.
35. Valatkaite, I.; Vasilecas, O. 2005. *On Business Rules Automation: The BR-Centric IS Development Framework*, Advances in Database and Information Systems, 9th East European Conference, Tallin, p. 349-350.
36. Vasilecas, O.; Kalibatienė, D. 2008. *Šiuolaikinės duomenų bazės*, Klaipėdos universiteto leidykla, Klaipėdos universitetas, p. 180-190.
37. WebSphere Business Modeler v.7.0 [interaktyvus] 2014. [žiūrėta 2014-03-27]. Prieiga per internetą: http://www-01.ibm.com/support/knowledgecenter/websphere_ia/com.ibm.ia.wbmodeler_v7/wbmodeler/7.0/Overview/WBPMv70_Modeler_Overview.pdf
38. What is Business Rules Management? [interaktyvus] 2014. [žiūrėta 2014-03-20]. Prieiga per internetą: <http://www-01.ibm.com/software/websphere/products/business-rule-management/whatis/>
39. White S. A. 2004. *Introduction to BPMN*, Object Management Group, Needham, MA, USA, p. 2-9.
40. Wright, I.; Marshall, J. 2003. *The execution kernel of RC++: Rete*, a faster Rete with treat as a special case*, 4th International Conference on Intelligent Games and Simulation, London, p. 36-40.
41. Zeng, L.; Flaxer, D.; Chang, H.; Jeng J. 2002. *PLM_{flow} – Dynamic Business Process Composition and Execution by Rule Inference*, Technologies for E-services: third international workshop, TES, Hong Kong, p. 141-143

PRIEDAI

A priedas

A.1. Realizuotos taisyklių mašinos aukšto lygio taisyklė

Identifikatorius: 5

Pavadinimas: Nuolaidos apskaičiavimas

Pririoritetas: 7

Sąlyga: \$shoppingCart.Sum > 500 And \$shoppingCart.Sum < 1400

Veiksmas:

```
CREATE("discount")  
$discount.Value = $shoppingCart.Sum * 0.1  
$shoppingCart.Total = $shoppingCart.Sum - $discount.Value
```

A.2. Taisyklės, transliuotos iš pateikto 1.1. pavyzdžio į kompiliuojamą vykdymui modulį pavyzdys (pažymėta dalis nurodo taisyklės sąlygos ir veiksmo vietą kode)

```
Imports Neutron.RuleEngine  
Imports Neutron.RuleEngine.Memory  
Imports Neutron.RuleEngine.Execution  
Imports Neutron.RuleEngine.Template  
  
Public Class RuleTemplate : Inherits RuleTemplateBase  
  
    Public Sub New(ByVal memory As WorkingMemory, ByVal rule as RuleBase)  
        MyBase.New(memory, rule)  
    End Sub  
  
    Public Function ExecuteCondition() As Boolean  
        Try  
            Return (MEM("shoppingCart").Sum > 500 And MEM("shoppingCart").Sum <  
1400)  
        Catch ex As Exception  
            Me._ruleResult.ConditionException = ex  
  
            RuleEngineEvents.ExecuteEvent(RuleEngineEvents.OnConditionException,  
Me, RuleEventArgs.Load(True, ex.Message, ex))  
  
        End Try  
        Return False  
    End Function  
  
    Public Function ExecuteBody() As RuleExecutionResult  
        Try  
            CREATE("discount")  
  
            SyncLock MemoryElementProtector.GetLocker("shoppingCart")  
                MEM("discount").Value = MEM("shoppingCart").Sum * 0.1  
  
                MEM("shoppingCart").Total = MEM("shoppingCart").Sum -  
MEM("discount").Value  
            End SyncLock  
        Catch ex As ForceExecutionStopException  
            throw
```

```

        Catch ex As EndExecutionException
            throw
        Catch ex As exception
            Me._ruleResult.BodyException = ex

        RuleEngineEvents.ExecuteEvent(RuleEngineEvents.OnBodyException,
Me, RuleEventArgs.Load(True, ex.Message, ex))
        End Try
        If Me._eventFiringEnabled Then
            If (Not Me._rule.RulesEngine is Nothing) Then
                'Me._rule.RulesEngine.FireAllRules()
            End If
        End If
        Return Me._ruleResult
    End Function

    Public Overrides Function Execute() As RuleExecutionResult

        RuleEngineEvents.ExecuteEvent(RuleEngineEvents.OnExecutionStarted,
Me, RuleEventArgs.Load(False, "Rule" & Me._rule.Id & " execution started",
Nothing))
        If Me.ExecuteCondition() Then
            Me.ExecuteBody()
        End If
        RuleEngineEvents.ExecuteEvent(RuleEngineEvents.OnExecutionEnded,
Me, RuleEventArgs.Load(False, "Rule " & Me._rule.Id & " execution ended",
Nothing))
        Return Me._ruleResult
    End Function
End Class

```

A.3. Taisyklė iš 1.1. pateikto pavyzdžio kuri gali būti saugoma tiek atskiroje byloje, tiek atskiroje duomenų bazėje saugoma XML kalboje

```

<Rule>
  <Id>5</Id>
  <Title>Nuolaidos apskaičiavimas</Title>
  <Priority>7</Priority>
  <Condition>$shoppingCart.Sum > 500 And $shoppingCart.Sum < 1400</Condition>
  <Body>
    CREATE("discount")
    $discount.Value = $shoppingCart.Sum * 0.1
    $shoppingCart.Total = $shoppingCard.Sum - $discount.Value
  </Body>
</Rule>

```

B priedas

B.1. Įvykių generavimo taisyklės pavyzdys, naudojant simuliacijos prototipą

Įvykio pavadinimas: AkcijuVertesPokytis

Datos režis: nuo 2014-01-01 iki 2014-06-01

Laiko režis: nuo 08:00 iki 17:00 val.

Įvykio generavimo kodas:

```
Imone = "ImoneA"  
AkcijuVerte = Random(10.0, 15.0)
```

Įvykių kiekis: 100

C priedas

C.1. Taisyklių rinkinys R1

C.1.1. Taisyklė „Inicijuoti simuliaciją“

```
Identifikatorius: 1  
Pavadinimas: Inicijuoti simuliacija  
Pririoritetas: 1  
Sąlyga: Not Fact.Exists("SimuliacijaInicijuota")  
Veiksmai:  
If Not WatchPoints.Exists("Bank") Then  
    #Bank = 10000  
End If  
If Not WatchPoints.Exists("MyMoney") Then  
    #MyMoney = 10000  
End If  
If Not WatchPoints.Exists("SharesMoney") Then  
    #SharesMoney = 0  
End If  
If Not WatchPoints.Exists("ImoneA") Then  
    #ImoneA= 0  
End If  
If Not WatchPoints.Exists("ImoneB") Then  
    #ImoneB= 0  
End If  
If Not WatchPoints.Exists("ImoneC") Then  
    #ImoneC= 0  
End If  
If Not WatchPoints.Exists("ImoneD") Then  
    #ImoneD= 0  
End If  
If Not Fact.Exists("ImoneA") Then  
    Create("ImoneA")  
    $ImoneA.AkcijuVerte = 11  
End If  
If Not Fact.Exists("ImoneB") Then  
    Create("ImoneB")  
    $ImoneB.AkcijuVerte = 40  
End If  
If Not Fact.Exists("ImoneC") Then  
    Create("ImoneC")  
    $ImoneC.AkcijuVerte = 6
```

```

End If
If Not Fact.Exists("ImoneD") Then
    Create("ImoneD")
    $ImoneD.AkcijuVerte = 17
End If

CREATE("SimuliacijaInicijuota")

```

C.1.2. Taisyklė „Perskaičiuoti turimą banką“

Identifikatorius: 1
Pavadinimas: Perskaičiuoti turimą banką
Prioritetas: 1
Sąlyga: Fact.Exists("AkcijuVertesPokytis")
Veiksmas:
 Dim vertePriesTai = MEM(\$AkcijuVertesPokytis.Imone).AkcijuVerte *
 POINTS(\$AkcijuVertesPokytis.Imone)

 Dim verteDabar = POINTS(\$AkcijuVertesPokytis.Imone) *
 \$AkcijuVertesPokytis.AkcijuVerte

 #SharesMoney = #SharesMoney - vertePriesTai + verteDabar

 #Bank = #MyMoney + #SharesMoney

C.1.3. Taisyklė „Apskaičiuoti pokytį“

Identifikatorius: 2
Pavadinimas: Apskaičiuoti pokytį
Prioritetas: 2
Sąlyga: Fact.Exists("AkcijuVertesPokytis")
Veiksmas:
 CREATE("Pokytis")

 \$Pokytis.Proc = (\$AkcijuVertesPokytis.AkcijuVerte - MEM(
 \$AkcijuVertesPokytis.Imone).AkcijuVerte) /
 \$AkcijuVertesPokytis.AkcijuVerte

C.1.4. Taisyklė „Įvertinti turimų pinigų kiekį“ (1)

Identifikatorius: 3
Pavadinimas: Įvertinti turimų pinigų kiekį
Prioritetas: 3
Sąlyga: \$Pokytis.Proc >= 0.05
Veiksmas:
 Dim pirktiAkcijuVnt = 100
 Dim akcijuKaina = pirktiAkcijuVnt * \$AkcijuVertesPokytis.AkcijuVerte

 If akcijuKaina <= #MyMoney Then

 CREATE("PirktiAkcijas")

 \$PirktiAkcijas.Imone = \$AkcijuVertesPokytis.Imone
 \$PirktiAkcijas.Vnt = pirktiAkcijuVnt

 End If

C.1.5. Taisyklė „Įvertinti turimų pinigų kiekį“ (2)

```
Identifikatorius: 4
Pavadinimas: Įvertinti turimų pinigų kiekį
Pririoritetas: 3
Sąlyga: $Pokytis.Proc > 0 And $Pokytis.Proc < 0.05
Veiksmas:
Dim pirktiAkciuVnt = 50
Dim akcijuKaina = pirktiAkciuVnt * $AkciuVertesPokytis.AkciuVerte

    If akcijuKaina <= #MyMoney Then

        CREATE("PirktiAkcijas")

        $PirktiAkcijas.Imone = $AkciuVertesPokytis.Imone
        $PirktiAkcijas.Vnt = pirktiAkciuVnt

    End If
```

C.1.6. Taisyklė „Įvertinti turimų įmonės akcijų kiekį“ (1)

```
Identifikatorius: 5
Pavadinimas: Įvertinti turimų įmonės akcijų kiekį
Pririoritetas: 3
Sąlyga: $Pokytis.Proc < 0 And $Pokytis.Proc > -0.05
Veiksmas:
Dim parduotiVnt = 100

Dim turimosAkcijos= POINTS($AkciuVertesPokytis.Imone)

If turimosAkcijos >= parduotiVnt Then
    CREATE("ParduotiAkcijas")

    $ParduotiAkcijas.Imone = $AkciuVertesPokytis.Imone
    $ParduotiAkcijas.ParduotiVnt = parduotiVnt
End If
```

C.1.7. Taisyklė „Įvertinti turimų įmonės akcijų kiekį“ (2)

```
Identifikatorius: 5
Pavadinimas: Įvertinti turimų įmonės akcijų kiekį
Pririoritetas: 3
Sąlyga: $Pokytis.Proc <= -0.05
Veiksmas:
Dim parduotiVnt = 100

Dim turimosAkcijos= POINTS($AkciuVertesPokytis.Imone)

If turimosAkcijos >= parduotiVnt Then
    CREATE("ParduotiAkcijas")

    $ParduotiAkcijas.Imone = $AkciuVertesPokytis.Imone
    $ParduotiAkcijas.ParduotiVnt = parduotiVnt
End If
```

C.1.8. Taisyklė „Pirkti akcijas“

Identifikatorius: 6

Pavadinimas: Pirkti akcijas

Pririoritetas: 5

Sąlyga: Fact.Exists("PirktiAkcijas")

Veiksmai:

Dim kaina = \$PirktiAkcijas.Vnt * \$AkcijuVertesPokyttis.AkcijuVerte

#MyMoney = #MyMoney - kaina

#SharesMoney = #SharesMoney + kaina

#Bank = #MyMoney + #SharesMoney

Dim imonesAkcijosViso = POINTS(\$PirktiAkcijas.Imone) + \$PirktiAkcijas.Vnt

POINTS(\$PirktiAkcijas.Imone, imonesAkcijosViso)

MEM(\$PirktiAkcijas.Imone).AkcijuVerte = \$AkcijuVertesPokyttis.AkcijuVerte

C.1.9. Taisyklė „Parduoti akcijas“

Identifikatorius: 7

Pavadinimas: Parduoti akcijas

Pririoritetas: 5

Sąlyga: Fact.Exists("ParduotiAkcijas")

Veiksmai:

Dim parduotiVnt = \$ParduotiAkcijas.ParduotiVnt

If POINTS(\$ParduotiAkcijas.Imone) < parduotiVnt Then

 parduotiVnt = POINTS(\$ParduotiAkcijas.Imone)

End If

Dim parduodamaSuma = parduotiVnt * \$AkcijuVertesPokyttis.AkcijuVerte

#MyMoney = #MyMoney + parduodamaSuma

If #SharesMoney > parduodamaSuma Then

 #SharesMoney = #SharesMoney - parduodamaSuma

Else

 #SharesMoney = 0

End If

#Bank = #MyMoney + #SharesMoney

Dim isvisoImonesAkciju = POINTS(\$ParduotiAkcijas.Imone) - parduotiVnt

POINTS(\$ParduotiAkcijas.Imone, isvisoImonesAkciju)

MEM(\$ParduotiAkcijas.Imone).AkcijuVerte = \$AkcijuVertesPokyttis.AkcijuVerte

C.1.10. Taisyklė „Užbaigti procesą“

Identifikatorius: 99

Pavadinimas: Užbaigti procesą

Pririoritetas: 99

Sąlyga: true

Veiksmai:

```
MEM($AkciuVertesPokytis.Imone).AkciuVerte =  
$AkciuVertesPokytis.AkciuVerte
```

```
RETRACT("Pokytis")
```

```
RETRACT("AkciuVertesPokytis")
```

```
RETRACT("PirktiAkcijas")
```

```
RETRACT("ParduotiAkcijas")
```

C.2. Taisyklių rinkinys R2 (tik R1 rinkinio pakeistos taisyklės)

C.2.1. Taisyklė „Įvertinti turimų pinigų kiekį“ (1)

Identifikatorius: 3

Pavadinimas: Įvertinti turimų pinigų kiekį

Prioritetas: 3

Sąlyga: \$Pokytis.Proc >= 0.07

Veiksmai:

```
Dim pirktiAkciuVnt = 150
```

```
Dim akciuKaina = pirktiAkciuVnt * $AkciuVertesPokytis.AkciuVerte
```

```
    If akciuKaina <= #MyMoney Then
```

```
        CREATE("PirktiAkcijas")
```

```
        $PirktiAkcijas.Imone = $AkciuVertesPokytis.Imone
```

```
        $PirktiAkcijas.Vnt = pirktiAkciuVnt
```

```
    End If
```

C.2.2 Taisyklė „Įvertinti turimų pinigų kiekį“ (2)

Identifikatorius: 4

Pavadinimas: Įvertinti turimų pinigų kiekį

Prioritetas: 3

Sąlyga: \$Pokytis.Proc > 0.01 And \$Pokytis.Proc < 0.07

Veiksmai:

```
Dim pirktiAkciuVnt = 100
```

```
Dim akciuKaina = pirktiAkciuVnt * $AkciuVertesPokytis.AkciuVerte
```

```
    If akciuKaina <= #MyMoney Then
```

```
        CREATE("PirktiAkcijas")
```

```
        $PirktiAkcijas.Imone = $AkciuVertesPokytis.Imone
```

```
        $PirktiAkcijas.Vnt = pirktiAkciuVnt
```

```
    End If
```

C.2.3. Taisyklė „Įvertinti turimų įmonės akcijų kiekį“ (1)

Identifikatorius: 5

Pavadinimas: Įvertinti turimų įmonės akcijų kiekį

Pririoritetas: 3

Sąlyga: \$Pokyttis.Proc < -0.02 And \$Pokyttis.Proc > -0.07

Veiksma:

Dim parduotiVnt = 150

Dim turimosAkcijos= POINTS(\$AkciuVertesPokyttis.Imone)

If turimosAkcijos >= parduotiVnt Then

CREATE("ParduotiAkcijas")

\$ParduotiAkcijas.Imone = \$AkciuVertesPokyttis.Imone

\$ParduotiAkcijas.ParduotiVnt = parduotiVnt

End If

C.2.4. Taisyklė „Įvertinti turimų įmonės akcijų kiekį“ (2)

Identifikatorius: 6

Pavadinimas: Įvertinti turimų įmonės akcijų kiekį

Pririoritetas: 3

Sąlyga: \$Pokyttis.Proc <= -0.07

Veiksma:

Dim parduotiVnt = 250

Dim turimosAkcijos= POINTS(\$AkciuVertesPokyttis.Imone)

If turimosAkcijos >= parduotiVnt Then

CREATE("ParduotiAkcijas")

\$ParduotiAkcijas.Imone = \$AkciuVertesPokyttis.Imone

\$ParduotiAkcijas.ParduotiVnt = parduotiVnt

End If

C.3. Taisyklių rinkinys R3 (tik pakeistos ir pridėtos taisyklės)

C.3.1. Taisyklė „Įvertinti turimų pinigų kiekį“

Identifikatorius: 8

Pavadinimas: Įvertinti turimų pinigų kiekį

Pririoritetas: 99

Sąlyga: \$Pokyttis.Proc >= 0.1

Veiksma:

Dim pirktiAkciuVnt = 1000

Dim akciuKaina = pirktiAkciuVnt * \$AkciuVertesPokyttis.AkciuVerte

If akciuKaina >= #MyMoney Then

CREATE("Skolintis")

\$Skolintis.Suma = akciuKaina

End If

CREATE("PirktiAkcijas")

\$PirktiAkcijas.Imone = \$AkciuVertesPokyttis.Imone

\$PirktiAkcijas.Vnt = pirktiAkciuVnt

C.3.2. Taisyklė „Įvertinti turimų pinigų kiekį“ (1)

```
Identifikatorius: 4
Pavadinimas: Įvertinti turimų įmonės akcijų kiekį
Prioritetas: 3
Sąlyga: $Pokytis.Proc > 0.02 And $Pokytis.Proc < 0.05
Veiksmas:
Dim pirktiAkcijuVnt = 50

Dim akcijuKaina = pirktiAkcijuVnt * $AkcijuVertesPokytis.AkcijuVerte

If akcijuKaina <= #MyMoney Then

    CREATE("PirktiAkcijas")

    $PirktiAkcijas.Imone = $AkcijuVertesPokytis.Imone
    $PirktiAkcijas.Vnt = pirktiAkcijuVnt

End If
```

C.3.3. Taisyklė „Įvertinti turimų pinigų kiekį“ (2)

```
Identifikatorius: 3
Pavadinimas: Įvertinti turimų įmonės akcijų kiekį
Prioritetas: 3
Sąlyga: $Pokytis.Proc >= 0.05 And $Pokytis.Proc < 0.10
Veiksmas:
Dim pirktiAkcijuVnt = 100

Dim akcijuKaina = pirktiAkcijuVnt * $AkcijuVertesPokytis.AkcijuVerte

If akcijuKaina <= #MyMoney Then

    CREATE("PirktiAkcijas")

    $PirktiAkcijas.Imone = $AkcijuVertesPokytis.Imone
    $PirktiAkcijas.Vnt = pirktiAkcijuVnt

End If
```

C.3.4. Taisyklė „Skolintis pinigų“

```
Identifikatorius: 7
Pavadinimas: Skolintis pinigų
Prioritetas: 4
Sąlyga: Fact.Exists("Skolintis")
Veiksmas:
If Not WatchPoints.Exists("Skola") Then
    #Skola = 0
End If

#Skola = #Skola + $Skolintis.Suma
#MyMoney = #MyMoney + $Skolintis.Suma
#Bank = #MyMoney + #SharesMoney - #Skola

RETRACT("Skolintis")
```

C.3.5. Taisyklė „Užbaigti procesą“

Identifikatorius: 99

Pavadinimas: Užbaigti procesą

Prioritetas: 99

Sąlyga: true

Veiksmas:

```
MEM($AkciuVertesPokytis.Imone).AkciuVerte =  
$AkciuVertesPokytis.AkciuVerte
```

```
#Bank = #MyMoney + #SharesMoney
```

```
If WatchPoints.Exists("Skola") Then  
    #Bank = #Bank - #Skola  
End If
```

```
RETRACT("Pokytis")
```

```
RETRACT("AkciuVertesPokytis")
```

```
RETRACT("PirktiAkcijas")
```

```
RETRACT("ParduotiAkcijas")
```

D priedas

D.1. Simuliacijos įrankio prototipo įvykių valdymo lango vartotojo sąsaja

The screenshot displays the 'Dynamic Simulation Tool' interface. The top menu bar includes 'File' and 'Help'. Below it, a tabbed interface shows 'Events', 'Rules', 'Simulation', and 'Reporting'. The 'Events' tab is active, featuring two buttons: 'Load events from file...' and 'Save events to file...'. The main area is divided into two panels. The left panel, titled 'Event configuration', contains fields for 'Event name' (set to 'AkcijuVertesPokytyis'), 'Event data configuration' (set to 'Imone = "ImoneA"' and 'AkcijuVerte = Random(25, 40)'), 'Date range (random)' with an 'Enable' checkbox, 'From' and 'to' date pickers (01/01/2014 to 01/06/2014), 'Time' pickers (08:00 to 17:00), and 'Generate events count' (set to 50). The right panel, titled 'Events', displays a list of generated events, each with a 'Delete' button. The events are as follows:

Event	Time	Imone	AkcijuVerte
Event: AkcijuVertesPokytyis	2/11/2014 11:31:15 AM	ImoneB	16.6
Event: AkcijuVertesPokytyis	2/11/2014 1:23:16 PM	ImoneC	28
Event: AkcijuVertesPokytyis	2/11/2014 2:49:38 PM	ImoneA	12.9
Event: AkcijuVertesPokytyis	2/12/2014 10:36:53 AM	ImoneB	15.7
Event: AkcijuVertesPokytyis	2/12/2014 12:07:55 PM	ImoneC	24.5
Event: AkcijuVertesPokytyis	2/12/2014 12:44:35 PM	ImoneD	28.3
Event: AkcijuVertesPokytyis	2/13/2014 2:30:43 PM		

At the bottom of the interface, there are buttons for 'Show translated rule', 'Validate', 'Add >>', 'Clear all events', 'Sort', and 'Sort randomly'.

D.2. Simuliacijos įrankio prototipo taisyklių valdymo lango vartotojo sąsaja

The screenshot displays the 'Neutron Simulation' application window. The 'Rules' tab is active, showing a table of rules and a 'Rule editor' dialog box.

Rules Table:

Id	Title	Enabled	Edit	Delete
1	Inicijuoti simuliaciją		Edit	Delete
2	Apskaičiuoti pokytį		Edit	Delete
3	Įvertinti pokytį		Edit	Delete
4	Įvertinti pokytį		Edit	Delete
5	Įvertinti pokytį		Edit	Delete
6	Įvertinti pokytį		Edit	Delete
99	Užbaigti procesą		Edit	Delete
7	Įvertinti akcijų kiekį		Edit	Delete
8	Įvertinti pinigų kiekį		Edit	Delete
9	Pirkti akcijas		Edit	Delete
10	Parduoti akcijas		Edit	Delete

Rule editor dialog box:

- Rule id:** 2
- Priority:** 1
- Rule title:** Apskaičiuoti pokytį
- Rule condition:** true
- Rule body:**

```
CREATE("Pokytis")  
  
$Pokytis.Proc = ( $Akcijos.AkcijuVerte - MEM  
( $Akcijos.Imone).AkcijuVerte ) / $Akcijos.AkcijuVerte
```

Buttons: Show translation, Validate rule, Save, Cancel

D.3. Simuliacijos įrankio prototipo simuliacijos valdymo lango vartotojo sąsaja

Dynamic Simulation Tool

File Help

Events Rules Simulation Reporting

Simulation actions

Additional configuration

Simulation speed: 1 ms

Collision solver type: Priority

Simulation time: 5/31/2014 11:57 AM

Model instances

Id	Date	Status	Action
112.	25/05/2014 18:16:28	Finished	Event
113.	25/05/2014 18:16:28	Finished	Event
114.	25/05/2014 18:16:29	Finished	Event
115.	25/05/2014 18:16:29	Finished	Event
116.	25/05/2014 18:16:29	Finished	Event
117.	25/05/2014 18:16:30	Finished	Event
118.	25/05/2014 18:16:30	Finished	Event
119.	25/05/2014 18:16:31	Finished	Event
120.	25/05/2014 18:16:31	Finished	Event
121.	25/05/2014 18:16:32	Finished	Event
122.	25/05/2014 18:16:32	Finished	Event
123.	25/05/2014 18:16:32	Finished	Event
124.	25/05/2014 18:16:33	Finished	Event
125.	25/05/2014 18:16:33	Finished	Event
126.	25/05/2014 18:16:34	Finished	Event
127.	25/05/2014 18:16:34	Finished	Event
128.	25/05/2014 18:16:35	Finished	Event
129.	25/05/2014 18:16:35	Finished	Event
130.	25/05/2014 18:16:35	Finished	Event
131.	25/05/2014 18:16:36	Finished	Event
132.	25/05/2014 18:16:36	Finished	Event
133.	25/05/2014 18:16:37	Finished	Event

Current model instance

```

graph TD
    Start(( )) --> A[Perskaiciuoti turima banka]
    A --> B[Apskaiciuoti pokyti]
    B --> C[Ivertinti turimu pinigų kiekį]
    C --> D[Pirkti akcijas]
    D --> E[Užbaigti procesą]
    E --> End((( )))
  
```

Live working memory

Fact	Value
ImoneA	AkcijuVerte: 39.6
ImoneB	AkcijuVerte: 39
ImoneC	AkcijuVerte: 38.7
ImoneD	AkcijuVerte: 10
Simulation	Simuliacijai inicijuota

Simulation watch points

Watch point	Value
Bank	12186.2501639078
MyMoney	2787.25016390785
SharesMoney	9399
ImoneA	90
ImoneB	100
ImoneC	50
ImoneD	0

Event information

Name: AkcijuVertesPokytis

Date: 23/03/2014 15:35:28

Attributes:

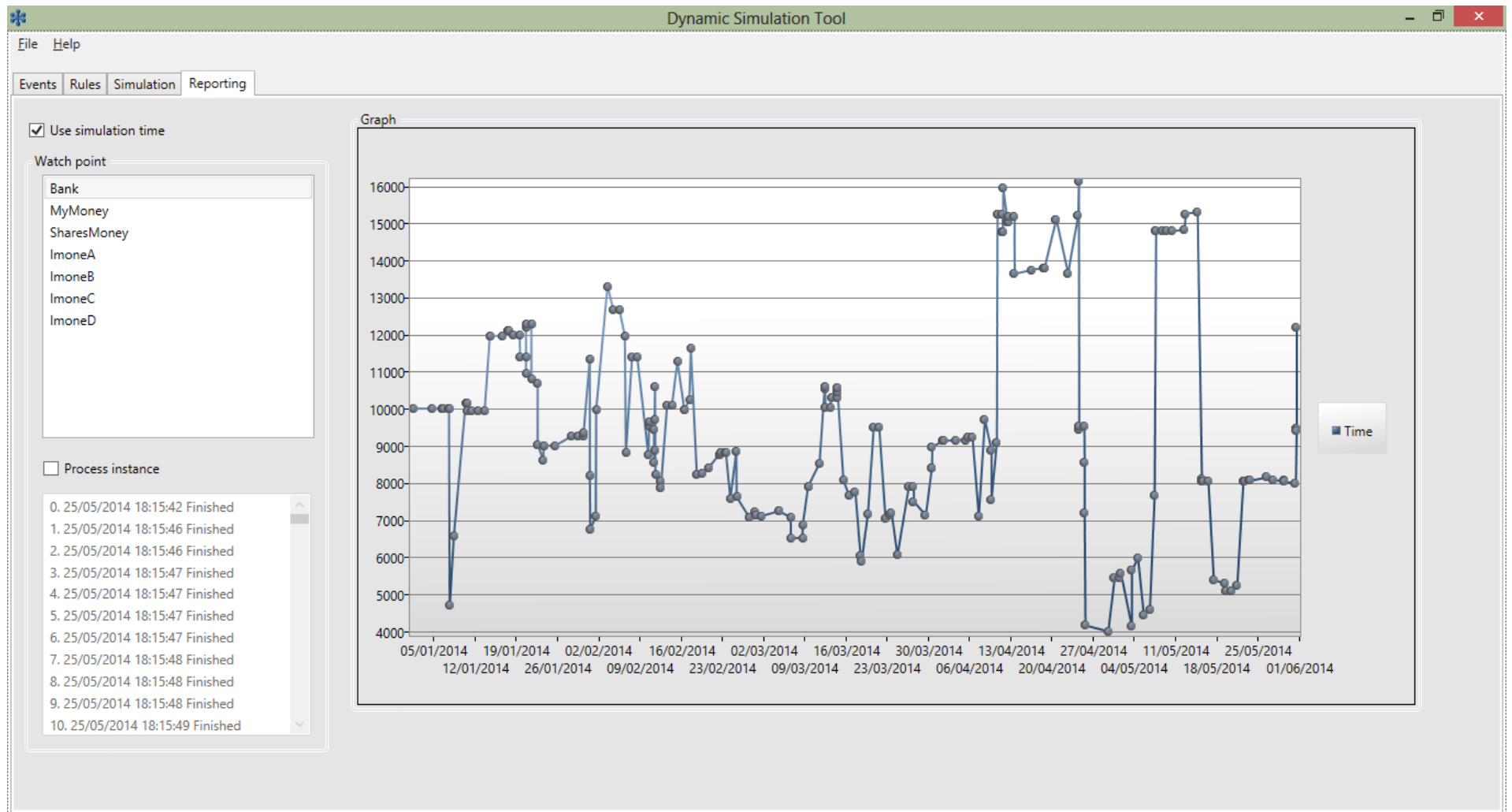
Imone = ImoneA

AkcijuVerte = 10.27

Simulation log:

- 25/05/2014 18:17:09 Fired rule Id: 3 Title: Ivertinti turimų pinigų
- 25/05/2014 18:17:10 Fired rule Id: 99 Title: Užbaigti procesą
- 25/05/2014 18:17:10 Process instance simulation finished.

D.4. Simuliacijos įrankio prototipo ataskaitų peržiūros lango vartotojo sąsaja



E priedas

E.1. Kompaktinis diskas

Kompaktinio disko turinys:

- *SimulationTool-Source/* - simuliacijos priemonės išeities kodai;
- *SimulationTool-Executable/* - simuliacijos priemonės vykdomoji programa;
- *Experiment-Files/* - simuliacijos taisyklių ir įvykių failai, kuriuos naudojant buvo atliekamas eksperimentinis tyrimas.
- *ISIFm-12_Martynas_Remeika_2014.docx* – baigiamasis darbas *docx* formatu.
- *ISIFm-12_Martynas_Remeika_2014.pdf* – baigiamasis darbas *pdf* formatu.