KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERINIŲ TINKLŲ KATEDRA

Mindaugas Škimelis

Atskirų resursų panaudos modelių sudarymas ir jų taikymas e. mokyme

Magistro darbas

Darbo vadovas
doc. dr. Kazys Baniulis

Kaunas, 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERINIŲ TINKLŲ KATEDRA

Mindaugas Škimelis

Atskirų resursų panaudos modelių sudarymas ir jų taikymas e. mokyme

Magistro darbas

Recenzentas:
doc. dr. Stasys Maciulevičius
2009-05-

Vadovas:
doc. dr. Kazys Baniulis
2009-05-

Atliko:
IFM-3/2 stud.
Mindaugas Škimelis
2009-05-21

Kaunas, 2009

SUMMARY

Nowadays software development is mainly based on development of small and reusable unified code artifacts (components). Quality and efficiency of reuse has huge impact on whole project lifecycle and maintenance efforts once the project is finished.

Goal of this study was to create a framework (codename *LiveModel*) for modeling usage of the separate resources (components) in the context of business and e-learning solutions. Flexible control of low-level Java objects was a key problem to solve. *LiveModel* framework was expanded with additional capabilities of UML modeling in *MagicDraw 15 UML* platform. *LiveModel Plugin* was created for this purpose. Finally *LiveModel* framework was successfully integrated to *TestTool 5 Author tool*. Creation and value of dynamic e-learning models was demonstrated.

SANTRAUKA

Šiandieninis programinės įrangos kūrimo procesas iš esmės yra paremtas nedidelių ir pakartotinai panaudojamų kodo artefaktų (komponentų) vystymu. Pakartotinio panaudojamumo proceso kokybė ir efektyvumas turi didelę įtaką projekto vystymo ciklui ir reikalingoms priežiūros pastangoms, kai projektas baigiasi.

Šio darbo tikslas buvo sukurti atskirų resursų (komponentų) panaudos logikos modeliavimo ir vykdymo karkasą (kodiniu pavadinimu *LiveModel*), pritaikomą verslo ir elektroninio mokymo sprendimams. Lankstus žemo lygio Java objektų valdymas buvo esminė šiame darbe išspręsta problema. *LiveModel* karkasas buvo išplėstas papildomomis UML modeliavimo galimybėmis *MagicDraw 15 UML* platformoje. Tam sukurtas *LiveModel* įskiepis. Galiausiai LiveModel karkasas buvo sėkmingai integruotas į nuotolinio mokymo sistemos TestTool 5 *Autoriaus* posistemę. Pademonstruotas dinaminių el. mokymo modelių kūrimo procesas ir vertė.

Turinys

1 Įvadas

Paskutinio dešimtmečio pasaulinis IT tobulėjimas stipriai veikia beveik visų planetos žmonių gyvenimus. Nuo sąskaitos banke iki skrydžio į kosmosą. Visose srityse atrandamos nišos, kur taikomos naujovės, šiuolaikiški inovatyvūs sprendimai. Žmonijos sugebėjimas greitai pritaikyti mokslo laimėjimus bei naujausias technologijas leidžia greitai sukurti papildomą ekonominę vertę ir skatina tolimesnį visuotinį tobulėjimą.

Biznio sektoriuose, kur organizacijos funkcionavimas grindžiamas informacinėmis technologijomis, reikalingas nuolatinis tobulėjimas ir gebėjimas sklandžiai prisitaikyti prie aplinkos pokyčių. Naujų sprendimų paieška tampa viena iš sudėtinių verslo modelio vystymo dalių. Vis dėlto labai dažnai tokie sprendimai virsta į brangius, ilgai kuriamus ir per daug sudėtingus galutinio vartotojo atžvilgiu. Priežasčių tam gali būti įvairių: maži biudžetai, darbuotojų kvalifikacijos stoka, nepakankamos naudojamos programinės įrangos galimybės, metodologiniai trūkumai, sudėtingumas.

Šiuolaikinės programinės įrangos kūrimas paremtas komponentų, turinčių savo paskirtį, savybes, gyvavimo trukmę, kūrimu. Apibrėžtą paskirtį turintys artefaktai gali būti taikomi daugybėje skirtingų dalykinių sričių. Pernaudojamumas (*reuse*) jau seniai tapo vienu iš esminių programinės įrangos kūrimo proceso dalių. Vis dėlto besiplečiant programų sistemoms, darosi labai sudėtinga manipuliuoti didelėmis komponentų savybių aibėmis. Modeliavimo kalbų vaidmuo tampa gyvybiškai svarbus. Apibrėžtų taisyklių pagalba išreikšta informacija, žinios bei struktūros sukuria aiškią vertę - modelį. Modeliavimas padeda tiksliau suprasti elementų paskirtį ir jų panaudojimo kontekstą. Tikslus susistemintos informacijos pateikimas tampa itin svarbia mokymosi proceso dalimi.

Konservatyviais vertinimais, šiai dienai pasaulinės elektroninio mokymo rinkos vertė siekia 38 milijardus eurų [21]. Juntamos aiškios tendencijos, jog su naujų technologijų atėjimu šis skaičius ateityje tik augs. Ši perspektyva daro elektroninio mokymo rinką žymiai patrauklesnę, bei skatina naujų, efektyvių mokymo sprendimų paiešką. Socialinių reiškinių kaitos fone mokymasis tampa nenutrūkstamu atributu kone kiekvieno žmogaus gyvenime. Natūraliai kyla klausimas: kaip įsisavinti ir atsirinkti reikalingą informaciją efektyviam mokymosi procesui?

Tradicinis elektroninis mokymas dažniausiai yra valdomas informacijos perdavimo paradigmos. Neretai šis faktas yra svarstomas kaip pirmosios elektroninio mokymo kartos

pagrindas. Šiandien jau yra koncentruojamasi sukurti pridėtinę ir išsaugomąją vertę kūrybinio proceso metu [4].

Tuo pat metu žmonės tampa išrankesni, vertina kokybę, kainą bei laiko sąnaudas, reikalingas mokomosios medžiagos įsisavinimui. Tam reikalinga specializuota programinė įranga, kuri leidžia greitai, patogiai bei interaktyviai perteikti žinias jų siekiančiam individui.

Šiame darbe apžvelgiama vystoma atskirų resursų (komponentų) panaudos modelių kūrimo metodologija bei jos taikymas e. mokyme.

2 Analitinė dalis

2.1 Tikslas

Pagrindinis šio darbo tikslas – išanalizuoti individualių resursų panaudos modelių kūrimo metodologiją ir atlikti taikymo elektroniniame mokyme tyrimą.

Uždaviniai:

- išanalizuoti individualių resursų panaudos modelių sudarymo ir taikymo galimybes.
- sukurti atskirų komponentų (toliau - resursų) panaudos modelių vykdymo karkasą (angl. *framework*) kodiniu pavadinimu *LiveModel*.
- sukurti *LiveModel* įskiepį *MagicDraw* platformai, panaudojant UML standartą.
- apžvelgti UML notacijos taikymo modelių sudaryme galimybes bei rasti sprendimus iškeltoms problemoms spręsti.
- pademonstruoti technines galimybes ir metodologiją kuriant bei pritaikant panaudos modelius elektroniniame mokyme.

2.2 Problemos

Reikalinga sistema, kuri palengvintų apibrėžtos dalykinės srities (arba srities fragmento) panaudos modelių kūrimą.

Programavimo technologijoms nuolat tobulėjant, kartu ir sudėtingėjant, programinės įrangos kūrimo ir palaikymo kaštai tampa vis didesni. Plačiai paplitusi objektinio programavimo paradigma neretai užgožia paprastesnių ir efektyvesnių sprendimų radimą bei taikymą.

Daugelyje programinės įrangos kūrimo projektų reikalingi profesionalūs įgūdžiai. Kaip bebūtų, yra ir kitų priežasčių. Žmonės, vedini praktinių, ekonominių, psichologinių ir edukacinių idėjų, be jokio profesionalaus pagrindo nori kurti programinę įrangą. Paprastai profesionalų darbo stilius ir rezultatų siekimo būdas negali būti taikomas technologinio pagrindo neturintiems individams [8]. Šiame darbe bandoma sumažinti šį atotrūkį, leidžiant kurti programinę įrangą paprastesniu būdu.

Realiame gyvenime yra daugybė dalykinių sričių, sudarytų iš vienetinių komponentų. Tai detalės, turinčios savo specifinę paskirtį ir komponuojamos siekiant išpildyti apibrėžtus dalykinės srities poreikius. Išgryninti ir nepriklausomi komponentai gali būti adaptuojami (pakartotinai panaudojami) kitose sistemose, kur jie sukuria papildomą pridėtinę vertę. Programiniame lygmenyje labai dažnai šių komponentų pakartotinis panaudojimas būna apribojamas platformos ar operacinės sistemos šeimos. Griežtas tipizavimas programavimo kalbose (*Java, C++, C# ir kt.*) taip pat neprideda lankstumo pernaudojant anksčiau minėtus komponentus.

Šiuolaikinis elektroninis mokymas paremtas mokomųjų objektų kūrimu ir jų pernaudojimu. Kiekvienas mokomasis objektas (*angl.: "learning object"*) gali būti naudojamas daugybėje įvairių kontekstų skirtingiems tikslams pasiekti [1]. Savo ruožtu mokymosi objektai yra sudaromi iš dar mažesnių vienetinių komponentų. Šių komponentų kūrimo ir panaudos problematika analizuojama šitame darbe.

KTU vystoma elektroninio mokymo sistema *TestTool 5* įgalina vartotojus kurti konceptualius mokomuosius objektus. Kūrimui panaudojami unifikuoti grafiniai komponentai, kurie turi apibrėžtą savybių aibę (*Pavyzdžiui.: pozicija, dydis, spalva ir kt.*). Mokymosi arba atsiskaitymo proceso metu studentai keičia norimų komponentų savybes. Studentui tai suteikia papildomas galimybes įsigilinti į dalykinę sritį ir efektyviai įsisavinti informaciją. Bene didžiausias pastebimas trūkumas yra tai, jog sukurti mokomieji objektai yra pakankamai statiški. Dinamiškumo trūkumas apriboja modeliavimo galimybes. Trukdo studentams detaliau įsigilinti į mokomuosius modelius.

Šiai problemai išspręsti reikalingas papildomas komponentų valdymo sluoksnis. Tokio sluoksnio atsiradimas įgalintų mokomųjų simuliacijų kūrimą *TestTool 5* sistemoje.

Rinkoje nepavyko rasti jokių novatoriškų sprendimų, galinčių valdyti žemiausio lygio mokymosi komponentų savybes. Šiai problemai spręsti reikia sukurti sistemą, galinčią keisti komponentų savybes pagal užprogramuotą logiką.

Kitas nemažiau svarbus aspektas – programinės kalbos suvaržymas. Logikos užrašymui reikia naudoti netipizuotą programavimo kalbą ar kalbas, nes mokomieji objektai gali turėti nesuderinamus tipus (tarp skirtingų programavimo kalbų).

Įvedus skirtingų programavimo kalbų taikymą programinėje įrangoje, būtina svarstyti unifikuotos modeliavimo kalbos (*UML*) taikymo galimybes.

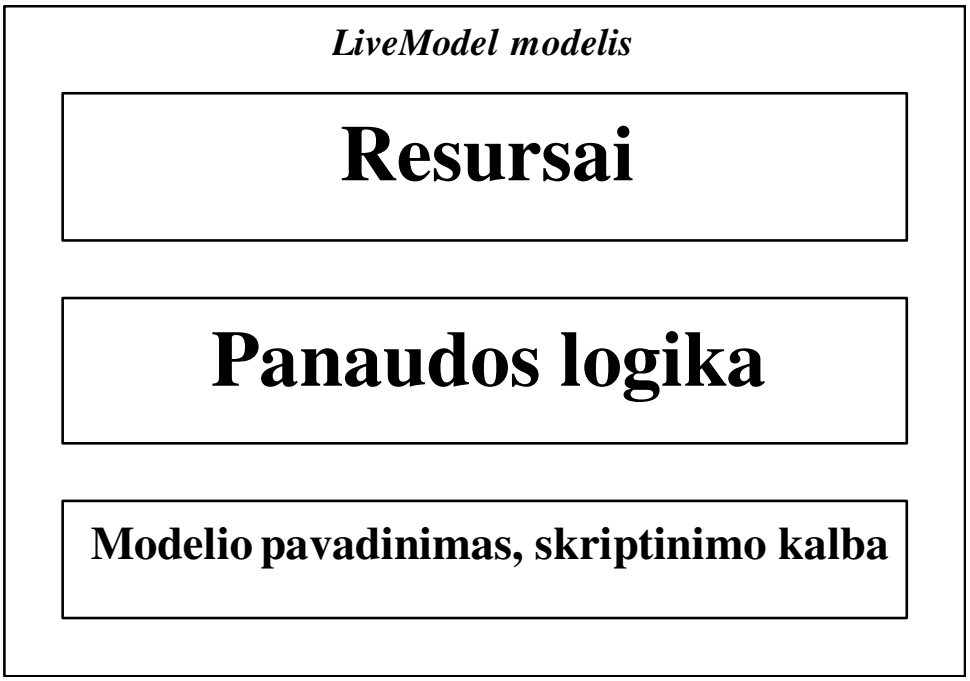
2.3 Galimi sprendimai

Kuriamą sistemą sudaro dvi pagrindinės sudedamosios dalys. Tai *LiveModel* karkasas ir *MagicDraw 15 UML* modeliavimo įrankiui skirtas įskiepis "*LiveModel Plugin"*. Taip pat skyriuje bus apžvelgiami *TestTool 5* sistemos galimybės ir patobulinimai. Būtent ši sistema *LiveModel* sistemos pagrindu bus praplėsta papildomomis modeliavimo galimybėmis. Šiame skyriuje apžvelgiamos naudojamos technologijos.

2.3.1 LiveModel modelio struktūra

LiveModel modelis – nepriklausoma duomenų struktūra, kurios sudedamosios dalys yra resursai, panaudos logika, modelio pavadinimas ir panaudos logikos kalbos pavadinimas (skriptinimo kalba).

Modelis saugomas atskiroje byloje, panaudojant "*Java Serialization API*" technologiją. Tokiu būdu ši byla gali būti lengvai pernešama ir vykdoma kitose aplinkose, kur yra palaikoma Java platforma.



1 pav. LiveModel modelio struktūra.

Apibrėžkime keletą sąvokų:

LiveModel modelis – objektinė duomenų struktūra, sauganti modeliavime naudojamus resursus, panaudos logiką bei kitus reikalingus nustatymus tam, kad sėkmingai būtų vykdomas sukurtas modelis.

Resursas – tai Java objekto aprašymas ir jo realizacija *LiveModel* aplinkai. Šis resursas visada turi "gyvą" atitikmenį *LiveModel* aplinkoje. Aprašymą sudaro: *pavadinimas*, *tipas*, *kelias iki fizinio kodo artefakto* (*Java Archive bylos*).

Savybė – LiveModel resurso skiriamasis bruožas, galintis būti tiek duomenų lauku (*vardas, tipas ir t.t.*), tiek specifine operacija (*eiti, siųsti laišką ir t.t.*)

Modelio pavadinimas – tai unikalus modelio vardas, identifikuojantis modelį.

Skriptinimo kalba - viena iš JSR 223 specifikaciją palaikančių programavimo kalbų, naudojama panaudos logikos aprašymui.

2.3.2 Modelio sudarymo procesas

LiveModel modelis gali būti sudaromas dviem skirtingais būdais:

- *MagicDraw UML* aplinkoje naudojantis klasių diagramoje aprašytais resursais bei veiklos diagramoje suformuota panaudos logika.

- Taikomosios programinės įrangos aplinkoje, kur integruotas *LiveModel*. Čia naudojami vidiniai taikomosios programinės įrangos resursai.

2.3.2.1 MagicDraw UML aplinkoje

LiveModel modelis *MagicDraw 15 UML* aplinkoje sudaromas pasinaudojant "*LiveModel Plugin*" įskiepiu šiai platformai.

LiveModel modelio struktūra aprašyta skyriuje "2.3.1 LiveModel modelio struktūra". Resursų aprašymui naudojama klasių diagrama. Klasių (*Class*) diagramoje aprašomi atskiri resursai su savo metodais (*operations*). Kiekvienam modelio vykdyme naudojamam resursui sukuriamas "*Instance Specification*" elementas. Šis elementas reprezentuoja realų nurodyto tipo objektą (pavyzdžiui: "*java.util.Properties*").

2 pav. *"Instance Specification"* elementas *"LiveModel"* resursui

Iš sukurtų resursų aprašymų klasių diagramoje resursai toliau yra naudojami panaudos logikos kūrimo procese.

Resursų operacijų apjungimui į vykdymo seką naudojama veiklos diagrama (*Activity*). Šios diagramos pagalba apibrėžiama resursų operacijų kvietimo tvarka. Šiam tikslui panaudojamas "*Call Operation Action*" elementas, kuris aprašo metodo (*operacijos*) kvietimą iš pasirinktos klasės.

3 pav. "*Call Operation Action*" elemento panaudojimas.

Kintamųjų aprašymui naudojami vidiniai veiklos diagramos elementai – "Pins". *LiveModel* kontekste šie elementai bus naudojami kaip kintamieji arba kintamųjų reikšmės.

Lentelė 1 "Pins" elemento naudojimo pavyzdys

Elementas	Paaiškinimas
	Argumentas. **Pavyzdys:** *webservice.operacija('EUR');*
	Kintamasis pavadinimu "rate". Rezultate **Pavyzdys:** *var rate = webservice.operacija('EUR');*

Šitokiu būdu atliktos transformacijos iš veiklos diagramos į kodą leis vartotojui greitai prisitaikyti prie besikeičiančios aplinkos. Sukurtas modelis saugomas į bylą.

2.3.2.2 Taikomųjų programų aplinkoje – TestTool

Taikomojoje programinėje įrangoje *LiveModel* modeliavimo procesas valdomas pasinaudojant *LiveModel API*.

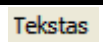
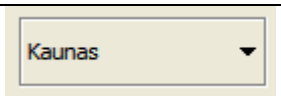
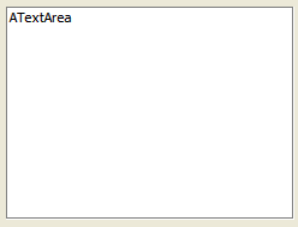
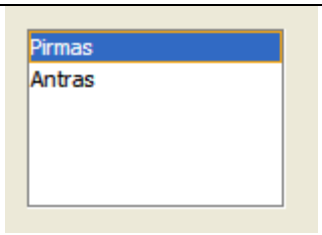
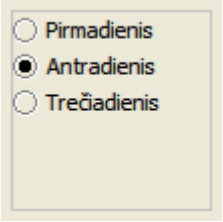
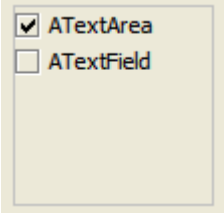
Objektai, sukurti taikomojoje programinėje įrangoje, yra tiesiogiai talpinami į *LiveModel* modelį. Šių resursų panaudos logika gali būti užkraunama atitinkamai iš bylos ar teksto įvedimo lauko (priklausomai nuo realizacijos). Bandomoji *LiveModel* realizacija atlikta *TestTool 5* aplinkoje.

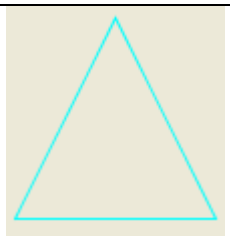
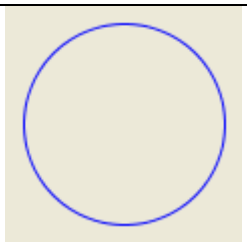
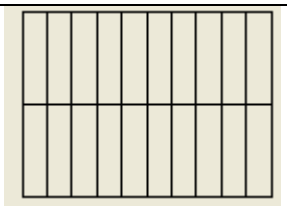
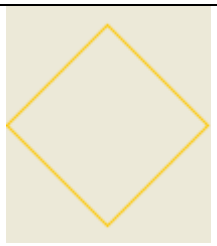
Darbui su *LiveModel* karkasu TestTool 5 aplinkoje reikia:

- Sukurti naują TestTool variantą.
- Pridėti naujus komponentus (*resursus*) į naują variantą.
- Papildyti *LiveModel* modelį pasirinktais resursais.
- Priskirti panaudos logiką iš pasirinkto šaltinio (byla, teksto įvedimo laukas ar kt.)..
- Vykdyti simuliaciją, nurodant vykdymo laiką ir dažnį.

Panaudos logikos sudarymui svarbiausia žinoti TestTool 5 sistemos komponentų turimas savybes (operacijas).

Lentelė 2 Pagrindinės TestTool komponentų savybės

Elementas	*Pavadinimas*	*Pagrindinės savybės/metodai*
	**ALabel**	Text, Position, Location, Background, Foreground, Font, Size. moveComponent,
	**ATextField**	Text, Position, Location, Background, Foreground, Font, Size.
	**AComboBox**	Position, Location, Background, Foreground, Font, Size. AddItem, RemoveItem, SelectedItems,
	**ATextArea**	Text, Position, Location, Background, Foreground, Font, Size.
	**AList**	Position, Location, Background, Foreground, Font, Size. AddItem, RemoveItem, SelectedItems,
	**ARadioGroup**	Position, Location, Background, Foreground, Font, Size. AddItem, RemoveItem, SelectedItems,
	**ACheckGroup**	Position, Location, Background, Foreground, Font, Size. AddItem, RemoveItem, SelectedItems,

	**ATriangle**	Position, Location, Background, Foreground, Font, Size.
	**ARectangle**	Position, Location, Background, Foreground, Font, Size.
	**AEllipse**	Position, Location, Background, Foreground, Font, Size.
	**ALine**	P1, P2, Background, Foreground.
	**ATable**	Position, Location, Background, Foreground, Font, Size.
	**ARhomb**	Position, Location, Background, Foreground, Font, Size.
	**AImage**	Image, Position, Location, Background, Foreground, Font.

2.3.3 Sistemos variklis - JSR 223

2003 metų kasmetinėje *JavaOne* konferencijoje buvo aptartas ir sukurtas naujas Java bendruomenės projektas kodiniu numeriu JSR 223 (*angl.:* JSR - "*Java Specification Request*"). Šis projektas sulaukė labai didelio susidomėjimo, nes tai buvo pirmasis oficialus žingsnis panaudojant ne griežtai tipizuotas programavimo kalbas Java platformoje.

JSR 233 (*"Scripting for the Java™ Platform"*) apima sąveiką tarp Java ir skriptinimo kalbų. Projekto rezultatas yra nedviprasmiškai apibrėžtas standartinis API, kuris tarnauja kaip tiltas tarp Java ir skriptinimo kalbų. Java objektai yra naudojami skirtingų skriptinimo kalbų kontekste.

Konferencijos metu suformuoti laukiami rezultatai:

- Standartinė programinė sąsaja tarp Java ir kandidatuojančių skriptinimo kalbų, nepaisant skriptinimo kalbos kilmės (*Java* ar ne).
- "*Java Servlet*" technologijos palaikymas.
- Standartinis internetinių puslapių pakavimas, kai naudojamos ne Java šeimos programavimo kalbos. Web aplikacijų priežiūros, saugumo, resursų valdymo, klasių užkrovimo metodologija neturi keistis.

Viena iš pirmųjų projekto apžvalgų, atliktų mokslinėje erdvėje, leido daryti išvadą, kad ši nauja technologija dar labiau skatins R-technologijos (*angl.: reuse*) naudojimą net pačiose sudėtingiausiose srityse [3]. Būtent išskirtinė sujungimo (*angl.: bridge*) paklausa Web aplinkoje leido sukurti puikią technologiją, įgalinančią vartotojus lanksčiai integruotis tarp skirtingų sistemų.

2.3.3.1 JSR 223 palaikomos programavimo kalbos

Lentelėje pateikiamos pagrandinės ir populiariausios *JSR 223* palaikomos programavimo kalbos.

Lentelė 3 JSR 223 palaikomos programavimo kalbos

Kalba	Apibūdinimas	Sprendimas.Versija.
AWK	AWK yra bendros paskirties kalba, kuri buvo sukurta apdoroti tekstinei informacijai failuose.	Jawk 0.14

BeanShell	BeanShell yra mažas, nemokamas Java interpretatorius su objektų skriptinimo kalbos savybėmis. BeanShell dinamiškai įvykdo standartinę Java sintaksę.	BeanShell 2.0b5
ejs	"*ejs*" ("*Embedded JavaScript*") yra *JSP* stiliaus skriptinimo variklis *JavaScript*. Palaikomos visiems gerai žinomas išraiškas: <%= *expr* %>, <% *code* %>	Visa sprendimas yra vienoje *JavaScript* byloje.
FreeMarker	*FreeMarker* yra *Java* pagrindu sukurtas bendro naudojimo šablonų variklis.	FreeMarker 2.3.8
Groovy	*Groovy* yra jautri, dinamiška kalba Java 2 platformai, kurioje yra daugybė savybių, kurios patinka žmonėms tokiose kalbose, kaip *Python*, *Ruby* ir *Smalltalk*. Visos šios savybės tampa pasiekiamos naudojant Java stiliaus sintaksę.	Groovy 1.1 beta-2
Jaskell	*Jaskell* yra funkcinio programavimo kalba.	Jaskell 1.0
Java	http://java.sun.com	Java Compiler API (JSR 199)
JavaScript	*Rhino 1.6R4* paremtas *JavaScript* scriptinimo variklis.	Rhino 1.6R4
JavaScript	Web naršyklės gimtasis JS interpretatorius yra apgaubtas kaip *javax.script* API.	Web naršyklės JS sprendimas (testuota su *Firefox* 1.5.0)
Jelly	*Jelly* yra įrankis kuris XML paverčia į besivykdantį kodą. *Jelly* pasiskolino daug gerų idėjų iš *JSP*, *Velocity* bei *Ant*.	Jelly 1.0
JEP	JEP yra Java biblioteka skirta interpretuoti ir įvertinti matematines išraiškas. *JEP* palaiko BigInteger, BigDecimal, kompleksinių skaičių operacijas, Vektorių ir matricų aritmetikas.	JEP (Java Math Expression Parser) 2.4.0
JEXL	Java išraiškų kalba, kuri gali būti lengvai pritaikoma įvairiose aplikacijose.	Jexl 1.0
jst	"*jst*" (JavaScript Templates) yra JSP/ASP/PHP-stiliaus šablonų variklis skirtas *JavaScript*.	TrimPath JavaScript Templates (1.0.38)

JudoScript	Skriptinimo kalba su galinga bendro tikslo programavimo kalba.	JudoScript 0.9
JUEL	JUEL tai *Java Unified Expression Language*; Standartas (JSR-245).	JUEL 2.1.0-rc2
OGNL	OGNL (Object-Graph Navigation Language) tai yra išraiškų kalba, kurios pagalba disponuojama Java Objektų savybėmis.	OGNL 2.6.9
Pnuts	Pnuts yra paprasta, bet galinga skriptinimo kalba kuri turi praplečiamų modulių sistemą.	Pnuts 1.1
Python	*Python* yra dinaminė, objektinė programavimo kalba, kuri gali būti naudojama daugelyje programinės įrangos projektų.	Jython 2.2b1
Ruby	*Ruby* yra interpretuojama skriptinimo kalba greitam objektiniam programavimui.	JRuby 1.0
Scheme	Scheme programavimo kalba su daugybe programavimo paradigmų (imperatyvine, funkcine, žinučių perdavimo).	SISC 1.16.6
Sleep	*Sleep* yra skriptinimo kalba Java pagrindu. Daugiausia įkvėpta *Perl* kalbos.	Sleep 2.0
Tcl	*Tcl* (“*Tool Command Language”*) labai galingas, bet lengvai išmokstama dinaminio programavimo kalba. Ji tinka labai plačiam panaudojimui.	Jacl 1.3.3
Velocity	*Velocity* yra Java pagrindu sukurtas bendro naudojimo šablonų variklis.	Velocity 1.5
XPath	Technologija leidžianti pasiekti *XML* elementus nenaudojant *DOM*.	JDK javax.xml.xpath sprendimas.
XSLT	*XML* transformacijų galimybės.	JDK javax.xml. transform sprendimas.

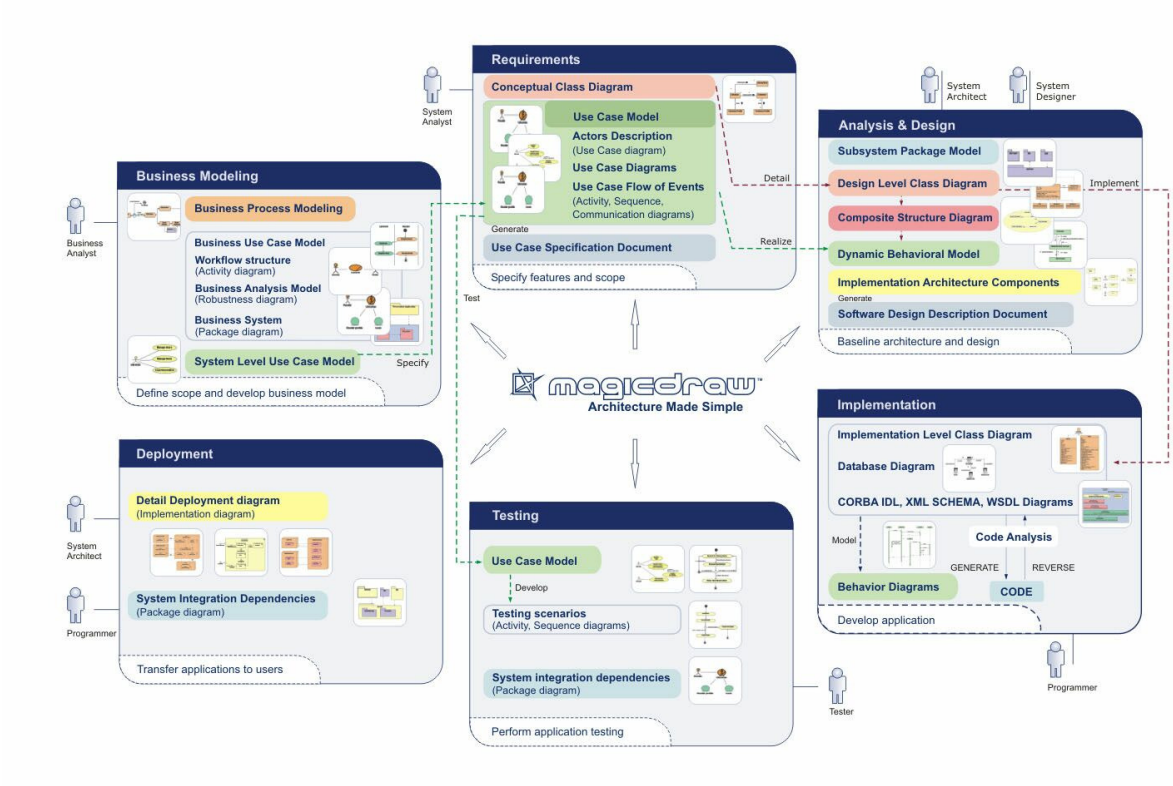
2.3.4 Java 1.6 platforma

Java 1.6 yra naujausia ir funkcionaliausia *Java* šeimos platforma. Darbe naudojama *TestTool 5* sistema yra pilnai parašyta naudojant *Java 1.5*. Siekiant suderinamumo su naujausiomis technologijos, esama Java versija turi būti atnaujinta iki 1.6., nes JSR 223 palaikomas tik nuo *Java 1.6* versijos.

2.3.5 MagicDraw 15 ir UML

Darbui su komponentais (resursais) bei jų savybėmis reikalingos UML modeliavimo galimybės. *LiveModel* modelio kūrimas tūri būti automatizuotas procesas.

Lietuvoje kuriamas *MagicDraw 15* programinės įrangos paketas pripažįstamas kaip vienas geriausių rinkoje esančių produktų. UML standarto palaikymo tikslumas viršija 90%.

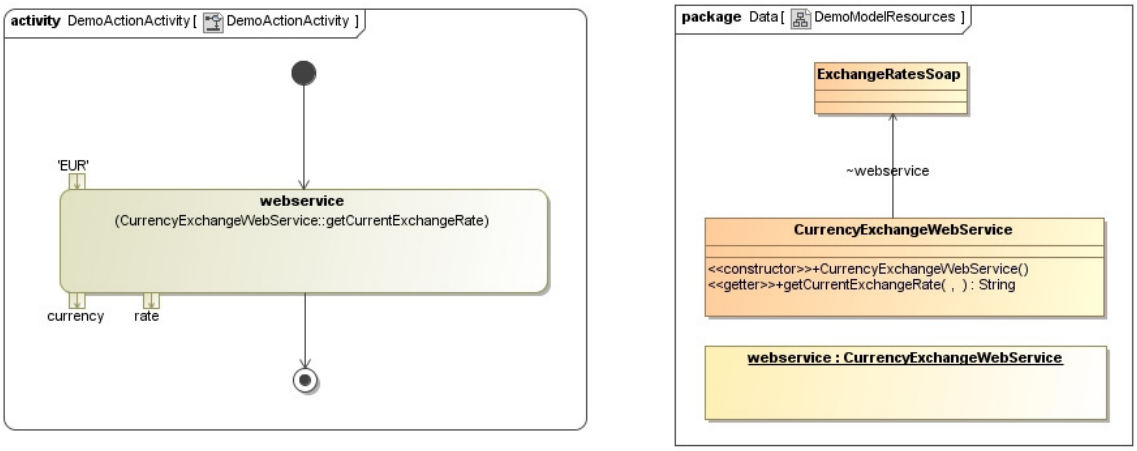


4 pav. MagicDraw programinės įrangos kūrimo proceso kontekste

MagicDraw™ UML atitinka naujausius Java bei UML technologijų standartus, turi vieną iš patikimiausių išeities kodų inžinerijos mechanizmų Java, C#, C++ ir CORBA IDL programavimo kalboms bei gali vykdyti šių kalbų kodo atvirkštinę inžineriją, duomenų bazių schemų atvirkštinę inžineriją, kodo bei duomenų bazių schemų generavimą. *MagicDraw™ UML* panaudoja "*roundtrip*" technologiją, leidžiančią keisti tiek OO modelį, tiek programos kodą bet kokiu eiliškumu, juos nuolat sinchronizuojant. Taip pat *MagicDraw™ UML* - vienas iš nedaugelio rinkoje esančių paketų, leidžiančių greičiau nubraižyti UML diagramas bei turintis UML diagramų semantinio teisingumo tikrinimo ir modelio validavimo mechanizmą.

MagicDraw™ UML Teamwork Server programinė įranga turi komandinio darbo galimybę, leidžiančią daugeliui programinės įrangos kūrimo inžinierių dirbti su tuo pačiu OO modeliu vienu metu.

LiveModel Plugin projekte siekiama pasinaudoti klasių (*Class*) ir veiklos (*Activity*) diagramų teikiamais privalumais. Klasių diagrama bus naudojama komponentų (resursų) aprašymui, o veiklos diagrama - sistemos elgsenos modeliavimui bei transformavimui iš PIM į PSM modelį.



5 pav. UML klasių ir veiklos diagramos

Resursų aprašymui klasių diagramoje būtina išnaudoji atvirkštinės kodo inžinerijos savybes. Tai ypač palengvintų ir pagreitintų jau egzistuojančių sistemų reinžineriją kuriant *LiveModel* panaudos modelius.

MagicDraw UML paketas taip pat labai patrauklus dėl savo vystomos *Open API* strategijos, kurios pagalba galima kurti specializuotus įskiepius išnaudojant vidinius *MagicDraw* modelius, kas ir bus atliekama šio darbo metu.

2.3.6 MagicDraw įskiepis

Greitam ir efektyviam *LiveModel* modelių kūrimui naudojamas *LiveModel* įskiepis. Įskiepis gebės konvertuoti *MagicDraw* vidinį komponentų modelį į *LiveModel* panaudos modelį. Tikslui pasiekti bus naudojamos šios operacijos:

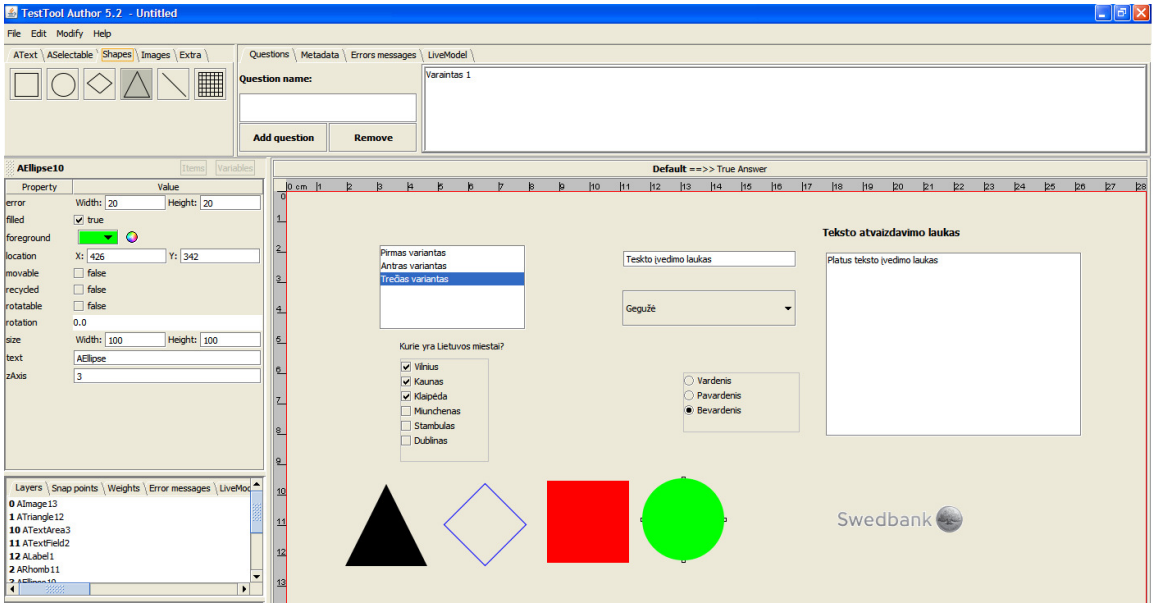
- Resursų aprašymas ir kūrimas klasių diagramos lygmenyje.

- Resursų panaudos logikos modeliavimas veiklos diagramos lygmenyje.
- Resursų panaudos logikos transformavimas į norimą skriptinimo kalbą (PIM [23] transformacija į PSM [21]).
- *LiveModel* modelio saugojimas į bylą.

2.3.7 LiveModel ir TestTool 5 sistema

LiveModel projektas yra visiškai nepriklausomas nuo jokių kitų artefaktų. Vienintelė priklausomybė siejama su *Java 1.6*, kurios pagrindu sistema ir sukurta.

TestTool 5 mokymo sistema yra viena iš *LiveModel* taikymo sričių. Dėl savo unifikuotais komponentais paremtos architektūros TestTool sistema turi puikias galimybes tapti galingu mokomųjų objektų modeliavimo įrankiu.



6 pav. TestTool sistemos unifikuoti komponentai ir jų savybės

TestTool komponentas dėl savo išgryninto individulių savybių panaudojimo yra puikus pretendentas į *LiveModel* resursus.

2.4 Priimto sprendimo pagrindimas

Šiame skyriuje bus apžvelgiami faktai, kuriais grindžiamas vienos ar kitos technologijos naudojimas.

2.4.1 JSR 223 naudojimo pagrindimas

Atlikus tyrimą, rinkoje nepavyko rasti jokių kitokių alternatyvų, išskyrus *JSR 223*. Šis projektas tenkina keliamus reikalavimus:

- Java objektai pasiekiami skirtingose programavimo kalbose.
- Palaikomos skirtingos programavimo kalbos. Apie 40.
- Java 1.6 palaikymas.
- Griežto tipizavimo nebuvimas.
- Java bendruomenės procesas ("*Java Community Process*") užtikrina pilną palaikymą ateityje.
- *TestTool 5* sistemos komponentai yra 100% parašyti Java kalboje, tai daro JSR 223 idealiu kandidatu plečiant TestTool 5 sistemą naujomis modeliavimo galimybėmis.

2.4.2 Java 1.6 pagrindimas

Kadangi *JSR 223* palaikomas tik nuo Java 1.6, nepaliekama jokių kitų galimybių.

2.4.3 MagicDraw 15 ir LiveModel įskiepis

Plačiai paplitus grafinių abstrakčių UML modelių kūrimui, vis svarbiau tampa išnaudoti kitų žmonių sukurtą įdirbį. UML standartas, kuris yra palaikomas OMG grupės, jau vystomas 15 metų. Per tą laiką šis standartas tapo varančiąja jėga programų kūrimo procese.

MagicDraw turi išvystytą stiprų vidinį komponentinį modelį. Jis laisvai pasiekiamas per patogias ir atviras programines sąsajas. Išvystyta įskiepių kūrimo metodologija.

Naudojant šį įrankį, *LiveModel* žengs stiprų pradinį žingsnį vystant efektyvius panaudos modelius.

2.5 Projektavimo metodologijos ir technologijų analizė

Projektas vykdomas remiantis *IBM Rational* kompanijos patirtimi ir jos teikiama *Rational Unified Process* (*RUP*) metodika, o tai užtikrins gerą produkto kokybę, taip pat produkto kūrimo proceso valdymą ir atlikimą laiku. Projektinė dokumentacija bus paruošta naudojant *Unified Modeling Language* (*UML*) notaciją, tuo teikiant galimybę pakartotinai panaudoti projekte surastus problemų sprendimus kituose projektuose.

Technologijų analizė atlikta skyriuose *2.3 Galimi sprendimai* ir 2.4 Priimto sprendimo pagrindimas.

2.6 Analitinės dalies išvados

- Analitinėje dalyje apžvelgtos rimtos el. mokymo problemos, nusistatyti darbo tikslai.
- Apžvelgtos TestTool 5 sistemos savybės ir reikalingi patobulinimai.
- Apibūdinti bendri darbo su *LiveModel* karkasu principai.
- Pasiūlytos ir pagrįstos technologijos visoms iškeltoms problemos spręsti.

3 Projektinė dalis

3.1 Tikslas

Magistro darbo projektinės dalies tikslas - sukurti atskirų komponentų (toliau *resursų*) panaudos modelių vykdymo karkasą (*angl. framework*) kodiniu pavadinimu *LiveModel*. Tai pat pademonstruoti technines galimybes ir metodologiją kuriant bei pritaikant panaudos modelius elektroniniame mokyme. Elektroninio mokymo idėjos vystomos TestTool 5 sistemos pagrindu. Alternatyvus modelių kūrimas pasinaudojant "*MagicDraw 15" LiveModel* įskiepiu. Remiantis analizės dalyje pateiktomis sistemos perprojektavimo rekomendacijomis buvo sukurta nauja programinė įranga bei sukurta visa projekto dokumentacija nuo reikalavimų specifikacijos iki vartotojo vadovo.

Pateikiame sukurtos techninės - projektinės dokumentacijos svarbiausius aspektus.

3.2 Bendra apžvalga

Šioje dokumento dalyje bus nagrinėjami šie punktai:

Architektūrinis vaizdas. Aprašomas pasirinktas sistemos architektūrinis atvaizdavimas, apžvelgiami skirtingi vaizdai į PS.

Architektūriniai tikslai ir apribojimai. Pateikiami kuriamai sistemai keliami tikslai ir apribojimai, pasirenkamų sprendimų galimybės, įrankių panaudojamumas, panaudojamų komercinių produktų privalumai ir trūkumai.

Panaudojimo atvejų vaizdas. Aprašomi sistemos aktoriai ir jų veiksmai.

Loginis vaizdas. Čia bus pateikta sistemos klasių diagrama ir sistemos objektų sąveikos.

Proceso vaizdas. Pateikiamos objektų bendradarbiavimo ir scenarijų diagramos.

Realizacijos vaizdas. Pateikiama komponentų diagrama.

3.3 Architektūrinis vaizdas

Dokumente sistemos architektūra pateikiama keliais vaizdais. Šie vaizdai yra pateikiami kaip *MagicDraw* modeliai naudojant unifikuotą modeliavimo kalbą (UML). Sistemos architektūra pateikta remiantis RUP (*Rational Unified Process*) rekomendacijomis. Sistemos specifikacija pateikta šiais vaizdais, kuriems įgyvendinti reikia UML diagramų:

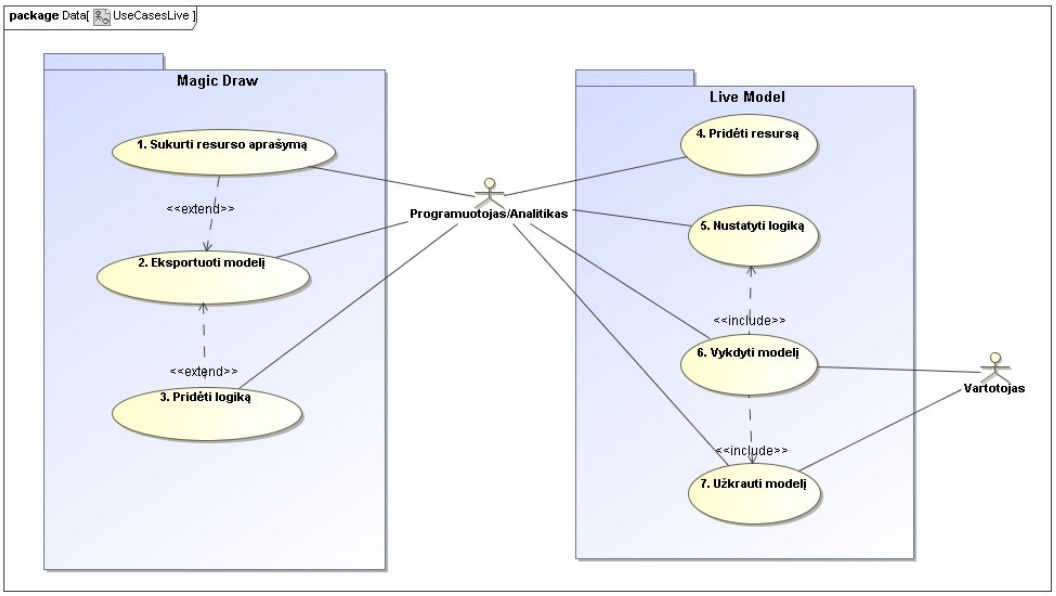
1. Panaudojimo atvejų vaizdas (panaudojimo atvejų diagrama);
2. Sistemos statinis vaizdas (paketai ir klasių diagramos);
3. Sistemos dinaminis vaizdas (būsenų, veiklos, sekų, bendradarbiavimo diagramos);
4. Išdėstymo vaizdas (išdėstymo diagrama).

3.3.1 Architektūriniai tikslai ir apribojimai

Yra keletas reikalavimų ir apribojimų, kurie turi įtakos sistemos architektūrai. Jie yra:

- Sistema turi būti realizuota Java aplinkoje;
- Sistema turi užtikrinti našumą ir klaidų kontrolę;
- Sistema turi palaikyti įvairias skriptinimo kalbas pagal JSR-223 specifikaciją;
- Programinės įrangos koncepcijos turi būti suderinamos su *MagicDraw* vidiniu komponentų modeliu ir jo galimybėmis;
- *MagicDraw* įskiepiui kurti turi būti naudojama *Open API* koncepcija;
- Sudarant sistemos architektūrą, turi būti atsižvelgta į būtinas programos vykdymo charakteristikas, apibrėžtas reikalavimų specifikacijoje.

3.3.2 Panaudojimo atvejų vaizdas



7 pav. Panaudos atvejų vaizdas

Panaudos atvejis: **1. Sukurti resurso aprašymą**

Tikslas:	Aprašo realų resursą "*LiveModel*" suprantamu formatu.
Aktoriai:	Programuotojas/Analitikas
Ryšiai su kitais PA:	Praplečia "Eksportuoti modelį"
Prieš-sąlyga:	Paruošti resursai (Sukompiliuotos klasės, paruošti failai ir t.t.)
Sužadinimo sąlyga:	Į "MagicDraw" klasių diagramą įdedama klasė atitinkamame pakete ir priskiriamas vardas.
Po-sąlyga:	Klasių diagramoje atsiranda eksportui paruošta klasė.
Pagrindinis scenarijus:	Į "MagicDraw" modelį tam tikrame pakete įdedama klasė, kuri atspindi resursą. Nurodoma fizinė failo (sukompiliuotų Java klasių) vieta. Saugoti "MagicDraw" modelį.
Alternatyvūs scenarijai:	Rankinis Modelio failo sukūrimas per programinę sąsają.

Panaudos atvejis: **2. Eksportuoti modelį**

Tikslas:	Eksportuoja iš resursų ir logikos sudarytą modelį į modelio failą.
Aktoriai:	Programuotojas/Analitikas

Prieš-sąlyga:	Aprašyti resursai, aprašyta logika.
Sužadinimo sąlyga:	"*MagicDraw*" įskiepio meniu spaudžiamas mygtukas "Export Live Model"
Po-sąlyga:	Nurodytame kataloge sukuriamas modelio ir resursų aprašymo failai.
Pagrindinis scenarijus:	Modelyje aprašomi resursai. Modelyje aprašoma panaudos logika. Eksportuojamos pasirinktos klasės.
Alternatyvūs scenarijai:	Rankinis Modelio failo sukūrimas per programinę sąsają

Panaudos atvejis: **3. Pridėti logiką**

Tikslas:	Aprašo procesą, kurio metu analitikas/programuotojas gali nustatyti/pakeisti jau aprašytą modelio logiką.
Aktoriai:	Programuotojas/Analitikas
Ryšiai su kitais PA:	Praplečia "Eksportuoti modelį"
Prieš-sąlyga:	Aprašyti resursai
Sužadinimo sąlyga:	Logikos įrašymo lange aprašoma modelio elgsena.
Po-sąlyga:	Logika paruošta išsaugojimui.
Pagrindinis scenarijus:	Atveriamas logikos aprašymo langas. Naudojant turimus/numatomu resursų vardus kuriamas norimas logikos aprašymas.
Alternatyvūs scenarijai:	-

Panaudos atvejis: **4. Pridėti resursą**

Tikslas:	Aprašo procesą, kurio metu analitikas/programuotojas gali pakeisti jau aprašytą logiką.
Aktoriai:	Programuotojas/Analitikas
Prieš-sąlyga:	Modelis užkrautas
Sužadinimo sąlyga:	Modelio valdymo interfeiso metodui perduodamas objektas arba resurso aprašymas, reikalingas konkrečiam modeliui.
Po-sąlyga:	Resursas sėkmingai patalpintas modelio kontekste.
Pagrindinis scenarijus:	Sukuriamas objektas atmintyje arba paruošiamas resurso aprašymo failas. Kviečiamas modelio valdymo interfeiso metodas objekto išsaugojimui.
Alternatyvūs scenarijai:	Logika pati kviečia modelio valdymo interfeiso metodą resursų pridėjimui, nuo tada pati rūpinasi tolimesniu objekto valdymu.

Panaudos atvejis: **5. Nustatyti logiką**

Tikslas:	Aprašo procesą, kurio metu analitikas/programuotojas gali pakeisti jau aprašytą logiką.
Aktoriai:	Programuotojas/Analitikas

Prieš-sąlyga:	Modelis užkrautas
Sužadinimo sąlyga:	Analitikas/programuotojas pakoreguoja egzistuojančią logiką ir vykdo jos išsaugojimą.
Po-sąlyga:	Modelyje atnaujinama panaudos logika, kuri kito vykdymo metu bus naudojama.
Pagrindinis scenarijus:	Aprašyti panaudos logiką. Prijungti trūkstamus resursus, naudotus logikoje. Saugoti panaudos logiką.
Alternatyvūs scenarijai:	Rankiniu būdu pakoreguojama logika, esanti modelyje. Modelis užkraunamas iš naujo.

Panaudos atvejis: **6. Vykdyti modelį**

Tikslas:	Aprašo procesą, kurio metu bet kuris vartotojas vykdo pasirinktą modelį.
Aktoriai:	Visi
Ryšiai su kitais PA:	Nustatyti logiką, Aprašyti resursai
Prieš-sąlyga:	Užkrautas modelis
Sužadinimo sąlyga:	Vartotojas kviečia pasirinkto modelio vykdymą
Po-sąlyga:	Modelis paleistas ir veikia.
Pagrindinis scenarijus:	Kviečiamas modelio valdymo interfeiso metodas, modelio vykdymui perduodant pasirinkto modelio vardą.
Alternatyvūs scenarijai:	Nėra

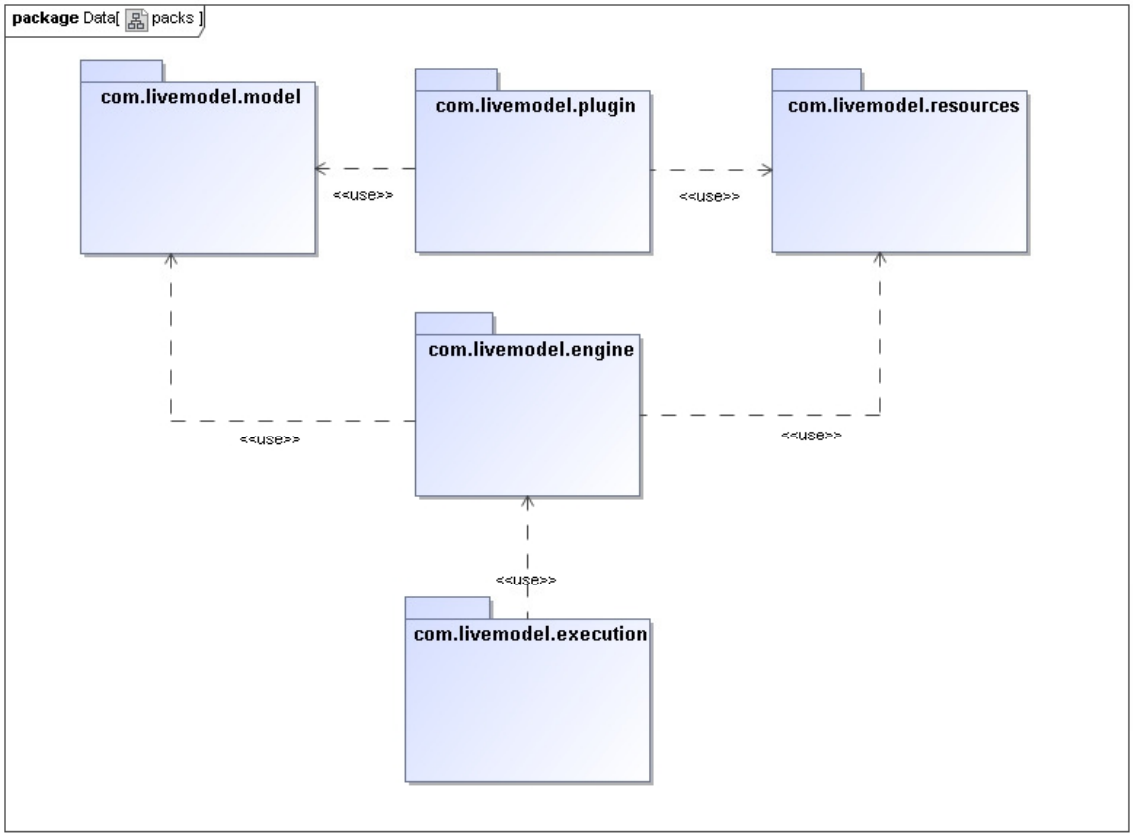
Panaudos atvejis: **7. Užkrauti modelį**

Tikslas:	Aprašo procesą, kurio metu vartotojas gali užkrauti (paruošti vykdymui) norimą modelį.
Aktoriai:	Visi
Prieš-sąlyga:	Paruoštas modelis ir resursai užkrovimui.
Sužadinimo sąlyga:	Vartotojas kviečia pasirinkto modelio užkrovimo vykdymą.
Po-sąlyga:	Užkrautas modelis, paruoštas vykdymui.
Pagrindinis scenarijus:	Kviečiamas modelio valdymo interfeiso metodas, modelio užkrovimui perduodant pasirinkto modelio vardą.
Alternatyvūs scenarijai:	Nėra

3.3.3 Loginis vaizdas

3.3.3.1 Apžvalga

Sistema suskirstyta į 5 pagrindinius paketus:

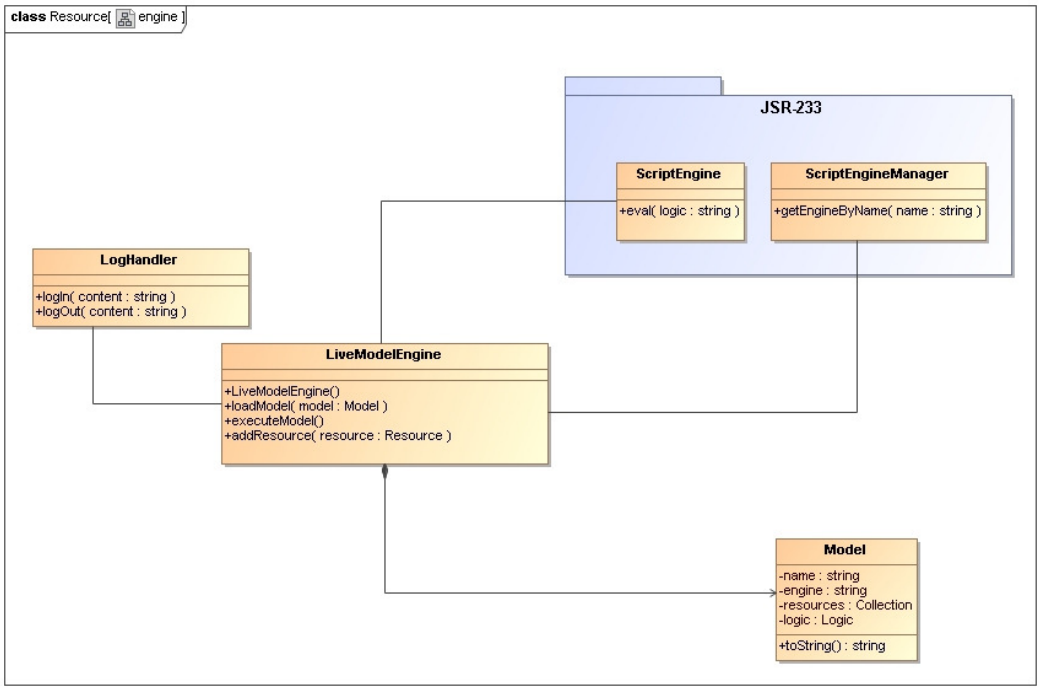


8 pav. Loginis sistemos vaizdas

Kiekvieno paketo trumpas paaiškinimas:

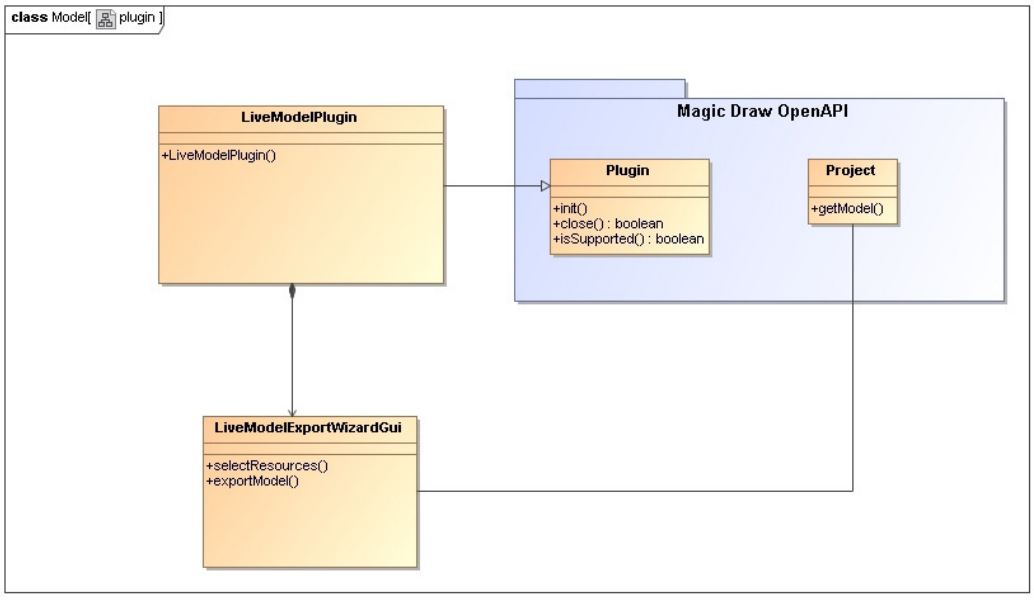
- ***Model*** – šis paketas atsakingas už modelio pilnumo valdymą sistemoje ir už jos ribų. Jame yra klasės, realizuojančios modelio ir jo sudedamųjų dalių struktūrą.
- ***Engine*** – šis paketas atsakingas už modelių valdymą. Jame yra klasės, realizuojančios modelio užkrovimą į vykdymo aplinką. Modelio vykdymo kontrolė.
- ***Resources*** – šis paketas apibrėžia skirtingus resursų tipus. Nustatomos bendros ir individualios resursų savybės.
- ***Execution*** – šis paketas atsakingas už modelio vykdymą skirtinguose kontekstuose operacinės sistemos lygmenyje.
- ***Plugin*** – pakete pateikiamos klasės, atsakingos už *MagicDraw LiveModel* veikimą. PIM transformacijos į PSM.

3.3.3.2 “Engine” paketas



9 pav. “Engine” paketo klasės

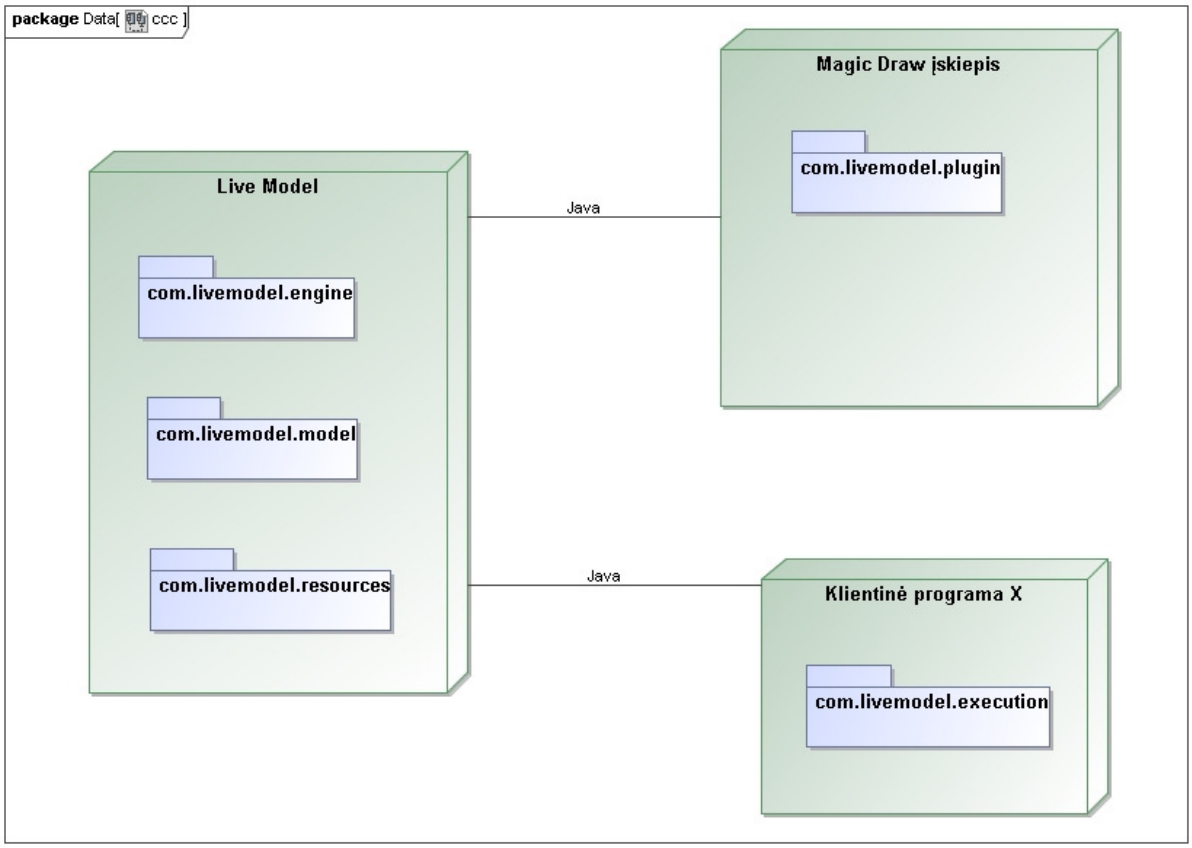
3.3.3.3 “Plugin” paketas



10 pav. “Plugin”paketo klasės

Šiame pakete saugomos klasės, skirtos LiveModel įskiepio MagicDraw 15 UML aplinkai, realizacijai aprašyti.

3.4 Realizacijos vaizdas



11 pav. Realizacijos vaizdas

3.4.1 MagicDraw sisteminiai reikalavimai

MagicDraw diegiamas Microsoft Windows, UNIX/Linux šeimos, MAC OS operacinėse sistemose.

Reikalavimai sistemai:

Procesorius: 1.5 GHz

RAM atmintis: 512 MB

Kietasis diskas: 2 GB

3.4.2 LiveModel sisteminiai reikalavimai

Live Model diegiamas bet kurioje operacinėje sistemoje, kuri turi Java palaikymą.

Reikalavimai sistemai:

Procesorius: 200 MHz

RAM atmintis: 512 MB

Kietasis diskas: 20 MB

3.4.3 Klientinės programos "X" sisteminiai reikalavimai

Reikalaujama, jog "*Klientinė programa X*" (Pavyzdžiui.: *TestTool 5*) veiktų *Java* pagrindu.

Tikslesni sisteminiai reikalavimai priklauso nuo konkrečios programos savybių, bet turi tenkinti minimalius "*LiveModel*" reikalavimus.

3.5 Projektinės dalies išvados

- Projektinė dalis buvo vystoma 3 semestrus (18 mėnesių), per kuriuos: pateikta projekto paraiška, sukurtas projekto planas, specifikuoti reikalavimai, atliktas architektūrinis bei detalus architektūrinis specifikavimas, suprogramuota ir ištestuota sistema, dokumentuota bei įdiegta.
- Projekto eigoje sukaupta svarbi patirtis, kuri leistų panašų projektą pradėti ir pabaigti per keletą kartų trumpesnį laiką.
- Programinės įrangos kūrimui naudojamas RUP (*Rational Unified Process*) programinės įrangos kūrimo procesas.
- Projekto rezultatas: sėkmingai sukurta ir įdiegta programinė įranga.

4 Tyrimo dalis

4.1 LiveModel modelių sudarymo MagicDraw aplinkoje eksperimentinis tyrimas

4.1.1 Tyrimo tikslas

Tyrimo tikslas – ištirti sukurtos sistemos metodologijos įsisavinimo procesą, apžvelgti rezultatus, pateikti rekomendacijos tobulinimui.

Tyrimo rezultatai turi atskleisti:

- Dokumentacijos tikslumą;
- Modelio kūrimo proceso suprantamumą;
- Trūkumus;
- Laiko sąnaudas, reikalingas pilnam proceso įsisavinimui.

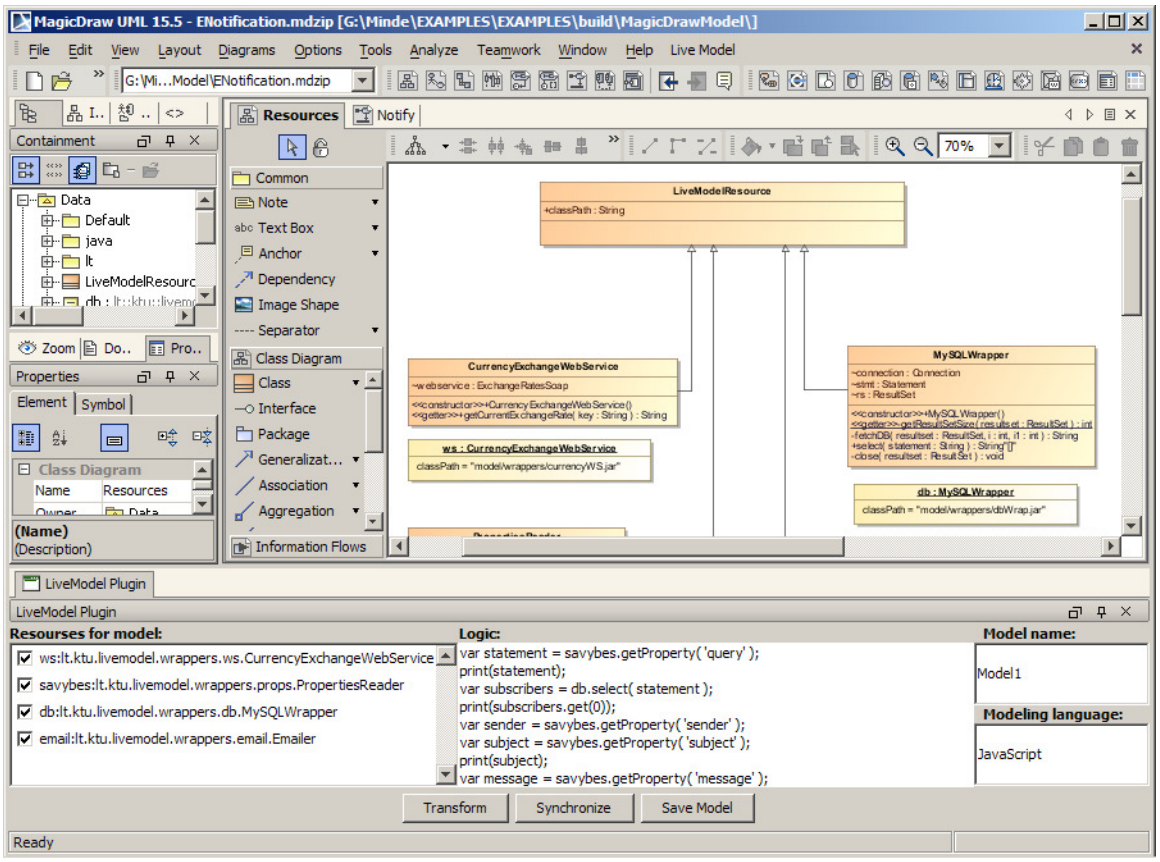
Tyrimą atliekama kitas asmuo, turintis aukštąjį magistro išsilavinimą informatikos srityje. Tyrėjui buvo pateikta tik programinė įranga, projekto dokumentacija, sukurta per 1-3 semestrus, bei užduotis. Tyrėjo buvo paprašyta sukurti atliktų žingsnių ir kuriamų konceptų aprašymą pačio tyrimo metu. Eksperimento išvados sudaromos kartu su tyrėju. Skyriai 4.1.2 - 4.1.6 aprašomi tyrėjo.

4.1.2 Apžvalga

LiveModel įskiepį, skirtą *MagicDraw,* randame įrankio meniu juostoje. Vartotojo sąsajai naudojama standartinė įrankio aplinka, papildyta *LiveModel* parametrų panele. Modeliavimui naudojamas 2 esminės *MagicDraw* UML diagramos – klasių (objektų) diagrama, leidžianti kurti nuo platformos nepriklausomą konceptualų modelį, bei veiklos diagrama, nusakanti veiklos logiką. Modeliavimą eigą būtų galima nusakyti taip:

1. Klasių modelio sudarymas (kartu ir naudojamų objektų sukūrimas.)
2. Panaudos logika užrašoma abstrakčiu modeliu naudojant veiklos diagramą.
3. Live model kodo generavimas, eksportavimas *xml* formatu norima programavimo kalba. Įrankis suprojektuotas palaikyti daugiau kaip 40 skirtingų programavimo kalbų pirmajai versijai realizuota *JavaScript*.

4. Eksportuoto modelio vykdymas.



12 pav. *MagicDraw* su *LiveModel* įskiepiu.

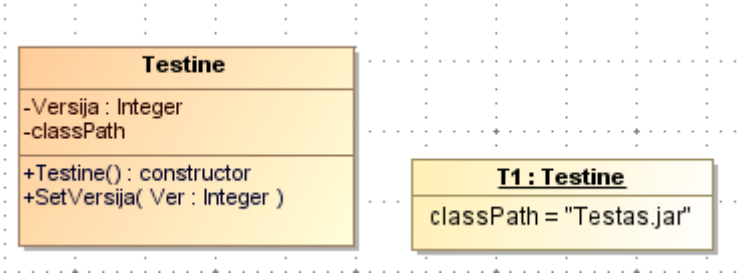
Toliau detaliau nagrinėjamas įskiepio panaudojimas.

4.1.3 Klasių diagramos sukūrimas

Pirmas etapas susideda iš konceptų (programinių klasių) paruošimo, kurie vėliau naudojami modeliavimui. Yra keli būdai aprašyti reikiamas klases:

- Naujų resursų sukūrimas naudojant *MagicDraw* atvirkštinės inžinerijos *"Reverse Engineering"* įrankius.
- Naujų klasių sukūrimas MagicDraw įrankio pagalba ir kodo generavimas. Modelio siejimas su fiziniais, egzistuojančiais kodo artefaktais.
- Jau realizuotų komponentų naudojimas, modelyje generuojant ar kuriant tik reikiamus atributus ir metodus ir siejant su fiziniais failais.

Norint klasę naudoti panaudos logikos modeliavime, būtina sukurti jos objektus. Modelio klasės objekto susiejimas su fizinėmis klasėmis, saugomomis Java archyve („*jar*"), atliekamas apsirašant vidinį klasės atributą *classPath*.



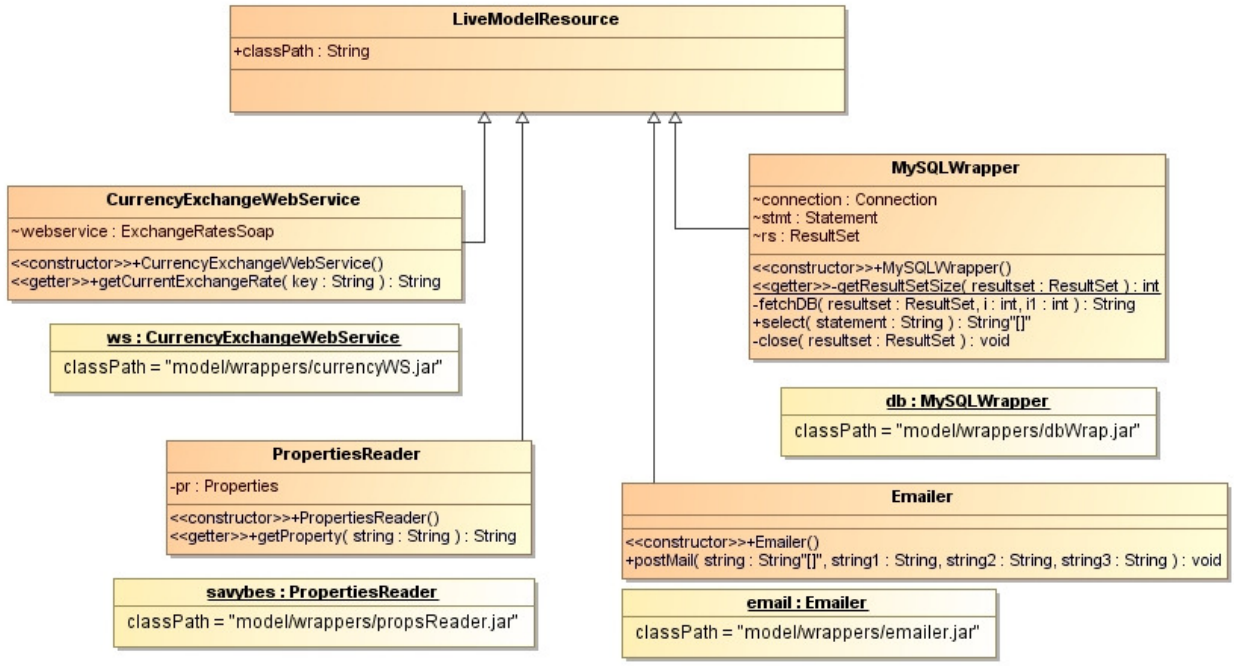
13 pav. Klasė ir jos objektas, su nustatytu parametru į fizinę realizaciją.

Objektų iš *LiveModel* resursų kūrimą apibrėžia UML elemento „*Instance Specification*" naudojimas. Šis elementas UML notacijoje atstovauja realaus objekto sukūrimą. Kai šis elementas yra sukurtas pasirinktai klasei, jį galima naudoti panaudos logikos kontekste kaip pilnavertį modelio resursą.

Kaip pavyzdys modeliuojama dalykinė sritis „Automatizuotas valiutų kursų siuntimas elektroniniu paštu nustatytiems gavėjams".

Lentelė 4. Resursų aprašai

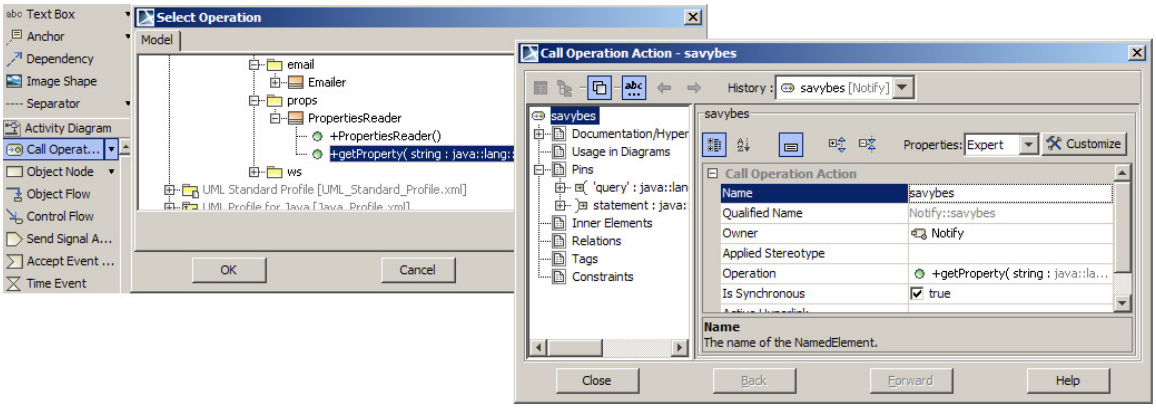
Pavadinimas	Paskirtis
CurrencyExchangeWebService	Klasė, naudojanti oficialią Lietuvos banko Web Service paslaugą valiutų tarpusavio santykiams gauti. Formuoja siunčiamą turinį.
Emailer	Klasė, naudojanti SMTP protokolą automatizuotam elektroninių laiškų siuntimui.
MySqlWraper	Klasė pagal nustatytus prisijungimo parametrus jungiasi prie duomenų bazės, reikalingos informacijos nuskaitymui.
PropertiesReader	Nustatymų failui skaityti skirta klasė. Programavimo dinamiškumui naudojamas tekstinis parametrų failas.



14 pav. Automatizuotas valiutų kursų siuntimas elektroniniu paštu. Klasių (objektų) modelis.

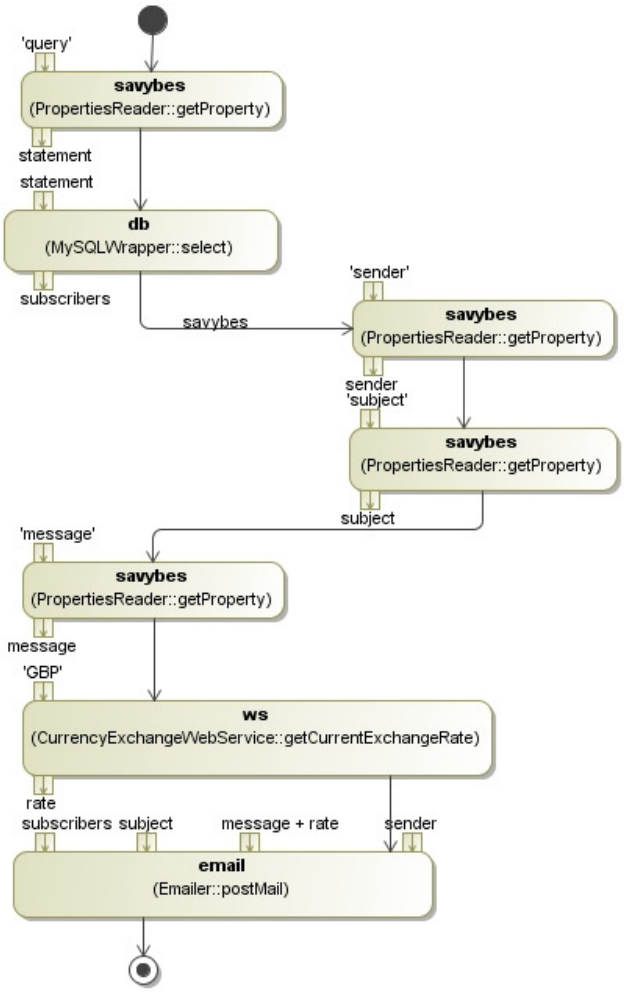
4.1.4 Panaudos logikos kūrimas

Antras etapas. Panaudos logika modeliuojama veiklos diagrama. Tam pritaikytas šios diagramos komponentas - *CallOperationAction,* reikalaujantis nurodyti, kokios klasės operacija (metodas) bus naudojama. Vėliau nustatomi metodų parametrų (Pins) vardai, jei reikia – galimos reikšmės.



15 pav. „*CallOperationAction*“ komponentais paremtas panaudos logikos modeliavimas.

Toliau pateikiamas sudarytas panaudos logikos modelis, apibrėžiantis, kaip turėtų funkcionuoti dalykinės srities realizacija.



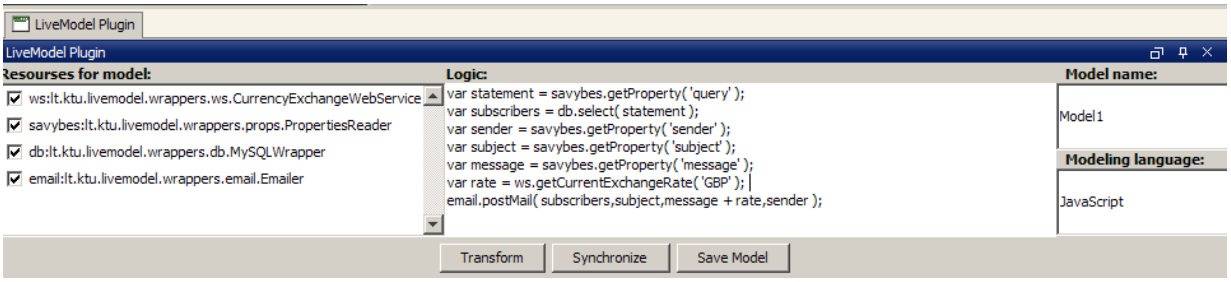
16 pav. Panaudos logikos modelis

Panaudos modelis ir realizuoja esminį *LiveModel* uždavinį – nesudėtingą atskirų komponentų modeliavimą. Pakartotinis objektinių programavimo komponentų panaudojimas ir galimybė panaudos atvejį modeliuoti grafiškai daro procesą aiškesnį, lengviau ir greičiau pakeičiamą.

4.1.5 Transformacija į kodą

Iš grafinio modelio generuojamas pasirinktos programavimo kalbos kodas (mygtukas *Transform*), kuris gali būti saugomas kaip modelio projektas ir toliau vykdomas *LiveModel* aplinkoje. *LiveModel* projekto išsaugojimas nėra susijęs su *MagicDraw* įrankiu, kuriamu

projektu ir neprideda jokios papildomos informacijos į MD saugomus projekto failus. Žemiau pateikti analizuojamam panaudos atvejui nustatyti parametrai ir sugeneruotas kodas.



17 pav. LiveModel parametrų parinkimas ir sugeneruotas kodas

Kodo generavimo taisyklės:

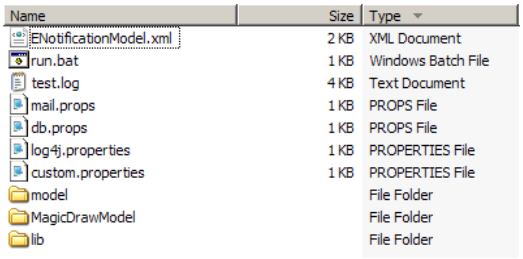
- Kiekviena metodo grąžinama reikšmė virsta lokaliu kintamuoju.
- Operacija su nurodytais kintamaisiais modeliuojama kaip procedūra.
- Transformacijos vykdomos nuosekliai pagal nustatytus ryšius.

4.1.6 Modelio vykdymas.

Transformuotas modelis saugomas *.xml* formatu. Prieš išsaugant transformaciją, palikta galimybė redaguoti gautą kodą (modelio vystymo etape tai naudinga savybė). Modelio vykdymo failai:

Lentelė 5. Klasių aprašai

Failų tipas	Paskirtis
<Modelio_pavadinimas>.xml	*LiveModel* projekto modelis.
run.bat	Konsolinė aplinka projekto vykdymui.
Test.log	Klaidų registro byla.
*.props, *.properties	Parametrų bylos.
.jar, arba reikiami katalogai	Kelias iki su modeliu susietų klasių.
katalogas lib	Kitos reikalingos bibliotekos



18 pav. Modelio vykdymui reikalingi failai

Modelio vykdymui naudojama vidinė *LiveModel*, „Execution" paketo klasė "*com.livemodel.engine.execution.LiveModel*".

4.1.7 Rekomenduojami tobulinimai

4.1.7.1 Tyrėjo rekomendacijos

- Plėsti transformacijų metodologiją realizuojant ciklų (*loop*) ir sąlygų (*fork*) panaudojimą.
- Tiesioginis modelio vykdymas iš įskiepio.
- Java archyvų suradimo ir priskyrimo resursui supaprastinimas.

4.1.7.2 Kūrėjo rekomendacijos

- Pritaikyti "*Open API*" idėjas sukurtų transformacijų prijungimui.
- *LiveModel* integravimo į kitas sistemas sudėtingumo lygio nustatymo tyrimas.
- Tikslinti projekto dokumentaciją.
- Plėsti UML modelio transformacijų skaičių *LiveModel* modeliams.

4.1.8 Bendros išvados

- Sukurtas pilnai veikiantis, apibrėžtos dalykinės srities modelis.
- Modelio kūrimas yra specifinis procesas ir reikalauja išankstinės dokumentacijos analizės.
- Pirminis modelio kūrimo ir naudojimo pagrindų įsisavinimas truko maždaug 4 valandas.
- Rezultatas sufleruoja, jog, toliau vystant šią technologiją, ji gali duoti didelės papildomos vertės daugeliui kuriamų sistemų.

5 Eksperimentinė dalis

5.1 Elektroninio mokymo eksperimentiniai tyrimai

Šie elektroninio mokymo eksperimentai buvo atlikti naudojant *TestTool 5* sistemą su integruota *LiveModel* aplinka.

5.1.1 Automobilių srautų judėjimo lygties mokomasis modelis

5.1.1.1 Tikslas

Praplėtus *TestTool 5* sistemą naujomis modeliavimo galimybėmis, atsirado galimybė kurti interaktyvius mokymosi modelius, paremtus laiko simuliavimu. Kaip vienas iš pavyzdžių paimta automobilių srautų judėjimo lygtis.

Literatūroje [23] aptariama viena iš daugelio siūlomų automobilių srautų judėjimo lygčių:

$$x_n''(t+T) = \lambda(x_{n-1}'(t) - x_n'(t)), \tag{1}$$

čia t - laikas, T- reakcijos laikas, λ - jautrumo parametras, $x_{n-1}'(t), x_n'(t)$ - atitinkamai *n-1*-ojo ir *n* - ojo automobilių greičiai laiko momentu *t*.

Pasinaudojant *TestTool 5* sistemos komponentais bei *LiveModel* sistemos galimybėmis, bus sukurta grafinė lygties vizualizacija.

Tyrimo rezultatai turi būti aptarti iš elektroninio mokymo ir modelių kūrimo metodologijos perspektyvų.

5.1.1.2 Sprendimas

Simuliacija vyksta *Autoriaus* programos aplinkoje. Lygtis paremta būsimo autotransporto priemonės pagreičio skaičiavimu laiko momentu *t*, žmogaus reakcijos laikotarpyje *T*. Konstanta *T* nusako vidutinę žmogaus reakcijos trukmę į jo aplinkoje vykstančius pasikeitimus (pavyzdžiui.: "*priekyje važiuojančio automobilio stabdymas"*). Statistinis šio parametro vidurkis 0.3 sekundės dalys.

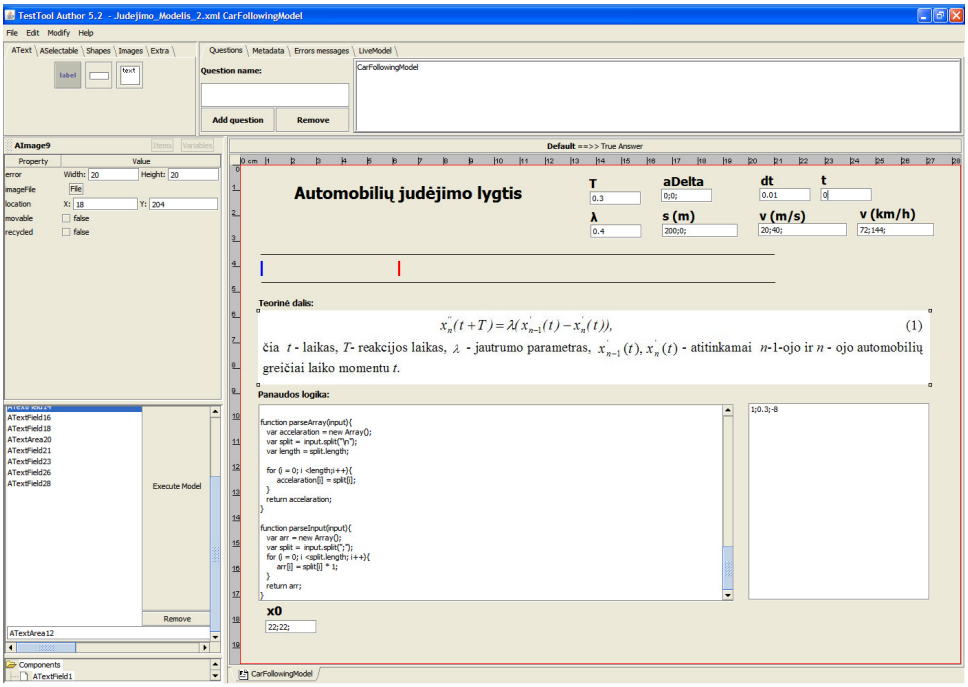
Simuliuojamu laiko momentu *t* kiekvienai transporto priemonei yra paskaičiuojamas būsimas pagreitis laiko momentu *t+T*. Iš judėjimo lygties aišku, jog pagreitis yra antro laipsnio nueito kelio išvestinė.

Lentelė 6. Modelio kintamųjų paskirtis ir reikšmės

Kintamasis / konstanta	Reikšmė	Paskirtis
dt	*0.01*	Simuliacijos vykdymo tikslumas. 100 milisekundžių.
λ	*0.4*	Jautrumo parametras, kuris parodo, kaip paskui važiuojanti mašina reaguoja į priekyje judančia mašiną.
T	*0.3*	Žmogaus vidutinis reakcijos laikas.
s	200;0;	Transporto priemonės pozicija nuo pradinio taško. Metrais.
v	*20;40;*	Matas: m/s Priekyje važiuojanti transporto priemonė važiuoja dukart lėčiau negu besivejanti.
t	*0 (pradinė)*	Einamasis simuliacijos laiko momentas.
aDelta	*0;0;*	Reguliuojamas pagreičio pokytis kiekvienam automobiliui. Matas:m/s.

Modelio vizualizacijoje automobilių atvaizdavimui naudojami du *ARectangle* komponentai (žr. *15 pav.*). Kaip jau minėta (2.2 Problemos), *TestTool 5* mokomųjų objektų kūrimui panaudojami unifikuoti grafiniai komponentai, kurie turi apibrėžtą savybių aibę (*pvz.*: ***pozicija, dydis, spalva*** *ir kt.*).

Iš viso šiame modelyje kiekvienai transporto priemonei suteikiamos 3 skirtingos savybės (atributai): ***pozicija***, ***greitis***, ir ***pagreitis***. Papildomoms savybėms įgyvendinti panaudojami *ATextField* ir *ATextArea* komponentai.



19 pav. Pradinė situacija

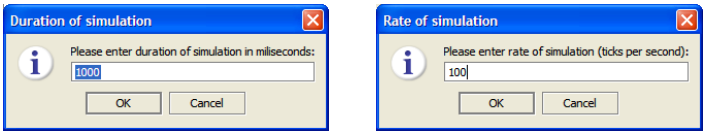
Pradiniu laiko momentu tarp automobilių yra 200 metrų skirtumas. Antrasis automobilis važiuoja dvigubai greičiau negu pirmasis.

Modelio panaudos logika pateikta priede. Žiūrėti skyrių "9.1 *Automobilių srautų judėjimo panaudos logika*".

5.1.1.3 Rezultatas

Modelio skaičiavimai vykdomi 100 milisekundžių intervalais (*dt*). Tai simuliacijos tikslumas.

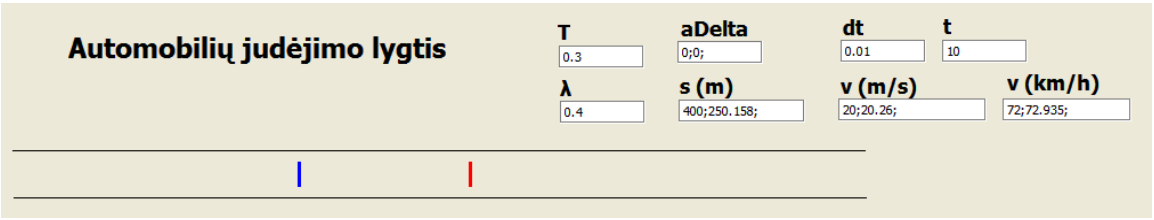
TestTool 5 aplinkoje simuliacijos vykdymui reikia nurodyti simuliacijos trukmę bei simuliacijų dažnį per sekundę.



20 pav. Simuliacijos trukmė ir dažnis

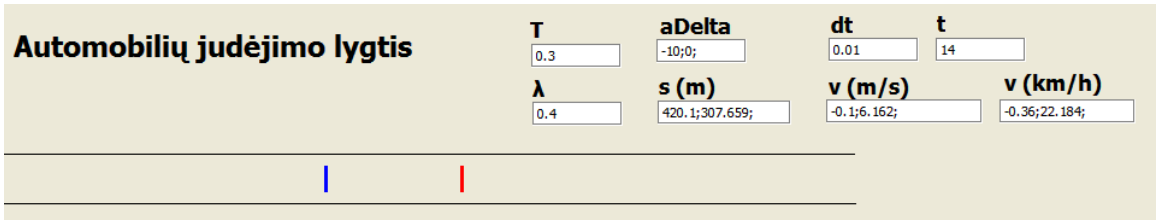
Tokiu būdu studentas gali kontroliuoti modelio vykdymo progresą. Tarpais tarp vykdymų galima keisti tarpinių skaičiavimų reikšmes (*greitis*, *pagreitis*).

Po 10 sekundžių simuliacijos vykdymo pirmasis automobilis (raudona) nuvažiavo 200 metrų (greitis pastovus –20 *m/s*). Antrasis automobilis nuvažiavo ~250 metrų, nes pradinis jo greitis buvo 40 *m/s*. Simuliacijos metu antrosios transporto priemonės vairuotojas nuolat mažino greitį, nes priekyje važiuojančio automobilio greitis buvo dvigubai mažesnis. 10-tosios sekundės pabaigoje abiejų automobilių greičiai beveik susilygino ir siekia ~72 *km/h*.



21 pav. Situacija po 10 sekundžių

Nusprendus sustabdyti priekyje važiuojantį raudonąjį automobilį, koreguojamas jo pagreitis keičiant *aDelta* parametrą (-10;0;). Atstumas tarp automobilių ~150 metrų.



22 pav. Situacija po 14 sekundžių

Kaip matome, raudonasis automobilis visiškai sustojo. Tuo pat metu mėlynasis sumažino greitį iki 22 *km/h*. Pratęsus simuliaciją nustatyta, jog pirmasis automobilis sustojo 20-ties metrų kelio atkarpoje, o mėlynasis - 70-ties metrų kelio ruože. Tarp automobilių buvo saugus atstumas, todėl mėlynasis automobilis stabdė tolygiai saugiame kelio ruože.

5.1.1.4 Išvados

- Pademonstruota galimybė modeliuoti sudėtingus matematinius modelius.

- Pademonstruotas komponentų savybių praplėtimas pasinaudojant kitais komponentais.
- Studentas gali valdyti simuliaciją ir keisti norimus modelio parametrus bei stebėti tų pakeitimų įtaką.
- Pasinaudojant šiuo modeliu, vairuotojams galima pademonstruoti, kaip yra svarbu laikytis saugaus atstumo nuo priekyje važiuojančios transporto priemonės.
- Modelio kūrimo metu gilinamos studento matematikos ir programavimo žinios.
- Modelio vystymo trukmė – 5 valandos.

5.1.2 Draudimo procesų simuliavimo mokomasis modelis

5.1.2.1 Tikslas

Sukurti matematinį-statistinį modelį, palengvinantį draudimo įmokų apskaičiavimą draudimo bendrovėms.

Įvertinti sukurto modelio tikslumą, taikymo galimybes. Apžvelgti potencialius pataisymus.

5.1.2.2 Sprendimas

Uždaviniui spręsti naudojamas *Monte Carlo* metodas. Tai pat panaudojamos statistinės vyrų ir moterų mirties tikimybės. Statistiniai duomenys gauti iš vienos didžiausių Lietuvos draudimo bendrovių. Dėl marketinginių priežasčių bendrovės pavadinimas neatskleidžiamas.
Draudimo modelis remiasi dviejų atsitiktinių dydžių generavimu. Vienas dydis naudojamas draudžiamojo lyčiai nustatyti, kitas – patikrinti, ar draudžiamasis mirė per einamus metus, atsižvelgiant į jo amžių.

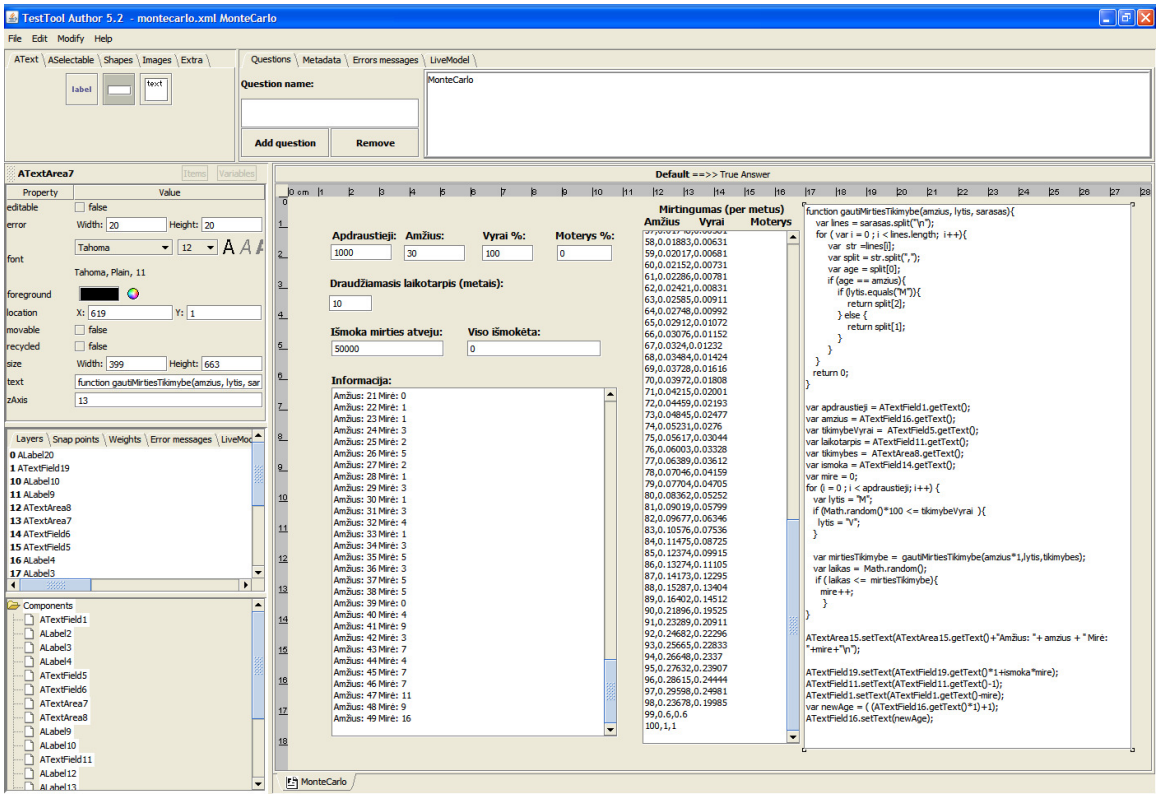
Lentelė 7. Modelio kintamųjų paskirtis

Kintamasis/konstanta	Paskirtis
Apdraustieji	Modeliuojamas apdraustųjų skaičius.
Amžius	Pradinis gyventojo amžius
Vyrai %	Procentinė vyrų dalis

Moterys %	Procentinė moterų dalis
Draudžiamasis laikotarpis metais	
Išmoka mirties atveju	Išmokama pinigų suma vieno apdraustojo mirties atveju
Viso išmokėta	Bendra išmokėta draudimo suma.

5.1.2.3 Rezultatas

Sukurtas modelis vykdomas *n* kartų. Čia *n* – draudžiamas laikotarpis. Kasmet kiekvienam draudžiamajam bandoma pritaikyti mirties tikimybę. Jeigu apdraustasis miršta, bendras apdraustųjų skaičius automatiškai mažinamas.



23 pav. 50-ties metų draudžiamojo laikotarpio pavyzdys

Iš aukščiau pateikto paveikslo matome, jog penkiasdešimtaisiais metais mirė 16 apdraustųjų.

Panaudos logika pateikta priede. 9.2 Draudimo procesų modelio panaudos logika

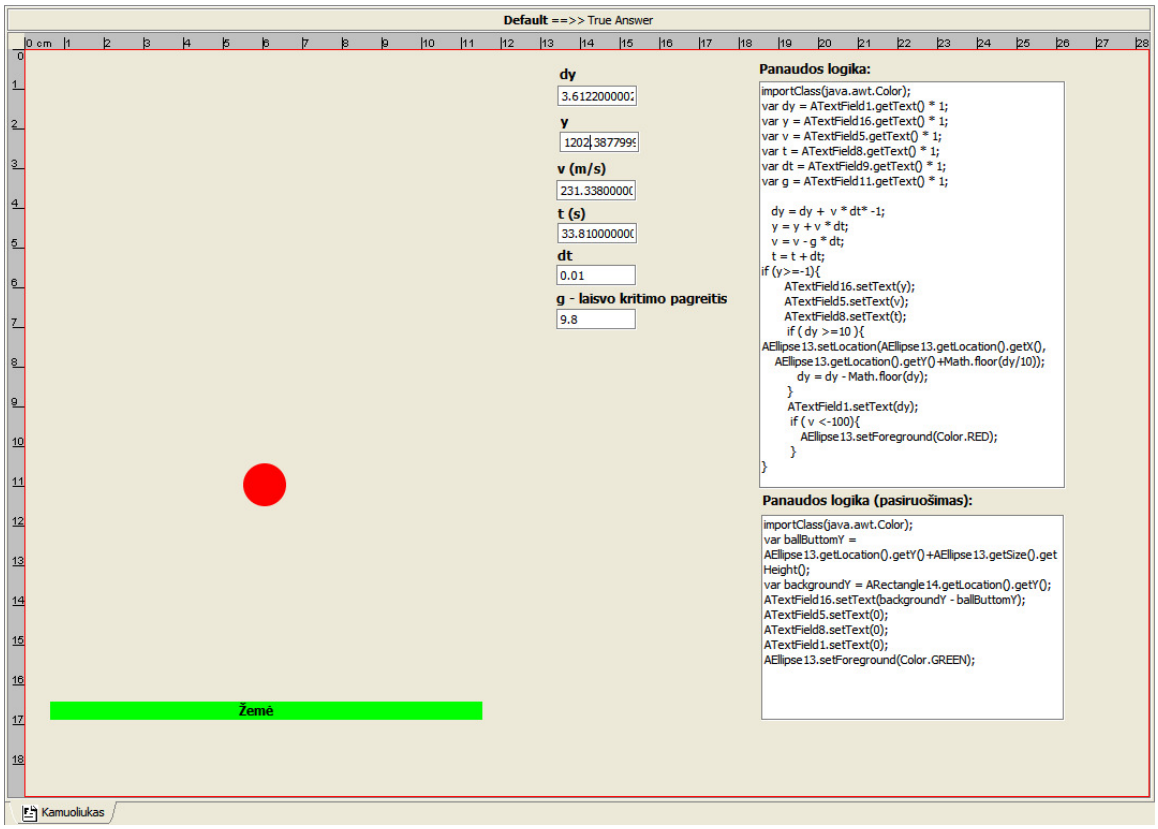
5.1.2.4 Išvados

- Pademonstruota galimybė modeliuoti draudimo procesus.
- Galimybė lanksčiai keisti modelio parametrus suteikia vartotojui daugiau panaudojimo galimybių.
- Dėl naudojamo metodo paklaidų modelį būtina vykdyti keletą kartų. Tikslumo įvertinimui siūloma skaičiuoti vidutinę kvadratinę paklaidą (MSE).
- Modelį galima patobulinti automatiškai apskaičiuojant įmoką, kurią apdraustasis turi sumokėti metų laikotarpyje.
- Besimokančiajam modelis leidžia suprasti, kaip atliekamos simuliacijos naudojant atsitiktinius dydžius, *Monte Carlo* metodą.
- Modelis sukurtas ir ištestuotas per 2 valandas.

5.1.3 Laisvojo kritimo pagreičio mokomasis pavyzdinis modelis

5.1.3.1 Pavyzdys

Demonstruojamas laisvo kritimo modelio veikimas *TestTool 5* sistemoje.



24 pav. Laisvojo kritimo modelis

Kamuoliuko nueito kelio pokytis perskaičiuojamas kiekvieno modelio vykdymo metu žingsniu *dt (0.01)*.

Panaudos logika pateikiama priede: ***9.3 Laisvo kritimo modelio panaudos logika***

5.1.3.2 Išvados

- Šis mokomasis objektas turi dvi skirtingas panaudos logikas (paruošimo ir kamuoliuko kritimo).
- Studentas gali keisti kamuoliuko aukštį padėdamas jį į norimą poziciją virš žemės paviršiaus.
- Studentas gilina fizikos, matematikos bei programavimo žinias.
- Kadangi modelį reikia tobulinti, norint gauti tikslesnius modeliavimo rezultatus, tai gali būti puiki programavimo-optimizavimo užduotis studentui. Pavyzdžiui, laisvo

kritimo pagreičio reikšmės įvertinimas skirtinguose aukščiuose virš žemės paviršiaus.

- Modelį reikia tobulinti, jei norima simuliuoti keleto kūnų kritimą vienu metu. Tai taip pat gali būti gera užduotis studentui.
- Tyrimas atskleidė, jog ateityje TestTool integracijos sprendimą būtų galima praplėsti skirtingų modelių vykdymu vienu metu.
- Modelis sukurtas ir ištestuotas per 1 valandą.

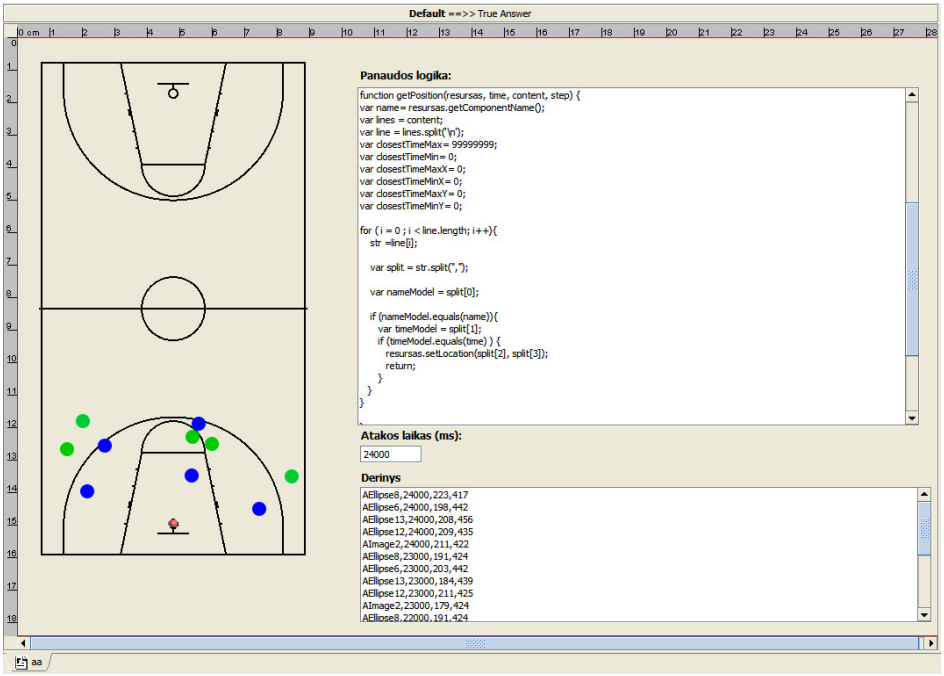
5.1.4 Krepšinio derinių sudarymo ir demonstravimo pavyzdinė aplinka

5.1.4.1 Pavyzdys

Šis modelis skirtas krepšinio derinių sudarymui ir vaizdavimui.

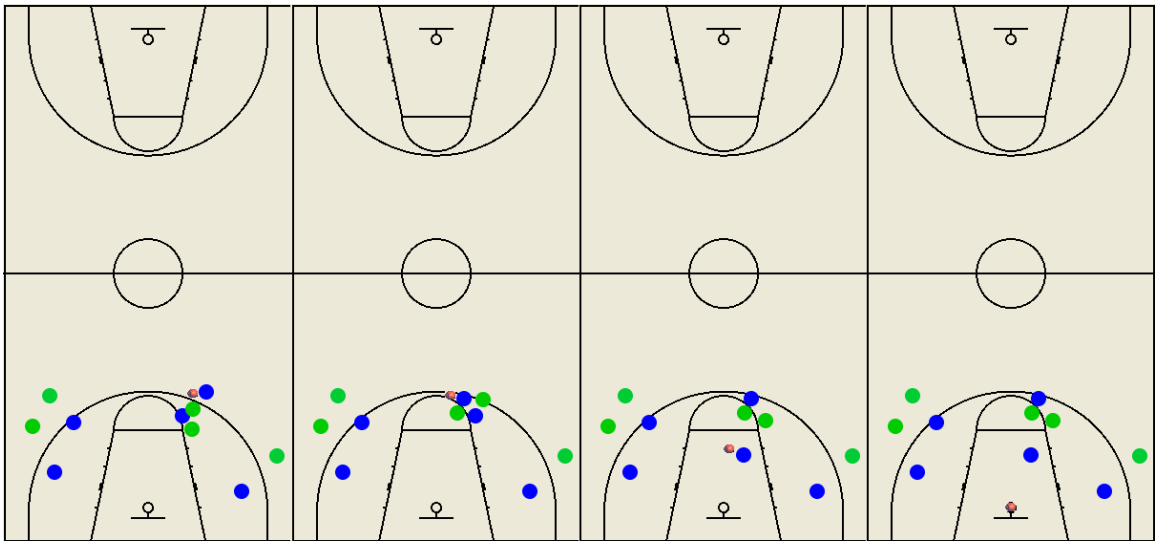
Modelyje palaikomi 2 skirtingai režimai:

- Derinio sudarymas
- Derinio vaizdavimas



25 pav. Krepšinio derinių kūrimas ir vaizdavimas

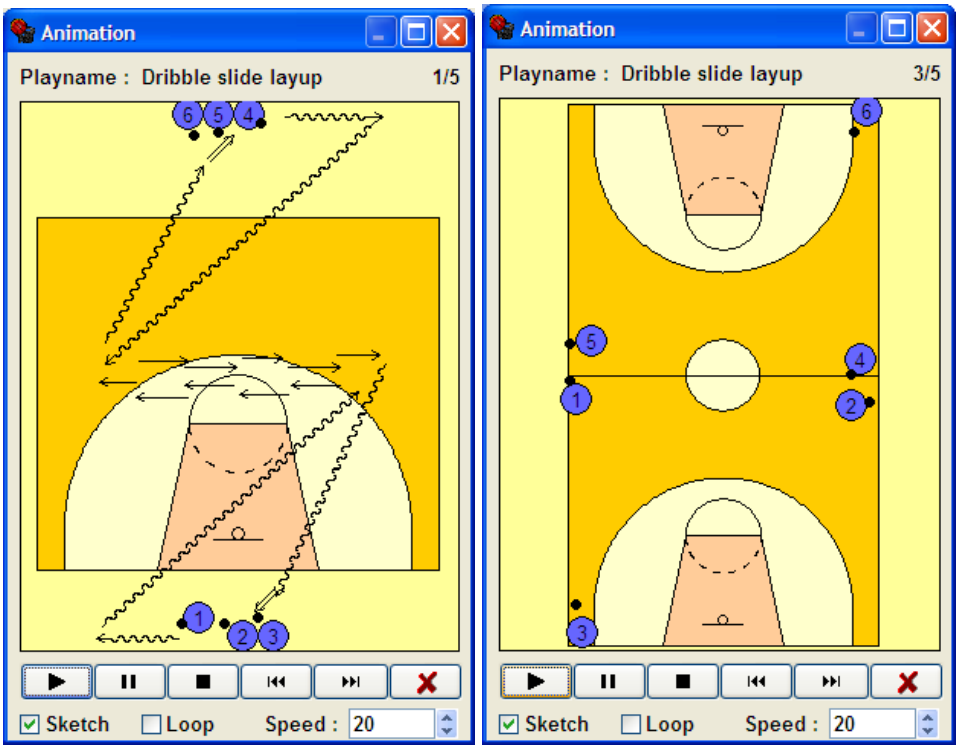
Modelio vykdymas paremtas 24 sekundžių ataka. Tai laiko pokyčiu grįsta simuliacija. *24 paveiksle* pateikiamas 4 sekundžių "du-prieš-du" derinys.



26 pav. Krepšinio derinių 4 sekundžių simuliacija

5.1.4.2 Sprendimų Palyginimas

Sukurtas modelis palyginamas su "*Basketball Playbook 0.8*" krepšinio derinių kūrimo ir vizualizavimo paketu. Palyginamos tik esminės savybės.



27 pav. Krepšinio derinių kūrimas ir vaizdavimas "*Basketball Playbook 0.8*" pakete

Lentelė 8. Modelio kintamųjų paskirtis ir reikšmės

Savybė	Basketball Playbook 0.8	TestTool model
Žaidėjų judėjimas nustatomas pasinaudojant linija	+	- TestTool sistema turi komponentą - linija. Nebūtų sudėtinga praplėsti modelio koordinačių saugojimo funkcionalumą pagal nubrėžtas linijas.
Vykdymo valdymas (pauzė, tęsimas, paleidimas nuo tam tikro momento)	+	+
Judėjimas	+	+- TestTool modelyje žaidėjų judėjimas tiesiogiai priklauso nuo komponentų judėjimo tikslumo. Norint, kad žaidėjai pastoviai judėtų simuliacijos metu, reiktų patobulinti modelį.
Simuliacijos greičio reguliavimas	+	+
Modelio keitimo lankstumas	- Šis paketas yra specializuota darbo priemonė, todėl vartotojui nesuteikiama jokių įrankio veikimo tobulinimo galimybių.	+ *TestTool* modelis gali būti lengvai tobulinamas, vystant sudėtingesnius sprendimus. Pavyzdžiui, besimokančio modelio kūrimas.

5.1.4.3 Išvados

- Pademonstruota galimybė kurti ir vaizduoti dinamiškus krepšinio derinius TestTool 5 sistemoje.
- Modelį reikia tikslinti, norint išgauti pastovų žaidėjų judėjimą. Puiki užduotis besimokančiam studentui.
- Derinio demonstravimas priešinga (laikui) kryptimi būtų puikus modelio patobulinimas.
- Nesudėtinga pakeisti išsaugotų koordinačių formatą integracijos su kitomis sistemomis atveju.
- Sukurta aplinka palengvina krepšinio derinių kūrimo procesą.
- Puiki galimybė vystyti "besimokantį" krepšinio modelį.
- Atliktas grubus pagrindinių funkcijų palyginimas tarp TestTool modelio ir "Basketball Playbook 0.8".
- "Basketball Playbook 0.8" yra labiau tinkamas masiniam vartojimui, bet reikia pabrėžti faktą, jog TestTool modelio kūrimas truko maždaug 3-4 valandas. Patobulinus modelį pagal aukščiau pateiktas išvadas, kokybės santykis taptų panašus.

6 Išvados

1. Naujas atskirų resursų panaudos modelių vykdymo karkasas *LiveModel* bei išvystyta panaudojimo metodologija suteikia naujų modelių kūrimo, taikymo ir vizualizavimo galimybių elektroninio mokymo sistemose.

2. Darbo metu atliktas eksperimentinis *LiveModel* karkaso pritaikymas egzistuojančios sistemos *TestTool 5* aplinkai leido pademonstruoti naujas dinaminių modelių kūrimo galimybes ir taikymus e-mokymo srityje.

3. Sukurtas *LiveModel* įskiepis *MagicDraw 15* platformai leidžia efektyviai išnaudoti šios modeliavimo sistemos galimybes kuriant resursų panaudos modelius.

4. Panaudos logikos generavimui naudojamos UML grafinio modeliavimo kalbos klasių (*Class*) ir veiklos (*Activity*) diagramos, iš kurių atliekamos transformacijos į kodą. Tai padidina sprendimo gavimo efektyvumą.

5. *TestTool 5* sistemai sukurti eksperimentiniai mokymo modeliai rodo, kad sistema gali tarnauti kaip modeliavimo aplinka sudėtingesniems moksliniams tyrimams vykdyti.

6. Kito asmens atliktas sukurtos sistemos naudojimo tyrimas atskleidė tik minimalius sistemos ir išvystytos metodologijos trūkumus.

7. Eksperimentiniai tyrimai įrodė, jog mokomųjų modelių kūrimo trukmė juntamai trumpesnė, negu tokių pat modelių kūrimas nuo nulio.

7 Literatūra

[1] Marcus Specht, Milos Kravcik, (2006) *Athoring of Learning Objects in Context,* International Journal on E-Learning, Vol.5, Issue 1, 2006

[2] *Component-based software engineering* [Žiūrėta 2007 11 09], prieiga internete [http://en.wikipedia.org/wiki/Component-based_software_engineering]

[3] Mark Baker and Rahim Lakhoo, (2006) *Application Reuse through Portal Frameworks*, prieiga internete [www.allhands.org.uk/2006/proceedings/papers/693.pdf]

[4] Aukstakalnis, N. Baniulis, K. Pauliute, J. Slotkiene, A. , (2008) *Graphical model: The means for simulation-based learning*,
Information Technology Interfaces, ITI 2008. 30th International Conference. (p. 471-476)

[5] JSR-000223 Scripting for the JavaTM Platform [Žiūrėta 2007 11 09], prieiga internete [http://jcp.org/aboutJava/communityprocess/pr/jsr223/]

[6] Scripting Support in Mustang by Paul Jensen, Partner;
Object Computing, Inc. (OCI) [http://www.ociweb.com/jnb/jnbFeb2006.html]
[Žiūrėta 2007 11 09]

[7] A Pattern-Based Approach for the Consistent Design of Interaction Interfaces [http://www.diva-portal.org/ntnu/abstract.xsql?dbid=1089] [Žiūrėta 2007 11 10]

[8] An Approach to Developing Extensible Application Composition Environments for End Users
[http://www.diva-portal.org/ntnu/abstract.xsql?dbid=736] [Žiūrėta 2007 11 11]

[9] Bite Qiu; Xu Han; (2006) *Modeling support for Application Families*. Tezės. [Žiūrėta 2007 11 15]
[http://www.diva-portal.org/vxu/abstract.xsql?dbid=979]

[10] World Wide Web Consortium (WC3) [Žiūrėta 2007 11 09], prieiga internete [http://www.w3.org/]

[11] Scripting Engines and Scripting Applications. [Žiūrėta 2007 11 16], prieiga internete [https://scripting.dev.java.net/]

[12] Development of a Demand Driven Dom Parser. [Žiūrėta 2007 11 16], prieiga internete [http://www.diva-portal.org/ntnu/abstract.xsql?dbid=1298]

[13] RELAX NG Specification [Žiūrėta 2007 11 16], prieiga internete [http://relaxng.org/spec-20011203.html]

[14] Document Object Model (DOM) [Žiūrėta 2007 11 16], prieiga internete [http://www.w3.org/DOM]

[15] Calvin J. Ribbens, Srinidhi Varadarajan, Naren Ramakrishnan, *A Runtime Framework for Adaptive Compositional Modeling*. Tezės. [Žiūrėta 2007 11 16], prieiga internete [http://scholar.lib.vt.edu/theses/available/etd-05162004-212101]

[16] Kendall Scott. Fast Track UML. 2004.

[17] Reikalavimų specifikacijos dokumentas [http://elgrid.ktu.lt/~mindaugas/modelweb/skimelis_spec.doc]

[18] Magic Draw UML. [http://www.magicdraw.com/]

[19] An introduction to Model Driven Architecture [Žiūrėta 2007 11 09], prieiga internete [http://www.ibm.com/developerworks/rational/library/3100.html]

[20] Platform-independent model [Žiūrėta 2007 11 09], prieiga internete [http://en.wikipedia.org/wiki/Platform-independent_model]

[21] Electronic learning, [Žiūrėta 2009 03 09], prieiga internete [http://en.wikipedia.org/wiki/E-learning]

[22] Platform-specific model [Žiūrėta 2007 11 09], prieiga internete [http://en.wikipedia.org/wiki/Platform-specific_model]

[23] Weng Yan-lin1 Wu Tie-jun1, *Car-following models of vehicular traffic*, Journal of Zhejiang University - Science A, Vol. 3, Number 4 / September, 2002, (p. 412 – 417)

8 Terminų ir santrumpų žodynas

Learning object – mokomasis objektas.

MDA – modeliu paremta architektūra (***Model driven architecture***).

PIM – nuo platformos nepriklausomas modelis (***Platform independent model***).

PSM – platformai specifinis modelis (***Platform specific model***).

TestTool 5 – nuotolinio testavimo sistema kuriama ir tobulinama KTU, Informatikos fakultete.

WS (*Web Service*) – tai programinės įrangos komponentai, sukurti tam, kad leistų kurti kintamo dydžio, laisvai susietas, nepriklausančias nuo platformos programas.

WC3 (*World Wide Web Consortium*) - pasaulinio žiniatinklio konsorciumas.

WWW (*World Wide Web*) - pasaulinis interneto tinklas.

XML (*eXtensible Markup Language*) - W3C rekomenduojama bendros paskirties ženklinimo kalba, skirta specialios paskirties ženklinimo kalbų kūrimui bei duomenų keitimuisi tarp skirtingų sistemų.

DOM (*Document Object Model*) - Nuo platformos ir kalbos nepriklausomas objektinis modelis.

JRE (*Java Runtime Environment*) – Java vykdymo aplinka.

SDK (*Software Development Kit*) – Programinės įrangos kūrimo rinkinys.

Open API – atvira aplikacijų programavimo sąsaja.

JSR-223 (*Java Specification Request*) – naujo projekto užklausa, inicijuojama Java bendruomenės narių.

RUP (*Rational Unified Process*) – IBM kompanijos patirtimi paremta metodika programų kūrimo procesui organizuoti.

OMG (Object Management Group) – OMG™ yra tarptautinė, atvira ir nepelno siekianti organizacija, kuri rūpinasi modeliavimo standartų palaikymų ir plėtojimu.

JavaOne – kasmet vykstanti konferencija, orientuota į naujovių vystymą Java platformoje.

9 Priedai

9.1 Automobilių srautų judėjimo panaudos logika

```
var t = ATextField23.getText() * 1;
var T = ATextField1.getText() * 1;
var l = ATextField3.getText() * 1;
var dt = ATextField18.getText() * 1;
var vArr =  parseInput(ATextField21.getText());
var aArr =  parseInput(ATextField14.getText());
var sArr =  parseInput(ATextField16.getText());
var arrayAcc =  parseArray(ATextArea20.getText());

function findAcc(array, n, tt){
    for (ii = 0; ii < array.length ; ii++){
      var text = array[ii];
       var splitI =text.split(";");
       if (splitI[0]==n && splitI[1] == tt ) {
           return splitI[2] * 1;
       }
   }
  return 0;
}

var totalCar = 2;
for ( i = 0 ; i < totalCar; i++) {
   var a = findAcc(arrayAcc, i, t);
   if (vArr[i] <0){
      continue;
   }
   sArr[i] = sArr[i] + vArr[i] * dt;
   vArr[i] = vArr[i] + a * dt +aArr[i] * dt;
}
var newV = "";
var newVkm = "";
var newS = "";
for ( i = 0 ; i < totalCar; i++) {
   newV = newV + Math.round(vArr[i]*1000)/1000 + ";";
   newS = newS + Math.round(sArr[i]*1000)/1000 + ";";
   newVkm  = newVkm + Math.round(((vArr[i]*3600)/1000)*1000)/1000 + ";";
}
ATextField21.setText(newV);
ATextField16.setText(newS);
ATextField28.setText(newVkm);

for ( i = 1 ; i < totalCar; i++) {
```

```
ATextArea20.setText(i+";"+(Math.round((t+T)*100)/100)+";"+Math.round((l*(vArr[i-1]-
vArr[i]))*10000)/10000+"\n"+ATextArea20.getText());
}

t = Math.round((t + dt)*100)/100;
ATextField23.setText(t);

if (arrayAcc.length > 100) {
   var newArr = parseArray(ATextArea20.getText());
   ATextArea20.setText("");
   for (j = Math.round((T/dt)*1)/1; j >0; j--){
      ATextArea20.setText(newArr[j] + "\n"+ ATextArea20.getText());
   }
}

var x0 = ATextField26.getText().split(";");
ARectangle6.setLocation(Math.floor(x0[0]*1+sArr[0]),ARectangle6.getLocation().getY());
ARectangle5.setLocation(Math.floor(x0[1]*1+sArr[1]),ARectangle5.getLocation().getY());


function parseArray(input){
   var accelaration = new Array();
   var split =  input.split("\n");
   var length = split.length;

   for (i = 0; i <length;i++){
        accelaration[i] = split[i];
   }
   return accelaration;
}

function parseInput(input){
   var arr = new Array();
   var split =  input.split(";");
   for (i = 0; i <split.length; i++){
        arr[i] = split[i] * 1;
   }
   return arr;
}
```

9.2 Draudimo procesų modelio panaudos logika

```
function gautiMirtiesTikimybe(amzius, lytis, sarasas){
    var lines = sarasas.split("\n");
    for ( var i = 0 ; i < lines.length;  i++){
         var  str =lines[i];
         var split = str.split(",");
```

```
            var age = split[0];
            if (age == amzius){
                if (lytis.equals("M")){
                    return split[2];
                } else {
                    return split[1];
                }
            }
        }
       return 0;
}

var apdraustieji = ATextField1.getText();
var amzius = ATextField16.getText();
var tikimybeVyrai =  ATextField5.getText();
var laikotarpis = ATextField11.getText();
var tikimybes =  ATextArea8.getText();
var ismoka = ATextField14.getText();
var mire = 0;
for (i = 0 ; i < apdraustieji; i++) {
   var lytis = "M";
   if (Math.random()*100 <= tikimybeVyrai  ){
     lytis = "V";
   }

   var mirtiesTikimybe =  gautiMirtiesTikimybe(amzius*1,lytis,tikimybes);
   var laikas =  Math.random();
    if ( laikas <=  mirtiesTikimybe){
      mire++;
       }
}

ATextArea15.setText(ATextArea15.getText()+"Amžius: "+ amzius + " Mirė: "+mire+"\n");

ATextField19.setText(ATextField19.getText()*1+ismoka*mire);
ATextField11.setText(ATextField11.getText()-1);
ATextField1.setText(ATextField1.getText()-mire);
var newAge = ( (ATextField16.getText()*1)+1);
ATextField16.setText(newAge);
```

9.3 Laisvo kritimo modelio panaudos logika

9.3.1 Modelio logika

```
importClass(java.awt.Color);
```

```
var dy = ATextField1.getText() * 1;
var y = ATextField16.getText() * 1;
var v = ATextField5.getText() * 1;
var t = ATextField8.getText() * 1;
var dt = ATextField9.getText() * 1;
var g = ATextField11.getText() * 1;

   dy = dy +  v * dt* -1;
   y = y + v * dt;
   v = v - g * dt;
   t = t + dt;
if (y>=-1){
       ATextField16.setText(y);
       ATextField5.setText(v);
       ATextField8.setText(t);
        if ( dy >=10 ){
AEllipse13.setLocation(AEllipse13.getLocation().getX(),
AEllipse13.getLocation().getY()+Math.floor(dy/10));
          dy = dy - Math.floor(dy);
        }
        ATextField1.setText(dy);
         if ( v <-100){
            AEllipse13.setForeground(Color.RED);
         }
}
```

9.3.2 Paruošimo logika

```
importClass(java.awt.Color);
var ballButtomY = AEllipse13.getLocation().getY()+AEllipse13.getSize().getHeight();
var backgroundY = ARectangle14.getLocation().getY();
ATextField16.setText(backgroundY - ballButtomY);
ATextField5.setText(0);
ATextField8.setText(0);
ATextField1.setText(0);
AEllipse13.setForeground(Color.GREEN);
```

9.4 Straipsnis

Testavimo sistemos adaptavimas matematinių žinių testavimui

Rasa Vaškevičiūtė, Mindaugas Škimelis
Kauno technologijos universitetas
Studentų 50-416

Anotacija

Straipsnyje iškeliamas uždavinys adaptuoti *TestTool* nuotolinio žinių testavimo sistemą nuodugniam matematinių žinių testavimui. Analizuojami plačiausiai taikomi integruojami matematiniai paketai, sukurti Java technologijos pagrindu, t.y. *JEP*, *JbcParser*, *Mathematica* bei *WebEQ*. Atskirai aptariamos kiekvieno paketo savybės, pateikiami pagrindiniai privalumai ir trūkumai. Parenkamas paketas, kuris labiausiai tinka iškeltam uždaviniui spęsti.

Įvadas

Nuotolinio testavimo sistema *TestTool (TT)*, sėkmingai naudojama ne tik Kauno technologijos universitete (KTU), bet ir už jo ribų, iki šiol nebuvo galima nuodugniai testuoti matematinių žinių, kadangi nebuvo galimybės įvertinti išraiškos pagal jos prasmę, taip pat nebuvo galimybės grafiškai įvesti ir atvaizduoti matematines išraiškas. Taigi reikalinga integruoti papildomą modulį, suteiksiantį *TT* papildomas savybes, leisiančias išsamiai tikrinti testuojamų asmenų žinias matematikos srityje.

Šiuo metu KTU matematinių gebėjimų testavimui naudojama *EDU Campus* [5] sistema, suskurta *Brownstone* tyrimų grupės. Šios sistemos vienas iš pagrindinių minusų yra tas, kad klausimai yra programuojami MathML kalba, o tai reikalauja specialiųjų žinių, kurioms įgyti reikalingas papildomos studijos. Dėl šios priežasties *EDU Campus* vartotojams (dėstytojams) reikia skirti papildomo laiko ir pinigų keltis kvalifikaciją. Dėl minėtos priežasties *EDU Campus* tampa mažiau patraukli sistema. Norint suteikti galimybę fundamentaliųjų mokslų fakultete naudotis ne tik minėtąja sistema, bet ir KTU sukurta *TT,* nuspręsta analizuoti rinkoje esamus matematinius paketus, išrinkti tinkamiausią ir praplėsti *TT* funkcionalumą taip, kad būtų galima grafiškai įvesti matematines išraiškas, jas vaizduoti ir palyginti pagal prasmę. Taip pat vienas pagrindinių tokio išplėtimo reikalavimų yra tas, kad kuriant klausimus nereikėtų išmanyti jokios programavimo kalbos.

Šiame straipsnyje pateikiama keturių populiariausių rinkoje esančių matematinių paketų analizė ir įvertinamas jų tinkamumas nuotolinio testavimo sistemai *TT*.

Matematiniai paketai skirti matematinei duomenų analizei

JEP

Java Math Expession Parser(JEP) [1] – tai matematinių išraiškų konvertavimo ir skaičiavimo atviro kodo biblioteka, suderinama su Java 1.2 versija. Į *JEP* yra įtraukta daug įprastų funkcijų bei konstantų tokių kaip π ir *e*. Palaikomi *Unicode* simboliai (taip pat ir graikiškos raidės), kas ypač aktualu matematikoje.

JEP yra suprantami visi pagrindiniai loginiai ir aritmetiniai operatoriai, tačiau skirtingiems duomenų tipams galimos ne visos operacijos. *JEP* išskiriami keturi duomenų tipai: realieji (angl. *double*), kompleksiniai skaičiai, simbolių eilutė (angl. *string*) ir vektoriai. Realiesiems skaičiams galimos visos loginės ir aritmetinės operacijos, o kompleksiniams skaičiams negalimos visos loginės operacijos išskyrus: loginį „ne“ (!) „lygu“ (==) ir „nelygu“ (!==). Simbolių eilutei leidžiamos tik dvi operacijos: sudėtis ir „lygu“, „nelygu“; vektoriams priskirtos tik daugyba ir dalyba.

Matematinės funkcijos yra skirstomos į standartines ir skirtas kompleksiniams skaičiams. Standartinėms funkcijoms priklauso pagrindinės trigonometrinės ir algebrinės funkcijos tokios kaip: sinusas, hiperbolinis kosinusas, natūrinis logaritmas, eksponentė, kvadratinė šaknis, suma ir t.t. – visas šias funkcijas galima panaudoti su realiaisiais skaičiais, beveik visos taikomos kompleksiniams skaičiams, o vektoriams ir simbolių eilutėms galima tik viena funkcija – *str()*,

t.y. skaičiaus pavertimas simbolių eilute. Kompleksinėms yra priskiriamos tik keturios funkcijos: realiosios ir menamos dalies išskyrimas, kompleksinio modulio radimas ir kompleksinio skaičiaus sudarymas. Visos išvardintos funkcijos tinka realiems ir visos išskyrus kompleksinio skaičiaus sudarymą – kompleksiniams skaičiams. Kadangi *JEP* yra biblioteka, matematinių funkcijų sąrašą galima išplėsti savomis funkcijomis.

Šiam paketui siūlomas išplėtimas *DJEP*, kuris suteikia galimybę diferencijuoti, prastinti išraiškas, palaiko visus veiksmus su matricomis ir vektoriais, taip pat yra kelių išraiškų analizės galimybė (išraiškas atskiriant kabliataškiu).

JEP ir *DJEP* galima gauti nemokamai su GNU General Public Licence (GPL) licencija arba reikia pirkti (pagal komercinę licenciją). Kartu su paketu gaunamas programos kodas.

JbcParser

JbcParser [2] taip pat kaip ir *JEP* yra matematinių išraiškų analizės ir įvertinimo paketas. Jame kaip ir *JEP* yra iš anksto aprašytų funkcijų, kurių sąrašą galima praplėsti savomis. Šis paketas savo funkcijomis ir aprašytais operatoriais labai panašus į *JEP*. Operatoriai abiejų paketų visiškai sutampa, tik *JbcParser* supranta daugiau skirtingų tipų skliaustų nei JEP. Iš anksto aprašyta tik konstanta π, visoms kitoms konstantoms ir kintamiesiems reikšmes reikia priskirti.

Matematinių funkcijų atžvilgiu *JbcParser* gana sunku palyginti su *JEP*, nes abu paketai realizuoja skirtingas funkcijas, nors, žinoma, kai kurios iš jų sutampa, pvz. sinusą, hiperbolinį sinusą, logaritmą realizuoja abu paketai. Reikia pastebėti, kad trigonometrinių funkcijų atžvilgiu stipresnis yra *JEP*, nes jame realizuota daugiau ir įvairesnių tokio pobūdžio funkcijų. Tačiau aritmetinių ir loginių funkcijų prasme pirmauja *JbcParser*, kuris turi tokias funkcijas kaip apvalinimas iš apačios ar viršaus, minimumo ar maksimumo radimas, kėlimas bet kokiu laipsniu, trupmeninės dalies atmetimas ir t.t.

Mathematica

Wolfram Research tyrimų grupės sukurta *Mathematica* [3] – tai matematinė programinė įranga, kuri gali būti integruota į kitas sistemas kaip jų komponentė. Kadangi *Mathematica* yra matematinės analizės programinė įranga, operatoriai ir funkcijos nėra aprašomos atskirai. Pagrindiniai *Mathematica* privalumai: sudėtingi simboliniai skaičiavimai, duomenų analizė ir vizualizacija, lygčių, diferencialinių lygčių sprendimas, simbolinių ir skaitinių reiškinių prastinimas, matematinis modeliavimas ir simuliacija (nuo paprastų kontrolinių sistemų iki sudėtingų biologinių sistemų, cheminių reakcijų ir t.t.), profesionaliai paruošti elektroniniai ar spausdinami raportai ir ataskaitos.

Mathematica paketo branduolys į Java kalba parašytą programą integruojamas gana paprastai: tai daroma per jau realizuotą branduolio ir Java sąsają – *JLink*. Kūrėjai taip pat pateikia komponentus matematinių išraiškų vaizdavimui ir operavimui su jomis. Vienas iš pagrindinių šio paketo privalumų yra tas, kad matematines išraiškas jam galima pateikti ir gauti MathML formatu. Tai yra labai pravartu kuriant nuotolinio testavimo sistemą.

Mathematica, kaip ir anksčiau minėtieji matematiniai paketai yra realizuota Java technologijos pagrindu, todėl ji tinka daugeliui platformų: Windows 98 ir naujesniems, Windows NT ir naujesniems, Mac OS X įvairioms versijoms, Linux Red Hat Enterprise nuo 2.1 ir Enterprise Server nuo 8 versijos, Suse Professional nuo 9 versijos, Sun Solaris nuo 8 versijos.

Kadangi, skirtingai nei *JEP* ir *JbcParser, Mathematica* yra ne biblioteka, o programinė įranga, kompiuterinei įrangai keliami didesni reikalavimai, t.y. skirtingai nei anksčiau aprašytuose paketuose, kur pakanka tik turėti tam tikros versijos Java RE, norint, kad *Mathematica* veiktų taip, kaip priklauso, jau išskiriami ir sisteminiai reikalavimai: minimalus atminties dydis 128MB, rekomenduojamas atminties dydis 256MB ar daugiau, kietasis diskas 400-550MB.

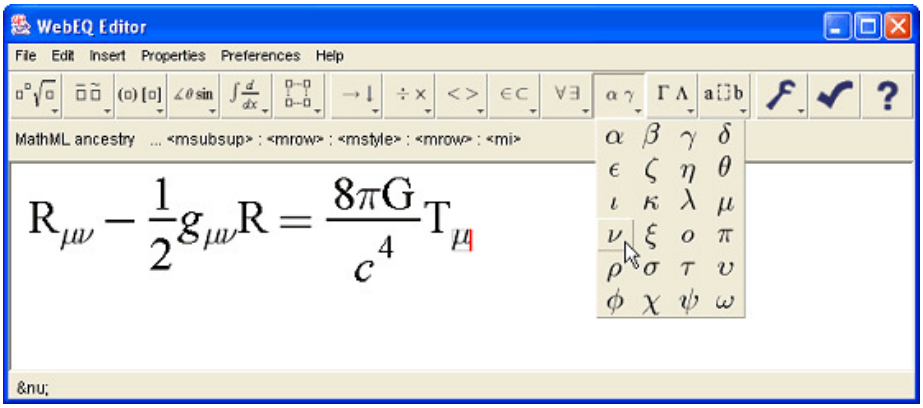
Atsižvelgiant į *Grid* technologijos vystymąsi yra išleista ir *gridMathematica*, skirta gridui. Dėl išsamesnės informacijos apie šią versiją reikia kreiptis į Lietuvos mokslo draugiją. Kita versija, *webMathematica,* yra skirta interaktyviems, dinaminiams matematinio pobūdžio interneto puslapių kūrimui.

WebEQ

Design Science sukurtas WebEQ [4] paketas skirtas kurti internetinius puslapius su interaktyviomis matematinėmis užduotimis, tačiau jį galima pritaikyti ir kuriant Java programas. Dauguma el. mokymo kompanijų, mokomųjų portalų naudoja būtent šį paketą. Iš visų jų galima paminėti: *asknLearn*, *Brownstone Learning* (jų sukurtą *EDU Campus* naudoja KTU fundamentaliųjų mokslų fakultetas), *Carnegie Learning Inc*., *WebCT*. Kadangi *WebEQ* yra sukurtas Java ir MathML technologijų pagrindu, jį taip pat būtų galima integruoti ir į *TT* sistemą.

WebEQ savybės, naudingos *TT* sistemai:

- grafinis formulių įvedimas, parodytas 1 pav.
- matematinės išraiškos išsaugojimas MathML formatu
- išraiškos, išsaugotos MathML formatu atvaizdavimas
- grafikų piešimas
- loginis išraiškų palyginimas



1 pav. **Grafinis formulių įvedimas**

Pagrindinis *WebEQ* trūkumas palyginus su *Mathematica* yra tas, kad jis neturi matematinio branduolio sprendžiančio sudėtingesnius nei algebrinius veiksmus, t.y. *WebEQ* gali palyginti integravimo rezultatą su pateiktu etalonu, tačiau negali atlikti integravimo veiksmų.

Taip pat kaip ir *Mathematica* programinei įrangai *WebEQ* yra keliami sisteminiai reikalavimai, kurie yra išskirti į reikalavimus kūrimo aplinkai ir reikalavimus pristatymo aplinkai (1 lentelė). Kūrimo aplinka – tai aplinka, kurioje dirbama su *WebEQ Developers Suite,* o pristatymo aplinkoje vartotojui yra pateikiamas internetinis puslapis su integruotu *WebEQ*.

Sisteminiai reikalavimai 1 lentelė

Operacinė sistema	Kūrimo aplinka	Pristatymo aplinka
Windows 98, Windows ME, Windows 2000, Windows XP ar naujesnė	Sun Java VM 1.4.1 ar naujesnė	○ Microsoft Java VM 1.1.8, Sun Java VM 1.3.1 ar Sun Java VM 1.4.1. ○ Microsoft Internet Explorer 5.5 ar naujesnis, Netscape 4.7, Netscape 7.01, Mozilla 1.6 ir FireFox 0.9.
Mac OS 9	Nebepalaiko	○ Mac Runtime for Java 2.2 ar naujesnis. ○ Microsoft Internet Explorer 5.1, Netscape 6.2, Netscape 7.01.
Mac OS X (10.2.3 ar naujesnė)	visos Java versijos instaliuotos su OS	○ visos Java versijos instaliuotos su OS priimtinos. ○ Safari 1.2, Microsoft Internet Explorer 5.1.
Solaris, Linux (visos versijos, kurios palaiko Sun Java VM 1.4.1)	Sun Java VM 1.4.1 ar naujesnė	○ Sun Java VM 1.3.1 ar Sun Java VM 1.4.1. ○ Netscape 6.2.2, Netscape 7.01, Mozilla 1.6 ir FireFox 0.9.
Laisva vietos diske	60MB	40MB
RAM	64MB	32MB

Matematinių paketų apibendrinimas

Iš aprašytų keturių paketų (*JEP, JbcParser, Mathematica* ir *WebEQ*) nuotolinio testavimo sistemoje *TT* buvo nuspręsta pritaikyti *WebEQ*. Toks pasirinkimas buvo padarytas atsižvelgiant į kelis kriterijus:

- reikalavimų atitikimą
- integravimo sudėtingumą
- kainą

WebEQ nėra pigiausias iš visų paminėtų paketų (jo kaina 495 USD), tačiau jo kaina geriausiai atitinka vykdomas funkcijas. *Mathematica* (apie 2854 USD), žinoma, turi daugiau funkcijų reikalingų matematikams, tačiau didžiosios jų dalies *TT* nereikia. Integravimo požiūriu, geriausia būtų buvę rinktis *JEP* (700 USD) arba *JbcParser* (19.95 USD), tačiau jie kaip ir *Mathematica* neatitiko vieno pagrindinių reikalavimų paketui, t.y. nebuvo realizuotas grafinis matematinių išraiškų įvedimas. Be to, *JEP* ir *JbcParser* nepalaiko MathML formato, kuris labai patogus aprašant matematines išraiškas.

WebEQ savybės

Grafinis matematinių išraiškų įvedimas ir saugojimas MathML formatu

Grafinio matematinių išraiškų įvedimo pavyzdys matomas 1 pav. Įvesti matematinę išraišką tokiu būdu buvo vienas iš pagrindinių reikalavimų parenkant matematinį paketą. Visų pirma taip yra patogiau nei rašyti formulę eilutės pavidalu, vaizdžiau ir mažesnė klaidos tikimybė.

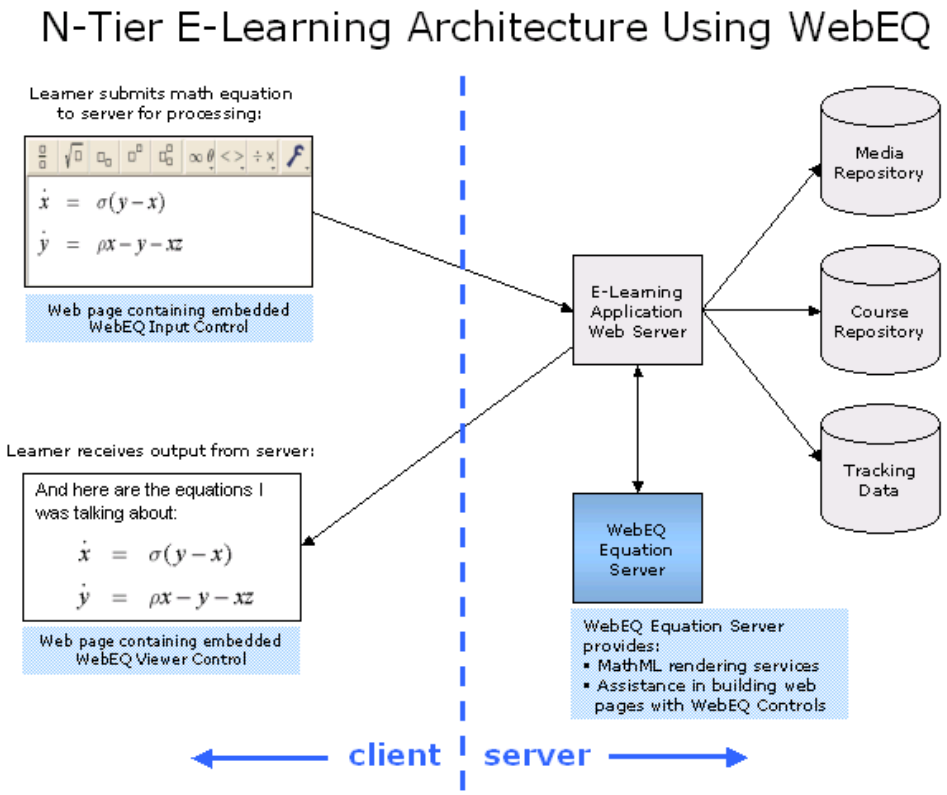
Tai, kad *WebEQ* suteikia galimybę išraišką išsaugoti MathML formatu, taip pat yra didelis paketo privalumas, nes klausimai *TT* sistemoje yra saugomi XML failuose, kuriuose toks formulių įrašymo būdas yra natūralesnis, nei jos užrašymas eilutės pavidalu.

WebEQ architektūra

2 pav. [4] vaizdžiai parodyta *WebEQ* architektūra, kuri labai tinka kuriant nuotolinio testavimo sistemą, nes serveris yra atskiriamas nuo kliento.

Kliento pusėje studentai gali įvesti matematinę informaciją ar peržiūrėti rezultatus. Tam naudojami *WebEQ Controls* (toliau kontroliniai elementai). *WebEQ* sprendimų biblioteka čia naudojama kaip šablonų bei pavyzdinio kodo šaltinis.

Serverio pusėje gali būti naudojamas *WebEQ Equation Server*, kurios gali generuoti išraiškų atvaizdus iš MathML ar kurti sąsajas tarp *WebEQ* kontrolinių elementų ir MathML įvesties.



2 pav. **WebEQ architektūra**

Kaip matoma 2 pav. klientas, t.y. studentas, įveda bei perduoda serveriui (*E-Learning Application Web Server*) matematinę išraišką. Serverio pusėje, jei reikia, kreipiamasi į *WebEQ Equation Server,* kur išraiška gali būti konvertuota į MathML formatą, įvertinta, ar sulyginta su pateiktu etalonu. Vėliau reikalingi duomenys išsaugomi duomenų bazėje (*Media Repository, Course Repository, Tracking Repository*). Klientui atgal iš serverio nusiunčiamas formulių atvaizdas.

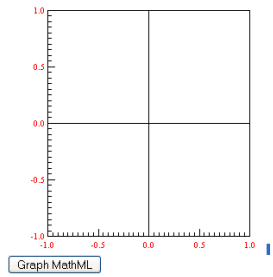
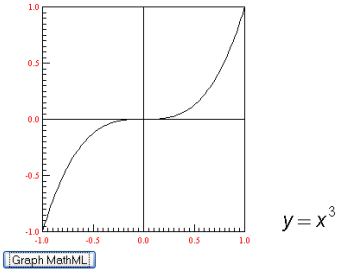
WebEQ sprendimų biblioteka

Sprendimų biblioteka yra *WebEQ Developers Suite* komponentas. Joje yra *JavaScript* bibliotekos ir tinklapių šablonai padedantys kurti dinaminius puslapius. Čia yra pateikiami įvairūs *WebEQ* panaudojimo pavyzdžiai kuriant testus, interaktyvius grafikus, animaciją ir pažingsninius sprendimus. Įdomiausias būtų pastarasis. Šis modulis būtų naudingas praktikos ar uždavinio aiškinimo režime. Čia autorius turi galimybę pasirinkti iš dviejų pažingsninio uždavinio sprendimų tipų: vieno arba daugelio kelių. Vieno kelio sprendimas rodo kaip uždavinių spręsti pažingsniui. Antrasis variantas suteikia autoriui galimybę sudaryti sprendimų medį, t.y. nuo studento pasirinkto sprendimo žingsnio priklauso veiksmų eiga ir paaiškinimai.

Grafikų vaizdavimas

WebEQ Graph Control – tai apletas skirtas atvaizduoti grafikus. Juo galima nubraižyti MathML aprašytas funkcijas ir nelygybes. Tuo pačiu metu gali būti braižomi keli grafikai. Grafikų piešimas pavaizduotas 1 lentelėje.

Grafikų vaizdavimas 2 lentelė

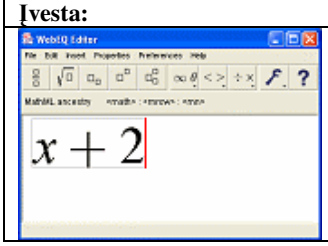
Užduotis	Rezultatas
	

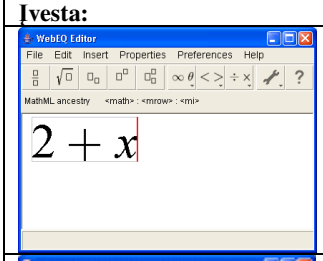
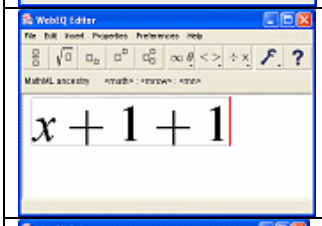
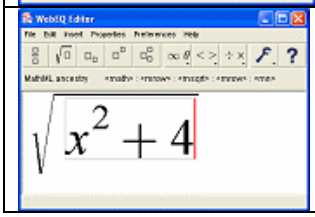
Kadangi grafikai yra piešiami interaktyviai, vėliau jie gali būti panaudoti kuriant klausimus tokius kaip: „kuris iš pavaizduotų grafikų atitinka duotą išraišką?“. Šios savybės *TT* sistema iki šiol taip pat neturėjo. Interaktyvumas reikalingas todėl, kad planuojama sudaryti galimybę kurti dinaminius klausimus, t.y. klausimus su kintamais parametrais, o tokio tipo klausimuose statiniai grafikų atvaizdai nebetenka prasmės.

Išraiškų įvertinimas

WebEQ Evaluation Control yra apletas, kuris apskaičiuoja (įvertina) matematines išraiškas pateiktas MathML formatu. Kaip pavyzdys 2 lentelėje pateikiamas įvairių išraiškų palyginimas su *(x+2)*. Ši funkcija yra būtina, kuomet testo atsakymas turi būti įvedamas, o ne pasirenkamas vienas iš kelių.

Pavyzdinis išraiškų palyginimas su (x+2) 3 lentelė

Įvesta:	***Evaluation Control* analizė:**		**Pastabos:**
	✓	Ekvivalentu $x + 2$.	Analogiškų išraiškų palyginimas. Tokia išraiškas galima palyginti ir simboliškai, nes jos yra identiškos. Lyginant išraiškas, papildomas išraiškos formatavimas (pvz., papildomi tarpai) ignoruojami.

Įvesta:	*Evaluation Control* analizė:		Pastabos:
	✓	Ekvivalentu $x + 2$.	Šiuo atveju jau reikalingas loginis išraiškų palyginimas – simbolinio nebepakanka, nes išraiškos nėra identiškos, kaip kad praeitame pavyzdyje.
	✓	Ekvivalentu $x + 2$.	Norint palyginti šias išraiškas, reikalingas matematinis išraiškų prastinimas, *WebEQ* sugeba tai atlikti.
	✗	Nėra ekvivalentu $x + 2$.	Šiame pavyzdyje parodyta, kad *WebEQ* galima palyginti ir sudėtingesnes išraiškas, t.y. suprastinus šį reiškinį nėra gaunamas reiškinys ekvivalentus lyginamajam.

Iš 3 lentelės [4] galima matyti, jog testuojamajam *TT* sistemos vartotojui nereikia pateikti vienareikšmį atsakymą – jį galima pateikti keliais variantais, svarbu tik tai, kad atsakymas būtų ekvivalentus teisingajam. Integravus *WebEQ* paketą į *TT* sistemą, atsiranda galimybė nustatyti pateikto atsakymo ekvivalentiškumą atsakymo etalonui.

Išvados

- Straipsnyje aptarti *JEP*, *JbcParser*, *Mathematica* ir *WebEQ* matematiniai paketai. Parodyti jų trūkumai ir privalumai atsižvelgiant į nuotolinės testavimo sistemos *TT* poreikius. Iš visų apžvelgtų paketų *WebEQ* geriausiai tenkina iškeltus reikalavimus – atsakymo ekvivalentiškumo etalonui nustatymas, grafinis formulių įvedimas ir vaizdavimas.

- Išsamiai aprašytas *WebEQ* paketas. Parodyta, kaip šis paketas padeda išspręsti iškeltas dvi pagrindines problemas: išraiškų ekvivalentiškumo nustatymą ir grafinį formulių įvedimą bei vaizdavimą. WebEQ paketo integravimas – vienas paprasčiausių būdų praplėsti TT funkcionalumą taip, kad būtų galima nuodugniai testuoti matematinius gebėjimus.

Literatūra

[1] Singular Systems. JEP. Prieiga internetu: http://www.singularsys.com/jep/index.html [žiūrėta 2006 03 27]
[2] Bestcode. JbcParser. Prieiga internetu: http://www.bestcode.com/html/jbcparser.html [žiūrėta 2006 03 27]
[3] Wolfram Research. Mathematica. Prieiga internetu: http://www.wolfram.com/products/mathematica/index.html [žiūrėta 2006 03 27]
[4] Design Science. WebEQ. Prieiga internetu: http://www.dessci.com/en/products/webeq/ [žiūrėta 2006 04 20]
[5] Brownstone. EDU Campus. Prieiga internetu: http://www.brownstone.net/products/ecampus.asp [žiūrėta 2006 04 20]

Summary

The main idea of this article is to discuss about problems when adopting distance testing system *TestTool* to thoroughly test mathematical skills. There are reviews of most commonly used mathematical packages based on Java technology, i.e. *JEP*, *JbcParser*, *Mathematica* and *WebEQ*. The article shows up advantages and drawbacks of each package. Finally, a package that best meets given requirements is chosen.