KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Ligita Tamašauskienė

SCRALL GRAFINIO REDAKTORIAUS KŪRIMAS IR TYRIMAS

Magistro darbas

Darbo vadovas
prof. Eduardas Bareiša

Kaunas, 2010

Turinys

Paveiksliukai

SUMMARY

This is Master‘s document. This document purpose is to understand and to analyze UML modeling languages. To find the best solution for user, how works in software engineering industry. Was decided to create Scrall editor tool and to analyze and compare it with other UML graphic languages.

Scrall (Starr‘s Concise Relatio Action Language) - is yet another action language at the same semantic level as other relational action languages such as SMALL, TALL, OAL and ASL. All of these languages are platform independent, object-oriented methods of specifying relational data access, computation, data processing and low level branching decisions.

Scrall is primarily a graphical language. Symbols are networked together with object, data and control flows.

Scrall is fully compatible with UML 2.0 action semantics. It is slightly consistent with UML 2.0 activity and action symbols. Unfortunately, the current OMG standard is not sufficiently expressive in the area of relational data access. An attempt will be made to adjust some of the symbols and notation of Scrall so that it fits in as much as possible with standard UML notation.

At the model level data is organized relationally. The action language should maximize the power of this paradigm rather than supress it in favor of object oriented programming level semantics.

1. ĮVADAS

1.1. Dokumento paskirtis

Šis dokumentas yra magistro baigiamasis darbas. Jame yra apžvelgiami, grafinės kalbos modeliavimo programos. Šiame darbe, stangiausi apžvelgti pagrindines grafinių modeliavimo įrankių reikalavimus. Dokumente iškirtas Scrall grafinis redaktorius, kuris buvo kuriamas viso magistro metu.

Apžvelgiamos pagrindinės Scrall naudojimo savybės, naudojami architektūriniai sprendimai kuriant programinę įrangą. Taip pat buvo siekiama, įvertinti Scrall grafinio redaktoriaus perspektyvas, jos išvystymą į StarUML redaktorių, kuris yra jau pateiktas atviram visuomenės naudojimui.

1.2. Santrauka

Šiuolaikiniame pasaulyje yra vykdoma milijonai IT projektų, kurie tampa vis didesni ir sudėtingesni. Šių projektų tikslas yra laiku ir neviršijant nustatyto biudžeto pagaminti vartotojo reikalavimus atitinkančią ir kokybišką programinę įrangą. Tačiau, pasaulio IT projektų statistika rodo, kad nemaža dalis tokių projektų turi rimtų problemų, kaip viršytas biudžetas, vėlavimas arba neišpildomi iškelti ir apsibrėžti tikslai. Įvairių apklausų duomenimis [3], tik mažiau nei ketvirtadalis visų IT projektų pasiekia iškeltus reikalavimus ir tikslus laiku, neviršydami biudžeto. Esant tokiai nepalankiai statistikai imtasi ieškoti priemonių, kaip su jomis kovoti. Vienas iš sprendimų, visus reikalavimus sistemai aprašyti grafinių redaktorių pagalba. Tokie grafiniai redaktoriai, kaip UML, supaprastina ir palengvina sistemos supratimą ir kūrimą.

Šio magistrinio darbo tikslas, papildyti turimą UML redaktorių, kuris palaiko OCL standartą, Scrall grafinės kalbos elementais. Sukurti Scrall apribojimams grafinę notaciją ir integruoti juos į UML diagramas.

Visas darbas yra orientuotas į programų inžinerijos srityje dirbančią akademinę visuomenę.

2. GRAFINIŲ REDAKTORIŲ ANALIZĖ

Šiuo metu, programinės įranga jau yra neatsiejama daugumos verslo organizacijų infrastruktūros dalis. Kuo toliau, tuo labiau programinės įrangos apimtys auga ir sudėtingėja. Projektavimas, vis labiau tampa svarbia programinės įrangos inžinerijos dalimi. Programinės įrangos projektavimas ir architektūros dokumentavimas yra svarbūs sistemos palaikymui – plėtimui, defektų taisymui, sistemos adaptavimui skirtingoms platformoms, integravimui su kitomis sistemomis.

Aiškumas – sistema padaroma aiški ir lengvai suprantama. Sistemos supratimas yra pirmas žingsnis tiek statyboje tiek sistemos kūrime. Tai apima žinojimą, iš ko sistema yra sudaryta, kaip ji elgiasi, ir pan. Sistemos modeliavimas garantuoja, kad tai taps lengvai skaitoma ir, svarbiausiai, bus lengva dokumentuoti. Sistemos aprašymas padaro lengvai suprantamą jos struktūrą ir elgesį.

Panaudojimo patogumas iš naujo - šalutinis produktas skirtas daryti sistemą lengvai skaitomą. Po to, kai sistema sumodeliuojama, siekiant padaryti tai lengvai suprantama, mes galime nustatyti panašumus ar dubliavimąsi, juos išreiškiant funkcionalumu, ypatybėmis, ar struktūra.

2.1. Situacijos Lietuvoje įvertinimas

Lietuvoje vis daugiau atsiranda įmonių, kurios užsiima programinės įrangos kūrimu. Pagal „Visos Lietuvos“ įmonių katalogą, jau randama virš 500 įmonių, kurios užsiima programinės įrangos kūrimu ir jos priežiūra. Nenuostabu, kad kuo toliau, tuo labiau populiarėja UML, kuris stipriai palengvina programinės įrangos kūrimo darbą. Lietuva neatsilieka kurdama UML įrankius, kaip MagicDraw. Jį kuria UAB “Baltijos programinė įranga“, bei yra pripažinta ir plačiai naudojama netik Lietuvoje, bet visame pasaulyje.

2.2. UML grafinis redaktorius

UML yra standartinė kalba skirta vizualizavimui, specifikavimui, konstravimui ir dokumentavimui. Ji naudojama sistemoms, kurios naudoja programinę įrangą.

UML naudoja visuose technologiniuose procesuose, visose gyvavimo ciklo stadijose. Šios kalbos pagrindinis privalumas yra tai, kad ji nėra susieta konkrečia sistemų kūrimo metodika ar sistemų įgyvendinimo technologija.

UML paveldėjo geriausius modeliavimo bruožus:

- duomenų modeliavimas;
- verslo modeliavimas;
- objektų modeliavimas;
- komponentų modeliavimas.

Naujausia UML specifikacijos versija 2.2 aprašo per 200 modelio elementų ir 13 diagramų suskirstytų į tris pagrindines kategorijas:

1. Struktūrinės diagramos: tai priemonės, skirtos grafiškai parodyti sistemos komponentų struktūros hierarchiją. Duomenų srauto elementų diagramos realizuojamos kaip programų dalių hierarchija. Šios diagramos naudojamos tada, kai siekiama parodyti statinį projektavimo organizavimo vaizdą..

Struktūrinės diagramos:

- Klasių diagrama (class) – atvaizduojama sistemos struktūra, jos klasės ir jų atributai, bei ryšiai tarp klasių ir operacijos.
- Objektų diagrama (object) – vaizduoja dalinį arba visą modeliuojamos sistemos vaizdą konkrečiu momentu.
- Sudėtinė, kompozicinės struktūros diagrama (composite) – vaizduojama vidinė klasių struktūra bei bendradarbiaujant suteiktos galimybės..
- Paketų diagrama (package) – vaizduojama sistemos suskaidymą į atskiras logines grupes ir kokia yra jų tarpusavio priklausomybė.
- Komponentų diagrama (component) – vaizduojamas kaip sistemos išskaidymas į komponentus ir jų tarpusavio priklausomybės.
- Išdėstymo diagrama (deployment) – modeliuojamas fizinis sistemos elementų išdėstymas.

2. Elgsenos diagramos – vaizduoja kaip kas turi elgtis modeliuojamoje sistemoje.

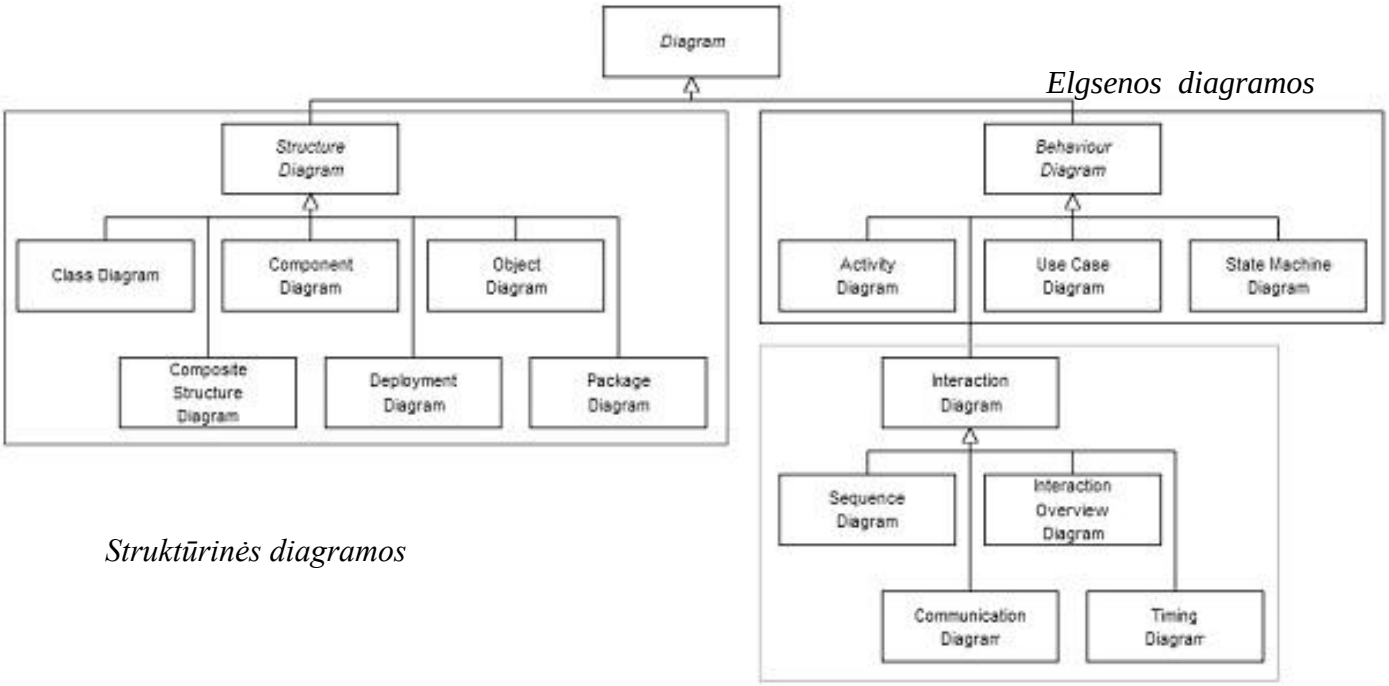
Elgsenos diagramos yra:

- Būsenų diagrama (State) – vaizduoja klasės judėjimą. Šiomis diagramomis paaiškinamas judėjimas ir elgsena.
- Panaudos diagrama (Use Case) – vaizduojami funkciniai sistemos reikalavimai, išoriniai aktoriai, jų elgsena ir sistemos veiksmai su jais.
- Veiklos diagrama (Activity) – vaizduojama sistemos elgsena ir jos elgsena su tam tikros sistemos srautais. Taip pat vaizduoja sistemos komponentų veiklų sekas.

3. Sąveikos diagramos – parodoma objektų bendradarbiavimas, jų tvarką keičiantis pranešimais.

Sąveikos diagramos:

- Komunikacijų diagrama (Communication) – pranešimais atvaizduojama objektų arba jų dalių sąveika.
- Laikinio pasiskirstymo diagrama (Timing) – vaizduojami modelyje esantys laikini apribojimai.
- Sąveikos apžvalgos diagrama (Interaction Overview) – vaizduojamos viršūnės, kurios atitinka sąveikos diagramas.
- Sekų diagrama (Sequence) – vaizduojamos pranešimų sekos, skirtos objektų bendradarbiavimui. Sekos nurodo objektų gyvavimo trukmę.



Elgsenos diagramos

Struktūrinės diagramos

Sąveikos diagramos

1 pav. UML 2.0 diagramų klasifikavimas

Magistrinio darbo metu kuriamas grafinis redaktorius remiasi struktūrinėmis (klasių, objektų) ir elgsenos diagramomis (sąveikos, veiklos).

2.2.1. OCL

Šis standartas yra skirtas Objektinei analizei ir projektavimui. Objekto apribojimo kalba OCL yra neatsiejama UML dalis. OCL patogi tuo, kad priešingai nuo kitų formalaus specifikavimo kalbų, nenaudoja sudėtingų matematinių išraiškų. UML modelis gali nusakyti sistemos architektūra bei veikimo principus, tačiau UML standartas nenumato galimybės apibrėžti galimų UML objektų, atributų ar operacijų reikšmių, todėl ir buvo sukurtas dar vienas standartas [6].

OCL susideda iš keturių pagrindinių dalių.:

1. konteksto, kuris apibrėžia apribotą situaciją, kurioje tvirtinimas yra galiojantis
2. ypatybės, kuri atstovauja kai kurioms konteksto savybėms (pvz., jei situacija yra klasė, ypatybė galėtų būti atributas).
3. operacijos (pvz., aritmetinis), kuri valdo arba apibūdina ypatybes
4. raktinių žodžių (pvz., jei, tada, dar, ir, ar ne, reiškia), kurie naudojami nustatyti, sąlyginėms išraiškoms.

OCL yra deklaratyvi kalba tam, kad apibūdintų taisykles, kurios prisitaiko prie UML modelių, išvystytų IBM ir dabar UML standarto dalyje. Pradžioje OCL specifikacijos buvo tiktais oficialus kalbos ištęsimas į UML.

2.3. Egzistuojančių UML redaktorių analizė

Magic Draw UML

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. MagicDraw UML yra 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, leisdama daugeliui programinės įrangos kūrimo inžinierių dirbti su tuo pačiu OO modeliu vienu metu.

Rational Rose

Rational Rose tai CASE – priemonė skirta PĮ projektavimui bei analizės etapų automatizavimui bei kodų generavimui. Pagrindinis Rational Rose/C++ funkcionalumas leidžia rengti projektinę dokumentaciją diagramų ir specifikacijų pavidalu ir generuoti programinius kodus su C++.

ArgoUML

ArgoUML yra UML grafinio vaizdavimo priemonė, parašyta Javoje ir išleista pagal BSD Licencija. Remiantis Javos pritaikymu, tai pasiekia bet kokios platformos, palaikomos Javos. ArgoUML išsivystymas labai nukentėjo nuo darbuotojų trūkumo. ArgoUML palaiko XMI, OCL standartus.

Together

Together yra Borland produktas, kuri sujungia Java IDE. Iš pradžių šis įrankis turėjo savo pagrindas JBuilderiui su UML modeliavimo įrankiu.

Techniškai, Together yra Eclipse įskiepis. Together palaiko UML 1,4 modeliavimą, fizinį duomenų modeliavimą, projektavimo modelius, kodo dizaino atpažinimo šabloną ir pakartotinį naudoti, bei dokumentacijos generavimą. Together kalba yra neutrali UML 2.0 diagramų, verslo procesų modeliavimo ir loginių duomenų modeliavimo kalba.

SmartDraw

SmartDraw yra verslo grafinė programinė įranga, išvystyta SmartDraw.com. SmartDraw yra naudojama, norint sukurti verslo grafiką, kaip struktūrinės schemos, organizacinė struktūra, diagramos, laiko grafikai, proto žemėlapiai, aukšto planai, ir kitos diagramos. SmartDraw yra taip pat pasiekiamas dviejuose specifiniuose pramonei sprendimuose: SmartDraw Teisėtas Leidinys ir SmartDraw Sveikatos priežiūros Leidinys. [5]

Dar yra daugybė UML grafinių redaktorių, kurie yra tiek mokami tiek ne, tačiau jų visų neįmanoma išvardinti ir aprašyti.

2.4. SCRALL grafinio redaktoriaus analizė

Šiame magistriniame darbe buvo siekiama praplėsti UML redaktoriaus, su OCL apribojimais, galimybes. Buvo sukurta programinė įranga turinti bazines grafinio modeliavimo įrankio savybes. Sistema palaiko įskiepius, per kuriuos yra realizuotos sistemos funkcijos. Buvo stengiamas įdiegti funkcijas, kurios leistų taikyti Scrall apribojimus UML klasėms. Įmanoma konvertuoti Scrall diagramas taip, kad jos atitiktų glaustas UML veiklos diagramas. Scrall redaktoriaus pradininkas yra Leon Starr.

Scrall yra dar viena veiklos kalba, kuri atvaizduoja veiklos diagramų ryšius kaip ir kitos tokios kalbos: SMALL, TALL, OAL ir ASL. Visos šios kalbos yra nuo platformos nepriklausomos, metodai objektiškai orientuoti į specifinius duomenų priėjimo ryšius,

apskaičiavimą ir duomenų procesus. Šios kalbos pradininkas yra Steve Mellor, kuris yra sukūręs SMALL kalbą.

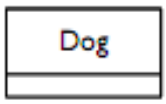
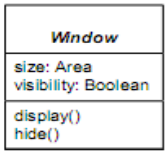
Scrall yra visiškai suderinama su UML 2.0 veiklos semantika. Ši grafinė modeliavimo kalba palaiko UML 2.0 veiklos ir būsenų diagramų simboliais. Deja, dabartinis OMG (Object managment group) standartas nepakankamai atvaizduoja duomenų ryšius. Buvo stengiasi sukurti tokį Scrall redaktorių, kurio simboliai ir žymėjimas būtų pakankamai artimi standartiniam UML žymėjimui. Scrall diagramas turi būti galima lengva perbraižyti į veiklos ir būsenos diagramas.

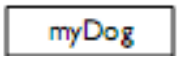
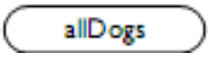
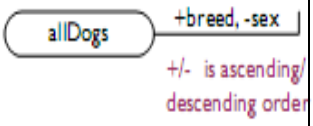
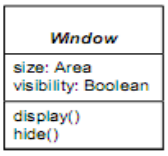
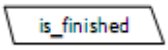
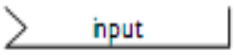
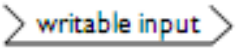
Su Scrall galima:

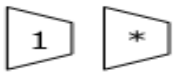
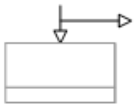
- Minimizuoti simbolių tipų kiekį
- Palengvinti aktorių paiešką jungiamiesiems elementams
- Užims mažiau vietos (diagramos)
- Glausčiau pateikiamas sudėtinių objektų veiksmas

Lentelėje nr.1 pateikti keli pagrindiniai Scrall elementai, bei palyginami su standartiniai UML elementais.

1 lentelė. Scrall ir standartinių UML elementų palyginimas

Tipas	Žymėjimas Scrall	Žymėjimas UML	Scrall aprašymas
Sintaksė			
		nėra	Kvadratiniai skliaustai pažymi, kad visi gali būti pasirinktas vienas iš siūlomų elementų
		nėra	Riestiniai skliaustai pažymi, kad vienas arba keli elementai yra leidžiami
Informacijos saugyklos			
Klasė			UML simboliu atitikmuo, tačiau sutrumpintas. Čia reikalingas tik klasės vardas

Vienas objektas			Jei egzistuoja tik vienas elementas, tai siūloma jį vaizduoti kvadrate
Keli objektai			Jei egzistuoja tam tikra grupė elementų jie žymimi ovaliam kvadrate
Tam tikrų objektų grupė			Gali būti, kad reikia naudoti vieną arba kelis objektus. Kiekvienas elementas yra klasės atributas. Kiekvienas atributas yra didėjantis (+), arba mažėjantis (-).
Kintamasis		nėra	Vaizduojama laikina informacija bet kokiam objektui t.y. tarpinė objekto reikšmė.
parametrai		nėra	Vaizduojamas „praeinamas“ įvykis arba operacijos parametras
parametrai		nėra	Operacijos parametras, kuris nurodo, kad gali pasikeisti įvesties duomenų reikšmės
Perduodama informacija			
Objektų			Tai nuoroda, kuri per duoda objekto informaciją, kurią perduoda klasės ar operacijos
Informacijos			Tiksli reikšmė. Informaciją perduoda laikini kintamieji ir atributai.

Valdymo			Valdomoji rodyklė nurodo sąlygines kryptis ir sudaro papildomas veiklas
Veiksmai			
Pažymėjimas		1 1…*	Pažymi vieną arba daug objektų
Radimas		1 1...*	Randa vieną arba daug objektų
Sukūrimas			Objektas nurodytas tiesiai į klasės simbolį pažymi sukuriamą objektą
Ištrynimas			Veiksmas ištrynimui
Išsišakojimas			Pasirinkimų galimybės simbolis yra priimtas iš UML
Priskyrimas		nėra	Nustatomos operacijos privalo priklausyti tai pačiai klasei.
Paskyrimas			Parametrų reikšmės gali pereiti į įvykį

Scrall pirmiausia yra grafinė modeliavimo kalba, kurios simboliai yra kartu susiejami objektų, duomenų ir kontrolės srautais.

SCRALL apribojimai sprendimui

- Sprendimas remiasi UML 2.0 ir OCL 2.0 standartų specifikacijomis.

- Sistema bendradarbiauja su Microsoft Visual Studio 2005 aplinka.
- Scrall redaktorius patogus ir lengvai suprantamas vartotojui.
- Scrall grafinis redaktorius nemokamas.

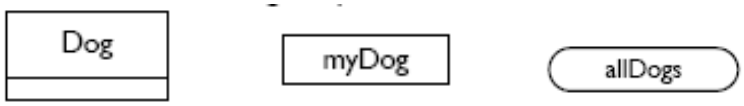
SCRALL diegimo aplinka

- Sprendimas diegiamas kompiuteryje su Windows operacine sistema ir .NET 2.0 karkasu.
- Windows 2000, Windows XP, Windows Vista, Windows 7
- 128 MB RAM (256MB rekomenduojama)
- 110 MB kietojo disko vieta (150MB rekomenduojama talpa)
- SVGA arba didesnės raiškos (1024x768 rekomenduojama)

2.4.1. Scrall diagramos pagrindinės taisyklės

1. Duomenų saugyklos

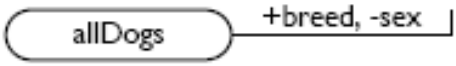
Duomenys atvaizduojami klasėmis arba objektais.



2 pav. Scrall klasė, objektas ir sudėtinė objektų saugykla

Klasė (2 pav.) kaip ir UML diagramose atlieka ta pačią funkciją, tačiau Scrall diagramose klasėje rašomas tik jos vardas. Atributai ir klasės operacijos nėra aprašomos. Kadangi tik klasės pavadinimas yra naudojamas, apatinė klasės dalis su horizontalia linija, vaizduojama tik grafikos dizaino sumetimais.

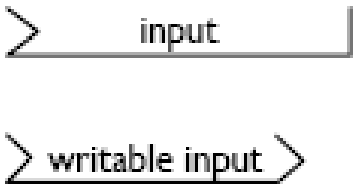
Jei egzistuoja tik vienas objektas (2 pav.) jis yra atvaizduojamas stačiakampiu. Joje taip pat aprašomas tik objekto vardas. Atributai ir operacijos neaprašomos. Norint atvaizduoti daugiau nei viena objektą, t.y. tam tikrų objektų grupę, jie vaizduojami stačiakampyje su suapvalintais kraštais.



3 pav. Elementas skirtas priskirtų objektų grupei

Objektų grupė, gali būti rūšiuojama pagal kelis ar keliolika skirtingų kriterijų (raktų) (3 pav.). Kriterijai, pagal kuriuos rūšiuojami, klasifikuojami objektai yra aprašomi ir prijungiami prie tos objektų grupės. Kiekvienas rūšiavimo kriterijus yra objektų grupės atributas. Šie atributai gali būti žymimi didėjimo (+) ir mažėjimo tvarka (-). Kiekvienas kriterijus (raktas) yra atskiriamas kableliais ir skaitoma iš kairės į dešinę.

4 pav. Elementas tarpinėms reikšmėms atvaizduoti

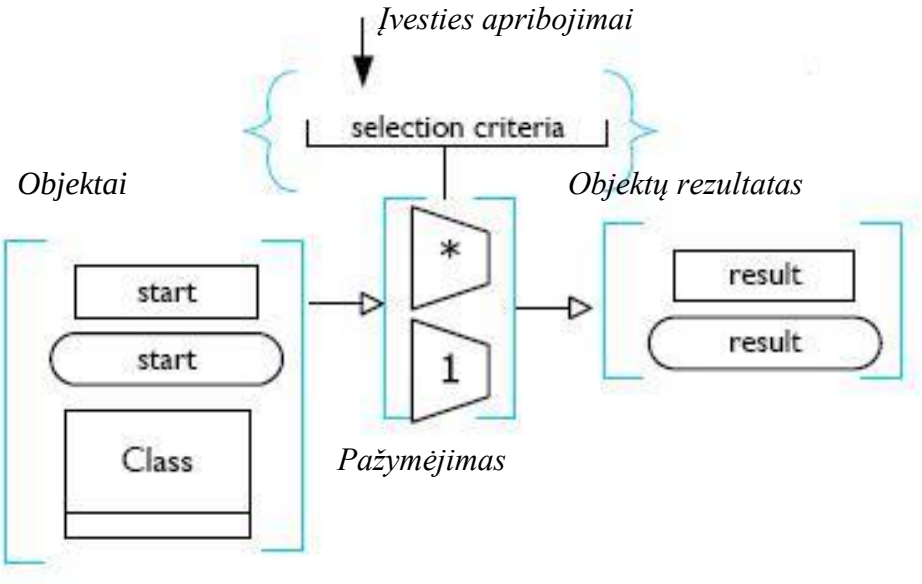


5 pav. Elementai įvedami parametrams vaizduoti

4 paveikslėlyje atvaizduojama, kokios būna objektų, klasių tarpinės reikšmės. Duomenų saugyklos tarpinių duomenų kiekis yra nevaizduojamas.

5 paveikslėlyje vaizduojami įvedami parametrai. Viršutinis paveiksliukas vaizduoja, kokie parametrai yra įvedami pereinamuoju įvykiu, o apačioje vaizduojami parametrai, kurių pagalba operacijos metu gali būti pakeistos įvedamos reikšmės.

2. Objektų žymėjimas



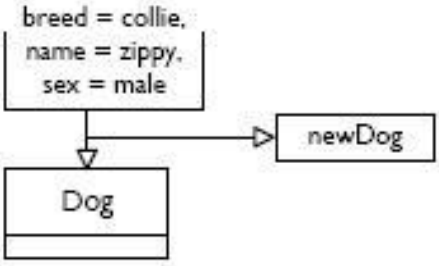
6 pav. Objektų žymėjimas

Įvesties duomenys gali turėti neribotą kiekį duomenų. Jei, šie duomenys (6 pav.) turi kelis pažymėjimus, vadinasi, gali būti naudojamas bet kuris arba visi iš karto vaizduojami kriterijai. Objektai arba klasės laikomos pradiniais įvesties duomenimis. Įvesties duomenys gali būti sudėtiniai ir žymimi: (veislė = Kolis) ir (amžius > 5). Simboliais =, <, > žymi operatorių ryšius. Objektų kiekio žymėjimas: 1 arba *. 1 – nurodomas, kad tik vienas objektas, * - nurodo, kad gali būti naudojamas keli arba visi objektai.

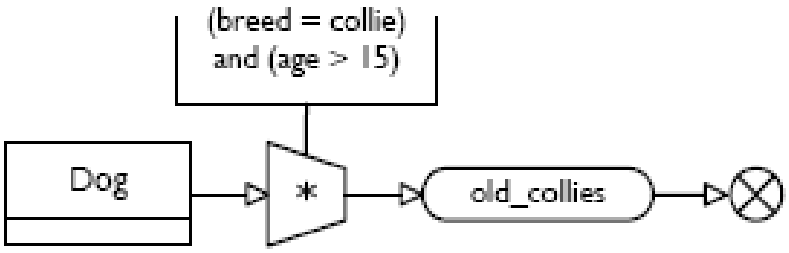
3. Objektų kūrimas arba ištrynimas

Naujo objekto sukūrimas pavaizduotas 7 paveikslėlyje. Objektas nurodytas tiesiai į klasės simbolį, vaizduoja naujo objekto sukūrimą. Gali būti sukurtas vienas objektas arba visa objektų grupė.

Objekto ištrynimas vaizduojamas 8 paveikslėlyje. Ištrynimo simbolis vaizduojamas su perbrauktu apskritimu. Galima ištrinti viena objektą arba visą objektų grupę.



7 pav. Naujo objekto sukūrimas



8 pav. Pažymėto objekto ištrynimas

2.5. StarUML grafinio redaktoriaus analizė

Kaip ir kiekviena programinė įranga privalo būti tobulinama, nes kitaip bus pasmerkta žlugti. Scrall redagavimo kalba buvo pervadinta į StarUML ir padaryta dar labiau panaši į standartinį UML redaktorių.

StarUML tapo atviro kodo įrankiu, skirtu UML/MDA modeliavimui bei darbui ant Win32 platformos. Šiuo įrankiu siekiama modeliavimą padaryti kuo paprastesnį, pakankamai lankstų, greitai vystomą bei nemokamą. Pagrindinis StarUML projekto tikslas sukurti kuo paprastesnį programinės įrangos modeliavimo įrankį bei platformą, kuri sugebėtų pakeisti kitus didelius iš komercinės paskirties UML įrankius, kaip Borland's Together. StarUML palaiko UML 2.0 standartą.

StarUML palaiko diagramas:

- Panaudos atvejų;
- Klasių;
- Sekų;
- Veiklos;
- Bendradarbiavimo;
- Būsenų;
- Komponentų;
- Realizavimo;
- Sudėtinės struktūros.

StarUML yra daugiausia parašyta su Delphi. Tačiau, StarUML yra daugiakalbis projektas ir nesurištas su specifine programavimo kalba, tokiu būdu, bet kokios programavimo kalbos, gali būti panaudotos, kad išvystytų StarUML. (pavyzdžiui, C/C ++, Java, Delphi, JScript, VBScript, C #, VB.NET...)

StarUML palaiko įskiepius ir taip leidžia praplėsti savo funkcionalumą naujais modelių bei diagramų tipais.

Savybės:

- Palaiko įskiepius
- Generuoja kodą iš UML klasių diagramų
- Leidžia kurti OSL apribojimus klasėms, tačiau naudoja juos tik kaip papildomą tekstinę informaciją

Privalumai [2]:

- Gali būti kuriami profesionalūs, sujungti modeliai;
- Kodas ir dokumentacija išskaidyta į atskirus paketus;
- Palaiko papildomą importavimą ir eksportavimą;
- Nemokamas.

Trūkumai:

- Pakankamai sudėtinga pradedantiesiems;
- Palaiko kodo generavimą tik Windows Com modeliui;
- Platformos nesikerta;
- Nebevystoma.

StarUML sistemos reikalavimai

- Intel® Pentium® 233MHz arba didesnė
- Windows 2000, Windows X, Windows Vista, Windows 7
- Microsoft Internet Explorer 5.0 arba naujesnė
- 128 MB RAM (256MB rekomenduojama)
- 110 MB kietasis diskas (150MB rekomenduojama talpa)
- CD-ROM diskas
- SVGA arba didesnės raiškos (1024x768 rekomenduojama)

Palaiko kalbas:

- Java, Kodo generatorių ir atvirkštinį variklį.
- C++, Kodo generatorių ir atvirkštinį variklį.
- C#, Kodo generatorių ir atvirkštinį variklį.

StarUML pagrindinės įdiegtos funkcijos pateiktos lentelėje nr. 2 [1]:

2 lentelė. StarUML pagrindinės funkcijos

Funkcija	Aprašymas
Palaiko tikslų UML standarto modelį	StarUML griežtai pritaikyta UML standartinei specifikacijai, apibrėžta OMG programinės įrangos modeliavimu. Įvertinta, kad projekto informacijos rezultatai, gali būti saugomi 10 metų ar daugiau, priklausomybė nuo specifinės UML sintaksės ir semantikos gali būti ganėtinai pavojinga.
Atviros programinės įrangos modeliavimo formatas	Skirtingai nuo daugelio egzistuojančių produktų, kurie turi ir valdo savo specifinius modelio formatus, StarUML valdo visas rinkmenas standartiniame XML formate. Kodai, parašyti lengvai skaitoma struktūra ir jų formatu, gali būti pakeisti patogiai vartojant XML. Atsižvelgiant į tai, kad XML yra pasaulinis standartas, tai yra tikrai

	didelis pranašumas, garantuojant, kad programinės įrangos modeliai lieka naudingi daugiau nei dešimtmetį.
MDA palaikymas	StarUML palaiko UML Profilį. Tai maksimizuoja UML ištęstinumą, darant modeliavimo pritaikymą, net tokiose srityse kaip finansai, gynyba, e.prekyba, draudimas ir aeronautika. Gali būti sukurti Nepriklausomos Platformos Modeliai (PIM), ir automatiškai sugeneruoti Specifinės Platformos Modelio (PSM) kodai.
Metodologijos ir platformų pritaikymas	StarUML valdo metodo sąvoką, kurdamas aplinkas, kurios prisitaiko prie bet kokios metodologijos/proceso. Ne tiktai taikomųjų struktūrų modelių platformoms kaip .NET ir J2EE, bet taip pat ir pagrindinės programinės įrangos modelių struktūros (pvz. 4+1 vaizdo modelis, ir taip toliau) gali būti lengvai apibrėžtos.
Puikus ištęstinumas	Visos StarUML įrankių funkcijos yra automatizuotos pagal Microsoft COM. Bet kokia kalba, kuri palaiko COM (Visual Basic Script, Java Script, VB, Delphi, C ++, C #, VB.NET, Python ir t.t.) gali būti pavartota, norint kontroliuoti StarUML ar vystyti integruotus elementus.
Programinės įrangos modelio patikrinimo funkcija	Vartotojai gali padaryti daug klaidų per programinės įrangos modeliavimą. Tokios klaidos gali būti labai brangios jei paliktos neištaisytos iki galutinis kodo kūrimo stadijos. Norint išvengti šios problemos, StarUML automatiškai patikrina vartotojo išvystytą programinės įrangos modelį, palengvindamas ankstyvą klaidų radimą, ir leisdamas labiau nepriekaištingą ir užbaigtą programinės įrangos išsivystymą.
Naudingas Add-ins	StarUML apima daugybę naudingų Add-ins su įvairiais funkcionalumais: generuoja programos tekstus programavimo kalbose ir paverčia programos tekstus į modelius, importuoja Rational Rose failus, naudodamas XMI keičia modelio informaciją su kitais įrankiais, ir palaiko projektuojamą struktūrą. Add-Ins palaiko papildomą panaudojimą, produktyvumą, lankstumą ir tarp veiksmingumą dėl modeliavimo informacijos.

2.6. Verslo galimybės ir perspektyvos

Vis daugiau ir daugiau yra kuriama programinių įrangų, kurios kuriamos vis didesnės ir sudėtingesnės. Tam, kad palengvinti programinių įrangų kūrimą ir supratimą, vis daugiau kuriama papildomų įrankių. Šiomis dienomis yra sukurta ir vis dar kuriama daugybė UML modeliavimo įrankių. Dėl didelės šių įrankių pasiūlos rinkoje su nauja modeliavimo kalba įsitvirtinti yra pakankamai sudėtinga ir reikalauja nemažai pastangų.

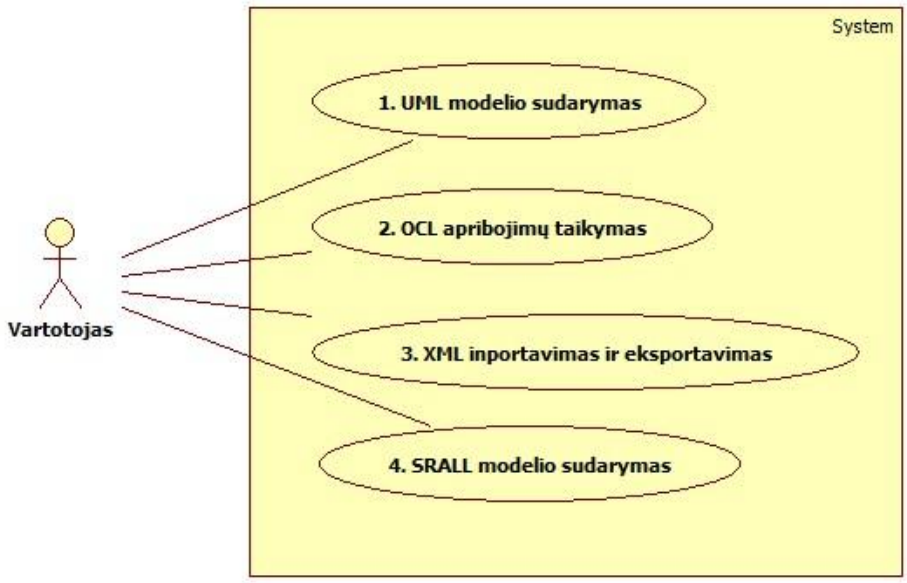
Nemažai įrankių be UML taip pat palaiko dar ir kitokius modeliavimo įrankius. Paprastas UML įrankis vartotojų nenustebins ir nesudomins, kadangi jie jau dabar turi nemaža pasirinkimą. Siekiant sudominti naujus vartotojus ir vėliau netgi gauti finansinės naudos, reikalinga jau esamus įrankius papildyti naujomis galimybėmis, savybėmis, kurių joks kitas įrankis neturi ir nėra numatyta jų specifikacijoje. Tačiau norint įdiegti naujas savybes, kurios gali sudominti vartotojus, reikalauja papildomų lėšų.

Didelė dalis tokių UML įrankių yra sukurti pasitelkiant Java arba C++ kalbas. Šis Scrall kuriamas įrankis yra realizuotas .NET karkaso pagalba, tai suteikia tam tikrų privalumų. Naujos modeliavimo įrankio savybės ir funkcijos, suteikia puikią galimybę įsitvirtinti rinkoje.

3. SCRALL PROJEKTAS

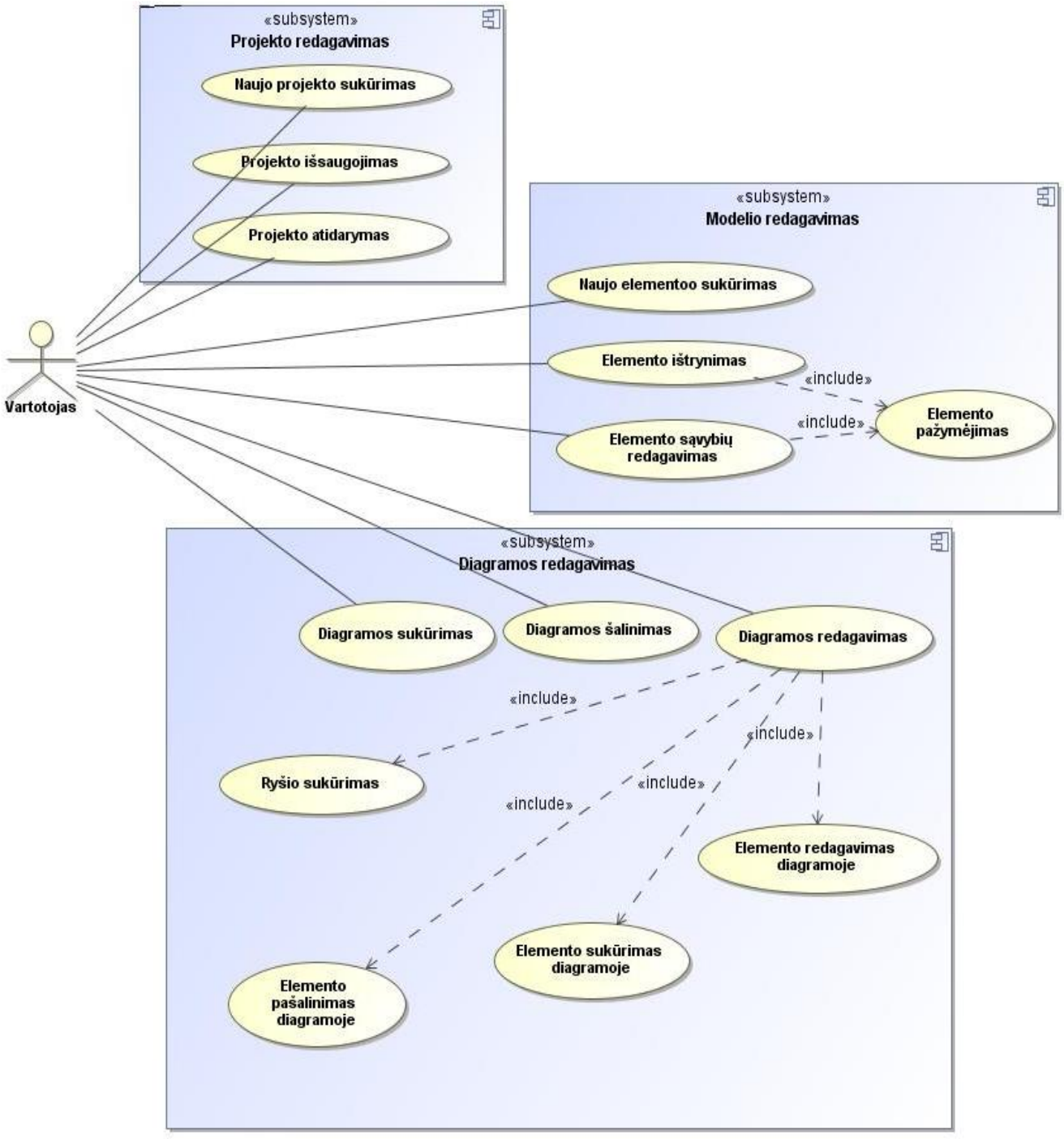
Šio projekto tikslas sukurti Scrall grafinį redaktorių, kuris palaiko UML 2.0 standartą. Siekiama įdiegti Scrall redaktorių į UML standartą su OCL apribojimais. Scrall redaktorius skirtas palengvinti ir supaprastinti veiklos ir būsenų diagramų braižymą. Scrall palengvina tarpinių sistemos būsenų atvaizdavimą, duomenų perdavimo atvaizdavimą ir sumažinti bei supaprastinti braižomų diagramų apimtis.

Visos sistemos ir vartotojo ribos pateikiamos 1 pav. Šiame projekte pagrindinis dėmesys skiriamas Scrall modelio sudarymui.



9 pav. Sistemos ir vartotojo ribos

3.1. Scrall redaktoriaus funkciniai reikalavimai



10 pav. Funkcinių reikalavimų panaudos diagrama

Scrall grafinis redaktorius skirtas projektuotojui palengvinti darbą. Su šiuo grafiniu redaktoriumi, galima paprasčiau, lengviau ir greičiau atvaizduoti norimus programų sistemos elementus ir jų tarpusavio ryšius. Scrall grafinis redaktorius skirtas projektuotojui kurti ir lengviau valdyti visą informacinę sistemą.

Scrall redaktoriaus posistemės:

1. Projekto redagavimas – skirta atvaizduoti visas projekto kūrimo ir redagavimo savybes.
2. Modelio redagavimo posistemė – skirta sujungti visas modelio redagavimo funkcijas.
3. Diagramos redagavimo posistemė – skirta sujungti visas diagramos kūrimo ir redagavimo funkcijas.

Funkciniai reikalavimai:

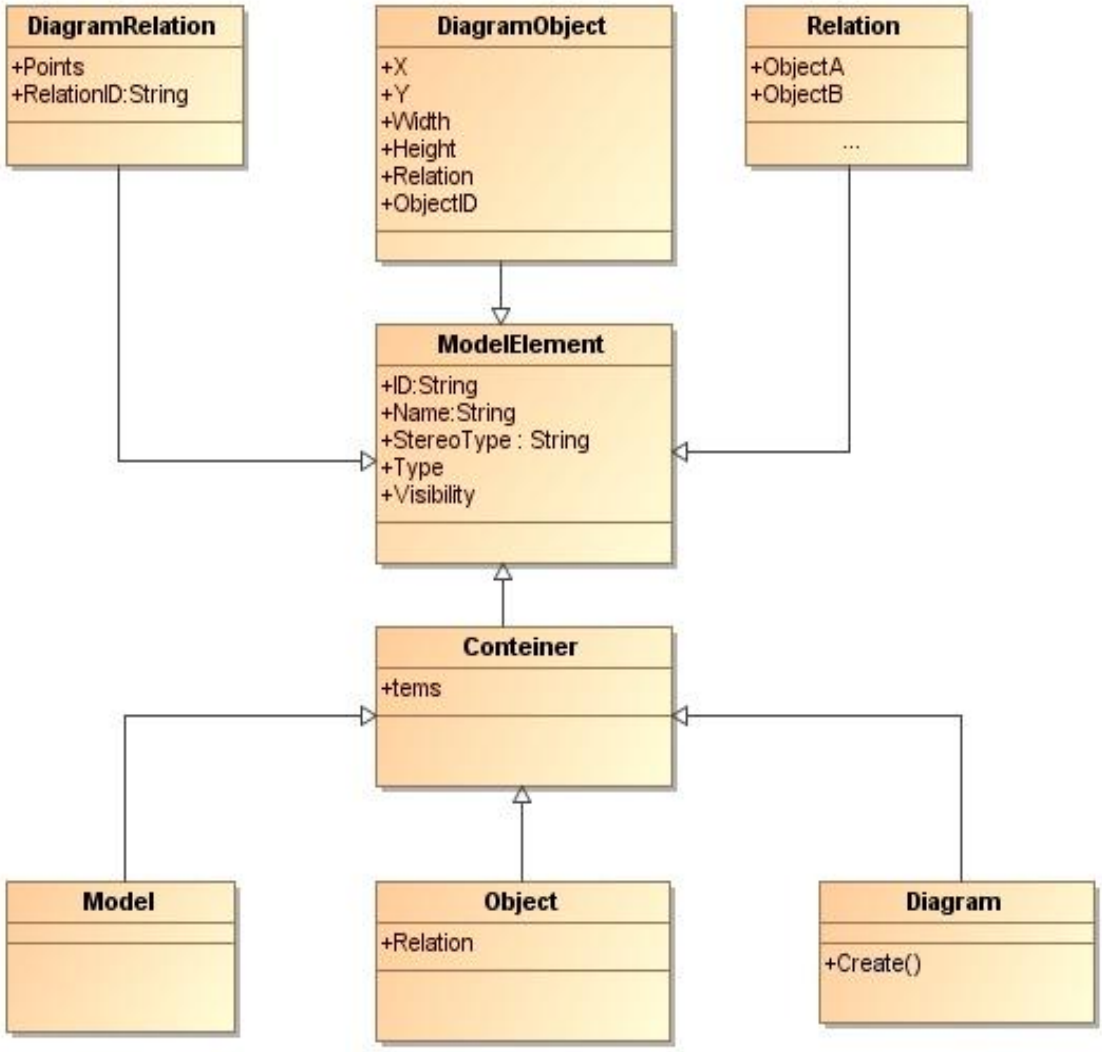
1. Sistema turi suteikti galimybę kurti visiškai naujus projektus esančių įskiepių pagalba. Vartotojas, kuria naujus projektus pagal pasiūlytus šablonus.
2. Sistema turi suteikti galimybę kurti grafinius Scrall modelius. Turi būti pateikiamas įrankis, kurio pagalba grafiškai išreiškiamas Scrall modelis.
3. Sistema privalo leisti redaguoti diagramas:
   3.1. Kurti diagramas;
   3.2. Keisti diagramų išsidėstymą;
   3.3. Keisti diagramos ryšius;
   3.4. Trinti diagramas;
   3.5. Išsaugoti diagramas.
4. Sistema turi leisti redaguoti diagramos elementus:
   4.1. Kurti elementus;
   4.2. Keisti elementų padėtį;
   4.3. Keisti elementų išsidėstymą;
   4.4. Keisti elementų vardus
   4.5. Trinti elementus;

4.6. Išsaugoti elementus.

5. Sistema turi leisti redaguoti diagramoje esančių elementų ryšius:
   5.1. Kurti ryšius tarp elementų;
   5.2. Redaguoti ryšius tarp elementų;
   5.3. Trinti ryšius tarp elementų.
6. Sistema privalo užtikrinti visišką diagramų išsaugojimą bet kurioje diagramos kūrimo stadijoje.
7. Grafinių Scrall modelių importavimo funkcija privalo suprasti XML formatą. Importavimas turi atitikti konkrečios XML specifikaciją. XML importavimas palaiko XMIL 2.0 specifikaciją.
8. Scrall grafinį modelį turi būti galima išsaugoti kaip paveiksliuką .jpg arba .bmp formatu.
9. Grafinių Scrall modelių eksportavimo funkcija privalo suprasti XMI formatą.
10. UML į kurį diegiamas Scrall grafinis redaktorius privalo palaikyti OCL apribojimus susietus su UML modelio objektais. Turi būti palaikoma OCL 2.0 specifikacija.
11. Elementai, kuriuos norima redaguoti pažymimi su pele apibrėžiant tą sritį.
12. Kuriant nauja elementą suteikiama galimybė nustatyti elemento dydį, išskyrus tiems elementams, kurie turi dydžio apribojimus.

3.2. Redaktoriaus reikalavimai duomenims

Scrall redaktoriaus reikalavimai duomenims pateikti 11 paveikslėlyje.



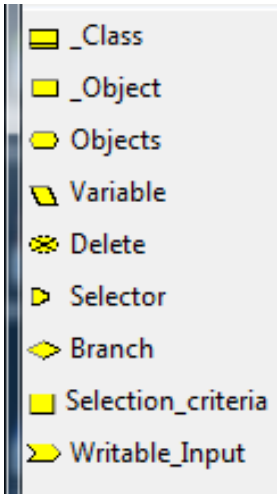
11 pav. Reikalavimai duomenims

3.3. Reikalavimai vartotojo sąsajai

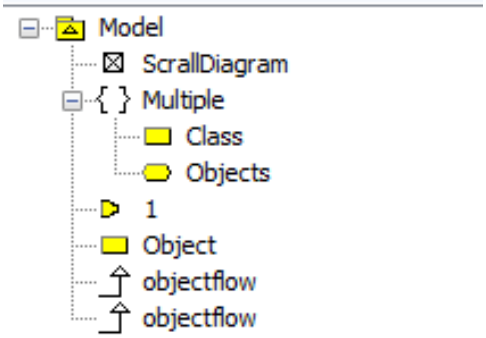
Vartotojo sąsajos specifikacija padeda klientui lengviau suprasti reikalavimus, jis lengviau įsitraukia i reikalavimų rinkimo bei analizės procesą. Reikalavimai detaliau išsiaiškinami dar reikalavimų rinkimo fazėje, ir išvengiama reikalavimų pasikeitimų vėlesnėse sistemos kūrimo fazėse.

1. Išvaizda turi būti panaši į standartinius modeliavimo įrankius. Sistemos vartotojai yra profesionalai, kurie jau yra susidūrę su panašiais modeliavimo įrankiais, todėl išvaizda artima modeliavimo įrankiams būtų privalumas. Standartiniai meniu, grafiniai elementai matomi ir pasiekiami pagrindiniame lange. Didžioji lango dalis paliekama modeliui.
2. Vartotojas turi turėti galimybę paprastai ir lengvai įsijungti Scrall redaktorių (1 priedas).
3. Scrall diagramos braižymas pagrindiniame lange. Vartotojas turi lengvai suprasti kur kurti ir kaip braižyti elementus.
4. Aiškios diagramos redagavimo galimybės. Vartotojas turi galėti redaguoti elementus naudodamas kuo paprastesnes funkcijas.
5. Aiškio diagramos objekto ištrynimo funkcijos. Vartotojas turi sugebėti ištrinti objektą arba ryšius naudodamas standartines funkcijas. Turi būti numatyta galimybė ištrinti tiek pagrindiniame lauke tiek diagramos struktūriniame diagramos medyje.
6. Modelio braižymo langas turi susidėti iš trijų pagrindinių dalių: braižomų objektų sąrašas, braižymo langas ir struktūrinis braižomo modelio langas (13 pav.).
7. Vartotojo sąsaja neturi turėti ryškių, vartotoją erzinančių ar blaškančių spalvų ir paveiksliukų.
8. Sistema privalo palaikyti lietuviškus simbolius, kaip: ąčęėįšųū.
9. Visi braižomi objektai ir jų ryšiai atvaizduojami elementų pasirinkimo lauke (12 pav.)
    - Braižomam elementui turi būti pritaikyti apribojimai, apie kuriuos informuojamas vartotojas
    - Braižomas objektas taip pat atvaizduojamas ir struktūriniame medyje.
    - Pagal Scrall taisykles dalis braižomų objektų turi turėti fiksuotą dydį

- Braižomi ryšiai tarp elementų turi būti apriboti pagal Scrall veiklos taisykles.



12 pav. Objektų parinkimo meniu



13 pav. Struktūrinis braižomo modelio medis

3.4. Nefunkciniai reikalavimai

Nefunkciniai reikalavimai turi didelę įtaka kuriamos sistemos architektūrai, taigi ir pasirinktam sistemos realizavimo sprendimui. Šie reikalavimai skirti tam, kad vėliau būtų išvengta pakeitimų programinės įrangos kūrime.

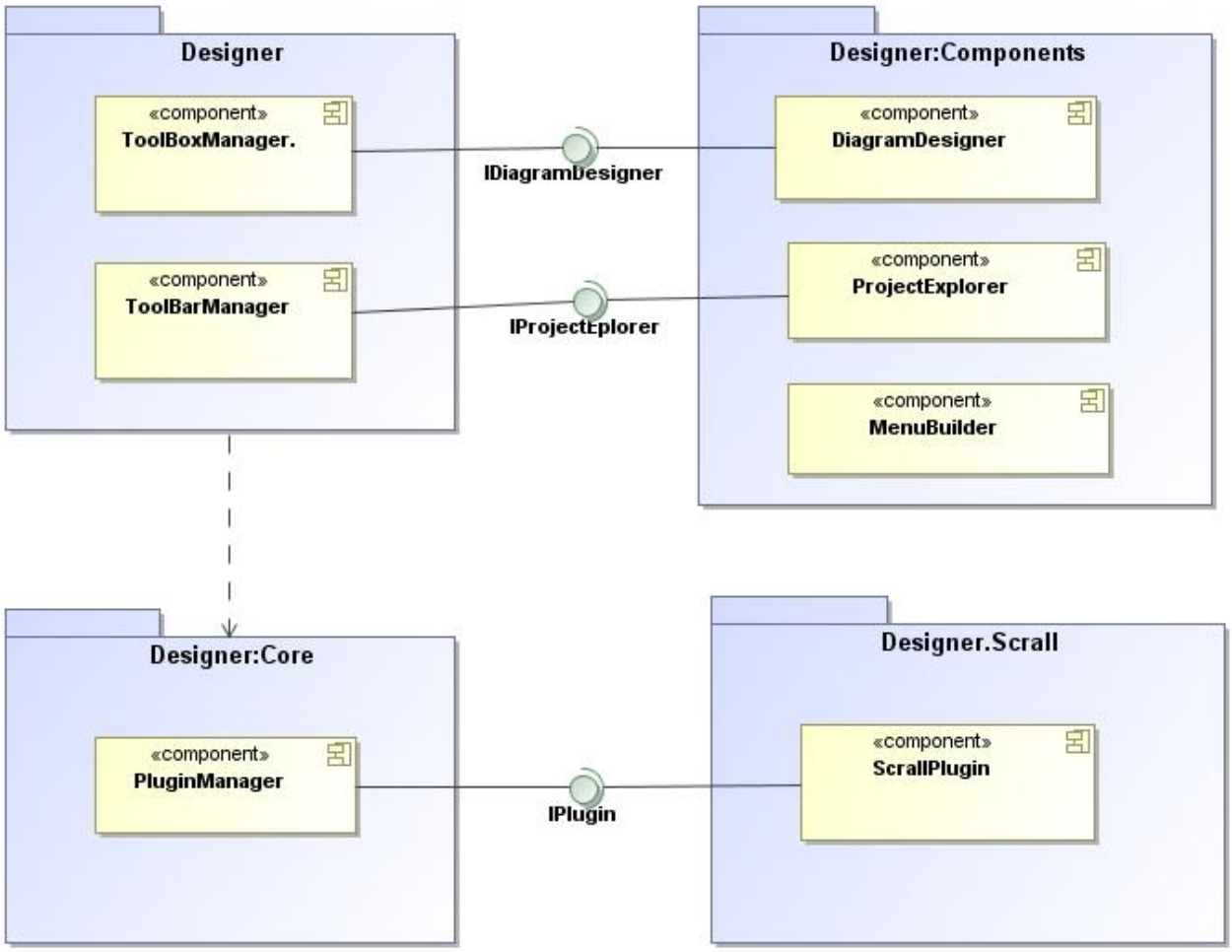
1. Išvaizda turi būti panaši į standartinius modeliavimo įrankius. Nesunkus dirbančiųjų su kuriama programine įranga apmokymo procesas. Kuo sistema greičiau suprantama tuo darbas tampa greitesnis ir efektyvesnis
2. Duomenų saugojimas turi būti greitas. Sistema negali trukdyti nuosekliam vartotojo darbui. Duomenų saugojimas trunka ne ilgiau kaip 2 sekundes.
3. Sistema turi veikti stacionariame kompiuteryje su Windows operacine sistema. Dirbama su personaliais kompiuteriais, kuriuose įdiegta Windows operacinė sistema.
4. Reikalingas sistemos palaikymas iškilus klaidoms. Iškilusios problemos ar klaidos turi būti sprendžiamos.
5. Sistemoje negalima naudoti ką nors įžeidžiančių terminų ar iliustracijų, ji turi būti neutrali. Sistema gali naudotis skirtingi vartotojai. Turi būti naudojami standartiniai ir aiškūs terminai.
6. Sistema gali naudotis tik legalia programine įranga. Nelegalios programinės įrangos naudojimas yra nepriimtinas, privaloma naudoti tik licencijuotą programinę įrangą.
7. Elementai bei jų ryšiai turi būti intuityviai suprantami.
8. Elementai bei jų ryšiai turi būti lengvai iškviečiami.
9. Programinė įranga turi būti apsaugota nuo pašalinių asmenų kodo pakeitimo.
10. Programinė įranga turi būti saugi naudoti operacinėje sistemoje.

3.5. Architektūros reikalavimai ir apribojimai

Kuriama sistema yra realizuota naudojant .NET 2.0 karkasą. Šis karkasas yra skirtas dirbti su Windows operacine sistema. Nors yra tokia galimybė, kurią galima pritaikyti Linux operacinėje sistemoje, tačiau dėl laiko ir lėšų stygiaus nuspręsta tai palikti vėlesniuose darbuose. Siekiant pritaikyti programą abiems sistemoms užtruktų labai daug laiko ir kiltu grėsmė neįvykdyti užsibrėžtų tikslų.

Kadangi projekto įgyvendinimą riboja ne tik laikas bet ir lėšos, atsisakyta visų komercinių komponentų. Visą sistemą bei jos sąsają sudarys .NET karkasas ir visi ją tenkinantys komponentai.

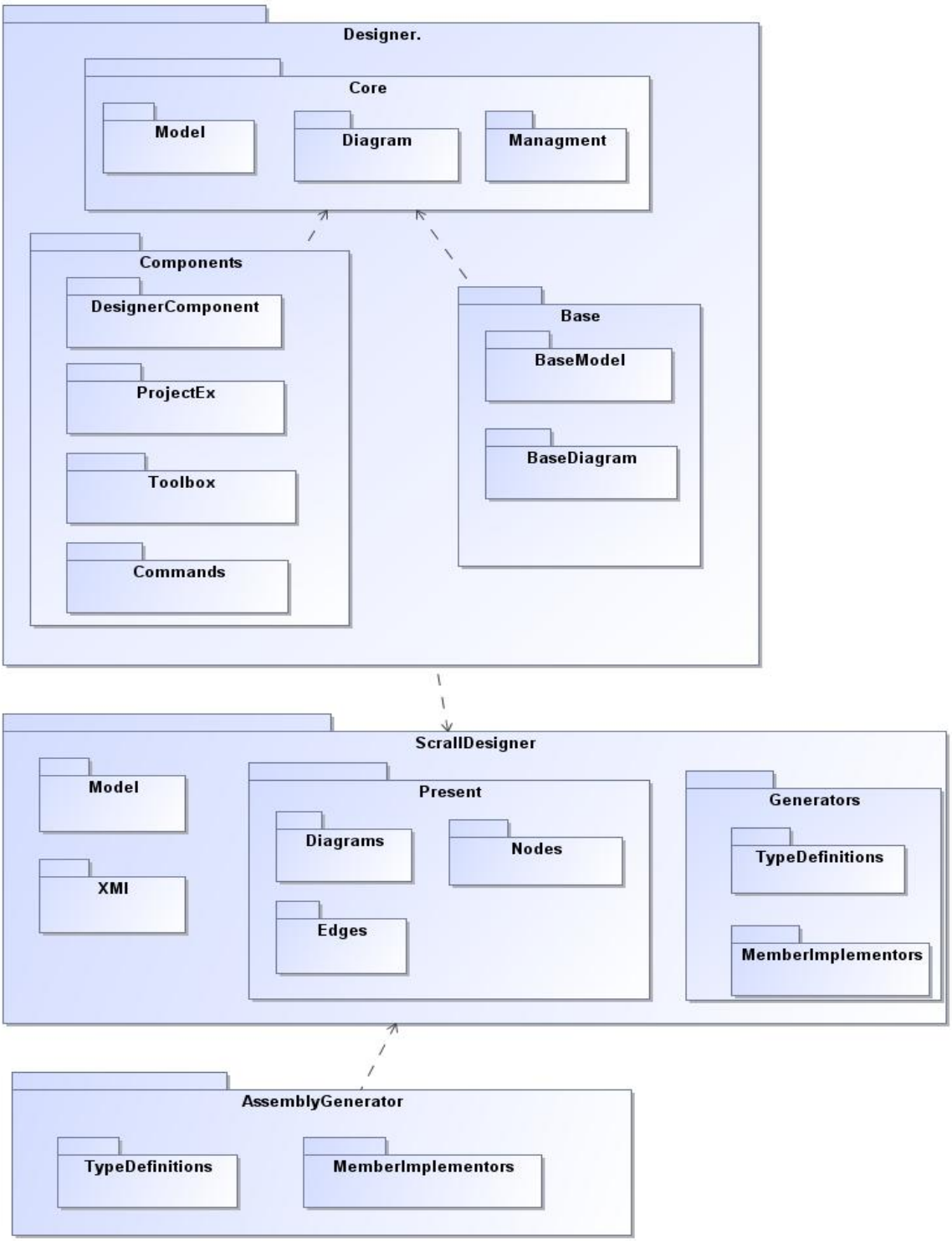
3.6. Statinis sistemos vaizdas



14 pav. Statinis sistemos vaizdas

Pagrindiniai sistemos komponentai:

- DiagramDesigner – skirtas diagramų redagavimui.
- ProjectExplorer – skirtas kuriamo ir redaguojamo medžio atvaizdavimui. Taip susijęs su visomis redagavimo funkcijomis.
- MenuBulder – skirtas kuriamos diagramos meniu elementų atvaizdavimą.
- ToolBoxManager – leidžia parinkti elementus ir sukuria įrankių juostas.
- ToolBarManager – leidžia papildomas diagramų valdymo taisykles.
- PluginManager – atsakingas už visus įskiepius, bei jų valdymą.



15 pav. Scrall paketų diagrama

Paketai

Designer – šiame pakete yra visi Scrall redaktoriaus paketai ir elementai.

- Core – saugomi visi svarbiausi interfeisai ir klasės. Tai yra sistemos pagrindinė dalis.
    - Model – aprašomi modelio elementui;
    - Diagram – aprašomos diagramos ir jų elementai;
    - Managment – aprašomi įskiepių interfeisai ir jų valdymas.
- Components – saugomi baziniai Scrall sistemos komponentai.
    - DesignerComponent – komponentas, kuris suteikia galimybe redaguoti diagramas.
    - ProjectEx – komponentas, kurio pagalba atvaizduojama visa diagramos medžio struktūra.
    - Toolbox – komponentas, kuris atvaizduoja įrankių juostas ir pagalbinius interfeisus.
    - Commands – komponentas, kuris vaizduoja bazines klasės diagramas, bei realizuoja modelio redagavimo funkcionalumą.
- Base – laikomos bazinės klasės, kurios realizuoja modelio ir diagramų elementus
    - BaseModel – modelio kūrimui laikomos bazinės klasės.
    - BaseDiagram – diagramų kūrimui laikomos bazinės klasės.

ScrallDesigner – laikomos Scrall modelio ir diagramų elementai.

- Model – laikomi Scrall modeliai.
- Present – laikomi grafiniai Scrall elementai.
    - Diagrams – laikomos Scrall diagramos.

- Nodes – laikomi diagramų objektai.
- Edges – laikomi diagramų ryšiai.

❖ Generators – laikoma bibliotekos skirtos generuoti klases.

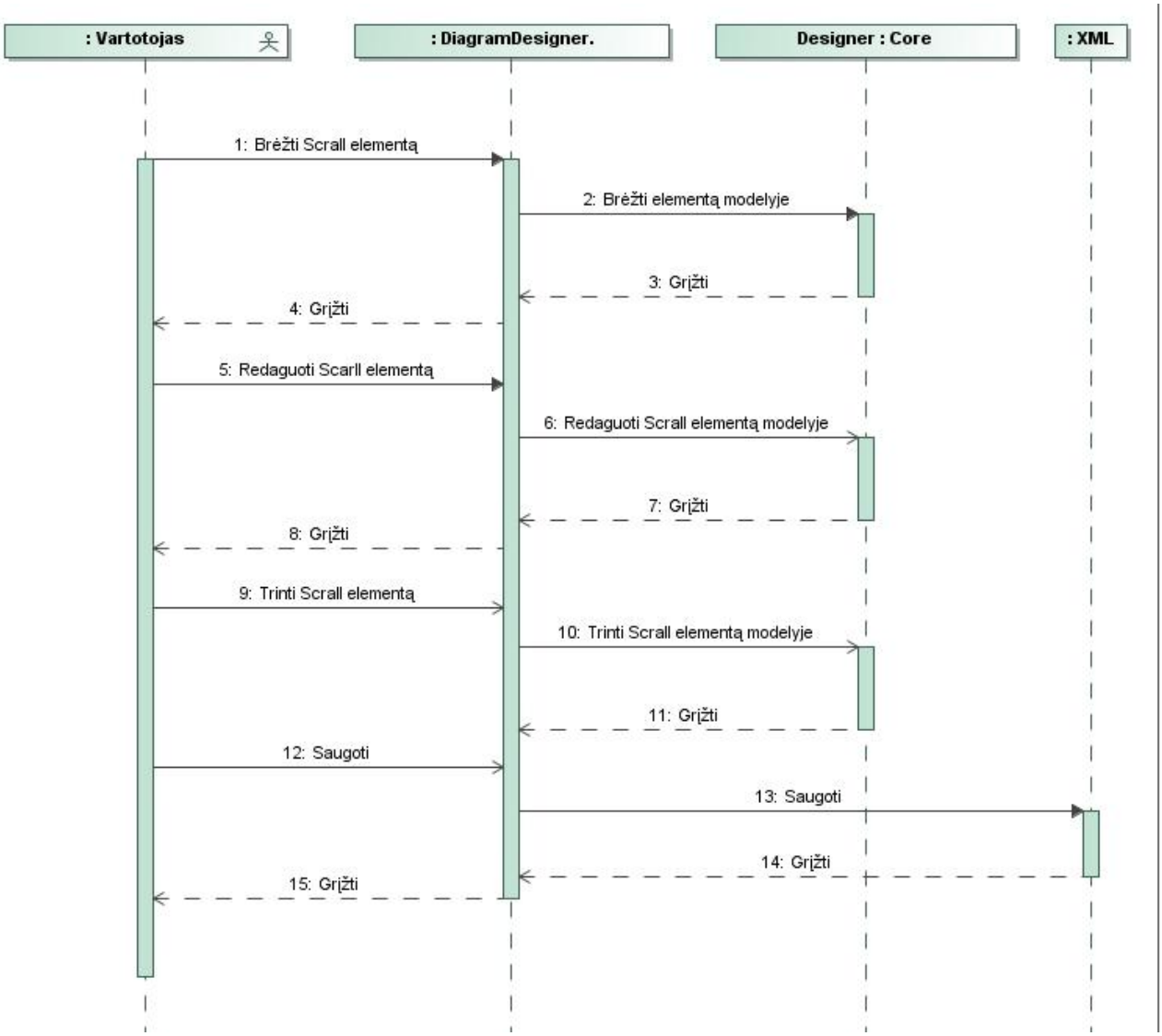
- TypeDefinitions – klasės, kurios aprašo Scrall modelio elementų generavimą.
- MemberImplementors – klasės, kurios yra Scrall elementų sudedamieji komponentai.

AsemblyGenerator – paketas skirtas generuoti .Net bibliotekas

- TypeDefinitors – laikomi klasių interfeisai, numeracija, generavimas.
- MemberImplementors – laikomos visos sudėtinės dalys: metodai, laukai, savybės.

3.7. Sistemos elgsenos modelis

Scrall grafinio redaktoriaus vartotojas (IS projektuotojas), kuris braižo, redaguoja, trina diagramos elementus iššaukia įvairius Scrall diagramos veiksnius, toms procedūroms atlikti. (16 pav.).



16 pav. Scrall sistemos elgsenos modelis

4. SCRALL IR STARUML PALYGINIMAS

4.1. Eksperimento koncepcija

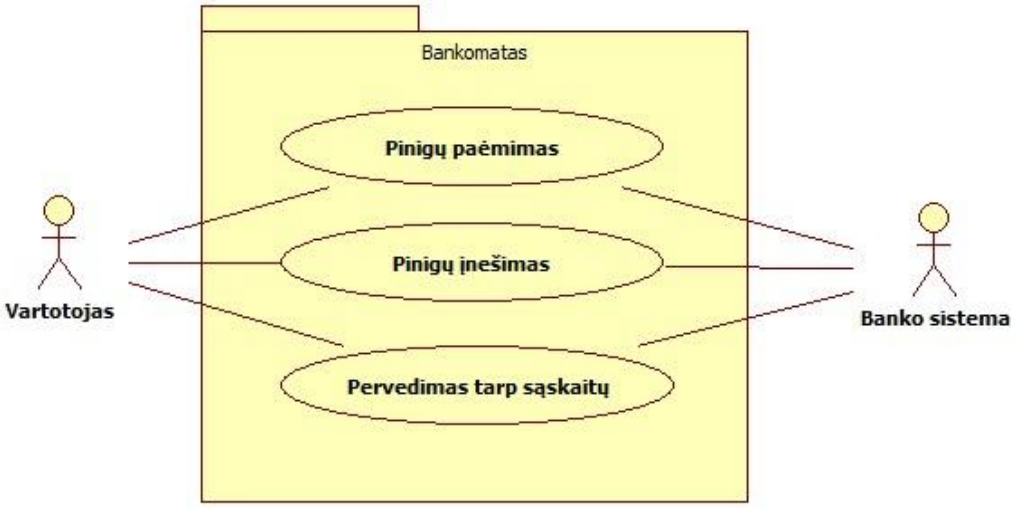
Eksperimentui buvo pasirinkta išanalizuoti bankomato modelio atvaizdavimą pasitelkiant Scrall ir StarUML grafinius redaktorius. Šioje dalyje, norima apžvelgti pagrindines diagramų braižymo taisykles bei apribojimus, išsiaiškinti, kuris modelis yra tinkamesnis. Šios sistemos yra pasirinktos dėl pagrindinės priežasties: Scrall grafinis redaktorius yra evoliucionavęs į StarUML. Pagrindinė dalykinės srities paskirtis yra inicijuoti įvairias Scrall ir StarUML braižymo taisykles.

Nagrinėjimas bankomato modelis, kuriame bendrauja žmogus ir bankomatas. Papildomai yra naudojama sistema ir banko kortelė. Kadangi, Scrall grafinis redaktorius yra skirtas daugiau atvaizduoti sistemų tarpusavio sąveiką ir kokia veikla tarp jų atliekama, tai tokios diagramos kaip klasės ar elgsenos bus vaizduojamos tik su StarUML redaktoriumi.

Bankomatai, šiuolaikinėje visuomenėje yra labai populiarus ir atlieka nemažą vaidmenį. Į bankomatus bankai investuoja didelius pinigus, siekiant, kad žmonės patogiai, saugiai ir bet kokiu paros metu galėtų pasiimti pinigų. Pinigų išėmimo metu sistema ne tik privalo taisyklingai identifikuoti asmenį, bet užtikrinti, kad pinigų išėmimas būtų saugus ir patikimas, bei pinigai būtų nuskaičiuojami būtent nuo to asmens sąskaitos. Tam, kad bankomato sistema taisyklingai, reikalingos papildomos saugumo priemonės, kaip banko kortelė ir saugos kodas.

Pagrindiniai panaudos atvejai atvaizduoti 17 paveikslėlyje. Be visų šių priemonių, turi būti įdiegta pakankamai aiški ir suprantama vartotojui sistema. Sistema turi būti kuo paprastesnė, nes orientuojamasi į visus žmones, neišskiriant nei amžiaus nei specialybės ar išsimokslinimo lygio. Taip pat, pinigų išdavimas turi būti apribotas, tam tikromis sąlygomis.

Bankomato tikslas yra užtikrinti korektiškas pinigų operacijas



17 pav. Bankomato panaudos atvejų modelis

Bankomato aktoriai:

Banko vartotojas. Tai asmuo, kuris tam tikrame banke, sąskaitoje turi pasidėjęs pinigų. Taigi ir to banko bankomatuose jis turi teisę pinigus pasiimti, įnešti naujus arba pervesti į kitą sąskaitą.

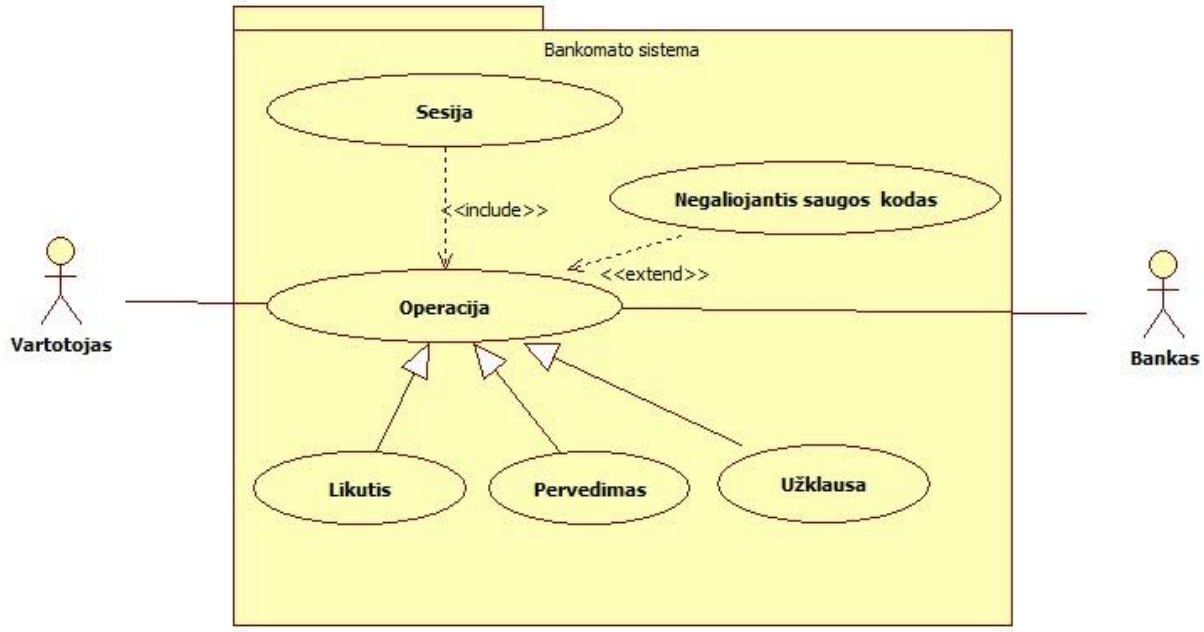
Banko sistema. Tai programinė sistema, kuri aptarnauja vartotojus ir tų vartotojų sąskaitas. Šis sistema yra atsakinga už korektišką pinigų apskaitą ir už veiksmų galingumą.

Pagrindiniai panaudos atvejai:

Pinigų paėmimas. Banko vartotojas turi teisę pasiimti pinigų iš savo banko sąskaitos. Bankinė sistema turi nukreipti vartotoją į savo sąskaitą, iš kurios gali pasirinkti kokia pinigų suma paimti, kuri bus nuskaičiuojama būtent iš tos sąskaitos.

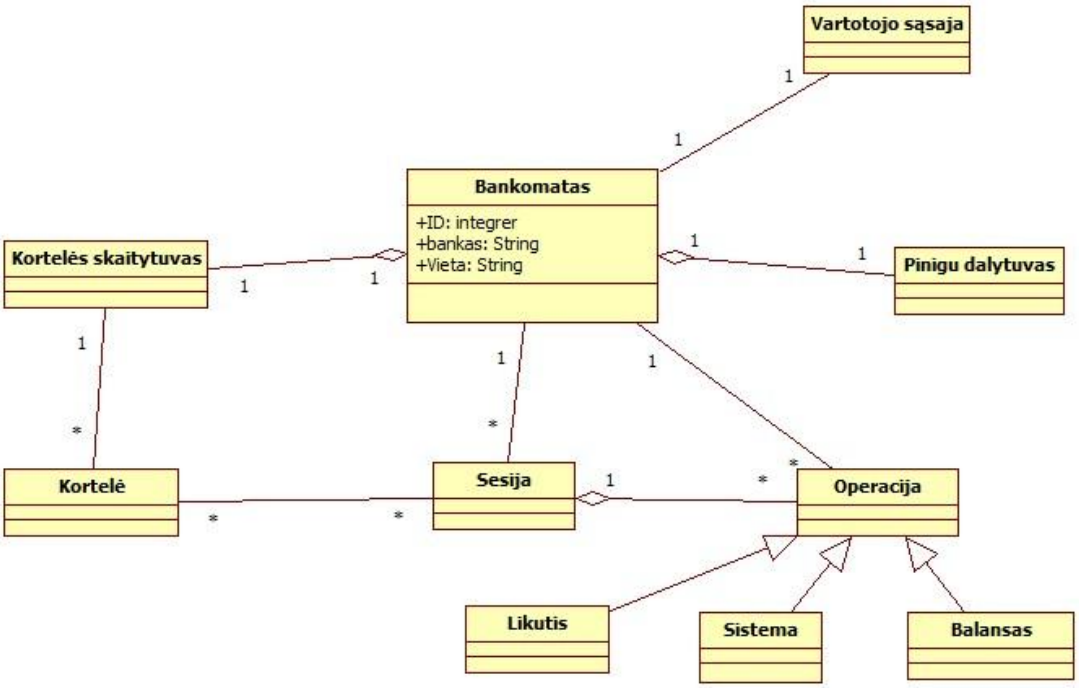
Pinigų įnešimas. Banko vartotojas turi teisė bankomato pagalba įnešti pinigus į savo sąskaitą. Bankinė sistema įneštus pinigus turi priskaičiuoti į tą sąskaitą. Ši funkcija yra įdiegta ne į visus bankomatus.

Pervedimas tarp sąskaitų. Banko vartotojas turi teisę savo pinigus pervesti iš vienos sąskaitos į kitą.

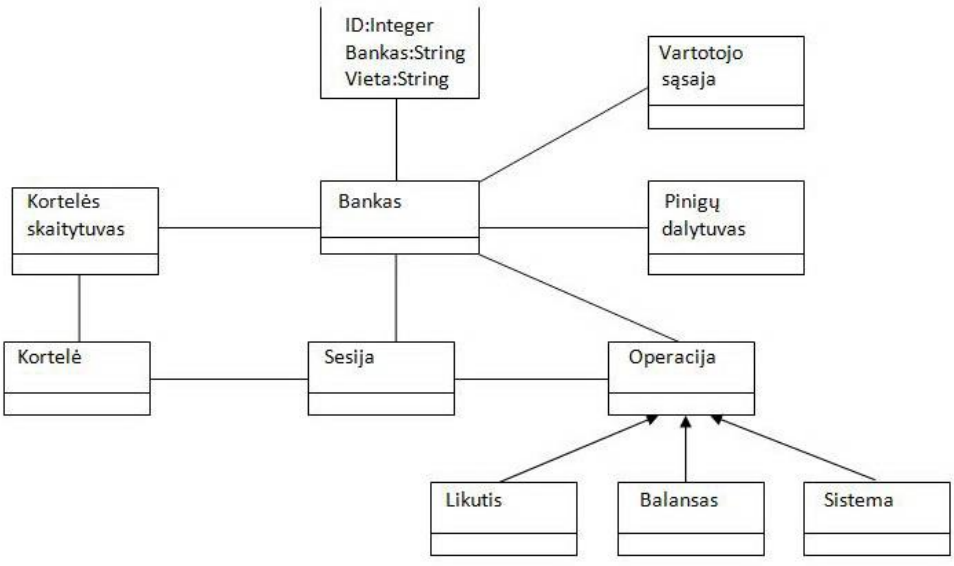


18 pav. Bankinės sistemos panaudos atvejai

Visa bankomato pagrindinė sistema dažniausiai yra atvaizduojama klasių diagrama. Pagrindinėje klasių diagramos struktūroje atvaizduojami pagrindiniai komponentų ryšiai bei tarpusavio sąveika. Scrall redaktorius nėra pritaikytas vaizduoti klasių diagramas, tad iš turimų elementų galima atvaizduoti tik pagrindinę struktūrą. Į StarUML redaktorių yra įdiegta klasių braižymo funkcija, tad su šiuo redaktoriumi neiškyla jokių problemų. Bankomato klasių diagrama pavaizduota 19 ir 20 pav.



19 pav. Bankomato modelio klasių diagrama (StarUML)



20 pav. Bankomato modelio klasių diagrama (Scrall)

Kaip matote su Scrall redaktoriumi atvaizduoti kasių diagramą yra pakankamai sudėtinga braižyti. Šio redaktoriaus pagalba galima tik tai atvaizduoti kokios yra

egzistuojančios klasės ir kurios jų tarpusavio sąveikos. Scrall redaktoriuje nėra numatytos galimybės atvaizduoti klasės atributus ir operacijas.

StarUML redaktorius yra visiškai pritaikytas braižyti klasių diagramas. Jos braižomos greitai ir suprantamai. Jis beveik niekuo nesiskiria nuo kitų įprastų UML redaktorių.

Klasės:

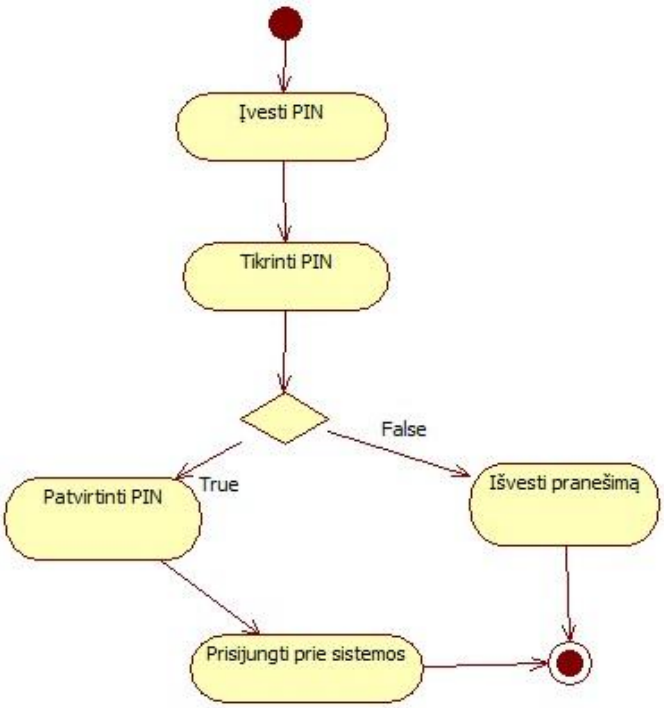
Bankas – klasėje bankas aprašomi pagrindiniai sistemos komponentai (atributai). Tai pagrindinė klasė, kuri yra tarpininkė tarp vartotojo ir bankinės sistemos.

Vartotojo sąsaja – sąsaja, kuri padeda vartotojui suprantama kalba bendrauti su bankomatu.

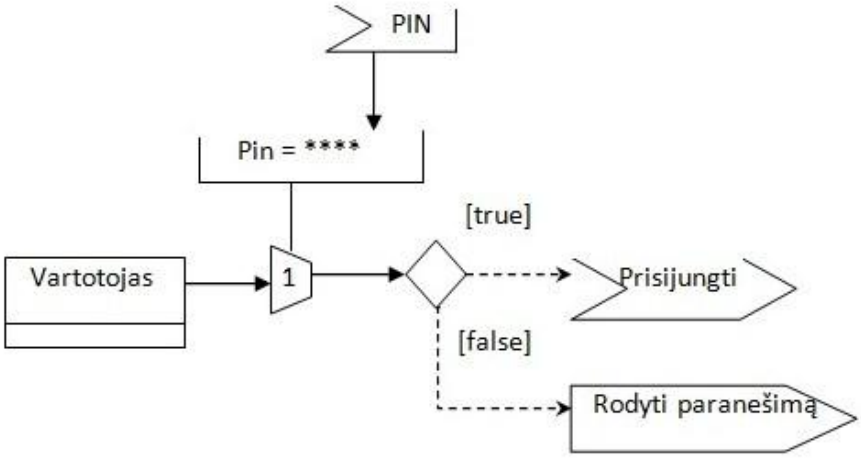
Pinigų dalytuvas – tai klasė, kuri yra atsakinga už pinigų išdavimą vartotojui.

Kortelės skaitytuvas – tai klasė, kuri yra atsakinga už kortelės duomenų nuskaitymą ir tinkamo vartotojo identifikavimą.

Kortelė – tai klasė, kuri atsakinga už kortelėje esančius duomenis bei jų saugumą.

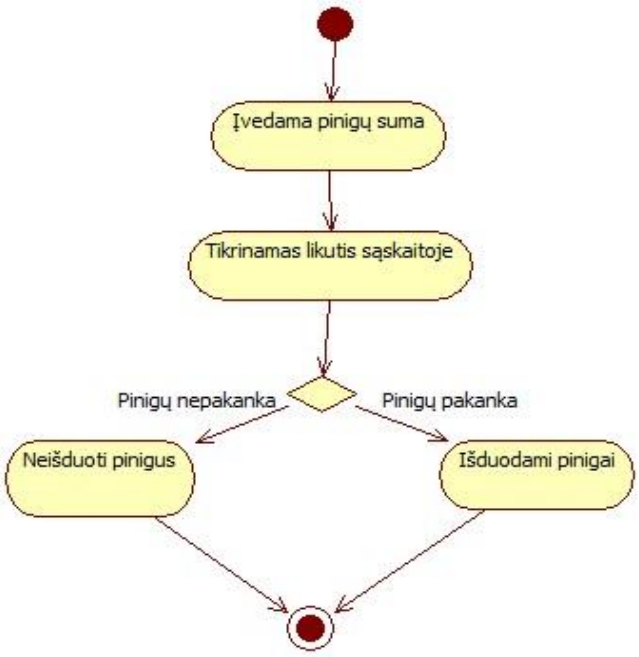


21 pav. Veiklos diagrama prisijungimui prie bankomato sistemos (StarUML)

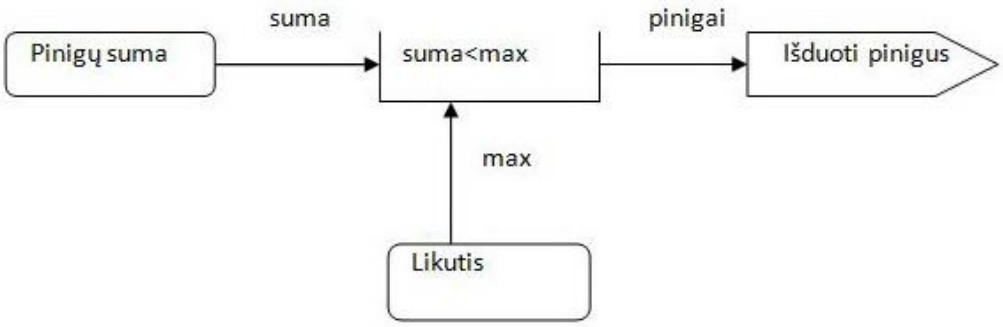


22 pav. Scrall diagrama prisijungimui prie bankomato sistemos

Diagramos aprašančios vartotojo prisijungimą prie sistemos su Scrall redaktoriumi (21 pav.) ir su StarUML redaktoriumi (22 pav.) yra visiškai skirtingos, nors siekiama atvaizduoti tą patį. Scrall diagrama yra truputį detalesnė, kadangi yra numatyta galimybė atvaizduoti kokie yra įvedamieji duomenys (PIN), bei iškirta atskira klasė, skirta nurodyti kas tuos duomenis įveda. Iš simbolio, žyminčio išsišakojimą visada ryšys bus vaizduojamas punktyrine linija.



23 pav. Veiklos diagrama pinigų išdavimui vartotojui (StarUML)



24 pav. Scrall diagrama pinigų išėmimui iš bankomato

Ne taip kaip paprastoje veiklos diagramoje su Scrall redaktoriumi paprasčiausias funkcijas galima atvaizduoti labai elementariai ir paprastai (24 pav.). Nereikalaujama braižyti pradžios arba pabaigos taškų. Visada Scrall diagramos skaitomos iš kairės į dešinę arba iš viršaus į apačią. Ne visada reikalinga braižyti atskirą išsišakojimo simbolį. Scrall diagramoje užtenka su pasirinktu elementu nurodyti prie kokios sąlygos bus taip vykdoma. Žinoma, jei norima pavaizduoti, kad esant skirtingom situacijom bus atliekami skirtingi veiksmai, tada išsišakojimo simbolis bus reikalingas.

4.2. Scrall ir StarUML grafinio redaktoriaus įvertinimas

„+“ – yra, „-„ – nėra

„1“ – blogai, „2“ – vidutiniškai, „3“ – gerai

3 lentelė. Scrall ir StarUML grafinių redaktorių įvertinimas

Kriterijus	Scrall	StarUML
Patogu braižyti elementus	2	3
Patogu redaguoti elementus	3	3
Patogu trinti elementus	3	3
Patogu elementus išdėstyti	3	3
Diagramos taisyklių įsisavinimas	2	1
Diagramos suprantamumas	2	1
Paprastumas	3	1
Diagramos išsaugojimas XML faile	+	+
Diagramos išsaugojimas į paveiksliuką	+	+
Diagramos užkrovimas iš failo	+	+
Diagramos pašalinimas	+	+
Diagramos redagavimas	+	+
Scrall įdiegimas į standartines UML diagramas	-	+
Skirtingų diagramų braižymas	-	+
Modelio generavimas į kodą	-	+

5. IŠVADOS

1. Kadangi vis daugiau naujų sudėtingų sistemų yra kuriama arba senos yra atnaujinamos, tai šių sistemų projektavimas tampa vis sudėtingesnis ir svarbesnis. Kadangi ši sritis yra pakankamai paklausi, tai visas šio magistrinio darbas yra susijęs su Scrall grafiniu redaktoriumi kūrimu bei jo analizavimu.
2. Grafinį redaktorių nuspręsta įgyvendinti pasitelkiant .Net karkasą. Tai suteikia galimybę vėlesniam programos vystymui. Realizuojant šį projektą buvo stengiamasi išvengti dalies kodavimo darbų naudojant teikiamas tiesioginės inžinerijos galimybes. Tokiu būdu pasiteisina šio įrankio naudojimas.
3. Scrall redaktorių nuspręsta išanalizuoti ir palyginti su StarUML redaktoriumi, kadangi Leon Starr (Scrall grafinės kalbos sukūrėjas) teigia, kad StarUML yra Scrall evoliucija.
4. Pastebėta, kad StarUML redaktorius yra kiek įmanoma suvienodintas su standartiniais UML redaktoriais. Scrall specifinių elementų neišliko. Taip galėjo atsitikti dėl to, kad Scrall redaktoriaus idėja nebuvo iki galo išbaigta.
5. Scrall redaktorius turi nemažai galimybių būti tobulinamas ir pateikiamas visuomenei. Deja tokį redaktorių riboje ribotos diagramų braižymų galimybės. Visas atvaizdavimas susitelktas tik į duomenų srautų pažymėjimą bei jų tarpusavio sąveiką, jų būsenas ir jų valdymą. Šis redaktorius galėtų būti kaip viena iš galimų UML diagramų, pasirenkamų UML redaktoriaus diagramų sąraše.

6. LITERATŪRA

[1] StarUML [žiūrėta 2010. 05.01] Prieiga internete http://staruml.sourceforge.net/docs/user-guide%28en%29/ch01.html

[2] StarUML [žiūrėta 2010. 05.01] Prieiga internete http://staruml.software.informer.com/

[3] S. Kaukėnas, D. Šilingas. UML panaudojimas IT specialistų darbo optimizavimui. "Informacinės technologijos verslui – 2003" konferencijos pranešimų medžiaga, VU KHF, 2003, p.p. 39-44.

[4] Adriana BUZGAR, Introduction to UML 2.0, Konstanz University, Socrates-Erasmus Exchange "Alexandru Ioan Cuza" University, 2006

[5] Cameron Sturdevant SmartDraw 7s Gains Arent Just Symbolic [žiūrėta 2010.05.11], prieiga internete http://www.eweek.com/c/a/IT-Infrastructure/SmartDraw-7s-Gains-Arent-Just-Symbolic/

[6] Randy Miller, Practical UML: A Hands-On Introduction for Developers, [žiūrėta 2010. 05.06] Prieiga internete http://edn.embarcadero.com/article/31863

[7] Visual OCL – User guide [žiūrėta 2010.05.11], prieiga internete http://tfs.cs.tu-berlin.de/vocl/userguide/group1/index.html

[8] Rajni Pamnami, Pramila Chawan, Satish Salunkhe, Object Oriented UML Modeling for ATM Systems, VJTI University, Mumbai

[9] Bank managment, By G.Shakir , [žiūrėta 2010. 05. 20] Prieiga internete http://www.scribd.com/doc/2909460/UML-DIAGRAMSBANK-MANAGEMENT

[10] UML, prieiga internete [žiūrėta 2008.05.20] , http://jdjua.com/uml.htm

[11] Leon Starr, Scrall, Starr's Concise Relation Action Language, Model Integration, LLC, 2003

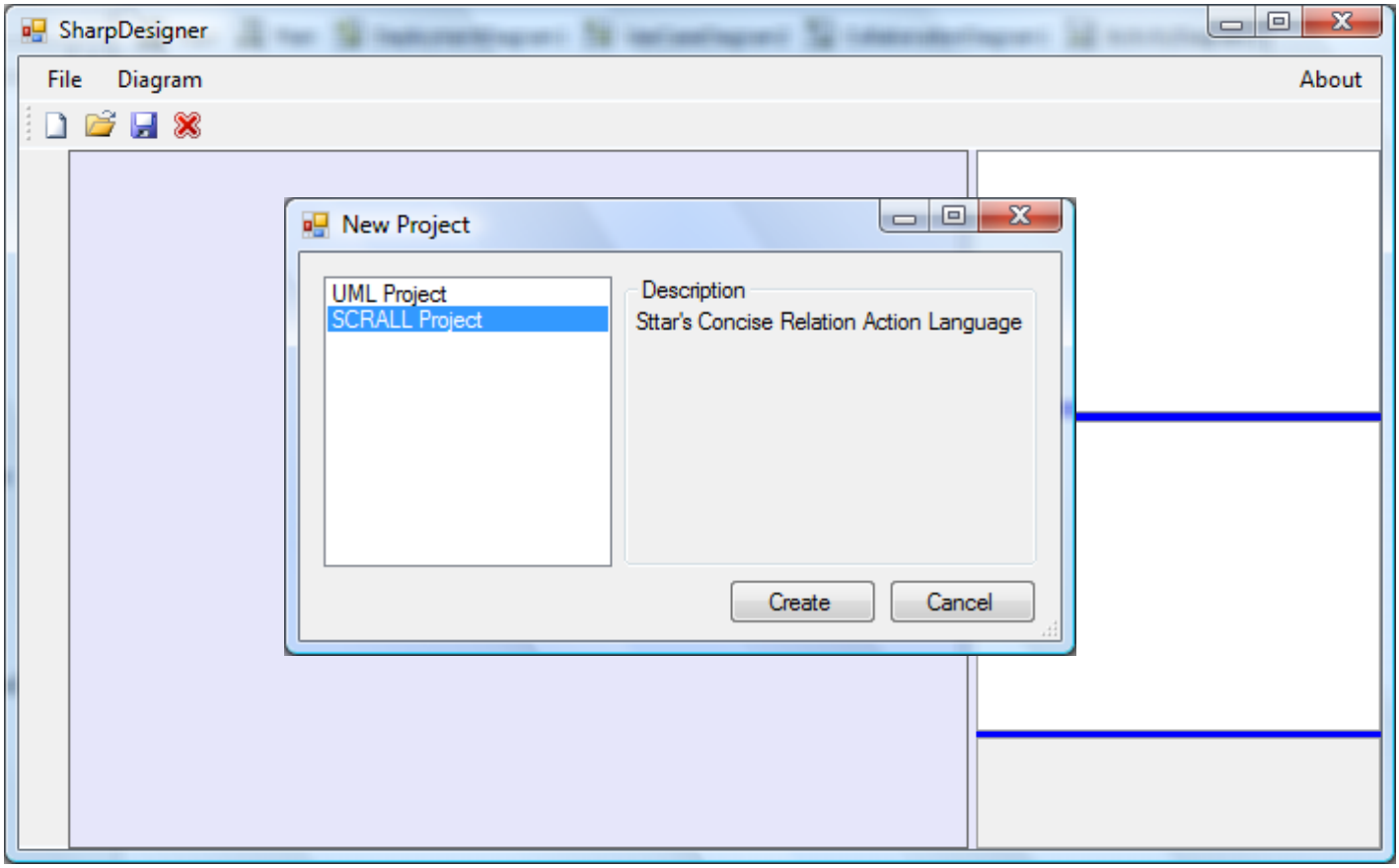
[12] Leon Starr, Scrall, Elevator Example Action, Model Integration, LLC, 2003

7. TERMINŲ IR SANTRUMPŲ ŽODYNAS

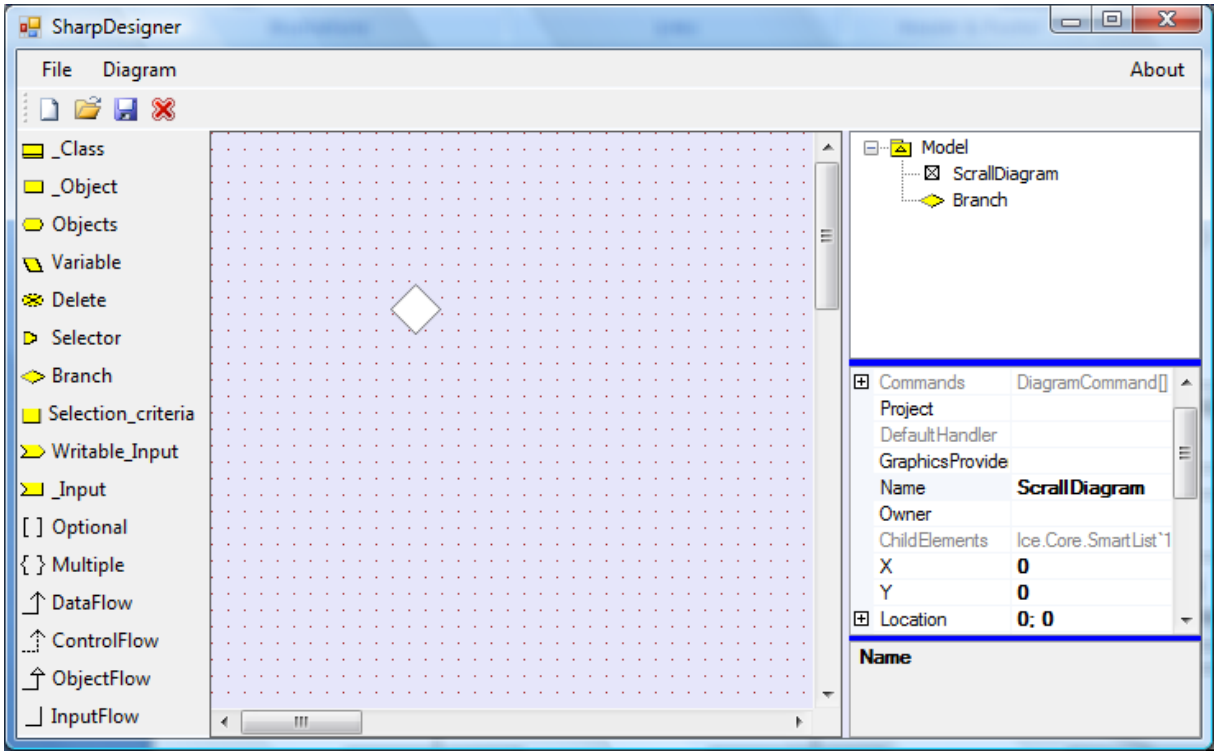
Rational Rose	Tai CASE priemonė skirta PĮ projektavimui bei analizės etapų automatizavimui, taip pat kodų generavimui.
Object Management Group (OMG)	Tai konsorciumas, kuris iš pradžių buvo skirtas nustatyti ir paskirstyti Objektinės sistemos standartus, o dabar yra orientuota į modeliavimą (programų, sistemų ir verslo procesų) bei modelio paremtus standartus.
XMI	XML meta duomenų tarpusavio apsikeitimas.
MDA	modeliu paremta architektūra tai metodas aprašantis modelių transformacijas į kodą.
OCL	(Object Constraint Language) objektų apribojimo kalba.
SCRALL	(Starr's Concise Relational Action Language) grafinė modeliavimo kalba.
XML	(Extensible Markup Lanuage) išplėstinė žymų kalba.
CASE	(Computer-Aided Software Engineering) programinės įrangos projektavimas kompiuterio pagalba.
Microsoft .NET Framework	Microsoft Windows operacinės sistemos komponentas. Jis suteikia kitoms programoms galimybę naudotis daugybe jau paruoštu įvairių bibliotekų

8. PRIEDAI

Priedas 1. Naujos diagramos sukūrimo vaizdas



Priedas 2. Diagramos kūrimo darbalaukis



Diagramos braižymo darbalaukis