KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PRAKTINĖS INFORMATIKOS KATEDRA

Tomas Mozeris

Veiklos taisyklėmis grindžiamos reikalavimų specifikacijos panaudojimas projektuojant informacines sistemas

Magistro darbas

Darbo vadovas

lekt. dr. K. Kapočius

Kaunas, 2012

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PRAKTINĖS INFORMATIKOS KATEDRA

Tomas Mozeris

Veiklos taisyklėmis grindžiamos reikalavimų specifikacijos panaudojimas projektuojant informacines sistemas

Magistro darbas

Recenzentas

lekt. dr. L. Motiejūnas

2012-05-24

Darbo vadovas

doc. dr. K. Kapočius

2012-05-24

Atliko

IFM-0/4 gr. stud.
T. Mozeris
2012-05-24

Kaunas, 2012

Turinys

SUMMARY **5**

TERMINŲ IR SANTRUMPŲ ŽODYNAS **6**

1. ĮVADAS **7**

2. VEIKLOS TAISYKLĖMIS GRINDŽIAMOS REIKALAVIMŲ SPECIFIKACIJOS PANAUDOJIMO PROJEKTAVIMO METU ANALIZĖ **8**

2.1. ANALIZĖS TIKSLAS 8
2.2. TYRIMO OBJEKTAS, SRITIS IR PROBLEMA 8
2.3. TRADICINIŲ INFORMACINĖS SISTEMOS KŪRIMO METODŲ APŽVALGA 8
2.3.1. Sistemų kūrimo gyvavimo ciklo metodas *8*
2.3.2. Lanksčiųjų metodų apžvalga *10*
2.3.3. Objektiškai orientuotų sistemų analizės ir projektavimo metodas *11*
2.3.4. Informacinių sistemų kūrimo metodų palyginimas *13*
2.4. UML MODELIAVIMO KALBOS APŽVALGA 14
2.5. VT KONCEPCIJOS APŽVALGA 16
2.5.1. GUIDE projekto uždaviniai ir veiklos taisyklių kategorijos *16*
2.5.2. Semantikos veiklos žodyno ir veiklos taisyklių specifikacijos semantika *17*
2.5.3. Produkcinių veiklos taisyklių apžvalga *17*
2.6. VEIKLOS TAISYKLIŲ ĮTAKA TRADICINIAMS INFORMACINIŲ SISTEMŲ PROJEKTAVIMO METODAMS 18
2.7. VT GRINDŽIAMOS REIKALAVIMŲ SPECIFIKACIJOS PANAUDOJIMO PROJEKTUOJANT SISTEMAS GALIMYBĖS 19
2.8. SIEKIAMO SPRENDIMO VARTOTOJŲ ANALIZĖ 21
2.8.1. Vartotojų aibė, tipai ir savybės *21*
2.8.2. Vartotojų tikslai ir problemos *21*
2.9. ANALIZĖS APIBENDRINIMAS 21

3. VEIKLOS TAISYKLĖMIS PAREMTOS SPECIFIKACIJOS TRANSFORMAVIMO Į UML MODELIUS METODAS **22**

3.1. APIBENDRINTA METODIKOS SCHEMA 22
3.2. FUNKCIJŲ (IR VEIKLOS SPRENDIMŲ) TRANSFORMAVIMO Į PANAUDOJIMO ATVEJŲ MODELĮ PROCESO APRAŠAS 23
3.3. STRUKTŪRINIŲ VT (PRAPLĖSTO KONCEPCINIO DUOMENŲ MODELIO) TRANSFORMAVIMO Į DUOMENŲ KLASIŲ DIAGRAMĄ PROCESO APRAŠAS 27
3.4. NESTRUKTŪRINIŲ VT TRANSFORMAVIMO/PRITAIKYMO PROCESO APRAŠAS 29
3.4.1. Atmetimo, apribojimo taisyklių pritaikymas (transformavimas) *29*
3.4.2. Leidimo taisyklės pritaikymas *30*
3.4.3. Skaičiavimo taisyklės pritaikymas *30*
3.4.4. Išvedimo taisyklės pritaikymas *31*
3.4.5. Išvados taisyklės pritaikymas *31*
3.4.6. Taisyklės jungiklio pritaikymas *32*
3.4.7. Proceso jungiklio pritaikymas *32*
3.4.8. Duomenų jungiklio pritaikymas *33*
3.4.9. Reikšmės priskyrimo taisyklės pritaikymas *33*
3.4.10. Pateikimo taisyklės pritaikymas *34*
3.4.11. Proceso trigerio taisyklės pritaikymas *35*
3.4.12. Taisyklės trigerio pritaikymas *35*
3.5. CRUD MATRICOS PRITAIKYMO APRAŠAS 35

4. VEIKLOS TAISYKLIŲ SAUGYKLOS TRANSFORMAVIMO Į UML MODELIUS METODO TYRIMAS **37**

4.1. DALINĖS METODO REALIZACIJOS PROTOTIPO PROJEKTAS 37
4.1.1. Funkciniai reikalavimai *37*
4.1.2. Kompiuterizuojamų procesų modeliai *39*
4.1.3. Vartotojo sąsajos reikalavimai *41*
4.2. APIBENDRINTAS SISTEMOS PROJEKTAS 42

4.2.1. Duomenų struktūros modelis *42*
4.2.2. Sistemos realizacijai pasirinktos priemonės *43*
4.2.3. Funkcijų hierarchijos įvedimo internetinės sąsajos navigavimo planas *44*

5. SUKURTO ĮRANKIO NAUDOJIMO IR FUNKCIONALUMO BANDYMAS 45

5.1. PASIRENGIMAS EKSPERIMENTUI 45
5.2. EKSPERIMENTO SU SUKURTUOJU ĮRANKIU EIGA 46
5.2.1. Funkcijų, aktorių, ryšių tarp aktoriaus ir funkcijos kūrimo eiga *46*
5.2.2. Suformuluotos funkcijų hierarchijos transformavimo į panaudojimo atvejų diagramą eiga *48*
5.3. SUKURTO ĮRANKIO ĮVERTINIMAS 52
5.4. APIBENDRINTAS SUKURTOSIOS METODIKOS ĮVERTINIMAS 53

6. IŠVADOS 54

7. LITERATURA 55

8. PRIEDAI 56

1 PRIEDAS 56

Application of the Business Rules Based Requirements Specification during the Design of Information Systems

Summary

Information system (IS) development process has a relatively consistent structure, although it does not guarantee that all user requirements are represented correctly in the final system. Typically IS development consists of the following phases: requirements capture, analysis and specification, design, development, testing, installation and system support. The complete and correct requirement specification facilitates the further success of the project, therefore it is extremely important. This simple fact is one of the reasons why new approaches, such as those based on the business rules (BR) concept, keep on emerging. According to the BR concept, the business rules are separated from other system constitutive objects, thus creating an environment for a more accurate requirement capture.

The main goal of this work was to examine the business rules based requirements specification method developed at KTU Department of Information system and come up with the ways of how to use the specified requirements during the system design phase. The decision was made to create the requirements-to-UML diagrams transformation methodology, which is presented in this document. The methodology was evaluated using a partial prototype implementation showing it can be applied in practice.

Terminų ir santrumpų žodynas

IS (angl. *Information System*) – Informacinė sistema.

VT – veiklos taisyklė.

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

OMG (angl. *Object Managment Group*) – tarptautinė standartų organizacija.

MDA (angl. *Model Driven Architecture*) – modeliais pagrįsta architektūra.

SBVR (angl. *Semantics of Business Vocabulary and Business Rules*) - semantikos veiklos žodyno ir veiklos taisyklių specifikacija.

PRR (angl. *Production Rule Representation*) – produkcinių veiklos taisyklių vaizdavimas.

ERR (angl. *Enhanced Entity-Relationship model*) – išplėstinis esybių ryšių modelis.

OCL (angl. *Object Constraint Language*)- objektinių reiškinių kalba.

1. Įvadas

Informacijos sistemų (IS) kūrimo procesas turi gan nusistovėjusią struktūrą, tačiau ji neužtikrina aukštos kuriamų sistemų kokybės ir optimalaus atitikimo vartotojų poreikiams. Paprastai IS kūrimas susideda iš šių etapų: reikalavimų surinkimo, analizės ir specifikavimo, projektavimo, realizavimo, testavimo, diegimo ir sistemos palaikymo. Būtent išsamus ir teisingas reikalavimų specifikacijos sudarymas užtikrina tolimesnę projekto sėkmę. Tai inicijuoja naujų metodų kūrimą, kurie akcentuoja veiklos taisyklių (VT) koncepciją. Pagal šią koncepciją IS kūrimo metu taisyklės atskiriamos nuo kitų sistemą sudarančių objektų, taip sukuriant prielaidas tikslesniam reikalavimų fiksavimui.

Šio darbo tikslas išnagrinėti KTU Informacijos sistemų katedroje sukurtą veiklos taisyklėmis grindžiamos reikalavimų specifikacijos sudarymo metodą ir pagerinti tiriamo metodo pritaikomumą sistemos projektavimo stadijoje. Pagal minėtą metodą, baigus reikalavimų specifikavimo procesą, visi surinkti ir struktūrizuoti reikalavimai fiksuojami saugykloje. Analizuojant šiuos duomenis siekiama turimus reikalavimus transformuoti į UML kalbos diagramas.

Siekiant įvykdyti iškeltą tikslą, darbo metu buvo sprendžiami šie uždaviniai:

1. Išnagrinėti VT koncepcijos pagrindus, veiklos taisyklėmis grindžiamos reikalavimų specifikacijos sudarymą ir įtaką IS kūrimo stadijoms.
2. Sukurti metodiką, pagal kurią galima transformuoti duotus reikalavimų specifikavimo rezultatus į UML diagramas.
3. Išbandyti ir įvertinti sukurtą metodiką pasinaudojant specialiai šiam tikslui sukurtu programiniu prototipu.

2. Veiklos taisyklėmis grindžiamos reikalavimų specifikacijos panaudojimo projektavimo metu analizė

2.1.Analizės tikslas

Analizės metu siekiama susipažinti su pagrindiniais informacinių sistemų projektavimo metodais. Ištirti veiklos taisyklių kūrimo principus, VT grindžiamą reikalavimų specifikavimo metodą, jo įtaką skirtingiems kūrimo etapams. Apžvelgti UML kalbos diagramas, jų panaudojimą sistemų kūrime.

2.2.Tyrimo objektas, sritis ir problema

Tyrimo objektas: veiklos taisyklėmis grindžiamos IS reikalavimų specifikacijos panaudojimas sistemos projektavimo metu, jos pritaikymas projektuojant UML diagramas.

Sritis: informacijos sistemų reikalavimų specifikavimo ir projektavimo metodai, akcentuojant veiklos taisyklių koncepcijos sprendimus, reikalavimų specifikacijos panaudojimą modeliuojant UML diagramas.

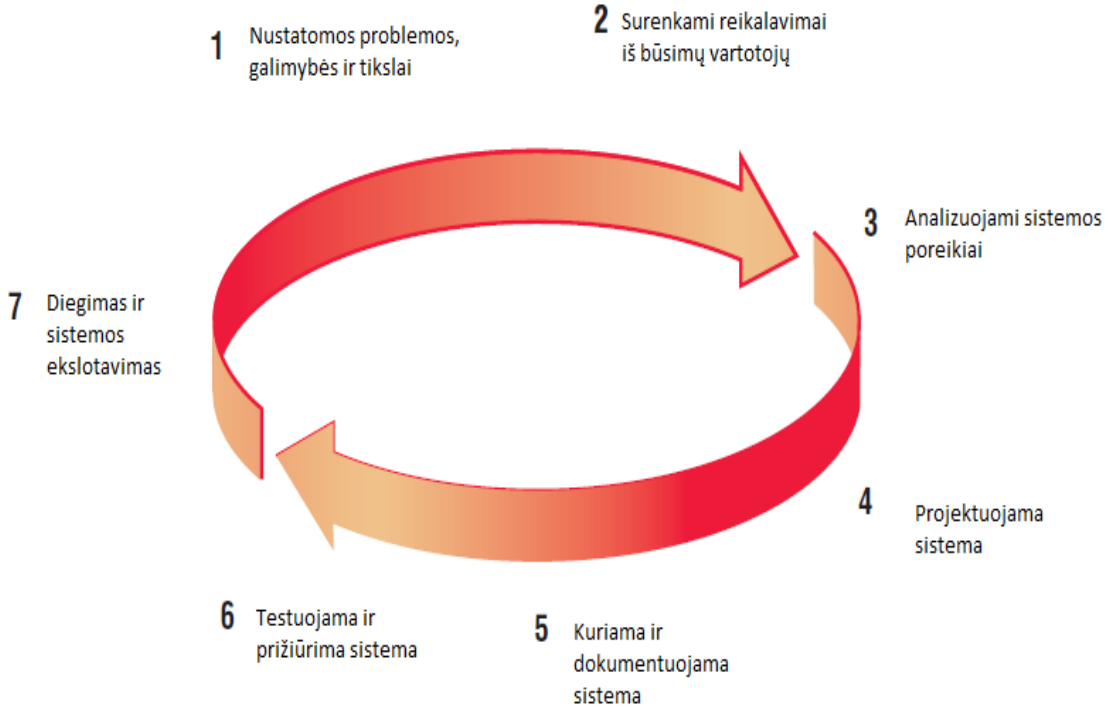
Problema: KTU Informacijos sistemų katedroje turimas VT grindžiamas reikalavimų specifikavimo metodas nėra susietas su jokiu sistemų projektavimo metodu. Todėl šiuo metu neapibrėžta, kaip pagal minėtą metodą specifikuoti reikalavimai turėtų būti panaudojami sistemų projektavimo metu.

2.3.Tradicinių informacinės sistemos kūrimo metodų apžvalga

2.3.1. Sistemų kūrimo gyvavimo ciklo metodas

Tradicinis informacinių sistemų kūrimo metodas, vadinamas sistemų kūrimo gyvavimo ciklu (angl. *SDLC – systems development life cycle*) yra laipsniškas etapų perėjimas nuo sistemos analizės, projektavimo iki kūrimo ir diegimo. Metodas užtikrina kuriamos informacijos organizuotumą. Išskiriami septyni sistemų kūrimo etapai [10], pavaizduoti 1.1 paveiksle. Nors etapai išskirti, tačiau retai kada pavyksta

juos atlikti nuosekliai, todėl dažnai kelios veiklos gali būti vykdomos lygiagrečiai ar kartojamos.



1.1 pav. Sistemų kūrimo gyvavimo ciklo metodo etapų schema [10]

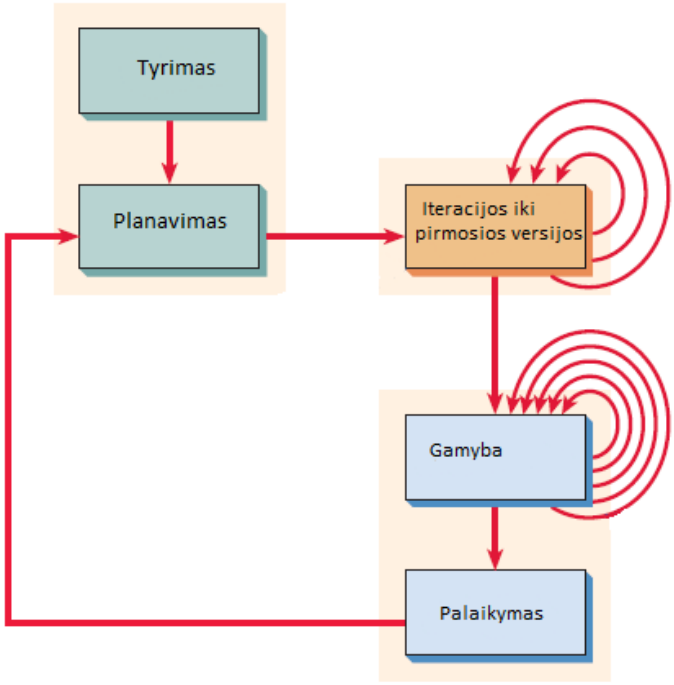
1. Nustatmos problemos, galimybės ir tikslai. Pirmame sistemų kurimo gyvavimo ciklo etape tiksliai identifikuojamos problemos, galimybės ir nusakomi tikslai. Šis etapas yra kritinis viso projekto sėkmei. Analitikai tiria rinką, nustato galimybės, kuriomis kuriama sistema pagerins ir pagreitins kompiuterizuojamą procesą. Asmenų grupės, dalyvaujančios etape: vartotojai, analitikai ir projektą koordinuojantys sistemų vadybininkai. Veiklos – apklausos, sukauptų žinių apibendrinimas, suformuotas projekto grafikas, dokumentuojami gauti rezultatai.
2. Surenkami reikalavimai iš būsimų vartotojų. Analitikas, naudodamasis įvairiais įrankiais, fiksuoja vartotojų darbo pobūdį, kokius dokumentus jie naudoja.
3. Analizuojami sistemos poreikiai. Nustatomos įvestys, procesai ir rezultatai verslo funkcijoms. Braižomos duomenų srautų, veiklos, sekų diagramos. Nustatomi struktūrizuoti teiginiai, apibrėžiantys sąlygas, veiksmus ir

veiksmų taisykles, kurie atvaizduojami įverčių lentelėmis ir įverčių medžiais.

4. Projektuojama sistema. Realizuojamas sistemos loginis modelis, projektuojamas duomenų bazės modelis. Realizuojamos UML diagramos, kuriomis remiantis bus kuriama sistema.
5. Kuriama ir dokumentuojama sistema. Analitikas parenga tikslią kuriamos programinės įrangos dokumentaciją. Programuotojai realizuoja sistemą, užtikrina kokybišką ir teisingą sistemos veikimą.
6. Testuojama ir prižiūrima sistema. Sistema testuojama remiantis duomenų šablonais. Tikrinamas gautų reazultatų tikslumas. Aptikus klaidas sistema yra koreguojama, užtikrinamas korektiškas jos veikimas.
7. Diegimas ir sistemos eksplotavimas. Šiame etape sistema diegamia tiesioginiams jos vartotojams, prireikus vartotojai apmokomi naudotis su naująja sistema. Jei prieš tai buvo sena sistema, analitikas paruošia migracijos planą.

2.3.2. Lanksčiųjų metodų apžvalga

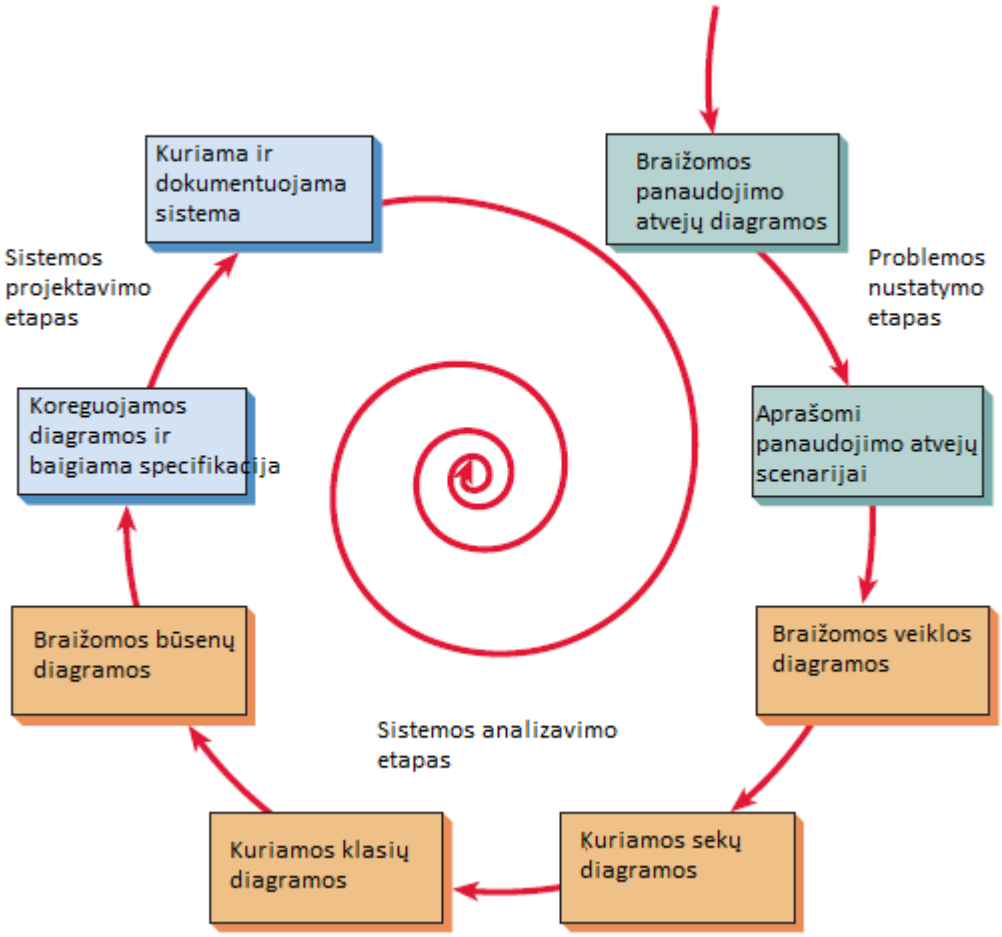
Lankstieji (angl. *Agile*) metodai orientuoti į darbo našumą, projekto lankstumą ir prisitaikymą prie besikeičiančių veiklos reikalavimų. Lanksčiųjų metodų manifestas [13] buvo pasirašytas 2001 metais, tačiau šie metodai egzistavo jau nuo 1990. Labiausiai paplitę metodai: XP, Scrum, adaptyvus kūrimas. Juose naudojamas iteracinis programinės įrangos kūrimo mechanizmas, kuris reglamentuoja projektavimo, programavimo ir testavimo pakartotines iteracijas. Metodas koncentruojasi į spartesnį veikiančio produkto pateikimą, nei dokumentacijos pilnumo ir nuoseklumo užtikrinimą. 1.2 paveiksle pavaizduoti lanksčiųjų metodų etapai.



1.2 pav. Lanksčiųjų metodų etapų schema [10]

2.3.3. Objektiškai orientuotų sistemų analizės ir projektavimo metodas

Objektiškai orientuotų sistemų kūrimo metodas akcentuoja greitai besikeičiančias dinamiškas verslo aplinkas, siekiant užtikrinti greitą ir patikimą programinės įrangos kūrimo procesą [10]. Šis metodas naudoja UML modeliavimo standartą. Metodo etapai pavaizduoti 1.3 paveiksle. Objektiškai orientuoto kūrimo metodo etapai turi panašumų su sistemų kūrimo gyvavimo ciklo metodu, kadangi abu šie metodai reikalauja tikslaus ir išsamaus modeliavimo bei dokumentavimo, tačiau trunka ilgesnį laiką, nei naudojantis lanksčiuoju metodu.



1.3 pav. Objektiškai orientuotos informacinės sistemos kūrimo metodikos schema [10]

UML projektavimo etapai, kurie naudojami objektiniame informacinių sistemų kūrime:

1. Problemos nustatymo etape identifikuojami aktoriai ir jų atliekamos funkcijos kuriamoje sistemoje, braižomos panaudojimo atvejų diagramos, kuriose ryšiais susiejami aktoriai su jiems priskiriamomis vykdyti funkcijomis. Kiekvienas panaudojimo atvejis aprašomas detaliau panaudojimo atvejo scenarijumi.
2. Sistemos analizavimo etapas prasideda braižant veiklos diagramas, kurios apibūdina pagrindines veiklas panaudojimo atvejų modelyje. Viena ar daugiau sekų diagrama iliustruojamas kiekvienas panaudojimo atvejis.
3. Braižomos klasių diagramos. Panaudojimo atvejų diagramoje naudojami daiktavardžiai gali būti grupuojami į klases.

4. Tolimesnis žingsnis sistemos analizės procese yra braižyti būsenų mašinų diagramas, kurios gaudžiai siejasi su klasių diagramomis. Mašinų diagramos padeda suprasti sudėtingus procesus, kurie negali pilnai būti atskleisti naudojant sekų diagramas.
5. Sistemos projektavimo etape peržvelgiamos sukurtos diagramos, prireikus jos modifikuojamos. Sukurtos diagramos panaudojamos klasių, jų atributų ir operacijų kūrimui. Šiame etape analitikas detaliai aprašo kiekvieną klasę, atributą, operaciją. Specifikuojami įvesties ir išvesties reikalavimai.
6. Dokumentuojama ir kuriama sistema. Kuo detalesnė specifikacija ir išbaigtesni modeliai, tuo greičiau vyks sistemos kūrimas.

2.3.4. Informacinių sistemų kūrimo metodų palyginimas

Visi trys paminėti sistemų kūrimo metodai turi daug bendro tarpusavyje. Juose analizuojama organizacija, planuojamas darbų grafiko kalendorius, nustatomi reikalingi ištekliai, ruošiamas projekto planas. Renkama sistemos kūrimui reikalinga informacija, kuri gaunama apklausiant organizacijos narius. Kriterijai, pagal kuriuos reikia pasirinkti informacinės sistemos kūrimo metodą pateikti 1.1 lentelėje.

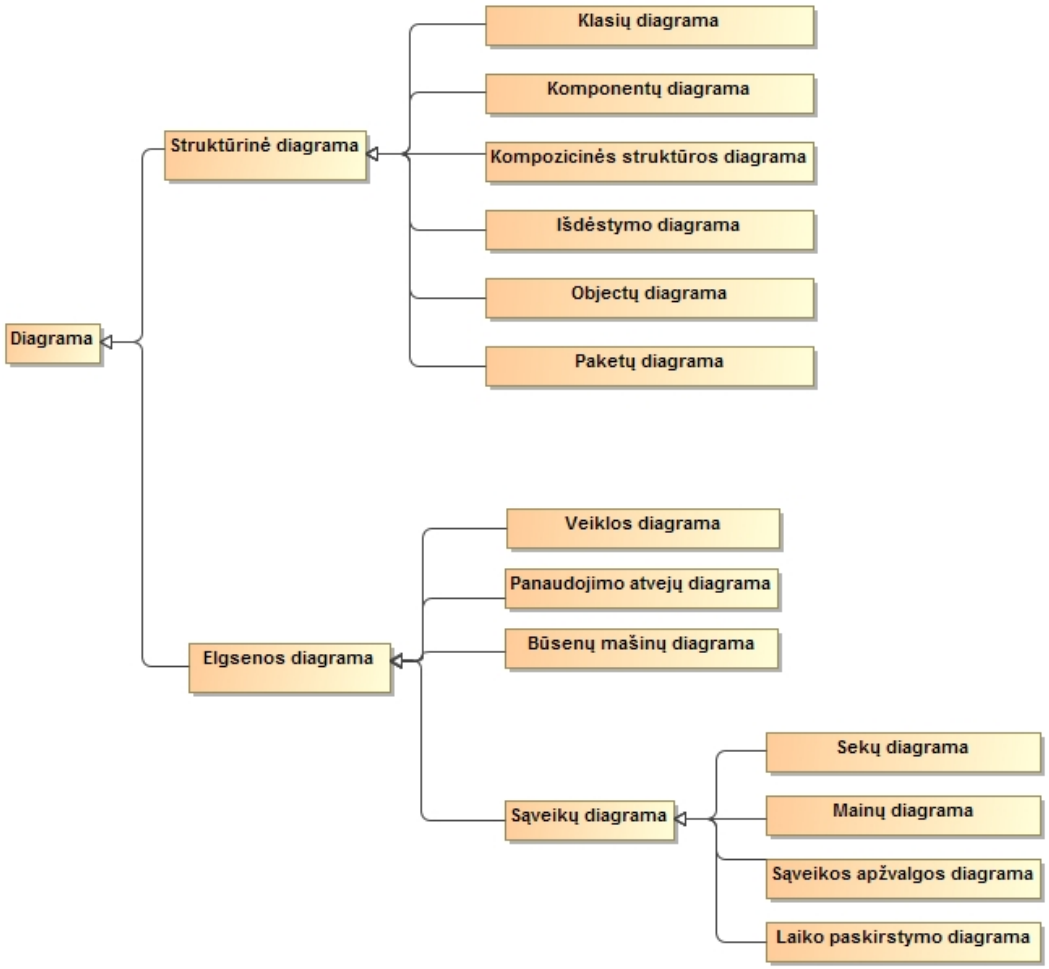
1.1 lentelė. Informacinės sistemos kūrimo metodo pasirinkimo kriterijai

Metodas	Kada vartoti
Tradicinis informacinių sistemų kūrimo metodas	Tobulinant sistemą, kuri buvo kuriama ir dokumentuojama naudojant šį metodą.
	Kai svarbu dokumentuoti kiekvieną sistemos kūrimo etapą.
	Projekto valdymas yra saugesnis ir patikimesnis naudojant SDLC.
	Kai pakankamai resursų ir laiko užbaigti pilną tradicinio sistemų kūrimo metodo ciklą.
	Didelę įtaką turi bendravimas su užsakovais ir išsiaiškinimas, kaip sistema turės funkcionuoti.
Lankstieji metodai	Kai yra patyręs lanksčiųjų metodų vadovas organizacijoje.
	Informacinės sistemos turi būti kuriamos greitai, priklausomai nuo dinamiškai besikeičiančių reikalavimų.
	Sistema nefunkcionuoja ir nėra laiko aiškintis kuriame etape buvo suklysta.
	Kai užsakovas yra patenkintas su nuolatiniais sistemos tobulinimais.

Metodas	Kada vartoti
Objektiškai orientuotas metodas	Modeliuojama sritis klasifikuojasi į klases.
	Kai organizacija palaiko UML modeliavimą.
	Sistemos vystomos tolygiai - vienas posistemis vienu laiko momentu.
	Kai galimas likutinės programinės įrangos panaudojimas.
	Kai yra galimybė sudėtingiausias problemas spręsti iškart.

2.4. UML modeliavimo kalbos apžvalga

Unifikuota modeliavimo kalba (UML - angl. *Unified modeling language*) yra metamodelis grafinių notacijų, kurios naudojamos modeliuoti programinę įrangą, ypač realizuoti objektiniu projektavimo metodu. UML yra atviras standartas, kontroliuojamas atviro konsorciumo įmonių – OMG (angl. *Object Managment Group*). UML kalba glaudžiai susijusi su MDA (angl. *Model Driven Architecture*) - programinės įrangos kūrimo metodu, kuris generuoja programinį kodą iš modelių. UML 2 sudaryta iš 13 diagramų, kurių tipai aprašyti 1.2 lentelėje ir klasifikacija pateikta 1.4 paveiksle.



1.4 pav. UML 2 hierarchijos diagrama [9]

Išskiriamos trys pagrindinės UML diagramų kategorijos:

- Struktūrinės diagramos (angl. *structure diagrams*) – skirtos modeliuoti sistemos elementus, kurie yra nepriklausomi nuo laiko.
- Elgsenos diagramos (angl. *behavior diagrams*) – skirtos modeliuoti sistemos funkcijas.
- Sąveikos diagramos (angl. *interaction diagrams*) – elgsenos diagramų pogrupis, skirtas modeliuoti objektų sąveikoms.

1.2 lentelė. UML 2 diagramos ir jų panaudojimas

Diagrama	Panaudojimas	Atsiradimas
Veiklos (angl. *activity*)	Ši diagrama parodo proceso veiksmų seką. Naudojama analizuojant dalykinės srities procesus ir detalizuoti panaudojimo atvejų funkcionalumą.	UML 1
Klasių (angl. *class*)	Aprašoma sistemos struktūra aibėmis objektų, turinčių tuos pačius atributus, operacijas ir ryšius.	UML 1
Mainų (angl. *communication*)	Pranešimais atvaizduoja sąveikas tarp objektų ar jų dalių. Ši diagrama apjungia klasių, sekų ir panaudojimo atvejų diamgramų informaciją į statinę ir dinaminę sistemos elgseną.	UML 1
Komponentų (angl. *component*)	Vaizduojamas sistemos išskaidymas į komponentus, jų tarpusavio priklausomybes.	UML 1
Kompozicinės struktūros (angl. *composite structure*)	Vaizduojama vidinė klasių struktūra, struktūros suteiktas bendradarbiavimo galimybes.	UML 2
Išdėstymo (angl. *deplayment*)	Modeliuojamas fizinis sistemos elementų išdėstymas.	UML 1
Sąveikos apžvalgos (angl. *interaction overview*)	Veiklų diagramų tipas, vaizduojantis viršūnes, kurios atitinka sąveikos diagramas.	UML 2
Objektų (angl. *object*)	Vaizduoja dalinį arba visą modeliuojamos sistemos vaizdą konkrečiu momentu.	Neoficialiai UML 1
Paketų (angl. *package*)	Modeliuojamas sistemos suskaidymas į atskiras logines grupes. Pagrindinis dėmesys koncentruojamas į priklausomybes tarp loginių grupių.	Neoficialiai UML 1
Sekų (angl. *sequence*)	Skirta vaizduoti pranešimų sekoms tarp objektų. Sekos nurodo objektų gyvavimo trukmę.	UML 1
Būsenų mašinų (angl. *state machine*)	Aprašo galimą klasės, panaudojimo atvejo, posistemio ar kito modelio elemento elgseną. Būsenų mašiną sudaro įvykiai, objekto reakcijos į įvykius ir būsenų sekos, kurias įgyja objektas reakcijos į įvykius išdavoje.	UML 1
Laikinio pasiskirstymo (angl. *timing*)	Sąveikos diagramų tipas, kuriuo apibrėžiami modelyje esantys laikini apribojimai.	UML 2
Panaudojimo atvejų (angl. *use case*)	Vaizduojami funkciniai sistemos reikalavimai, išoriniai aktoriai, jų elgsena ir sistemos funkcijos.	UML 1

2.5. VT koncepcijos apžvalga

2.5.1. GUIDE projekto uždaviniai ir veiklos taisyklių kategorijos

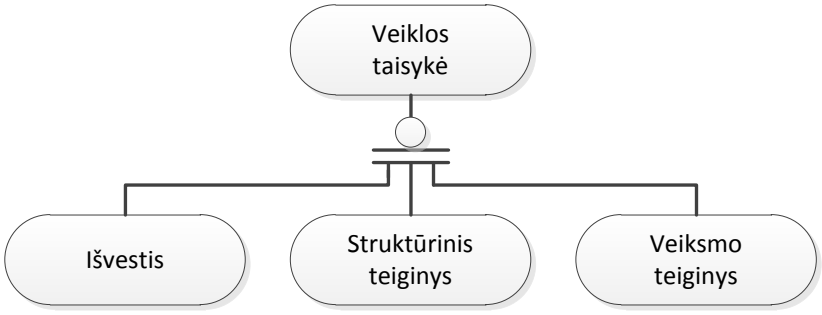
GUIDE veiklos taisyklių projektas skirtas formalizuoti koncepcijai, kuri identifikuoja ir struktūrizuoja taisykles, kurios apibrėžia įmonės struktūrą ir operacijas. Šis projektas standartizavo įmonės funkcijų ir struktūros užrašymo metodą veiklos taisyklių pavidalu. Taip taisyklės buvo fiksuojamos ir dokumentuojamos suprantamesniu ir kompetentingesniu būdu [11].

Išskiriami keturi GUIDE projekte iškelti uždaviniai:

- Apibrėžti ir apibūdinti veiklos taisykles, susijusias sąvokas, nustatant kas yra ir kas nėra veiklos taisyklė.
- Apibrėžti konceptualų veiklos taisyklių modelį, siekiant išreikšti kas yra veiklos taisyklė ir kaip ji atspindima informacinėse sistemose.
- Užtikrinti atvirkštinę programinės įrangos inžineriją – išgauti veiklos taisyklės iš egzistuojančių sistemų.
- Apibrėžti naujų sistemų kūrimą remiantis veiklos taisyklėmis.

Remiantis GUIDE, veiklos taisyklės išskiriamos į kategorijas, pateiktas 1.5 pav.:

- struktūrinis teiginys (angl. *structural assertion*) – apibrėžta mintis arba fakto konstatavimas, išreiškiantis tam tikrą organizacijos struktūros aspektą.
- veiksmo teiginys (angl. *action assertion*) – apribojimą ar sąlygą reiškiantis sakinys, kuris nustato arba valdo organizacijos veiksmus.
- išvestis (angl. *derivation*) – sakinys, išreiškiantis žinias, kurios savo ruožtu išvedamos iš kitų veiklos žinių.



1.5 pav. Veiklos taisyklių kategorijos remiantis GUIDE [11]

2.5.2. Semantikos veiklos žodyno ir veiklos taisyklių specifikacijos semantika

Semantikos veiklos žodyno ir veiklos taisyklių specifikacija (SBVR - angl. *Semantics of Business Vocabulary and Business Rules*) yra veiklos taisyklių apibrėžimo standartas, orientuotas į semantinę natūralios kalbos struktūrą [6]. Tai priemonė aprašyti semantines formuluotes, t.y veiklos taisyklės nepaverčiamos į formalią kalbą, o išlaikoma konkreti sakinių struktūra, leidžianti formalizuoti VT. Verslo žodynas apibrėžiamas, kaip vieta, kur sukaupti specializuoti terminai ir koncepcijų apibrėžimai, kuriuos naudoja analizuojama įmonė.

Taigi SBVR taikomas dvejopai:

- SBVR taikomas veiklos taisyklėms ir verslo žodynams. Kiti veiklos modelių aspektai taip pat turi būti kuriami, įskaitant ir verslo procesus bei organizacijos struktūrą, bet jie turi būti apdorojami naudojant kitas OMG iniciatyvas.
- Veiklos modeliams, įskaitant ir tuos modelius, kuriuos palaiko SBVR, tam, kad nusakytų veiklas, o ne informacines sistemas, kurios tas veiklas palaiko.

2.5.3. Produkcinių veiklos taisyklių apžvalga

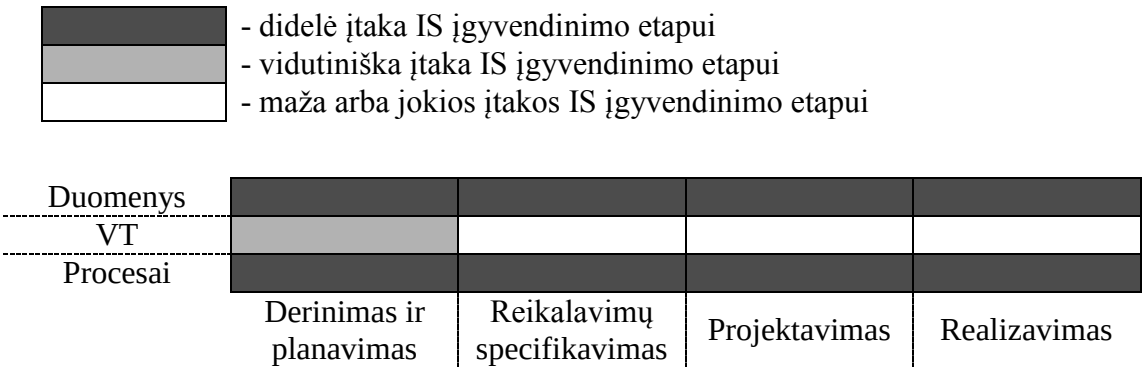
Produkcinių veiklos taisyklių vaizdavimas (PRR angl. *Production Rule Representation*) atitinka daugelį reikalavimų, keliamų veiklos taisyklėms, programinės įrangos sistemoms, OMG standartams ir kitiems taisyklių standartams. Jis suteikia galimybę taikyti VT komponentus kuriant informacines sistemas. Standartas palaikomas UML, aprašomas loginiais teiginiais [7].

- Suteikia veiklos taisyklių atvaizdavimo standartą, kuris yra suderinamas su veiklos taisyklių variklio gamintojų veiklos taisyklių apibrėžimais. Taigi jis gali būti naudojamas VT pasikeitimui tarp veiklos taisyklių modeliavimo įrankių.
- Suteikia veiklos taisyklių atvaizdavimo standartą, kuris lengvai susisieja su veiklos taisyklėmis.

- Suteikia pavyzdžius, kaip UML gali būti naudojamas veiklos taisyklių palaikymui standartizuotu ir naudingu būdu.
- Suteikia standartinį veiklos taisyklių atvaizdavimą, kuris gali būti naudojamas kaip pagrindas kitiems taisyklių valymo variantams, tokiems kaip: W3C Rule Interchange Format ir RuleML.

2.6.Veiklos taisyklių įtaka tradiciniams informacinių sistemų projektavimo metodams

Tradicinė informacinė sistema gali būti traktuojama kaip dvisluoksnė, t.y. susidedanti iš duomenų valdymo lygmens ir vartotojo arba atvaizdavimo lygmens. Veiklos taisyklių tokioje IS galima rasti kiekviename jos lygmenyje, tačiau dauguma jų yra „paslėptos“ duomenų, procesų ir kituose modeliuose, o realizavus sistemą – programiniame kode. Taigi dauguma taisyklių realizuotos daugiau nei vienoje vietoje, todėl jos gana žymiai atitolusios nuo pradinio pavidalo [1]. Vis dėlto pagrindinis dviejų lygmenų IS trūkumas – taisykles sunku aptikti ir pakeisti. 1.6 pav. pavaizduota veiklos taisyklių įtaka informacinių sistemų kūrimo etapuose.

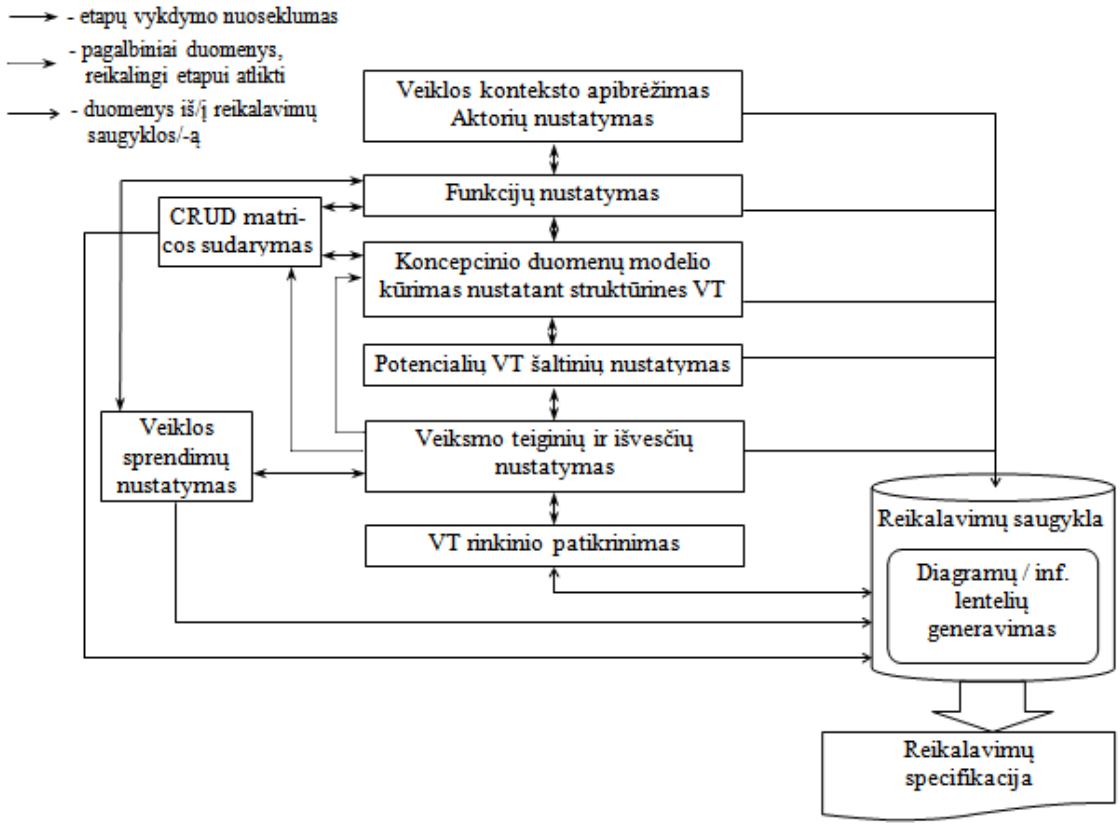


1.6 pav. Veiklos taisyklių įtaka IS įgyvendinimo etapuose tradiciniame modelyje [3]

2.7.VT grindžiamos reikalavimų specifikacijos panaudojimo projektuojant sistemas galimybės

Tiriamo reikalavimų specifikavimo proceso bendroji schema pateikta 1.7 paveiksle. Reikalavimų specifikavimo procesas pradedamas veiklos konteksto apibrėžimu, kur identifikuojami asmenys, kurie yra potencialūs kuriamos IS vartotojai arba tiesiog gali būti naudingi renkant reikalavimus kuriamai IS. Vėliau yra nustatomos funkcijos, veiklos sprendimai, struktūrinės ir nestruktūrinės VT ir jų šaltiniai, o rezultatai saugomi reikalavimų saugykloje. Nestruktūrinių VT nustatymas atliekamas analizuojant sprendimus, kurie turi būti priimti vykdant konkrečias funkcijas.

Svarbi sukurto metodo savybė yra CRUD matricos sudarymas. Tiksli CRUD matricos struktūra gali kisti priklausomai nuo analitiko poreikių ar taikomo metodo ypatumų [4]. Atsižvelgiant į pristatomo metodo savybes, CRUD matricoje siūloma fiksuoti CRUD ryšius tarp funkcijų ir svarbių terminų.

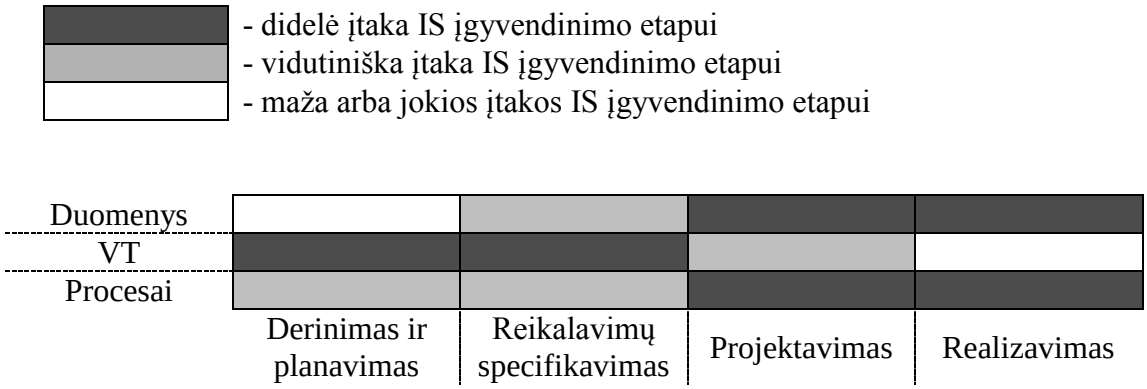


1.7 pav. Bendroji reikalavimų specifikavimo proceso schema [12]

Siekiamame sprendime veiklos taisyklės tarnautų kaip pagrindinis informacijos šaltinis informacinių sistemų derinimo, planavimo, reikalavimų specifikavimo, o projektavimo etape jos būtų transformuojamos į UML diagramas. Veiklos taisyklių įtaka konkretiems IS kūrimo etapams išauga, tai iliustruota 1.8 paveiksle. Tai leidžia greitai ir efektyviai svarstyti kuriamos IS reikalavimus ir pakeitimus pirmose dviejose stadijose - tiek projekto kūrėjams, tiek užsakovui. Užsakovo dalyvavimas projektavimo ir realizavimo etapuose yra ribojamas, tačiau pakeitimai įmanomi.

VT paremto IS kūrimo modelio privalumai:

- Projektas gali būti įgyvendintas naudojant stabilius ir patikimus įrankius, be padidėjusio investavimo į darbuotojų skaičių ir naują programinę įrangą.
- Įgyvendinimas nereikalauja sudėtingų sprendimų (VT saugykla ir administravimo įranga).
- Projektavimo etapas tampa našesnis, kadangi reikalavimai gerai išgryninami ir suformuluojami dviem dimensijomis (duomenims ir procesams).



1.8 pav. Veiklos taisyklių įtaka IS įgyvendinimo etapuose siekiamame modelyje [3]

2.8.Siekiamo sprendimo vartotojų analizė

2.8.1. Vartotojų aibė, tipai ir savybės

Vartotojų aibę sudaro informacinių sistemų kūrėjai ir jos užsakovai. Analitikas surenka reikalavimus konkrečiai sistemai, juos suformuluoja ir įveda į veiklos taisyklių saugyklą. Užsakovų veiklos ekspertas gali paskelbti veiklos taisykles negaliojančiomis arba pridėti naujų veiklos taisyklių.

2.8.2. Vartotojų tikslai ir problemos

Naudojant veiklos taisyklių saugyklą yra pagerinama komunikacija tarp verslo atstovų ir analitikų, kadangi visos VT užrašomos vienu pavidalu, suprantamu abiems pusėms. Pakeitimų įvedimas supaprastėja, nes užsakovai gali redaguoti reikalavimų saugyklą. Taip sprendžiamas komunikavimo problemos tarp abiejų suinteresuotų pusių.

2.9.Analizės apibendrinimas

Analizės metu buvo apibendrinti sistemų kūrimo gyvavimo ciklo, lanksčiojo, objektiškai orientuoto informacinių sistemų projektavimo ir kūrimo metodai. Aprašytos UML kalbą sudarančios diagramos. Aptartas GUIDE projektas, produkcinių veiklos taisyklių vaizdavimas, aprašyta semantikos veiklos žodyno ir veiklos taisyklių specifikacija. Suformuluota veiklos taisyklių įtaka tradiciniuose projektavimo metoduose. Ištirtas veiklos taisyklėmis grindžiamas reikalavimų metodas ir suformuluotos gairės siekiamam sprendimui.

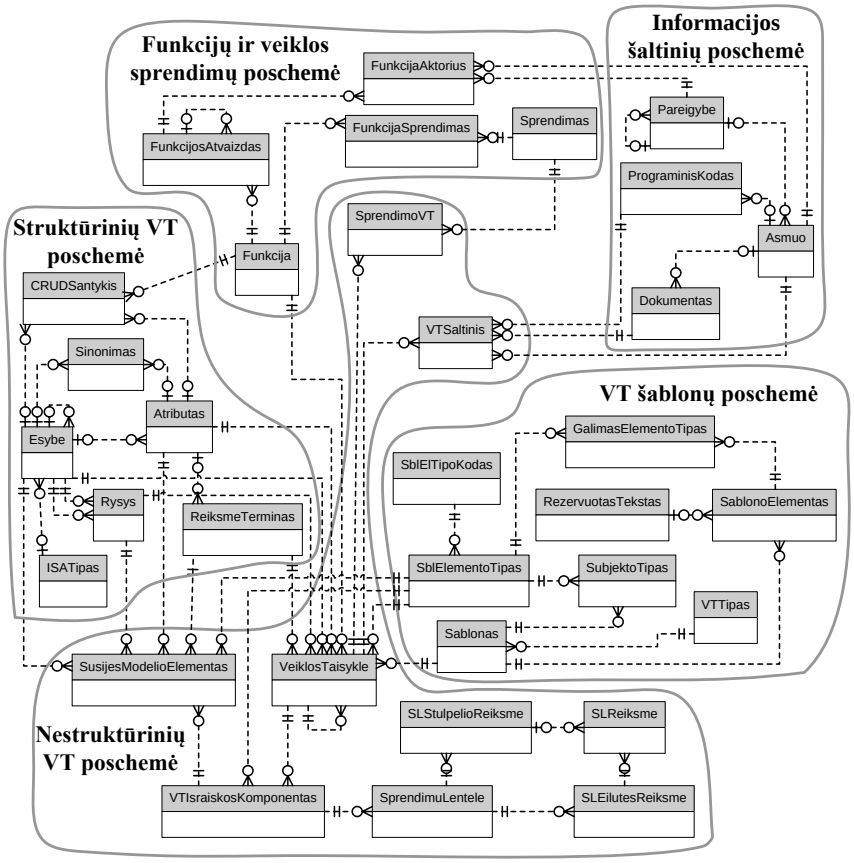
Atsižvelgiant į analizės rezultatus, priimtas sprendimas veiklos taisyklėmis grindžiamą reikalavimų specifikavimo metodą pritaikyti objektiškai orientuotam projektavimui, kadangi jis užtikrintų plačiausią veiklos taisyklių specifikacijos taikymą projektavimo etapui.

3. Veiklos taisyklėmis paremtos specifikacijos transformavimo į UML modelius metodas

3.1.Apibendrinta metodikos schema

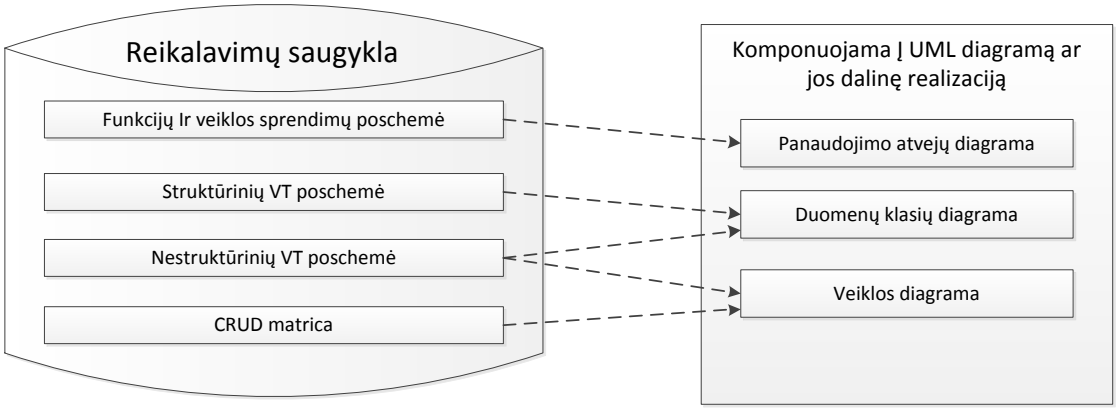
Pagal VT grindžiamą specifikavimo metodą saugykloje išsaugomi kuriamos sistemos reikalavimai. Reikalavimų saugyklos žinių bazės modelis susideda iš penkių poschemių, atvaizduotų 2.1 paveiksle:

1. Informacijos šaltinių poschemėje saugoma informacija apie asmenis, pareigybes, dokumentus ar programinio kodo elementus, kurie yra vienaip ar kitaip svarbūs specifikuojamos dalykinės srities kontekste ir/arba įtakoja nustatomus reikalavimus.
2. Funkcijų ir veiklos sprendimų poschemė;
3. Struktūrinių veiklos taisyklių (koncepcinio duomenų modelio) poschemė;
4. Veiklos taisyklių šablonų poschemė;
5. Nestruktūrinių veiklos taisyklių poschemė.



2.1 pav. Supaprastintas reikalavimų saugyklos loginės struktūros modelis [1]

Siūlomas metodas transformuoja poschemių atributus į UML diagramų elementus. Apibendrinta reikalavimų saugyklos transformacijos metodikos schema pavaizduota 2.2 paveiksle.

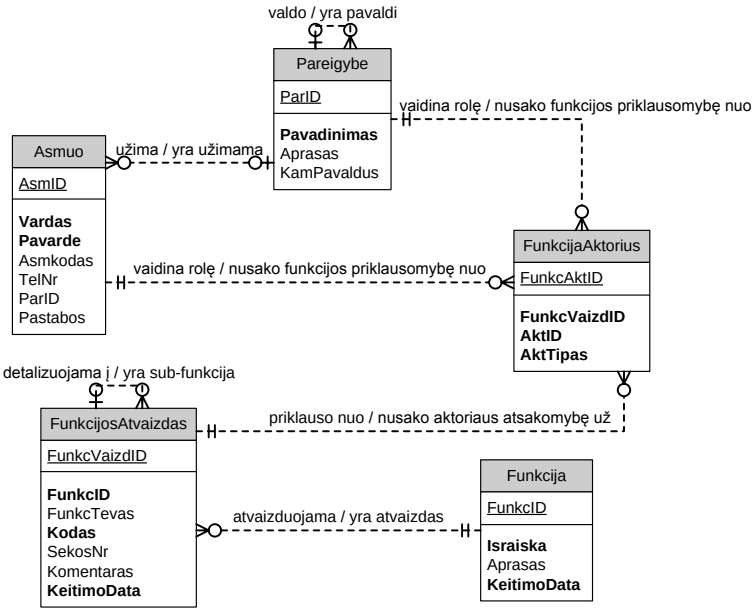


2.2 pav. Apibendrinta metodikos schema

3.2.Funkcijų (ir veiklos sprendimų) transformavimo į panaudojimo atvejų modelį proceso aprašas

Nagrinėjamame veiklos taisyklėmis paremtos reikalavimų specifikacijos metodo apraše funkcijos apibrėžtis skamba taip: funkcija – tai veikla, kurią organizacija atlieka arba turi atlikti ateityje tam, kad įgyvendintų savo tikslus. Sudarant funkcijų rinkinį išskiriamos pagrindinės funkcijos, kurios detalizuojamos kitomis funkcijomis. Funkcijos susiejamos su aktoriais, t.y asmenimis ar pareigybėmis atsakingomis už tam tikros funkcijos vykdymą. Funkcijos saugomos (žr. 2.3 pav.) loginiame saugyklos modelyje, o jų nustatymo rezultatas yra funkcijų hierarchijos modelis. Funkcijų nustatymo procesas baigiamas, kai sudaroma pilna funkcijų hierarchija, kurios reikalavimai:

- Kiekviena funkcija privalo būti skaidoma į dvi ar daugiau sub-funkcijų arba neskaidoma iš viso.
- Priimama, kad funkcijai priskirti aktoriai vienodai susiję ir su visomis šios funkcijos žemesnių hierarchijos lygių funkcijomis. Todėl aktoriai turi būti priskirti kaip įmanoma aukštesnio hierarchijos lygio funkcijoms.

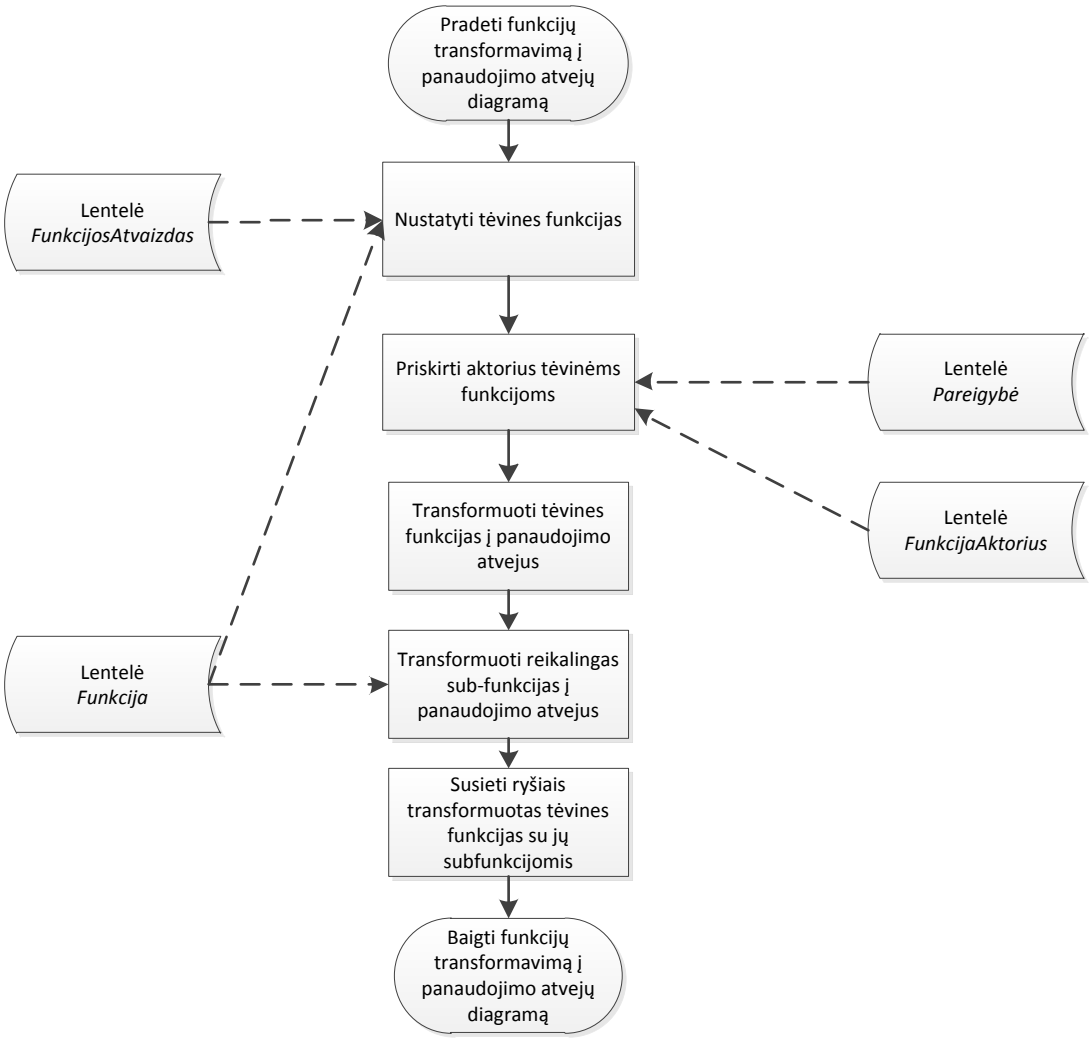


2.3 pav. Funkcijų saugyklos loginis modelis [1]

Panaudojimo atvejų modelis nusako sistemos kontekstą – su kokiais aktoriais sąveikauja sistema. Diagrama atvaizduoja panaudojimo atvejus, aktorius bei jų tarpusavio ryšius. Aktoriai su jiems priskiriamomis funkcijomis vaizduojami asociaijos ryšiu. Agregavimo (angl. *include*) ryšys nurodo, jog tėvo panaudojimo atvejis susideda iš vaiko panaudojimo atvejo. Vaiko panaudojimo atvejai būtini tam, kad būtų galima įvykdyti tėvo panaudojimo atvejį. Išplėtimo (angl. *extend*) ryšys jungia panaudojimo atvejus tada, kai tėvo panaudojimo atvejį reikia išplėsti papildoma elgsena, kuri įterpiama išplėtimo taške, jei išpildoma tam tikra sąlyga. Struktūrizuoti panaudojimo atvejai aprašo tik esminį elgesį ir nebūna nei per daug dideli, nei per daug detalūs. Kiekviena naujai nustatyta funkcija ar sub-funcija saugoma `Funkcija` ir `FunkcijosAtvaizdas` lentelėse, kur `Funkcija` lentelėje apibrėžiama funkcijos išraiška, o `FunkcijosAtvaizdas` lentelė nurodo hierarchines priklausomybes tarp funkcijų. Kadangi sistema gali tūrėti daug funkcijų ir jas detalizuojančių sub-funkcijų, tai analitikui reikia įvertinti kurios iš funkcijų bus vaizduojamos panaudojimo atvejų diagramoje, o kurios bus naudojamos sekų ir veiklos diagramose. Remiantis funkcijų hierarchijos sudarymo metodu aktoriai yra priskiriami tėvinėms funkcijoms, kur aktorių atitiks lentelės `Pareigybe` atributas `Pareigybe.Pavadinimas`. Šis atributas siejamas su auksčiausio lygio funkcijomis asociacijos ryšiu. Transformuoti funkcijų hierarchiją į panaudojimo

atvejų modelį reikia laikytis tokios bazinės veiksmų sekos, kuri atvaizduota 2.4 paveiksle:

1. Nustatyti tėvines funkcijas.
2. Priskirti aktorius tėvinėms funkcijoms.
3. Transformuoti tėvines funkcijas į panaudojimo atvejus.
4. Transformuoti reikalingas sub-funkcijas į panaudojimo atvejus.
5. Susieti ryšiais transformuotas tėvines funkcijas su jų sub-funkcijomis.

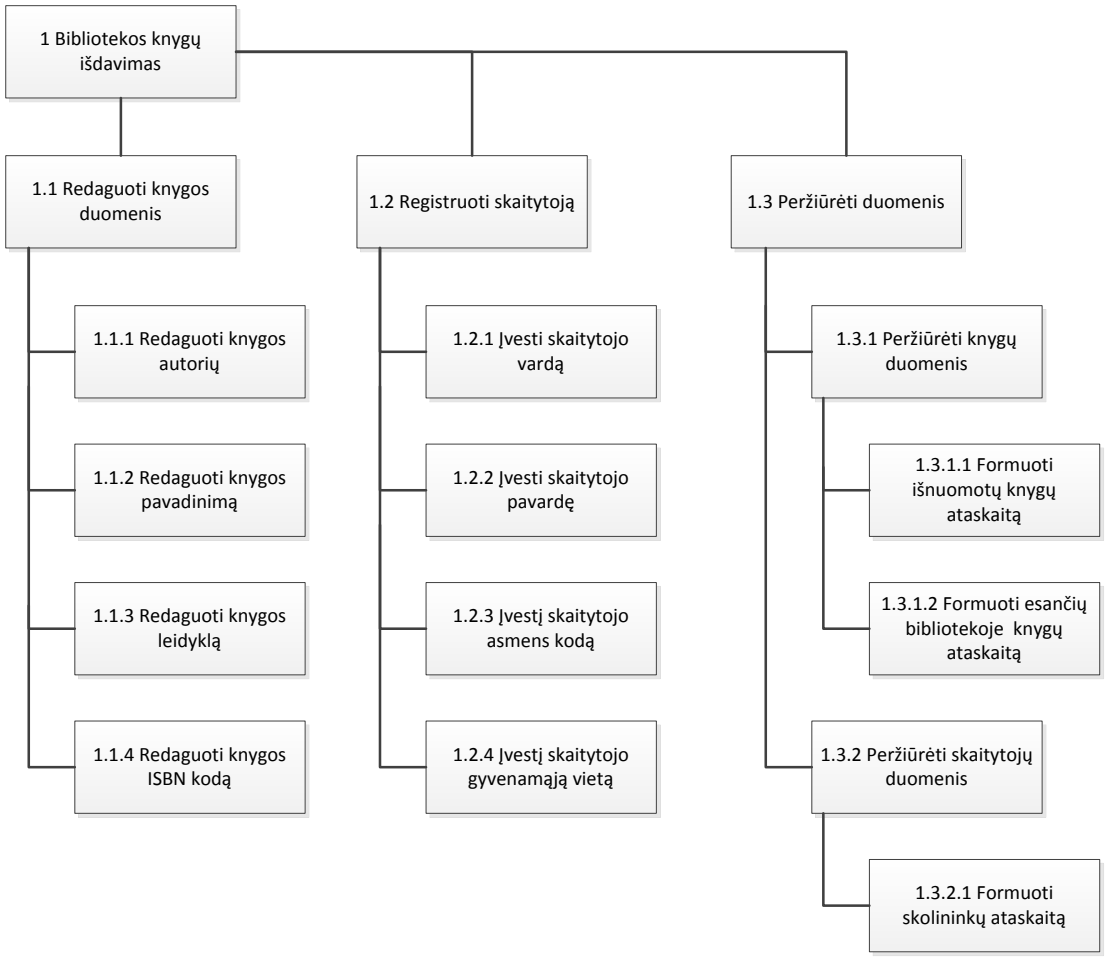


2.4 pav. Funkcijų hierarchijos transformavimo į panaudojimo atvejų modelį iliustruojanti veiklų diagrama

2.1 lentelė. Funkcijų hierarchijos saugyklos transformavimas į panaudojimo atvejų modelį

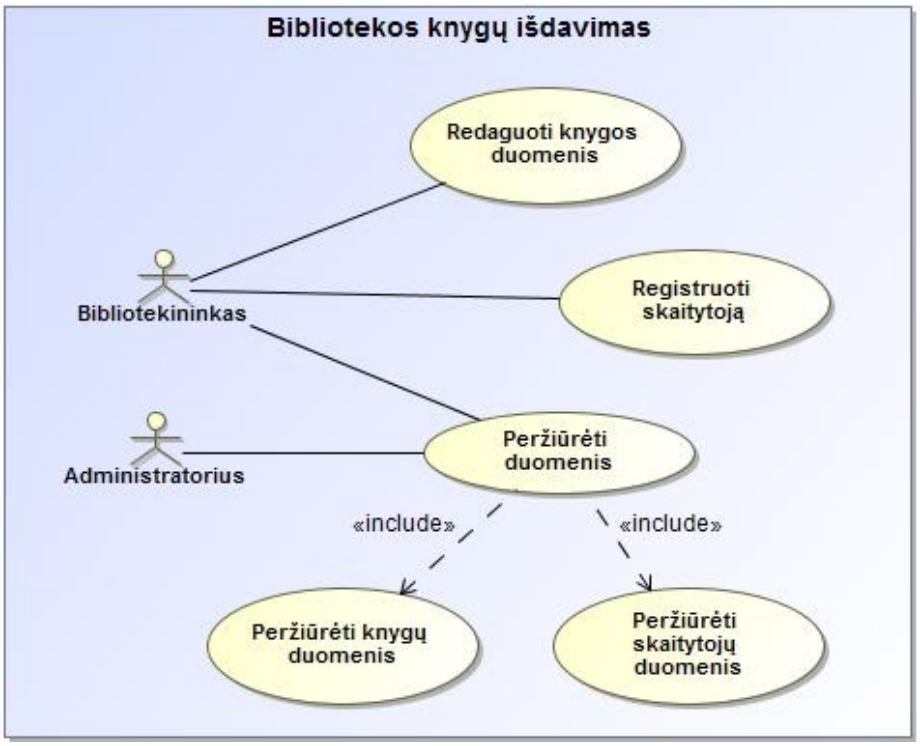
Reikalavimų objektas	Reikalavimų metamodelio elementas(-ai)	Gali būti transformuojama į UML objektą (-us)
Funkcija	*Funkcija.Israiska*	Panaudojimo atvejis
Funkciją atliekanti pareigybė	*Pareigybė.Pavadinimas*	Aktorius
Funkcijų atvaizdai	*FunkcijosAtvaizdas.FunkcVaizdID, FunkcijosAtvaizdas.FunkcTevas*	Ryšius tarp panaudojimo atvejų, ryšius tarp panaudojimo atvejų ir aktoriaus

Analitikas turi įvertinti funkcijų hierarchijoje naudojamą tėvinių funkcijų ir sub-funkcijų skaičių. Jei tėvinių funkcijų skaičius yra didelis, tai pridedamos sub-funkcijos panaudojimo atvejų modelį pavers griozdišką ir sunkiai skaitomą. Nagrinėjant 2.5 paveiksle suformuotą funkcijų hierarchiją priimame, jog sistemos pavadinimas bus „*Bibliotekos knygų išdavimas*“.



2.5 pav. Pavyzdinė VT grindžiamo reikalavimų metodo funkcijų hierarchijos diagrama

Šiuo atveju pareigybei bibliotekininkas yra priskirtos funkcijos: *„1.1 Redaguoti knygos duomenis“*, *„1.2 Registruoti skaitytoją“* ir *„1.3 Peržiūrėti duomenis“*,o pareigybei administratorius priskirta 1.3 funkcija. Tai funkcijos 1.1, 1.2 ir .1.3 bus laikomos tėvinėmis, o jas detalizuojančios funkcijos - sub-funkcijomis. 2.6 paveiksle pateikta panaudojimo atejų diagrama, kuri pagal suformuotą metodą buvo transformuota iš funkcijų hierarchijos. Visos tėvinės funkcijos buvo transformuotos į panaudojimo atvejus, o joms priskirtos pareigybės į aktorius. Sub-funkcijomis detalizuotas panaudojimo atvejis „*1.3 Peržiūrėti duomenis“*, jis agregavimo ryšiu sujungtas su funkcijomis *„1.3.1 Peržiūrėti knygų duomenis“* ir *„1.3.2 Peržiūrėti skaitytojų duomenis“*.

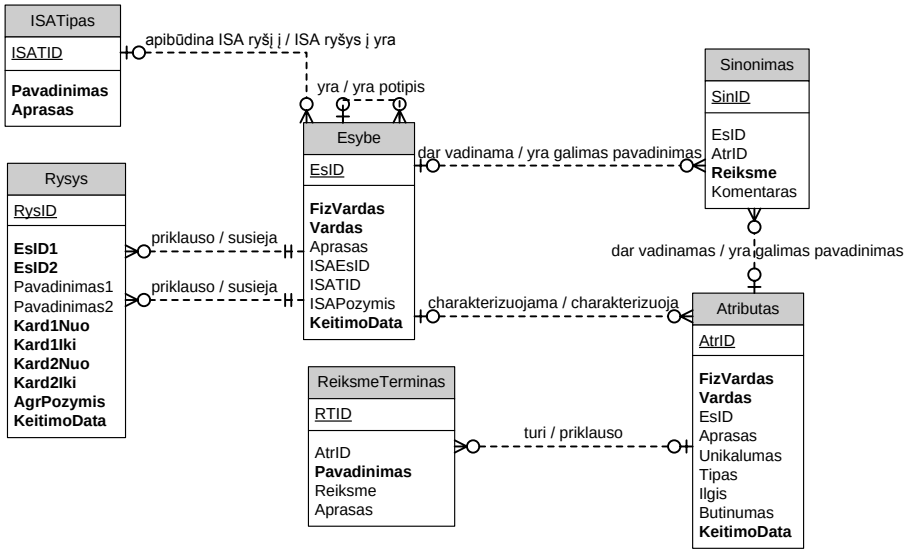


2.6 pav. Funkcijų hierarchijos transformacija pagal pasiūlytą metodą į panaudojimo atvejų diagramą

3.3.Struktūrinių VT (praplėsto koncepcinio duomenų modelio) transformavimo į duomenų klasių diagramą proceso aprašas

VT grindžiamame reikalavimų specifikavimo metode struktūrinėms veiklos taisyklėms fiksuoti naudojamas faktų modelis. Kitaip tariant terminai ir faktai fiksuojami kaip koncepcinio duomenų modelio – išplėstinės esybių ryšių (ERR angl.

Enhanced Entity-Relationship model) diagramos – elementai. Loginio saugyklos modelio fragmentas skirtas saugoti struktūrinėms veiklos taisyklėmis pavaizduotas 2.7 paveiksle.



2.7 pav. Saugyklos loginio modelio fragmentas struktūrinėms VT atvaizduoti [1]

ERR ir UML duomenų klasių diagramos turi nemažai bendro, abi jos aprašo aibes objektų, turinčių tuos pačius atributus, ryšius, tačiau ERR diagramos neturi operacijų sąrašo. Pagal veiklos taisyklėmis grindžiamos reikalavimų specifikacijos metodą fiksuojami elementai yra: terminai ir faktai. Terminus sudaro esybės, atributai, charakteristikos, o faktui – esybių sąryšiai, prototipų sąryšiai ir atributų sąryšiai. 2.2 lentelėje išskiriami loginio saugyklos modelio elementai, kurie gali būti transformuojami į klasių diagramą.

2.2 lentelė. Struktūrinių veiklos taisyklių transformavimas į duomenų klasių modelį

Reikalavimų objektas	**Reikalavimų metamodelio elementas(-ai)**	**Gali būti transformuojama į UML objektą (-us)**
Esybė	*Esybe.Vardas*	Klasės pavadinimas
Atributas	*Atributas.FizVardas*	Klasės atributas
Atributo požymiai	*Atributas.Unikalumas, Atributas.Tipas, Atributas.Ilgis*	Klasės atributo unikalumo požymį, tipą ir reikšmės ilgį
Ryšiai tarp esybių	*Ryšys.EsID1, Ryšys.ESID2, Ryšys.Kard1Nuo, Ryšys.Kard1Iki, Ryšys.Kard2Nuo, Ryšys.Kard2Iki*	Asociacijos ryšius tarp esybių

3.4.Nestruktūrinių VT transformavimo/pritaikymo proceso aprašas

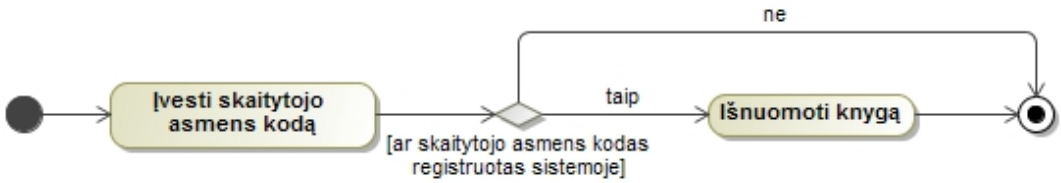
Nestruktūrinėms veiklos taisyklėms modeliuoti buvo pasirinkta bibliotekos dalykinė sritis. 1 priede pateikti modifikuoti *BRS RuleSpeak* natūraliosios kalbos šablonai [1], kuriuose suformuluotos nestruktūrinės veiklos taisyklės bibliotekos dalykiniai sričiai. Šiame skyriuje suformuluotos gairės kiekvienam taisyklių tipo panaudojimui UML kalboje.

3.4.1. Atmetimo, apribojimo taisyklių pritaikymas (transformavimas)

Atmetimo taisyklė (1.1 punktas) – loginio tipo nestruktūrinė VT, skirta užtikrinti duomenų teisingumui. Šablone naudodami subjekto tipai: terminas, faktas ar duomenų elementas. Šio tipo taisykles galime pateikti komentarais duomenų klasių diagramoje, kur taisyklės fiksuos atributo ar esybės galimas įgyti reikšmes, sąsaja su kitais atributais, ar loginę funkciją. Suformuluotos atmetimo taisyklės *„Skaitytojas neturi išsinuomoti knygos, jeigu delspinigių suma viršija 50 LT“* transformacija į duomenų klasių diagramos komentarą pateikta 2.8 paveiksle.

2.8 pav. Komentarų realizuota atmetimo taisyklė duomenų klasių diagramoje

Komponuodami vieną ar daugiau atmetimo taisyklių galime suformuoti veiklos diagramą, kur taisyklės sąlyga atitinka sprendimo tašką, o subjektas su faktu atitiks teigiamą ar neigiamą sprendimo taško atšakos veiklą. Turėdami suformuluotą atmetimo taisyklę *„Skaitytojo asmens kodas turi būti registruotas sistemoje, jeigu nori išsinuomoti knygą“* galime transformuoti į veiklos diagramos fragmentą, pateiktą 2.9 paveiksle.



2.9 pav. Atmetimo taisyklė transformuota į veiklos diagramą

3.4.2. Leidimo taisyklės pritaikymas

Leidimo taisyklė (1.2 punktas) – šio tipo nestruktūrinės VT taisyklės konstatuoja ar paaiškina kada yra leidžiama vykdyti veiklą. Tai bendros kategorijos taisyklė, turinti panašią šablono konstrukciją su atmetimo taisykle, tačiau subjektais be termino, fakto ir duomenų elemento gali būti kita taisyklė ir procesas.

Taisyklės panaudojimas analogiškas 1.1 punkto atmetimo taisyklei, kur taisyklė gali būti pritaikoma duomenų klasių modeliui komentaro pavidalu ar realizuojama veiklos diagrama.

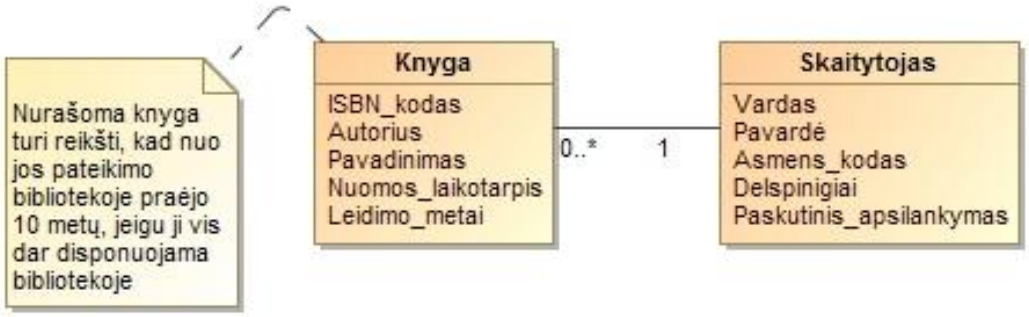
3.4.3. Skaičiavimo taisyklės pritaikymas

Skaičiavimo taisyklė (2.1 punktas) – ši VT pagal šablonus *b6* ir *b7* gali būti suformuluota tiek teiginiais, tiek formule. Taisyklė išreiškia teiginį ar aritmetinę formulę, nurodo, kaip turi būti apskaičiuojama tam tikra skaitmeninė reikšmė. Šios VT panaudojimas galimas objektinių reiškinių kalba (OCL angl. *Object Constraint Language*), kur taisyklės aprašomos formalizuota UML kalba. Kai VT subjektas yra duomenų elementas, tai šią taisyklę galime prijungti prie duomenų klasių modelio, kuri paaiškins atributo reikšmę ir skaičiavimą. Suformuluojame skaičiavimo taisyklę *„Delspinigiai = nuomos_laikotarpis -60 * 0.1 LT, jeigu nuomos_laikotarpis > 60 dienų“*. Šią taisyklę galime transformuoti į komentarą duomenų klasių diagramoje, kur atributo delspinigiai komentaras nurodys, kaip šis atributas yra apskaičiuojamas. Šios taisyklės panaudojimo iliustracija pateikta 2.10 paveiksle.

2.10 pav. Klasių diagramos komentaru realizuota skaičiavimo taisyklė

3.4.4. Išvedimo taisyklės pritaikymas

Išvedimo taisyklė (2.2 punktas) – naudojama norint pagal tam tikras aplinkybes padaryti išvadą. Šio tipo taisyklės subjektu dažniausiai bus duomenų esybė ar elementas. Plačiausias taisyklės panaudojimas duomenų klasių modelyje nurodant esybės ar atributo įgyjamas reikšmes realizuojant komentaru. Kitas panaudojimo būdas - įkomponuoti ar pateikti taisyklės fragmentą veiklų diagramoje, kur ši taisyklė dažniausiai atitiks sprendimo tašką. Suformuluotą išvedimo taisyklę *„Nurašoma knyga turi reikšti, kad nuo jos pateikimo bibliotekoje praėjo 10 metų, jeigu ji vis dar disponuojama bibliotekoje“* pateikiame esybės *Knyga* komentaru duomenų klasių modelyje 2.11 paveiksle.



2.11 pav. Išvedimo taisyklė transformuotą į komentarą duomenų klasių diagramoje

3.4.5. Išvados taisyklės pritaikymas

Išvados taisyklė (3.1.1 punktas) – tai taisykė, pagal kurią atsižvelgiant į tam tikras aplinkybes padaroma išvada. Subjektais taisyklėje bus esybė, atributas ar ryšys. Kadangi šios taisyklės subjektai apibūdina duomenų elementus, tai tikslingiausia ją

taikyti kaip komentarą duomenų klasių modelyje. Kitas panaudojimo metodas transformuoti į platesnio pobūdžio veiklos diagramą, kur apjungiamos panašaus tipo nestruktūrinės veiklos taisyklės.

3.4.6. Taisyklės jungiklio pritaikymas

Taisyklės jungiklis (3.1.2 punktas) – taisyklė, kuri įgalina ar išjungia kitą taisyklę tam tikromis konkrečiomis aplinkybėmis. Šios VT subjektas yra tam tikra taisyklė. Jungiklio taisyklės tipas plačiausiai naudojama nusakyti taisyklių išimtims. Apjungiant kelias taisykles galime suformuoti veiklos diagramą, kurioje atsispindės taisyklių veikimo logika. Turime suformuluotą atmetimo nestruktūrinę taisyklę *„Skaitytojas neturėtų išsinuomoti knygos, jeigu delspinigių suma viršija 10 litų“*. Tarkime biblioteka išduoda savo skaitytojams lojalaus skaitytojo korteles, su kuriomis taikomos lengvatos ir nuolaidos. Taigi pritaikydami taisyklės jungiklį suformuluotajame naują taisyklę *„Skaitytojas neturėtų išsinuomoti knygos, jeigu delspinigių suma viršija 10 litų, nebent pateikia lojalaus skaitytojo kortelę“*. Taisyklės jungiklis pavaizduotas 2.12 paveiksle.

2.12 pav. Taisyklės jungiklio transformavimas į veiklos diagramą

3.4.7. Proceso jungiklio pritaikymas

Proceso jungiklis (3.1.3 punktas) – taisyklė panašaus tipo kaip ir taisyklės jungiklis, tačiau naudojamas subjektas yra procesas, kur taisyklė tam tikromis

aplinkybėmis įgalina arba išjungią procesą. Proceso jungiklį tikslinga naudoti veiklos diagramose. Taisyklė dažniausiai transformuosis į keletą veiklų ir sprendimo tašką. Tą pačia veiklą iliustruojančias taisykles galime grupuoti ir atspindėti platesnę veiklą, atskleisti vykdymo logiką.

3.4.8. Duomenų jungiklio pritaikymas

Duomenų jungiklis (3.1.4 punktas) – naudojamas šalinti ar sukurti duomenis tam tikromis aplinkybėmis. Taisyklės subjektas yra duomenų elementas. Duomenų jungiklis gali būti panaudojamas duomenų klasių diagramoje komentaru, kai susiklosčius tam tikroms aplinkybėms atributui ar esybei turi būti taikomi tam tikri priskirti veiksmai. Taisyklės *„Vartotojo paskyra esybėje Skaitytojas turi būti ištrinta, jeigu nuo paskutinio apsilankymo praėjo daugiau nei 5 metai“* pritaikymas pavaizduotas 2.13 paveiksle.

2.13 pav. Duomenų jungiklio taisyklės pritaikymas duomenų klasių diagramoje

3.4.9. Reikšmės priskyrimo taisyklės pritaikymas

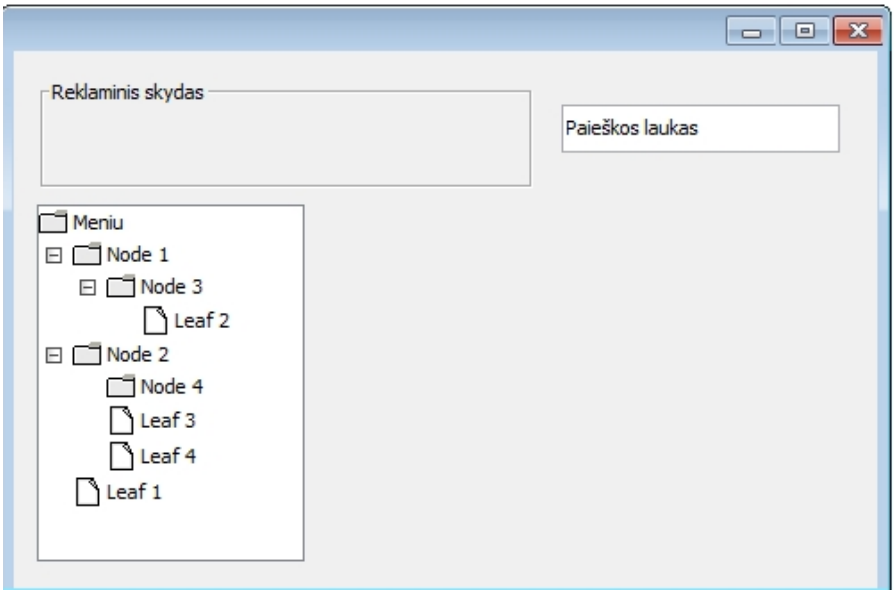
Reikšmės priskyrimo taisyklė (3.2.1 punktas) – skirta priskirti saugomam duomenų elementui tam tikrą reikšmę. Kadangi galimi subjektai yra terminas, faktas ir duomenų elementas, tai taisyklę tikslingiausia naudoti duomenų klasių diagramoje. 2.14 paveiksle pateiktas reikšmės priskyrimo taisyklę *„Skaitytojo esybės atributas Paskutinis_apsilankymas turi įgyti reikšmę dienos data, kai skaitytojas išsinuomoja ar grąžina knygą“* iliustruojantis pavyzdys.

2.14 pav. Reikšmės priskyrimo taisyklės pritaikymas duomenų klasių diagramoje

3.4.10. Pateikimo taisyklės pritaikymas

Pateikimo taisyklė (3.2.2 punktas) – nurodo duomenų pateikimo formą. Šio tipo taisyklė gali nurodyti atvaizdavimą ekrane, ataskaitoje ar kitose formose. Esant pakankamai pateikimo taisyklių galime suformuoti prototipinę vartotojo sąsają, naudodami vartotojo sąsajos diagramą. Diagramos pavyzdys su žemiau suformuluotomis taisyklėmis pateiktas 2.15 paveiksle. Veiklos diagrama galima formuoti navigavimo planą tarp sistemos langų. Pateikimo taisyklių pavyzdžiai:

- Reklaminis skydas turi būti pateiktas sistemos viršutinėje dalyje.
- Paieškos laukas turi būti pateiktas šalia reklaminio skydelio.
- Meniu punktai turi būti pateikti po reklaminiu skydeliu kairėje pusėje.



2.15 pav. Pateikimo taisyklių realizacija vartotojo sąsajos diagrama

3.4.11. Proceso trigerio taisyklės pritaikymas

Proceso trigerio taisyklė (3.3.1 punktas) – situacijų taisyklė, kuri automatiškai susiklosčius tam tikroms aplinkybėms iškviečia kitą taisyklę ar procedurą. Kai taisyklės subjektas yra procesas, ją galime įkomponuoti į platesnio masto veiklos diagramą, atspindinčią sistemos elgseną. Šio tipo taisyklė veiklos diagramoje atitiks sprendimo tašką ir iškvies kitą procesą. Kai taisyklės subjektas yra procedūra, proceso trigerio taisyklę galime komponuoti į sistemos duomenų klasių modelį komentaro pavidalu prie šią taisyklę vykdančios operacijos.

3.4.12. Taisyklės trigerio pritaikymas

Taisyklės trigeris (3.3.2 punktas) – taisyklės tipas, kai susidarius tam tikroms aplinkybėms yra automatiškai paleidžiama kita taisyklė. Kadangi taisyklės subjektas yra kita taisyklė, tai tikslinga taisyklę naudoti veiklos diagramose. Grupuojant panašią veiklą atliekančias funkcijas galime nubraižyti detalią veiklų diagramą.

3.5. CRUD matricos pritaikymo aprašas

CRUD matrica atskleidžia, kaip konkretūs terminai yra naudojami vykdant konkrečias funkcijas. Šiuo atveju terminai – tai koncepcinio duomenų modelio esybių ir atributų pavadinimai. Sisteminiai įvykiai neturėtų figūruoti VT išraiškose ir niekada negali būti VT subjektais. Pavyzdžiui, jeigu turime funkciją, kurios formuluotė nėra pakankamai informatyvi (pvz.: *„Nustatyti knygos vertę”*), kuri pateikta 2.3 lentelėje CRUD matricos pavidalu. Lentelė rodo, jog atliekant knygos įvertinimo funkciją bus nuskaitomos knygos išleidimo datos, nuomojimo pradžios datos ir naujos knygos kainos atributų reikšmės. Vertės reikšmė taip pat bus sukuriama ir/arba atnaujinama.

2.3 lentelė. CRUD matricos vienai funkcijai pavyzdys

Funkcija:	Nustatyti knygos vertę			
Terminas	**C**	**R**	**U**	**D**
Knyga.Verte	✓	✓	✓	
Knyga.Nusidevejimas_metams		✓		
Knyga.Nuomojimo_pradzios_data		✓		
Knyga.Naujos_kaina		✓		

Detalizuojant funkciją *„Nuskaityti knygos vertę“* veiklos diagrama, galime komentaru prie veiklų iliustruoti CRUD matricos reikšmes, kurias naudos konkreti numatoma veikla. Veiklos diagramos pavyzdys su CRUD reikšmių panaudojimu pateiktas 2.16 paveiksle.

2.16 pav. CRUD matricos pritaikymas komentaru veiklos diagramoje

4. Veiklos taisyklių saugyklos transformavimo į UML modelius metodo tyrimas

Tyrimui atlikti suformuluoti tokie uždaviniai:

1. Suprojektuoti ir sukurti dalinai metodą realizuojantį prototipą.
2. Atlikti bandomosios dalykinės srities analizę, remiantis sukurtuoju metodu ir panaudojant sukurtą prototipą.

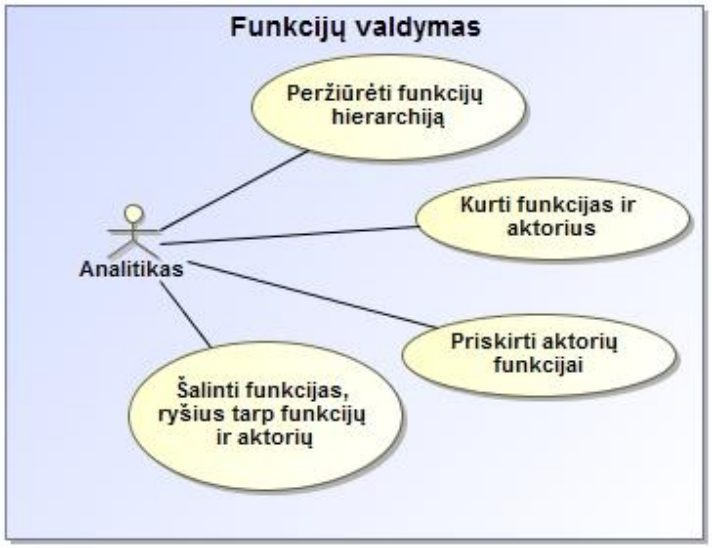
4.1. Dalinės metodo realizacijos prototipo projektas

Projekto uždavinys – suprojektuoti ir realizuoti programą, skirtą veiklos taisyklėmis grindžiamos reikalavimų specifikacijos funkcijų ir veiklos sprendimų poschemės transformavimui į panaudojimo atvejų diagramą.

4.1.1. Funkciniai reikalavimai

Prototipas realizuos funkcijų poschemės duomenų valdymą MySQL duomenų bazėje. 4.1 paveiksle suformuluotos funkcijos, kurias analitikas galės atlikti:

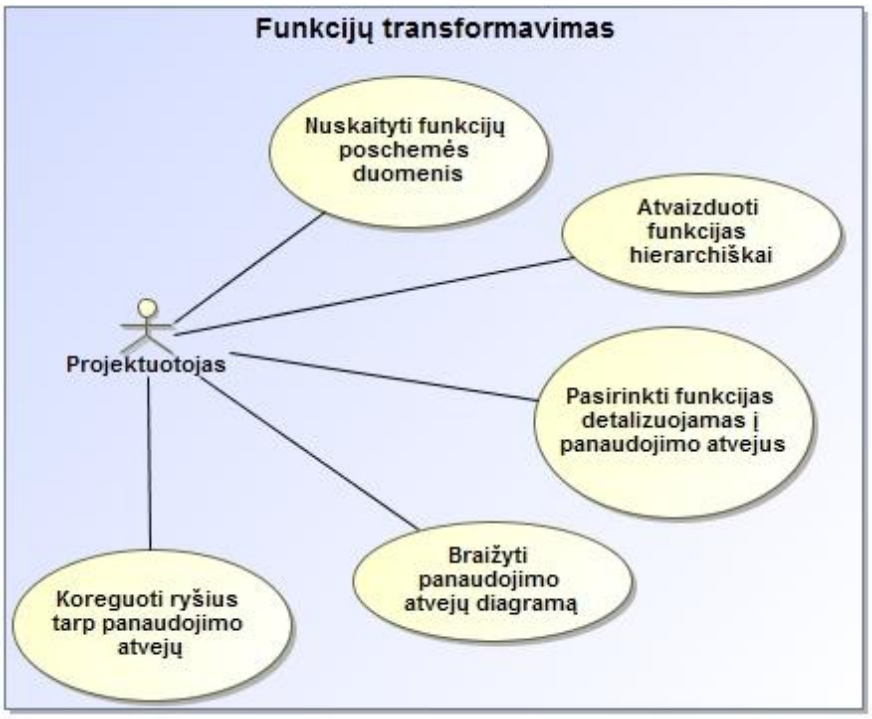
- Peržiūrėti funkcijų hierarchiją. Interneto naršyklės lange atvaizduojamos jau įverstos funkcijos, kurios hierarchiškai struktūrizuotos.
- Kurti funkcijas ir aktorius. Ši funkcija leidžia fiksuoti naujas funkcijas, kurios gali būti formuojamos kaip tėvinės ar priskiriamos jau sukurtai funkcijai. Leidžia fiksuoti aktorius.
- Priskirti aktorių funkcijai. Pagrindinis tikslas - susieti kuo aukštesnio lygio funkciją su aktoriumi.
- Šalinti funkcijas, ryšius tarp funkcijų ir aktorių. Galimybė pašalinti sukurtas funkcijas, panaikinti ryšį tarp aktoriaus ir funkcijos.



4.1 pav. Funkcijų hierarchijos valdymo panaudojimo atvejų diagrama

Funkcijų hierarchijos transformavimą į panaudojimo atvejų diagramą atliks įrankis, kurio funkcijos pavaizduotos 4.2 paveiksle ir aptartos žemiau:

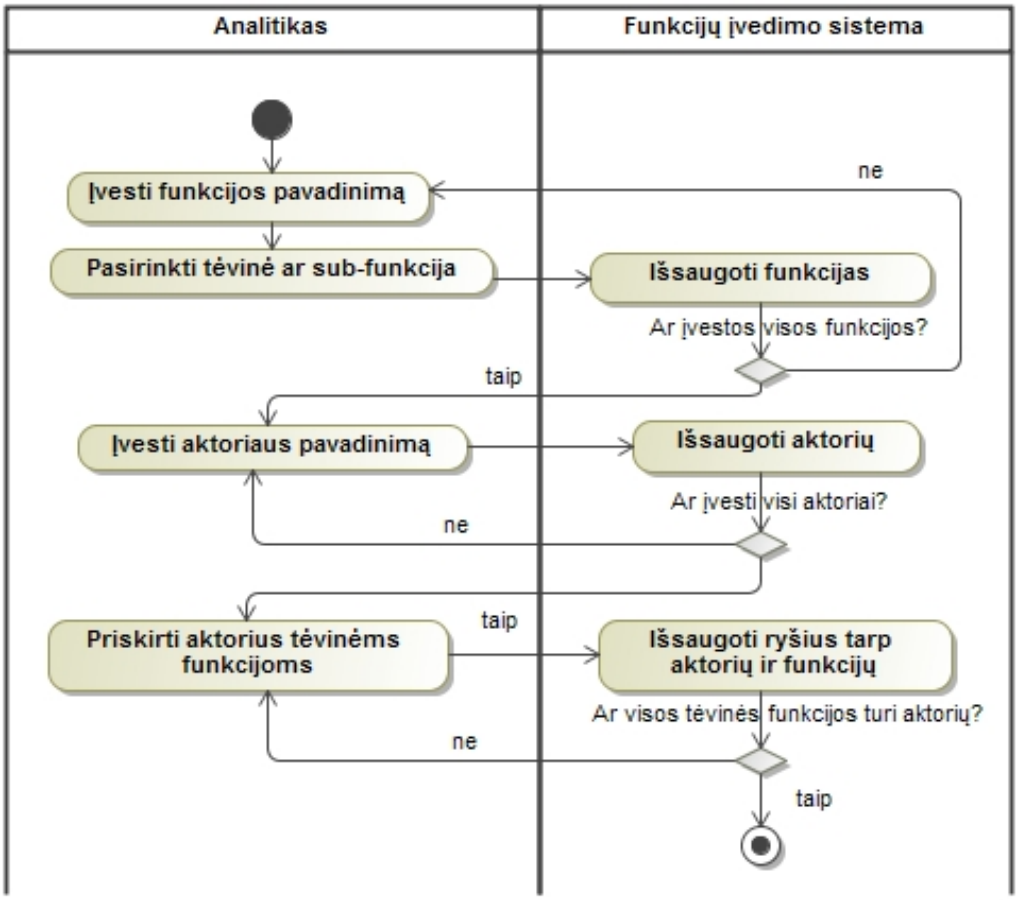
- Nuskaityti funkcijų poschemės duomenis. Ši funkcija nuskaitys duomenų bazėje įvestas funkcijas, jų ryšius ir priskirtus aktorius.
- Atvaizduoti funkcijas hierarchiškai – kuriamo įrankio lange funkcijos suskirstomos pagal įvedimo metu suformuluotą hierarchiją.
- Pasirinkti funkcijas detalizuojamas į panaudojimo atvejus. Pagrindinis tikslas – pasirinkti funkcijas, kurios bus naudojamos panaudojimo atvejų diagramoje.
- Braižyti panaudojimo atvejų diagramą. Funkcija, kuri transformuoja pažymėtas funkcijas į panaudojimo atvejų diagramą.
- Koreguoti ryšius tarp panaudojimo atvejų. Galimybė keisti ryšius tarp panaudojimo atvejų. Pagal nutylėjimą visi panaudojimo atvejai bus sujungti asociacijos ryšiu. Galimi ryšiai tarp panaudojimo atvejų: asociacijos, agregavimo ir išplėtimo.



4.2 pav. Funkcijų transformavimo į panaudojimo atvejų modelį diagrama

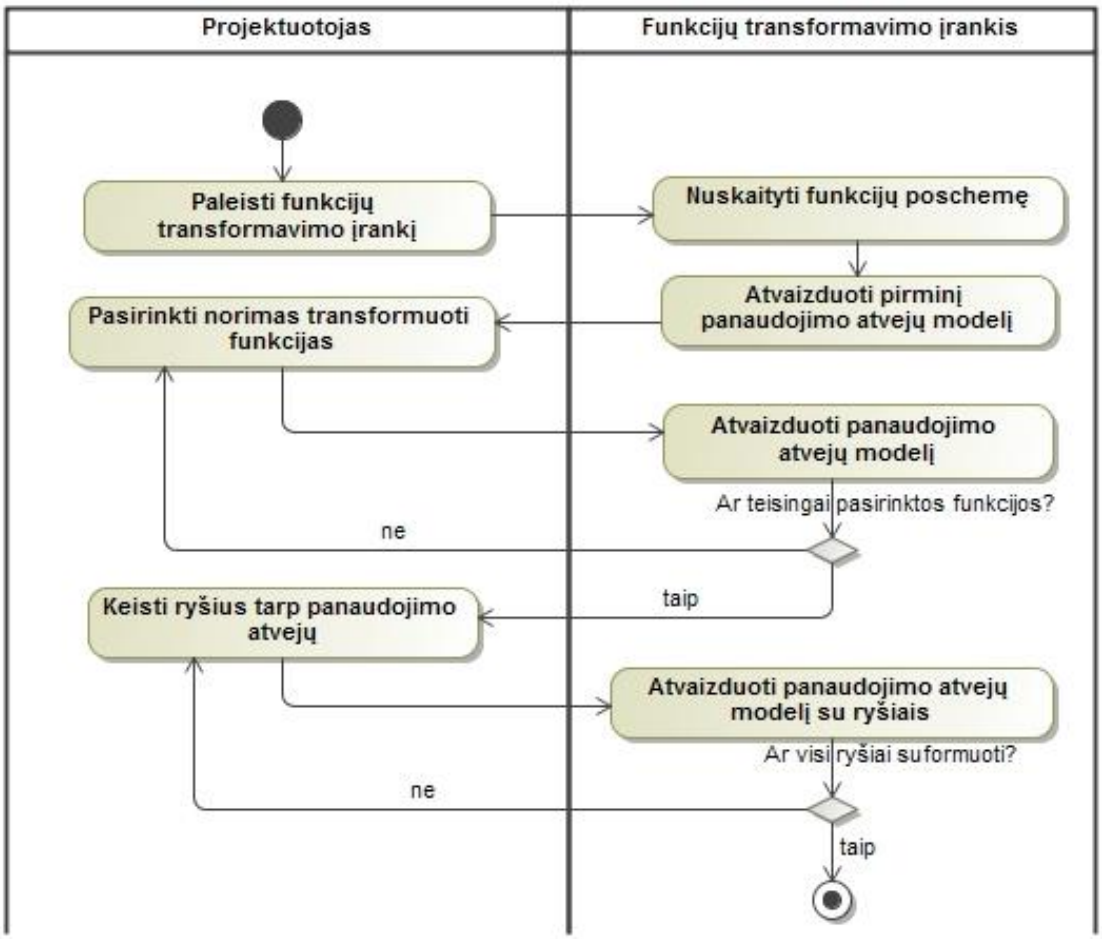
4.1.2. Kompiuterizuojamų procesų modeliai

Funkcijų poschemės duomenų įvedimas pradedamas nuo tėvinių funkcijų formavimo. Tėvinės funkcijos detalizuojamos sub-funkcijomis, kurių suskirstymas į hierarchiją saugomas `FunkcijosAtvaizdas` duomenų bazės lentelėje. Sukuriami aktoriai, kuriems priskiriamos tėvinės funkcijos `FunkcijosAktorius` duomenų bazės lentelėje. Siūlomas funkcijų įvedimo eiliškumo procesas pavaizduotas veiklos diagramoje 4.3 paveiksle.



4.3 pav. Funkcijų poschemės įvedimo veiklos diagrama

Funkcijų transformavimas pradedamas tik teisingai ir pilnai įvedus funkcijas, aktorius ir susiejus ryšius tarp funkcijų ir aktorių. Paleidus įrankį, jis turi nuskaityti duomenų bazę ir atvaizduoti pirminį panaudojimo atvejų modelį, kurį projektuotojas gali keisti. Transformavimo metodika pateikta veiklų diagrama 4.4 paveiksle.



4.4 pav. Funkcijų poschemės transformavimo veiklos diagrama

4.1.3. Vartotojo sąsajos reikalavimai

Funkcijų įvedimo vartotojo sąsajos reikalavimai:

- Intuityvus meniu išdėstymas – meniu punktai turi būti tinkamai suformuluoti ir išdėstyti taip, kad juos būtų galima greitai pasiekti.
- Funkcijos įvedimo metu pasirinkti tai bus tėvinė ar suformuluotos funkcijos sub-funkcija. Pateikiamas funkcijų sąrašas pasirinkimui.
- Funkcijų susiejimas su aktoriumi vykdomas pasirenkant iš funkcijų ir aktorių sąrašo.

Transformavimo įrankio vartotojo sąsajos reikalavimai:

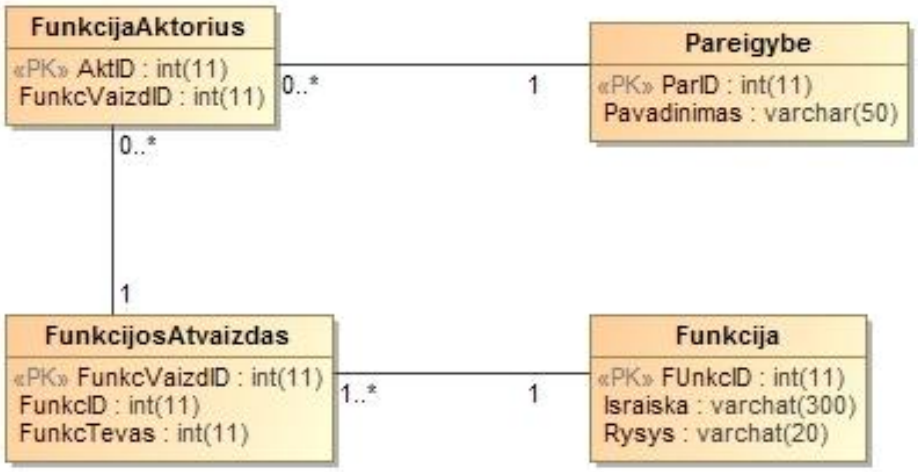
- Grupuojama funkcijų hierarchija. Funkcijos turi būti grupuojamos į išskleidžiamą sąrašą, kuris sutaupo ekrano vietą, kai funkcijų kiekis yra didelis.

- Patogus norimų transformuoti funkcijų pasirinkimas. Norimos transformuoti funkcijos pasirenkamos vienu pelės paspaudimu.
- Ryšių tarp panaudojimo atvejų lange turi būti vaizduojamos tik tos funkcijos, kurias projektuotojas transformuos į panaudojimo atvejų diagramą.
- Galimybė keisti mastelį. Suformavus panaudojimo atvejų diagramą turi būti galimybė pasirinkti norimą mastelio dydį.

4.2.Apibendrintas sistemos projektas

4.2.1. Duomenų struktūros modelis

Eksperimento tikslais veiklos taisyklėmis grindžiamo reikalavimų specifikavimo metodo funkcijų ir veiklos sprendimų poschemė buvo realizuota MySQL duomenų bazių valdymo sistemos aplinkoje (schema pateikta 4.5 pav.). Esybės ir atributai apibūdinti 4.1 – 4.4 lentelėse.



4.5 pav. Reikalavimų saugyklos loginis duomenų modelis naudojamos įrankyje

4.1 lentelė. Funkcija - duomenys apie reikalavimų analizės metu fiksuojamas nagrinėjamoje veikloje vykdomas funkcijas.

Atributas	Aprašas	Galimos reikšmės	Tipas
FunkcID	Unikalus funkcijos identifikacinis numeris.	Įvedimo metu priskiriamas funkcijos identifikacijos numeris	int(11)
Israiska	Funkcijos pavadinimas.	Bet koks tekstas.	varchar(300)
Rysys	Ryšys tarp panaudojimo atvejų	Asociacijos, agregavimo, išplėtimo.	varchar(20)

4.2 lentelė. FunkcijosAtvaizdas - duomenys apie Funkcijų hierarchijos (FH) diagramoje vaizduojamą nagrinėjamos veiklos funkciją.

Atributas	Aprašas	Galimos reikšmės	Tipas
FunkcVaizdID	Unikalus funkcijos atvaizdo identifikacinis numeris.	Įvedimo metu priskiriamas funkcijos atvaizdo identifikacijos numeris	int(11)
FunkcID	Funkcijos, kurią atvaizduoja šis atvaizdas, id.	Pasirinkta *FunkcID* reikšmė iš lentelės *Funkcija*.	int(11)
FunkcTevas	Funkcijos, kurios vaikine funkcija pagal FH diagramą yra ši funkcija, id.	Tokia pasirinkta *FunkcVaizdID* reikšmė iš jau esančių lentelėje *FunkcijosAtvaizdas*, kad *FunkcVaizdID* ≠ *FunkcTevas*.	int(11)

4.3 lentelė. FunkcijaAktorius - duomenys apie funkcijoms priskirtus vykdyti aktorius.

Atributas	Aprašas	Galimos reikšmės	Tipas
FunkcVaizdID	Funkcijos atvaizdo identifikacinis numeris.	Pasirinkta *FunkcVaizdID* reikšmė iš lentelės *FuncijosAtvaizdas*.	int(11)
AktID	Aktoriaus identifikacinis numeris.	Pasirinkta *ParID* iš lentelės *Pareigybe*.	int(11)

4.4 lentelė. Pareigybe - duomenys apie aktorius, kurie vykdo funkcijas projektuojamoje sistemoje.

Atributas	Aprašas	Galimos reikšmės	Tipas
ParID	Unikalus pareigybės identifikacinis numeris.	Įvedimo metu priskiriamas aktoriaus identifikacinis numeris.	int(11)
Pavadinimas	Pareigybės pavadinimas.	Bet koks tekstas.	int(50)

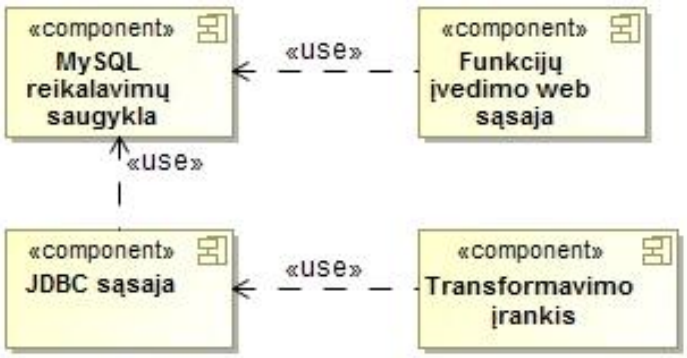
4.2.2. Sistemos realizacijai pasirinktos priemonės

Veiklos taisyklėmis grindžiamo reikalavimų specifikavimo saugykla buvo realizuota Microsoft Access duomenų bazėje. Tačiau ši duomenų bazė yra uždaro kodo ir integracijai kliūčių sukelia lietuviškos raidžių koduotės. Projektuojant sistemą buvo pasirinkta:

- MySQL duomenų bazė – atviro kodo reliacinių duomenų bazių valdymo sistema, kuri suderinama su daugeliu programavimo kalbų.
- PHP programavimo kalba – realizuoti internetinei sąsajai, kuri formuos funkcijų hierarchiją.

- Java objektinė programavimo kalba – realizuoti įrankiui, kuris duomenų bazės sukauptus duomenis transformuos į panaudojimo atvejų diagramą.

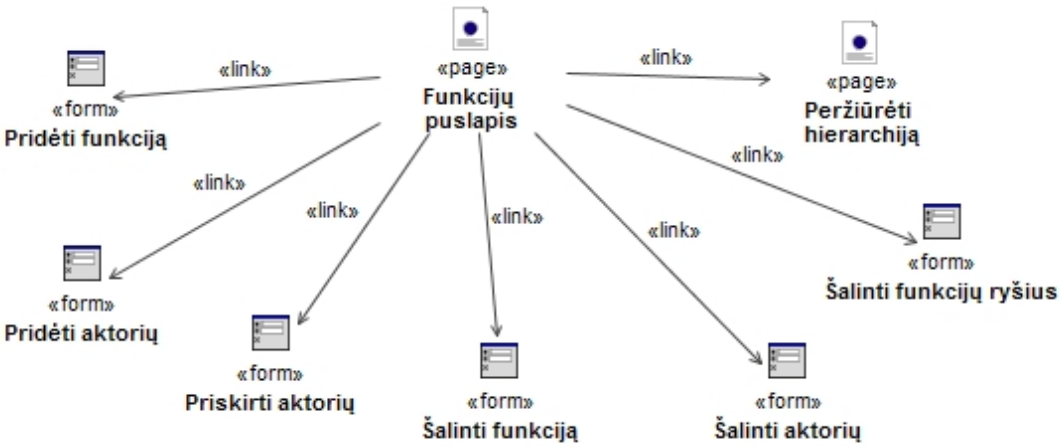
Kuriamo sprendimo pagrindiniai komponentai pateikti 4.6 paveiksle.



4.6 pav. Prototipą iliustruojanti komponentų diagrama

4.2.3. Funkcijų hierarchijos įvedimo internetinės sąsajos navigavimo planas

Projektuojamame funkcijų hierarchijos įvedimo sprendime siekiama viename puslapyje išdėstyti nuorodas, kuriomis projektuotojas galės pasiekti formas, reikalingas funkcijų, aktorių ir ryšių įvedimui ar formatavimui. Navigavimo planas pateiktas 4.7 paveiksle.



4.7 pav. Funkcijų hierarchijos įvedimo navigavimo planas

5. Sukurto įrankio naudojimo ir funkcionalumo bandymas

Trečiojoje šio darbo dalyje suformuotos veiklos taisyklėmis paremtos reikalavimų specifikacijos transformavimas realizuotas tik dalinai, t.y. realizuota funkcijų ir veiklos sprendimų poschemės transformacija į panaudojimo atvejų diagramą. Eksperimentiniam sprendimo bandymui buvo pasirinkta analizuoti bibliotekos dalykinę sritį. Suformuluotos funkcijos ir aktoriai pagal 3.2 skyriuje aptartą bibliotekos dalykinės srities funkcijų hierarchiją.

Svarbu tai, jog funkcijos ir veiklos sprendimai yra tik viena iš analizuojamos reikalavimų specifikacijos komponenčių, tačiau kitų komponenčių – koncepcinio duomenų modelio bei veiklos taisyklių rinkinio – transformavimas šioje dalyje nedemonstruojamas. Toks sprendimas priimtas atsižvelgiant į tai, jog minėtų reikalavimų dedamųjų transformavimas su pavyzdžiais jau aptartas trečiojoje šio darbo dalyje. Be to, jų transformavimo funkcionalumas šiuo metu nėra kompiuterizuotas ir tyrimas būtų tik hipotetinis.

5.1. Pasirengimas eksperimentui

Prototipo realizaciją sudaro dvi dalys:

- http://www.tommoze.lt – internetinė sąsaja taisyklių įvedimui. Ši sąsaja paleidžiama surinkus adresą interneto naršyklėje. Analitikas gali įvesti, šalinti aktorius, priskirti funkcijas aktoriams, pažiūrėti suformuluotą funkcijų hierarchiją.
- Funkcijų poschemės duomenų transformavimo įrankis realizuotas JAVA objektine programavimo kalba. Jis transformuoja funkcijų hierarchiją, kuri saugoma MySQL duomenų bazėje.

Norint sėkmingai naudotis įrankiu internetinio puslapio konfigūravimo *DirectAdmin* panelėje reikia įvesti *IP* adresą, kuris bus naudojamas prisijungimui su transformavimo įrankiu (žr. 5.1 pav.).

5.1 pav. IP adreso priskyrimas transformavimo įrankio funkcionavimui užtikrinti

5.2.Eksperimento su sukurtuoju įrankiu eiga

5.2.1. Funkcijų, aktorių, ryšių tarp aktoriaus ir funkcijos kūrimo eiga

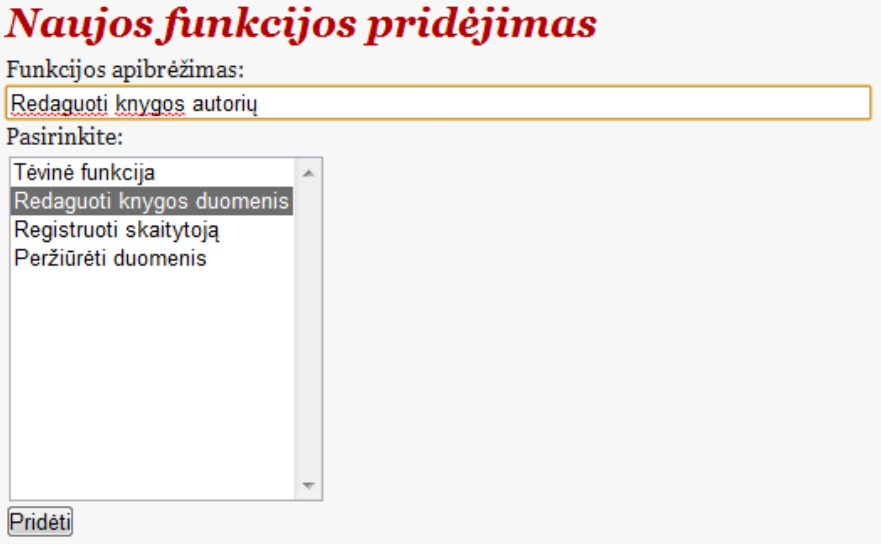
Interneto naršyklėje adreso lange surenkame adresą http://www.tommoze.lt. Atsidariusiame lange matome metodo informaciją su programos langais. Pasirinkus meniu punktą *Funkcijos* atvaizduojami galimi veiksmai formuojant funkcijų hierarchiją (žr. 5.2 pav).

5.2 pav. Internetinės sąsajos veiksmų pasirinkimo langas

Norint suformuoti funkcijų hierarchiją, kurią transformuosime į panaudojimo atvejų modelį reikia atlikti šią veiksmų seką:

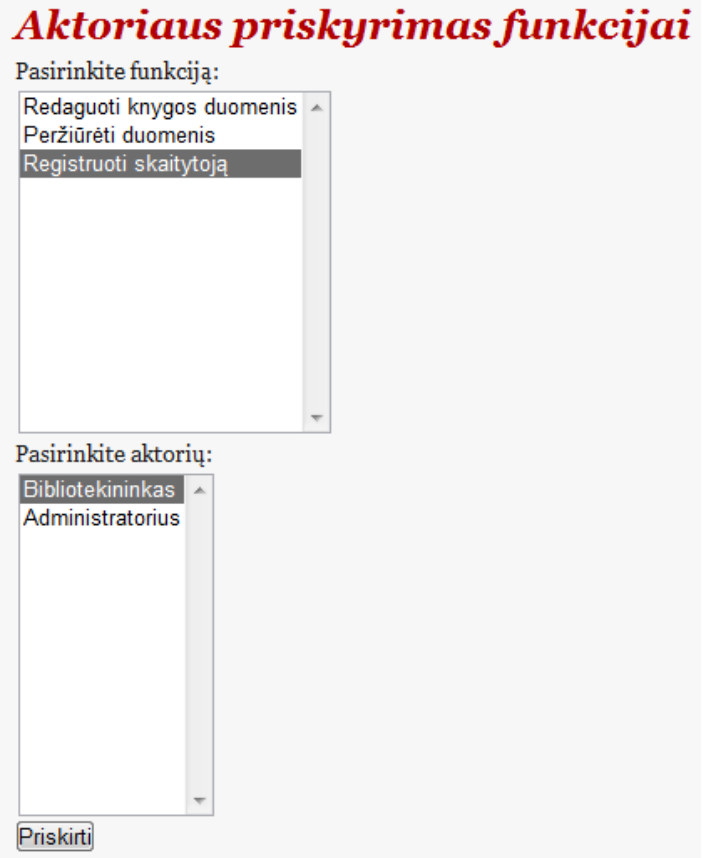
- *Funkcijos* lange pasirenkame punktą *Pridėti funkciją*. Atidarytoje įvedimo formoje suformuluojamas funkcijos apibrėžimas. Pasirinkimo lange nurodoma tai bus tėvinė funkcija ar kurią jau įvestą funciją detalizuojanti sub-funkcija. Funkcijos įvedimo forma pateikta 5.3 pav.

- *Funkcijos* lange pasirenkame punktą *Pridėti aktorių.* Šis punktas leidžia sukurti aktorių, kuris bus susiejamas su jam galimomis vykdyti funkcijomis.
- *Funkcijos* lange pasirenkame punktą *Priskirti aktorių.* Šioje formoje funkcijoms priskiriamas aktorius. Aktorių priskyrimo forma pavaizduota 5.4 paveiksle.



5.3 pav. Funkcijos įvedimo forma

Aktoriaus priskyrimo formoje atvaizduojamos tik tėvinės funkcijos, kadangi pagal VT grindžiamą reikalavimų metodą aktoriai turi būti susiejami su kuo aukstesnio lygio funkcijomis. Priskyrimas vykdomas pasirenkant norimą funkciją ir jai priskiriamą aktorių.

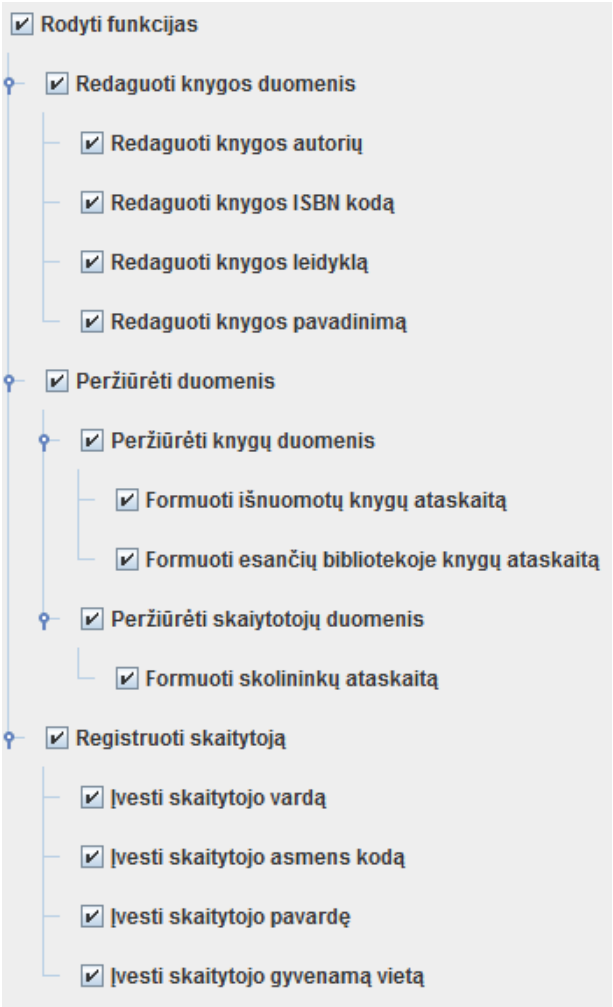


5.4 pav. Aktoriaus priskyrimo funkcijai forma

Prireikus interneto sąsajoje galima šalinti jau sukurtas funkcijas, aktorius, ryšius tarp panaudojimo atvejų, atitinkamai pasirenkant nuorodas iš *Funkcijos* meniu punkto.

5.2.2. Suformuluotos funkcijų hierarchijos transformavimo į panaudojimo atvejų diagramą eiga

Paleidus transformavimo įrankį, jis sugeneruoja panaudojimo atvejų diagramą iš visų įvestų funkcijų. Pagal nutylėjimą visi panaudojimo atvejai sujungiami asociacijos ryšiu. Dalykinė sritis eksperimentui buvo pasirinkta pagal 3.2 punkte aptartą bibliotekos funkcijų hierarchiją. Įrankio suformuluota hierarchija pavaizduota 5.5 paveiksle.



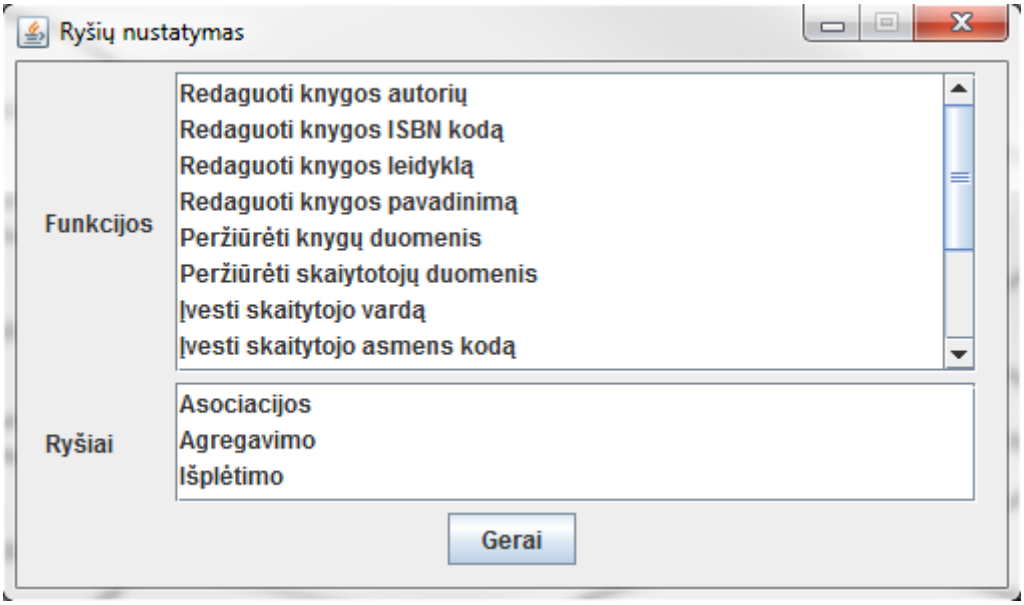
5.5 pav. Įrankio suformuluota bibliotekos dalykinės srities funkcijų hierarchija

Kadangi suformuluotų funkcijų skaičius gali būti didelis, įrankio funkcijų hierarchijoje pritaikytas medžio principas, kuris visas sub-funkcijas „suskleidžia“ į tėvinę funkciją. Paspaudus ant tėvinės funkcijos išskleidžiamas sub-funkcijų sąrašas. Bibliotekos dalykinės srities funkcijų hierarchija transformuota į panaudojimo atvejų diagramą pavaizduota 5.6 paveiksle.

5.6 pav. Įrankyje suformuluota pradinė panaudojimo atvejų diagrama

Projektuotojas gali redaguoti norimas atvaizduoti funkcijas nuimdamas ar uždėdamas varneles funkcijų hierarchijoje. Įrankis iš kart perbraižo panaudojimo atvejų diagramą vos tik pakeitus norimų transformuoti funkcijų kiekį. Interneto sąsajoje atlikus funkcijų hierarchijos pakeitimus įrankyje reikia pasirinkti meniu *Taisa* ir pasirinkti punktą *Perkrauti hierarchiją*. Tada įrankis nuskaitys naujai suformuotus duomenis bei pateiks transformuotą panaudojimo atvejų diagramą.

Norint pakeisti ryšius tarp panaudojimo atvejų iškviečiamas ryšių keitimo langas, kuris pasirenkamas meniu *Taisa* ir pasirinkus *Ryšiai*. Ryšių keitimo lange pateikiamos funkcijos, kurioms galima pakeisti ryšius. Pagal nutylėjimą nuskaičius duomenų bazę visi ryšiai būna asociacijos tipo. Ryšių redagavimo langas pateiktas 5.7 paveiksle.



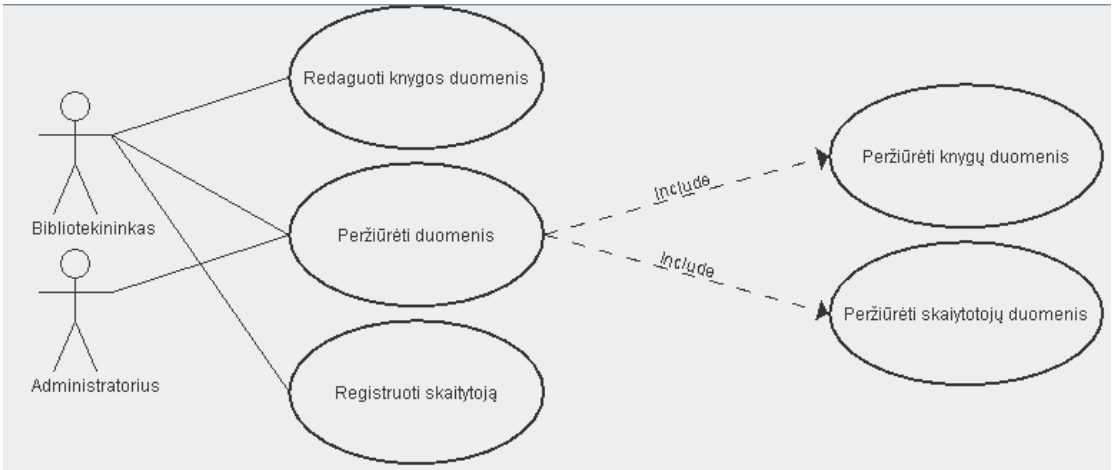
5.7 pav. Ryšių keitimo langas

Pasirinkus norimą panaudojimo atvejo funkciją nurodomas pageidaujamas priskirti ryšys. Patvirtinus pasirinkimą mygtuku *Gerai* panaudojimo atvejų diagrama yra perbraižoma atvaizduojant norimą ryšį.

Norėdami suformuluoti neperkrautą ir pagrindines funkcijas apibrėžiančią panaudojimo atvejų diagramą funkcijų hierarchijoje pažymime funkcijas:

- Redaguoti knygos duomenis
- Registruoti skaitytoją
- Peržiūrėti duomenis
- Peržiūrėti knygų duomenis
- Peržiūrėti skaitytojų duomenis

Nustačius transformuojamoms funkcijoms ryšius, įrankyje suformuluota panaudojimo atvejų diagrama pavaizduota 5.8 paveiksle.



5.8 pav. Įrankio suformuota panaudojimo atvejų diagrama

5.3.Sukurto įrankio įvertinimas

Vertinimui pasirinkta trijų balų sistema, kur:

- 1 – blogas funkcionalumas;
- 2 – vidutinis funkcionalumas;
- 3 – geras funkcionalumas;

5.1 lentelė. Įrankio įvertinimas

Kriterijus	Vertinimas	Komentaras
Funkcijų, aktorių, ryšių įvedimas	3	Realizuotas funkcijų poschemės duomenų įvedimas formomis, kur galima lengvai priskirti sub-funkcijas, aktorius funkcijoms
Funkcijų šalinimas	2	Funkcijų šalinimo formoje atvaizduojamos tik tos funkcijos, kurios neturi jas detalizuojančių funkcijų. Norint ištrinti tėvinę funkciją reikia atlikti jos vaikinių funkcijų šalinimą ir tik po to atvaizduojamas tėvinės funkcijos šalinimas
Funkcijų hierarchijos atvaizdavimas transformavimo įrankyje	3	Hierarchija atvaizduojama medžio principu, kur įdiegtas medžio „suskleidimas“.
Funkcijų atvaizdavimas į panaudojimo atvejų diagramą	3	Norimų funkcijų transformavimas į panaudojimo atvejų diagramą vykdomas „varnelės“ uždėjimu ar nuėmimu. Diagramoje galima keisti ryšius, mastelį.
Panaudojimo atvejų diagramos saugojimas įrankyje	Nėra	Ši funkcija nebuvo realizuota transformavimo įrankyje

5.4.Apibendrintas sukurtosios metodikos įvertinimas

Nors prototipu realizuota ir išbandyta tik funkcijų hierarchijos duomenų poschemės transformacija į panaudojimo atvejų diagramą, tačiau parengtos transformavimo gairės trečiame skyriuje, pagal kurias galima transformuoti kitas reikalavimų specifikacijos dedamąsias. 5.2 lentelėje apibrėžtas siūlomas specifikacijos transformavimas į UML kalbos diagramas.

5.2 lentelė. Siūlomos VT taisyklėmis grindžiamos reikalavimų specifikacijos dedamųjų transformavimas

Reikalavimų specifikacijos dedamoji	Pasiūlytas (dalinis) transformavimas į UML modelį(-ius)
Funkcijų hierarchija	Panaudojimo atvejų diagramą
Struktūrinės veiklos taisyklės	Duomenų klasių modelį
Nestruktūrinės VT specifikuotos BRS RuleSpeak šablonais	Veiklos diagramos,komentarai duomenų klasių diagramoje priskirti atributams ir esybėms
CRUD matrica	Komentarais veiklos diagramose

6. Išvados

1. Analizės metu nustatyta, jog taikant veiklos taisyklių koncepciją informacinių sistemų reikalavimų specifikavimo fazėje, gali būti gauta geriau realią situaciją atitinkanti reikalavimų specifikacija. Ji savo ruožtu gali būti naudojama projektuojant sistemą ne tik veiklos taisyklėmis grindžiamais, bet ir tradiciniais, veiklos taisyklių neišskiriančiais metodais.
2. Išanalizavus tradicinius sistemų projektavimo metodus pasirinkta veiklos taisyklėmis grindžiamą reikalavimų specifikaciją transformuoti į UML kalbos diagramas, remiantis objektiškai orientuotu sistemų kūrimo metodu.
3. Suformavus veiklos taisyklėmis grindžiamų reikalavimų specifikacijos transformavimo metodą parodyta, jog teoriškai reikalavimai gali būti dalinai automatiškai transformuojami į UML panaudojimo atvejų, veiklos ir klasių diagramas ar jų elementus.
4. Parengtos gairės modifikuotais BRS RuleSpeak natūraliosios kalbos šablonais užfiksuotoms nestruktūrinėms veiklos taisyklėms transformuoti į UML diagramų elementus leidžia dalinai automatizuoti šių ir kitų pagal nagrinėtą metodą užfiksuotų reikalavimų formalizavimą.
5. Sukūrus ir išbandžius dalinę prototipinę metodikos realizaciją, parodyta, jog reikalavimų specifikavimo metu sukurto modifikuoto funkcijų hierarchijos modelio transformavimas į UML panaudojimo atvejų diagramą gali būti kompiuterizuotas. Tai leidžia daryti prielaidą, jog ir kiti metodikoje numatyti transformavimo veiksmai gali būti kompiuterizuoti ir taikomi praktikoje.

7. Literatura

[1] Kęstutis Kapočius (2006). *Disertacija. Veiklos taisyklių struktūrizavimo modeliai ir jų taikymas kuriant informacijos sistemas.*

[2] Kęstutis Kapočius, Egidijus Merkevičius, Gintautas Garšva (2008). *Function-Decision-Rule Modeling as a Bankruptcy Prediction Model Specification Tool.*

[3] Kestutis Kapočius, Gintautas Garšva (2008), *Business Rules as Part of Information Systems Life Cycle: Possible Scenarios.*

[4] Von Halle, B. (2001). *Business rules applied: building better systems using the business rules approach*.

[5] Ross, R. G. (1997). *The Business Rule Book: Classifying, Defining an Modeling Rules*.

[6] OMG Available Specification (2008). *Semantics of Business Vocabulary and Business Rules.* Prieiga per internetą: < http://www.omg.org/spec/SBVR/1.0/>

[7] OMG Available Specification (2009). *Production Rule Representation, v1.0.* Prieiga per internetą: < http://www.omg.org/spec/PRR/1.0/>

[8] The Business Rules Group, *Business Rules Manifest*o. Prieiga per internetą: < http://www.businessrulesgroup.org/brmanifesto/BRManifesto.pdf>

[9] Marting Folower (2003). *UML Distilled: A Brief Guide to the Standart Object Modeling Language Third Edition.*

[10] K. Kendall, J. Kendall (2011) *Systems Analysis and Design 8th ed.*

[11] The Business Rules Group (2000). *Defining Business Rules ~What Are They Really? formerly, known as the GUIDE Business Rules Project Final Reportrevision 1.3*. Prieiga per internetą: < http://www.businessrulesgroup.org/first_paper/BRG-whatisBR_3ed.pdf>

[12] Kęstutis Kapočius, Rimantas Butleris (2006). *Repository for Business Rules Based IS Requirements*.

[13] Agile metodo manifesto tinklapis. Prieiga per internetą: <http://agilemanifesto.org/>

8. Priedai

1 Priedas

Modifikuoto RuleSpeak modelio aprašai ir VT pavyzdžiai kiekvienam taisyklių tipui [1]

Kategorija	Neformalus apibrėžimas	Galimi subjekto tipai	Šablonas	VT pavyzdžiai iš bibliotekos veiklos
1.1 Atmetimo taisyklė/atmetimas/apribojimo taisyklė (*rejector*)	Apribojimas, kurio nepažeidžiant užtikrinamas duomenų teisingumas (neprieštaringumas)	Terminas, faktas, duomenų elementas	*b1.* <Subjektas> TURI|NETURI|TURĖTŲ |NETURĖTŲ <faktas> [(, jeigu |, kol) <sąlyga>].	Skaitytojas turi būti priregistruojamas bibliotekoje, jeigu pateikia asmens tapatybę įrodantį dokumentą. Tapatybę įrodantis dokumentas turi turėti asmens kodą.
			b2. <Subjektas> gali | turėtų <faktas> TIK jeigu | kol <sąlyga>.	Skaitytojas gali išsinuomoti knygą tik jeigu nuomojamų knygų skaičius neviršija 10.
			b3. <Subjektas> gali | turėtų <faktas> TIK <sąlyga>	Skaitytojas gali skaityti literatūrą bibliotekoje tik palikdamas skaitytojo pasą bibliotekininkei.
1.2 Leidimo taisyklė (*permission statement*)	Strateginė nuostata ar paaiškinimas, leidžiantis vykdyti veiklą.	Terminas, faktas, taisyklė, procesas, duomenų elementas	*b4.* <Subjektas> GALI <faktas | VT raktažodis> [(,net jeigu |,kol) <sąlyga>].	Skaitytojas gali išsinuomoti knygą, kol delspinigių suma neviršija 10 litų.
			b5. <Subjektas> NETURI <faktas> | <VT raktažodis> [(, jeigu |, kol) <sąlyga>].	Skaitytojas neturi gauti antros vienodos knygos.
2.1 Skaičiavimo taisyklė (*computation rule*)	Teiginys ar aritmetinė formulė, nurodanti, kaip apskaičiuoti tam tikrą skaitmeninę reikšmę.	Apskaičiuojamas terminas (reikšmė), duomenų elementas	*b6.* <Subjektas> turi|neturi|turėtų|neturėtų BŪTI APSKAIČIUOJAMAS|-a|-i|-os kaip <matematinė formulė> [(, jeigu |, kol) <sąlyga>].	Delspinigiai už literatūrą turi būti apskaičiuojami pagal esamą dienos negražinimo tarifą padaugintą iš negražintų dienų skaičiaus.
			b7. Sutrumpintas variantas: <Subjektas> = <matematinė formulė> [(, jeigu |, kol) <sąlyga>].	Delspinigiai už literatūrą = dienos negražinimo tarifas padaugintas iš negražintų dienų skaičiaus.

2.2 Išvedimo taisyklė (*derivation rule*)	Teiginys ar loginė išraiška, nusakanti, kaip nustatyti taip/ne pobūdžio rezultatą.	Išvestinis terminas (atributas, esybė, reikšmė), duomenų elementas	*b8.* <Subjektas> turi|neturi|turėtų|neturėtų REIKŠTI, KAD <loginė išraiška> [(, jeigu |, kol) <sąlyga>].	Neatsakingas skaitytojas turi reikšti, kad delspinigių suma viršija 50 lt.
			b9. Sutrumpintas variantas: <Subjektas> REIŠKIA, KAD | NEREIŠKIA, KAD <loginė išraiška> [(, jeigu |, kol) <sąlyga>].	Neatsakingas skaitytojas turi reikšti, kad delspinigių suma viršija 50 lt.
3.1.1 Išvados taisyklė (*inference rule*)	Taisyklė, pagal kurią ,atsižvelgiant į tam tikras aplinkybes, padaroma išvada.	Terminas, duomenų elementas	*b10.* <Subjektas> turi|neturi|turėtų|neturėtų BŪTI LAIKOMAS|-a|-i|-os <terminas> (, jeigu |, kol) <sąlyga>.	Knygos turi būti laikomos neišnuomojamomis, jeigu jų rinkos vertė viršija 500 lt.
			b11. Sutrumpintas variantas: <Subjektas> YRA | NĖRA <terminas> (, jeigu |, kol) <sąlyga>.	Knygos yra neišnuomojamos, jeigu jų rinkos vertė viršija 500 lt.
3.1.2 Taisyklės jungiklis (*rule toggle*)	Taisyklė, "įjungianti" ar "išjungianti" kitą taisyklę, priklausomai nuo konkrečių aplinkybių. Naudojama taisyklių išimtims nusakyti.	Taisyklė	*b12. Neformalus variantas:* <Taisyklės teiginys> NEBENT | IŠSKYRUS <sąlyga>.	Neišnuomojamą knygą skaitytojas gali išsinuomoti nebent sumokėjęs 5% knygos vertės.
			b13. Formalus variantas: <Taisyklės pavadinimas | kodas> turi|neturi|turėtų|neturėtų BŪTI TAIKOMA|-as (, jeigu |, kol) <sąlyga>.	Neišnuomojamų knygų nuoma neturėtų būti taikoma, jeigu skaitytojas yra nesumokėjęs delspinigių.
3.1.3 Proceso jungiklis (*process toggle*)	Taisyklė, esant tam tikroms aplinkybėms "įjungianti" ar "išjungianti" procesą.	Procesas	*b14.* <Subjektas> turi|neturi|turėtų|neturėtų BŪTI ĮGALINTAS|-a | BŪTI UŽDRAUSTAS|-a (, jeigu |, kol) <sąlyga>.	Knyga neturi būti priimta iš skaitytojo, jeigu matomi knygos pažeidimai.
3.1.4 Duomenų jungiklis (*data toggle*)	Taisyklė, esant tam tikroms aplinkybėms šalinanti (ar sukurianti atsitiktinius) duomenis.	Duomenų elementas	*b15.* <Duomenų elementas> turi|neturi|turėtų|neturėtų BŪTI SUKURTAS|-a | IŠTRINTAS|-a jeigu | kol <sąlyga>.	Delspinigiai neturi būti ištrinti, kol skaitytojas negražina knygos ir nesumoka delspinigių sumos.

3.2.1 Reikšmės priskyrimo taisyklė (*imprint rule*)	Taisyklė, priskirianti saugomam duomenų elementui tam tikrą reikšmę.	Terminas, faktas, duomenų elementas	*b16.* <Subjektas> turi|neturi|turėtų|neturėtų ĮGYTI REIKŠMĘ <terminas | reikšmė> [kai | jeigu <sąlyga>].	Skaitytojas turi įgyti reikšmę skolininkas, jeigu pasibaigus knygos nuomos terminui jos dar negražina 30 dienų.
3.2.2 Pateikimo (atvaizdavimo) taisyklė (*presentation rule*)	Taisyklė, kuria nustatoma duomenų pateikimo forma (ekrane, ataskaitoje ir pan.).	Terminas, faktas, duomenų elementas	*b17.* <Subjektas> turi|neturi|turėtų|neturėtų BŪTI PATEIKIAMAS|-a|-i|-os [<vaizdavimo terpė>] <vaizdavimo būdas> [jeigu | kol <sąlyga>].	Knygų paieškos lange turi būti pateikiama informacija apie knygos nuomos būseną.
3.3.1 Proceso trigeris (*process trigger*)	Taisyklė, pagal kurią, susidarius tam tikroms aplinkybėms, automatiškai vykdomas procesas ar procedūra.	Procesas, procedūra	*b18.* <Subjektas> turi|neturi|turėtų|neturėtų BŪTI VYKDOMAS|-a , kai | kol <sąlyga>.	Siųsti-knygos-nuomos-termino-pabaigos-įspėjimą turi būti vykdomas, kai iki knygos gražinimo datos lieka 10 dienų.
3.3.2 Taisyklės trigeris (*rule trigger*)	Taisyklė, pagal kurią, susidarius tam tikroms aplinkybėms, automatiškai paleidžiama kita taisyklė.	Taisyklė	*b19.* <Taisyklės pavadinimas | kodas> turi|neturi|turėtų|neturėtų BŪTI PALEISTA|-as , kai <sąlyga>.	Anuliuoti delspinigius turi būti paleista, kai skaitytojas miršta.