KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Egidijus Kumža

Tikslais grindžiamų reikalavimų modeliavimo ir sekimo metodika

Magistro darbas

Darbo vadovas
prof. dr. L. Nemuraitė

Kaunas, 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Egidijus Kumža

Tikslais grindžiamų reikalavimų modeliavimo ir sekimo metodika

Magistro darbas

Recenzentas
doc. dr. V. Pilkauskas
2009-01-12

Vadovas
prof. dr. L. Nemuraitė
2009-01-12

Atliko IFM-3/4 gr. stud.
Egidijus Kumža
2009-01-12

Kaunas, 2009

Turinys

Paveikslų sąrašas

Lentelių sąrašas

Kumža E. **Methodology for Goal Based Requirements Representation and Tracing:** Master of Information systems engineering / supervisor prof. PhD L. Nemuraite; Faculty of Informatics, Kaunas University of Technology – Kaunas, 2009 –77 p.

Summary

There are many Information System requirement specification methods, but not much of them are related with goals. This work presents analysis of methods related with goal modelling and requirement specification.

The field of this work is specification methods of goals and requirements.

The object of this work is goal driven requirement specification process.

The problem of the work is alignment of requirements to business goals and tracing the compliance of goals, requirements and design artefacts during development process.

The main result of analysis is revealing of perspectives for extending requirement specification method with goal modelling and analysis, relating and tracing goals to requirements and software artefacts, and integration of these steps into standard unified development process. For this purpose, a UML profile was created and implemented in CASE tool "Magic Draw". Experimental investigation of created means by applying them to IS development demonstrated the suitability of the approach.

Key words: requirement specification, Volere, RUP, Kaos, Tropos, Goals

Įvadas

Daugelyje organizacijų aktuali informacinių technologijų atitikimo veiklos poreikiams problema. Vienas iš būdų ją spręsti – susieti informacinių sistemų programinės įrangos reikalavimų specifikacijas su veiklos tikslais. Reikalavimų specifikavimo metodų yra, bet nedaugelis iš jų siejasi su tikslais. Praktikoje naudojami metodai nesusieti su tikslais. Dėl reikiamų modeliavimo priemonių CASE įrankiuose nebuvimo tikslų modeliavimas praktiniuose kūrimo procesuose yra neatliekamas arba atliekamas tik dalinai. Planuojant informacinių sistemų projektus, tikslinga analizuoti tikslus, nes taip galima padidinti informacinių sistemų tinkamumą veiklos poreikiams. Todėl tikslais grindžiamo reikalavimų specifikavimo metodo sukūrimas yra aktuali ir novatoriška problema.

Šio darbo *tyrimo sritis* yra tikslų ir reikalavimų specifikavimo metodai.

Darbo *tyrimo objektas* yra tikslais grindžiamas reikalavimų specifikavimo procesas.

Šio darbo *tyrimo problema* yra reikalavimų suderinamumas su veiklos tikslais, taip pat tikslų, reikalavimų ir juos realizuojančių artefaktų atitikimo sekimas projektavimo metu.

Darbo tikslas yra sudaryti reikalavimų specifikacijos modelį, kuriame specifikacijų elementai būtų susieti tarpusavyje ir turėtų ryšį su organizacijos tikslais. Taip pat reikia pateikti priemones tikslų, reikalavimų ir artefaktų atitikimui sekti projektavimo metu.

Darbo uždaviniai:

- išanalizuoti tikslų ir reikalavimų specifikavimo metodus, palyginti juos tarpusavyje;
- atlikti tikslų ir reikalavimų specifikavimo priemonių CASE įrankyje UML Magic Draw analizę;
- sudaryti reikalavimų specifikacijos metamodelį, kuris padėtų suderinti tikslus ir reikalavimus ir suteiktų galimybę atsekti reikalavimus projekte;
- išbandyti sudarytą metodą aprašant reikalavimų specifikacijas realiam projektui;
- palyginti sudarytą modelį su esamais sprendimais.

Atlikus reikalavimų specifikavimo metodų analizę, buvo atskleistos atskirų metodų savybės. *Volere* ir RUP reikalavimų specifikavimo metodai mažai dėmesio skiria tikslams. *Tropos* ir ypač KAOS metodai yra orientuoti į tikslus. Norint įtraukti tikslų analizę į praktinį projektavimo procesą, tikslinga panaudoti įvairių metodų elementus.

Todėl sukurtas tikslais grindžiamų reikalavimų metamodelis ir UML profilis, kuris leidžia vaizduoti tikslus, su jais susijusius panaudojimo atvejus ir detalizuotus reikalavimus.

Sukurtas profilis yra realizuotas UML CASE įrankio Magic Draw DSL kūrimo priemonėmis. Pritaikius šį profilį pavyzdinei sistemai modeliuoti, išsiaiškinta, kad jis leidžia

paprastai, nenaudojant papildomų paketų sumodeliuoti tikslus ir susieti juos su detaliai specifikuotais reikalavimais ir vėlesnių stadijų artefaktais.

Norint patikrinti metodikos funkcionalumą bei reikalavimų atsekamumą, metodika buvo testuojama. Eksperimentui buvo pasirinktas magistro darbas, kuriame buvo suprojektuota ir sukurta realiai veikianti internetinio aukciono sistema. Iš pirmo žvilgsnio darbas atrodo parašytas ir dokumentuotas puikiai, tačiau atlikus eksperimentą išaiškėjo, kad net nedideliame projekte yra daugybė neatitikimų ir praktiškai sunku atsekti, ar realizacija atitinka pradinius tikslus ir reikalavimus, ir kokiuose programinės įrangos artefaktuose realizuoti konkretūs reikalavimai. Reikalavimų atsekimo galimybės dar svarbesnės dideliuose projektuose, kuriuos kuria ne vienas dalyvis.

Taikydami sudarytą metodiką, analitikai, projektuotojai ir kūrėjai galėtų tiksliai susieti savo kuriamus artefaktus su ankstesnių stadijų artefaktais, o tai palengvintų projektų vertintojų, testuotojų ir integruotojų darbą ir pagerintų projektų kokybę.

Darbo struktūra:

- *Analizės* dalyje yra atliekama reikalavimų specifikavimo metodų analizė. Aprašoma tyrimo sritis, objektas, problema.
- *Projekto* dalyje aprašomas projekto tikslas, sukuriamas reikalavimų metamodelis ir aprašoma kiekviena jo dalis atskirai. Apibūdinama kokios diagramos kokiuose sistemos kūrimo etapuose turi būti vaizduojamos.
- *Realizacijos dalyje* aprašomas reikalavimų sekimo profilio kūrimas.
- *Eksperimentinio tyrimo dalyje* pateiktas sukurto profilio bei metamodelio eksperimentas realiai sistemai.
- *Išvadose* pateikiami apibendrinti tyrimo rezultatai.

Darbo tema buvo parašytas ir atspausdintas straipsnis, kuris buvo pristatytas konferencijoje „IT2008“.

1. Tikslais grindžiamų reikalavimų specifikavimo metodų analizė

Analizės tikslas – atlikti:

- Reikalavimų specifikavimo ir tikslų modeliavimo metodų analizę
  - TROPOS metodo analizė
  - KAOS metodo analizė
  - RUP panaudojimo atvejų šablono analizė
  - VOLERE šablono analizė

Šio darbo **tyrimo sritis** yra tikslų ir reikalavimų specifikavimo metodai.

Šios darbo **tyrimo objektas** yra tikslais grindžiamas reikalavimų specifikavimo procesas.

Šio darbo **tyrimo problema** yra reikalavimų suderinamumas su veiklos tikslais, tikslų ir reikalavimų atitikimo sekimas projektavimo metu.

1.1. Darbo tikslas ir uždaviniai

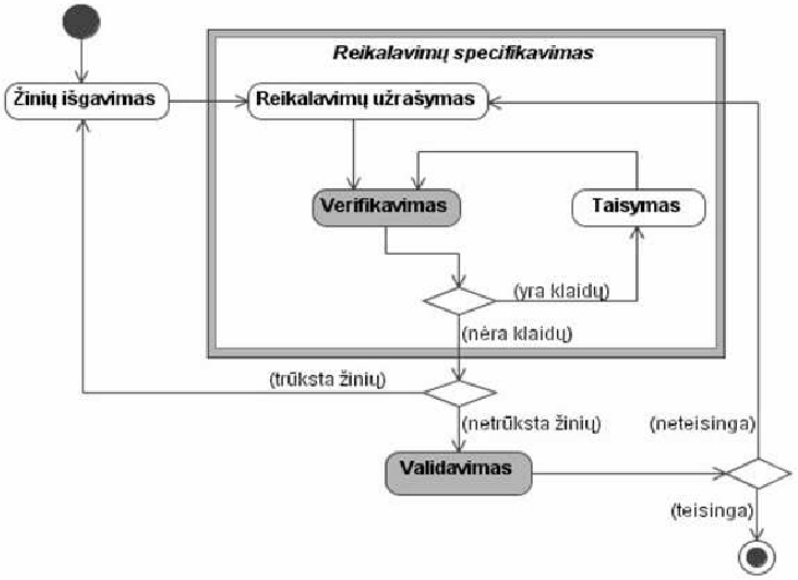
Darbo tikslas yra sukurti galimybes pagerinti informacinių sistemų projektų kokybę ir jų atitikimą veiklos tikslams, sudarant reikalavimų specifikavimo modelį ir CASE įrankio priemones, kurios padėtų atsekti reikalavimų specifikacijų elementus ir jų ryšius su organizacijos tikslais ir programinės įrangos artefaktais.

Darbo uždaviniai yra:

- išanalizuoti tikslų ir reikalavimų specifikavimo metodus, palyginti juos tarpusavyje;
- atlikti tikslų ir reikalavimų specifikavimo priemonių CASE įrankyje UML Magic Draw analizę;
- sudaryti reikalavimų specifikacijos metamodelį, kuris padėtų suderinti tikslus ir reikalavimus ir suteiktų galimybę atsekti reikalavimus projekte;
- išbandyti sudarytą metodą aprašant reikalavimų specifikacijas realiam projektui;
- palyginti sudarytą modelį su esamais sprendimais.

1.2. Tyrimo objekto analizė

Tyrimo objekto analizės tikslas išanalizuoti reikalavimų specifikacijos šablonų struktūrą ir numatyti galimus šablonų tobulinimo būdus. 1.1 paveiksle pateikta standartinė reikalavimų specifikavimo proceso diagrama [5].



1.1 pav. Reikalavimų specifikavimo proceso diagrama [5]

1.3. Tikslais grindžiamų reikalavimų specifikavimo metodų vartotojų analizė

Šiame darbe vartotojai yra tie žmonės, kurie specifikuoja reikalavimus (sistemų analitikai – projektuotojai ir t.t.). Vartotojai turi aukštą kvalifikaciją IT, reikalavimų analizės, projektavimo srityje.

Reikalavimų specifikavimo metodų vartotojai susiduria su problemomis:

- Reikalavimų susiejimas su veiklos tikslais
- Pertekliškumo išvengimas reikalavimų specifikacijose
- Reikalavimų ir tikslų atitikimo atsekimas projektavimo metu

1.4. Esamų sprendimų analizė

Atliekant esamų sprendimų analizę, pastebėta, jog reikalavimų specifikavimo procesui skiriama vis daugiau dėmesio ir išlaidų. Reikalavimų specifikavimas yra reikalavimų inžinerijos uždavinys, kurio metu išanalizuoti reikalavimai yra tinkamai dokumentuojami tolesniam naudojimui [8].

Modernizuojant šią sritį vis dažniau pereinama nuo tradicinio nuosekliojo (angl. Waterfall) metodo prie modernių ir efektyvių vystymosi ciklų:

- iteracinė reikalavimų inžinerija (Iterative Requirements Engineering)
- laipsniškoji reikalavimų inžinerija (Incremental Requirements Engineering)
- lygiagreti reikalavimų inžinerija (Parallel Requirements Engineering)
- riboto laiko planavimo reikalavimų inžinerija (Timeboxed Requirements Engineering)

Reikalavimų inžinerija su kiekviena diena vis labiau tobulėja. Pripažįstama daug naudingų reikalavimų uždavinių: veiklos analizė, reikalavimų rinkimas, reikalavimų analizė, reikalavimų specifikavimas ir reikalavimų valdymas. Egzistuoja daugybė reikalavimų analizavimo metodų, kurių dalis (pvz. panaudojimo atvejų analizė) yra plačiai pripažįstama kaip svarbus senesnių metodų patobulinimas. Tokie reikalavimų analizės metodai naudoja sudėtinius abstrakcijos tipus ir lygius, kurie pateikia skirtingas reikalavimų peržiūras skirtingoms auditorijoms [8].

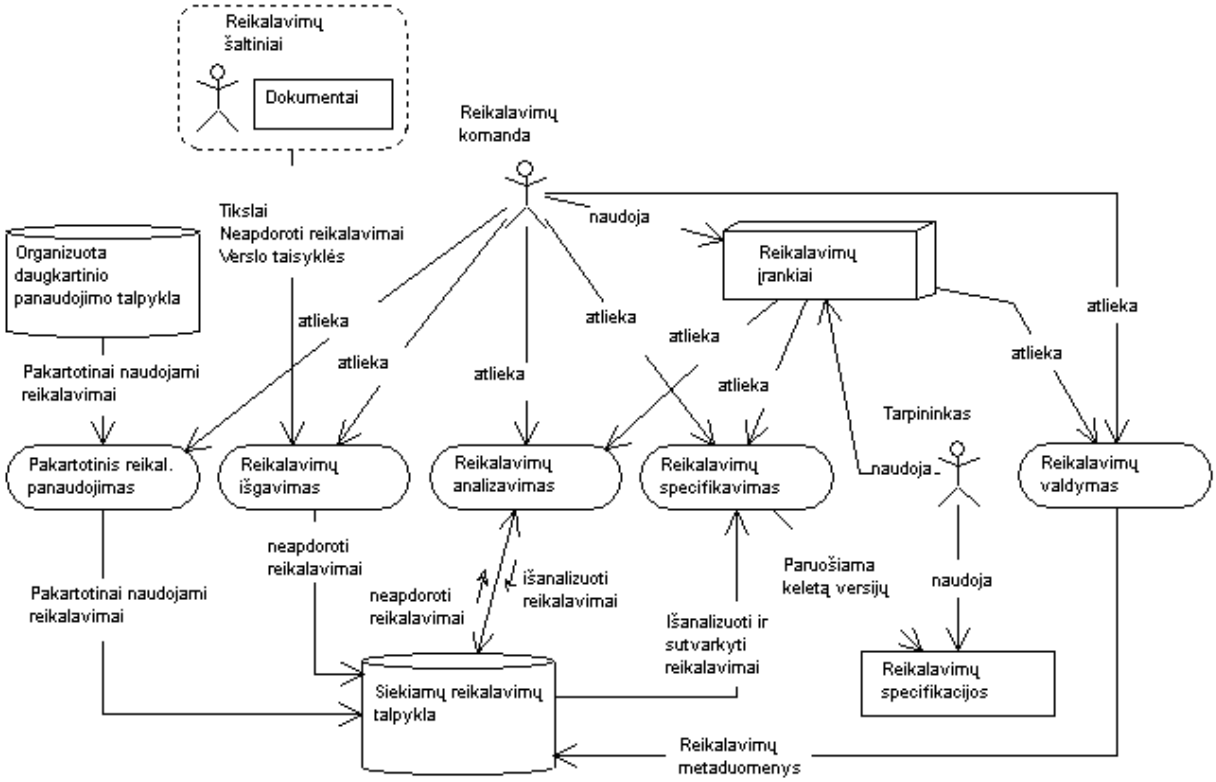
Visi šie reikalavimų analizės metodai daro įtaką reikalavimų specifikavimui paveikdami specifikuojamos informacijos tipus, turinį, formatą bei patį reikalavimų specifikavimo organizavimą. Galimi reikalavimų aprašymo būdai:

- tekstiniai reikalavimai natūralia kalba;
- reikalavimų modeliai grafinio modeliavimo kalba;
- sprendimų lentelės;
- formaliai specifikuoti reikalavimai specifikacijų kalba, ir kt.

Tobulinant reikalavimų specifikavimo procesą, didėja ir reikalavimų valdymo įrankių paklausa. Atsiradus reikalavimų saugykloms (angl. Requirements Repository), prireikia įrankių, kurių dėka būtų galima automatiškai sugeneruoti reikalavimų specifikacijas bei ataskaitas [1]. Šie įrankiai turi toliau išvardytas savybes:

- vartotojui priimtina grafinė sąsaja;
- reikalavimų inžinerijos palaikymas;
- reikalavimų inžinerijai gimingų veiklų palaikymas;
- kolektyvinio darbo galimybė;
- saugumas, išbaigtumas, plečiamumas;
- paskirstyta sistema;
- pakartotinis reikalavimų panaudojimas;

1.2 paveiksle pateikiamas reikalavimų specifikacijos būdas, kuris padeda išspręsti problemas, atsirandančias naudojant modernius reikalavimų valdymo įrankius, paremtus reikalavimų talpyklomis [8].



1.2 pav. Saugykla paremta reikalavimų specifikacija [8]

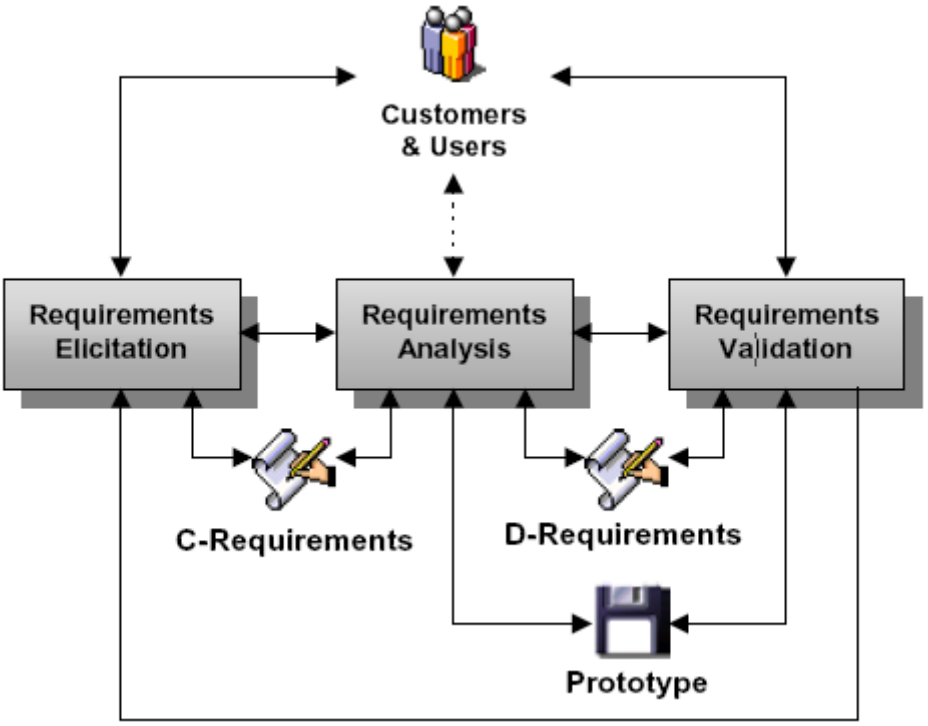
Atliekant reikalavimų specifikavimą būtina užtikrinti, kad specifikacija būtų neprieštaringa, o tuo pačiu būtų pasiekta maksimali jos kokybė. Norint sukurti kokybišką specifikaciją, reikalingas metodas, atitinkantis šiuos reikalavimus [20]:

- reikalavimų informacijos sistemai sudarymo eiga turi būti natūrali, t.y. turi atitikti tradicinį šio proceso eiliškumą;
- metodas turi sumažinti atotrūkį tarp vartotojo ir analitiko;
- modeliavimo procesas turi užtikrinti organizacijos veiklos ar jos išskirto fragmento modelio pilnumą;
- turi būti galimybė atlikti specifikacijos kokybės kontrolę;

1.4.1. Reikalavimų inžinerijos modelis

Reikalavimų inžinerijos modelį galima išskaidyti į tris dalis:

- **Reikalavimų išsiaiškinimas (Requirements Elicitation):** reikalavimai yra išsiaiškinami tarp kliento ir vartotojo naudojant įvairias išsiaiškinimo technikas, pavyzdžiui: interviu, tyrimai, dokumentacijos analizė ir t.t. Šio proceso rezultatas yra pradiniai reikalavimai, įvardijami kaip vartotojo reikalavimai (Customer-Oriented Requirements) arba trumpiau C reikalavimai (C-requirements) [9].
- **Reikalavimų analizė (Requirements Analysis) :** C reikalavimai yra analizuojami tam, kad būtų galima aptikti nesuderinamumus ir identifikuoti trūkstamus reikalavimus, dažniausiai kuriant struktūrizuotus objektinius modelius. Šio proceso metu klientai ir vartotojai gali bendradarbiauti. C reikalavimai yra transformuojami į programinės įrangos reikalavimus, žinomus kaip Developer-Oriented Requirements arba trumpiau D reikalavimai (D-requirements). Kitas šio proceso produktas yra kuriamos sistemos, kuri formaliai detalizuoja D reikalavimus, kurie gali būti automatiškai sugeneruoti [9], prototipas.



1.3 pav. Reikalavimų inžinerijos modelis [8]

- **Reikalavimų patvirtinimas (Requirements Validation):** klientai ir vartotojai privalo patvirtinti reikalavimus ir įvertinti prototipą, dažniausiai naudojamas tam kad išsiaiškinti naujus reikalavimus. Visas procesas kartojamas tol kol reikalavimai yra patvirtinami ir daugiau nebėra iškeliama naujų reikalavimų.

Reikalavimų specifikavimo procese vis didesnis vaidmuo tenka šablonams. Šablonas nusakomas kaip iš anksto parengtas kokio nors rašto projektas, naudojamas konkretiems struktūrizuotiems dokumentams kurti. Šablonų pritaikymas reikalavimų specifikavime leidžia greičiau ir tiksliau išsiaiškinti užsakovo pateiktus reikalavimus ir juos struktūrizuoti bei grupuoti tolesniam apdorojimui [9].

1.4.2. Volere metodo analizė

Yra gana daug šablonų, naudojamų reikalavimų rinkime ir specifikavime. Vienas iš jų yra Volere šablonas. Pirmoji Volere šablono versija buvo išleista 1995 metais [16]. Volere šablonas naudojamas pradiniame sistemos kūrimo etape ir yra kaip pagrindas užregistruoti vartotojų reikalavimus. Šablonas suskirstytas į skyrius pagal reikalavimų tipą. Šablonas padeda sukaupti reikalavimus, kuriuos pateikia vartotojai per interviu arba kurie užregistruoti analizuojamo objekto veiklą reglamentuojančioje dokumentacijoje. Tai atviras šablonas, kurį galima pritaikyti konkrečiam atvejui. Šablono skyrių, kuris netinka nagrinėjamam objektui, galima išmesti arba sukurti naują skyrių, kuris leidžia specifikuoti specifines dalykinės srities charakteristikas [6]. Volere šablono naudojimas mokymo tikslams yra nemokamas. Kitu atveju reikalaujama vienkartinės paramos už kiekvieną projektą. Volere šablonas apima šiuos reikalavimų tipus [12,17]:

- funkciniai reikalavimai (Functional requirements);
- nefunkciniai reikalavimai (Nonfunctional requirements);
- projekto apribojimai (Project constraints);
- projekto varovai (Project drivers);
- projekto išeiga (Project issues);
- testavimo reikalavimai (Testing requirements).

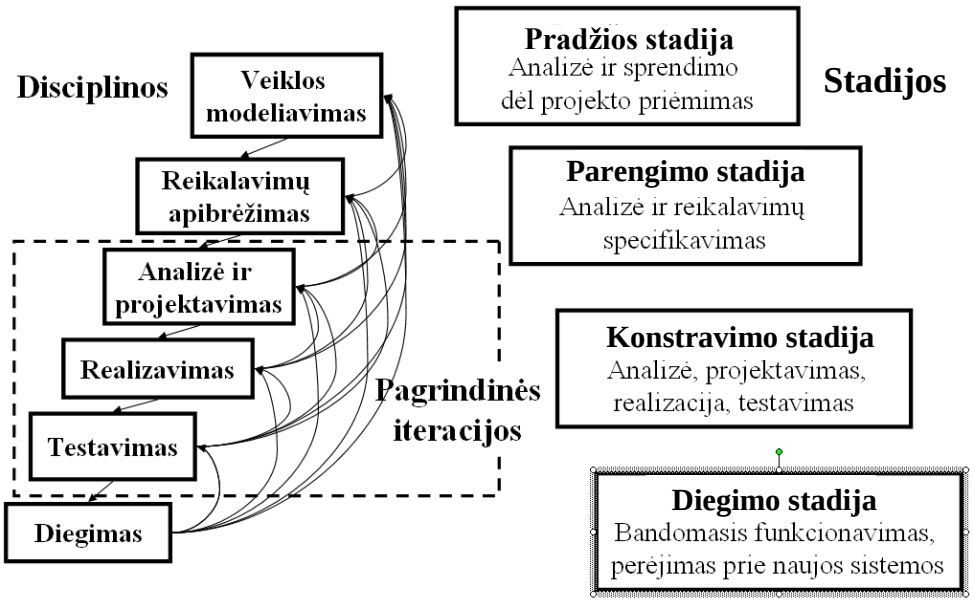
Naudojant Volere šabloną reikalavimams aprašyti, vadovaujamasi susisteminto proceso apribojimais, nusakančiais, kokio etapo metu ir kokie reikalavimai turi būti aprašyti. Kai kuriuose proceso etapuose naudojamas specializuotas apklausos lapas, padedantis surinkti reikalavimus, juos analizuoti ir apibendrinti [6].

1.4.3. RUP metodo analizė

Rational Unified Proces (RUP) [13] yra panaudojimo atvejų valdomas, architektūra grindžiamas, iteracinis, vykdomas palaipsniui, riziką mažinantis projektavimo procesas

[14,18]. Unifikuotas procesas siejamas su praktikoje taip pat plačiai paplitusia unifikuota modeliavimo kalba *Unified Modeling Language* (UML) [3].

RUP struktūra pateikiama 1.4 paveiksle. Kūrimo procesas susideda iš makroproceso, kurį sudaro stadijos, ir mikroproceso, kurį sudaro disciplinos [2]. Visos sistemos kūrimas vyksta kaip makroprocesas, atskiros sistemos dalys sukuriamos mikroprocesų vykdymo metu. Nors tam tikros disciplinos būdingesnės tam tikroms stadijoms, kiekvienos stadijos metu galima taikyti bet kurią discipliną, jei ji būtina. Pavyzdžiui, jau veiklos modeliavimo metu galima realizuoti sistemos dalies prototipą, jei to reikia projekto sprendimui priimti. Ir atvirkščiai, realizavimo metu galima papildomai modeliuoti veiklą tam, kad būtų galima geriau atlikti realizaciją.



.

1.4 pav. Unifikuotas sistemos kūrimo proceso gyvavimo ciklas

Pradžios stadija. Joje atliekama problemos analizė, kurios pabaigoje suformuojamos projekto užduotys ir priimamas sprendimas dėl projekto reikalingumo. Šios stadijos metu modeliuojami tiriamos veiklos procesai, analizuojama sistemos aplinka, galimos realizavimo technologijos, prototipai, sudaromas projekto planas, analizuojamos kainos, rizika, jos įveikimo metodai, formuojama projekto komanda. Stadijos pabaigoje, jei projektas bus vykdomas, sudaroma techninė užduotis.

Parengimo stadija. Jos metu detaliai analizuojami ir specifikuojami pradžios stadijoje nustatyti funkciniai ir nefunkciniai reikalavimai, sudaromi sistemos struktūros ir veikimo modeliai, tiksliai apibrėžiamos sistemos sąsajos ir sąveikos su kitomis sistemomis. Stadijos pabaigoje sukuriama reikalavimų specifikacija, kuri apibrėžia, kaip turi veikti ir atrodyti

siekiamas produktas nepriklausomai nuo to, kaip jis bus realizuotas ir kaip užtikrins siekiamas kokybines charakteristikas („ką“ darys sistema, o ne „kaip“).

Konstravimo stadija. Tai pagrindinė sistemos kūrimo stadija. Joje galima išskirti projektavimo, realizavimo ir testavimo dalis. Architektūrinio projektavimo metu identifikuojami sistemos posistemiai, komponentai, jų ryšiai. Detalaus projektavimo metu – atskiri komponentai. Realizacijos metu komponentai suprogramuojami, atliekamas kiekvieno elemento testavimas. Jungiant komponentus į sistemą, didelę veiklos dalį sudaro jų integravimas ir sistemos integravimo testavimas.

Diegimo stadija prasideda tada, kai projekto rezultatas perduodamas užsakovui. Šiame etape produktas bandomas realioje aplinkoje, taisomos jo gamybos klaidos.

Visas šis procesas vyksta persidengiančiomis iteracijomis, kiekvienoje iteracijoje realizuojama tam tikra reikalavimų dalis. Realizuojami reikalavimai pasirenkami pagal prioritetus. RUP apibrėžiamas kaip panaudojimo atvejų valdomas, architektūrinis, laipsniškasis (angl. *incremental*), iteracinis, riziką mažinantis procesas:

– RUP yra valdomas panaudojimo atvejų, nes panaudojimo atvejais grindžiamos visos jo veiklos ir stadijos: reikalavimai specifikuojami panaudojimo atvejais; analizė atliekama pagal panaudojimo atvejus; projektas konstruojamas analizuojant panaudojimo atvejus; realizacija testuojama pagal panaudojimo atvejus;

– RUP grindžiamas architektūra, jis suderina vartotojui orientuotą ir architektūrinį projektavimą;

– RUP yra iteracinis, vykdomas iteracijomis;

– RUP yra laipsniškasis, nes kiekvienos iteracijos metu sukuriamas naujas, papildomas funkcionalumas, įvertinant ankstesnių iteracijų patirtį ir pakartotinai naudojant jų rezultatus. Laipsniškasis kūrimas pradedamas žinant visus reikalavimus, tačiau jų apibrėžimas nėra detalus. Kiekvienas reikalavimų rinkinys sudaro išbaigtą sistemos dalį, kurią galima naudoti;

– RUP mažina riziką ne tik todėl, kad identifikuoja riziką ir numato būdus su ja kovoti. Pagal RUP, parengimo stadijoje identifikuojami pradiniai panaudojimo atvejai, tačiau analizė ir projektavimas neatliekamas visai sričiai, kuri gali būti labai didelė ir neaiškių ribų. Kiekvienoje iteracijoje atliekamas detalus pasirinktos funkcionalumo dalies (posistemio, uždavinio, komponento, funkcijos) projektavimas ir realizavimas. Kiekvienoje iteracijoje įgyjamos naujos žinios ir kūrimas vyksta kaskart su didesne patirtimi. Net ir nesėkmingo projekto atveju nuostolių rizika yra mažesnė.

RUP buvo sukurtas geriausių praktikų pagrindu:

– ***Kurti PĮ iteracijomis***. Neįmanoma iš karto apibrėžti visą sudėtingos šiuolaikinės sistemos problemą, suprojektuoti sprendimą, sukurti PĮ ir pabaigoje ištestuoti. Iteracinis kūrimas leidžia palaipsniui gilinti problemos supratimą ir „auginti“ sprendimą.

– ***Valdyti reikalavimus*** – sistemingai surinkti, keisti ir komunikuoti. Pagrindinis instrumentas reikalavimams surinkti yra panaudojimo atvejai.

– ***Taikyti komponentinę architektūrą***. Ši geroji praktika leidžia prisiderinti prie pokyčių ir leidžia efektyvų pakartotinį naudojimą, nes sistemingai sukuriama ir įtvirtinama gera architektūra.

– ***Vaizdinis PĮ modeliavimas*** naudojant grafinius įrankius ir vaizdinę kalbą UML.

– ***PĮ kokybės tikrinimas*** viso proceso metu leidžia užtikrinti produkto ir proceso kokybę, kokybės tikrinimas yra proceso dalis.

– ***PĮ pokyčių valdymas*** leidžia įsitikinti, kad kiekvienas pokytis yra priimtinas. Pokyčiai yra stebimi ir atsekami.

2005 metais IBM *Rational* (anksčiau buvusi *Rational Software Corporation*) peržiūrėjo gerąsias praktikas ir suformulavo jas iš naujo. RUP tapo nebe procesu, bet karkasu, leidžiančiu kurti proceso variantus įvairiems projektų atvejams: mažiems, vidutiniams ir dideliems, surinkimui iš komponentų, tinklo paslaugų (angl. *Web Services*) kūrimui. Gerosios praktikos tapo abstraktesnės.

Adaptuoti procesą. Vienas procesas negali tikti visiems, todėl jį galima pritaikyti komandos dydžiui ir pasiskirstymui, sistemos sudėtingumui, atitikimo lygiui. Projekto pradžioje, kol neaiškumo yra daugiau, reikia leisti daugiau kūrybiškumo. Vėlesnėse iteracijose reikia daugiau kontrolės, pavyzdžiui, pokyčių valdymui, defektų prevencijai. Mažiems projektams procesas gali būti lengvesnis. Kuo daugiau dalyvių, kuo sudėtingesnė technologija, kuo griežtesni standartų reikalavimai, tuo procesas labiau disciplinuotas.

Proceso adaptavimas taip pat reiškia nuolatinį jo tobulinimą. Įvertinimas turi būti atliekamas kiekvienos iteracijos ir kiekvieno projekto pabaigoje ir patirtis taikoma procesui gerinti. Projekto įvertinimai taip pat turi būti pritaikyti: proceso pradžioje jie abstraktesni, pabaigoje – tikslesni ir griežtesni.

Balansuoti konkuruojančių vartotojų prioritetus. Skirtingi vartotojai – galutiniai vartotojai, vadybininkai, išoriniai pirkėjai ir t. t. turi skirtingus poreikius. Jų balansavimas yra sudėtingas uždavinys. Pavyzdžiui, vieni nori, kad sistema atliktų visas reikiamas funkcijas,

kiti – kad ji kuo mažiau kainuotų. Poreikiai dažnai konfliktuoja. Todėl svarbu suprasti veiklos procesus ir juos susieti su programinės įrangos galimybėmis. Turint šį supratimą, galima efektyviai prioritetizuoti vartotojų poreikius. Projekto eigoje prioritetai gali keistis. Projekto komanda turi priimti šį faktą, kad poreikiai projektavimo metu keisis, kadangi keičiasi veikla. Norint suprasti, kaip galima patenkinti vartotojų poreikius, reikia išsiaiškinti, kokie ištekliai galimi. Pavyzdžiui, liktinės sistemos, pakartotinio naudojimo komponentai, šablonai.

Bendradarbiauti komandomis. Šios praktikos tikslas yra integruoti veiklos, PĮ kūrimo ir operacines komandas. Kadangi IS tampa vis labiau kritinės vykdant veiklą, šių vartotojų bendradarbiavimas yra neįkainojamas. Norint padidinti komandos našumą ir produkto kokybę, žmonės turi būti motyvuoti komunikuoti ir glaudžiai bendradarbiauti, taip pat mokytis ir įgyti naujų įgūdžių iš bendradarbių. Individų motyvavimas daryti geriausia, ką galima yra svarbus pirmasis žingsnis. Šis motyvavimas sustiprėja, kai jie pasijunta atsakingi už galutinį rezultatą. Kiekvienas narys turi suprasti projekto misiją ir viziją. Bendradarbiauti padeda atitinkama aplinka, programiniai įrankiai PĮ projektuoti (UML CASE įrankiai), elektroniniai projekto „kambariai“, konfigūravimo, pokyčių valdymo, testavimo įrankiai.

Iteracijomis demonstruoti vertę. Gera idėja yra pristatyti veikiančią PĮ ankstyvoje ciklo stadijoje, pademonstruoti naujos IS vertę ir gauti grįžtamąjį ryšį. Jie tikriausiai pastebės, kad kai kurie reikalavimai buvo pamiršti arba neteisingai interpretuoti. Efektyvi kūrimo metodologija turi susitaikyti su tuo, kad pokyčiai yra neišvengiami. Tradiciniai kūrėjai nemėgsta kintančių reikalavimų. RUP kviečia pasinaudoti pokyčiais ir juos valdyti. Nesvarbu, kaip vėlai gyvavimo cikle atsiranda pokyčiai, jei jie didina IS vertę, reikia juos priimti teigiamai. Taip pat tikslinga kuo anksčiau pastebėti riziką, o tai pasiekiama nuolat atliekant jos įvertinimą.

Kelti abstrakcijos lygį. Darbas žemu abstrakcijos lygiu ir neefektyviais įrankiais yra našumo kliūtis. Šiais laikais jau negalima efektyviai sukurti netrivialios sistemos, rašant kodą įprastomis programavimo kalbomis ir kuriant duomenų bazės lenteles tiesiai DBVS duomenų apibrėžimo kalba. Modeliais grindžiamas kūrimas (angl. *Model Driven Development*) leidžia padidinti našumą naudojant aukštesnio lygio modelius, įrankius ir kalbas. Automatinis ar pusiau automatinis kodo generavimas ir testavimas leidžia sumažinti nekūrybiško darbo dalį, išvengti klaidų ir susikoncentruoti į sistemos architektūrą ir kokybę. Modeliais grindžiamą kūrimą palaiko unifikuota modeliavimo kalba UML ir atitinkami CASE įrankiai. Aukšto abstrakcijos lygio architektūra leidžia lengviau valdyti sistemos sudėtingumą.

Sutelkti dėmesį į kokybę. Už kokybę yra atsakinga visa komanda, ne tik testuotojai. Todėl testavimas ir validavimas vykdomas viso projektavimo metu. Kiekviena disciplina turi

formalias ar neformalias produktų peržiūras. Analitikai turi užtikrinti, kad reikalavimai būtų testuojami. Kūrėjai yra atsakingi už jų kodo testavimą. Vadybininkai turi užtikrinti, kad testavimo planai būtų parengti. Testuotojai yra kokybės ekspertai. Jie padeda kitiems komandos nariams suprasti, kas yra sistemos kokybė ir atsakingi už funkcinį, sistemos ir veikimo lygio testavimą. Svarbiausios sistemos savybės realizuojamos projekto pradžioje. Projekto pabaigoje šios savybės būna jau seniai veikiančios ir gerai ištestuotos. Gera kokybė yra vienas svarbiausių RUP pranašumų. Testavimo aplinkos kūrimas vyksta kartu su sistemos kūrimu. Laipsniškojo testavimo priešingybė yra visos sistemos integravimo testavimas, atliekamas kūrimo pabaigoje. Taip paprastai daroma taikant nuoseklųjį gyvavimo ciklą.

Egzistuoja RUP variantai mažiems, dideliems ir labai dideliems projektams, kūrimui iš komercinių *Commercial Off-The-Shelf* (COTS) komponentų, tinklo paslaugų (angl. *Web Service*) architektūrai. RUP galima pritaikyti beveik kiekvienam projektui. Jis turi tik problemą dėl kontraktų (negalima sudaryti fiksuotų kontraktų dėl neaiškios darbų apimties), tačiau ją galima spręsti vykdant projektus dalimis, kai po kiekvienos iteracijos būtų priimami sprendimai dėl tolesnio projekto vykdymo ir dėl jo apimties ir kainos. Atitinkamai turėtų būti paprasčiau organizuojamos šių dalinių projektų kontraktų procedūros, kad jos netaptų projektų vykdymą trukdančiu veiksniu.

1.4.4. Tropos metodo analizė

Tropos yra nauja, į agentus orientuota (angl. *agent-oriented*) programinės įrangos inžinerijos metodologija, charakterizuojama trimis aspektais [4]:

- Pirma, daugiausia dėmesio skiriama veikloms, kurios specifikuoja nusistovėjusius reikalavimus, suvokiama kaip numatytoji sistema turi sąveikauti su organizacijos tikslais.
- Antra, siejasi su visomis sistemos reikalavimų analizės, sistemos projektavimo bei kūrimo fazėmis būdais, kurie pagrįsti bendra notacija, kurios elementai yra aktoriai, tikslai, potiksliai, planai, ištekliai ir priklausomybės.
- Trečia, metodologija, kuri remiasi būsimos sistemos modelio kūrimo idėja, yra praplečiama nuo koncepcinio lygmens iki vykdomų artefaktų.

Vienas iš Tropos metodologijos privalumų yra tas, kad yra ne tik keliami klausimai *kokia* ir *kaip*, bet ir *kodėl* programinė įranga yra vystoma. Tropos metodologijoje yra numatyta palaikyti visas analizės ir projektavimo veiklas programinės įrangos kūrimo procese [4,10].

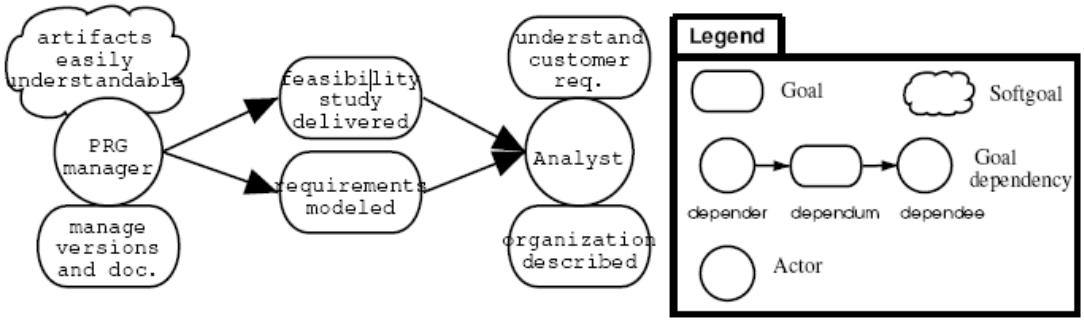
Tropos palaiko penkias programinės įrangos plėtojimo fazes:

- ankstyvieji reikalavimai (early requirements);
- vėlyvieji reikalavimai(late requirements);
- architektūrinis projektavimas (architectural design);
- detalus projektavimas(detailed design);
- vykdymas (implementation).

Plėtojimo fazės

Reikalavimų analizė yra pradinė fazė daugumoje programinės įrangos inžinerijos metodologijų. Svarbiausias tikslas reikalavimų analizėje Tropos metodu yra tas, kad kuriama sistema būtų sukurta pagal gerai surinktus funkciniais ir nefunkciniais reikalavimus.

Reikalavimų analizė Tropos metode dalijasi į dvi fazes: ankstyvųjų ir vėlyvųjų reikalavimų analizę. Abi dalys apima tiek konceptualius tiek metodologinius požiūrius. Pagrindinė idėja yra ta, kad ankstyvųjų reikalavimų analizė yra naudojama vėlyvųjų reikalavimų analizėje. Ankstyvųjų reikalavimų analizė apima problemos supratimą studijuojant egzistuojančią organizaciją. Tarpiniai elementai yra modeliuojami kaip tikslai ir tikslų priklausomybės tarp aktorių ir analizuojami kaip tam tikros formos iš tikslų analizės. Šioje fazėje organizacijos modelis apima tiesiogiai susijusius aktorius ir jų priklausomybes tikslams ir potiksliams. Modelis yra vaizduojamas aktorių diagramomis aprašant priklausomybes tarp aktorių ir tikslų diagramų, analizuojant tikslų atitikimus. Vėlyvųjų reikalavimų analizėje konceptualus modelis yra praplečiamas įtraukiant naujus aktorius, kurie pristato sistemą, ir priklausomybes su kitais aktoriais iš tos aplinkos. Šios priklausomybės apibrėžia visus funkcinius ir nefunkcinius reikalavimus, tam kad sistema būtų sukurta (system-to-be)[4].



1.5 pav. Aktorių diagrama [4]

Architektūrinio ir detalaus projektavimo fazės yra sukoncentruotos ties sistemos specifikacija, remiantis reikalavimų rezultatais iš prieš tai aprašytų dviejų fazių. Architektūrinis projektavimas aprašo sistemos visą architektūrą išreikštą per sub-sistemas,

sujungtas per duomenų bei valdymo srautus. Sub-sistemos vaizduojamos modelyje kaip aktoriai, o duomenų ir valdymo tarpusavio sąryšiai vaizduojami kaip priklausomybės. Detalaus projektavimo tikslas yra specifikuoti sąveikas.

Vykdymo (angl. *Implementation*) veikla leidžia palaipsniui, natūraliu būdu, detalizuoti projektavimo specifikacijas tarp vykdymo platformos konstravimo ir detalaus projektavimo notacijos [10].

1.4.5. Kaos metodo analizė

Tikslų aiškinimasis ir manipuliavimas jais yra natūrali reikalavimų inžinerijos dalis: reikalavimai pagal tipus realizuoja tikslus, kurie turi būti pasiekti. Ankstesnės reikalavimų inžinerijos technologijos orientavosi ties esybėmis ir veiklomis. Į tikslus orientuota metodologija pirmenybę teikia iš anksto suplanuotiems (tyčiniams) – tikslams – vystant programinės įrangos sistemas, su esybėmis ir veiklomis, kurios dabar remiasi tikslais. Tikslų apsvarstymas priveda prie sistemos projektavimo tyrinėjimo alternatyvų – "Kiekvienas gali išdėstyti tikslą, neturėdamas tikslių specifikacijų, kaip jis turi būti pasiektas" – tai padeda projektuotojui kurti geresnes sistemas.

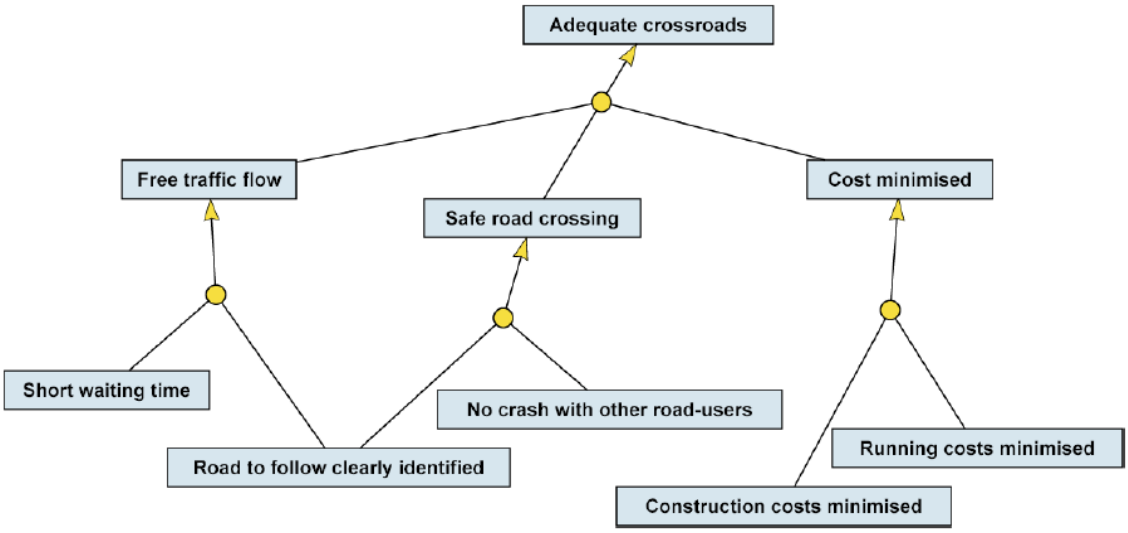
Galbūt vienas iš geriausių į tikslus orientuotų metodų yra KAOS metodas. KAOS yra tikslais grindžiama reikalavimų inžinerijos metodologija, kuri įgalina analitikus kurti reikalavimų modelius bei išgauti reikalavimų dokumentus iš KAOS modelių[11,15]. KAOS struktūra lengvina tolimesnių tikslų nustatymą – ir tai kas būtina, objektai, agentai, ir sistemos veiksmai. KAOS palengvina patį reikalavimų išgavimo ir modeliavimo procesą. Tikslai yra naudojami tikėtinų sistemos savybių aprašymui. Tikslai yra analizuojami ir geriau suprantami nustatant aukštesnio lygmens tikslus – tikslai, kurie paaiškins, kodėl šis tikslas yra pageidaujamas, jie yra aiškiau apibrėžti, detalizuojant juos iki subtikslų. Nustatant tarpusavio sąryšius tarp tikslų, yra sukuriamas tikslų grafas. Tikslų grafas yra semantinis tinklas iš IR/ARBA/XARBA (angl. *AND/OR/XOR*) sąryšių tarp tikslų, kur tikslai konjunkcijomis arba disjunkcijomis išskaidomi į potikslius. Konfliktiniai sąryšiai tarp tikslų gali būti aptikti.

Reikalavimų analizė KAOS metode išsiskaido į tris fazes [7]:

- informacijos rinkimas, kuris naudojamas kaip vedlys tikslams pasiekti;
- modeliavimas;
- specifikavimas ir sintezė.

1.6 paveiksle pavaizduota KAOS metodo tikslų diagrama. Stačiakampiai reiškia tikslus. Apskritimai reiškia pagrindinių tikslų papildymą žemesnių potikslių rinkiniu. Tam tikri tikslai buvo įvesti konkrečioms struktūroms, pavyzdžiui, „kainų sumažinimas" (angl. *Costs minimised*) arba „tinkamas kelių susikirtimas" (angl. *Adequate crossroads*). Tikslas gali

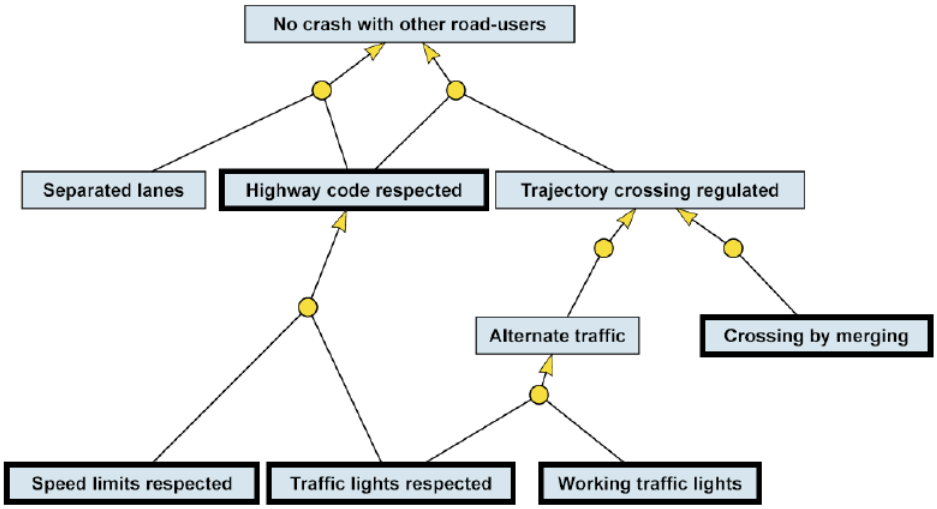
prisidėti prie daugiau kaip vieno tėvinio tikslo; pavyzdžiui, tikslas „aiškus maršrutas“ (angl. *Road to follow clearly identified)* įeina į tikslą „laisvas eismo srautas“ (angl. *Free traffic flow)*, kadangi vairuotojas negali sutrikti, kada pasikeičia kelio kryptis, jei kelio ženklai bus aiškūs; jis taip pat įeina į tikslą „saugus kelio kirtimas“ (angl. *Safe road crossing),* kadangi vairuotojas, dvejojantis apie jo/jos kelią ar ieškantis kelio ženklų, didina susidūrimo pavojų.



1.6 pav. Aktorių diagrama [4]

Alternatyvos apibūdina skirtingus būdus tikslams pasiekti. Jie gali būti motyvuoti skirtingos politikos ar projektavimo sprendimų, pvz., panaudota technologija ar sprendimo kokybė, kurią jie siūlo (patikimumas). Alternatyvos yra nebūtinai išskirtinės, tai reiškia, kad kiekvienas gali nuspręsti išvystyti sistemą naudojant įvairias alternatyvas, pavyzdžiui, pagerinti sistemos patikimumą. Alternatyvos neapibrėžia padalijimo, kadangi skirtingos alternatyvos gali turėti bendrus potikslius.

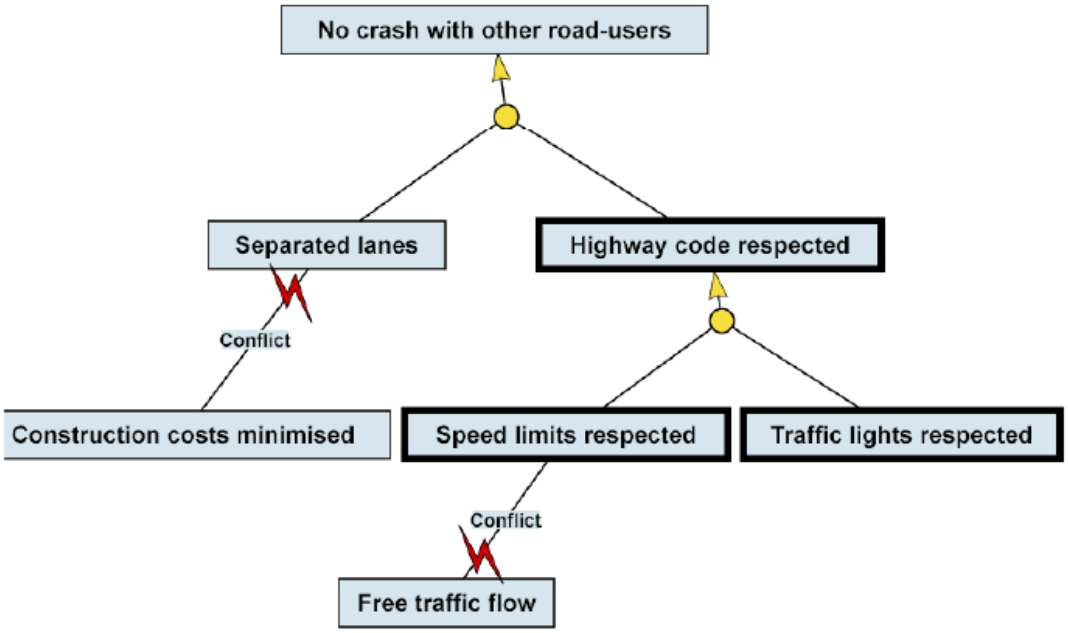
1.7 paveiksle pavaizduotos kai kurios alternatyvos, praplečiančios tikslą „Nėra avarijos su kitais vairuotojais“ (angl. *No crash with other road-users).*



1.7 pav. Alternatyvos [4]

Konfliktai. Tikslai, išreikšti skirtingų tarpininkų, gali būti prieštaringi. KAOS konfliktas atsiranda tarp dviejų tikslų, jei gali būti tokios sistemos būsenos, kuriose abu tikslai yra logiškai prieštaraujantys. Svarbu identifikuoti prieštaringus tikslus kaip galima anksčiau gyvavimo cikle, kad būtų sumažinti sistemos atmetimo pavojai.

Tikslas „Atskirtos eismo juostos“ (angl. *Separated lanes)* prieštarauja tikslui „Kūrimo kainų mažinimas“ (angl. *Construction costs minimised).* Tikslas „Greičio ribojimas“ (angl. *Speed limits respected*) gali prieštarauti tikslui „laisvas eismo srautas“ (angl. *Free traffic flow),* kadangi jis gali sumažinti eismo srautą. Konfliktai grafiškai vaizduojami tikslo grafe, jungiant prieštaringus tikslus (1.8 paveikslas).



1.8 pav. Konfliktai [10]

1.4.6. Tikslų ir reikalavimų modeliavimo galimybių UML CASE įrankyje MagicDraw UML analizė

Dauguma UML CASE įrankių neturi specialių priemonių specifikuoti tikslus ir susieti juos su reikalavimais. Taip pat nėra galimybių specifikuoti reikalavimus – jie tik vaizduojami panaudojimo atvejais, kurių turinys be specifikacijų duoda mažai informacijos. Tam reikia naudoti atskirus reikalavimų specifikavimo įrankius, pavyzdžiui, DOORS ar *RequisitePro*. UML CASE įrankyje *Magic Draw* yra galimybė sugeneruoti RUP reikalavimų šabloną, tačiau jo pildymo būdas yra sudėtingas, specifikacija perteklinė, užima labai daug vietos ir neturi ryšio su tikslais.

Tačiau yra perspektyvių priemonių, kurias būtų galima panaudoti:

- UML profilių ir dalykinių sričių kalbų DSL (angl. *Domain Specific Language*) kūrimo mechanizmas, kuris gali padėti sukurti tikslų modeliavimo priemones ir reikalavimų specifikavimo šablonus;
- Sistemų modeliavimo kalbos *System Modeling Language* (*SysML*) , kuri yra UML profilis, galimybės reikalavimams specifikuoti. *SysML* profilį galima panaudoti kaip prototipą, įgyvendinant *Volere* šablono elementus siekiamoje reikalavimų specifikacijoje.

1.4.7. Tikslais grindžiamų reikalavimų specifikavimo metodų analizės apibendrinimas

Atlikus reikalavimų specifikavimo metodų analizę buvo atskleistos atskirų metodų savybės. *Volere* ir RUP reikalavimų specifikavimo metodai mažai dėmesio skiria tikslams. *Tropos* ir ypač KAOS metodas yra orientuoti į tikslus. Norint įtraukti tikslų analizę į praktinį projektavimo procesą, tikslinga panaudoti įvairių metodų elementus. RUP procesas taikomas labai plačiai, todėl tikslinga kaip pagrindinį procesą naudoti RUP. Siekiamas reikalavimų šablonas turėtų išskaidyti reikalavimus iki atominių, kaip *Volere*, tačiau įtraukti ir panaudojimo atvejų scenarijus kaip RUP. *Tropos* metodas nagrinėja tikslus, tačiau iš esmės jis labai panašus į RUP. Tikslų modeliavimui turėtų būti taikoma KAOS metodologija. Galiausiai, tikslai turėtų būti siejami su kiekvienu atominiu reikalavimu. Metodų palyginimas pateikiamas 1.1 lentelėje.

1.1 lentelė Analizuotų metodų palyginimo lentelė

Palyginimo kriterijai	VOLERE	RUP	TROPOS	KAOS
Ar nagrinėja tikslus?	–	–	+–	+
Ar nagrinėja tikslu priklausomybes?	–	–	–	+
Ar išveda reikalavimus iš tikslu?	–	–	–	+–
Ar išskaido reikalavimus iki atominių vienetų?	+	–	–	–
Ar turi šablonus reikalavimams specifikuoti?	+	+	–	–

1.5. Siekiamas sprendimas

Analizės metu buvo nustatytos maksimalios savybės, kuriomis turėtų pasižymėti geras reikalavimų specifikavimo metodas ir atitinkamas įrankis. Šiame darbe dėl apimties ir laiko ribojimų bus sudarytas metodas, kuris apims pagrindinę savybių dalį ir leis:

- padaryti tikslų modeliavimą praktiškai naudojamą ir susietą su praktikoje paplitusiais reikalavimų šablonais;
- sudaryti tikslų ir reikalavimų trasavimo metodiką.

Šis darbas neapims tikslų konfliktų modeliavimo ir scenarijų specifikavimo, kurie galėtų būti kitų tiriamųjų darbų, tęsiančių šį tyrimą, uždaviniai.

1.6. Analizės išvados

- Reikalavimų specifikavimo galimybių analizė UML CASE įrankiuose rodo, kad trūksta gerų priemonių reikalavimams specifikuoti susiejant juos su tikslais, bei atsekti projektavimo procese.
- Tikslų modeliavimo metodų analizė parodė, kad Tropos ir Kaos idėjas galima pritaikyti išplečiant praktikoje taikomus projektavimo procesus, vykdomus UML CASE įrankiais.
- Norint išlaikyti projekto kokybę ir produktyvumą, reikia sudaryti efektyvų reikalavimų specifikavimo metodą, kuris sietųsi su tikslais ir tik tada naudoti reikalavimų specifikavimo priemones sekamumo supaprastinimui ir palengvinimui.
- UML CASE įrankio Magic Draw analizė parodė, kad tyrimo tikslams pasiekti galima panaudoti šio įrankio UML profilius bei DSL kūrimo galimybes.

  Šių galimybių pritaikymas bus tiriamas tolesniame darbo etape.

2. Tikslais grindžiamų reikalavimų sekimo metodikos projektas

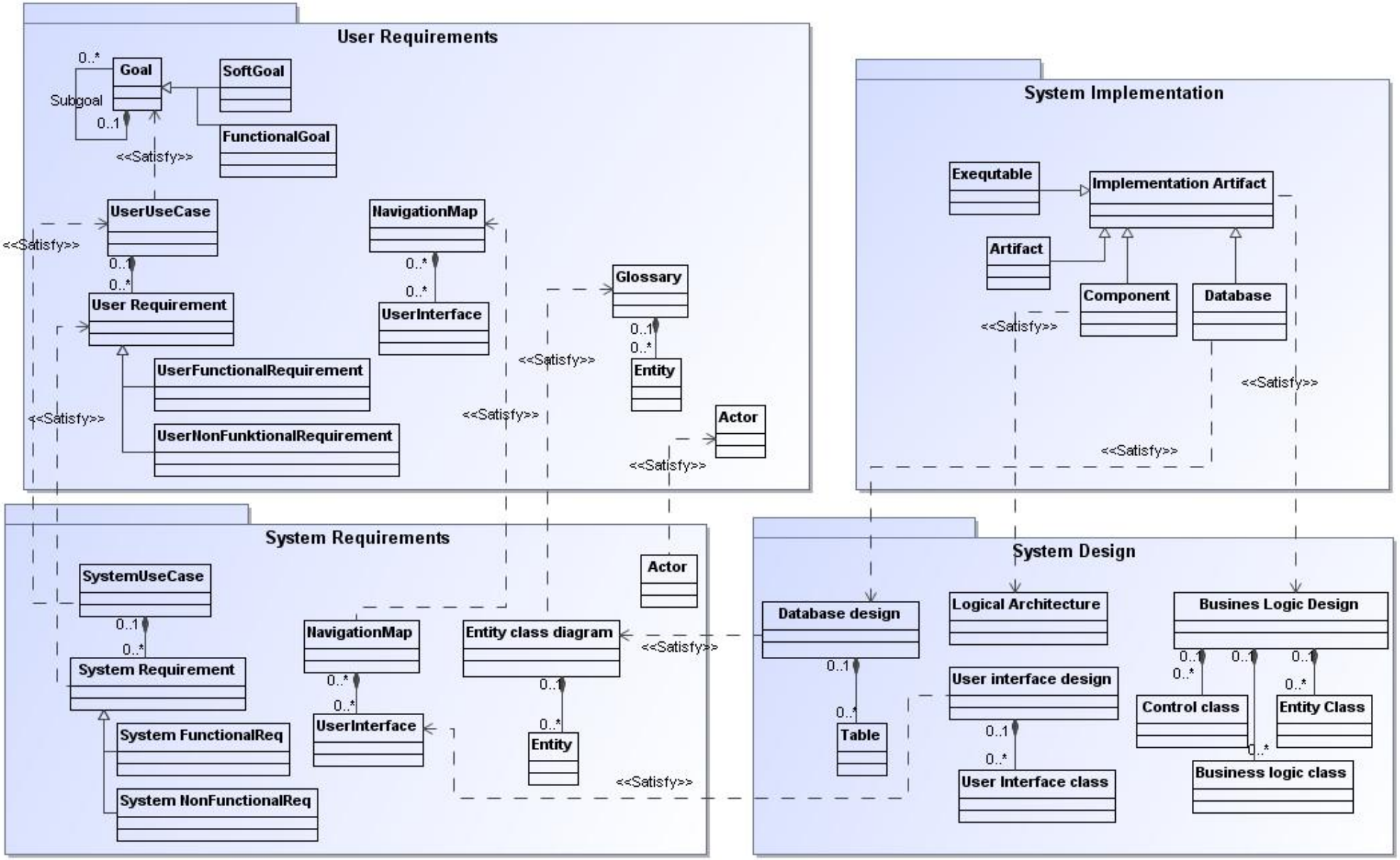
2.1. Projekto tikslas

Projekto tikslas – papildyti esamą arba sudaryti naują, išsamų, tikslais grindžiamą ir pritaikytą plačiai naudojamam CASE įrankiui reikalavimų specifikavimo šabloną. Reikia sukurti reikalavimų specifikacijos metamodelį, kuris padėtų suderinti tikslus, reikalavimus ir išvengti pertekliškumo. Konkrečiai šablonas bus sudaromas naudojant MagicDraw UML 15.5 CASE įrankį. Iš pradžių bus sukurti bendri šablono elementai, kuriais bus galima specifikuoti reikalavimus. Naudojant šiuos elementus, bus pateikiami reikalavimų specifikacijų pavyzdžiai.

2.2. Reikalavimų specifikacijos

2.2.1. Reikalavimų metamodelis

Reikalavimų specifikavimo metamodelis susideda iš 4 stambių dalių. Pirma dalis yra „Vartotojo reikalavimai“ (angl. *User Requirements*). Čia yra identifikuojami tikslai, braižomos tikslų diagramos. Iš tikslų yra pereinama prie reikalavimų. Aprašomi funkciniai ir nefunkciniai vartotojo reikalavimai. Antroji dalis yra „Sistemos reikalavimai“ (angl. *System Requirements*). Čia yra detalizuojami reikalavimai, aprašomi funkciniai ir nefunkciniai sistemos reikalavimai, aprašomos sistemos esybės. Trečioji dalis yra „Sistemos projektas“ (angl. *System Design*). Čia atsiranda duomenų bazės modelis, pateikiama sistemos loginė architektūra. Ketvirtoji dalis yra „Sistemos realizacija“ (angl. *System Implementation*). Čia vaizduojamos komponentų diagramos, realizavimo artefaktai, fizinė duomenų bazės schema. Metamodelyje matosi, kaip vienos dalies elementai siejasi su kitos dalies elementais. Reikalavimų specifikacijos metamodelis pateiktas `**2.1** paveiksle.



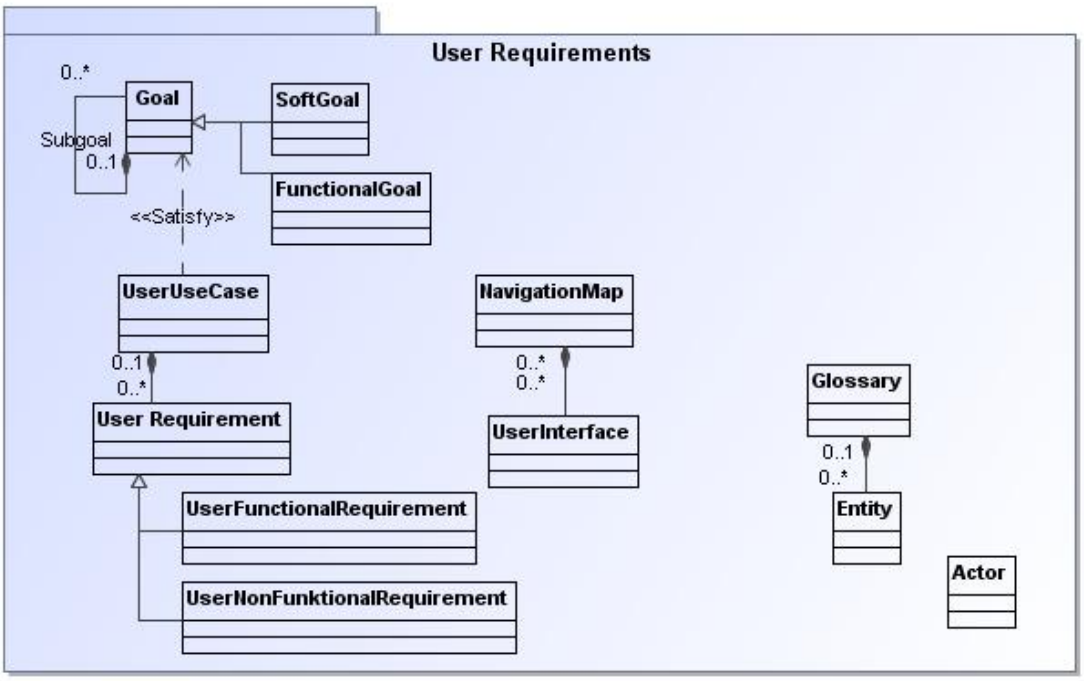
`2.1 pav. Reikalavimų specifikacijos metamodelis

2.2.2. Vartotojo reikalavimai (angl. *User Requirements*)

Metamodelio dalyje „Vartotojo reikalavimai“ (angl. *User Requirements*) yra apibrėžiami:

- kompiuterizuojamos veiklos tikslai ir veiklos procesai;
- vartotojų poreikiai (funkciniai reikalavimai);
- vartotojų tipai, aibės ir savybės, organizacinė struktūra;
- nefunkciniai reikalavimai (pajėgumai, saugumas, veikimo charakteristikos, palaikanti aplinka ir t. t.);
- sistemos veikimo ir vartotojų sąveikos scenarijai (panaudojimo atvejų aprašymai).

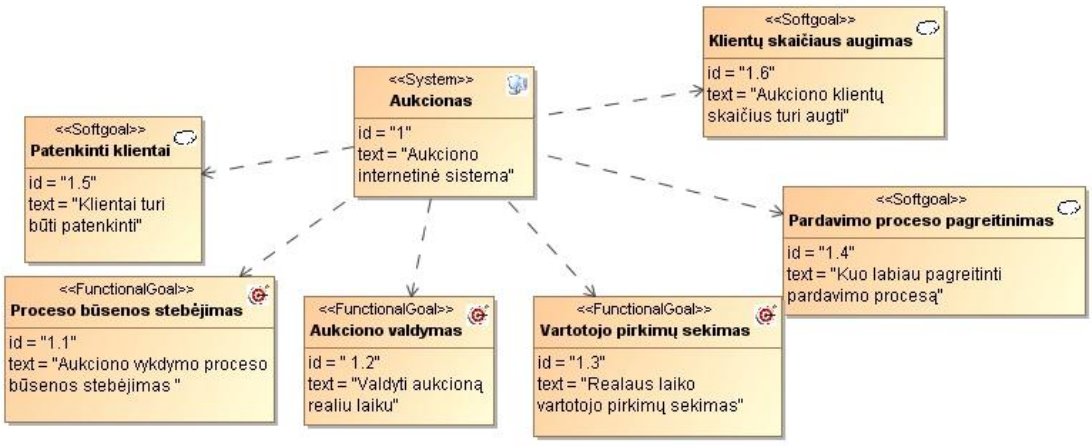
Tikslai gali būti funkciniai (angl. *Functional Goal*) arba nefunciniai, kitaip tariant, kokybiniai, negriežti tikslai (angl. *Soft Goal*). Šiuos dviejų tipų tikslus apibendrina bendras elementas „tikslas“ (angl. *Goal*). Taip pat yra aprašomas žodynas, kuris susideda iš konceptus vaizduojančių esybių. Tikslais grindžiamų reikalavimų sekimo metodikos dalis „Vartotojo reikalavimai“ (angl. *User Requirements*) pateikta 2.2 paveiksle.



2.2 pav. Metamodelio dalis „Vartotojo reikalavimai“

2.2.2.1. Tikslų modelis

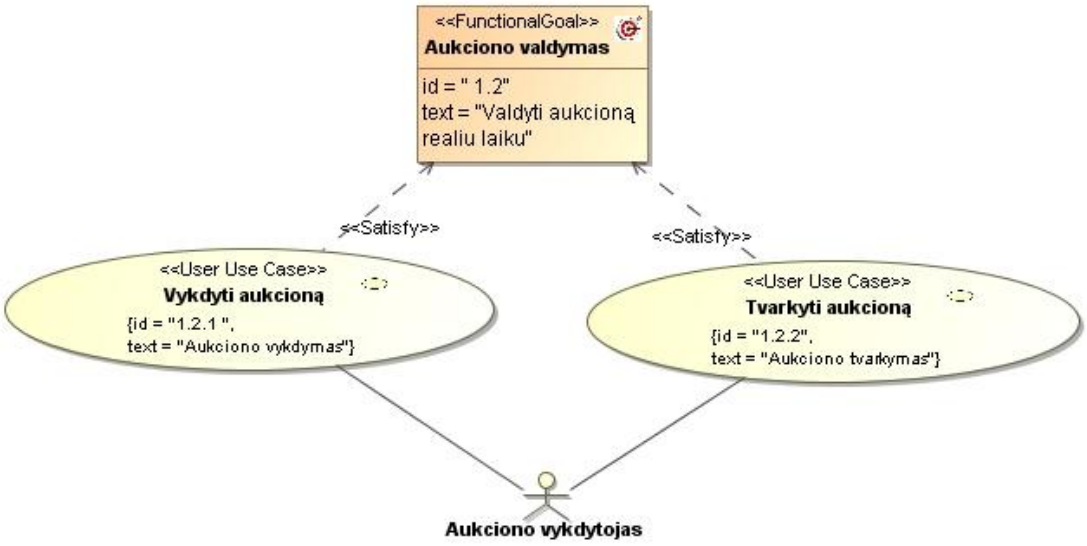
Tikslų modelyje yra vaizduojami funkciniai ir negriežti (nefunkciniai) vartotojo tikslai. Kiekvienas tikslas turi savo „id“ žymę, pagal kurią yra identifikuojamas, bei žymę „text“, kurioje yra detaliai aprašomas tikslas. Tikslų modelio pavyzdys pateiktas 2.3 paveiksle. Čia ir tolesniuose pavyzdžiuose vaizduojamas aukciono informacinės sistemos modelis, sudarytas [21] pagrindu. Šis darbas buvo tiriamas eksperimento metu, o čia sudaryti aukciono sistemos modeliai taip, kaip juos reikėtų sudaryti pagal sukurtą metodą, kad būtų galima atsekti reikalavimus.



2.3 pav. Tikslų diagrama

2.2.2.2. Perėjimas nuo funkcinių tikslų prie reikalavimų

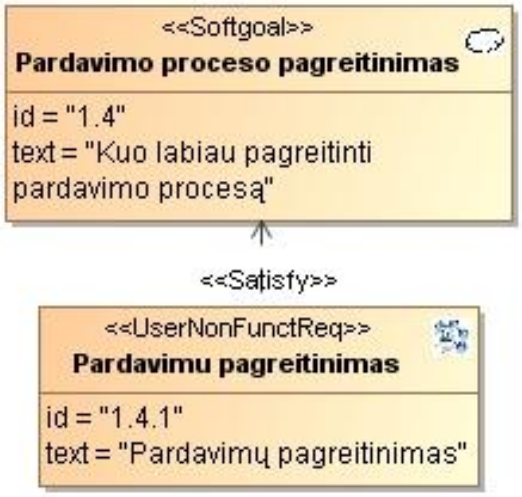
Dalyje „Vartotojo reikalavimai“ nuo funkcinių tikslų reikia pereiti prie funkcinių reikalavimų. 2.4 paveiksle pavaizduota, kaip nuo funkcinio tikslo „Aukciono valdymas“ yra pereinama prie panaudojimo atvejų, kurie savo ruožtu dalyje „Sistemos reikalavimai“ bus detalizuojami į sistemos funkcinius reikalavimus. Taip pat vaizduojamas aktorius „Aukciono vykdytojas“, kuris atlieka pavaizduotus veiksmus. Tikslas yra identifikuojamas žyme „id“ ir apibūdinamas žyme „text“, kurioje yra detaliai aprašomas. Panaudojimo atvejai taip pat yra identifikuojami „id“ žyme tam, kad vėliau būtų galima atsekti reikalavimus. Panaudojimo atvejo detaliam aprašymui naudojama žymė „text“. Aktorius yra identifikuojamas savo vardu, dėl to jam nėra sukurta atskiros identifikavimo žymės.



2.4 pav. Perėjimas nuo funkcinių tikslų prie reikalavimų

2.2.2.3. Perėjimas nuo nefunkcinių tikslų prie nefunkcinių reikalavimų

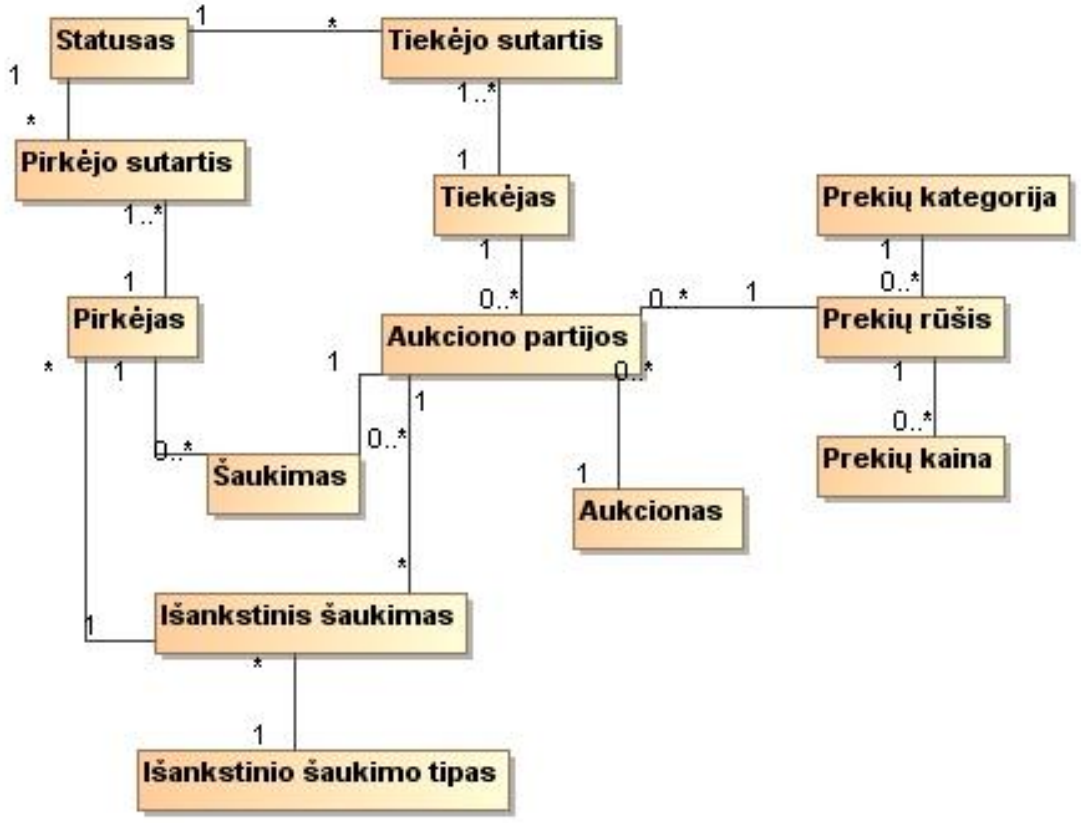
Dalyje „Vartotojo reikalavimai“ nuo nefunkcinių tikslų reikia pereiti prie nefunkcinių reikalavimų. 2.5 paveiksle pavaizduota, kaip nuo nefunkcinio tikslo (angl. *Soft Goal*) „Pardavimo proceso pagreitinimas“ yra pereinama prie vartotojo nefunkcinio reikalavimo „Pardavimu pagreitinimas“. Lengvas tikslas (angl. *Soft Goal*) yra identifikuojamas žyme „id“, bei apibūdinamas žyme „text“, kurioje yra detaliai aprašomas. Vartotojo nefunkciniai reikalavimai taip pat yra identifikuojami „id“ žyme tam , kad vėliau būtų įmanomas reikalavimų atsekimas.



2.5 pav. Perėjimas nuo nefunkcinių tikslų (angl. *Soft Goal*) prie nefunkcinių reikalavimų

2.2.2.4. Žodyno sudarymas

Dalyje „Vartotojo reikalavimai“ yra sudaromas esybių žodynas, pagal kurį „Sistemos reikalavimuose“ bus sudaroma esybių klasių diagrama. Esybės vaizduojamos be atributų ar operacijų. Atributai atsiras esybių klasių diagramoje „Sistemos reikalavimai“ dalyje. Esybės tarpusavyje jungiasi ryšiais, kurie turi kardinalumus. Žodyno diagrama pateikta 2.6 paveiksle.



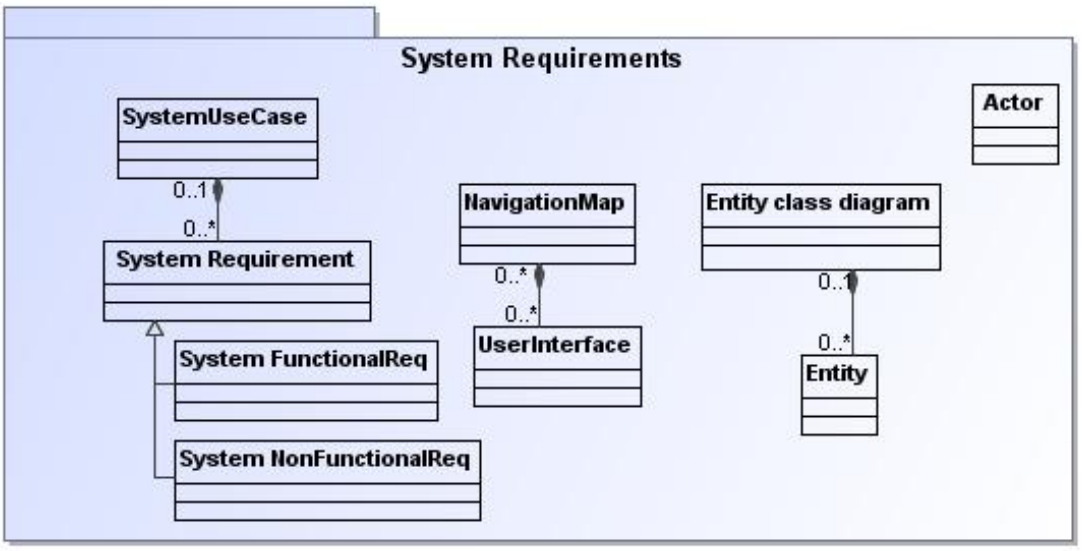
2.6 pav. Žodynas

2.2.3. Sistemos reikalavimai (angl. *System Requirements*)

Sistemos reikalavimų specifikacija gaunama sistemos reikalavimų analizės proceso rezultate, kurio tikslas – paversti vartotojų reikalavimus į pageidaujamos sistemos techninių reikalavimų aibę, pagal kurią bus atliekamas projektavimas. Nuo tikslų yra pereinama prie reikalavimų.

Čia yra detalizuojami reikalavimai, aprašomi funkciniai ir nefunkciniai sistemos reikalavimai, aprašomos sistemos esybės. Tikslais grindžiamų reikalavimų

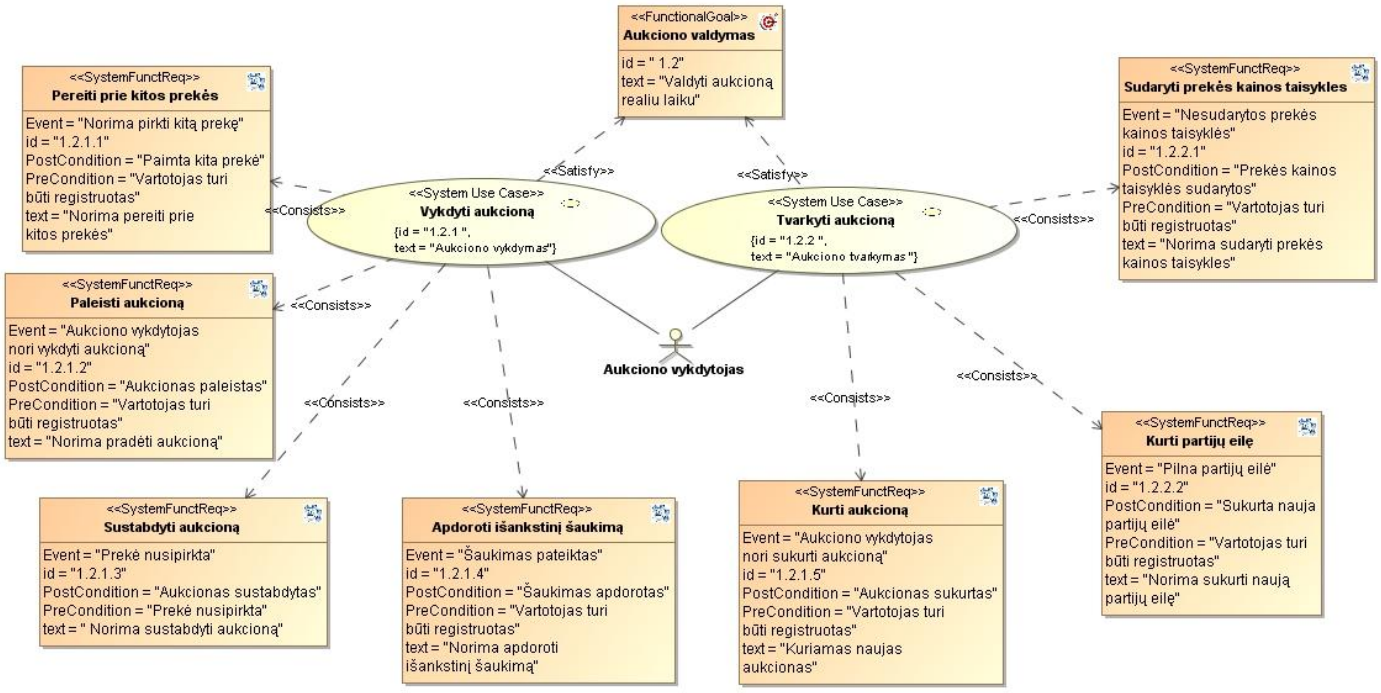
sekimo metodikos dalis „Sistemos reikalavimai“(angl. *System Requirements*) pateikta 2.7 paveiksle.



2.7 pav. Metamodelio dalis „Sistemos reikalavimai“

2.2.3.1. Funkcinių tikslų vaizdavimas reikalavimais

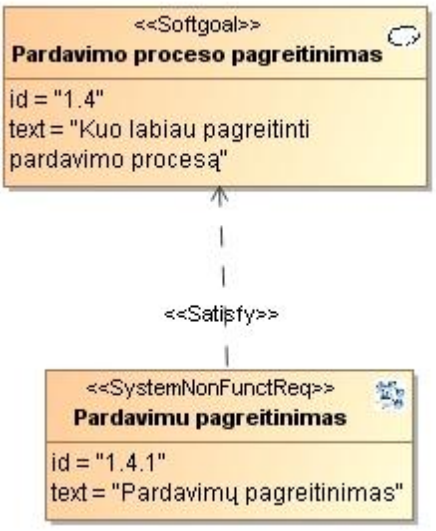
Dalyje „Vartotojo reikalavimai“ nuo funkcinių tikslų buvo pereita prie panaudojimo atvejų. Dabar ši diagrama yra detalizuojama pereinant prie sistemos funkcinių reikalavimų. Sistemos panaudojimo atvejai „Vykdyti aukcioną“ ir „Tvarkyti aukcioną“ yra detalizuojami į sistemos funkcinius tikslus. Taip pat vaizduojamas aktorius „Aukciono vykdytojas“, kuris atlieka pavaizduotus veiksmus. Aktorius yra identifikuojamas savo vardu, dėl to jam nėra sukurta atskiros identifikavimo žymės. Kiekvienas sistemos funkcinis reikalavimas turi būti realizuotas kaip operacija „Sistemos projektas“ dalyje. Sistemos funkciniai reikalavimai turi 5 žymes (angl. *tag*). Žymė „Event“ nurodo, koks įvykis iššaukia šią operaciją. Žyme „id“ sistemos reikalavimas yra identifikuojamas, o „text“ žyme yra detaliai aprašomas. Sistemos funkcinis reikalavimas turi „prieš“ ir „po“ sąlygas. Ties žyme „PreCondition“ aprašoma „prieš“ sąlyga, o ties „PostCondition“ aprašoma „po“ sąlyga.



2.8 pav. Funkcinių tikslų vaizdavimas sistemos funkciniais reikalavimais

2.2.3.2. Nefunkcinių tikslų perėjimas prie reikalavimų

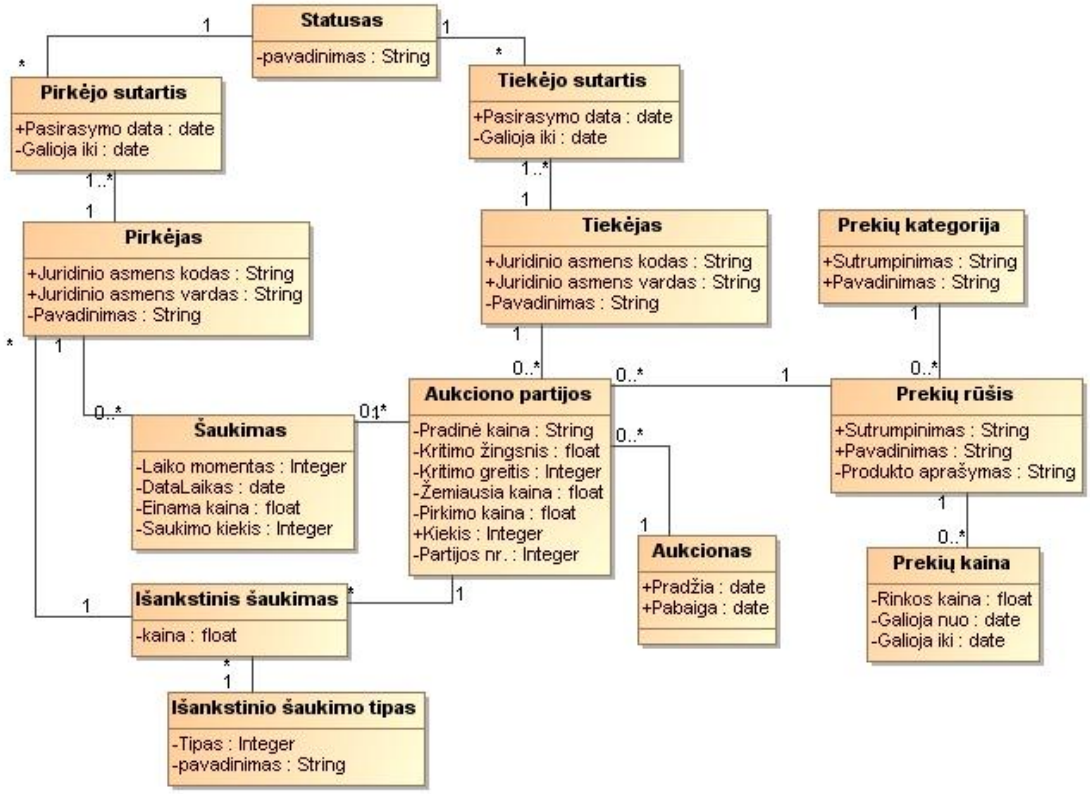
Dalyje „Vartotojo reikalavimai“ nuo nefunkcinių tikslų buvo pereita prie nefunkcinių vartotojo reikalavimų. Dalyje „Sistemos reikalavimai“ lieka tas pats perėjimas, tik vartotojo nefunkcinis reikalavimas tampa sistemos nefunkciniu reikalavimu. 2.9 paveiksle pavaizduota, kaip nuo nefunkcinio tikslo (Soft Goal) „Pardavimo proceso pagreitinimas“ yra pereinama prie sistemos nefunkcinio reikalavimo „Pardavimu pagreitinimas“.



2.9 pav. Perėjimas nuo nefunkcinių tikslų prie nefunkcinių sistemos reikalavimų

2.2.3.3. Esybių klasių diagrama

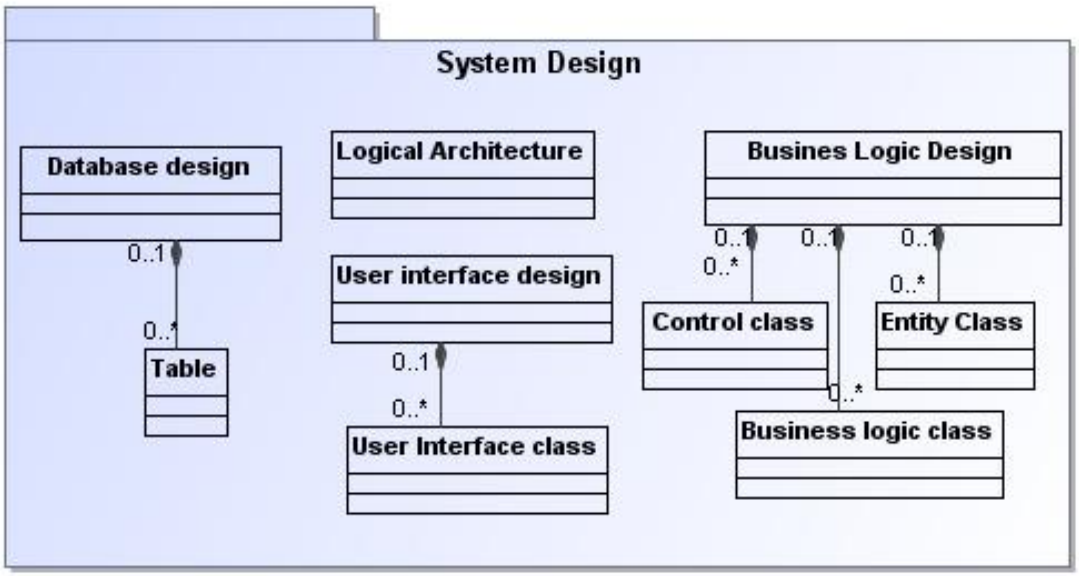
Dalyje „Vartotojo reikalavimai“ buvo sudarytas esybių žodynas. Pagal šį žodyną yra sudaryta esybių klasių diagrama. Esybės vaizduojamos su atributais. Esybės tarpusavyje jungiasi ryšiais, kurie turi kardinalumus. Esybių diagrama pateikta 2.10 paveiksle.



2.10 pav. Esybių klasių diagrama

2.2.4. Sistemos projektas (angl. *System Design*)

Sistemos projekte pagal esybių klasių diagramą yra sudaromas duomenų bazės modelis, pateikiama sistemos loginė architektūra. Yra braižomos veiklos logikos diagramos, pateikiama vartotojo sąsaja. Metamodelio dalis „Sistemos projektas“ (angl. *System Design*) pateikta 2.11 paveiksle.



2.11 pav. Metamodelio dalis „Sistemos projektas“

2.2.4.1. Loginė sistemos architektūra

Projekto sudarymo metu pirmiausia turi būti išskiriamos pagrindinės sistemos posistemės:

- **Aukciono valdymo posistemė**

  Šioje posistemėje bus realizuoti aukciono informacijos tvarkymo ir valdymo moduliai.

- **Pirkimų vykdymo posistemė**

  Šiame posistemyje bus realizuoti pirkėjų aukciono vykdymo moduliai, pirkėjų išankstinių šaukimų moduliai, tiekėjų aukciono stebėjimo moduliai.

- **Aukciono variklio paslaugos**

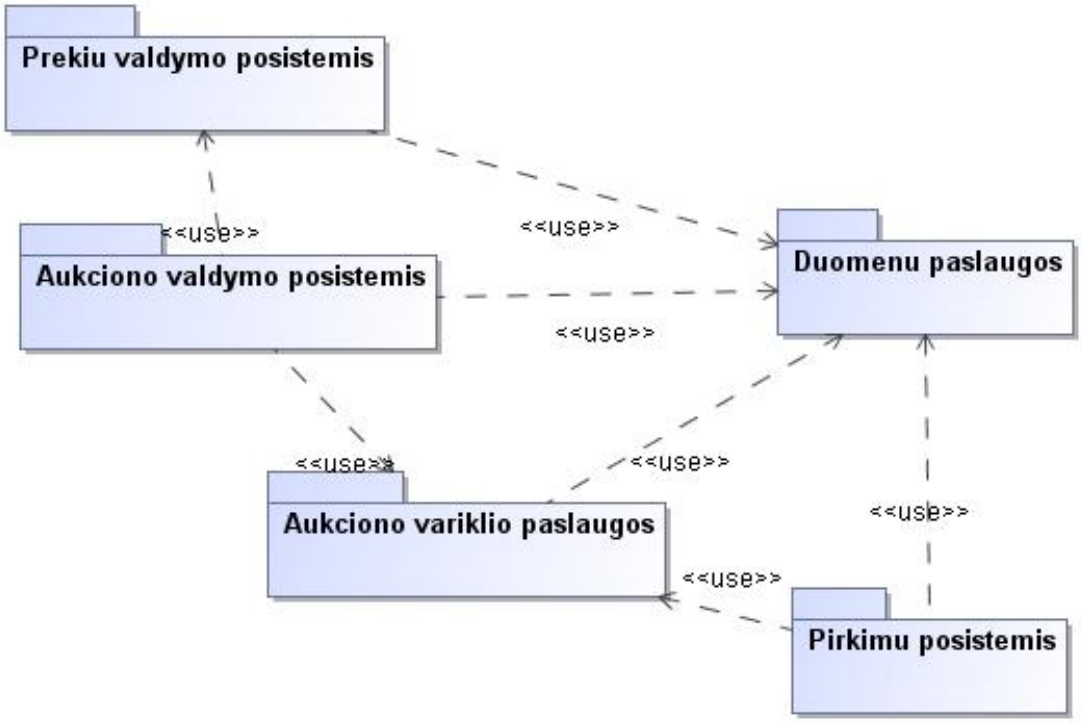
  Šiame modulyje bus realizuota interneto paslauga, kuri atliks aukciono variklio funkcijas ir logiką.

- **Prekių valdymo ir apskaitos posistemė**

  Prekių valdymo turi būti registruojami vartotojai, pirkėjai ir tiekėjai. Taip pat turi būti registruojamos tiekėjų atvežtos prekės.

- **Duomenų paslaugos**

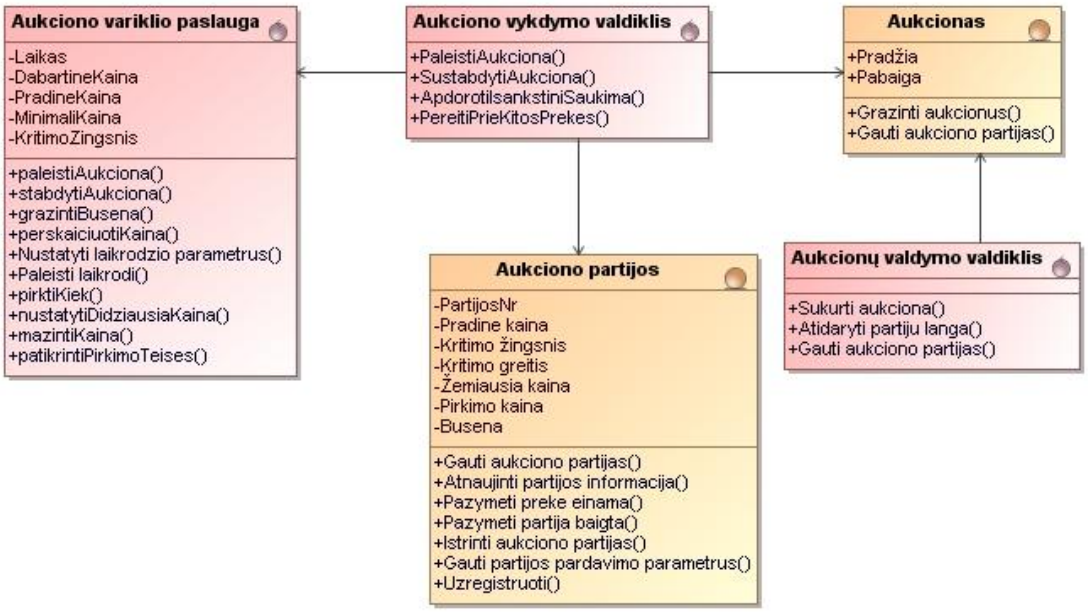
Loginė sistemos architektūra pateikta 2.12 paveiksle.



2.12 pav. Sistemos loginė architektūra

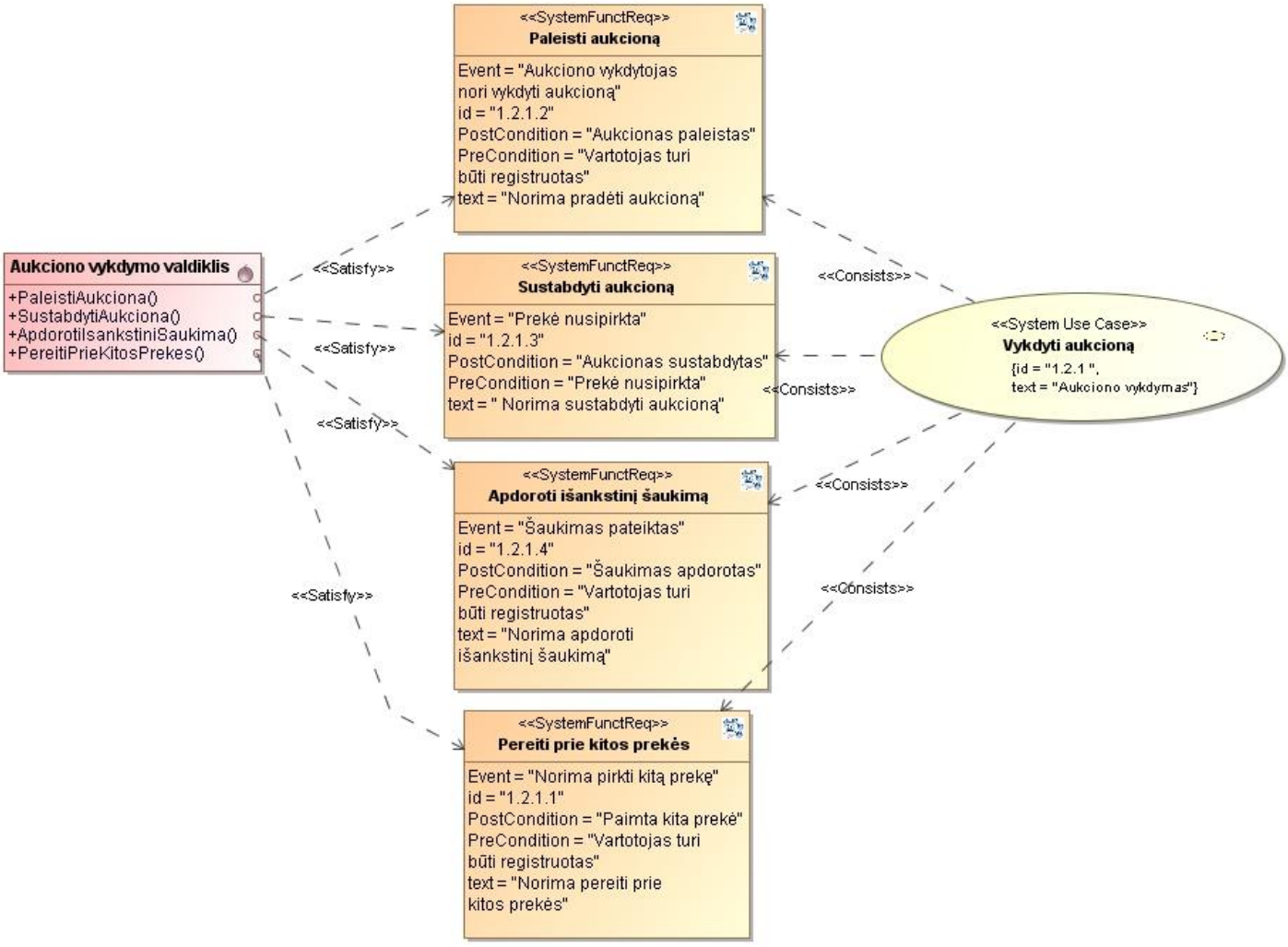
2.2.4.2. Veiklos klasių diagramos

Čia turi būti pateikiamos veiklos klasių diagramos. 2.13 paveiksle pateikta aukciono valdymo posistemio veiklos klasių diagrama.



2.13 pav. Aukciono valdymo posistemio vartotojo veiklos paslaugų klasių diagrama

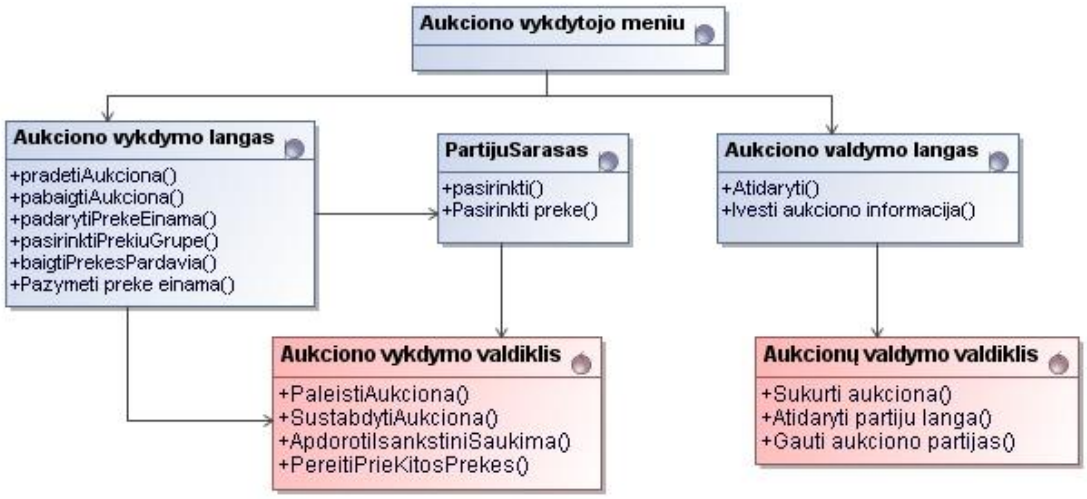
Kiekviena valdiklio operacija turi atitikti sistemos reikalavimus. 2.13 paveiksle esančioje diagramoje yra „Aukciono vykdymo valdiklis“, kuris turi tam tikras operacijas. Šios operacijos turi ateiti iš sistemos funkcinių reikalavimų. 2.14 paveiksle pavaizduota, kaip turi atsivaizduoti sistemos reikalavimai projekte, t.y. valdiklio operacijos sudaromos iš atitinkamų sistemos reikalavimų.



2.14 pav. Funkcinių reikalavimų vaizdavimas „Sistemos projekte“

2.2.4.3. Vartotojo sąsajos klasių diagrama

Metamodelio dalyje „Sistemos projektas“ (angl. *System Design*) turi būti pateikta vartotojo sąsajos klasių diagrama. Jos gali būti kelios. Kaip pavyzdį pateiksiu aukciono vykdymo posistemio vartotojo sąsajos klasių diagramą. Aukciono vykdymo posistemėje yra du pagrindiniai langai – aukciono vykdymo ir aukciono valdymo moduliai. Šio posistemio vartotojo sąsajos klasių diagrama pateikta 2.15 paveiksle.



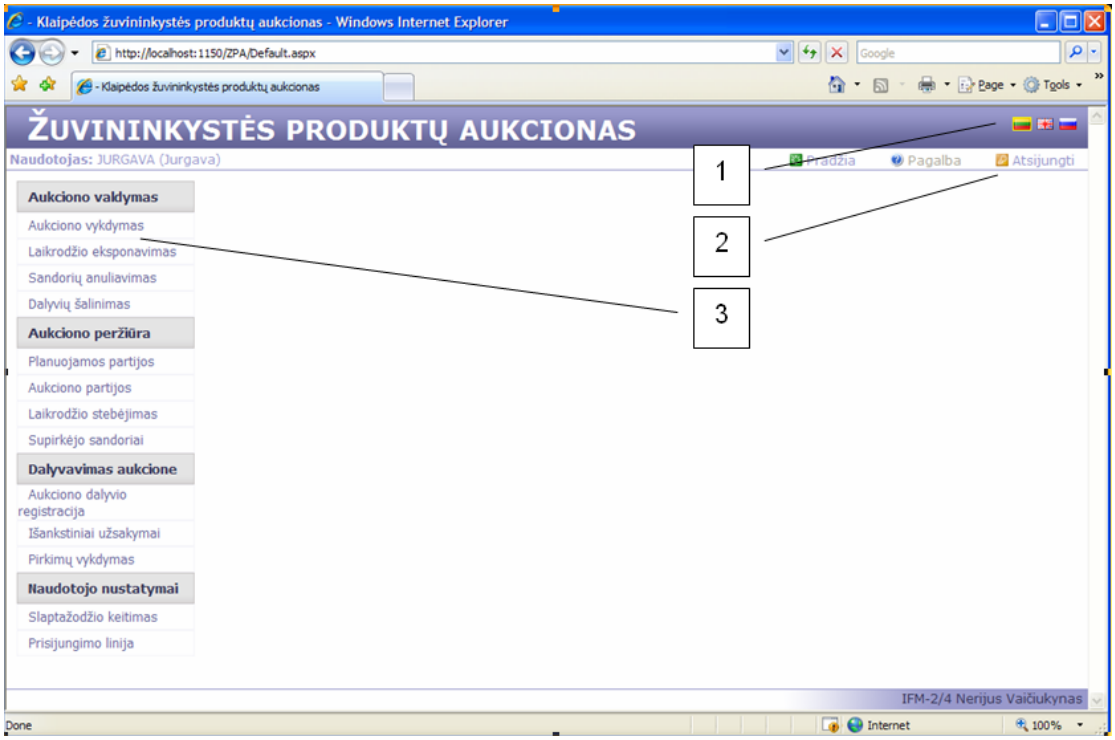
2.15 pav. Vartotojo sąsajos klasių diagrama

2.2.4.4. Vartotojo sąsaja

Čia turi būti pateikti vartotojo sąsajos langai. Kaip pavyzdys įdėtas pagrindinis „Žuvininkystės produktų aukciono“ sistemos langas (2.16 paveikslas). Lange yra pateikiamas pagrindinis meniu ir valdymo mygtukai.

Pagrindinio lango aprašymas:

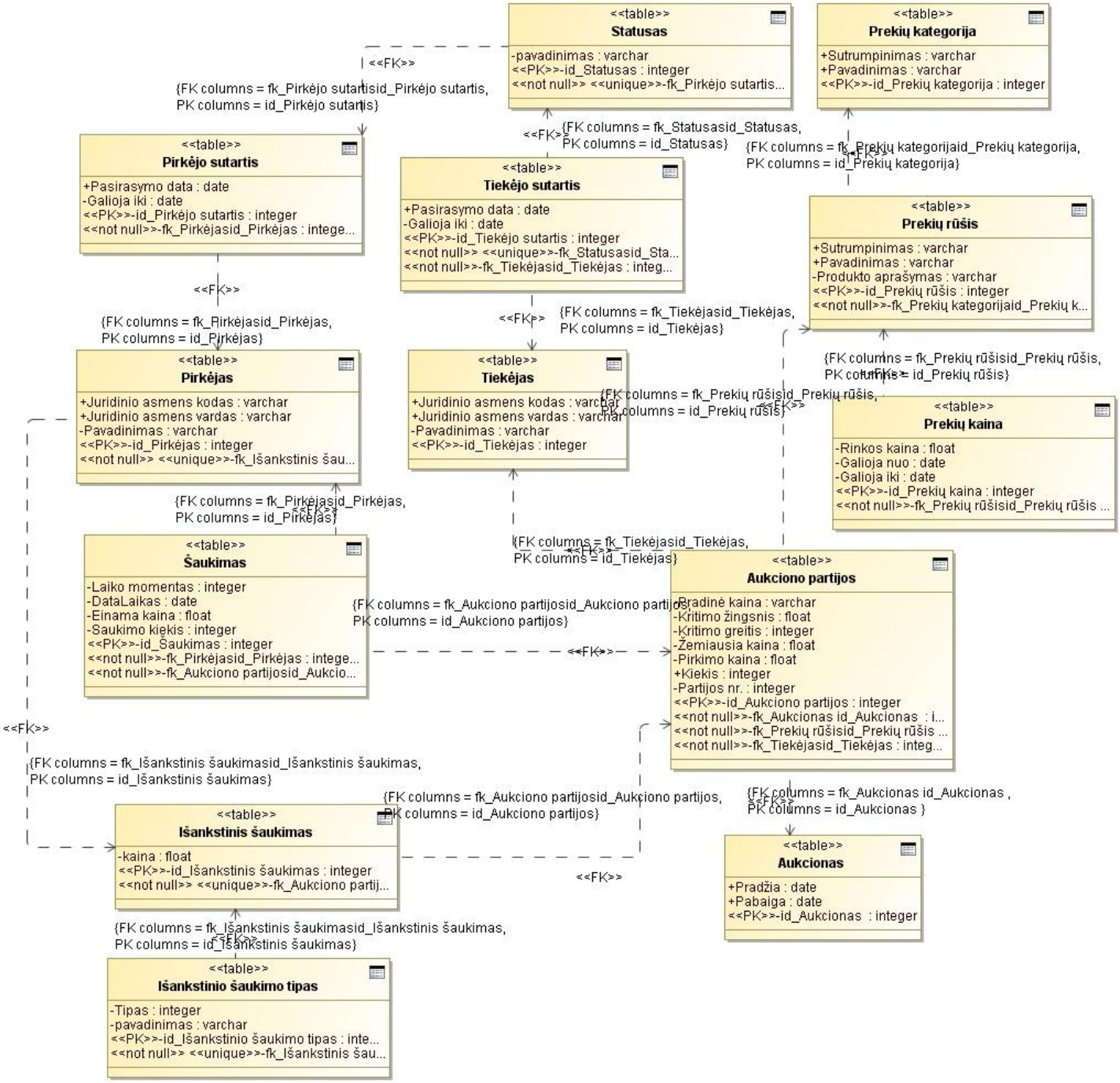
- vartotojo grafinės sąsajos kalbos parinkimo mygtukai;
- pagrindiniai programos valdymo mygtukai: navigacija į pradžią, pagalba, atsijungimas;
- pagrindinis meniu.



2.16 pav. Sistemos pagrindinis langas

2.2.4.5. Duomenų bazės schema

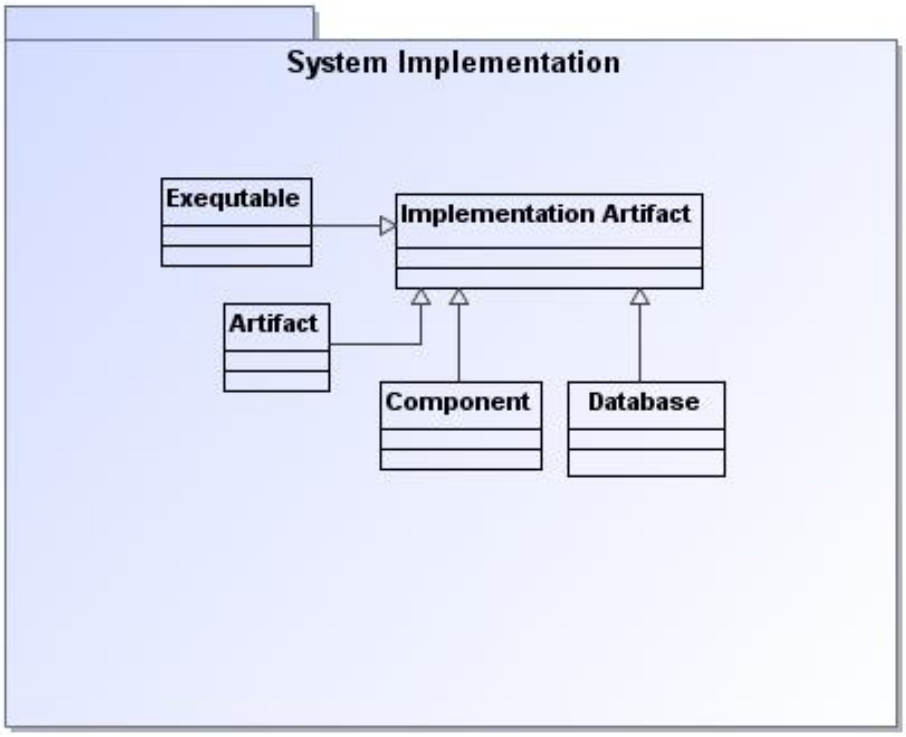
Pagal esybių klasių diagramą (2.10 paveikslas) turi būti sudaroma duomenų bazės schema. Naudojant paketą Magic Draw tai galima padaryti automatiškai. Sugeneruota duomenų bazės schema pateikta 2.17 paveiksle.



2.17 pav. Duomenų bazės schema

2.2.5. Sistemos realizacija (angl. *System Implementation*)

Tai yra paskutinė metamodelio dalis. Šioje dalyje atsiranda komponentai, kurie yra vaizduojami artefaktais. Braižomos komponentų diagramos. Šioje dalyje turi būti aprašoma duomenų bazės fizinė schema. Metamodelio dalis „Sistemos realizacija“ (angl. *System Implementarion)* pavaizduota 2.18 paveiksle.



2.18 pav. Metamodelio dalis „Sistemos realizacija“

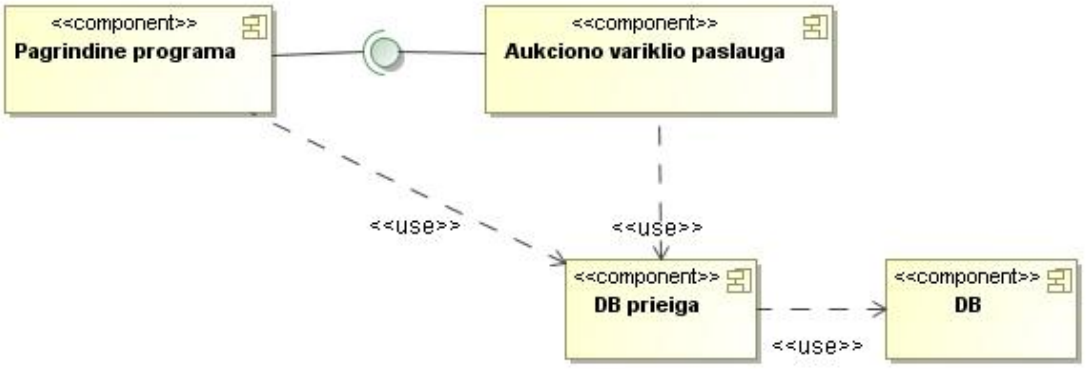
2.2.5.1. Komponentų diagramos

Komponentų diagramoje vaizduojami sistemą sudarantys komponentai. Kaip pavyzdį pateiksiu elektroninio aukciono informacinės sistemos komponentų diagramą.

Visą kuriamą informacinę sistemą sudaro 4 pagrindiniai komponentai:

- **Pagrindinė internetinė programa** – tai visa sistema ir jos valdymas.
- **Aukciono variklio paslauga** – tai aukciono valdiklis, kuriame saugoma aukciono vykdymo logika.
- **DB prieiga** – komponentas, kuriame yra saugoma DB prisijungimo informacija ir per jį atliekamas darbas su DB.
- **Duomenų bazė** – bendra sistemos duomenų saugykla.

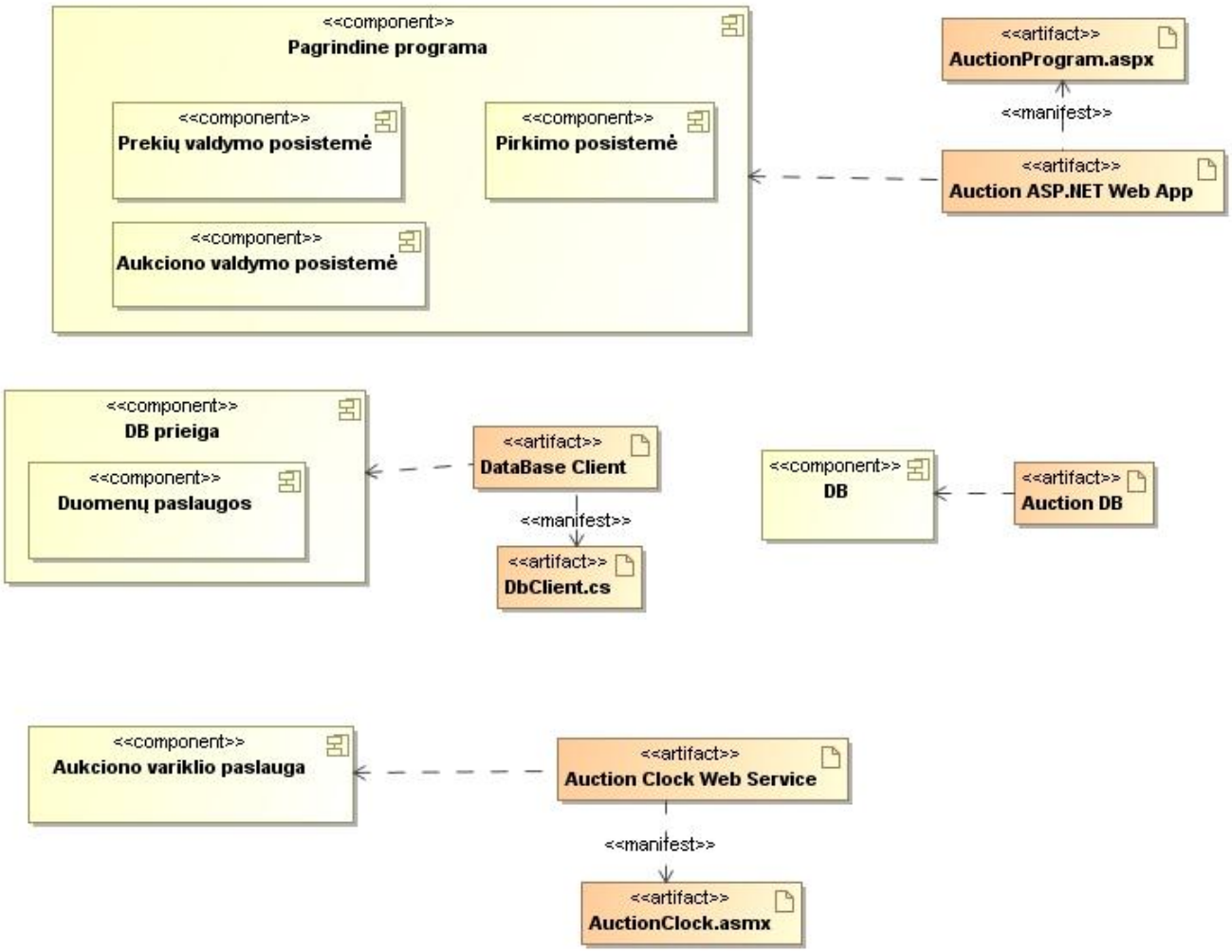
Sistemos komponentų diagrama pateikta 2.19 paveiksle.



2.19 pav. Komponentų diagrama

2.2.5.2. Komponentų vaizdavimas artefaktais

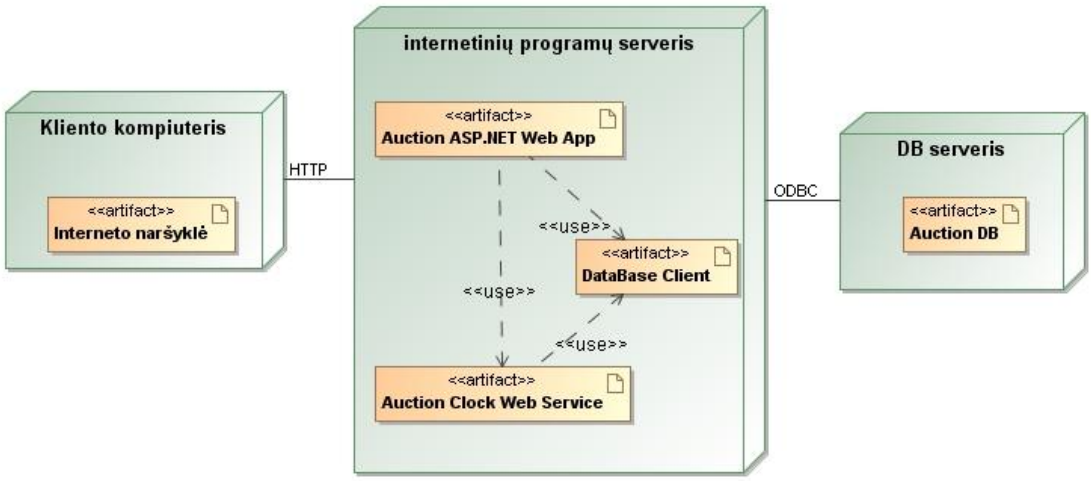
Visi loginiai komponentai yra vaizduojami artefaktais. Komponentų vaizdavimo artefaktais diagrama pateikta 2.20 paveiksle.



2.20 pav. Komponentų realizavimas artefaktais

2.2.5.3. Sistemos diegimo diagrama

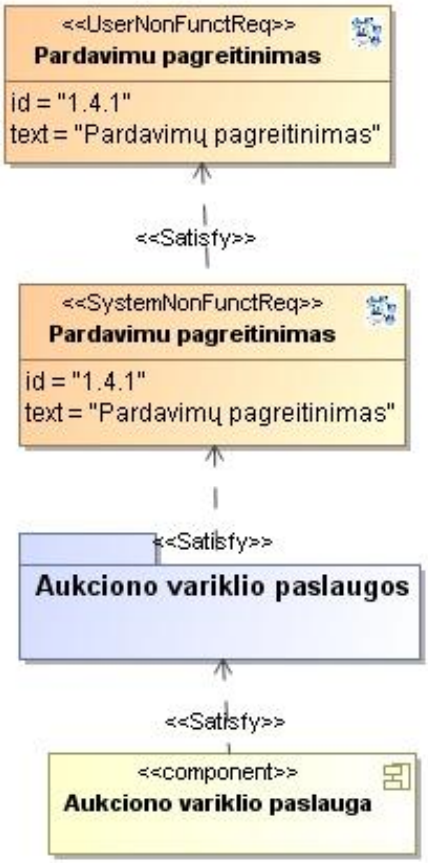
Sistemos artefaktai yra sudiegiami į techninius įrenginius. Sistemai įdiegti reikia interneto programų serverio ir duomenų bazės serverio. Sistema gali būti sudiegta ir viename fiziniame įrenginyje. Interneto programų serveryje turi būti sudiegta pagrindinė programa ir interneto paslauga. Artefaktų pasiskirstymas techniniuose įrenginiuose pateiktas 2.21 paveiksle.



2.21 pav. Sistemos diegimo diagrama

2.2.5.4. Nefunkcinių reikalavimų vaizdavimas sistemos realizacijoje

Sistemos funkciniai reikalavimai vaizduojami valdiklių operacijose „Sistemos projekto“ dalyje, o nefunkciniai reikalavimai vaizduojami sistemos komponentuose. 2.22 paveiksle matosi, kaip vartotojo nefunkcinis reikalavimas „Pardavimu pagreitinimas“ vaizduojamas į sistemos nefunkcinį reikalavimą, kuris savo ruoštu „Sistemos projekto“ dalyje atsispindėjo „Aukciono variklio paslaugoje“ ir galiausiai „Sistemos realizacijos“ dalyje šis reikalavimas vaizduojamas į komponentą „Aukciono variklio paslauga“. Ir taip kiekvienas reikalavimas turi būti vaizduojamas atitinkamuose elementuose. Jeigu iš pradžių buvo reikalavimas, bet galiausiai jis nebuvo pavaizduotas nei sistemos komponente, nei valdiklio operacijoje, tai reiškia, kad reikalavimas buvo „pamestas“ ir jis nebėra išpildomas.



2.22 pav. Nefunkcinių reikalavimų vaizdavimas sistemos realizacijoje

2.2.6. Funkciniai ir nefunkciniai reikalavimai

Funkciniai reikalavimai aprašo sistemos arba metodo funkcionalumą, sistemos paslaugas (paaiškina, kaip sistema turėtų reaguoti į ypatingus duomenų įvedimus ir kaip sistema elgsis ypatingose situacijose)**.** Funkciniai reikalavimai priklauso nuo programinės įrangos tipo, laukiamų vartotojų ir sistemos tipo, kur programinė įranga yra naudojama

Funkciniai vartotojo reikalavimai gali būti aukšto lygio teiginiai, apie tai, ką metodas ar sistema turi daryti, bet funkciniai sistemos reikalavimai turi detaliai aprašyti sistemos paslaugas arba metodą..

Nefunkciniai reikalavimai apriboja kuriamą sistemą, naudojamą metodą ar kūrimo procesą, t.y. apibrėžia sistemos savybes ir apribojimus kaip pvz. patikimumą. Nefunkciniai reikalavimai gali būti labiau svarbūs nei funkciniai reikalavimai. Jei jie nėra išpildomi, sistema yra nenaudinga.

Nefunkcinių reikalavimų klasės

Nefunkcinius reikalavimus galima suskirstyti į:

- Išvaizdos ir patrauklumo (angl. *Look and Feel*) reikalavimai – produkto išvaizdos apibudinimas.
- Naudojimo (angl. *Usability*) reikalavimai – produkto naudojimo patogumas.
- Našumo (angl. *Perfomance*) reikalavimai – kiek spartus, funkcionalus, tikslus turi būti produktas.
- Veikimo (angl. *Operational*) reikalavimai – kokioje aplinkoje turi veikti produktas, kokie papildomi reikalavimai keliami aplinkai.
- Priežiūros ir mobilumo (angl. *Maintainability and Portability*) reikalavimai – pakeitimai, kurių tikimasi ir laikas jiems atlikti skirtas.
- Saugumo (angl. *Security*) reikalavimai – produkto saugumas ir konfidencialumas.
- Kultūriniai ir politiniai (angl. *Cultural and Political*) reikalavimai – specialūs reikalavimai susiję su žmonėmis, kurie naudosis šiuo produktu.
- Teisiniai (angl. *Legal*) reikalavimai – kokius įstatymus ir standartus turi atitikti produktas.

Žinoma fiksuojant nefunkcinius reikalavimus, kartais yra sunku nustatyti, kuriai klasei vienas ar kitas reikalavimas turėtų būti priskirtas .

Išvaizdos ir patrauklumo (angl. *Look and Feel*) reikalavimai – produkto išvaizdos apibūdinimas, bet tai nėra detalus vartotojo sąsajos projektas. Šie reikalavimai labiau nusako tikslus, keliamus vartotojo sąsajai. Išvaizdos ir patrauklumo poreikius galima gauti panaudojant vartotojo sąsajos prototipus, tačiau reikia turėti omenyje, kad prototipas neišreiškia reikalavimų, jis tik parodo rezultatus, kuriuos turi pateikti reikalavimai. O klientui prototipas turi būti būdas, kuriuo jis galėtų lengviau išreikšti savo reikalavimus. Reikalavimai vartotojo sąsajai gali būti nusakyti nurodant grafinės sąsajos standartą.

Naudojimo (angl. *Usability*) reikalavimai – produkto naudojimo patogumo reikalavimai. Šių reikalavimo gavimo ir jų perkėlimo į produktą sėkmė lems darbo su produktu produktyvumą. Šie reikalavimai turi atspindėti būsimo produkto vartotojų įgūdžius, gebėjimą įvaldyti naują produktą, darbo su produktu specifiką. Naudojimo reikalavimai taip pat gali apimti:

- produkto priimtinumas vartotojams;
- produktyvumo didėjimas pradėjus naudoti produktą;
- klaidų galimumas;
- produkto naudojimas vartotojų, kurie kalba kita kalba, negu apipavidalintas produktas;
- produkto prieinamumas neįgaliems vartotojams;
- produkto naudojimo galimybės vartotojų be patirties.

Naudojimo reikalavimai yra sudaromi kombinuojant tai, ką klientas siekia laimėti naudodamas produktą ir ko klientas tikisi iš produkto.

Našumo (angl. *Perfomance*) reikalavimai – šie reikalavimai yra užrašomi, jei to reikia produktui: tam tikros užduotys turi būti atliktos per tam tikrą laiko tarpą. Ypač tai aktualu realaus laiko sistemoms. Prie našumo reikalavimų priskiriami ir talpos reikalavimai, nurodantys su kokiais kiekiais duomenų turi dirbti sistema. Apskritai kalbant apie našumo reikalavimus, reikėtų galvoti apie šiuos aspektus:

- užduoties atlikimo spartumas;
- rezultatų punktualumas, tikslumas;
- leidžiamų reikšmių aibė;
- transakcijų greitis;
- saugojimo vieta;
- išteklių panaudojimo efektyvumas;
- patikimumas – paprastai tai išreiškiama laiku tarp trikių.

Šie reikalavimai dažniausiai kyla iš darbinės aplinkos, nes kiekviena aplinka turi savo aplinkybes ir sąlygas.

Veikimo (angl. *Operational*) reikalavimai – šie reikalavimai aprašo aplinką, kurioje turi veikti produktas. Kartais aplinka turi įtakos sąlygoms, kuriose produktas turi būti sukurtas. Taip pat galima galvoti ir apie tai, su kokiomis sistemomis turės bendradarbiauti produktas. Veikimo reikalavimai gali apimti :

- sistemos veikimo aplinką;
- vartotojo sąlygas;
- partnerines ar bendradarbiavimo sistemas;
- produkto portatyvumas.

Norint atrasti šiuos reikalavimus, reikia atkreipti dėmesį į produkto ribas, apibrėžtas panaudojimo atvejų diagramoje, ir apgalvoti kiekvieno aktoriaus ir gretimos sistemos reikalavimus.

Priežiūros ir mobilumo (angl. *Maintainability and Portability*) reikalavimai – paprastai reikalavimų rinkimo metu nėra tikslai žinoma, kokio lygmens priežiūros gali prireikti produktui, be to ne visada žinomas ir priežiūros tipas. Tačiau kuriant sistemą galima nuspėti, kokios priežiūros gali prireikti. Reikia atkreipti dėmesį į galimus pasikeitimus:

- organizacijoje;
- aplinkoje;
- produktui taikomuose įstatymuose;
- veiklos taisyklėse.

Būtinai reikia specifikuoti galimų pokyčių tipus ir laiką, per kurį juos leidžiama atlikti. Šie reikalavimai turi nemažai įtakos produkto projektavimui. Taip pat fiksuojami reikalavimai produkto perkėlimui tiek fiziskai techniniame lygyje, tiek geografiškai.

Saugumo (angl. *Security*) reikalavimai – saugumas yra vienas iš reikalavimų tipų susijusių su didžiausia rizika. Kai rašomi šie reikalavimai, reikia galvoti, kad saugumas skiriamas produktui ir jo aplinkai. Yra išskiriami trys saugumo aspektai:

- konfidencialumas (angl. *confidentiality*)– produkto saugomi duomenys neturi būti bet kam prieinami, kad nebūtų galima jų paskelbti. Rašant reikalavimus šiuo aspektu yra specifikuojamas teisės, sąlygos, kurioms esant teisių suteikimas yra teisėtas. Nusakoma, kokie duomenys ar funkcijos yra leistinos tam tikriems vartotojams.
- Integralumas (angl. *integrity*)- ar produkto duomenys tokie patys kaip duomenų šaltinio ar autoriaus. Integralumas apsaugo nuo neteisingo duomenų naudojimo teisėtų vartotojų. Tai yra viena iš bendriausių duomenų teršimo, iškraipymo formų. Taip pat tikrina netinkamą duomenų naudojimą. Patikrina duomenų integralumą įvykus trikdžiams (pvz.: įtampos dingimas ir t.t.). Dažnai reikalaujama, kad produktas galėtų atlikti duomenų auditą.
- Pasiekiamumas (angl. *availability*)- reiškia, kad vartotojui, kuriam suteiktos teisės, neturi būti trukdoma pasiekti tai, ko jis nori, kad apsaugos sistemos

netrukdytų vartotojo darbo. Taip pat reikia įvertinti duomenų dingimą ir atkūrimą.

Saugumo reikalavimai gali būti užrašyti iš jau esančių reikalavimų, pvz.: produktas turi atlikti tik tai, ką specifikuoja reikalavimai. Fiksuojant saugumo reikalavimus patartina pasinaudoti saugumo ekspertų pagalba.

Kultūriniai ir politiniai (angl. *Cultural and Political*) reikalavimai – tai ypatingas veiksnys, kuris produktą gali padaryti nepriimtiną žmogui, nes tai neatitinka tradicijų, prioritetų ar išankstinės nuostatos. Pagrindinė kultūrinių reikalavimų priežastis - produkto įdiegimas skirtingos kalbos ar kultūros šalyje nei kūrėjų. Šių reikalavimų nustatymas yra sunkus, nes jie dažnai iškyla netikėtai ir iš pirmo žvilgsnio atrodo nelogiški. Taip pat kyla reikalavimų, ir dėl visiškai politinių priežasčių. Šiuos reikalavimus yra sunkiausia užrašyti , nes juos būna sunkoka logiškai suderinti su kitais reikalavimais.

Teisiniai (angl. *Legal*) reikalavimai – šie reikalavimai turi užtikrinti, kad kuriamas produktas neprasilenktų su įstatymais, kad atitiktų standartus, kurie jam keliami. Šių reikalavimų teisėtumui įvertinti reikėtų pasitelkti teisininkus. Išskiriant šiuos reikalavimus, reikia nustatyti visus įstatymus kurie gali galioti kuriamam produktui.

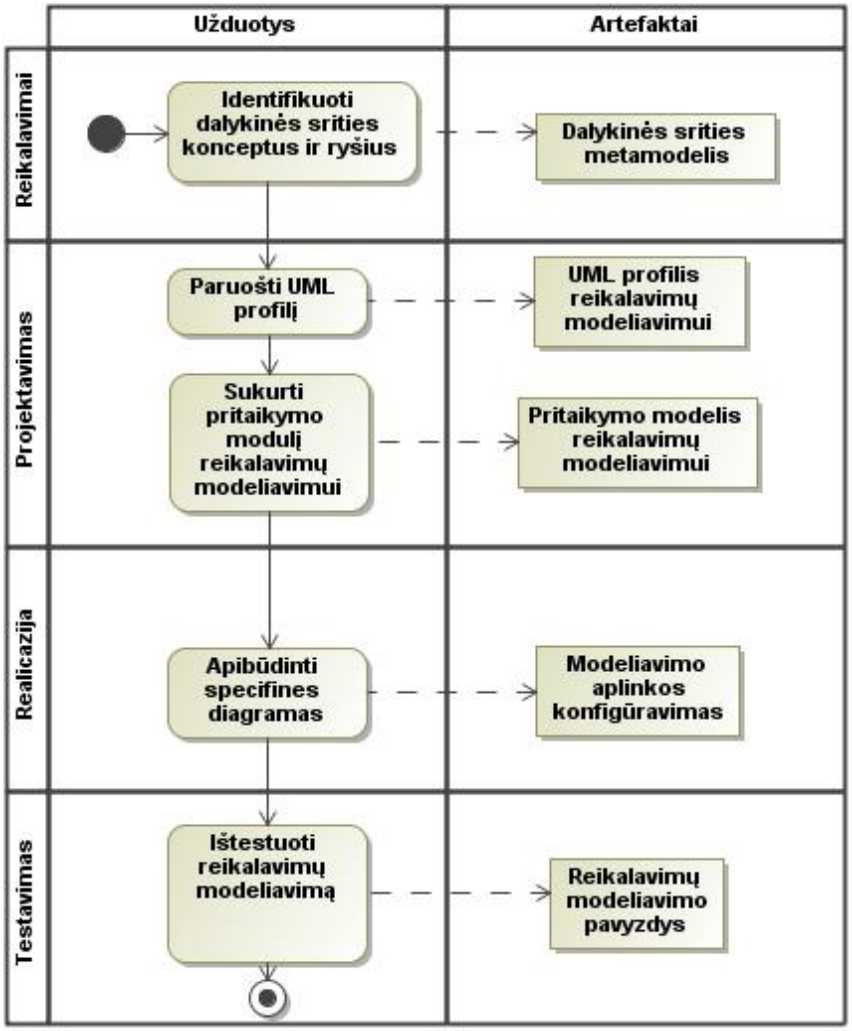
Nefunkciniai reikalavimai kuriamam metodui:

- Metodas turi tikti bet kokių reikalavimų specifikavimui.
- Metodo naudojamumas (angl. *usability*) turėtų būti paprastas.
- Specifikuoti reikalavimai turi būti pilni.
- Metodas turi būti išsamus ir sietis su tikslais.
- Metodas turi būti pritaikytas plačiai naudojamam CASE įrankiui.

3. Reikalavimų sekimo profilio realizacija

3.1. Reikalavimų sekimo profilio kūrimo procesas

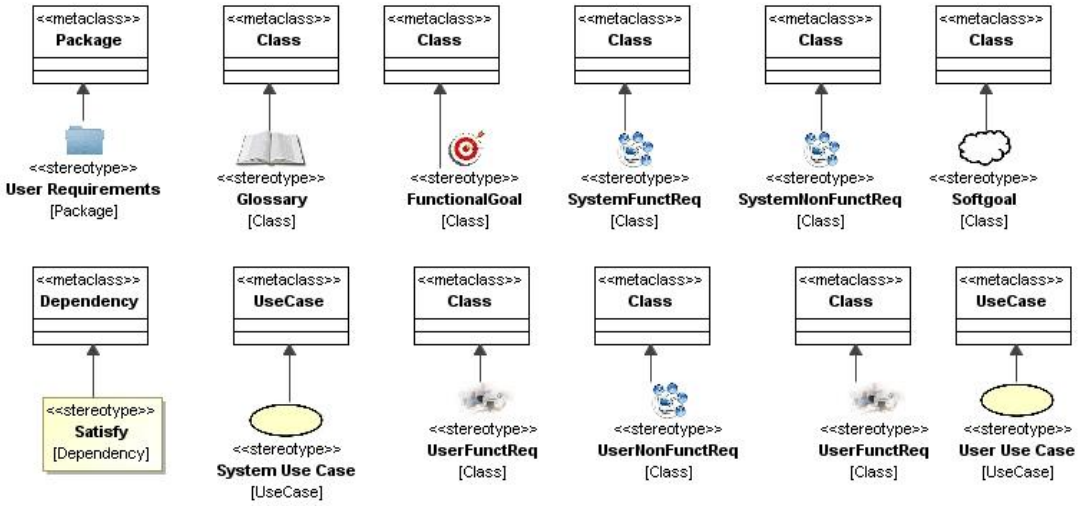
Reikalavimų sekimo profilis buvo kuriamas taikant tam tikrą procesą [19]. Iš pradžių reikia identifikuoti dalykinės srities konceptus ir ryšius. Tai padarius yra ruošiamas UML profilis ir sukuriamas reikalavimų sekimo modulis. Jame yra sukuriamos reikiamos diagramos ir jų elementai. Galiausiai sukurtas profilis yra ištestuojamas. Reikalavimų sekimo profilio kūrimo proceso diagrama pateikta 3.1 paveiksle.



3.1 pav. Reikalavimų sekimo profilio kūrimo diagrama [19]

3.2. Stereotipų modulio kūrimas

Buvo sukurtas profilis (stereotipų modulis), pavadintas „Stereotipai“. Jame yra aprašomi visi stereotipai, kurie naudojami metamodelio kūrime. Stereotipai gali turėti savo žymes. Detalus stereotipų aprašymas pateiktas 3.1 lentelėje. Stereotipai vaizduojami su elementais, kuriuos jie išplečia. 3.2 paveiksle pavaizduoti pagrindiniai sudaryti stereotipai ir jų išplečiami UML metamodelio elementai.



3.2 pav. Stereotipų ir jų giminingų elementų diagrama

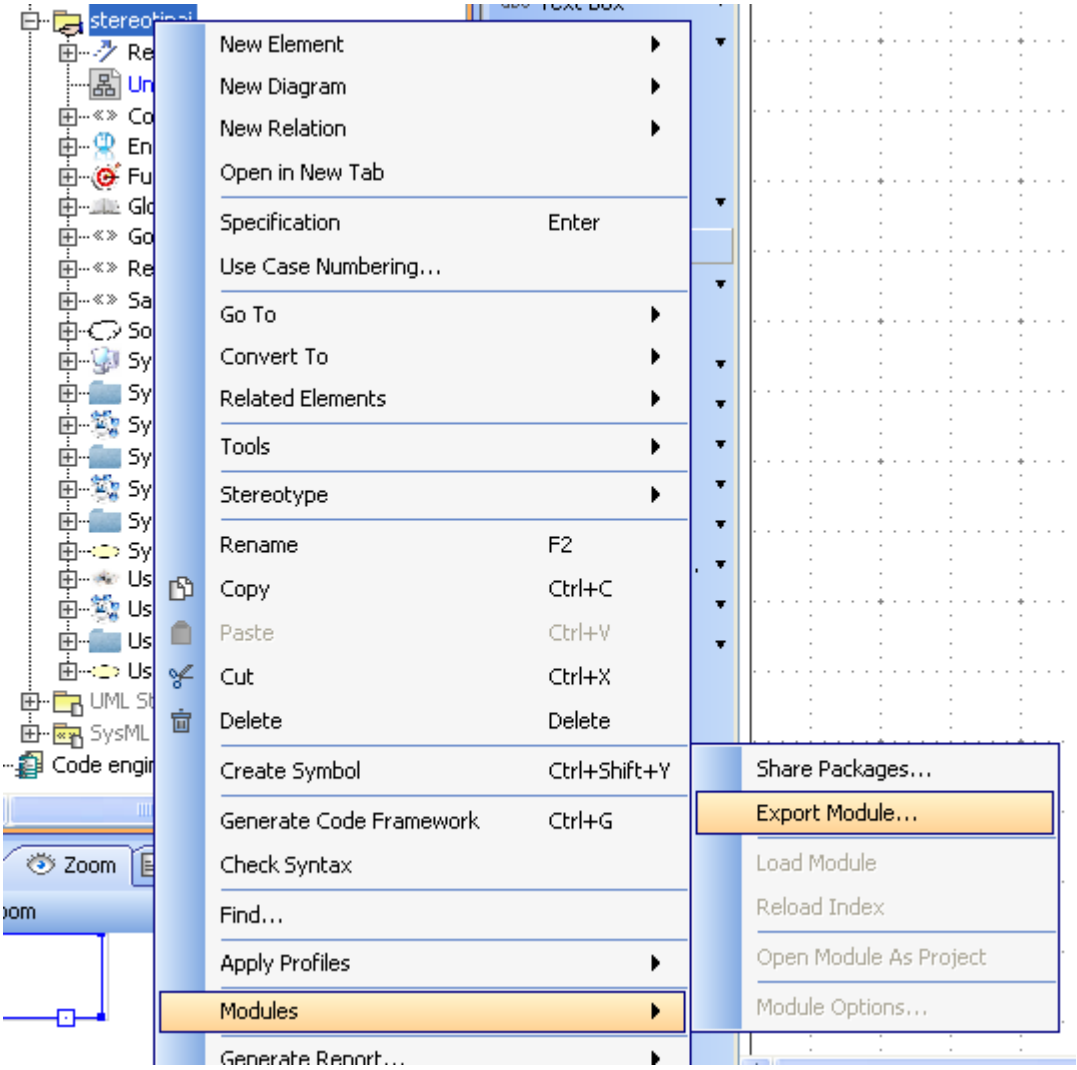
Stereotipų modulio „Stereotipai“ stereotipai aprašyti 3.1 lentelėje.

3.1 lentelė Stereotipų apibūdinimo lentelė

Stereotipo pavadinimas	Naudojama metaklasė	Žymių (tag) reikšmės	Aprašymas
Consists	Dependency	-	Ryšys tarp elementų
Entity	Class	-	Esybė
FunctionalGoal	Class	Id, text	Funkcinis tikslas
Glossary	Class	Id, text	Žodynas
Satisfy	Dependency	-	Ryšys tarp elementų
Softgoal	Class	Id, text	Nefunkcinis tikslas
System	Class	Id, text	Sistema
System Design	Package	-	Paketas
SystemFunctReq	Class	Id, text, Event, PreCondition, PostCondition	Sistemos funkcinis reikalavimas
System Implementation	Package	-	Paketas

SystemNonFunctReq	Class	Id, text	Sistemos nefunkcinis reikalavimas
System Requirements	Package	-	Paketas
System Use Case	UseCase	Id, text	Sistemos panaudojimo atvejis
UserFunctReq	Class	Id, text, Event, PreCondition, PostCondition	Vartotojo funkcinis reikalavimas
UserNonFunctReq	Class	Id, text	Vartotojo nefunkcinis reikalavimas
User Requirements	Package	-	Paketas
User Use Case	UseCase	Id, text	Vartotojo panaudojimo atvejis

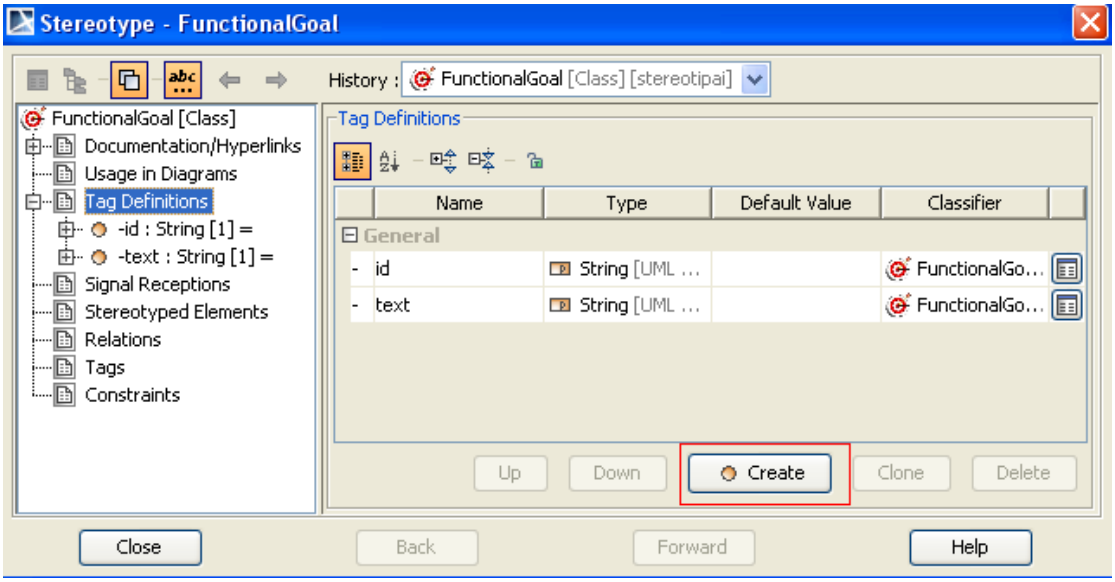
Sukūrus stereotipų modulį jis yra eksportuojamas spaudžiant „Export Module“. Tai daroma tam, kad vėliau kuriant diagramas būtų galima susikurtą modulį naudoti ir taikyti jame esančius stereotipus. Stereotipų paketo eksportavimas pavaizduotas 3.3 paveiksle.



3.3 pav. Stereotipų paketo eksportavimas

3.3. Stereotipų žymių kūrimas

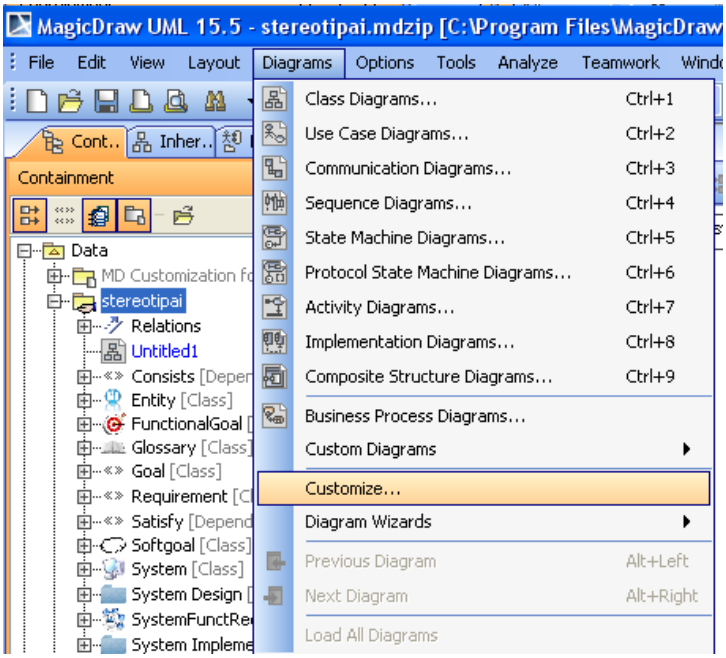
Susikūrus reikiamą stereotipą jeigu yra poreikis jam galima sukurti žymę (angl. *tag* reikšmę). Kuriant reikalavimų sekimo profilį daugumai stereotipų buvo sukurtos žymės, kad būtų realizuojamas reikalavimų atsekamumas. 3.4 paveiksle pavaizduota, kaip yra kuriamos žymės jau susikurtam stereotipui. Žymė kuriama spaudžiant mygtuką „Create“. Sukūrus žymę spaudžiamas mygtukas „Close“. Taip sukuriamos žymės ir likusiems stereotipams.



3.4 pav. Žymių kūrimas stereotipams

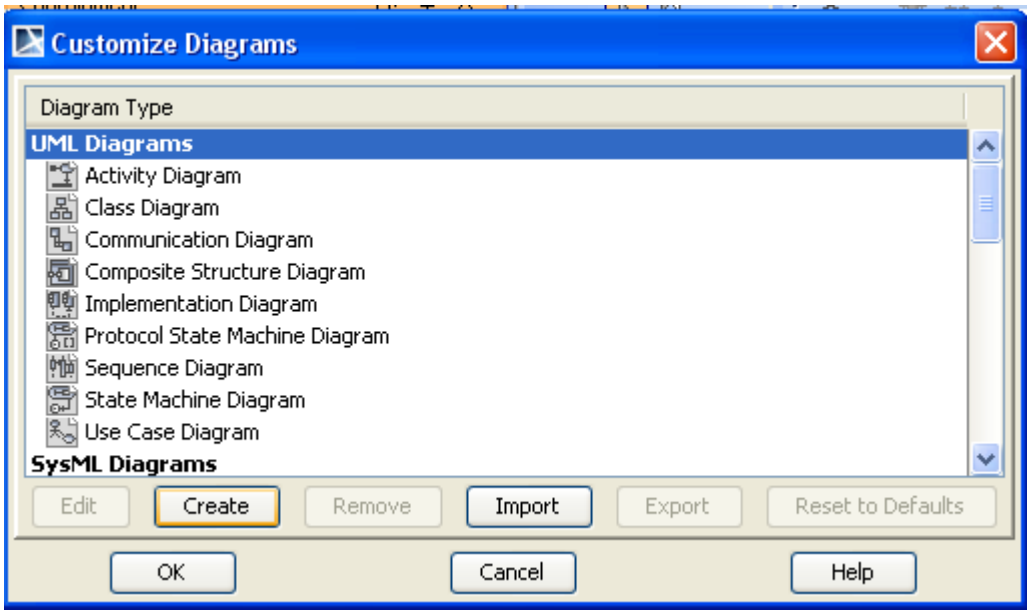
3.4. Naujo tipo diagramos kūrimas

Naujo tipo diagrama naudojant prieš tai susikurtą modulį „Stereotipai“, kuriama spaudžiant mygtuką „Diagrams->Customise...“ kaip parodyta 3.5 paveiksle.



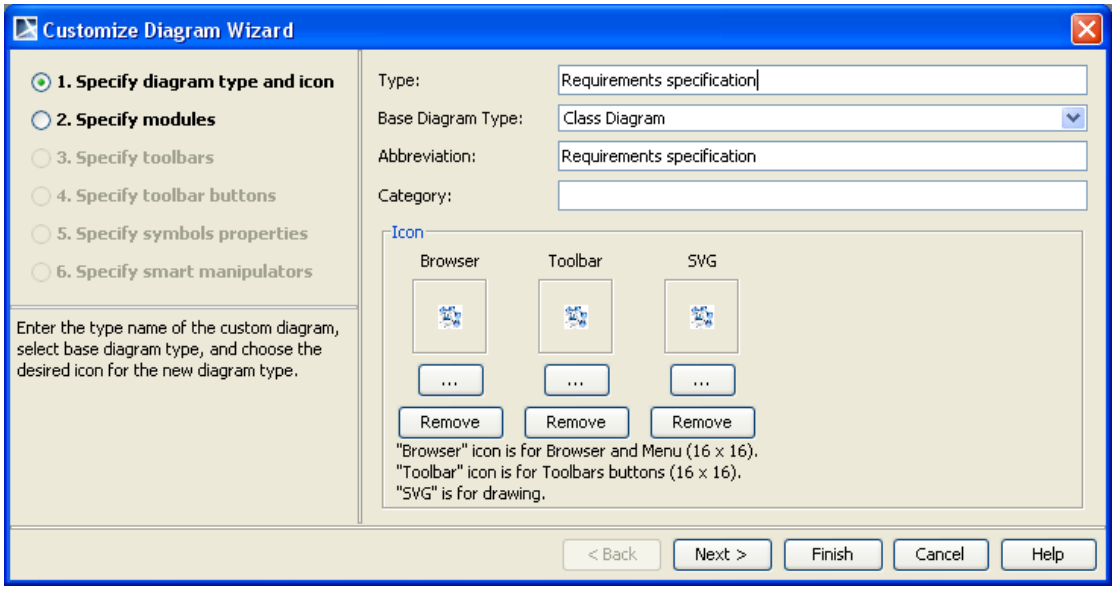
3.5 pav. Naujos diagramos kūrimas

Atsidariusiame lange spaudžiamas mygtukas „Create“ kaip parodyta 3.6 paveiksle.



3.6 pav. Diagramos kūrimo langas

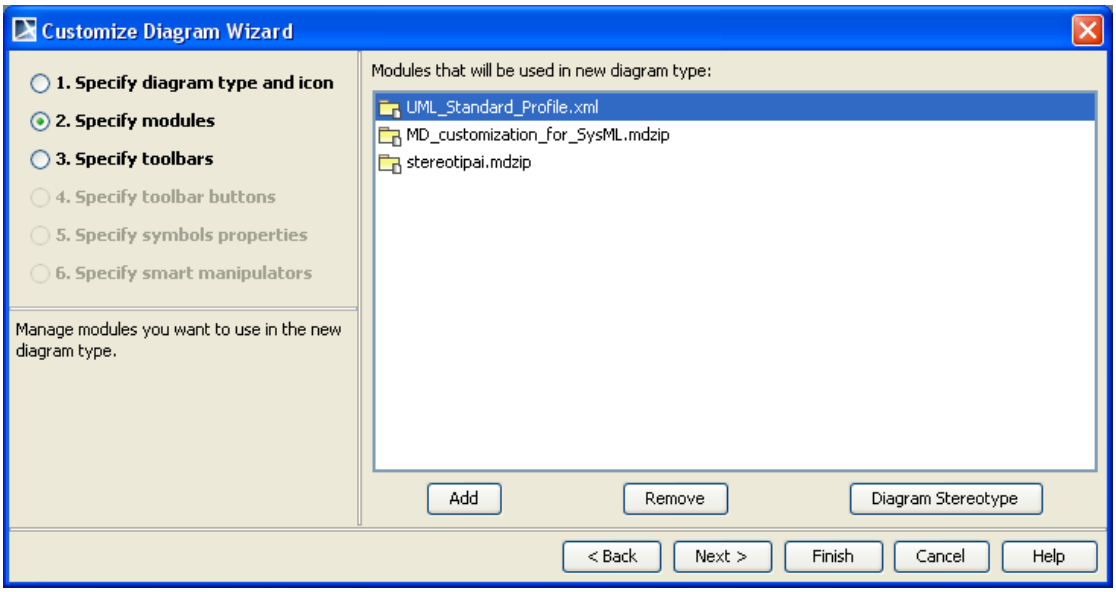
Atsidarys diagramų kūrimo vedlio langas. Visas diagramos kūrimo procesas susideda iš 6 žingsnių. Įvedamas diagramos tipas, bei trumpinys (angl. *Abbrevation*), bei pasirenkamas, koks bus diagramos bazinis tipas (klasių diagrama, sekų diagrama ir t.t.). Taip pat pasirenkama ikona, kuria bus vaizduojama sukurta diagrama. Kuriamos diagramos pirmas žingsnis pateikiamas 3.7 paveiksle.



3.7 pav. Diagramos tipo ir ikonos parinkimas

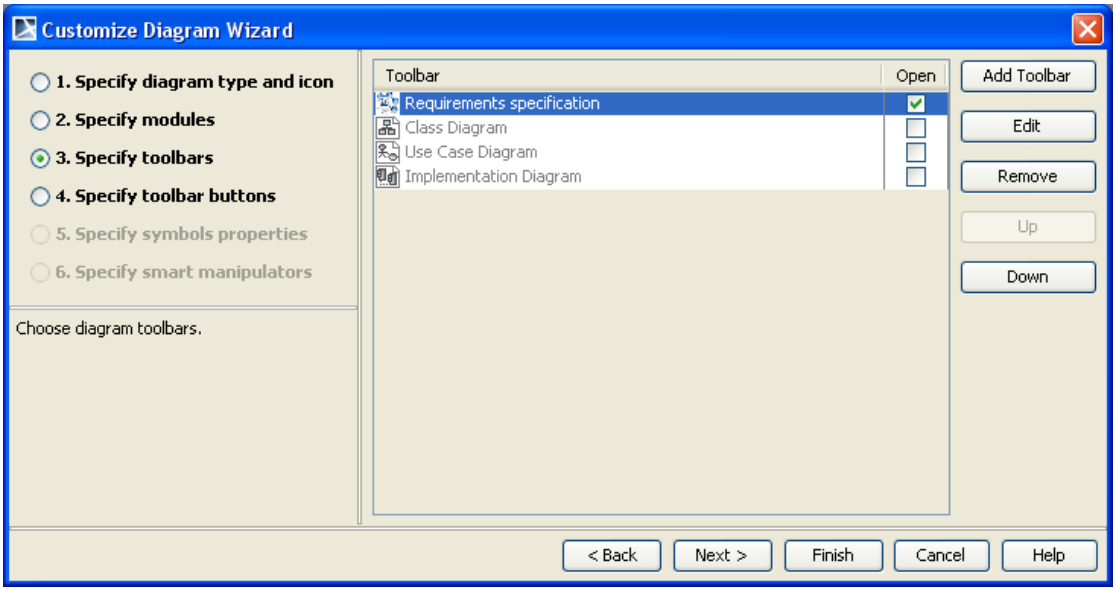
Antrame diagramos kūrimo žingsnyje nurodoma kokie moduliai bus naudojami. Spaudžiant mygtuką „Add“ yra pasirenkamas ankščiau sukurtas

„Stereotipai“ modulis. Analogiškai yra pridedami reikiami moduliai. Diagramos kūrimo antras žingsnis pateiktas 3.8 paveiksle.



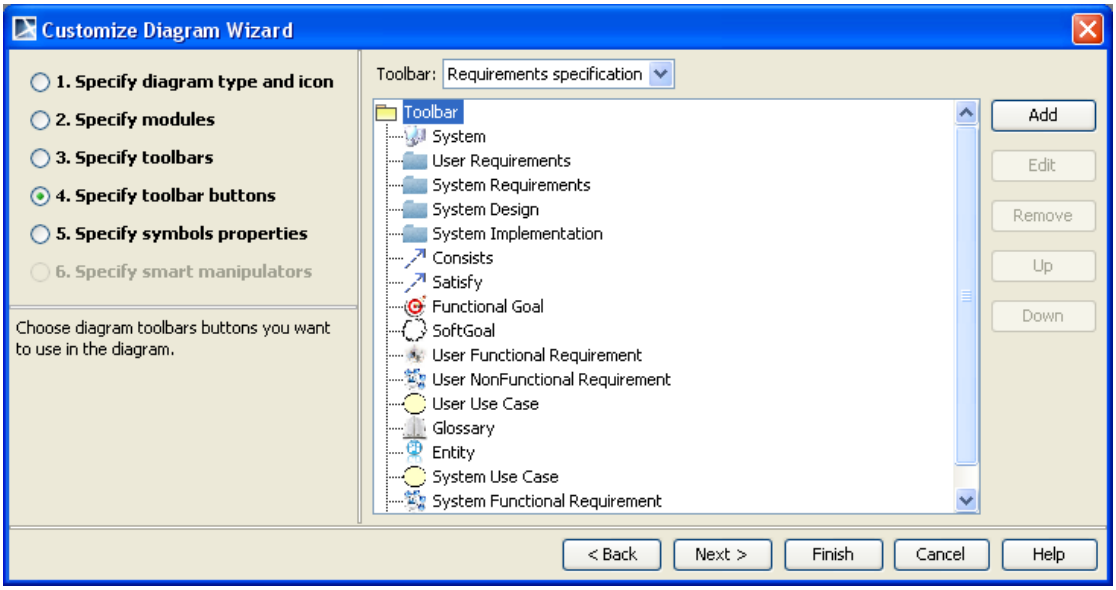
3.8 pav. Modulių pasirinkimas

Trečiame diagramos kūrimo žingsnyje nurodoma kokios dar diagramos be kuriamos bus naudojamos. Spaudžiant mygtuką „Add Toolbar“ yra pasirenkamos diagramos. Mygtukais „Up“ ir „Down“ jos yra išrikiuojamos norima tvarka. Kaip sudarinėjama įrankių juosta parodyta 3.9 paveiksle.



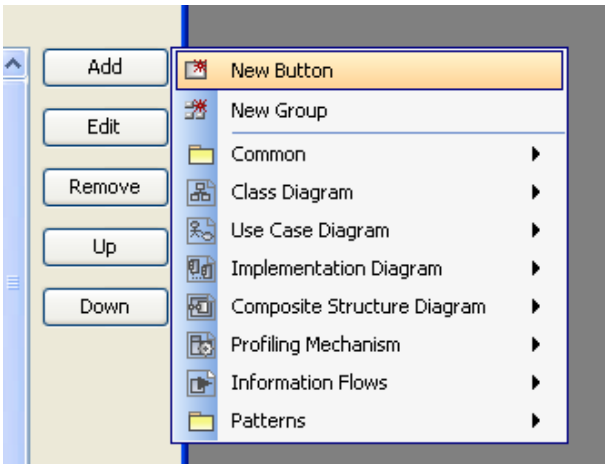
3.9 pav. Įrankių juostos sudarymas

Ketvirtame kūrimo žingsnyje yra kuriami mygtukai įrankių juostoje. Mygtukų kūrimo įrankių juostoje langas pavaizduotas 3.10 paveiksle.

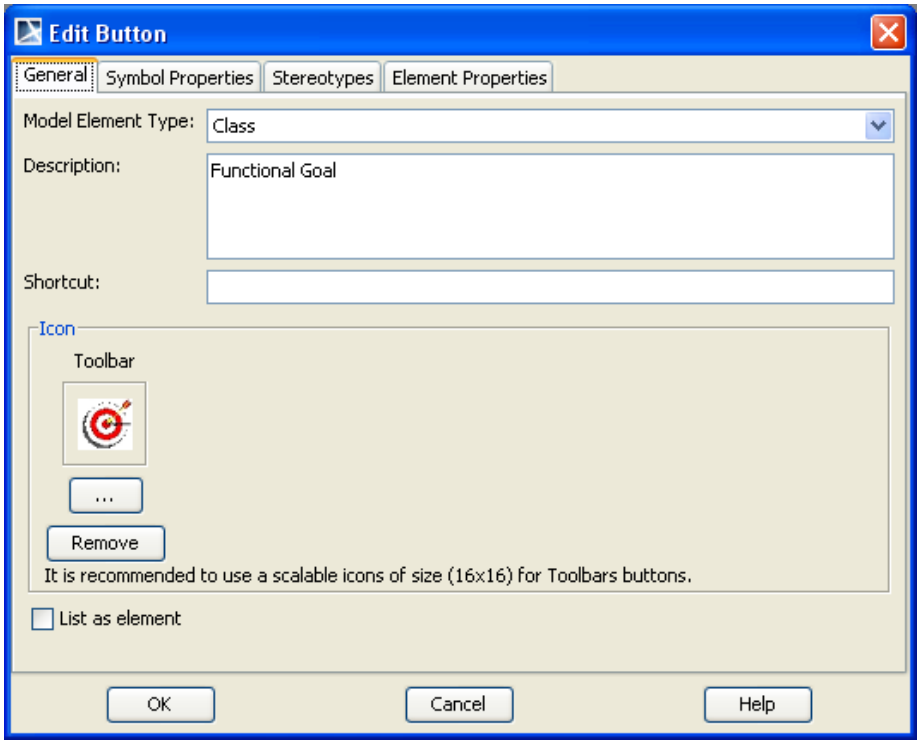


3.10 pav. Mygtukų įrankių juostoje kūrimo langas

Paspaudus mygtuką „Add->New Button“ (3.11 paveikslas) atsidaromas naujas langas (3.12 paveikslas), kuriame kuriami mygtukai naujai kuriamos diagramos įrankių juostai. Ties lauku „Model Element Type“ nurodomas koks bus kuriamo mygtuko tipas, ties „Description“ parašoma kaip jis vadinsis, parenkama jo ikona.

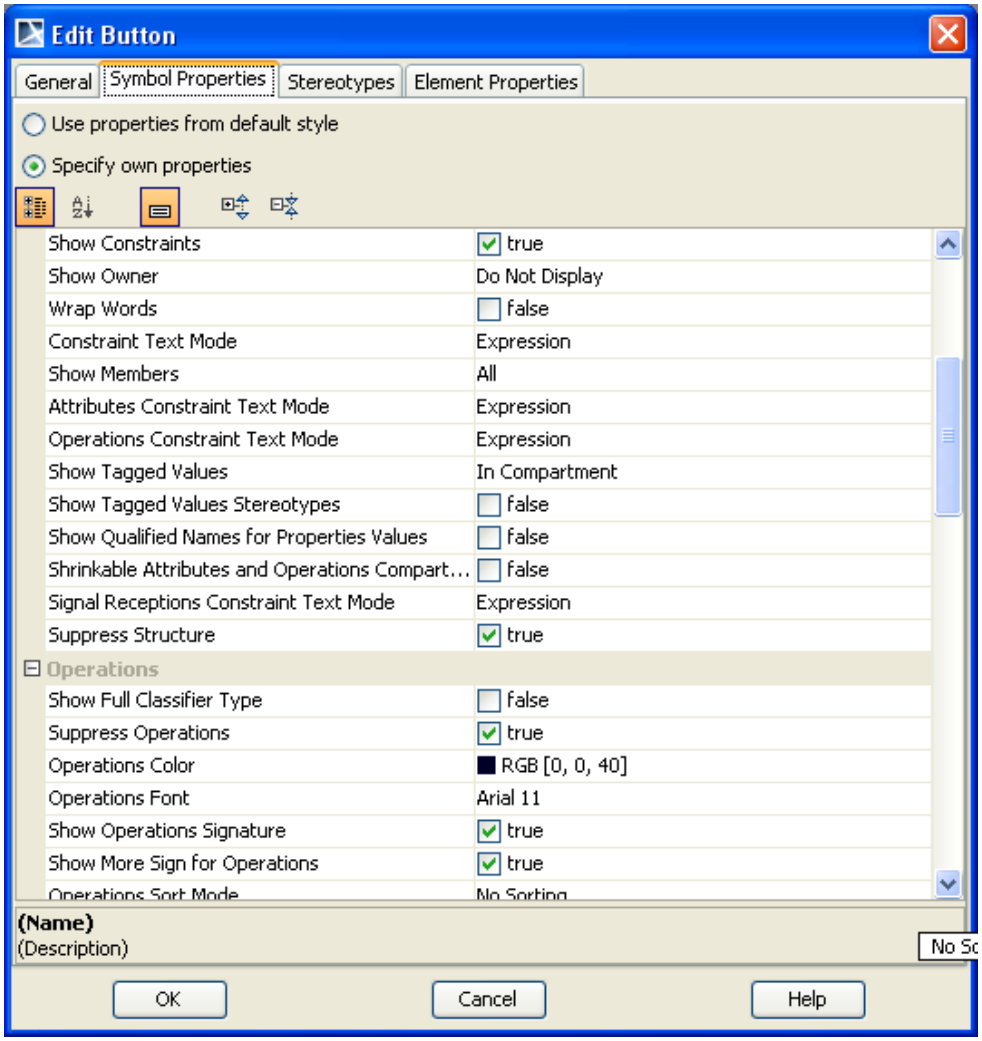


3.11 pav. Naujo mygtuko kūrimas

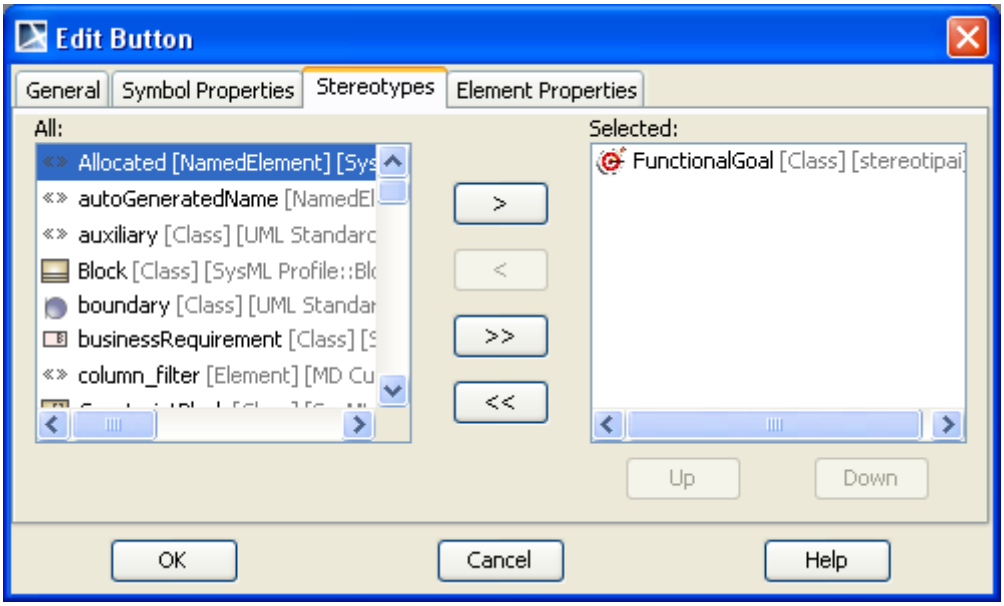


3.12 pav. Mygtuko kūrimo langas

Paspaudus šio lango kitą skiltį „Symbol Properties“ (3.13 paveikslas) galima kuriamam mygtukui nustatyti atskiras savybes, tokias kaip: žymių vaizdavimo būdas „Show Tagged Values“, atributų bei operacijų vaizdavimo būdas ir t.t. Trečioje lango skiltyje „Stereotypes“ (3.14 paveikslas) nurodomas stereotipas, kuris bus taikomas kuriamam mygtukui. Čia vaizduojami baziniai Magic Draw UML stereotipai, bei tie stereotipai, kurių modulį pasirinkome antrame žingsnyje (3.8 paveikslas).

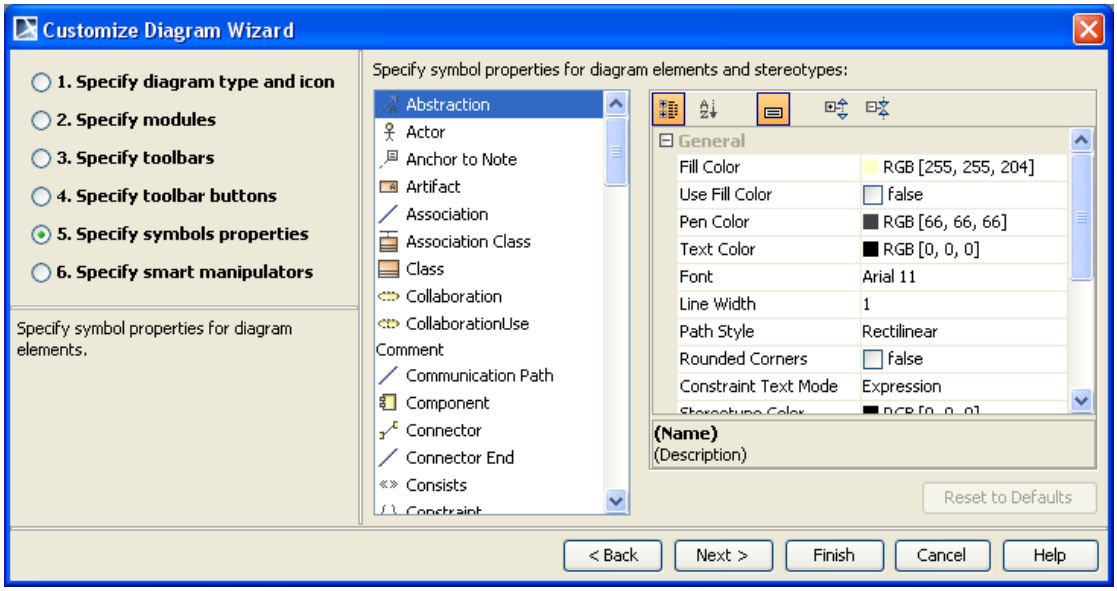


3.13 pav. Simbolio savybių langas



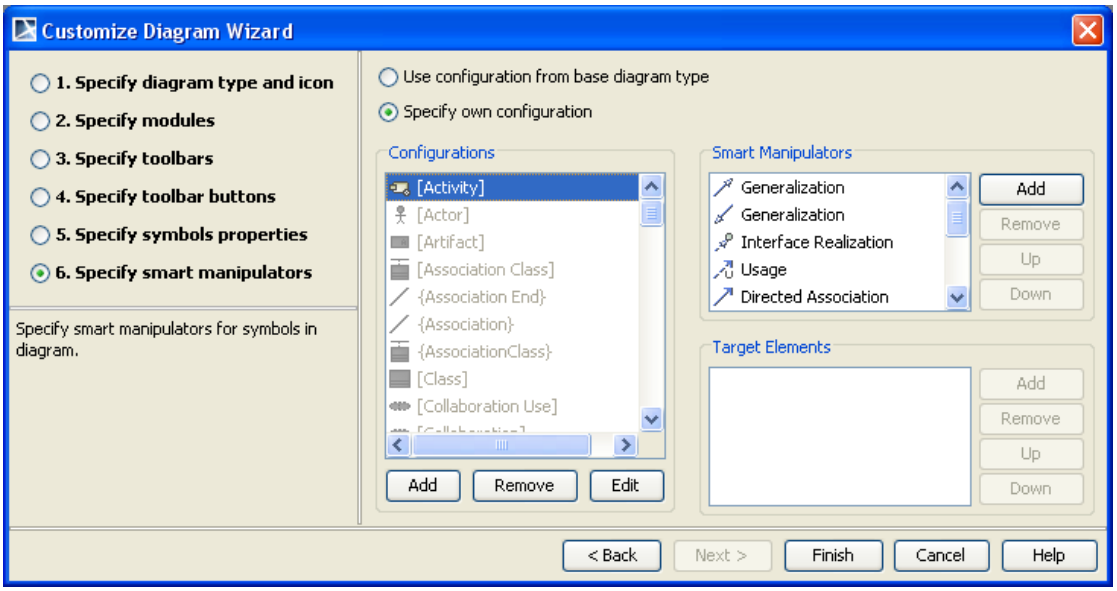
3.14 pav. Stereotipo priskyrimas

Penktame kūrimo žingsnyje galima nustatyti specialias simbolių savybes, kaip pavaizduota 3.15 paveiksle.



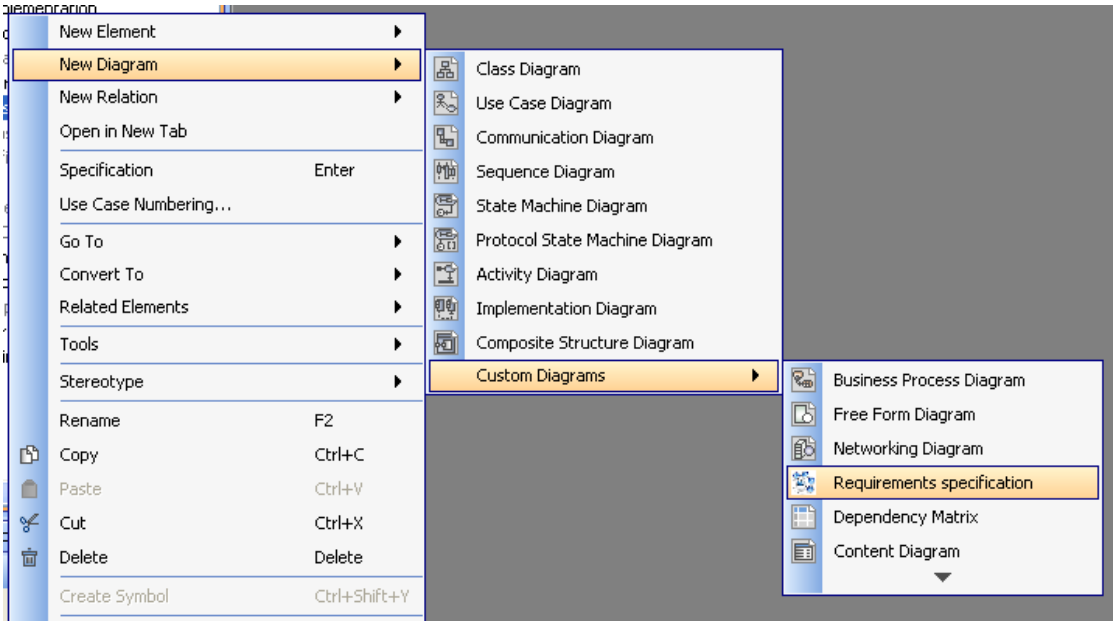
3.15 pav. Simbolių savybių nustatymo langas

Paskutiniame šeštame kūrimo etape (3.16 paveikslas), galima nustatyti elementam greitas manipuliacijas. Tam, kad padarytumėm simbolių vaizdavimą greitesnį, reikia nurodyti, kokio tipo ryšiai bus siūlomi jungti, kai atitinkamo tipo elementas pažymimas diagramoje. Tai padarius spaudžiamas mygtukas „Finish“, kurio paspaudimu užbaigiamas diagramos kūrimo procesas.



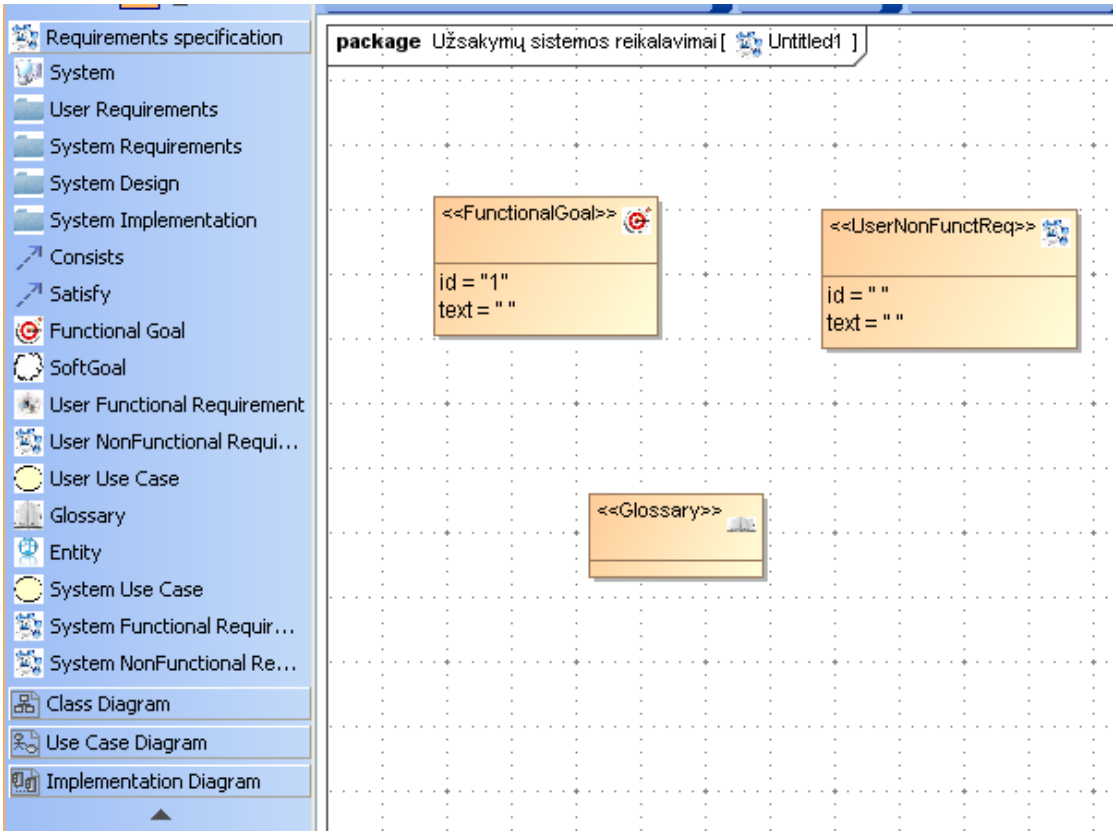
3.16 pav. Greitų manipuliacijų nustatymo langas

Tam, kad sukurtą diagramą būtų galima naudoti reikia išjungti ir iš naujo įjungti Magic Draw UML programą. Norint pasinaudoti sukurta diagrama spaudžiame kurti naują diagramą, kaip parodyta 3.17 paveiksle.



3.17 pav. Naujos diagramos kūrimas naudojant sukurtą diagramą

Atidaromas langas, kuriame galima kurti diagramas, naudojant sukurtos diagramos elementus (3.18 paveikslas).



3.18 pav. Diagramų kūrimas naudojant sukurtos diagramos elementus

4. Reikalavimų sekimo profilio eksperimentinis tyrimas

Darbo metu buvo sukurta tikslais grindžiamų reikalavimų sekimo metodika. Norint patikrinti metodikos funkcionalumą bei reikalavimų atsekamumą, reikia ištestuoti šią metodiką. Testavimui buvo pasirinktas N. Vaičiukyno magistro darbas tema: „Elektroninis aukcionas: dinamiški veiklos procesai internete“ [21]. Buvo suprojektuota ir sukurta realiai veikianti internetinio aukciono sistema. Testuosime sukurtą metodiką, nagrinėdami šį darbą, ir žiūrėsime, kaip yra tenkinami reikalavimai įvairiuose darbo etapuose ir ar įmanomas reikalavimų atsekimas.

4.1. Tikslų ir reikalavimų atsekamumas

Iš pradžių reikia nustatyti, kokie yra vartotojų tikslai. Geriausia tam tiktų tikslų diagrama, bet [21] dokumentacijoje tokios diagramos nepavyko rasti. Todėl bandysime tikslus identifikuoti pagal informaciją iš konteksto. Darbo apraše buvo rasta, kad:

„*Realaus laiko informacinėse sistemose pagrindiniai vartotojų tikslai būna tipiški: tai viso kompiuterizuojamo proceso vykdymo stebėjimo ir valdymo galimybės. Kaip pavyzdys detaliau bus pateiktas realaus laiko aukciono informacinės sistemos vartotojų tikslai ir problemos.*

Pagrindinės sprendžiamos problemos yra susijusios su sistemos dinamiškumu, todėl tai atsispindi šios sistemos vartotojų tiksluose.

Aukciono vykdytojas reikalavimai sistemai:

- *Realaus laiko aukciono valdymas:*
  - *aukciono laikrodžio paleidimas,*
  - *laikrodžio sustabdymas,*
  - *pakartotinas paleidimas.*
- *Realaus laiko vartotojo pirkimų sekimas: aukciono turi matyti, kokios ir kiek prekių yra nupirktos, aukciono būseną.*
- *Stebėti procesą realiu laiku ir gauti pranešimus, susijusius su verslo logika.*“

Labiausiai yra akcentuojami funkciniai tikslai o apie nefunkcinius mažai kalbama. Bet buvo minima, kad kiti egzistuojantys aukcionai veikia lėtai, informacija nėra atnaujinama greitai, o kuriamas aukcionas turi būti dinamiškas, vykti realiu laiku

ir vartotojai greitai gautų atnaujintą informaciją, tai buvo iškeltas nefunkcinis tikslas „Pardavimo proceso pagreitinimas“. Buvo sudarytas tikslų modelis, kuris pateiktas 2.3 paveiksle.

Tam, kad būtų ištestuota reikalavimų sekimo metodika, buvo paimtas vienas tikslas ir žiūrima, kaip jis realizuojamas visuose sistemos kūrimo etapuose. Dėl to buvo pasirinktas funkcinis tikslas „Aukciono valdymas“ bei nefunkcinis tikslas „Pardavimo proceso pagreitinimas“ ir stebima, kaip tie tikslai pereina į reikalavimus, kurie savo ruoštu atsivaizduoja kituose sistemos elementuose.

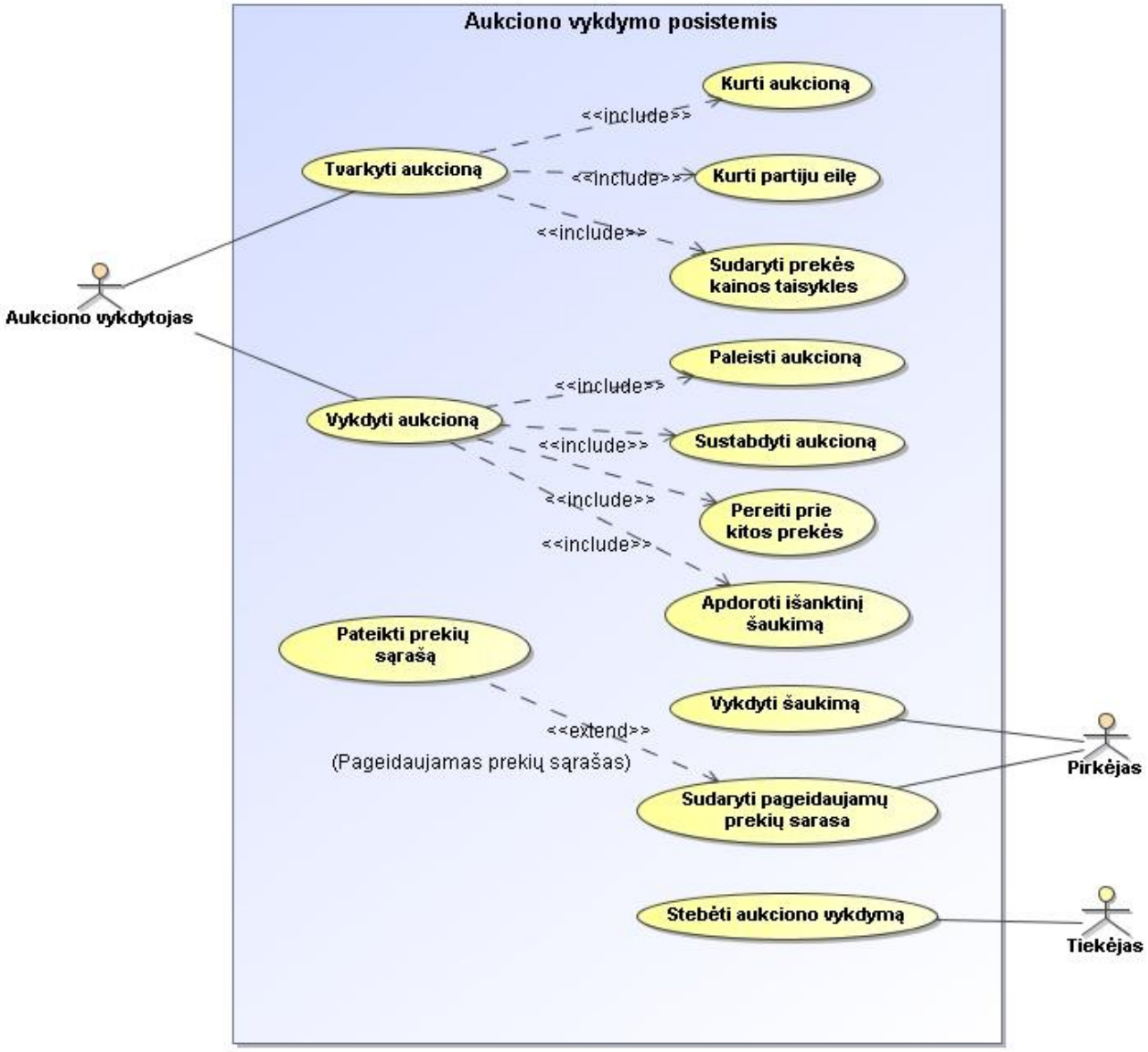
Buvo apibrėžti aktoriai, bei nurodyti reikalavimai. Aktorius „Aukciono vykdytojas“ turi galimybę tvarkyti aukciono duomenis („Tvarkyti aukcioną“) ir valdyti bei stebėti aukcioną („Vykdyti aukcioną“).

”

- *Aukciono vykdytojo turi turėti galimybę tvarkyti aukciono duomenis:*
  - *Sukurti aukcioną, nustatyti jo pradžią ir pabaigą;*
  - *Įtraukti prekių partijas į aukcioną;*
  - *Nustatyti partijų pardavimo eilę;*
  - *Sudaryti prekės kitimo aukciono vykdymo metu taisykles ;*
  - *Keisti aukciono duomenis.*
- *Aukciono vykdytojas turi turėti galimybę valdyti ir stebėti aukcioną:*
  - *Paleisti aukciono vykdymą (paleisti laikrodį);*
  - *Stabdyti aukciono laikrodį (stabdyti aukcioną);*
  - *Gauti aukciono vykdymo būsenas;*
  - *Stebėti aukciono vykdymo būseną;*
  - *Matyti aukciono partijos išankstinius pirkimus.*

“

Pagal iškeltus funkcionalumo reikalavimus turi būti sudaryta panaudojimo atvejų diagrama. Panaudojimo atvejų diagrama pateikta 4.1 paveiksle.



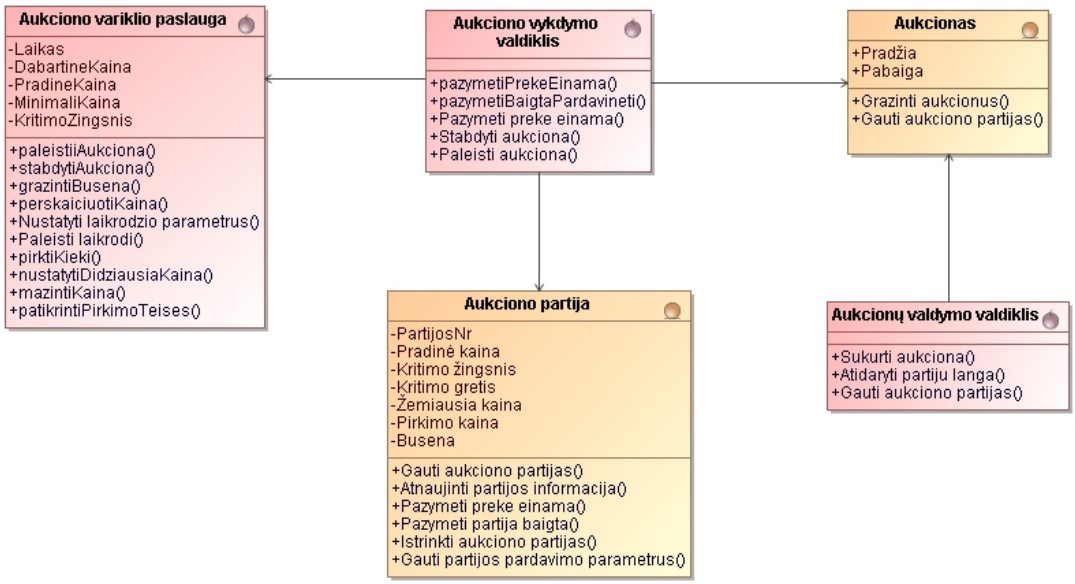
4.1 pav. Panaudojimo atvejų diagrama

Matome, kad išsikelti reikalavimai ir pavaizduoti reikalavimai diagramoje nesutampa, todėl jau pradinėje sistemos kūrimo stadijoje reikalavimai yra pametami. Toliau nagrinėsime tik tuos panaudojimo atvejus, kurie yra pateikti diagramoje (4.1 paveikslas) ir vykdomi aktoriaus „Aukciono vykdytojas“. Kaip panaudojimo atvejai tenkina tikslus pavaizduota 2.8 paveiksle. Nefunkcinis tikslas arba (angl. *Soft Goal*) atsivaizduoja i nefunkcinį reikalavimą, taip kaip pavaizduota 2.9 paveiksle.

4.2. Funkcinių reikalavimų vaizdavimas valdikliuose

Funkcinio tikslo „Aukciono valdymas“ funkciniai reikalavimai turi būti pavaizduoti „Sistemos projekto“ dalyje atitinkamuose valdikiuose. 4.2 paveiksle

pateikta aukciono valdymo posistemio vartotojo veiklos paslaugų klasių diagrama. Joje yra „Aukciono vykdymo valdiklis“, kuris turi savo operacijas. „Aukciono vykdymo valdiklis“ realizuoja panaudojimo atvejį „Vykdyti aukcioną“. Todėl operacijos turi atkeliauti iš panaudojimo atvejį „Vykdyti aukcioną“ atitinkančių reikalavimų. Atitinkantys funkciniai reikalavimai yra: „Paleisti aukcioną“, „Sustabdyti aukcioną“, „Pereiti prie kitos prekės“, „Apdoroti išankstinį šaukimą“. Kaip matome, valdiklyje funkcijos nesutampa su reikalavimais. Kaip teisingai turi būti sudarytos „Aukciono valdymo valdiklio“ operacijos pavaizduota 2.13 paveiksle.



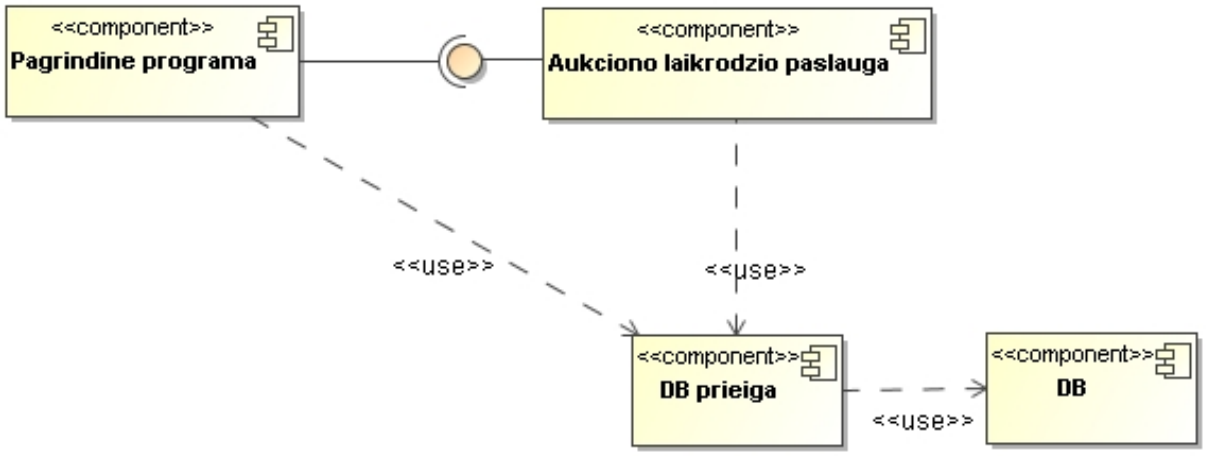
4.2 pav. Aukciono valdymo posistemio vartotojo veiklos paslaugų klasių diagrama

Kiekviena valdiklio operacija turi atitikti sistemos reikalavimą. 2.13 paveiksle pavaizduotoje diagramoje yra „Aukciono vykdymo valdiklis“, kuris turi tam tikras operacijas. Šios operacijos turi ateiti iš sistemos funkcinių reikalavimų. 2.14 paveiksle parodyta, kaip sistemos reikalavimai turėtų būti pavaizduoti projekte, t.y. valdiklio operacijos sudaromos pagal atitinkamus sistemos reikalavimus.

4.3. Nefunkcinių reikalavimų atsivaizdavimas sistemos komponentuose

Sistemos funkciniai reikalavimai „Sistemos projekto“ dalyje vaizduojami valdiklių operacijomis. Sunkiau atsekti nefunkcinius reikalavimus, kadangi daugelis jų tiesiogiai į projektą nevaizduojami. Tačiau kai kuriuos nefunkcinius reikalavimus galima pavaizduoti į sistemos komponentus. Taip būna tais atvejais, kai

nefunkciniams reikalavimams užtikrinti taikomos architektūrinės priemonės. Pavyzdžiui, aukciono sistemoje veikimui pagreitinti naudojamas Ajax komponentas (variklis). 2.22 paveiksle matosi, kaip vartotojo nefunkcinis reikalavimas „Pardavimu pagreitinimas“ tampa sistemos nefunkciniu reikalavimu, kuris savo ruožtu projekto dalyje realizuojamas „Aukciono variklio paslauga“ ir galiausiai realizacijos dalyje šis reikalavimas realizuojamas komponentu „Aukciono variklio paslauga“. Eksperimento metu sekant nefunkcinį reikalavimą „Pardavimu pagreitinimas“ pastebėta, kad jis turėjo būti realizuotas sistemos dalyje „Aukciono variklio paslauga“ ir galiausiai realizacijoje atitinkamu komponentu. Kaip matome pateiktoje komponentų diagramoje (4.3 paveikslas), tokio komponento kaip „Aukciono variklio paslauga“ nėra, o tai leidžia daryti išvadą, kad šis nefunkcinis reikalavimas buvo neišpildytas. Kaip turėtų būti realizuotas nefunkcinis reikalavimas, pateikta 2.22 paveiksle.



4.3 pav. Komponentų diagrama

4.4. Eksperimentų rezultatų apibendrinimas

Atlikus eksperimentinį tyrimą ir pažiūrėjus kaip yra realizuojami ir atsekami reikalavimai, elektroninio aukciono sistemoje paaiškėjo, kad nors elektroninio aukciono sistema buvo realizuota gerai, darbas apgintas ir įvertintas puikiai, bet pagal aprašymą galima daryti išvadą, kad iškelti reikalavimai nebuvo realizuoti. Realizuotos sistemos projekto dokumentacijos analizė parodė, kad:

- reikalavimai neatitinka aprašytų tikslų;
- kai kurie funkciniai reikalavimai dingsta;

- neaišku, ar realizuojami nefunkciniai reikalavimai, kodėl atsiranda kai kurie projekto elementai.

Galima daryti išvadą, kad dokumentacijos kokybė nėra gera, projektą sunku įvertinti ir pagal šią dokumentaciją vertinti, testuoti, integruoti ar palaikyti programinę įrangą būtų sunku.

Dėl to reikalavimų sekimo metodikos sudarymas yra svarbus dalykas, kuris leidžia išvengti šių trūkumų.

Ji ypač reikalinga kuriant dideles sistemas, kai kūrime dalyvauja daug vykdytojų.

4.1 lentelėje pavaizduotas analizės dalyje nagrinėtų metodų palyginimas su sukurta metodika.

4.1 lentelė Metodų palyginimo lentelė

Palyginimo kriterijai	VOLERE	RUP	TROPOS	KAOS	Sukurta metodika
Ar nagrinėja tikslus?	–	–	+–	+	+
Ar nagrinėja tikslu priklausomybes?	–	–	–	+	+
Ar išveda reikalavimus iš tikslu?	–	–	–	+–	+
Ar išskaido reikalavimus iki atominių vienetų?	+	–	–	–	+
Ar turi šablonus reikalavimams specifikuoti?	+	+	–	–	+

5. Išvados

1. Planuojant informacinių sistemų projektus, tikslinga analizuoti tikslus, nes taip galima padidinti informacinių sistemų tinkamumą veiklos poreikiams.

2. Dėl reikiamų modeliavimo priemonių CASE įrankiuose nebuvimo tikslų modeliavimas praktiniuose kūrimo procesuose yra neatliekamas arba atliekamas tik dalinai.

3. Todėl šiame darbe sukurtas tikslais grindžiamų reikalavimų UML profilis, kuris leidžia vaizduoti tikslus ir su jais susijusius panaudojimo atvejus bei detalizuotus reikalavimus.

4. Kuriant reikalavimų specifikavimo metamodelį, sujungti standartiniai RUP proceso, Volere reikalavimų specifikavimo šablono bei KAOS tikslų modeliavimo elementai.

5. Sukurto profilio, realizuoto UML CASE įrankio Magic Draw DSL kūrimo priemonėmis, taikymas pavyzdinei sistemai modeliuoti parodė, kad jis leidžia paprastai, nenaudojant papildomų paketų sumodeliuoti tikslus ir susieti juos su detaliai specifikuotais reikalavimais ir vėlesnių stadijų artefaktais.

6. Eksperimentinis praktikoje įdiegtos sistemos reikalavimų specifikacijos ir projekto tyrimas parodė, kad net nedideliam projekte yra daugybė neatitikimų ir praktiškai sunku atsekti, ar realizacija atitinka pradinius tikslus ir reikalavimus.

7. Taikydami sudarytą metodiką, analitikai, projektuotojai ir kūrėjai galėtų tiksliai susieti savo kuriamus artefaktus su ankstesnių stadijų artefaktais, o tai palengvintų projektų vertintojų, testuotojų ir integruotojų darbą ir pagerintų projektų kokybę.

8. Darbo tema buvo parašytas ir atspausdintas straipsnis, kuris buvo pristatytas konferencijoje „IT2008“.

6. Literatūra

[1] Adolph S., Bramble P. Patterns for effective use cases – 250 p.

[2] Booch, G., Maksimchuk, R. A., Engle, M. W., Young, B. J., Conallen, J., Housto, K. A. (2007) Object-Oriented Analysis and Design with Applications. Addison-Wesley.

[3] Booch, G., Rumbaugh, J., Jacobson, I. (2005) The Unified Modeling Language User Guide. Second edition, Addison-Wesley Professional.

[4] Bresciani P., Sannicolo F. Applying Tropos Early Requirements Analysis for defning a Tropos tool. ITC-irst Via Sommarive, 18, I-38050 Trento-Povo, Italy. - 4 p.

[5] Butkiene R. Paskaitų skaidrės [interaktyvus] [žiūrėta 2007.12.15]. Prieiga per internetą: ftp://isd.ktu.lt/Isd/Butkiene/T000M109/T000M109_5.pdf

[6] Butleris R., Danikauskas T. Reikalavimų specifikavimo ORACLE CASE terpėje plėtra [interaktyvus] [žiūrėta 2008.01.05]. Prieiga per internetą: www.leidykla.vu.lt/inetleid/inf-mok/19/str6.html

[7] Darimont R., Delor E., Rifaut A. Quality Starts with the Definition of Goals [interaktyvus] [žiūrėta 2007.12.16] Prieiga per internetą: http://www.objectiver.com/download/documents/papers/archives/QWE2001.pdf

[8] Donald G. Firesmith Modern Requirements Specification [interaktyvus] [žiūrėta 2007.10.05] Prieiga per internetą: http://www.jot.fm/issues/issue_2003_03/column6.

[9] Durán Toro A., Bernárdez Jiménez B., Ruiz Cortés A., Toro Bonilla M. A Requirements Elicitation Approach Based inTemplates and Patterns - 13 p.

[10] Giorgini P., Giunchiglia G., Gylopoulos J.Tropos:An Agent-Oriented Software Development Methodology.2004 Kluwer Academic Publishers-34 p.

[11] Heaven W. and y Finkelstein A. A UML profile to support requirements engineering with KAOS [interaktyvus] [žiūrėta 2007.12.15] Prieiga per internetą: http://www.cs.ucl.ac.uk/staff/A.Finkelstein/papers/umlreprofile.pdf

[12] James & Suzanne Robertson Volere Requirements Specification Template [interaktyvus] [žiūrėta 2007.12.03]. Prieiga per internetą: http://www.systemsguild.com/GuildSite/Robs/ Template.html

[13] Kruchten, P. (2003) Rational Unified Process, The: An Introduction. Third Edition. Addison Wesley.

[14] Nemuraitė L. Rational Unified Process (RUP) Unifikuotas projektavimo procesas. [interaktyvus] [žiūrėta 2007-09-20]. Prieiga per Internetą: ftp://isd.ktu.lt/isd/Nemuraite/ Moduliai/T120B609-2007/ISP2007-5.ppt

[15] Objectiver's methodology : KAOS the Goal-Driven Requirements Engineering Method [interaktyvus] [žiūrėta 2007-10-07]. Prieiga per Internetą: http://www.objectiver.com/en/ documentation/kaos/

[16] Requirements Modeling [interaktyvus] [žiūrėta 2007-10-05]. Prieiga per Internetą: http://www.volere.co.uk

[17] Robertson J., Robertson S. Volere Requirements Specification Template. 1995-2001 Atlantic Systems Guild. -52 p.

[18] RUP [interaktyvus] [žiūrėta 2007-10-07]. Prieiga per Internetą: http://en.wikipedia.org/wiki/Rational_Unified_Process

[19] Silingas D. Defining Domain Specific Languages with UML Profiles. Dr.Dobb's Architecture Design World, Chicago, July

[20] XI sekcija Veiklos procesų ir informacinių procesų analizė [interaktyvus] [žiūrėta 2007.12.20] Prieiga per internetą: www.ktu.lt/lt/apie_renginius/konferencijos/2006/k6_02/ IT2002/XIsekcija.pdf

[21] N. Vaičiukynas. Elektroninis aukcionas: dinamiški veiklos procesai internete. Magistro darbas. Kauno technologijos universitetas, 2008 m.

7. Terminų ir santrumpų žodynas

7.1 lentelėje pateikiamas darbe naudojamų terminų ir santrumpų žodynas.

7.1 lentelė Terminų ir santrumpų žodynas

Santrumpa, terminas	**Paaiškinimas**
UML (angl. *Unified Modeling Language*).	– unifikuota modeliavimo kalba
CASE (angl. *Computer Aided Software Engineering*)	– integruota IS kūrimo aplinka
RUP (angl. *Rational Unified Proces)*	– unifikuotas projektavimo procesas
DSL (angl.D*omain-Specific Language*)	– dalykinės srities modeliavimo kalba
SysML (angl. *System Modeling Language*)	– sistemų modeliavimo kalba.
PĮ	– programinė įranga
IS (angl. *Information System*)	– informacijos sistema.
DBVS	– duomenų bazių valdymo sistema
MDD (angl. *Model Driven Development*)	– modeliais grindžiamas kūrimas.

8. Priedai

1 priedas. Straipsnis tema: “ Tikslais grindžiamų reikalavimų specifikavimo metodų analizė“

Tikslais grindžiamų reikalavimų specifikavimo metodų analizė

Egidijus Kumža
Kauno Technologijos universitetas, Informacijos sistemų katedra
Studentų g. 50, Kaunas

Daugelyje organizacijų aktuali informacinių technologijų atitikimo veiklos poreikiams problema. Vienas iš būdų ją spręsti – susieti programinės įrangos reikalavimų specifikacijas su veiklos tikslais. Reikalavimų specifikavimo metodų yra, bet nedaugelis iš jų siejasi su tikslais. Straipsnyje analizuojami tikslų modeliavimo ir reikalavimų specifikavimo metodai. Atrinkus geriausias esamų metodų savybes, siekiama sudaryti papildytą reikalavimų specifikavimo šabloną, siejantį reikalavimus su tikslais, bei pritaikyti šį šabloną plačiai naudojamam CASE įrankiui.

Įvadas

Pastaruoju metu kuriant informacines sistemas ir siekiant jas geriau pritaikyti verslo poreikiams, vis dažniau kalbama apie tikslų analizę. Informacinių sistemų projektuose tikslinga analizuoti tikslus, tačiau tikslų modeliavimas praktikoje kol kas nėra plačiai paplitęs. Jis labiau nagrinėjamas mokslinėse publikacijose. Viena iš priežasčių yra ta, kad esami CASE įrankiai neturi reikiamų tikslų modeliavimo priemonių.

Todėl straipsnyje analizuojami keli žinomiausi tikslų modeliavimo bei reikalavimų specifikavimo metodai, siekiant atrinkti geriausias jų savybes ir sudaryti naują arba papildyti jau esamą reikalavimų specifikavimo šabloną, kuris būtų grindžiamas tikslais. Šios analizės pagrindu bus pritaikytas ar sudarytas naujas reikalavimų šablonas, kuris turėtų būti pakankamai išsamus, susietas su tikslais ir įgyvendintas plačiai naudojamame CASE įrankyje.

Yra gana daug šablonų, naudojamų reikalavimų rinkime ir specifikavime. Bet nedaugelis jų siejasi su tikslais. Toliau analizuojami pagrindiniai plačiai naudojami reikalavimų specifikavimo ir tikslų modeliavimo metodai (*Volere* [7], RUP [6], *Tropos* [1, 4], *Kaos* [5], *SysML* [8]).

Volere reikalavimų specifikavimo šablonas

Volere šablonas naudojamas pradiniame sistemos kūrimo etape ir yra pagrindas vartotojų reikalavimams specifikuoti. Šablonas suskirstytas į skyrius pagal reikalavimų tipus. Jis padeda surinkti reikalavimus, gaunamus vartotojų apklausos metu arba analizuojant analizuojamo objekto veiklą reglamentuojančią dokumentaciją. Tai atviras šablonas, kurį galima pritaikyti konkretiems atvejams. Šablono skyrių, kurie netinka nagrinėjamam objektui, galima nenaudoti, arba galima sukurti naujus skyrius, tinkamus specifikuoti specifines dalykinės srities charakteristikas [2]. *Volere* šablonas apima šiuos reikalavimų tipus [7]:

- funkciniai reikalavimai (angl. *Functional requirements*);
- nefunkciniai reikalavimai (angl. *Nonfunctional requirements*);
- projekto apribojimai (angl. *Project constraints*);
- projekto veiksniai (angl. *Project drivers*);
- projekto rezultatai (angl. *Project issues*);
- testavimo reikalavimai (angl. *Testing requirements*).

Naudojant *Volere* šabloną reikalavimams aprašyti, vadovaujamasi susisteminto proceso apribojimais, nusakančiais, kokio etapo metu ir kokie reikalavimai turi būti aprašyti. Kai kuriuose proceso etapuose naudojamas specializuotas apklausos lapas, padedantis surinkti reikalavimus, juos analizuoti ir apibendrinti [2]. *Volere* šablone stambesni reikalavimų vienetai yra panaudojimo atvejai, kurie išskaidomi į atskirus reikalavimus, atitinkančius įvykius. *Volere* reikalavimų šablono privalumas yra tas, kad išskiriami atominiai reikalavimų elementai, kurie gali būti tiek funkciniai, tiek

nefunkciniai. Tai leidžia atsekti kiekvieną atominį reikalavimą. Tačiau *Volere* šablone nėra specifikuojami panaudojimo atvejų scenarijai.

Standartinis kūrimo procesas RUP

Rational Unified Proces (RUP) yra panaudojimo atvejų valdomas, architektūra grindžiamas, iteracinis, vykdomas palaipsniui, riziką mažinantis projektavimo procesas [6]. RUP veikimas yra pagrįstas spiralės metodo principu. RUP procesas susideda iš keleto etapų ir iteracijų. Yra keturios RUP gyvavimo ciklo fazės:

- pradžia (angl. *Inception*);
- parengimas (angl. *Elaboration*);
- konstravimas (angl. *Construction*);
- perėjimas (angl. *Transition*).

Tai yra vienas iš populiariausių ir dažniausiai naudojamų projektavimo metodų. RUP reikalavimų šablono struktūra kiek kitokia, nei *Volere*. RUP šablono esmė yra panaudojimo atvejų scenarijai. Tačiau reikalavimų vienetai yra panaudojimo atvejai, kurie gali būti skirtingo stambumo, todėl reikalavimų atsekimas nėra toks akivaizdus.

Tropos programų inžinerijos metodika

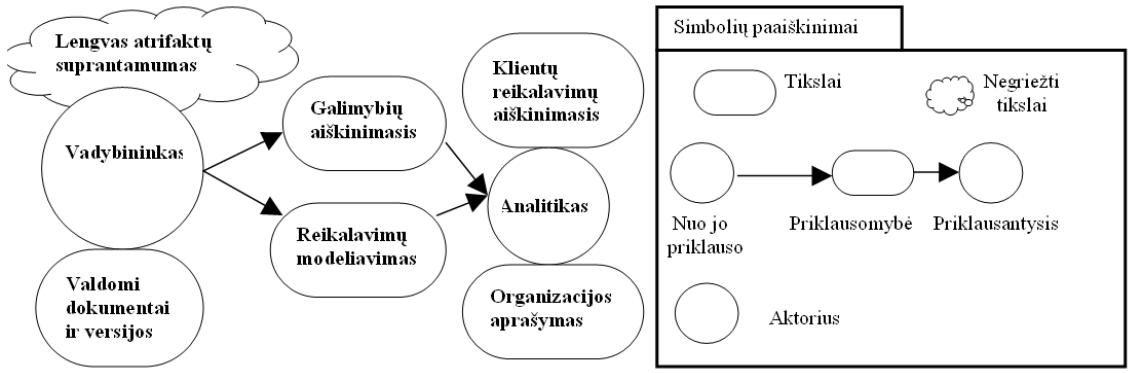
Tropos yra nauja, į agentus orientuota (*agent-oriented*) programinės įrangos inžinerijos metodologija, charakterizuojama trimis aspektais [1]: daugiausia dėmesio skiriama veikloms, kurios specifikuoja nusistovėjusius reikalavimus, siekiama suvokti, kaip numatytoji sistema turi sąveikauti su organizacijos tikslais; ji siejasi su visomis sistemos reikalavimų analizės, sistemos projektavimo bei kūrimo fazėmis bendra notacija, apimančia aktorius, tikslus, potikslius, planus, išteklius ir priklausomybes; metodologija remiasi būsimos sistemos modelio kūrimo idėja, kuri plečiama nuo koncepcinio lygmens iki vykdomų artefaktų.

Vienas iš *Tropos* metodologijos privalumų yra tas, kad yra ne tik keliami klausimai *kokia* ir *kaip*, bet ir *kodėl* programinė įranga yra kuriama. *Tropos* metodologijoje programinės įrangos kūrimo procese numatyta palaikyti visas analizės ir projektavimo veiklas [4, 1]. *Tropos* turi penkias programinės įrangos plėtojimo fazes:

- ankstyvieji reikalavimai (angl. *early requirements*)
- vėlyvieji reikalavimai(angl. *late requirements*)
- architektūrinis projektavimas (angl. *architectural design*)
- detalus projektavimas(angl. *detailed design*)
- realizacija (angl. *implementation*)

Svarbiausias *Tropos* reikalavimų analizės tikslas yra tas, kad kuriama sistema būtų sukurta pagal gerai surinktus funkcinius ir nefunkcinius reikalavimus. Reikalavimų analizė *Tropos* metode dalinama į dvi fazes: ankstyvųjų ir vėlyvųjų reikalavimų analizę. Abi dalys apima tiek koncepcinius, tiek metodologinius požiūrius. Pagrindinė idėja yra ta, kad ankstyvųjų reikalavimų analizė yra naudojama vėlyvųjų reikalavimų analizėje. Ankstyvųjų reikalavimų analizė apima problemos supratimą, kuris pasiekiamas studijuojant egzistuojančią organizaciją. Tarpiniai elementai yra modeliuojami kaip aktorių tikslai ir tikslų priklausomybės. Šioje fazėje organizacijos modelis apima tiesiogiai susijusius aktorius ir jų tikslų bei potikslių priklausomybes. Modelis vaizduojamas aktorių diagramomis, kurios aprašo aktorių ir tikslų priklausomybes, analizuojami tikslų atitikimai. Vėlyvųjų reikalavimų analizėje koncepcinis modelis praplečiamas įtraukiant naujus aktorius, kurie vaizduoja sistemą, ir priklausomybes su kitais aktoriais iš tos aplinkos. Šios priklausomybės apibrėžia visus funkcinius ir nefunkcinius reikalavimus tam, kad būtų sukurta siekiama sistema (angl. *system-to-be*)[1]
.

Architektūrinio ir detalaus projektavimo fazės yra sukoncentruotos ties sistemos specifikacija, remiantis reikalavimų rezultatais iš prieš tai aprašytų dviejų fazių. Architektūrinis projektavimas aprašo sistemos visą architektūrą išreikštą per posistemius, sujungtus duomenų bei valdymo srautais. Posistemiai vaizduojami kaip aktoriai, o duomenų ir valdymo tarpusavio sąryšiai vaizduojami kaip priklausomybės. Detalaus projektavimo tikslas yra specifikuoti sąveikas. Realizacijos (angl. *Implementation*) veikla leidžia palaipsniui, natūraliu būdu, detalizuoti projekto specifikacijas vykdymo platformos konstravimo ir detalaus projektavimo notacija [6]. 1 paveiksle pavaizduota *Tropos* metodo aktorių diagrama.



1 pav. *Tropos* metodo aktorių diagrama

KAOS tikslų modeliavimo metodas

Tikslų aiškinimasis ir manipuliavimas jais yra natūrali reikalavimų inžinerijos dalis: reikalavimai pagal tipus pristato tikslus, kurie turi būti pasiekti. Ankstesnės reikalavimų inžinerijos technologijos sukoncentruotos ties esybėmis ir veiklomis. Į tikslus orientuota metodologija pirmenybę teikia iš anksto suplanuotiems tikslams – vystant programinės įrangos sistemas, esybės ir veiklos remiasi tikslais. Tikslų analizė priveda prie sistemos projektavimo alternatyvų analizės – „Kiekvienas gali išdėstyti tikslą, neturėdamas tikslių specifikacijų, kaip jį pasiekti" [5]. Aukštas abstrakcijos lygis padeda projektuotojui kurti geresnes sistemas.

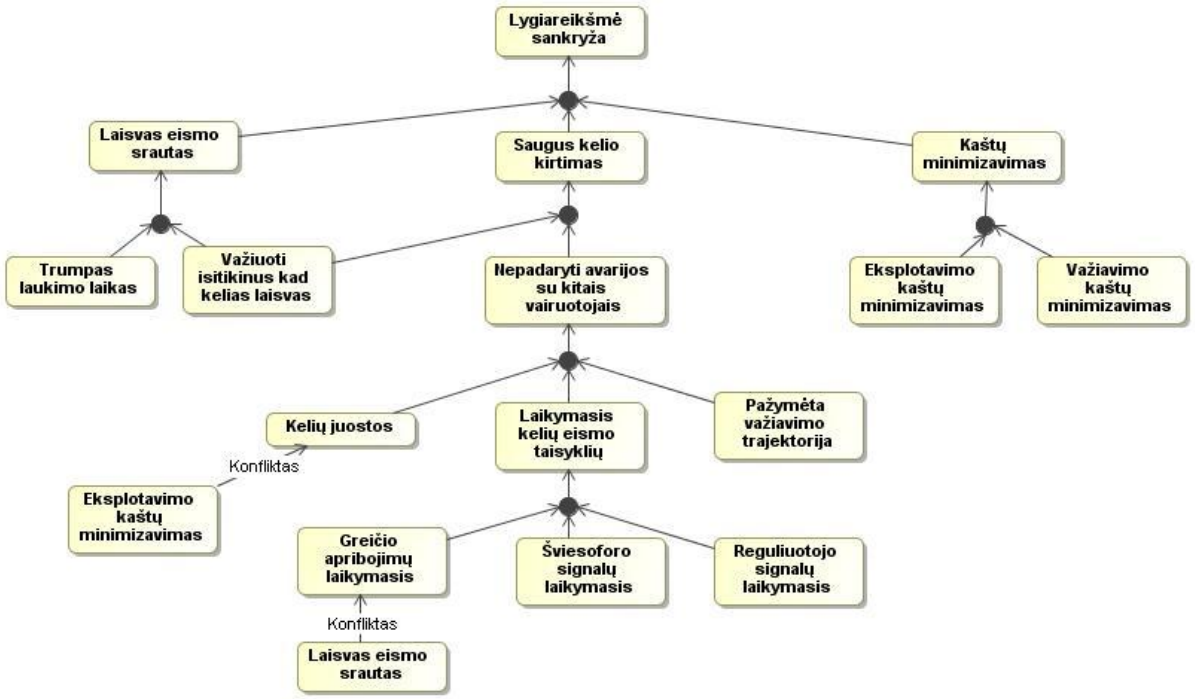
Vienas iš geriausių į tikslus orientuotų metodų yra KAOS. Tai tikslais grindžiama reikalavimų inžinerijos metodologija, kuri leidžia analitikams kurti reikalavimų modelius bei išgauti reikalavimų dokumentus iš KAOS modelių [5]. KAOS struktūra lengvina tolimesnių tikslų ir juos įgyvendinančių objektų, agentų bei sistemos veiksmų nustatymą. KAOS palengvina patį reikalavimų išgavimo ir modeliavimo procesą. Tikslai yra naudojami tikėtinų sistemos savybių aprašymui. Jie analizuojami ir geriau suprantami nustatant aukštesnio lygmens tikslus, kurie gali būti griežti ir negriežti. Nustatant tikslų tarpusavio sąryšius, sukuriamas tikslų grafas. Tikslų grafas yra semantinis tinklas su IR/ARBA/XARBA (AND/OR/XOR) sąryšiais. Naudojant konjunkcijas arba disjunkcijas tikslai išskaidomi iki negriežtų tikslų.

Reikalavimų analizė KAOS metode išsiskaido į tris fazes [3]:

- informacijos rinkimas, kuris naudojamas kaip vedlys tikslams pasiekti;
- modeliavimas;
- specifikacija, sintezė ir t.t.

Alternatyvos apibūdina skirtingus tikslų pasiekimo būdus. Jie gali būti motyvuoti skirtingos politikos ar projektavimo sprendimų, kurie lemia realizavimo technologiją ar sprendimo kokybę. Alternatyvos nebūtinai turi būti išskirtinės, tai reiškia, kad kiekvienas gali nuspręsti vystyti sistemą naudojant įvairias alternatyvas, pavyzdžiui, pagerinti sistemos patikimumą. Alternatyvos neapibrėžia padalijimo, kadangi skirtingos alternatyvos gali turėti bendrus negriežtus tikslus.

Konfliktai. Tikslai, išreikšti skirtingų tarpininkų, gali prieštarauti. KAOS konfliktas atsiranda tarp dviejų tikslų, jei gali būti sistemos būsenų, kuriose abu tikslai yra logiškai prieštaraujantys. Svarbu identifikuoti prieštaringus tikslus kaip galima anksčiau gyvavimo cikle, kad būtų sumažinti sistemos atmetimo pavojai. 2 paveiksle pateikiama KAOS metodo tikslų diagrama, kurioje yra pavaizduotos alternatyvos bei konfliktai. Stačiakampiai reiškia tikslus. Apskritimai reiškia pagrindinių tikslų papildymą žemesnių negriežtų tikslų komplektu.



2 pav. KAOS metodo aktorių diagrama, alternatyvos bei konfliktai

Metodų analizės apibendrinimas

Atlikus reikalavimų specifikavimo metodų analizę buvo atskleistos atskirų metodų savybės. *Volere* ir RUP reikalavimų specifikavimo metodai mažai dėmesio skiria tikslams. *Tropos* ir ypač KAOS metodas yra orientuoti į tikslus. Norint įtraukti tikslų analizę į praktinį projektavimo procesą, tikslinga panaudoti įvairių metodų elementus. RUP procesas taikomas labai plačiai, todėl tikslinga kaip pagrindinį procesą naudoti RUP. Siekiamas reikalavimų šablonas turėtų išskaidyti reikalavimus iki atominių, kaip *Volere*, tačiau įtraukti ir panaudojimo atvejų scenarijus kaip RUP. *Tropos* metodas nagrinėja tikslus, tačiau iš esmės jis labai panašus į RUP. Tikslų modeliavimui turėtų būti taikoma KAOS metodologija. Galiausiai, tikslai turėtų būti siejami su kiekvienu atominiu reikalavimu. Metodų palyginimas pateikiamas 1 lentelėje.

1 lentelė. Metodų palyginimų lentelė

Palyginimo kriterijai	VOLERE	RUP	TROPOS	KAOS
Ar nagrinėja tikslus?	–	–	+–	+
Ar nagrinėja tikslu priklausomybes ir konfliktus?	–	–	–	+
Ar išveda reikalavimus iš tikslu?	–	–	–	+–
Ar išskaido reikalavimus iki atominių vienetų?	+	–	–	–
Ar turi šablonus reikalavimams specifikuoti?	+	+	–	–

Tikslų ir reikalavimų modeliavimo galimybių UML CASE įrankiuose analizė

Dauguma UML CASE įrankių neturi specialių priemonių specifikuoti tikslus ir susieti juos su reikalavimais. Taip pat nėra galimybių specifikuoti reikalavimus – jie tik vaizduojami panaudojimo atvejais, kurių turinys be specifikacijų duoda mažai informacijos. Tam reikia naudoti atskirus reikalavimų specifikavimo įrankius, pavyzdžiui, DOORS ar *RequisitePro*. UML CASE įrankyje *Magic Draw* yra galimybė sugeneruoti RUP reikalavimų šabloną, tačiau jo pildymo būdas yra sudėtingas, specifikacija perteklinė, užima labai daug vietos ir neturi ryšio su tikslais.

Tačiau yra perspektyvių priemonių, kurias būtų galima panaudoti:

• *Magic Draw* priklausomybių matrica, kuri gali padėti trasuoti tikslus ir reikalavimus;

• UML profilių ir dalykinių sričių kalbų DSL (angl. *Domain Specific Language*) kūrimo mechanizmas, kuris gali padėti sukurti tikslų modeliavimo priemones ir reikalavimų specifikavimo šablonus;

• Sistemų modeliavimo kalbos *System Modeling Language* (*SysML*) [8], kuri yra UML profilis, galimybės reikalavimams specifikuoti. *SysML* profilį galima panaudoti kaip prototipą, įgyvendinant *Volere* šablono elementus siekiamoje reikalavimų specifikacijoje.

Išvados

Planuojant informacinių sistemų projektus, tikslinga analizuoti tikslus, nes taip galima padidinti jų tinkamumą veiklos poreikiams. Tačiau tikslų modeliavimas kol kas nėra plačiai naudojamas praktiniuose kūrimo procesuose. Viena to priežasčių – reikiamų modeliavimo priemonių nebuvimas CASE įrankiuose.

Atlikus žinomiausių tikslų modeliavimo ir reikalavimų specifikavimo metodų analizę, nustatyta, kad šiems tikslams galima sujungti standartinio proceso RUP, *Volere* reikalavimų specifikavimo šablono ir KAOS tikslų modeliavimo metodų elementus.

UML CASE įrankio *Magic Draw* analizė parodė, kad tyrimo tikslams pasiekti galima panaudoti šio įrankio įskiepį *SysML*, priklausomybių matricą, UML profilius bei DSL kūrimo galimybes. Naudojant šias technologijas, bus kuriamas reikalavimų specifikavimo priemonių prototipas, kuris leistų paprastai, nenaudojant papildomų paketų, sumodeliuoti tikslus, susieti su detaliai specifikuotais reikalavimais ir atsekti standartinio projektavimo proceso metu.

Literatūros sąrašas

[1] **Bresciani, P., Sannicolo, F.**. Applying Tropos Early Requirements Analysis for defning a Tropos tool. ITC-irst Via Sommarive, 18, I-38050 Trento-Povo, Italy.

[2] **R. Butleris, T. Danikauskas**. Reikalavimų specifikavimo ORACLE CASE terpėje plėtra. Prieiga per internetą: www.leidykla.vu.lt/inetleid/inf-mok/19/str6.html

[3] **Darimont, R., Delor, E., Rifaut, A.**. Quality Starts with the Definition of Goals. Prieiga per internetą: http://www.objectiver.com/download/documents/papers/archives/QWE2001.pdf

[4] **Giorgini, P., Giunchiglia, G., Gylopoulos, J.** Tropos: An Agent-Oriented Software Development Methodology. Kluwer Academic Publishers, 2004.

[5] **Heaven, W., Finkelstein, A.** A UML profile to support requirements engineering with KAOS. Prieiga per internetą: http://www.cs.ucl.ac.uk/staff/A.Finkelstein/papers/umlreprofile.pdf

[6] **Kruchten, P**. Rational Unified Process, The: An Introduction, Third Edition Addison Wesley, 2003.

[7] **Robertson, J., Robertson, S.**Volere Requirements Specification Template. Atlantic Systems Guild, 1995-2001.

[8] OMG Systems Modeling Language (OMG SysML), V1.0. OMG Available Specification, 2007.

Analysis of Methods for Goal Driven Requirement Specification

There are many requirement specification methods, but not much of them are related with goals. This article presents analysis of methods related with goal modelling and requirement specification. The main result of analysis is revealing of perspectives for extending requirement specification method with goal modelling and analysis, relating and tracing requirements to goals, and integration of these steps into standard unified development process.