KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Vida Motekaitytė

Grafinės vartotojo sąsajos generavimas iš dialogo specifikacijos

Magistro darbas

Darbo vadovas
prof. L. Nemuraitė

Kaunas, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Vida Motekaitytė

Grafinės vartotojo sąsajos generavimas iš dialogo specifikacijos

Magistro darbas

Recenzentas
doc. dr. J. Matickas
2007-05-25

Vadovas
prof. dr. L. Nemuraitė
2007-05-24

Atliko
IFM - 1/2 gr. stud.
Vida Motekaitytė
2007-05-15

Kaunas, 2007

Turinys

Graphical user interface generation using dialog specification

SUMMARY

User interface code generation is a complex task. Managing interacting objects, handling their performing functions is a real challenge for system developer. Various automated code generation tools offer creation just of particular pages, but not full scenario. Programmers have to take care of all system functionality. This paper deals with analysis and implementation such tools that enables scenario generation that allows creation not only static web pages but also transitions between them. This project's aim is to investigate data storage, transfer and transformation technologies, which are fitted to store and transmit parameters and attributes, also transform them to programme's code. Furthermore, to create graphical user interface software design method, which allows to generate graphical user interface using abstract model - dialog specification. In general the proposal is to make scenario's specifications from state transition model, save them in XML documents and transform to program code using XSL transformations.

To obtain the purpose in this work in the first part are analyzed: similar systems which can design graphical user interface or generate code fragments, formal and non formal system's requirements, architecture and possible solutions for the system developments, defined the quality criterions. In the second part of this work is proposed analysis of the system which was developed during the master work, analyzis of risk factors, and requirements for data. In the third part is given detailed view to the system's architecture, realization model, reasoning of suggested method also there is testing model and data, control example, generalization of created model and recommendations. In the fourth part is performed experimental research of the system and conclusions are in the final part.

ĮVADAS

Šiuo metu programinė įranga jau yra neatsiejama daugelio organizacijų infrastruktūros dalis, tačiau jos poreikis ir toliau auga. Ypač daug dėmesio yra skiriama informacinių sistemų modeliavimui ir kūrimui. Augant kuriamos programinės įrangos (PĮ) apimčiai ir sudėtingumui, projektavimas tampa vis svarbesne IT sistemų inžinerijos dalimi. Visame pasaulyje siūloma nemažai PĮ kūrimo metodų ir programų, kad būtų galima užtikrinti šio sudėtingo proceso efektyvumą ir kokybišką programinio kodo generavimą.

Vartotojo sąsajų, skirtų taikomosioms programoms ir informacinėms sistemoms, projektavimas ir kūrimas turi labai didelę įtaką visam programų kūrimo produktyvumui, reikalauja daug intelektualinių, finansinių, techninių ir ypač laiko išteklių [9]. Dėl šių priežasčių reikalingas sprendimas, leidžiantis sumažinti laiko išteklius ir padidinti tokių programų kūrimo proceso efektyvumą.

Programų sistemos modelį galima aprašyti įvairiais būdais, tačiau šiame darbe akcentuojamas programos kodo generavimas iš UML modelio. Iš modelio sugeneruojamo programinio kodo kiekis skiriasi priklausomai nuo modeliavimo kalbos bei kodo generatoriaus galimybių. Didinant iš modelio sugeneruojamo ir ranka parašomo programinės įrangos kodo santykį, sutaupoma laiko, išvengiama dalies klaidų, didinamas kodo pakartotinis panaudojimas.

Darbo tikslas – palengvinti projektuotojų ir programuotojų darbą programinės įrangos, skirtos vartotojų sąsajoms kurti, kūrimo procese, pagerinti kuriamų produktų kokybę, padidinti kūrimo našumą, sutaupyti žmogiškuosius ir laiko išteklius. Tam reikia:

- sukurti metodą automatizuotam vartotojo sąsajos gavimui iš abstraktaus modelio – dialogo specifikacijos (mokslinis uždavinys),
- sukurti įrankio prototipą, kuris įgyvendintų vartotojo sąsajos generavimą pagal sukurtą metodiką (inžinerinis uždavinys).

Norint sukurti metodą, kuris leistų generuoti bet kokios programavimo kalbos kodą iš UML diagramų specifikacijų, buvo suformuluoti **darbo uždaviniai**:

- ištirti duomenų modeliavimo, saugojimo ir transformavimo technologijas,
- atlikti galimų sprendimų analizę, palyginti egzistuojančius sprendimus ,
- išsiaiškinti vartotojų poreikius,
- pasiūlyti metodą ir suprojektuoti programos, leidžiančios gauti kodą ir grafinį vartotojo sąsajos vaizdą, prototipą,
- realizuoti programos prototipą, pateikiant pavyzdžius,

- įvertinti metodo rezultatus gaunamus su sukurta programa ir esamomis priemonėmis,
- padaryti išvadas.

Norint generuoti vartotojo sąsają iš UML diagramų, jas teko praplėsti specifiniais atributais, žyminčiais vartotojo sąsajos objektus (mygtukus, įvedimo laukus), nurodyti perėjimų eiliškumą ir jų vykdymo sąlygas.

Visas kodo generavimo metodas apima UML modelio objektų specifikavimą, XSLT transformacijų sukūrimą bei teisingą jų parinkimą, programos kodo fragmentų generavimą, interneto sąsajos generavimą. Metodą realizuojantis įrankis buvo sukurtas Visual Studio aplinkoje.

Šio darbo **tyrimo sritis** – programinės įrangos kūrimo automatizavimas, modeliais grindžiamas kūrimas. **Tyrimo objektas** – vartotojo sąsajos projektavimo ir kodo generavimo metodas ir įrankis.

Šiame darbe pristatoma vartotojo sąsajos modeliavimo koncepcija, nagrinėjamos sistemų savybės, kurios svarbios projektuojant sistemos sąveikos algoritmus, aprašoma automatizuoto vartotojo sąsajos kodo generavimo galimybė, panaudojant modeliavimo kalbą UML bei duomenų perdavimo ir transformavimo technologijas: XML ir XSLT. Analizuojamą automatizuotą vartotojo sąsajos kūrimo procesą sudaro 3 pagrindiniai etapai: 1) vartotojo sąsajos modelio sudarymas objektų diagramų pagalba; 2) transformacijų sukūrimas 3) kodo ir vartotojo sąsajos generavimas transformacijų pagalba.

Darbo struktūra.

Pirmame darbo skyriuje aprašytas analizės tikslas, tyrimo sritis, objektas, problema, atlikta galimų sukurto produkto vartotojų analizė, literatūros šaltinių analizė, apžvelgtos sistemos panašios į kuriamą sistemą Lietuvos ir tarptautiniu mastu, išanalizuota architektūra ir galimi programos įgyvendinimo priemonių variantai, apibrėžtos sistemos funkcijos, reikalavimai duomenims, nefunkciniai reikalavimai ir apribojimai, aptarti rezultato kokybės kriterijai.

Antrame darbo skyriuje pateikta sistemos, skirtos interneto sąsajos gavimui iš dialogo specifikacijos, reikalavimų specifikacija ir analizė, aprašytas ir pateiktas dalykinės srities modelis.

Trečiame skyriuje pristatomas projektas, sistemos architektūra, sistemos elgsenos modelis.

Ketvirtame skyriuje pateikiama sistemos realizacija, sistemos veikimo aprašymas, testavimo modelis bei duomenys.

Penktame skyriuje atliktas eksperimentinis sistemos tyrimas, pateikiami kontroliniai pavyzdžiai, aprašomos sistemos savybės, pateikiamas kokybės kriterijų įvertinimas, taikymo rekomendacijos.

Septintame skyriuje pateiktos darbo išvados.

Atliekant analizę ir pristatant sukurtą metodą buvo parašyti du straipsniai, tiesiogiai susiję su šia tema ir pristatyti tarpuniversitetinėse magistrantų, doktorantų konferencijose ir du straipsniai, kurie taip pat pristatyti konferencijose, susiję su informacinėmis technologijomis, tačiau nesiejami su magistro darbo tema. Straipsnių kopijos pateikiamos prieduose.

1. AUTOMATIZUOTO KODO GENERAVIMO SISTEMOS ANALIZĖ

1.1. Analizės tikslas

Šios analizės tikslas – atliekant sistemų, leidžiančių automatizuotą programų kūrimą, analizę įvertinti tokių sistemų poreikį, reikalingas funkcijas, ištirti egzistuojančių sistemų (Lietuvoje ir tarptautiniu mastu) galimybes išryškinant trūkumus ir aptariant privalumus, taip pat ištirti rinką, t.y. vartotojų aibę, jų tikslus, esančius sunkumus, žinant kokią sistemą norima kurti, pagal tai pasiūlyti metodą ir pagal jį sukurti modelį, leidžiantį gauti grafinės vartotojo sąsajos vaizdą iš abstraktaus modelio - dialogo specifikacijos, atlikti architektūros ir galimų įgyvendinimo priemonių variantų analizę, apibrėžti rezultato kokybės kriterijus, kad būtų galima įvertinti sukurtą produktą.

1.2. Tyrimo sritis, objektas ir problema

Visame pasaulyje gausu kompanijų ir pavienių asmenų, kuriančių taikomąsias programas. Šiame technologijų amžiuje siūloma nemažai PĮ kūrimo metodų ir programų, tam, kad būtų galima užtikrinti šio sudėtingo proceso efektyvumą ir kokybišką programinio kodo generavimą. Vartotojo sąsajų, skirtų taikomosioms programoms ir informacinėms sistemoms, projektavimas ir kūrimas turi labai didelę įtaką visam programų kūrimo produktyvumui, reikalauja daug intelektualinių, finansinių, techninių, o ypač laiko resursų, todėl turime rasti sprendimą kaip sumažinti laiko išteklius ir padidinti tokių programų kūrimo proceso efektyvumą.

Vartotojo sąsajos programinę įrangą sukurti sudėtinga, nes ji turi valdyti daug objektų, kurie būna susiję tarpusavio ryšiais ir atlieka tam tikras funkcijas [9]. Daugelis kodo generavimo įrankių leidžia sukurti tik tam tikros programavimo kalbos kodą arba tik tam tikrus puslapius, o scenarijų turi realizuoti pats programuotojas. Darbe nagrinėjamas sistemos modelis grindžiamas automatizuoto grafinės vartotojo sąsajos generavimo metodu, kurį realizavus galima generuoti scenarijų kodą, t.y. gauti vartotojo sąsajos vaizdą, perėjimus iš vieno puslapio į kitą arba gauti bet kurios programavimo kalbos programos kodo fragmentus. Scenarijų specifikacijas siūloma sudaryti iš objektų diagramos, saugoti jas XML dokumentų pavidale ir transformuoti į programos kodą naudojant XSLT.

Sukurtas įrankis bus skirtas PĮ kūrėjams, t.y. pavieniams asmenims tokiems kaip, programuotojai/projektuotojai/analitikai ar įmonėms, kuriančioms internetines sistemas ar programinę įrangą.

Šio darbo tyrimo sritis – programinės įrangos kūrimo automatizavimas, modeliais grindžiamas kūrimas. Tyrimo objektas – vartotojo sąsajos projektavimo ir kodo generavimo metodas ir įrankis.

Norint sukurti metodą, kuris leistų generuoti bet kokios programavimo kalbos kodą iš UML specifikacijų, teko išnagrinėti tiek programavimo kalbų, tiek UML veiksmų semantikos ypatybes. Norint generuoti vartotojo sąsają iš UML diagramų, jas teko praplėsti specifiniais atributais, žyminčiais vartotojo sąsajos objektus (mygtukus, įvedimo laukus), nurodyti perėjimų eiliškumą ir jų vykdymo sąlygas.

Kuriamo produkto vartotojams šis įrankis padės išspręsti dalį problemų: leis padidinti darbo našumą, taupyti laiką, pakartotinai panaudoti anksčiau sukurtas specifikacijas ir transformacijas, įtraukti naujas XSLT bibliotekas, realizuoti sukurtus modelius daugelyje programavimo kalbų.

1.3. Vartotojų analizė

Norint, kad sukurtas produktas būtų naudojamas ir naudingas, reikia apsibrėžti vartotojų aibę, išnagrinėti sunkumus, su kuriais jie susiduria savo darbe ir mažinant problemas sukurti efektyvų sprendimą.

1.3.1 Vartotojų aibė, tipai, charakteristikos

Kuriamos programinės įrangos naudotojui atstovauja analitikai, projektuotojai, programuotojai, o galutiniam sukurto produkto naudotojui (t.y. produkto, kuris bus kuriamas naudojant automatizuoto kodo/grafinės vartotojo sąsajos generavimo įrankį) – dalykinės srities specialistai, pvz., odontologai, kuriems bus sukurta informacinė sistema, kūrimui panaudojant siūlomą įrankį. Analitikai, projektuotojai – tai specialistai, susipažinę su sisteminės analizės metodais ir turintys praktinius tų metodų taikymo įgūdžius. Jie yra tarpininkai tarp būsimų sistemos naudotojų ir jos projektuotojų.

Pagrindinė informacijos rinkimo problema – tarpusavio bendravimas. Analitikams, būsimiems kuriamos programos naudotojams ir dalykines srities ekspertams vieniems kitus suprasti yra gana sudėtinga. Jie kalba skirtingų profesijų žargonais, naudoja skirtingas mąstymo schemas. Labai svarbi terminų problema ir kuo daugiau programos galimybių išreikštų vaizdžiai, kad būtų galima suprasti tiek sistemos naudotojui, tiek galutinio produkto, sukurto naudojant tą sistemą, užsakovui ir vartotojui.

Asmenų, kurie betarpiškai naudosis kuriama sistema, sąrašas bei informacija apie juos, pateikta 1 ir 2 lentelėse.

1 lentelė. Vartotojas – projektuotojas/programuotojas

Vartotojo kategorija	Projektuotojas/programuotojas
Vartotojo sprendžiami uždaviniai	Įrankis leidžia pakartotinai panaudoti anksčiau sukurtas specifikacijas ir transformacijas, įtraukti naujas XSLT bibliotekas, realizuoti sukurtus modelius daugelyje programavimo kalbų.
Patirtis dalykinėje srityje	Patyręs.
Patirtis informacinėse technologijose:	Patyręs.
Papildomos vartotojo charakteristikos:	Turi mokėti braižyti UML diagramas, o programuotojas rašyti XSL transformacijas.
Apmokymo poreikis:	Minimalus
Amžiaus grupė:	20-70

2 lentelė. Vartotojas – pavieniai asmenys, kuriantys internetinius puslapius

Vartotojo kategorija:	Internetinių puslapių kūrėjas
Vartotojo sprendžiami uždaviniai:	Vartotojas kuria grafines vartotojo sąsajas, programinę įrangą ir domisi automatiniu programos kodo generavimu. Kuriamas įrankis būtų naudojamas kaip pagalbinis įrankis automatizuoti PĮ scenarijaus sudarymo procesą. Diagramų pagalba galima projektuoti, t.y. sudaryti objektų diagramas, apimančias scenarijų aibę kiekvienam vartotojo tipui, po to sudaryti specifikacijas, kurias būtų galima dinamiškai transformuoti į puslapius.
Patirtis dalykinėje srityje	Patyręs.
Patirtis informacinėse technologijose:	Patyręs.
Papildomos vartotojo charakteristikos:	Turi mokėti braižyti UML diagramas, rašyti XSL transformacijas.
Apmokymo poreikis:	Minimalus
Amžiaus grupė:	20-70

Kuriama sistema skirta įmonėms ir pavieniams asmenims kuriantiems internetines sistemas. Sistema, neįgijus naudojimo teisių yra neprieinama.

Numatomas kuriamos sistemos vartotojas yra analitikas (projektuotojas) ir/arba programuotojas. Analitikas turi mokėti braižyti UML diagramas, o programuotojas rašyti/keisti XSL transformacijas. Galutiniais sistemos, sukurtos naudojantis siūloma PĮ, vartotojais gali būti bet kokios kvalifikacijos ar profesinio išsilavinimo privatūs asmenys, mokantys naršyti po internetines svetaines.

Taigi galima teigti, kad kuriama informacinė sistema iš jos vartotojo reikalauja pakankamų kompiuterinio raštingumo bei programavimo žinių ir supratimo apie sistemos projektavimą.

1.3.2 Vartotojų tikslai ir problemos

Kuriamas metodas sutrumpins projektavimo ir programavimo laiką, nes automatizuos kelis darbo etapus, leis sugeneruoti daugiau programos kodo nei naudojant kitas automatizuoto kodo generavimo priemones. Taip pat galima pastebėti, kad daugelis egzistuojančių programų leidžia sukurti tik tam tikrus puslapius, o scenarijų turi realizuoti pats projektuotojas. Todėl reiktų tokios priemonės, kuri leistų kurti scenarijų, t.y. aprašyti ar grafiškai atvaizduoti kas su kuo jungiasi, kokie perėjimai tarp puslapių.

1.3.3 Informacija apie užsakovo organizaciją

Projekto užsakovas yra Kauno Technologijos Universiteto (KTU) Informatikos fakulteto Informacijos sistemų katedra [20]. KTU - tai mokslo įstaiga, kuri yra suinteresuota, kad besimokantys ar pabaigę studijas joje studentai, gautų įgytą kvalifikaciją atitinkančius darbus. Siekiant tai užtikrinti, užsakovas pateikė pasiūlymą įgyvendinti tokį projektą, kurio realizavimo metu būtų sukurtas sistemos kūrimo metodas, leidžiantis generuoti grafinę vartotojo sąsają iš vartotojo dialogo specifikacijos.

Potencialiais kuriamos sistemos pirkėjais galėtų būti įmonės, kuriančios elektroninio verslo sistemas, programinę įrangą, siekiančios pagreitinti kūrimo laiką ir pagerinti savo veiklos kokybę.

Taip pat šiuo produktu galėtų susidomėti pavieniai asmenys, kurie kuria grafinbes vartotojo sąsajas, programinę įrangą ir domisi automatiniu programos kodo generavimu. Kuriamas įrankis būtų naudojamas kaip pagalbinis įrankis, skirtas PĮ scenarijaus sudarymo proceso automatizavimui. Diagramų pagalba bus galima projektuoti, t.y. sudaryti objektų diagramas, apimančias scenarijų aibę kiekvienam langui, po to sudaryti specifikacijas ir visa tai, būtų galima dinamiškai transformuoti į puslapius.

1.4. Problemos sprendimo metodų literatūros šaltiniuose analizė

Analizuojant medžiagą įvairiuose literatūros šaltiniuose, tampa aišku, kad bendru atveju kiekvienas projektas (arba programinių produktų linija) yra unikalus, naudoja unikalią realizacijos technologijų kombinaciją ir turi unikalius poreikius. Tai reiškia, kad praktiškai neįmanoma sukurti statiško programos kodo generavimo, kuris tenkintų visų projektų poreikius, todėl yra sukurta daug universalių įrankių, kurie savo viduje turi ir generatoriaus funkcijas. Kai

kurie projektai naudoja tik struktūrinio programos kodo bei pagalbinių priemonių generavimą, kiti generuoja dalį elgsenų nusakančio kodo [22]. Vieniems projektams visapusiškos UML specifikacijos naudojimas modeliavime būtų bereikalingas sudėtingumo padidinimas, kiti projektai tiesiog negali išsiversti be tam tikros UML dalies naudojimo. Dažniausiai dalis programos kodo yra sugeneruojama, o dalis parašoma ranka [22].

Daugelis egzistuojančių įrankių neturi galimybės sukurti būsimos sistemos prototipo arba reikia įdėti daug darbo rašant programini kodą. Pavyzdžiui, Rational Rose paketas sugeneruoja klasių aprašus, bet metodus reikia rašyti pačiam. Oracle Designer paketo trūkumas yra tas, kad jis daugiau yra skirtas sistemos projektuotojams. Naudojantis juo sistemos prototipą galima sukurti tik sudarius jos projektą. Be to, kiekvienas CASE įrankis automatizuoja tam tikrą informacinių sistemų kūrimo metodą. Pavyzdžiui, Rational Rose paketas automatizuoja Rational Unified Process (RUP) metodą, Oracle Designer – Oracle CASE metodą [9]. Kalbant apie interneto sąsajos kūrimą galima paminėti ir tokį įrankį kaip FrontPage, kuris leidžia matyti iš karto interneto sąsajos puslapius. Projektuoti sistemas ir gauti programinio kodo fragmentus leidžia MagicDraw UML įrankis, pastaruoju metu plačiai paplitęs ne tik Lietuvoje bet ir visame pasaulyje, jį naudoja didelės įmonės, kurios projektuoja, specifikuoja, programuoja dideles informacines sistemas.

Analizuojant straipsniuose, interneto šaltiniuose, kitoje literatūroje pateiktą medžiagą, galima vienareikšmiškai teigti, kad automatizuoto kodo kūrimo įrankių problema yra aktuali ne tik Lietuvos, bet tarptautiniu mastu. Daugelis mokslininkų siūlo įvairius metodus ir modelius, tačiau dažniausiai sistemų projektavimas yra paremtas UML diagramomis. Apibendrinant skaitytą medžiagą ir atliktus tyrimus, bei analizuojant, kurį metodą pasirinkti, būtų geriausia kuriamai sistemai projektuoti, toliau aprašomos pagrindinės ir dažniausiai projektavimo procese naudojamos UML diagramos.

Sekų diagrama (angl. *Sequence*). Ji detalizuoja panaudojimo atvejus, todėl šis modelis aprašo sistemos atliekamų darbų eigą, sąveiką tarp objektų [1]. Šios diagramos parodo darbų eigos, sąveikos tarp objektų tęstinumą laiko atžvilgiu, nusako, kada ir kokie veiksmai turi būti atliekami. Panaudojant pseudo kodą, galima įvesti sąlyginę darbų atlikimo seką, tai leidžia specifikuoti tokius veiksmus: sistemos elgseną, reakciją į įėjimus, ko sistema neturėtų daryti.

Čia perkeliama dalis nefunkcinių saugumo reikalavimų, t. y., griežtai apibrėžiama, kokie veiksmai galimi su duomenimis, taip pat specifikuojamos vartotojų teisės.

Klasių diagrama (angl. *Class*). Joje specifikuojamas sistemos duomenų modelis. Pagrindiniai reikalavimai šiam modeliui gaunami su įėjimo ir išėjimo srautais, kuriuos prieš tai turi apdoroti sistemos analitikas. Nedidelė dalis reakcijos į įėjimus taip pat persikelia į duomenų

modelį. Diagramoje modeliuojamos objektų klasės, jų atributai, su jomis atliekamos operacijos ir ryšiai tarp klasių [1].

Objektų diagrama (angl. *Object*) – labai panaši į klasių diagramą ir naudoja panašias notacijas ryšiams žymėti. Žiūrint į klasių diagramą nematome kaip klasės sąveikauja tarpusavyje programos veikimo metu ir kaip sukurti objektai vykdymo metu yra susiję su klasėmis tikroje sistemoje. Būtent objektų diagrama parodo ryšius tarp apibrėžtų ir vykdomų klasių ir ryšius tarp objektų. Objektų diagrama padeda atvaizduoti nedideles sistemos dalis, kai tuo tarpu klasių diagramos dažniausiai būna labai sudėtingos [7].

Bendradarbiavimo diagrama (angl. *Collaboration*). Ji artima sekų diagramai, nes taip pat parodo sistemos objektų sąveiką (bendradarbiavimą), tik joje neatsispindi sąveikos ar darbų eigos tęstinumas laike. Diagramose taip pat galima panaudoti pseudo kodą. Kaip ir sekų diagramoje, čia specifikuojami reikalavimai: veiksmai, ko sistema neturėtų daryti; sistemos elgsena; reakcija į įėjimus [7]. Kuriant sistemą, bendradarbiavimo ir sekų diagramas galima laikyti viena kitai alternatyviomis. Todėl poreikius galima atvaizduoti viena iš šių diagramų, kuri reikiamu atveju labiau tinka.

Būsenų diagrama (angl. *Statechart*). Joje specifikuojami reikalavimai sistemos elgsenai. Kadangi sistemos elgsena ir būsenos, į kurias pereis sistema, priklauso nuo įeinančių duomenų, tai reakcijos į įėjimus reikalavimai taip pat specifikuojami šioje diagramoje [1]. Išskiriamos objektų (taip pat sistemų ir posistemių) būsenos ir perėjimai tarp jų.

Veiklos diagrama (angl. *Activity*). Ji skirta elgsenos atvaizdavimui sistemos viduje. Nors ši diagrama gali būti panaudojama modeliuojant visos organizacijos veiklą. Esant didelei sistemai, jos veiklą patogu nagrinėti skaidant ją pagal panaudojimo atvejų diagramas. Įėjimo ir išėjimo srautai šioje diagramoje specifikuojami siekiant parodyti juos veiklos kontekste [7].

Darbe naudojamoms UML diagramoms projektuoti, kurti ir analizuoti bus pasitelktos šiuolaikines CASE priemonės – paketas „Microsoft Visio".

Automatizuotas programinės įrangos projektavimas yra vienas iš metodų, užtikrinančių vartotojų poreikių analizės ir specifikavimo etapo kokybiškumą. CASE priemonėmis iš painaus programinio kodo pereiti į lygmenį, kur architektūra ir projektavimas tampa vaizdingesni. Toks programų sistemų kūrimo procesas artimesnis vartotojui.

CASE priemonės naudojamos visame sistemos kūrimo cikle. Jomis organizuojamas ir valdomas programinės įrangos kūrimas ir ypač jos svarbios kuriant didelius ir sudėtingus projektus. CASE priemonės padeda susisteminti sistemos kūrimo procesą, griežčiau jį kontroliuoti.

CASE įrankiai siūlo įvairius programinės įrangos kūrimo metodus. Dalis priemonių pagrįstos struktūriniu metodu, bet pastaruoju metu daugiau orientuojamasi į objektiškai orientuotą programines įrangos kūrimą. Projektuojant sistemą bus naudojami UML modeliai: veiklos konteksto diagrama, panaudojimo atvejų diagrama, duomenų modelis, klasių diagrama, sekų diagramos.

Išanalizavus literatūroje pateiktus sprendimus bei įrankius (kurių analizė pateikiama 1.7 skyriuje), nuspręsta sistemoje vykdomam projektavimui naudoti objektų diagramas, kurios aiškiai leidžia atvaizduoti interneto sąsajos puslapius ir sąryšius/perėjimus tarp jų.

Atlikus literatūros analizę, dar prieš kuriant sistemą ir pagrindžiant jos poreikį, būtina atlikti situacijos Lietuvoje ir tarptautiniu mastu įvertinimą, kad būtų galima pagrįsti arba atmesti kuriamos sistemos naudingumą.

1.5. Situacijos Lietuvoje ir tarptautiniu mastu įvertinimas

Šiuo metu pasaulyje yra daug įmonių kuriančių informacines sistemas (IS) ir programinę įrangą. Nors tokio pobūdžio įmonės per pakankamai optimalų laiką sukuria PĮ ar IS, tačiau jos neišnaudoja visų technologinių galimybių, t.y. programų, kurios yra jau sukurtos ir skirtos greitesniam projektavimui ar kūrimui, siekiant supaprastinti ir patobulinti patį kūrimo procesą. Analizuojant probleminę sritį buvo peržvelgtos kai kurios panašios sistemos [23,25,27,31,33], kurios vienu ar kitu atveju padeda projektuotojui ir/arba programuotojui optimizuoti darbo procesą. Kuriamo metodo projektavimo metu nepavyko rasti įmonių, naudojančių grafinės vartotojo sąsajos kūrimui dialogo specifikaciją ar publikuojamų programų, kurios tai atliktų, tačiau yra sukurtų projektų, kurie leidžia internetinės sąsajos vaizdavimą būsenų diagramomis. Analizės metu buvo aptartos kai kurios panašios savo pobūdžiu sistemos, siekiant surasti silpnąsias egzistuojančių sprendimų vietas, bei atrinkti geriausias tokio pobūdžio sistemų savybes.

Lietuvos statistikos departamento duomenimis [21] pastaruoju metu interneto paslaugomis naudojasi 87,5% įmonių, o internetines svetaines turi 64,2% įmonių. Taigi Lietuvoje sparčiai didėjant įmonių, besinaudojančių interneto paslaugomis bei turinčių internetinius puslapius skaičiui, gana rimtas perspektyvas turi įmonė, kurianti programinę įrangą ar internetines svetaines. Žemiau pateiktose lentelėse (3, 4, 5 lentelės) esanti statistinė informacija yra paimta iš Lietuvos statistikos departamento interneto svetainės [21].

3 lentelė. Įmonių dalis, kurios naudojasi internetu (procentais)

	2001	2002	2003	2004	2005	2006
Iš viso	**58,6**	**65,5**	**68,5**	**79,8**	**85,2**	**87,5**

4 lentelė. Institucijų, turinčių interneto puslapius dalis (procentais)

	2003	2004	2005	2006
Iš viso	**53,7**	**56,9**	**59,3**	**64,2**

5 lentelė. Įmonių, kuriančių interneto puslapius/ PĮ dalis (procentais)

	2003	2004	2005	2006
Iš viso	**23,8**	**28,5**	**31,4**	**33,1**

Kasmet vis daugiau įmonių kuriančių internetines svetaines ar programinę įrangą, todėl galima teigti, kad kuriamas produktas turės vis didesnę paklausą.

1.6. Panašių sistemų (Lietuvos ir tarptautiniu mastu) analizė

Programinės įrangos rinkoje siūloma daug brandžių UML įrankių, leidžiančių patogiai ir efektyviai modeliuoti projektavimo sprendimus bei kurti taikomąsias programas. Toliau aptariami keli iš jų.

1.6.1 Rational Application Developer for WebSphere Software

IBM Rational Application Developer for WebSphere Software [31] yra projektavimo ir kūrimo įrankis, kuris leidžia UML pagalba kurti taikomąsias programas.

Šio įrankio pagalba galima: apjungti visus programinės įrangos projektavimo ir vystymo etapus; efektyviai kurti taikomąsias programas ir Web paslaugas; naudoti naujausias modeliavimo kalbų technologijas; peržiūrėti ir valdyti JAVA aplikacijas; supaprastinti projektavimo ir vystymo įrankių naudojimą; projektuoti UML diagramų pagalba; generuoti JAVA, C++ kodą.

Visapusė integruota kūrimo aplinka (IDE), leidžia vartotojams projektuoti, kurti, analizuoti, testuoti, aprašyti, išdėstyti Web, Web paslaugas, JAVA, J2EE bei taikomąsias programas. Kūrėjai gali prieiti prie plataus reikalavimų diapazono ir keisti valdymo funkcijas tiesiogiai iš Rational Application Developer for WebSphere Software.

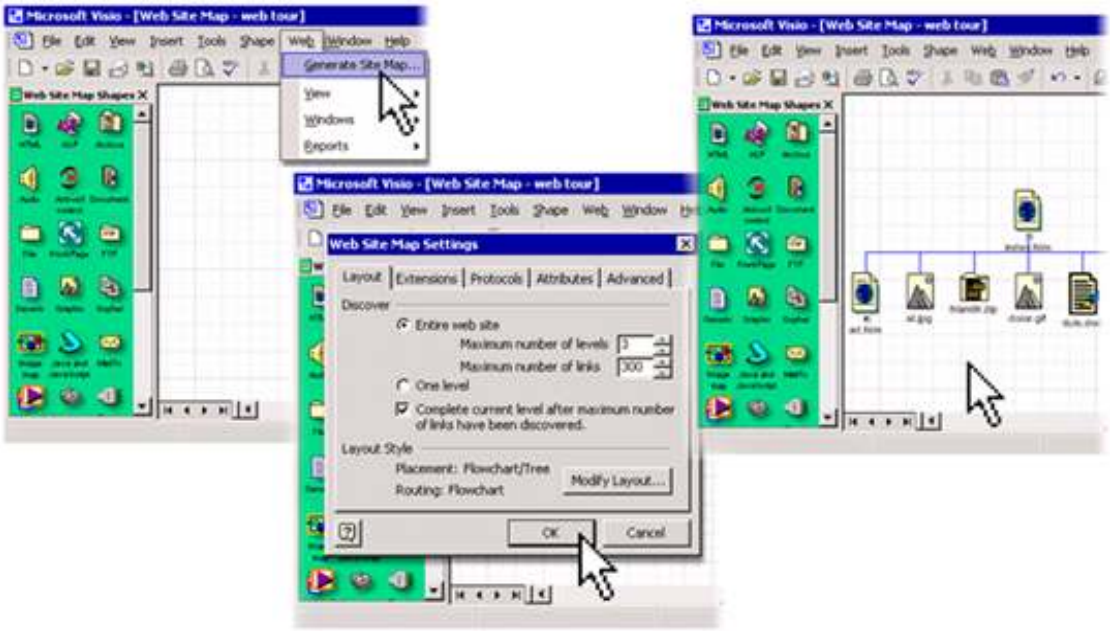
Ši programinė įranga suteikia galimybę paspartinti puslapio kūrimą naudojant RAD įrankius ir vedlius [31]. Kuriant taikomąsias programas yra naudojamos įvairios klasės, panaudojimo atvejų elementai, esybės, ryšiai ir diagramos naudojant „paimk ir nunešk“ (angl. drag-and-drop) simbolius, tai yra labai patogu vartotojui. Automatizuotomis priemonėmis paremtomis naudotinais kodavimo standartais galima pagerinti kodo kokybę. Galima vizualiai ir grafiškai redaguoti kodą pasinaudojant UML Visual Redaktoriumi skirtu JAVA ir EJB.

IBM Rational Application Developer for WebSphere Software yra objektiškai orientuotos UML programinės įrangos kūrimo įrankis skirtas vaizdiniam projektavimui ir komponentų kūrimui. Sukūrus diagramą galimas kodo generavimas įvairiomis programavimo kalbomis, kurias vartotojas gali pasirinkti, tai C++, Visual Basic, Java, Oracle8, CORBA arba Data Definition Language (Duomenų apibrėžimo kalba). Taip pat ši PĮ palaiko COM/DCOM (ActiveX), JavaBeans, ir Corba komponentų standartus. Šis įrankis nepalaiko XSLTir taip pat neleidžia naujų bibliotekų įtraukimo.

1.6.2 Microsoft Visio Professional 2002

Microsoft Visio Professional 2002 - tai diagramų kūrimo programa, leidžianti kurti įvairias UML diagramas. Visio pagalba sukurtos diagramos leidžia vizualiai aiškiai, glaustai ir efektyviai pateikti tekstą ir skaičius, kurie atskirai nebūtų tokie informatyvūs [25].

Su Visio galima kurti ir rekonstruoti esamas interneto ir intraneto svetaines, naudojant svetainės išdėstymo vedlį. Galima dirbti su figūromis, vaizduojančiomis HTML puslapius, Java programėles, daugialypę terpę, vaizdus ir kitokį svetainės turinį (1 pav.)



1 paveikslas. Projektavimas su MS Visio

Visio 2002 atspindi tris pagrindines patobulintas sritis:

- Visio diagramos informaciją vartotojams perteikia vaizdais
- Visio geriau integruota su kitomis taikomosiomis programomis.
- Visio pagalba galima vizualiai pateikti idėjas, informaciją ir kurti sistemas.

Visio pagalba galima sukurti vaizdinę diagramą, publikuoti ir bendrai naudotis savo diagramomis tinklapyje su pagerinta tinklapių įrašymo funkcija, importuoti ir eksportuoti diagramas vektorinės grafikos (SVG) formatu kaip naujo XML standarto tinklapio grafikos

objektus, integruoti verslo procesus ir sistemas pateikdami Visio sukurtas diagramas bei importuoti jas į Microsoft Excel, Microsoft Word, Microsoft Access, Microsoft SQL Server XML.

Grafinės vartotojo sąsajos projektavimas Visio Professional 2002 pagalba yra pakankamai paprastas. Jei naudotojas nusprendė sukurti IS, jam yra svarbu turėti aiškų vaizdą kaip atrodys kuriama sąsaja, t.y. įsivaizduoti ir matyti jos struktūrą, srautus, puslapių sujungimus ir panašiai. Visa tai galima atlikti naudojantis Visio Professional 2002 įrankiais [25].

Taigi iš pat pradžių reikia susikurti vaizdą, išanalizuoti objektus, jų tarpusavio sąveiką. Apmąstyti kaip puslapiai bus susiję tarpusavyje. Tam tikslui yra sukuriamas svetainės ir jos turinio grafinis vaizdas.

1.6.3 MagicDraw UML

„MagicDraw UML“ programinė įranga - tai UML ir programinės įrangos modeliavimo priemonė pritaikyta komandiniam darbui, nes leidžia daugeliui programinės įrangos kūrimo inžinierių dirbti su tuo pačiu OO modeliu vienu metu [27].

Naudodamiesi šia programa, verslo ir kompiuterinių sistemų analitikai, programuotojai, kokybės užtikrinimo specialistai ir dokumentacijos rengėjai gali kurti ir analizuoti OO sistemas ir duomenų bazes. „MagicDraw UML“ suteikia galimybę iš programos modelio generuoti Java, C#, C++ ir CORBA IDL programinį kodą arba iš jau esamo kodo sudaryti UML diagramas, tai yra PĮ kūrimo paketas skirtas programinės įrangos modeliavimui UML kalba ir jos kūrimo automatizavimui. MagicDraw UML yra vienas iš nedaugelio rinkoje esančių paketų, leidžiančių braižyti visas dvylika UML diagramų bei turintis UML diagramų semantinio teisingumo tikrinimo mechanizmą.

MagicDraw palaiko abipusę modelio ir kodo sinchronizaciją C++ ir Java programavimo kalboms. Leidžia generuoti kodą, nuskaityti UML modelį. Vartotojai turi galimybę atlikti atvirkštinę vykdomųjų .NET programų failų inžineriją atskleisdami programos modelį. Vykdomieji .NET failai gali būti sukompiliuoti iš programų parašytų Visual Basic, C#, JScript, Visual C++ kalbomis, taip pat ir kitomis kalbomis, naudojant ir ne Microsoft kompiliatorius (pavyzdžiui Eiffel, COBOL). MagicDraw integruota su IBM Eclipse 2.0 programavimo aplinka (IDE). Šios integracijos dėka vartotojai turi galimybę iš karto dirbti tiek su programos kodu, tiek su jos UML modeliu. Pakeitimai programos kode Eclipse aplinkoje iš karto atvaizduojami MagicDraw UML diagramoje ir atvirkščiai – pakeitimai UML klasių diagramoje iš karto atspindi pakeistame Java kalbos kode Eclipse aplinkoje [27].

Tai puikiai tinkanti priemonė programuotojams, nes ji turi galimybę iš sukurto programos modelio arba pagal pasirinktą dizaino šabloną sugeneruoti programos kodo fragmentus,

aprašančius sumodeliuotas klases, jų metodus bei ryšius, tačiau neturi XSLT importavimo galimybės ir leidžia generuoti tik tam tikrų kalbų kodo fragmentus.

1.6.4 FrontPage apžvalga

FrontPage programa pateikia profesionalaus projektavimo, kūrimo, duomenų ir publikavimo įrankius, kurie yra būtini kuriant dinamišką ir sudėtingą svetainę. Šios programos savybės, lankstumas ir funkcionalumas padeda kurti geresnius tinklapius.

FrontPage leidžia patobulinti tinklapių kūrimo įgūdžius trijose pagrindinėse srityse:

- Projektavime: leidžia naudotis patobulintais dizaino įrankiais ir sukurti įspūdingesnes svetaines. Nauji maketai ir grafikos įrankiai palengvina norimo dizaino svetainės sukūrimą.
- Kodavime: leidžia naudoti projektavimo įrankius geresniam kodui generuoti ir tokiu būdu įgyti daugiau kodavimo patirties. Vidiniai skripto rašymo įrankiai leidžia dirbti lengvai ir interaktyviai. Naudojant profesionalius kodavimo įrankius galima rašyti kodą greičiau, efektyviau ir kruopščiau.
- Išplėtime: leidžia kurti XML duomenų svetaines naudojantis pirmąja tokio pobūdžio rinkoje siūloma WYSIWYG tipo („what you see is what you get" - „ką matote, tą ir gaunate“) XSLT redagavimo programa. Patobulintos publikavimo savybės ir parinktys padeda greičiau publikuoti tinklapius [23].

Su FrontPage 2003 sukurto tinklapio pavyzdys (2 pav.)

2 pav. Su FrontPage 2003 sukurto tinklapio pavyzdys

FrontPage 2003 projektavimo įrankių pagalba galima generuoti išsamų HTML kodą. Taip pat galima pritaikyti programavimo žinias, naudojantis profesionalių kodavimo įrankių teikiamais privalumais. Skripto rašymo įrankių pagalba galima organizuoti dinaminį interaktyvų

bendravimą su savo tinklapio naršytojais. Dirbti kodavimo įrankiais nėra sudėtinga, todėl galima juos naudoti mokantis koduoti HTML kalba. Naudojant vaizdo perskyrimą galima matyti pokyčius kodo rodinyje (Code View), kurie ten automatiškai atsiranda padarius pakeitimus dizaino rodinyje (Design View). Galima pasirinkti, modifikuoti ir tvarkyti žymes naudojantis sparčiuoju žymių parinkikliu Quick Tag Selector ir redagavimo priemone Quick Tag Editor.

Su Microsoft IntelliSense technologijomis galima supaprastinti kodo užrašymą ir taip išvengti klaidų. Technologijos veikia su HTML, CSS stiliaus lapais, XSLT, Microsoft JScript ir Microsoft ASP.NET [23].

Yra galimybė pašalinti Microsoft Word arba kitomis tinklapio tvarkymo programomis sugeneruotus kodus.

Su FrontPage 2003 prisijungus prie Windows Server 2003 su Microsoft Windows SharePoint tarnybomis [23], galima atvaizduoti ir keisti duomenis, paimtus iš įvairių šaltinių, įskaitant XML, bei naudojantis WYSIWYG tipo redagavimo priemone, kurti interaktyvių duomenų svetaines. Vartotojai gali skelbti informaciją tinklapyje ir naudoti daugybę publikavimo pasirinkčių turėdami vien tik naršyklę.

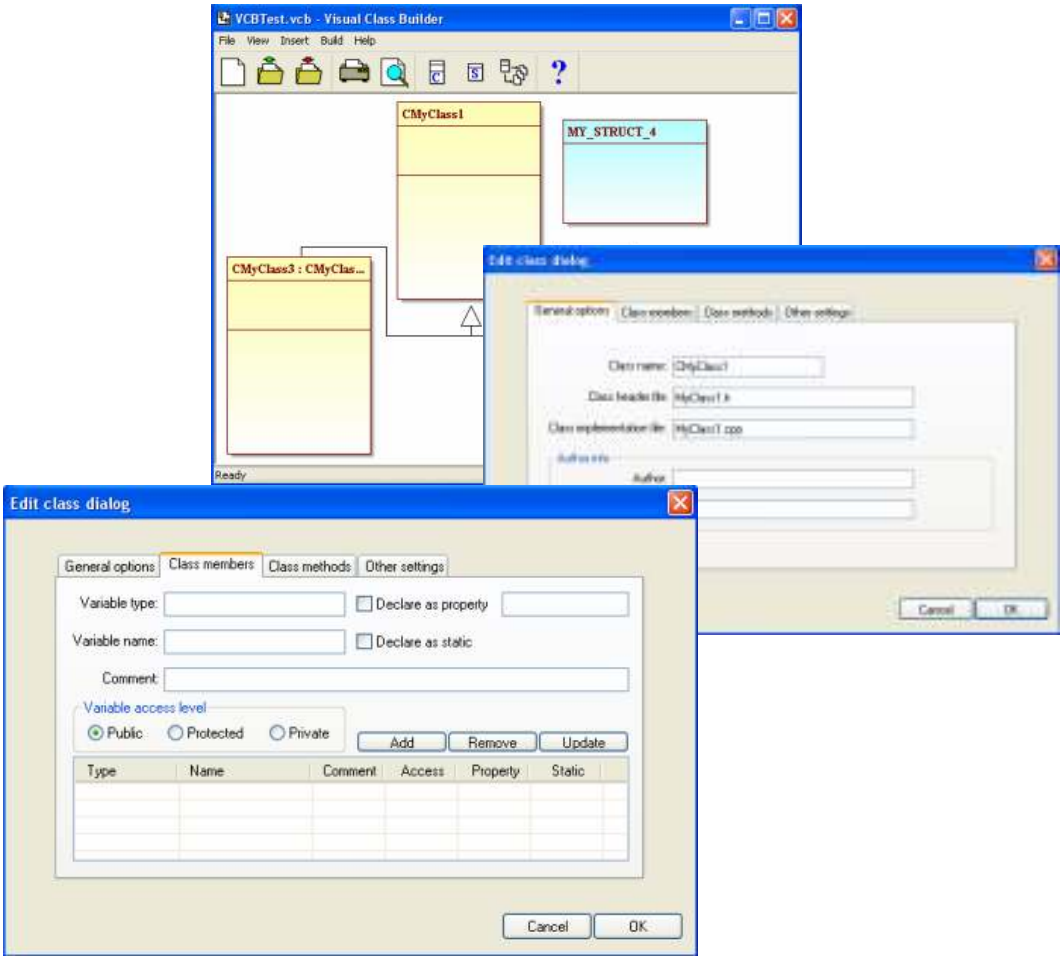
FrontPage 2003 pagalba galima kontroliuoti savo programos kodą: HTML, ASP ir XML pirminių kodų apsaugos charakteristikos programoje FrontPage 2003 leidžia importuoti tekstų redaktoriumi ar kitu HTML redagavimo įrankiu sukurtą kodą, jo nepertvarkant ir nepakeičiant.

Galima transformuoti HTML kodus į kodus, suderinamus su XML. Microsoft FrontPage metodai ir įrankiai leidžia sukurti ir prižiūrėti būtent tokį tinklapį, kokio nori pats vartotojas. Galima dirbti gerai pažįstamų Office programų aplinkoje neturint jokių programavimo žinių arba galima tiesiogiai naudoti HTML kodavimą.

1.6.5 VisualClassBuilder

VisualClassBuilder įrankis leidžia generuoti kodą iš klasių modelio, kuris yra panašus į UML klasių diagramą ir, kuris parodo vartotojo sukurtų klasių loginę struktūrą [33]. Taip pat šio įrankio pagalba galima kurti taikomąsias programas ir atlikti įvairius pakeitimus jų struktūroje naudojant grafinę vartotojo sąsają, o tai leidžia projektuotojams ir/ar programuotojams taupyti laiko resursus.

VisualClassBuilder turi grafinę vartotojo sąsają, kurios pagalba galima kurti ir valdyti UML klasių diagramas, atliekant pakeitimus matomus ekrane (3 pav.).



3 pav. VisualClassBuilder įrankio langai

Klasės objekto nustatymų dialogo lange galima įvesti klasės vardą, naudojamus failus, klasių diagramos failo pavadinimą, taip pat trumpą aprašymą, autoriaus pastabas arba autoriaus vardą, pavardę (pvz. Jei skirtingas klases kontroliuoja skirtingi žmonės). Nustatymai surašyti klasės kintamųjų ir savybių dialogo lange bus aiškiai suprantami C++ programuotojams. Taip kaip galima pridėti klasės kintamuosius, taip pat ir klasės metodus, kuriuos galima šiame lange ir aprašyti.

Taigi VisualClassBuilder v1.0 versija turi šias galimybes: klasių generavimą, struktūrinių elementų generavimą (tai atliekama panašiai kaip ir su klasių generavimu), paveldimų klasių.objektų palaikymą, projekto saugojimą/atidarymą, klasių diagramų spausdinimą.

1.6.6 Analizuotų sprendimų apibendrinimas

Prieš tai aptartos plačiai naudojamos automatizuoto projektavimo/kodo generavimo priemonės (Rational Application Developer for WebSphere Software, MagicDraw, Microsoft Visio, VisualClassBuilder) ir viena iš internetinės sąsajos kūrimo priemonių (FrontPage) vartotojams pateikia keletą programos kodo projektavimo atvejų:

- turimo kodo keitimas į modelį;
- modelio keitimas į programos kodą;
- turimo modelio ir kodo apjungimas.

Vienas nuo kito nepriklausantys pakeitimai kode arba modelyje gali būti perkeliami į modelį arba programos kodą be jokio duomenų praradimo. Kodą galima generuoti kiekvienam diagramos elementui bei visai elementų aibei. Tačiau šitos automatizuoto projektavimo priemonės palaiko C++, Java, Visual Basic, Corba IDL kalbų sintaksę bei leidžia generuoti klasių aprašus ir sąsajas iš klasių diagramų. Tai yra, šių įrankių pagalba galimas programos kodo generavimas labai ribotam programavimo kalbų kiekiui, programuotojai, kuriantys aplikacijas su ASP ir PHP tokiu atveju nebeturi galimybių naudotis šia PĮ. Lyginamoji įrankių anbalizė pateikiama 6 lentelėje.

6 lentelė.Lyginamoji sistemų analizė

Įrankis	UML	XML	Kodo generavimas	XSLT	Naujų bibliotekų įtraukimas
IBM Rational Application Developer for WebSphere Software	Taip	Taip	C++, Visual Basic, Java, Oracle8, CORBA	Ne	Ne
Microsoft Visio Enterprise 2002	Taip	Taip	C++, Visual Basic, Java (kodo karkasas)	Ne	Ne
MagicDraw UML	Taip	Taip	Java, C#, C++ ir CORBA IDL	Ne	Ne
VisualClassBuilder	Taip	Ne	C++	Ne	Ne
FrontPage 2003	Ne	Taip	HTML, ASP	Taip	Ne
Pasiūlytas algoritmas	Taip	Taip	Priklausomai nuo XSLT	Taip	Taip

Dar vienas pastebėjimas, kad šios programos arba leidžia tik projektuoti, tik kurti internetinę sąsają arba generuoti kodą, tačiau ribojant generuojamo kodo kalbą, t.y. nėra apjungimo, kad suprojektavus UML diagramų pagalba būtų galima sugeneravus sąsają matyti jos vaizdą ir gauti kokios norima kalbos kodą.

Todėl tikslas sukurti PĮ , kurioje bus galima išgauti bet kurios programavimo kalbos kodą, bus galima įtraukti naujas transformacijas, ko negalima padaryti anuose programose, bei bus galima apjungti projektavimą su sąsajos atvaizdavimu ir gauti bet kurios norimos programavimo kalbos kodą.

1.7. Architektūros ir galimų įgyvendinimo priemonių variantų analizė

Norint specifikuoti informacinę sistemą, reikia pasirinkti tam geriausiai tinkančią modeliavimo kalbą. Šiuo metu yra daug metodų, skirtų informacinėms sistemoms modeliuoti, pradedant matematiniais, baigiant grafiniais. Turėtume parinkti tokį metodą, kuris vaizdžiai specifikuotų kuriamą sistemą, būtų lengvai suprantamas naudotojui, be to, šiuo įrankiu turėtų būti nesudėtingai transformuojamas ir generuojamas programos kodas.

1.7.1 Vartotojo sąsajos projektavimo principų analizė

Tinkama vartotojo sąsaja su duomenų bazės vartojimu turi pateikti reikiamą informaciją vartotojui patogiu būdu. Reikia pripažinti, kad sąsaja gali būti viena reikšmingiausių sistemos komponentų. Sąsajos projektavimas turi būti orientuotas vartotojui. Sąsaja turi būti nuosekli ir logiška bei padėti vartotojui atstatyti klaidas.

Norint priimti gerus projektavimo sprendimus, reikia žinoti ir taikyti projektavimo principus, kurie yra išbandyti ir pasiteisinę praktikoje. Projektavimo šablonai [26] yra vienas iš būdų aprašyti elgsenos ir struktūros ryšius tarp komponentų. Šablonai apibendrina projektų panašumus ir aprašo kontekstą, kuriame konkretus šablonas gali būti pritaikytas nepriklausomai nuo naudojamos kalbos. Tai suteikia šiuos privalumus:

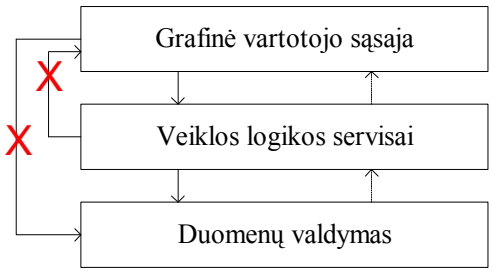
- naudojamos aukštesnio lygmens abstrakcijos;
- gali būti taikomas standartinis UML žymėjimas;
- yra išvystyta projektavimo metodika;
- galima padidinti projektavimo automatizavimą ir tuo pačiu produktyvumą;
- pasiekiama aukštesnė projekto kokybė;
- skatinamas atkartojimas ir dalijimasis patirtimi.

Naudojant projektavimo šablonus gali būti sumažinamas projekto sudėtingumas, pakeliamas abstrakcijos lygmuo ir užtikrinama aukštesnė projekto kokybė. Objektiškai orientuotų metodų [17], tame tarpe ir projektavimo šablonų, taikymas sulaukė daug dėmesio iš PĮ projektuotojų.

Vartotojo sąsajai projektuoti šiuo metu dažnai taikomas MVC (angl. Model-View-Controller) šablonas. Modelių peržiūros kontroleris (MVC) yra sistemos infrastruktūros karkasas GUI (grafinei vartotojo sąsajai) projektuoti, kuris leidžia įvairų objektų pristatymą ir atskiras sąveikas tarp objektų [19].

Vienas iš plačiausiai taikomų sprendimų - sluoksniuota posistemių architektūra, kurioje bendravimas tarp sluoksnių yra griežtai ribojamas hierarchijos - aukščiausio lygio sluoksnis bendrauja tik su po juo esančiu sluoksniu, šis su dar žemesniu ir t.t. Nėra leidžiamas

bendravimas iš žemesnių sluoksnių į aukštesnius arba “peršokant” tarpinius sluoksnius [11]. Toks sistemų organizavimas labai palengvina sistemų palaikomumą. Sluoksniuota architektūra dažnai jungiama su MVC šablonu, kuris siūlo skaidyti programinę įrangą į vartotojo sąsajos, duomenų modelio ir veiklos logikos dalis (4 pav.).



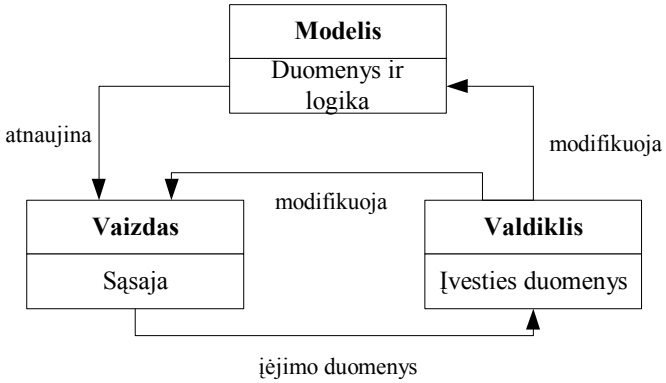
4 pav. Sluoksniuota trijų lygių architektūra pagal MVC projektavimo šabloną

Sluoksniuotos architektūros, pagrįstos MVC šablonu, schema puikiai tinka taikyti daugumoje programinės įrangos sistemų, kadangi tiek vartotojo sąsaja, tiek duomenys yra praktiškai bet kurioje programinėje įrangoje [26].

Pagrindinis MVC principas yra pareigų išskaidymas. MVC taikomojoje programoje modelio klasė susijusi tik su taikomosios programos struktūra ir logika. Modelio klasei nėra svarbu kaip būsena susijusi su vartotoju ir kokius įėjimo duomenis ji gauna. Priešingai, vaizdo klasė apima tik vartotojo sąsajos kūrimą pagal atnaujinimus, kuriuos gauna iš modelio. Ji nesusijusi su taikomosios programos logika, net su įėjimų apdorojimu, ji tik prižiūri, kad sąsaja atspindėtų esamą modelio būseną. Galiausiai, valdymo klasė apima tiktai vartotojo įvedamų duomenų transformavimą į atnaujintus, kurie atitinka modelį. Jai nesvarbu kaip įvedimo duomenys yra gaunami arba ką modelis daro su tais atnaujinimais.

Atskiriant kodą, kuris daro įtaką vartotojo sąsajai, į modelį, vaizdą ir valdymo klasių laukus, leidžiamas skirtingas tos pačios informacijos (modelio) atvaizdavimas (vaizdus), vartotojo sąsajų (vaizdų) papildymas, redagavimas, ištrynimas tiek sudarymo procese, tiek jo vykdymo metu, skatinamas pakartotinis naudojimas (vienas vaizdas gali būti naudojamas su skirtingais modeliais), leidžiamas sąsajos, logikos arba įvesties duomenų atnaujinimas taikomojoje programoje daugeliui vykdytojų tuo pačiu metu nedarant jokio efekto kitam kodui, galima kūrėjams susitelkti ties vienu taikomosios programos aspektu vienu metu.

5 paveikslas vaizduoja MVC ryšių ciklą. Pradinis taškas cikle, kaip paprastai būna, yra duomenų įvedimas. Kita programos dalis taip pat gali pradėti vykdyti ciklą modifikuojant modelį tiesiogiai.



5 pav. MVC ryšių ciklas

Reiktų pastebėti, kad 5 paveikslas rodo paprasčiausią MVC atvejį, su vienu modeliu, vienu vaizdu ir vienu valdikliu. Praktikoje dažnai yra naudojama du arba daugiau vaizdų vienam modeliui.

Vaizdas ir valdiklis yra nedalomi po vieną. MVC reikalauja, kad kiekvienas vaizdas turėtų valdiklį (net jeigu jis turi nulinę reikšmę (NULL)) ir kiekvienas valdiklis - vaizdą.

Kai kurie vaizdai neleidžia vartotojo duomenų įvedimo. Pavyzdžiui, vaizdas gali būti paprasta diagrama, grafikas, kuris negali būti redaguojamas. Kai vaizdas nepripažįsta duomenų įvedimo, jam nereikia valdiklio, paverčiančio naudotojo įvedamus duomenis į modelio atnaujinimą [26].

Kuriamos PĮ sugeneruotas programos kodas atitiks MVC architektūrą. Visi duomenys saugomi XML formate, o vartotojo sąsaja ir jos logika bus sugeneruojama transformacijų pagalba.

1.7.2 Modeliavimo kalbų analizė

Šiuo metu UML *de facto* tapo programinės įrangos sistemų formaliojo specifikavimo kalba. Ja galima vizualizuoti, specifikuoti, konstruoti ir dokumentuoti kuriamas sistemas. UML kalbą sudaro keliolika diagramų rūšių, leidžiančių formaliai specifikuoti įvairiausias sistemas skirtingais požiūriais [8]. UML modeliavimo kalba yra standartinė projektavimo priemonė, kuri visuotinai naudojama pasaulyje ir praktiškai neturi rimtesnių “konkurentų”. Tačiau, siekiant neapsiriboti išankstinėmis nuostatomis, buvo palyginta keletas geriau žinomų ir plačiau naudojamų specifikavimo metodų:

- OMT - Objektų modeliavimo technika (angl. *Object Modeling Technique*) [17].
- SDL - Specifikavimo ir aprašymo kalba (angl. *Specification and Description Language*) [12].
- UML - Unifikuota modeliavimo kalba (angl. *Unified Modeling Language*) [3, 1].

- VHDL - labai aukšto lygio projektavimo kalba (angl. *Very High-level Design Language*) [11, 18].

Šių modeliavimo kalbų palygunimas ir palyginimo kriterijai pateikiami 7 ir 8 lentelėse.

7 lentelė. Modeliavimo kalbų palyginimo kriterijai

Nr.	Kriterijus	Aprašymas
1.	Tikslumas	Ši savybė nustato, ar naudojantis technologija galima tiksliai (idealiai) aprašyti objektą, ar ji tik leidžia suformuoti jo apytikslį vaizdą. Galimos reikšmės: taip ir ne.
2.	Grafinis atvaizdavimas	Šis kriterijus nurodo, ar specifikacija įtraukia į save grafinius elementus (pvz., paveikslėlius, diagramas, grafus). Galimos reikšmės: taip ir ne.
3.	Objektiškai orientuotas	Šis atributas apibrėžia, ar specifikacija naudoja tokias sąvokas kaip paveldimumas, klasė, polimorfizmas ir t.t. Galimos reikšmės: taip ir ne.
4.	Populiarumas	Savybė, parodanti kiek populiari naudojime yra technologija. Populiarumas apibrėžia technologijos žinomumą, dažną naudojimą, darbui su ja skirtų įrankių gausą ir t.t. Galimos reikšmės: nepopuliarus, pusiau populiarus, populiarus

8 lentelė. Modeliavimo kalbų palyginimas

Kriterijus / Metodas	Tikslumas	Grafinis atvaizdavimas	Objektiškai orientuotas	Populiarumas
OMT	tikslus	taip	taip	populiarus
SDL	tikslus	taip	taip	pusiau populiarus
UML	pusiau tikslus	taip	taip	populiarus
VHDL	tikslus	ne	ne	pusiau populiarus

Šiame darbe, palyginus modeliavimo metodus, informacinių sistemų modeliavimui nuspręsta naudoti UML kalbą dėl šių priežasčių:

- šiuo metu UML yra populiariausia ir plačiausiai naudojama informacinių sistemų modeliavimo kalba.
- UML siūlo grafinį sistemos vaizdavimo būdą priimtiną ir kūrėjui, ir užsakovui.

Padidinti specifikacijos tikslumą į UML modelį bus įvesti būsenų atributai, kurie leis išsamiau nusakyti sistemos struktūrą ir veikimą bei pasitarnaus kaip informacija programos kodo generavimui.

Naudojant UML, galima modeliuoti sistemą skirtingais abstrakcijos lygiais. Pavyzdžiui, kuriamos sistemos esybes ir jų ryšius vaizduojanti klasių diagrama gali būti naudojama reikalavimų analizės metu, o vėliau pagal ją gali būti sukuriama detali realizacijos klasių diagrama, kurioje nurodomi specifiniai realizacijos kalbos duomenų tipai, atliekamos reikalingos

ryšių transformacijos, pridedamos tik realizacijai reikalingos savybės, tokios kaip identifikaciniai kodai. Modeliavimas skirtingais abstrakcijos lygiais leidžia glaudžiau susieti programinės įrangos architektūros projektavimą su reikalavimų analizės veikla.

UML specifikavimo kalba leidžia aprašyti tikrovę supaprastinus ją iki valdomo informacijos kiekio, sisteminti bei klasifikuoti surinktas žinias. UML modeliavimas - tai formalizuotas būdas užfiksuoti, perduoti ir pasinaudoti žiniomis.

Projektuojant sistemas, naudojama tam tikra griežtai apibrėžta sutartinių ženklų sistema, leidžianti vienareikšmiškai aprašyti pageidaujamų sistemos paslaugų ir perdavimo aplinkos elgseną. Šiuo metu unifikuotoji modeliavimo kalba (UML – Unified Modeling Language) *de facto* tapo programinės įrangos sistemų formaliojo specifikavimo kalba [8]. UML vartojama kaip standartinis įrankis kuriant programos "brėžinius". Ja galima vizualizuoti, specifikuoti, konstruoti ir dokumentuoti kuriamas sistemas:

- UML – tai kalba. Ja sudaro žodynas ir taisyklės, pagal kurias jungiami turimi žodžiai ir gaunamos reikšminės konstrukcijos. Modeliavimo kalboje žodynas ir taisyklės orientuoti i koncepcinį ir fizinį sistemos pateikimą.
- UML – tai vizualizavimo kalba. Tai ne paprastas grafinių simbolių rinkinys. Kiekvienas simbolis turi griežtai apibrėžtą semantiką, kuri leidžia vieno programuotojo parašytą modelį vienaprasmiškai suprasti kitam arba netgi programai.
- UML – tai specifikavimo kalba. Šiame kontekste terminas *specifikavimas* reiškia tikslių, nedviprasmiškų ir baigtų modelių kūrimą. UML leidžia specifikuoti visus sprendimus susijusius su analize, projektavimu ir realizacija, aptinkami programinės sistemos kūrimo metu [34].
- UML – tai konstravimo kalba. Tai ne programavimo kalba, bet ja sukurti modeliai gali buti tiesiogiai interpretuojami įvairiomis programavimo kalbomis. Kitais žodžiais tariant, UML modeli galima atvaizduoti tokiomis kalbomis kaip C++, Java, Visual Basic ir netgi reliacinėmis duomenų bazės lentelėmis.

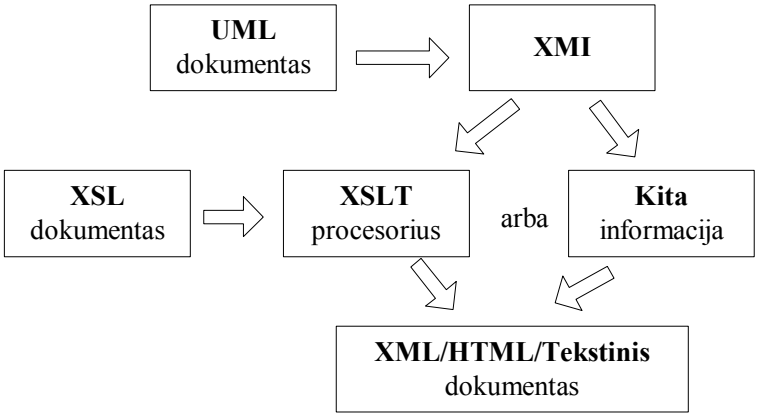
UML kalbą sudaro keletas diagramų rūšių, leidžiančių formaliai specifikuoti įvairias sistemas skirtingais požiūriais. Statiniai sistemos aspektai modeliuojami naudojant tokius elementus kaip klasių ir objektų diagramos. Jomis galima vizualizuoti, specifikuoti ir dokumentuoti viską, kas yra sistemos viduje, įskaitant klases, sąsajas, komponentus, mazgus, visu šių išvardintų esybių atskirus egzempliorius, taip pat ir jų tarpusavio sąveiką [19].

Dinaminiams sistemos aspektams UML kalboje modeliuoti yra naudojamos sąveikos, būsenų ir veiklos diagramos. Kuriamoje PĮ būsenų (state) ir veiklos (activity) diagramomis naudotojas galės modeliuoti norimą sistemą.

1.7.3 Duomenų perdavimo ir transformavimo priemonių analizė

UML modeliams perduoti žiniatinklyje ar keistis jais tarp įvairių modeliavimo įrankių Object Management Group (OMG) grupė 1997 metais pasiūlė XMI (angl. *XML Metadata Interchange Format*) duomenų perdavimo formatą [13]. XMI tiesioginė paskirtis buvo ir yra keistis modelių meta duomenimis. Jis savyje apjungia tris standartus (6 pav.) [29]:

1. XML (angl. *Extensible Markup Language*) – W3C grupės standartas.
2. UML – OMG grupės modeliavimo standartas.
3. MOF (angl. *Meta Object Facility*) – OMG grupės standartas meta duomenų saugojimui [28].



6 pav. XMI duomenų perdavimo formatas

UML modelių tikslumui padidinti siūloma surinkti parametrus, apibūdinančius konkrečias sistemos būsenas. Parametrų surinkimui, saugojimui ir perdavimui tinkama XML sintaksė. XML – lankstus tekstinis formatas, sukurtas SGML (angl. Standard Generalized Markup Language) kalbos pagrindu, kuri skirta įvairių tipų duomenų apsikeitimui.

XML kalba pasirinkta , nes [10]:

- **XML yra meta kalba.** XML pagalba gali būti apibrėžiama ir aprašoma bet kokio tipo informacija. Iš esmės XML yra praplečiama. Vartotojai, vartotojų grupės ir organizacijos naudoja XML, kaip nuo platformos nepriklausomą kalbą. Ji apibrėžia sintaksę, naudojamą įvairiems tikslams ir visų tipų duomenų modelių specifikavimui.
- **XML yra tekstinis formatas ir jį lengva skaityti.** Dokumentai lengvai skaitomi tiek taikomosioms programoms, tiek žmonėms. Pagal apibrėžimą XML dokumento turinys yra sužymėtas žymėmis (angl. *tag*). Tokios žymės kaip <mygtukas> ar <tekstas> aiškiai nusako informacijos, kuri saugoma dokumente, pobūdį. Taip pat kiekviename XML dokumente gali būti nurodyta, kaip tas

žymes interpretuoti (schema). Dokumento skaitymas yra paprastas ir nereikalauja programos, kuria buvo sukurtas tas dokumentas.

- **XML yra nepriklausomas nuo jo atvaizdavimo.** XML atskiria dokumento turinį nuo jo atvaizdavimo schemos. Stilių bylos nusako dokumento elementų formatavimą, jos gali būti pritaikomos bet kokiam XML dokumentui be jo turinio koregavimo [26, 27].
- **XML palaiko daugiakalbiškumą.** XML naudoja Unicode kodavimą, todėl tai leidžia atvaizduoti visų pasaulio kalbų simbolius, įskaitant kinų ir japonų, o tai svarbu daugiakalbėje visuomenėje.
- **XML yra paprastai palaikomas.** XML dokumentus lengvai interpretuoja daugelis programų, visos žiniatinklio naršyklės. Šių dokumentų koregavimui užtenka paprasčiausio tekstinio redaktoriaus.
- **XML yra atviras standartas.** XML standartas yra palaikomas daugelio pagrindinių informacinių sistemų kūrėjų. Jį palaiko didelės IT kompanijos tokios, kaip IBM, Microsoft, Oracle, Sun Microsystems.

Sudarytai sistemos specifikacijai, užrašytai UML modeliais ir papildytai XML duomenimis pakeisti į programos kodą, reikia pritaikyti transformacijas. Taigi, toliau bus panagrinėjamos jau esamos standartizuotos transformavimo kalbos su dar tik bandančiomis įsitvirtinti:

- XSLT - Išplečiamos stilių kalbos transformacijos (angl. Extensible Stylesheet Language Transformations) [14]
- VMT - Vaizdinė modelių transformacija (angl. Visual Model Transformation) [32]
- QVT - Užklausa/Vaizdas/Transformacija (angl. Query/View/Transform) [30]

Toliau visos aukščiau pateiktos transformavimo kalbos bus palygintos pagal suformuotus kriterijus ir tinkamumą šio darbo tikslams pasiekti (9 ir 10 lentelės).

9 lentelė. Transformavimo kalbų palyginimo kriterijai

Nr.	Kriterijus	Apibūdinimas
1	Populiarumas	Naujos kalbos reikalauja daugiau pastangų jas įsisavinant ir pritaikant uždavinių sprendimui, todėl svarbu rasti optimalų sprendimą, galintį pasiūlyti maksimaliai gerą rezultatą, panaudojant standartines priemones. Galimos reikšmės: taip, ne, dalinai
2	Paveldimumas	Atributas, apibūdinantis transformacijų atnaujinimo ir palaikymo paprastumą. Galimos reikšmės: taip, ne

3	Grafinis atvaizdavimas	Ši savybė svarbiausia vieną UML modelį keičiant į kitą. Grafiniai modeliai pateikia aukštesnio lygio transformuojamos sistemos vaizdą, kuris yra lengviau suprantamas sistemos projektuotojams nei žodinis aprašas. Galimos reikšmės: taip, ne
4	Dvikryptis	Transformavimo kalba yra dvikryptė, jei ji palaiko tokį transformacijos aprašą, kuris yra pritaikomas, keičiant specifikaciją abiem kryptimis. Tokių transformacijų specifikavimas ir palaikymas yra paprastesnis. Galimos reikšmės: taip, ne
5	XML palaikymas	Šis kriterijus nurodo, ar standartiškai yra numatytas XML palaikymas, t.y., ar transformacija gali priimti arba sukurti XML. XML palaikymas į save įtraukia pagrindinių kalbos elementų, atributų ir aprašų traktavimą. Kriterijus naudingas, atliekant teksto konvertavimą į UML ir atvirkščiai. Duomenų perdavimui šiame darbe bus naudojamas XML, todėl jo palaikymas transformacijose yra svarbus. Galimos reikšmės: taip, ne
6	Atvirkštinis projektavimas (angl. *reverse engineering*)	Teksto keitimas į UML. Programos kodas dažniausiai egzistuoja be sąryšio su aukštesnio lygio, grafiniais UML modeliais. Galimos reikšmės: taip, ne
7	UML keitimas į tekstą	Savybė reikalinga specifikacijos transformacijai į programos kodą. Galimos reikšmės: taip, ne
8	Standartizuotas	Savybė, nusakanti ar transformacija yra paskelbta standartu ar ne. Galimos reikšmės: taip, ne

10 lentelė. Transformavimo kalbų palyginimas

Metodas / **Kriterijus**	**XSLT**	**VMT**	**QVT**
Populiarumas	taip	dalinai	ne
Paveldimumas	ne	taip	taip
Grafinis atvaizdavimas	ne	taip	taip
Dvikryptis	ne	ne	taip
XML palaikymas	taip	ne	ne
Atvirkštinis projektavimas	taip	ne	ne
UML keitimas į tekstą	taip	ne	ne
Standartizuotas	taip	ne	ne

XML duomenų transformacijai naudosiu XSL (angl. Extensible Stylesheet Language) transformacijas. XSL kalba skirta XML duomenų atvaizdavimui. Tai pilnai specifikuota, populiari ir žinoma XML transformavimo kalba. Jos naudojimo privalumas – programuotojui nereikia įsisavinti naujos kalbos sintaksės, o tai pagerina darbo našumą.

1.8. Apibendrinimas

Visą sistemą nutarta kurti panaudojant programavimo kalbą C#.

Informacinės sistemos modeliavimui nuspręsta naudoti UML kalbą dėl šių priežasčių:

- šiuo metu UML yra populiariausia ir plačiausiai naudojama informacinių sistemų modeliavimo kalba.
- UML siūlo grafinį sistemos vaizdavimo būdą priimtiną ir kūrėjui, ir užsakovui.

Duomenų saugojimui, perdavimui bei transformavimui pasirinkta XML kalba.

Modelių transformavimui į programos kodą - XSL transformacijos. Jų pasirinkimą lėmė suderinamumas su XML kalba, standartų palaikymas ir paprastumas.

Sistema projektuojama, naudojant RUP projektavimo metodiką ir CASE priemonę „MS Visio Professional 2003“, kuri leidžia greičiau ir efektyviau sudaryti bei analizuoti reikiamas UML diagramas. Renkantis projektavimo įrankį buvo atsižvelgta į mums reikalingą programos funkcionalumą. “MS Visio Professional 2003” UML projektavimo sąsaja ir galimybėmis yra galingas įrankis, atitinkantis kūrėjo poreikius, projektuojant kuriamą informacinę sistemą.

1.9. Siekiamos sistemos apibrėžimas

Ši, interneto sąsajos generavimo iš dialogo specifikacijos, sistema skirta projektuotojams/programuotojams, kurie kuria taikomąsias programas arba nori gauti sistemos kodą.

Grafinė vartotojo sąsaja - tam tikras bendravimo susitarimas tarp dviejų programinių ar aparatinių komponentų. Sąsajos skirtos abstrakčiai aprašyti apsikeitimą duomenimis, kad vienam komponentui nereikėtų žinoti nieko daugiau apie kitą komponentą. Komponentas gali būti įrenginys, bibliotekinė funkcija, programos modulis, programa, klasės objektas ir t.t.

Generavimas – automatizuotas kūrimas.

Dialogas – sąryšis tarp vartotojo ir sistemos.

Specifikacija – charakteristikos, kurios nusako objekto atributus, jų savybes.

Siekiama sukurti interneto sistemų kūrimo metodą ir įrankį, kuris leistų generuoti interneto sąsają iš vartotojo dialogo specifikacijos. Sąsajos programinę įrangą (PĮ) yra sudėtinga sukurti, nes dažnai ji turi valdyti daug objektų, kurie turi tarpusavio ryšius bei atlieka tam tikras funkcijas. Daugelis egzistuojančių programų leidžia sukurti tik tam tikrus puslapius, o scenarijų turi realizuoti pats projektuotojas. Todėl reiktų tokios priemonės, kuri leistų kurti scenarijų, t.y. aprašyti ar grafiškai atvaizduoti kas su kuo jungiasi, kokie perėjimai tarp puslapių.

Kuriamas įrankis turėtų automatizuoti PĮ scenarijaus sudarymo procesą.

1.10. Darbo tikslas ir siekiami privalumai

Darbo tikslas – sukurti interneto sistemų kūrimo metodą ir įrankį, kuris leistų generuoti interneto sąsają iš vartotojo dialogo specifikacijos. Sąsajos programinę įrangą (PĮ) yra sudėtinga sukurti, nes dažnai ji turi valdyti daug objektų, kurie turi tarpusavio ryšius bei atlieka tam tikras funkcijas. Daugelis egzistuojančių programų leidžia sukurti tik tam tikrus puslapius, o scenarijų turi realizuoti pats projektuotojas. Todėl reiktų tokios priemonės, kuri leistų kurti scenarijų, t.y. aprašyti ar grafiškai atvaizduoti kas su kuo jungiasi, kokie perėjimai tarp puslapių.

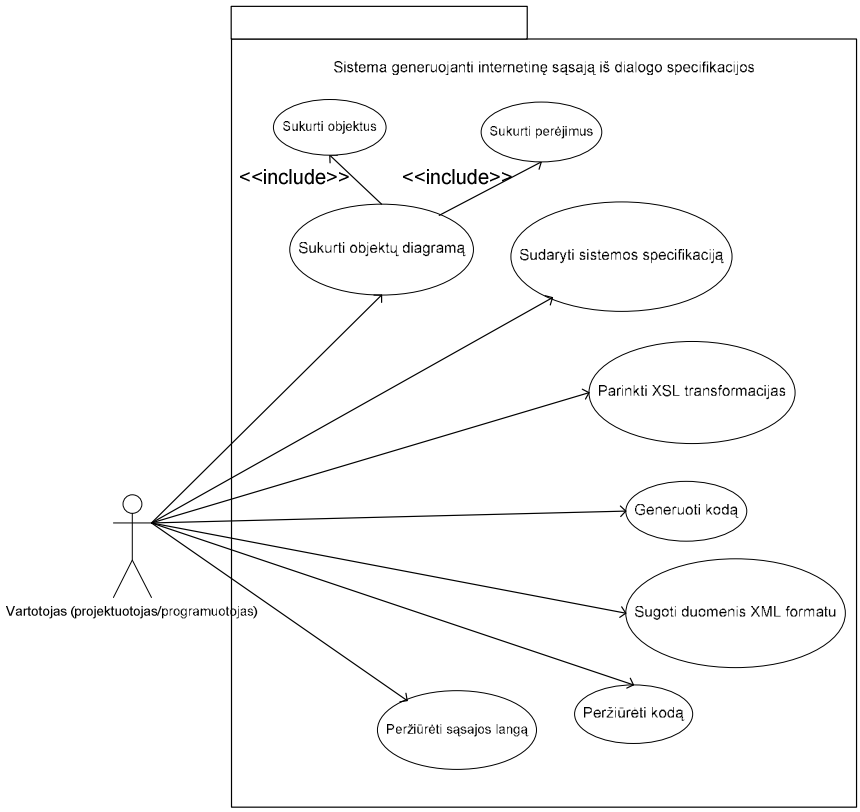
Siekiamas įrankis automatizuoja PĮ scenarijaus sudarymo procesą. Kūrimo metodui realizuoti remiamasi projektavimo procesu, pagal kurį reikia aprašyti kuriamos sistemos panaudojimo atvejus, sudaryti vartotojo ir sistemos sekų diagramas, būsenų diagramas, apimančias scenarijų aibę kiekvienam vartotojo tipui, po to sudaryti specifikacijas, kurias būtų galima dinamiškai transformuoti į puslapius.

1.11. Kompiuterizuojamos sistemos funkcijos

Kuriama sistema yra skirta sumažinti žmogiškųjų išteklių poreikį ir jų panaudojimą internetinių sistemų ar programinės įrangos kūrimo procese. Šioje PĮ, bus galima išgauti bet kurios programavimo kalbos kodą, bus galima įtraukti naujas transformacijas, ko negalima padaryti nagrinėtose jau sukurtose programose, bei bus galima apjungti projektavimą su sąsajos atvaizdavimu.

Siekiamas įrankis turėtų automatizuoti PĮ scenarijaus sudarymo procesą. Kūrimo metodui realizuoti bus remiamasi projektavimo procesu, pagal kurį reikia aprašyti kuriamos sistemos panaudojimo atvejus, sudaryti vartotojo ir sistemos sekų diagramas, būsenų diagramas, apimančias scenarijų aibę kiekvienam vartotojo tipui, po to sudaryti specifikacijas, kurias būtų galima dinamiškai transformuoti į puslapius.

Besinaudojantiems programų sistema vartotojams yra užtikrinamas sistemos funkcionalumas, kuris aprašytas (11-19 lentelėse) ir pavaizduotas toliau (7 pav.). Detalesnis panaudojimo atvejų aprašymas pateiktas reikalavimų specifikacijoje.



7 pav. Sistemos panaudojimo atvejų diagrama

1.12. Panaudojimo atvejų sąrašas

Panaudojimo atvejų aprašymai/specifikacijos pateikti 11-19 lentelėse.

11 lentelė. Panaudojimo atvejo „Sukurti būsenų diagramą“ specifikacija

1. Panaudojimo atvejis	Sukurti objektų diagramą
Aktoriai	Projektuotojas/programuotojas, sistema
Ryšiai su kitais PA	Apima panaudojimo atvejį “ Sukurti objektus ” (ryšio tipas <<include>>) Apima panaudojimo atvejį “ Sukurti perėjimus ” (ryšio tipas <<include>>)
Prieš sąlygos	Sistema turi būti veikianti Vartotojas atsidaro pradinį sistemos darbo langą
Sužadinimo sąlyga	Vartotojas pasirenka iš įrankių juostos braižymo įrankį.
Po sąlyga	Vartotojas gauna objektus ir perėjimus tarp jų
Pagrindinis scenarijus	1. Vartotojas atsidaro programos langą 2. Pasirenka įrankį braižymui iš įrankių juostos 3. Vartotojas braižo objektus 4. Vartotojas išsaugo atliktą darbą (išsaugo diagramą) pasirinkdamas įrankį iš meniu ar įrankių juostos
Alternatyvus scenarijus	Neteisingai nubraižyta diagrama (pvz.: jei neparinkti perėjimai tarp objektų) Vartotojas nutraukia darbą su sistema.

12 lentelė. Panaudojimo atvejo „Sukurti objektus “ specifikacija

2. Panaudojimo atvejis	Sukurti objektus
Aktoriai	Projektuotojas/programuotojas, sistema

Ryšiai su kitais PA	Nėra
Prieš sąlygos	Sistema turi būti veikianti Vartotojas atsidaro pradinį sistemos darbo langą
Sužadinimo sąlyga	Vartotojas pasirenka iš įrankių juostos braižymo įrankį.
Po sąlyga	Vartotojas gauna objektus, kuriuos vėliau gali susieti ryšiais (perėjimais)
Pagrindinis scenarijus	1. Vartotojas atsidaro programos langą 2. Pasirenka įrankį iš įrankių juostos objektui braižyti 3. Vartotojas braižo objektus 4. Vartotojas išsaugo atliktą darbą
Alternatyvus scenarijus	Vartotojas nutraukia darbą su sistema.

13 lentelė. ***Panaudojimo atvejo „Sukurti perėjimus " specifikacija***

3. Panaudojimo atvejis	Sukurti perėjimus
Aktoriai	Projektuotojas/programuotojas, sistema
Ryšiai su kitais PA	Nėra
Prieš sąlygos	Sistema turi būti veikianti, vartotojas atsidaro pradinį sistemos darbo langą
Sužadinimo sąlyga	Vartotojas pasirenka iš įrankių juostos braižymo įrankį.
Po sąlyga	Vartotojas gauna objektus susietus perėjimais
Pagrindinis scenarijus	1. Vartotojas atsidaro programos langą 2. Pasirenka įrankį iš įrankių juostos perėjimams braižyti 3. Vartotojas braižo perėjimus tarp objektų 4. Vartotojas išsaugo atliktą darbą
Alternatyvus scenarijus	Vartotojas nutraukia darbą su sistema.

14 lentelė. Panaudojimo atvejo „Sudaryti sistemos specifikaciją" specifikacija

4. Panaudojimo atvejis	Sudaryti sistemos specifikaciją
Aktoriai	Projektuotojas/programuotojas, sistema
Ryšiai su kitais PA	Nėra
Prieš sąlygos	Turi būti sukurta būsenų diagrama
Sužadinimo sąlyga	Vartotojas paspaudžia ant objekto arba perėjimo ir atsidariusiame dialogo lange aprašinėja objekto atributus, funkcionalumą (pvz.: koks mygtukas turi būti lange ir kokius objektus jis susieja tarpusavyje).
Po sąlyga	Vartotojas gauna specifikaciją
Pagrindinis scenarijus	1. Vartotojas atsidaro programos langą 2. Pasirenka objektą arba dar pradžioje kai pasirenka objekto brėžimą yra galimybė iš karto aprašyti objekto atributus 3. Paspaudžia ant objekto arba perėjimo 4. Atsidariusiame dialogo lange aprašinėja objektų atributus, perėjimus, nurodo pavadinimą, ilgį, tipą ir pan. 5. Specifikacijas galima redaguoti 6. Vartotojas išsaugo atliktą darbą
Alternatyvus scenarijus	Nėra

15 lentelė. Panaudojimo atvejo „Parinkti XSL transformacijas" specifikacija

5. Panaudojimo atvejis	Parinkti XSL transformacijas
Aktoriai	Projektuotojas/programuotojas, sistema
Ryšiai su kitais PA	Nėra

Prieš sąlygos	Nėra
Sužadinimo sąlyga	Vartotojas įtraukia jau parašytas transformacijas
Po sąlyga	Vartotojas gauna sugeneruotą programos kodą
Pagrindinis scenarijus	1. Vartotojas atsidaro programos langą 2. Įtraukia jau parašytas transformacijas 3. Pritaiko transformacijas 4. Gauna kodą
Alternatyvus scenarijus	-

16 lentelė. Panaudojimo atvejo „Saugoti duomenis XML formatu“ specifikacija

6. Panaudojimo atvejis	Saugoti duomenis XML formatu
Aktoriai	Projektuotojas/programuotojas, sistema
Ryšiai su kitais PA	Nėra
Prieš sąlygos	Turi būti nubraižyta objektų diagrama Turi būti sudaryta specifikacija
Sužadinimo sąlyga	Paspaudžiamas mygtukas saugoti XML
Po sąlyga	Vartotojas gauna XML kodą
Pagrindinis scenarijus	1. Vartotojas atsidaro programos langą 2. Atsidaro reikiamą objektų diagramą, kuriai jau yra parašyta specifikacija 3. Saugo duomenis XML formatu 4. Peržiūri XML kodą
Alternatyvus scenarijus	Nėra

*17 lentelė. **Panaudojimo atvejo „Generuoti kodą“ specifikacija***

7. Panaudojimo atvejis	Generuoti kodą
Aktoriai	Projektuotojas/programuotojas, sistema
Ryšiai su kitais PA	Nėra
Prieš sąlygos	Turi būti nubrėžta objektų diagrama Turi būti sudaryta specifikacija Turi būti pritaikytos transformacijos Turi būti išsaugoti XML duomenys
Sužadinimo sąlyga	Nėra
Po sąlyga	Vartotojas gauna programos kodą
Pagrindinis scenarijus	Vartotojas sukūręs objektų diagramą, ją specifikavęs, pritaikęs XSLT gali gauti programos kodą. Generuoti kodą, tai yra apjungti XML su XSLT ir gauti rezultatą programos kodo pavidalu.
Alternatyvus scenarijus	Nėra

18 lentelė. Panaudojimo atvejo „Peržiūrėti kodą“ specifikacija kalbų palyginimas

8. Panaudojimo atvejis	Peržiūrėti kodą
Aktoriai	Projektuotojas/programuotojas, sistema
Ryšiai su kitais PA	Nėra
Prieš sąlygos	Turi būti nubraižyta objektų diagrama Turi būti sudaryta specifikacija
Sužadinimo sąlyga	Pasirinkti kodo peržiūros puslapį (XML/XSL)
Po sąlyga	Vartotojas mato kodą
Pagrindinis scenarijus	Vartotojas pasirinkęs mygtuką gali peržiūrėti programos kodą

Alternatyvus scenarijus	Nėra

19 lentelė. Panaudojimo atvejo „Peržiūrėti sąsajos langą“ specifikacija

9. Panaudojimo atvejis	Peržiūrėti sąsajos langą
Aktoriai	Projektuotojas/programuotojas, sistema
Ryšiai su kitais PA	Nėra
Prieš sąlygos	Turi būti nubraižyta būsenų diagrama Turi būti sudaryta specifikacija Turi būti pritaikytos transformacijos
Sužadinimo sąlyga	Nėra
Po sąlyga	Vartotojas gauna internetinės sąsajos langą
Pagrindinis scenarijus	Vartotojas pasirinkęs peržiūrėjimo mygtuką gali peržiūrėti internetinės sąsajos langą
Alternatyvus scenarijus	Nėra

1.13. Rizikos faktorių analizė

Numatytos rizikos, jų tikimybės ir įtaka visam projektui yra pateiktos 20 lentelėje.

20 lentelė. Sistemos kūrimo rizikos

Nr.	Rizika	Tikimybė	Įtaka
1.	Reikalavimų pasikeitimas	Žema	Rimta
2.	Techninės įrangos gedimais	Vidutiniška	Leistina
3.	Architektūros pasikeitimas	Vidutiniška	Rimta
4.	Prasta dokumentacija	Vidutiniška	Leistina
5.	Neefektyvios CASE priemonės	Žema	Rimta
6.	Reikiamų komponentų nebuvimas	Aukšta	Maža
7.	Žmogaus sveikatos problemos	Vidutiniška	Leistina
8.	Projekto vėlavimas	Vidutiniška	Leistina
9.	Projekto nutraukimas	Žema	Rimta

Galimi rizikos vengimo ir jos žalos mažinimo variantai pateikti 21 lentelėje.

21 lentelė. Sistemos kūrimo rizikų planas

Nr.	Rizikos faktorius	Problemos sprendimas
1.	Reikalavimų pasikeitimas	Išgauti visus reikalavimus kuo detaliau dar analizės procese. Laiko paliktas rezervas, skirtas pasikeitusiems reikalavimams realizuoti. Reikalavimų skirstymas pagal prioritetus.
2.	Techninės įrangos gedimas	Atsarginių duomenų kopijų darymas. Papildomos techninės įrangos, apsaugančios nuo visiško sistemos sustojimo, naudojimas.
3.	Architektūros pasikeitimas	Kaip įmanoma detaliau specifikuoti sistemos architektūrą specifikavimo fazėje, siekiant eliminuoti visus neaiškumus ir prieštaravimus.
4.	Prasta dokumentacija	Kaip įmanoma tiksliau ir išsamiau rengti dokumentaciją.

5.	Neefektyvios CASE priemonės	Naudoti patikrintas CASE priemones, turėti joms alternatyvas.
6.	Reikiamų komponentų nebuvimas	Pakankamas laiko rezervas reikiamam komponentui sukurti arba modifikuoti esamus pritaikant pagal poreikius.
7.	Žmogaus sveikatos problemos	Skiepai ir kitos panašios medicininės priemonės. Laiko rezervas.
8.	Projekto vėlavimas	Nerealizuoti mažiausio prioriteto reikalavimų.
9.	Projekto nutraukimas	Bandyti rasti kitą sistemos užsakovą.

1.14. Rezultato kokybės kriterijai

Programinės įrangos kokybės įvertinimui galima panaudoti ISO/IEC 9126 standartą. Šis standartas pateikia programinės įrangos kokybės įvertinimo modelį, kuris gali būti taikomas įvairios programinės įrangos kokybės vertinimui. ISO/IEC 9126 standartas pateikia 6 programinės įrangos kokybės charakteristikas bei pateikia rekomenduojamas kokybės sub-charakteristikas.

Pagal kokybės vertinimo modelį, kokybės vertinimas atliekamas tokiu metodu: nustatomos kokybės vertinimo charakteristikos; charakteristikoms nurodomos sub-charakteristikos; sub-charakteristikoms yra priskiriamos kokybės metrikos, kurių pagalba yra atliekamas kokybės matavimas.

ISO/IEC 9126 pateikiamos kokybės charakteristikos, pavaizduotos 22 lentelėje.

22 lentelė. Kokybės charakteristikos

Charakteristika	Apibūdinimas
Funkcionalumas	Ar programinėje įrangoje yra prieinamos reikalaujamos funkcijos?
Patikimumas	Kaip patikima yra programinė įranga?
Panaudojamumas	Ar lengva naudoti programinę įrangą?
Efektyvumas	Kiek efektyvi yra programinė įranga?
Palaikomumas	Kaip lengva modifikuoti programinę įrangą?
Portatyvumas	Kaip lengva perkelti programinę įrangą į kitą veikimo aplinką?

ISO/IEC 9126 standarte rekomenduojamos charakteristikų sub-charakteristikos pateiktos 23 lentelėje.

23 lentelė. Kokybės charakteristikų sub-charakteristikos

Charakteristika	Subcharakteristika	Apibūdinimas
Funkcionalumas	Tinkamumas	Produkto funkcijų, atliekančių reikiamas užduotis, pilnumas bei atitikimas reikalavimams.

	Tikslumas	Produkto veikimas pateikiant teisingus arba sutartus rezultatus.
	Atitikimas standartams	Produkto atitikimas įvairiems standartams: PĮ kūrimo standartams, įstatymams ir pan.
	Saugumas	Produkto galimybės uždrausti neautorizuotą priėjimą prie programos arba programos duomenų.
Patikimumas	Brandumas	Nesėkmingų programos veikimo atvejų dėl gedimų dažnis.
	Gedimų tolerancija	Programos galimybė palaikyti nustatytą funkcionavimo lygį atsiradus tam tikriems gedimams.
	Atkuriamumas	Programos gebėjimas atstatyti funkcionavimo lygį bei prarastus duomenis nesėkmingo programinės operacijos atlikimo atveju nustatytose laiko bei kaštų ribose.
Panaudojamumas	Suprantamumas	Vartojo pastangos reikalingos programos loginio konteksto atpažinimui.
	Išmokstamumas	Vartotojo pastangos reikalingos siekiant išmokti dirbti su programa.
	Valdymas	Vartotojo pastangos reikalingos programos operacijų atlikimui bei jų valdymui.
Efektyvumas	Laiko režimas	Programos atsako bei veikimo laiko našumas.
	Resursų režimas	Programos naudojamų resursų apimtis bei resursų panaudojimo našumas.
Palaikomumas	Analizuojamumas	Pastangų apimtis, reikalinga programų trūkumų arba defektų analizei, arba modifikuojamų programos dalių nustatymui.
	Keičiamumas	Pastangų apimtis, reikalinga modifikacijoms, klaidų pašalinimui arba perėjimui prie kitos funkcionavimo aplinkos.
	Stabilumas	Rizikos dydis susijęs su nenuspėjamu funkcionavimu po programos modifikacijų.
	Testuojamumas	Pastangų apimtis, reikalinga atliekant programinės įrangos pripažinimą tinkančia.
Portatyvumas	Prisitaikymas	Programos galimybės prisitaikyti prie skirtingų funkcionavimo aplinkų
	Įdiegimo galimybės	Pastangų apimtis, reikalinga diegiant programinę įrangą nustatytoje funkcionavimo aplinkoje.
	Atitikimas standartams	Programinės įrangos atitikimas portatyvumo standartams.

	Pakeičiamumas	Galimybė panaudoti programinę įrangą vietoje kitos programinės įrangos jos funkcionavimo aplinkoje.

1.15. Analizės išvados

- Atlikus egzistuojančių projektavimo ir internetinės sąsajos kūrimo sistemų tyrimą, pastebėta, kad jos netenkina vartotojų poreikių: esamos sistemos leidžia arba tik projektuoti, arba tik kurti interneto sąsają, nėra integruotų įrankių, leidžiančių sujungti šiuos etapus ir matyti UML diagramų pagalba suprojektuotos sąsajos vaizdą.
- Todėl nutarta sukurti metodą ir įrankį, kuris leistų interneto sąsajos projektuotojams grafiškai projektuoti sąsajos veikimą ir jungti veikimo modelį su kuriamos sąsajos vaizdu.
- Atlikus modeliavimo kalbų palyginimą, nustatyta, jog vartotojo sąsajos specifikavimui tikslinga pasirinkti unifikuotąją modeliavimo kalbą dėl jos populiarumo ir išraiškos galimybių.
- Vartotojo sąsajos specifikavimui pasirinktos objektų diagramos, kadangi jos geriausiai atspindi į vartotojų įvykius reaguojančios sistemos veikimo modelį.
- Duomenų saugojimui, perdavimui bei transformavimui pasirinkta XML kalba.
- Atlikus lyginamąją transformavimo kalbų analizę, nustatyta, kad modelių transformavimui į programos kodą geriausiai tinka XSL transformacijos. Jų pasirinkimą lėmė suderinamumas su XML kalba, standartų palaikymas ir paprastumas.
- Analizės rezultate sudarytas kuriamos sistemos prototipas, susiejantis sąsajos elgseną, pavaizduotą būsenų diagrama, su sąsajos vaizdais, kuriuos matys ir naudos sukurtos sistemos vartotojai.

2. GRAFINĖS VARTOTOJO SĄSAJOS GENERAVIMO SISTEMOS REIKALAVIMŲ SPECIFIKACIJA IR ANALIZĖ

2.1. Reikalavimų specifikacija

Reikalavimai programinei įrangai išdėsto, ką sistema turi daryti ir apibrėžia jos vykdymo ir realizavimo apribojimus. Toliau pateikiami sistemos nefunkciniai ir funkciniai reikalavimai.

2.2. Nefunkciniai reikalavimai ir apribojimai

Nefunkciniai reikalavimai apriboja kuriamą sistemą arba kūrimo procesą, t.y. apibrėžia sistemos savybes ir apribojimus kaip pvz. patikimumą, atsakymo laiką ir reikalavimus atminčiai. Apribojimai yra įvedimo/ išvedimo įrenginio galimybės ir pan. Nefunkciniai reikalavimai gali būti labiau svarbūs nei funkciniai reikalavimai. Jei jie nėra išpildomi, sistema yra nenaudinga.

2.2.1 Reikalavimai sistemos išvaizdai

Pagrindiniai reikalavimai sistemos grafinei išvaizdai pateikiami 24 lentelėje.

24 Lentelė.

Reikalavimas #:	13	Reikalavimo tipas:	10a	Įvykis/panaudojimo atvejis #:	/1-9
Aprašymas:	Intuityvi sąsaja.				
Pagrindimas:	Sistema naudosis patyrę kompiuterių vartotojai.				
Šaltinis:	Projektuotojas/programuotojas				
Tikimo kriterijus:	Patyrę kompiuterių vartotojai intuityviai elgiasi su sistema, numato kur ko ieškoti.				
Užsakovo tenkinimas:	4			Užsakovo netenkinimas:	4
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:	Nėra				
Istorija:	Užregistruotas 2006 kovo 10d.				

2.2.2 Reikalavimai panaudojamumui

Pagrindiniai reikalavimai sistemos panaudojamumui pateikiami 25 ir 26 lentelėse.

25 lentelė.

Reikalavimas #:	15	Reikalavimo tipas:	11a	Įvykis/panaudojimo atvejis #:	/1-5
Aprašymas:	Sistemoje turi būti patogus duomenų įvedimas				
Pagrindimas:	Projektuotojui neturi kilti sunkumų įvedant specifikaciją dėl programos sudėtingumo.				
Šaltinis:	Užsakovas				

Tikimo kriterijus:	Projektuotojai intuityviai supranta ir lengvai išmoksta įvesti duomenis į sistemą.		
Užsakovo tenkinimas:	3	Užsakovo netenkinimas:	4
Priklausomybės	Nėra	Konfliktai:	Nėra
Papildoma medžiaga:	Nėra		
Istorija:	Užregistruotas 2006 kovo 10d.		

26 lentelė.

Reikalavimas #:	16 Reikalavimo tipas: 11b Įvykis/panaudojimo atvejis #: /5		
Aprašymas:	Įvedama specifikacija gali turėti būsenų, perėjimų, atributų pavadinimus ir kitus žymėjimus įvairiomis nacionalinėmis kalbomis.		
Pagrindimas:	Būsenų diagramos specifikavimui gali būti naudojamos įvairios kalbos, nes gali pasitaikyti projektuotojų kalbančių kitomis kalbomis.		
Šaltinis:	Užsakovas		
Tikimo kriterijus:	Sistema veikia teisingai nepriklausomai nuo specifikacijose naudojamų kalbų.		
Užsakovo tenkinimas:	2	Užsakovo netenkinimas:	4
Priklausomybės	Nėra	Konfliktai:	Nėra
Papildoma medžiaga:	Nėra		
Istorija:	Užregistruotas 2006 kovo 10d.		

2.2.3 Reikalavimai vykdymo charakteristikoms

Pagrindiniai reikalavimai sistemos vykdymo charakteristikoms pateikiami 27 lentelėje

27 lentelė.

Reikalavimas #:	17 Reikalavimo tipas: 12a Įvykis/panaudojimo atvejis #: /1-11		
Aprašymas:	Kuriamam produktui greičio reikalavimai nėra taikomi, bet algoritmo programinė realizacija turi būti optimizuota, kad neverstų vartotojo pernelyg ilgai laukti darbo rezultatų.		
Pagrindimas:	Laukimas vartotojui kelia neigiamas emocijas sistemos atžvilgiu		
Šaltinis:	Užsakovas		
Tikimo kriterijus:	Transformacija iš specifikacijos į programos kodą gali būti vykdoma ne ilgiau kaip 20 sekundžių.		
Užsakovo	4	Užsakovo netenkinimas:	3

tenkinimas:			
Priklausomybės	Nėra	Konfliktai:	Nėra
Papildoma medžiaga:			
Istorija:	Užregistruotas 2006 kovo 10d.		

2.2.4 Reikalavimai sistemos priežiūrai

Pagrindiniai reikalavimai sistemos priežiūrai pateikiami 28 lentelėje

28 lentelė.

Reikalavimas #:	18 Reikalavimo tipas: 13a Įvykis/panaudojimo atvejis #: /1-11		
Aprašymas:	Kuriama sistema lengvai papildoma naujais algoritmais ir funkcijomis.		
Pagrindimas:	Modelių transformavimo ir validavimo algoritmai turi būti realizuoti kaip atskiras, nepriklausomas ir lengvai modifikuojamas sistemos modulis.		
Šaltinis:	Sistemos administratorius.		
Tikimo kriterijus:	Lengvas naujų algoritmų papildymas bei modifikavimas.		
Užsakovo tenkinimas:	3	Užsakovo netenkinimas:	2
Priklausomybės	Nėra	Konfliktai:	Nėra
Papildoma medžiaga:			
Istorija:	Užregistruotas 2006 kovo 10d.		

2.2.5 Reikalavimai saugumui

Pagrindiniai reikalavimai sistemos saugumui pateikiami 29 lentelėje

29 lentelė.

Reikalavimas #:	19 Reikalavimo tipas: 14a Įvykis/panaudojimo atvejis #: /1-11		
Aprašymas:	Išlaikomas vientisumas - sistemos duomenys vienareikšmiškai atitinka šaltinio perduotus (iš jo gautus) duomenis.		
Pagrindimas:	Užfiksuojama yra tai ką vartotojas įveda į sistemą.		
Šaltinis:	Projektuotojas/programuotojas		
Tikimo kriterijus:	Vartotojas gauna rezultate (XML kode), tai ką suvedė specifikacijoje (t.y. suvesti pradiniai duomenys atitinka galutinius duomenis), o sąsają mato tokią, kokia turėtų būti pagal jo atliktą projektavimą.		
Užsakovo tenkinimas:	4	Užsakovo netenkinimas:	5
Priklausomybės	Nėra	Konfliktai:	Nėra
Papildoma medžiaga:	Nėra		

Istorija:	Užregistruotas 2006 kovo 10d.

Informacijos sistemai keliami bendri apribojimai ir reikalavimai pateikti 30 lentelėje.

30 lentelė. Sistemos apribojimų ir nefunkcinių reikalavimų specifikacija

Kam skirtas reikalavimas	Reikalavimo tipas	Reikalavimas
Techninei įrangai	Sudėtis	Minimalus procesorius - Pentium III 800 MHz. Operatyvioji atmintis – 256 MB.
Programinei įrangai	Sudėtis	Operacinė sistema yra svarbi: Windows
	Mobilumas	Sistema turi būti perkeliama Windows operacinę sistemą. Rezultatai turi būti išvedami korektiškai, ir suprantamoje formoje.
Vartotojo sąsajai	Naudojimas	Sąsaja turi būti ypač lengvai suprantama, vartotojui neturėtų kilti papildomų sunkumų.

2.3. Funkciniai reikalavimai

Funkciniai reikalavimai aprašo sistemos funkcionalumą arba 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 sistema turi daryti, bet funkciniai sistemos reikalavimai turi detaliai aprašyti sistemos paslaugas.

Pagrindinių funkcinių reikalavimų specifikacija pateikiama 31-42 lentelėse.

31 lentelė.

Reikalavimas #:	1	Reikalavimo tipas:	9.1	Įvykis/panaudojimo atvejis #:	1/1
Aprašymas:	Objektų diagramos sukūrimas.				
Pagrindimas:	Projektuotojui turi būti suteikta galimybė sukurti objektų diagramą taip, kad jam būtų aišku kur yra pradinis langas (t.y. pirmas objektas/procesas), kuris seka iš jo, t.y. į kurį langą bus pereinama kai matysime internetinės sąsajos langus.				
Šaltinis:	Projektuotojas/programuotojas				
Tikimo kriterijus:	Nubraižyta diagrama turi atitikti UML standartą				
Užsakovo tenkinimas:	4			Užsakovo netenkinimas:	3
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:	Nėra				
Istorija:	Užregistruotas 2006 kovo 10d./perregistruotas 2007 sausio 3d.				

32 lentelė.

Reikalavimas #:	2	Reikalavimo tipas:	9.1	Įvykis/panaudojimo atvejis #:	1/2
Aprašymas:	Objektų braižymas.				
Pagrindimas:	Projektuotojui turi būti suteikta galimybė pasirinkti objekto brėžimų įrankį. Objektų diagramą gali sudaryti ir vienas objektas, programa ir tokiu atveju privalo veikti teisingai.				
Šaltinis:	Projektuotojas/programuotojas				
Tikimo kriterijus:	Turi būti nubraižytas bent vienas objektas				
Užsakovo tenkinimas:	5			Užsakovo netenkinimas:	5
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:	Nėra				
Istorija:	Užregistruotas 2006 kovo 10d./ perregistruotas 2007 sausio 3d.				

33 lentelė.

Reikalavimas #:	3	Reikalavimo tipas:	9.1	Įvykis/panaudojimo atvejis #:	1/3
Aprašymas:	Perėjimų braižymas.				
Pagrindimas:	Projektuotojui turi būti suteikta galimybė pasirinkti perėjimų brėžimų įrankį. Objektų diagrama turi turėti perėjimus tarp norimų objektų, kurie vartotojo nuomone, turi būti susieti.				
Šaltinis:	Projektuotojas/programuotojas				
Tikimo kriterijus:	Nubraižyti perėjimai tarp norimų objektų				
Užsakovo tenkinimas:	4			Užsakovo netenkinimas:	4
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:	Nėra				
Istorija:	Užregistruotas 2006 kovo 10d./ perregistruotas 2007 sausio 3d.				

34 lentelė.

Reikalavimas #:	4	Reikalavimo tipas:	9.1	Įvykis/panaudojimo atvejis #:	1/4
Aprašymas:	Specifikacijos įvedimas.				
Pagrindimas:	Projektuotojui turi būti suteikta galimybė įvesti į sistemą objektų specifikaciją klaviatūros pagalba, turi būti galimas pasirinkimas.				
Šaltinis:	Projektuotojas/programuotojas				
Tikimo kriterijus:	Įvestas reikalavimas turi atitikti sistemos naudojamų schemų sintaksę bei semantiką. Turi būti ar atributas yra pateikiamas duomenų bazei ar gaunamas, ar				

	abu variantai.		
Užsakovo tenkinimas:	5	Užsakovo netenkinimas:	4
Priklausomybės	Nėra	Konfliktai:	Nėra
Papildoma medžiaga:	Nėra		
Istorija:	Užregistruotas 2006 kovo 10d./ perregistruotas 2007 sausio 3d.		

35 lentelė.

Reikalavimas #:	5	Reikalavimo tipas:	9.1	Įvykis/panaudojimo atvejis #:	1/5
Aprašymas:	XSL transformacijų parinkimas.				
Pagrindimas:	Turi būti sukurtas XSL failas.				
Šaltinis:	Projektuotojas/programuotojas				
Tikimo kriterijus:	Sėkmingai parinktas XSL failas ir įtrauktas į sistemą.				
Užsakovo tenkinimas:	4			Užsakovo netenkinimas:	5
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:	Nėra				
Istorija:	Užregistruotas 2006 kovo 10d./ perregistruotas 2007 sausio 3d.				

36 lentelė

Reikalavimas #:	6	Reikalavimo tipas:	9.1	Įvykis/panaudojimo atvejis #:	1/6
Aprašymas:	Duomenų saugojimas XML formatu.				
Pagrindimas:	Duomenys turi atitikti XML sintaksę.				
Šaltinis:	Projektuotojas/programuotojas				
Tikimo kriterijus:	Sėkmingai išsaugoti duomenys.				
Užsakovo tenkinimas:	4			Užsakovo netenkinimas:	3
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:	Nėra				
Istorija:	Užregistruotas 2006 kovo 10d./ 10d./ perregistruotas 2007 sausio 3d.				

37 lentelė.

Reikalavimas #:	7	Reikalavimo tipas:	9.1	Įvykis/panaudojimo atvejis #:	1/7
Aprašymas:	Įvesto į sistemą projektavimo modelio transformavimas į kodą.				
Pagrindimas:	Sistema turi atlikti specifikacijos transformavimą į programos kodą. Tai viena iš pagrindinių kuriamos sistemos funkcijų.				
Šaltinis:	Projektuotojas/programuotojas				
Tikimo kriterijus:	Specifikacija turi būti teisingai transformuojama į programos kodą.				

Užsakovo tenkinimas:	5	Užsakovo netenkinimas:	5
Priklausomybės	Nėra	Konfliktai:	Nėra
Papildoma medžiaga:	Nėra		
Istorija:	Užregistruotas 2006 kovo 10d./ perregistruotas 2007 sausio 3d.		

38 lentelė.

Reikalavimas #:	8	Reikalavimo tipas:	9.1	Įvykis/panaudojimo atvejis #:	1/7
Aprašymas:	Generuojamas kodas.				
Pagrindimas:	Norint gauti programos kodą, specifikacija (kuri braižoma UML formate, o saugoma XML formate) ir transformacijos atitinkamai turi atitikti XSL standartą.				
Šaltinis:	Projektuotojas/programuotojas				
Tikimo kriterijus:	Programos kodas bus sugeneruotas				
Užsakovo tenkinimas:	4			Užsakovo netenkinimas:	3
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:	Nėra				
Istorija:	Užregistruotas 2006 kovo 10d./ perregistruotas 2007 sausio 3d.				

39 lentelė.

Reikalavimas #:	9	Reikalavimo tipas:	9.1	Įvykis/panaudojimo atvejis #:	1/8
Aprašymas:	Sistemos sugeneruotas XML kodas turi būti atvaizduojamas tekstiniu pavidalu.				
Pagrindimas:	Projektuotojas turi gauti transformacijos metų gautą programos kodą lengvai skaitomu bei suprantamu teksto pavidalu.				
Šaltinis:	Projektuotojas/programuotojas				
Tikimo kriterijus:	Pateikiamas pilnas programos kodas, kurį projektuotojas toliau gali pritaikyti savo reikmėm ir iš jo gauti norimą rezultatą.				
Užsakovo tenkinimas:	5			Užsakovo netenkinimas:	5
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:	Nėra				
Istorija:	Užregistruotas 2006 kovo 10d.				

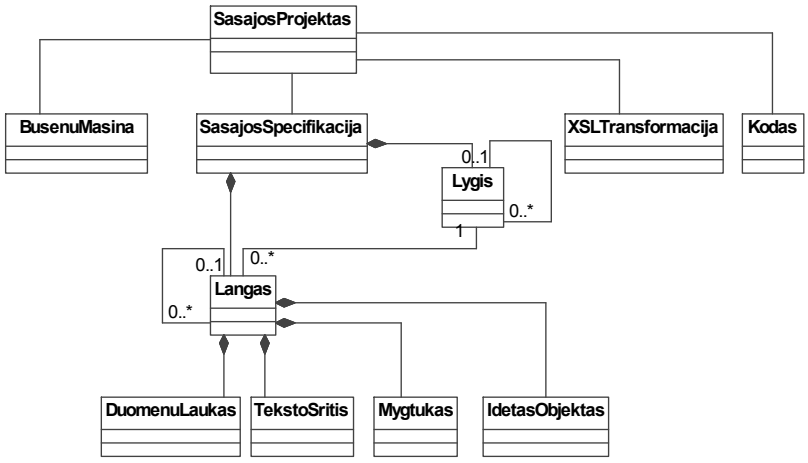
40 lentelė.

Reikalavimas #:	10	Reikalavimo tipas:	9.1	Įvykis/panaudojimo atvejis #:	1/9

Aprašymas:	Peržiūrėti sąsajos langą.		
Pagrindimas:	Sistemoje turi būti galimybė peržiūrėti vartotojo sąsajos langą, kuri suprojektavo pats vartotojas, taip pat turi matytis funkcionalumas, t.y. perėjimai tarp langų jei objektai buvo sujungti ir atliktas perėjimo specifikavimas		
Šaltinis:	Projektuotojas/programuotojas		
Tikimo kriterijus:	Sąsajos langas toks, kokį suprojektavo ir specifikavo vartotojas		
Užsakovo tenkinimas:	5	Užsakovo netenkinimas:	5
Priklausomybės	Nėra	Konfliktai:	Nėra
Papildoma medžiaga:	Nėra		
Istorija:	Užregistruotas 2006 kovo 10d./ perregistruotas 2007 sausio 3d.		

2.4. Reikalavimai duomenims

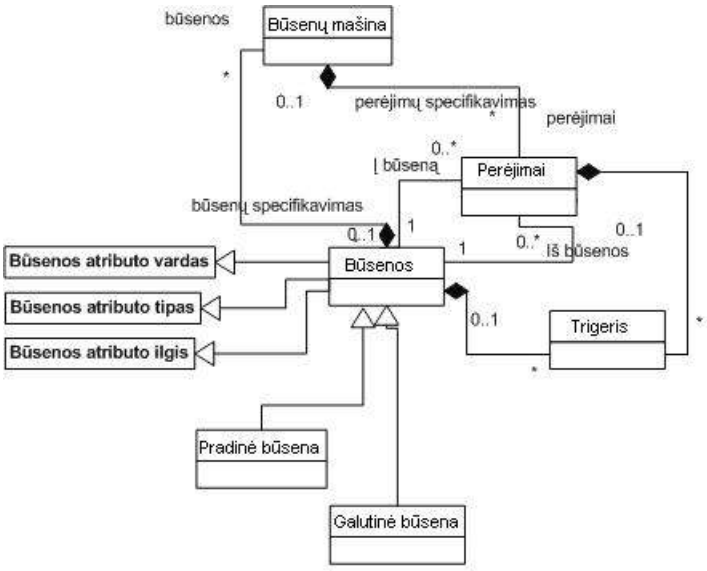
Sudarant sistemos modelį naudojamos objektų diagramos. Kiekvienam objektui, kuris atitinka sistemos atskirą dalį, tarkime grafinės vartotojo sąsajos lango vaizdą, sudaroma specifikacija, aprašomi perėjimai tarp objektų. Kitaip tariant specifikuojama sistemos sąsaja. Taigi sudarant sistemos modelį, kiekvienam panaudojimo atvejui yra sukuriama vienas ar daugiau objektų, perėjimai tarp sistemos dalių vaizduojami perėjimais iš vieno objekto į kitą. Perėjimus tarp objektų iššaukia įvykiai – vartotojo sąveika su sistemos sąsaja (dialogas). Sąsajos specifikacijos duomenų modelis pavaizduotas 8 paveiksle.



8 pav. Duomenų modelis

Sudaromo sistemos modelio sudėtis grindžiama objektų ir perėjimų tarp jų sistemos semantika. Sistemos objektas suprantamas kaip sistemos būsena; objektai, kurių būsenos išsaugomos informacinėje bazėje, traktuojami kaip esybės; objektus, kurie keičia esybių būsenas,

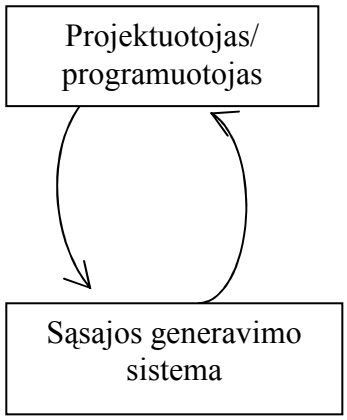
atstovauja sistemos sąsajos. Perėjimus tarp objektų iššaukia išorinių aktorių sąveikos su sistemos sąsajomis, kurios gali vykdyti perėjimus sąveikaudamos su kitomis (tame tarpe išorinėmis) sąsajomis, kurios turi būti aprašytos specifikacijoje.



9 pav. Vartotojo sąsajos specifikacijos metamodelis

Sukurta sistema atlieka specifikacijos transformavimą į programos kodą. Veiklos įvykių sąrašas, kuris apima visus veiklos įvykius, už kuriuos yra atsakinga nagrinėjama veikla, pateiktas 20 lentelėje. Veiklos įvykiai - tai vartotojo išskiriami veiksmai, atliekami veiklos metu. Reakcija (atsakymas) į kiekvieną įvykį atvaizduoja veiklos dalį, įeinančią į bendras veiklą sudarančias funkcijas. Nagrinėjamai veiklos sričiai apibrėžti naudojama konteksto diagrama (10 pav.). Šią sritį tenka ištirti, norint teisingai sukurti sistemą.

1. Objektų diagrama
2. Duomenų saugojimas XML formate
3. Sistemos specifikacija
4. Transformacija
5. XSLT pasirinkimas
6. Kodo generavimo užklausa
7. Kodo peržiūros užklausa
8. Sąsajos peržiūros užklausa



1. Įvesta objektų diagrama
2. Duomenų XML formate išsaugojimo patvirtinimas
3. Sistemos specifikacijos įvedimo patvirtinimas
4. Transformacijos įvedimo patvirtinimas
5. XSLT įvedimo/pasirinkimo patvirtinimas
6. Kodo generavimo rezultatas
7. Kodas
8. Sąsaja

10 pav. Veiklos konteksto diagrama

Veiklos kontekstas apima plačiau, nei kuriamos sistemos atliekamos funkcijos. Kol nesuprasime darbo (veiklos), kuriam turi talkinti sistema, tol nesukursime tinkamos sistemos,

kurį “įsirašys” į aplinką. Veiklos kontekstas apibrėžia dominančią veiklą ir jos naudojamus bei formuojamus informacijos srautus. Veiklos “atsakomybė” prasideda kai informacijos srautas įeina į sistemą ir baigiasi, kai rezultatinis srautas išeina iš sistemos.

Veiklos įvykių sąrašas, kuris apima visus veiklos įvykius, už kuriuos yra atsakinga nagrinėjama veikla, pateiktas 41 lentelėje. Veiklos įvykiai - tai vartotojo išskiriami veiksmai, atliekami veiklos metu. Reakcija (atsakymas) į kiekvieną įvykį atvaizduoja veiklos dalį, įeinančią į bendras veiklą sudarančias funkcijas.

41 lentelė. Veiklos padalinimas

Eil. Nr.	Įvykio pavadinimas	Įeinantys/išeinantys informacijos srautai
1.	XML duomenų importas	XML duomenys *(įeina)*
2.	Objektų diagrama (Projektuotojas nubraižo objektų diagramą, aprašo kiekvieną objektą, aprašo perėjimus tarp objektų)	Objektų diagrama, kuri fiziškai saugoma kaip xml duomenys *(įeina)*
3.	Sistemos specifikacija (Objektų atributų įvedimas)	Sistemos specifikacija pateikta objektų diagrama, specifikacijos įvedimo patvirtinimas *(išeina)*
4.	Transformacija. (Projektuotojas įveda XSL transformacijas)	XSL transformacijos *(įeina)*
5.	XSLT pasirinkimas (Projektuotojas pasirenka XSL transformacijas jei jas turi išsaugotas faile)	XSL transformacijos *(įeina)*
6.	Kodo generavimo užklausa	Kodo generavimo rezultatas: programos kodas *(išeina)*
7.	Kodo peržiūros užklausa	Kodas *(išeina)*
8.	Sistema transformuoja specifikaciją į programos kodą.	Įvesta specifikacija *(įeina)* Gautas programos kodas *(išeina)*
9.	Grafinės vartotojo sąsajos peržiūra	Sąsajos peržiūros užklausa *(įeina)* Sąsaja *(išeina)*

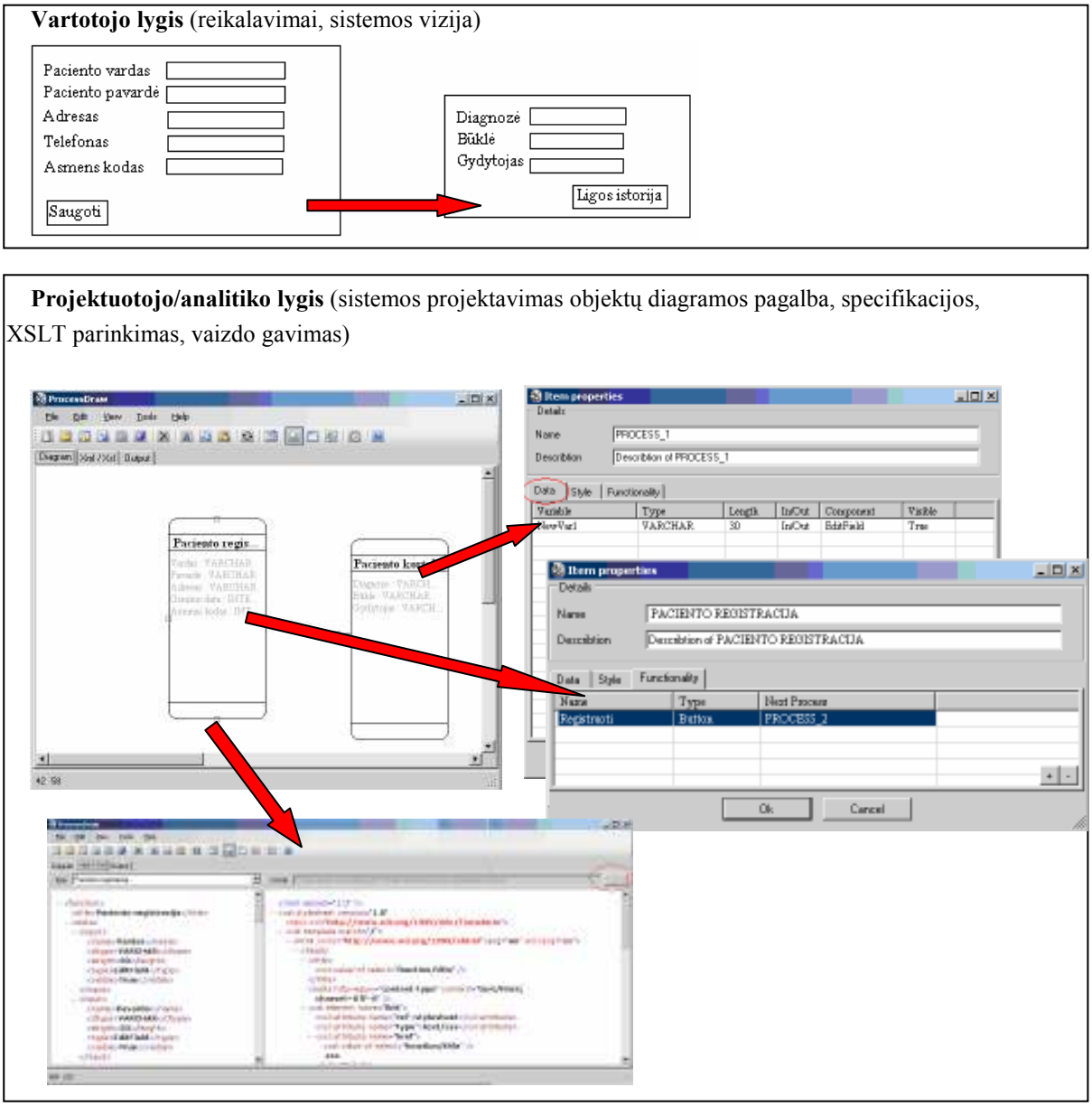
3. GRAFINĖS VARTOTOJO SĄSAJOS GENERAVIMO IŠ DIALOGO SPECIFIKACIJOS SISTEMOS MODELIS

3.1. Sistemos metodo ir modelio pagrindimas, esmės išdėstymas

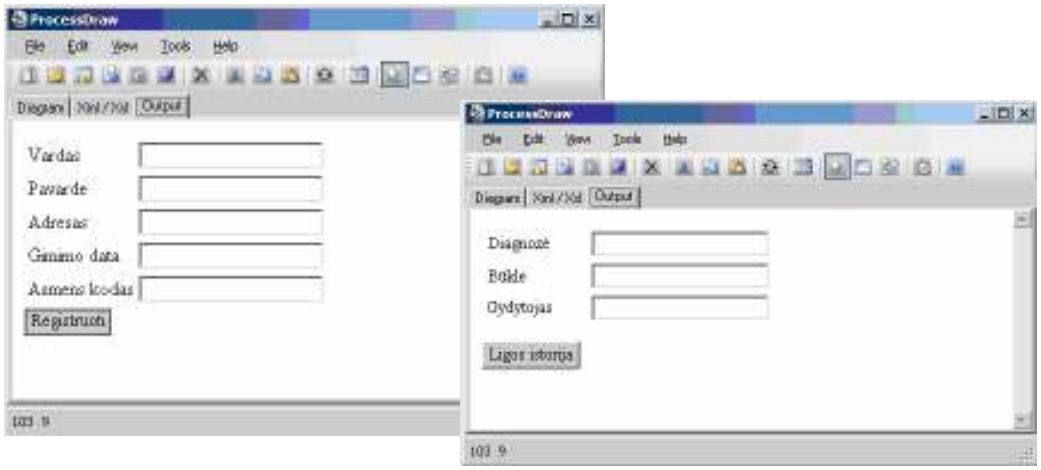
Programa (PĮ) vartotojo sąsajai projektuoti apima tris pagrindines dalis:

- sistemos funkcionalumo sudarymą t.y. modelio sukūrimą,
- atributų specifikavimą tam tikrai veiklai, t.y. modelio dalių specifikavimą,
- modelio transformavimą į programos kodą.

Programa leidžia specifikuoti būsimą objektų išvaizdą ir numatyti kaip objektai sąveikaus tarpusavyje ar kokias operacijas su jais bus galima atlikti. Toliau pateiktame paveikslėlyje, pagal konkretų pavyzdį, pateiktas 3 lygių modelis (11 pav.).

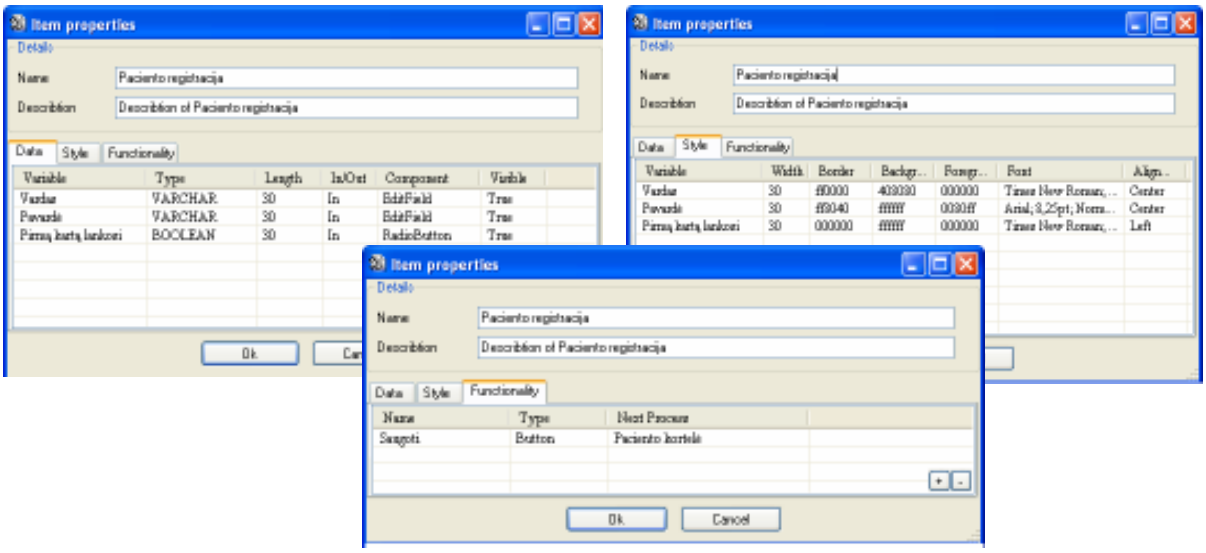


Vartotojo ir projektuotojo/analitiko lygis (vaizdas, kuris aiškus ir sistemos vartotojui, ir projektuotojui/analitikui)



11 pav. Iliustruotas trijų lygių modelis

Nubraižius objektų diagramą, būtina įvesti atributus ir juos specifikuoti (12 pav.). Atributai šiuo atveju, tai laukai, kurie bus matomi vartotojo sąsajos lange.



13 pav. Atributų specifikavimo langai

Kai pasirenkamas atributo duomenų koregavimas (Data) (12 pav. kairėje) tuomet galima keisti kintamojo/atributo pavadinimą (Variable), duomenų tipą (Type), atributo ilgį (Length), priskirti ar jis bus įeinantis ar išeinantis (In/Out), taip pat vaizdavimo tipą (Component) ir ar jis bus matomas lange ar ne (Visible). Kai pasirenkamas atributo stiliaus koregavimas (Style) (12 pav. dešinėje) lange matomas atributo pavadinimas, laukelio plotis, šrifto spalva, objekto (pvz. internetinio puslapio) fono spalva, šrifto stilius, išlygiavimas. DU kartus paspaudus ant laukų

Boeder, Background, Foreground bus atidaromi dialogo laukai naujame lange ir galimi pasirinkimai.

Funkciniai atributai, tai atributai, kurie sieja objektus (pvz. jei tarsime, kad vienas objektas yra viena internetinės sąsajos forma, tai funkcinis atributas bus mygtukas (mygtukas - funkcinio atributo tipas), kuris leis pereiti iš vienos formos į kitą.) (12 pav. centre) Norint specifikuoti funkcinį atributą, reikia įvesti jo pavadinimą, nurodyti, kokius du objektus jis sieja.

Taigi programoje galima specifikuoti būsimą objektų išvaizdą, dydį, vietą lange ir numatyti, kaip objektai sąveikaus tarpusavyje ar kokias operacijas su jais bus galima atlikti.

3.2. Sistemos architektūra – statinės struktūros modelis

Architektūros specifikacija yra parengta, naudojant Rational Unified Process (RUP) projektavimo metodiką, ir CASE priemonę „MS Visio Professional 2002“, kuri leido greičiau ir efektyviau sudaryti bei analizuoti reikiamas UML diagramas. Sistemos architektūrai pavaizduoti yra naudojami penki vaizdai: panaudojimo atvejų, loginis, procesų, išdėstymo ir duomenų. Visi jie ir juos sudarantys modeliavimo elementai yra pateikti 42 lentelėje.

42 lentelė. Vaizdai ir jų elementai

Vaizdas	Sudarantys elementai
Panaudojimo atvejų	sudaroma panaudojimo atvejų diagrama bei pateikiamos pagrindinių panaudojimo atvejų specifikacijos.
Loginis	pateikiama visos dalykinės srities klasių diagrama, sistemos išskaidymas į paketus.
Procesų	pateikiamos svarbiausių klasių objektų sekų, būsenų, bendradarbiavimo, veiklos diagramos.
Išdėstymo	Išdėstymo diagramos. Pateikiami aparatūriniai komponentai, kurie sudarys kuriamą PĮ.
Duomenų	Duomenų saugojimo modelis

Architekturinius apribojimus lemia pasirinktos sistemos kūrimo priemones: C# objektinio programavimo kalba, XML, XSLT.

Architektūros tikslai:

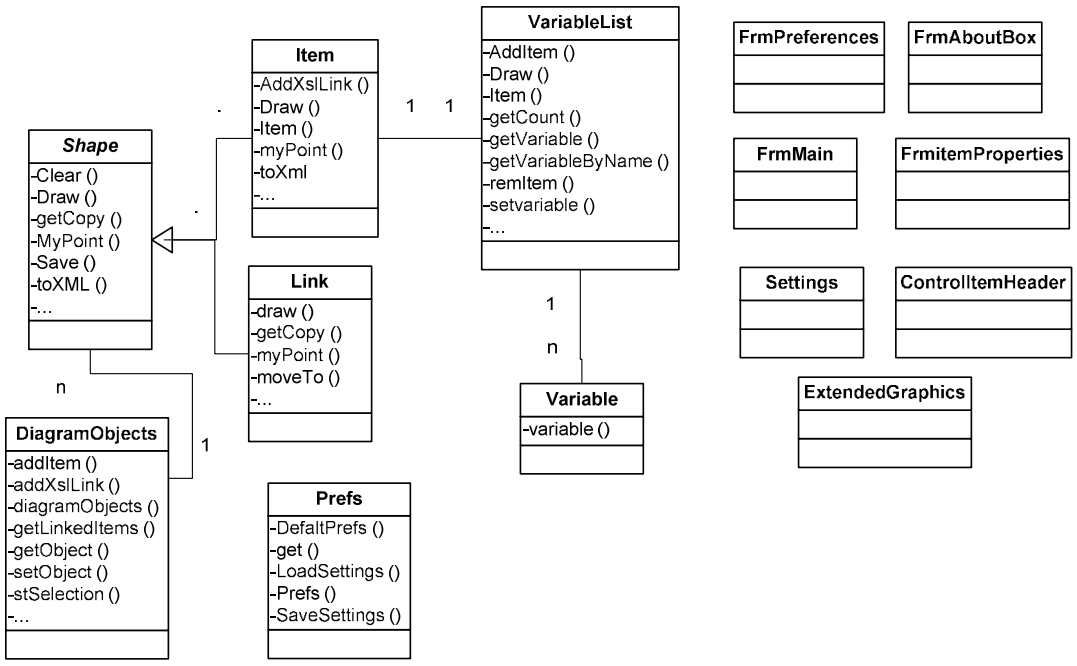
Projektuotojui turi būti suteikta galimybė sukurti objektų diagramą taip, kad jam būtų aišku kur yra pradinis, galinis objektai, aiškūs perėjimai tarp objektų; turi būti suteikta galimybė pasirinkti objektų brėžimo ir perėjimų tarp jų įrankį; transformacijos turi būti aiškiai parenkamos. Kai apjungiame XML su XSLT gautas kodas turi būti lengvai redaguojamas. Turi būti naudojami unicode (UTF-8) simbolių kodavimas. Galimybė išgauti bet kurios programavimo kalbos kodą. Galimybė įtraukti naujas transformacijas. Sukurtų komponentų pakartotinis panaudojimas kitose panašaus pobūdžio sistemose.

Architektūros apribojimai:

- Sistema yra kuriama, panaudojant C# programavimo kalbą;
- Pagrindinis projektavimo įrankis – „MS Visio Professional 2003“.

3.2.1 Klasių modelis

Pirmiausia reiktų apžvelgti sistemos klasių modelį, kuris orientuoja specifikaciją objektiniu aspektu. 14 paveiksle pavaizduotas sistemos klasių modelis ir žemiau aprašytos klasės.



14 pav. Vartotojo sąsajos generavimo klasių modelis

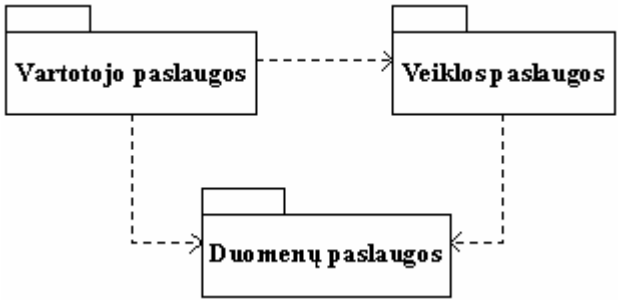
Shape - abstrakti tėvinė klasė, iš jos paveldi *Link* ir *Item* klasės. *Link* – klasė, sauganti informaciją apie ryšius tarp diagramos objektų. *Item* – klasė, kurioje saugoma informacija apie diagramos objektus. *Variable* – klasė, skirta saugoti informaciją apie diagramos objekto kintamuosius. *VariableList* - klasė turinti savyje objekto kintamųjų sąrašą, kitaip tariant, *Variable* yra vieno kintamojo informacija, o *VariableList* yra visų kintamųjų sąrašas. *DiagramObjects* - saugo visų diagramos objektų sąrašą ir atlieka veiksmus su jais. Prefs – klasė, skirta saugoti vartotojo nustatymams. Kitos klasės, prasidedančios FRM (angl. frames), saugo informaciją apie sistemos langus, kurie skirti vienos ar kitos informacijos suvedimui. *FrmItemProperties* – lango klasė, skirta suvesti informaciją apie Objekto kintamuosius, ji užpildo klasę *VariableList* klasės *Variable* objektais. *FrmMain* – pagrindinio lango klasė, skirta

piešti diagramai ir visų veiksmų atlikimui. *FrmPreferences* – atvaizduoja, suveda ir duoda redaguoti informaciją apie vartotojo nustatymus. *FrmAboutBox* - parodo „Apie“ dialogo langą, *ExtendedGraphics* – skirta piešti objekto dėžutei, esančios funkcijos leidžia nupiešti objektą su suapvalintais kraštais, keisti objekto vietą.

3.3. Loginė visos sistemos architektūra

Šiame skyriuje pateikiama sistemos loginė architektūra, jos išskaidymas į paketus ir kiekvieno paketo klasių diagramos.

Sistemai pasirinktas tipinis trijų lygių architektūros modelis. Pagal šį modelį, kuriamos programinės įrangos pagrindinį katalogą sudaro vartotojo ir veiklos paslaugos, sugrupuotos į atitinkamus paketus, pateiktus 15 paveiksle ir aprašus 43-45 lentelėse.



15 pav. Sistemos architektūros modelis

43 lentelė. Paketo „Vartotojo paslaugos“ specifikacija

Komponentas	**Vartotojo paslaugos**
Klasifikacija	Paketas
Apibrėžimas	Vartotojo paslaugų paketas, kurį sudaro formos (ribinės klasės), kurių pagalba sistemos naudotojai bendrauja su sistema. Šiame sluoksnyje yra pagrindinė forma, diagramų brėžimo sąsaja, transformacijų sąsaja.
Atsakomybės	Šis paketas naudojamas sąveikai tarp vartotojo ir sistemos teikiamo funkcionalumo. Duomenys vartotojui turi būti pateikiami suprantama forma. Vartotojo sąsajos sudarymui bus naudojamos C# kalba parašytos formos.
Apribojimai	Nėra.
Struktūra	Šio komponento subkomponentai yra formos, atitinkančios posistemio pagrindinį meniu ir submeniu punktus.
Sąveikavimas	Vartotojo sąsajos paketas naudoja „Duomenų paslaugų“ bei „Veiklos paslaugų“ paketus.

44 lentelė. Paketo „Veiklos paslaugos“ specifikacija

Komponentas	**Veiklos paslaugos**
Klasifikacija	Paketas
Apibrėžimas	Veiklos paslaugų paketą sudaro valdančios klasės, kuriomis

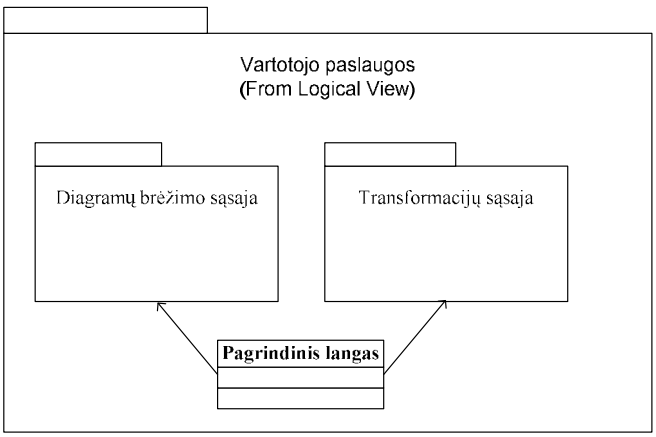
	įgyvendinamos veiklos taisyklės.
Atsakomybės	Šiame pakete esančios valdančiosios klasės atlieka veiksmus, numatytus panaudojimo atvejuose. Vartotojas per vartotojo sąsają, naudodamasis šio sluoksnio funkcionalumu, gali atlikti įvairias operacijas.
Apribojimai	Nėra.
Struktūra	Šio komponento subkomponentai yra valdymo klasės, realizuojančios duomenų bei operacijų valdymą.
Sąveikavimas	Paketas naudojasi „Duomenų paslaugų“ paketu ir yra naudojamas „Vartotojo paslaugų“ paketo.

45 lentelė. Paketo „Duomenų paslaugos“ specifikacija

Komponentas	**Duomenų paslaugos**
Klasifikacija	Paketas
Apibrėžimas	Duomenų paslaugų paketas – duomenų struktūrų ir prieigos klasės.
Atsakomybės	Šis paketas skirtas duomenų bei rezultatų saugojimui ir kaupimui.
Apribojimai	Nėra
Struktūra	Šio komponento subkomponentai yra duomenų struktūrų klasės.
Sąveikavimas	Yra naudojamas kitų paketų („Vartotojo paslaugų“ paketo ir „Veiklos paslaugų“ paketo)

3.3.1 Paketas „Vartotojo paslaugos“

Paketas „Vartotojo paslaugos“ yra skirtas atlikti tarpininko funkcijas tarp sistemos ir vartotojo. Jame yra sukurtos visos klasės, reikalingos pilnam vartotojo sąsajos funkcionalumui užtikrinti. Paketas „Vartotojo paslaugos“, kuris pateiktas 16 paveiksle, yra sudarytas iš dviejų paketų, kurie savyje turi aukščiausio lygio klases, kurių kiekviena yra skirta sistemos ryšiui su vartotoju užtikrinti.

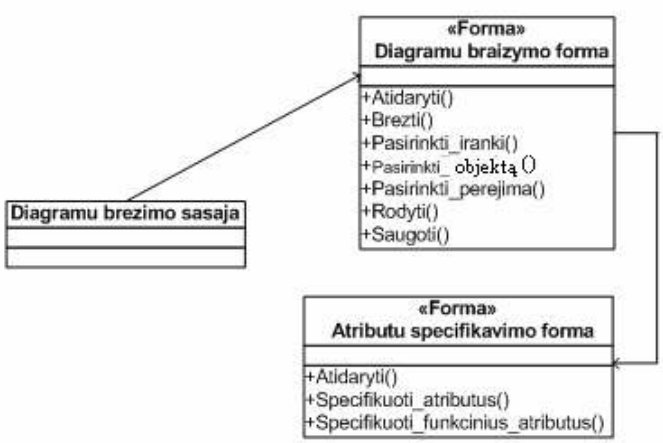


16 pav. Paketas „Vartotojo paslaugos“

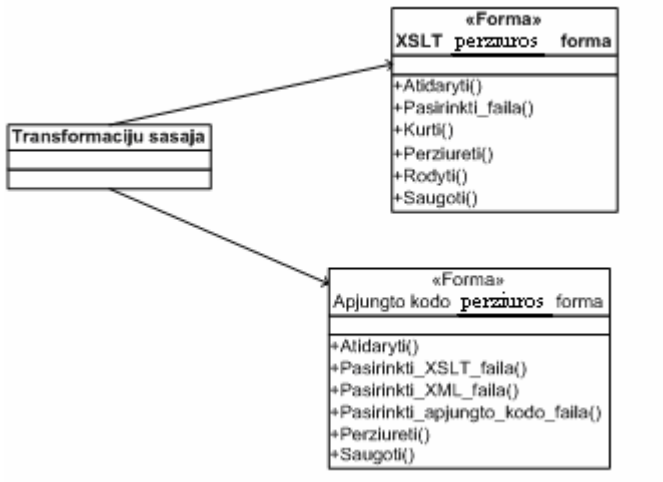
17 paveiksle pavaizduotas vartotojo paslaugų paketas, kurį sudaro pagrindinis langas, tai būtų galima pavadinti vartotojo sąsaja ir su ja susiję dar du paketai, tai diagramų brėžimo sąsajos paketas ir transformacijų sąsajos paketas. 17 paveiksle pavaizduotos su vartotojo sąsaja arba pagrindiniu langu susiję klasės, kitaip tariant detalizuoti prieš tai paminėti du paketai. Detalų klasių aprašymą galima rasti kituose šio skyriaus poskyriuose.

Pagrindiniame lange yra meniu iš kurio galima pasirinkti norimą komandą (pavyzdžiui, tai gali būti failo pasirinkimas, vienos iš formų lango atidarymas ir pan.).

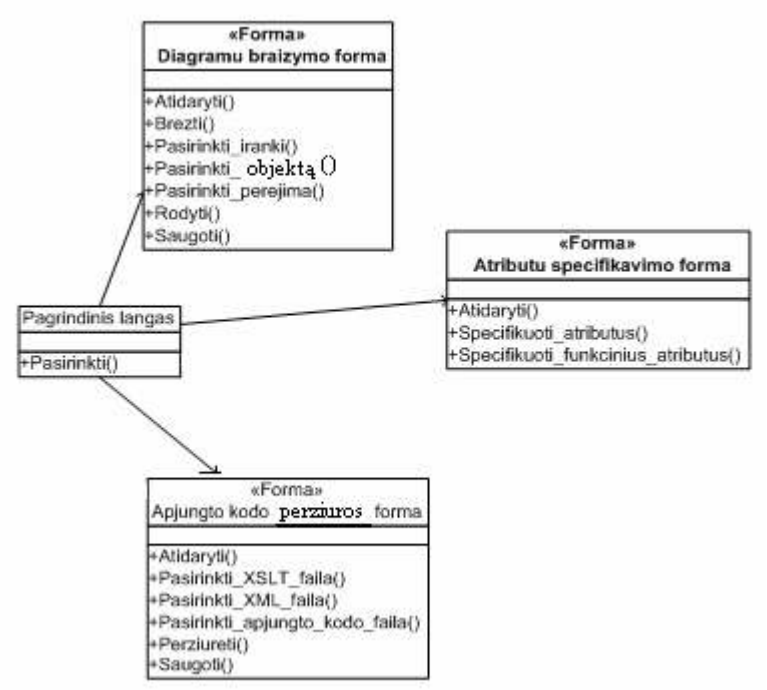
Kiekviena klasė iš vartotojo paslaugų paketo yra išskaidoma į smulkesnes klases. Šis skirstymas pavaizduotas 17-19 paveiksluose.



17 pav. Diagramų brėžimo sąsajos klasės



18 pav. Transformacijų sąsajos klasės



19 pav. Vartotojo sąsajos klasės

Trumpas kiekvienos klasės aprašymas:

Diagramų braižymo formos klasė suteikia galimybę bet kuriam vartotojui projektuoti norimą programinės įrangos sąsają braižant diagramas. Naudojamos funkcijos – formos lango atidarymas, diagramos brėžimas iš pradžių pasirinkus tinkamą įrankį (įrankį pradinei, tarpinei, galinei būsenoms bei perėjimams tarp jų atvaizduoti), tada nubrėžtos arba turimos būsenos pasirinkimas norint ją specifikuoti, perėjimo tarp būsenų pasirinkimas norint specifikuoti funkcinius atributus, diagramos rodymas bei saugojimas.

Atributų specifikavimo formos klasė suteikia galimybę naudotis sistema atliekant tokius veiksmus kaip: nubrėžtos būsenų diagramos būsenų ir perėjimų tarp jų specifikavimą, atributų aprašymą.

Apjungto kodo redagavimo formos klasė suteikia galimybę redaguoti apjungtą XML kodo failą su XSLT kodo failu.

3.3.2 Paketas „Veiklos paslaugos“

Paketas „Veiklos paslaugos“ yra skirtas atlikti tarpininko funkcijas tarp vartotojo sąsajos ir duomenų sąsajos. Jame yra sukurtos visos klasės, reikalingos pilnam sistemos funkcionalumui užtikrinti. Paketas „Veiklos paslaugos“ yra sudarytas iš dviejų aukščiausio lygio klasių, kurių kiekviena yra skirta sistemos ryšio su vartotoju užtikrinimui.

3.3.3 Paketas „Duomenų paslaugos“

Paketas „Duomenų paslaugos“ leidžia atlikti duomenų saugyklos funkcijas naudojant XML formato rinkmenas.

3.4. Sistemos elgsenos modelis (sekų, būsenų, veiklos diagramos)

Šiame skyriuje pateikiamos sąveikos, būsenų ir veiklos diagramos. Sąveikai atvaizduoti pagrindu pasirinktos sekų diagramos, tačiau kai kuriuose panaudojimo atvejuose parodytos ir bendradarbiavimo diagramos.

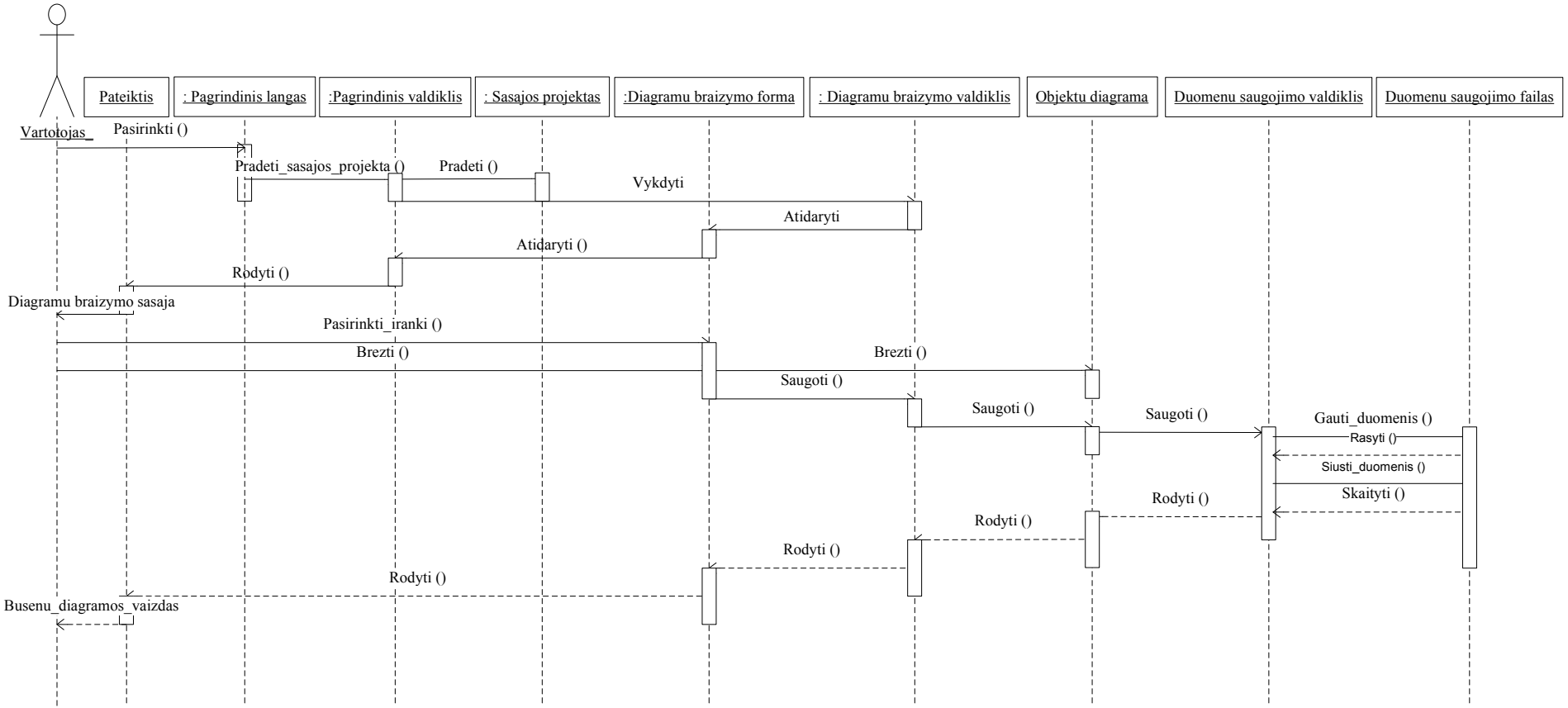
3.4.1 Panaudojimo atvejų sekų diagramos

Panaudojimo atvejo „*Sukurti objektų diagramą*“ sekų diagrama pavaizduota 20 paveiksle.

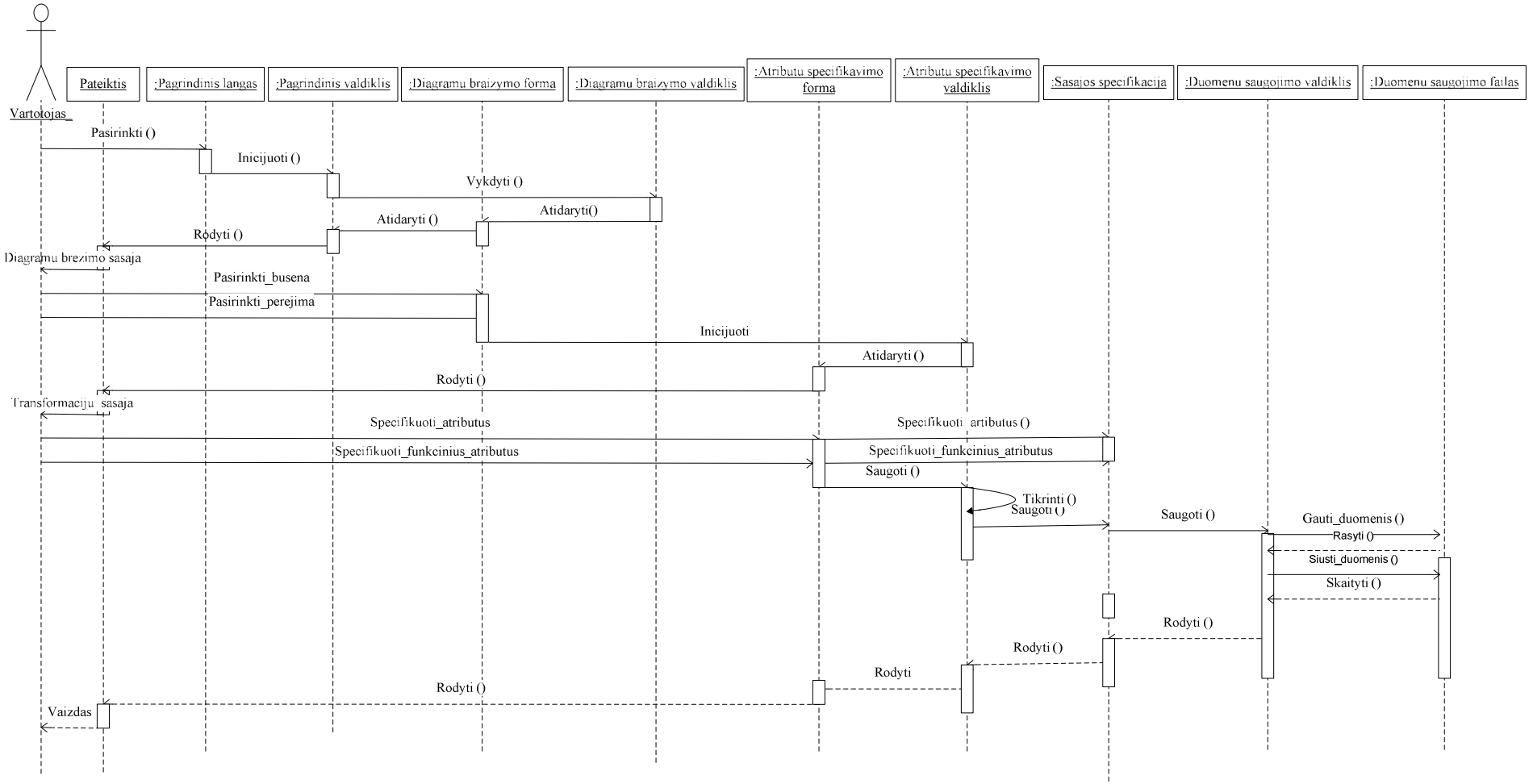
Panaudojimo atvejo „*Sudaryti sistemos specifikaciją*“ sekų diagrama pavaizduota 21 paveiksle.

Panaudojimo atvejo „*Parinkti XSL transformacijas*“ sekų diagrama pavaizduota 22 paveiksle.

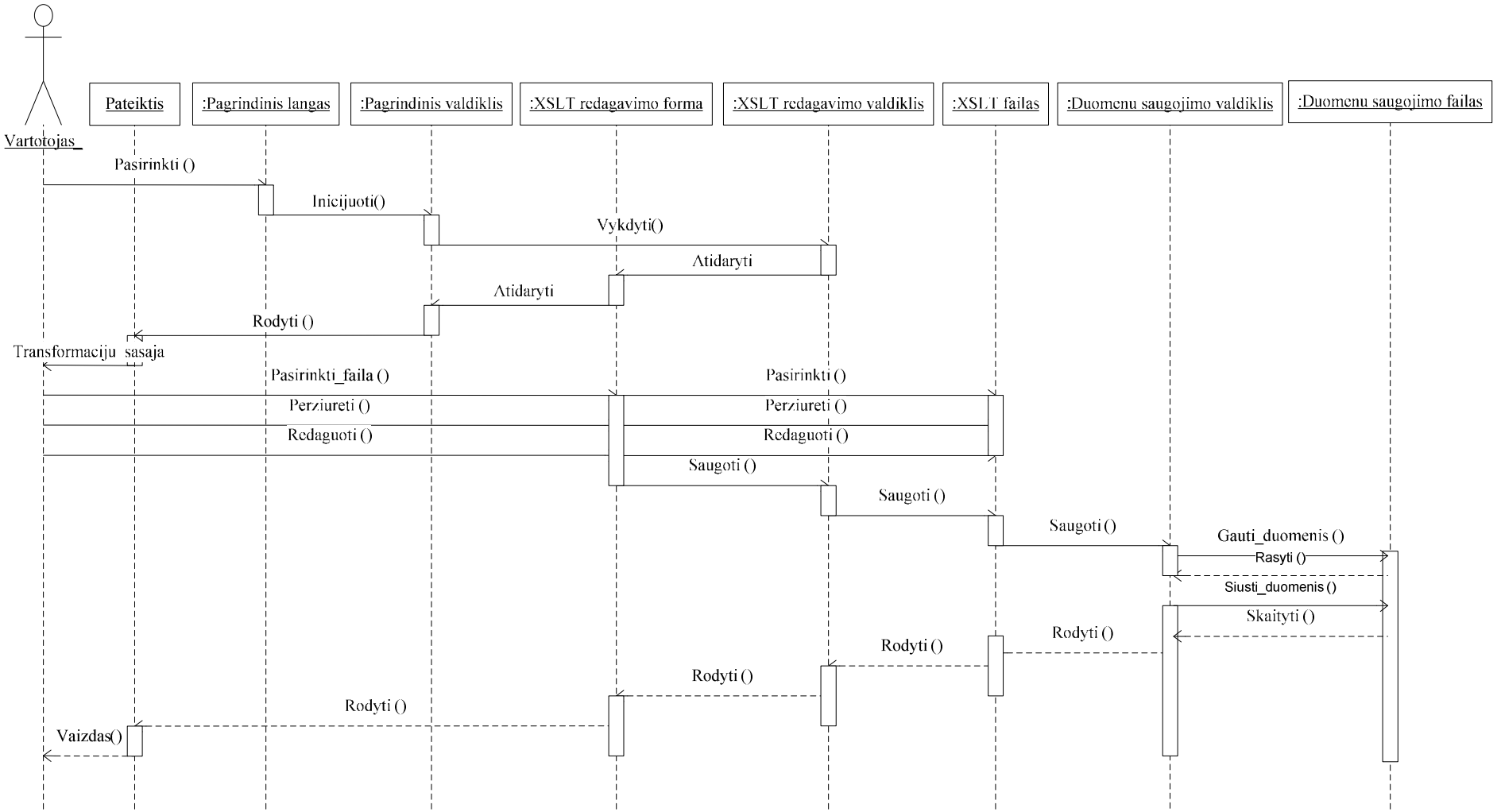
Panaudojimo atvejo „*Peržiūrėti kodą*“ sekų diagrama pavaizduota 23 paveiksle.



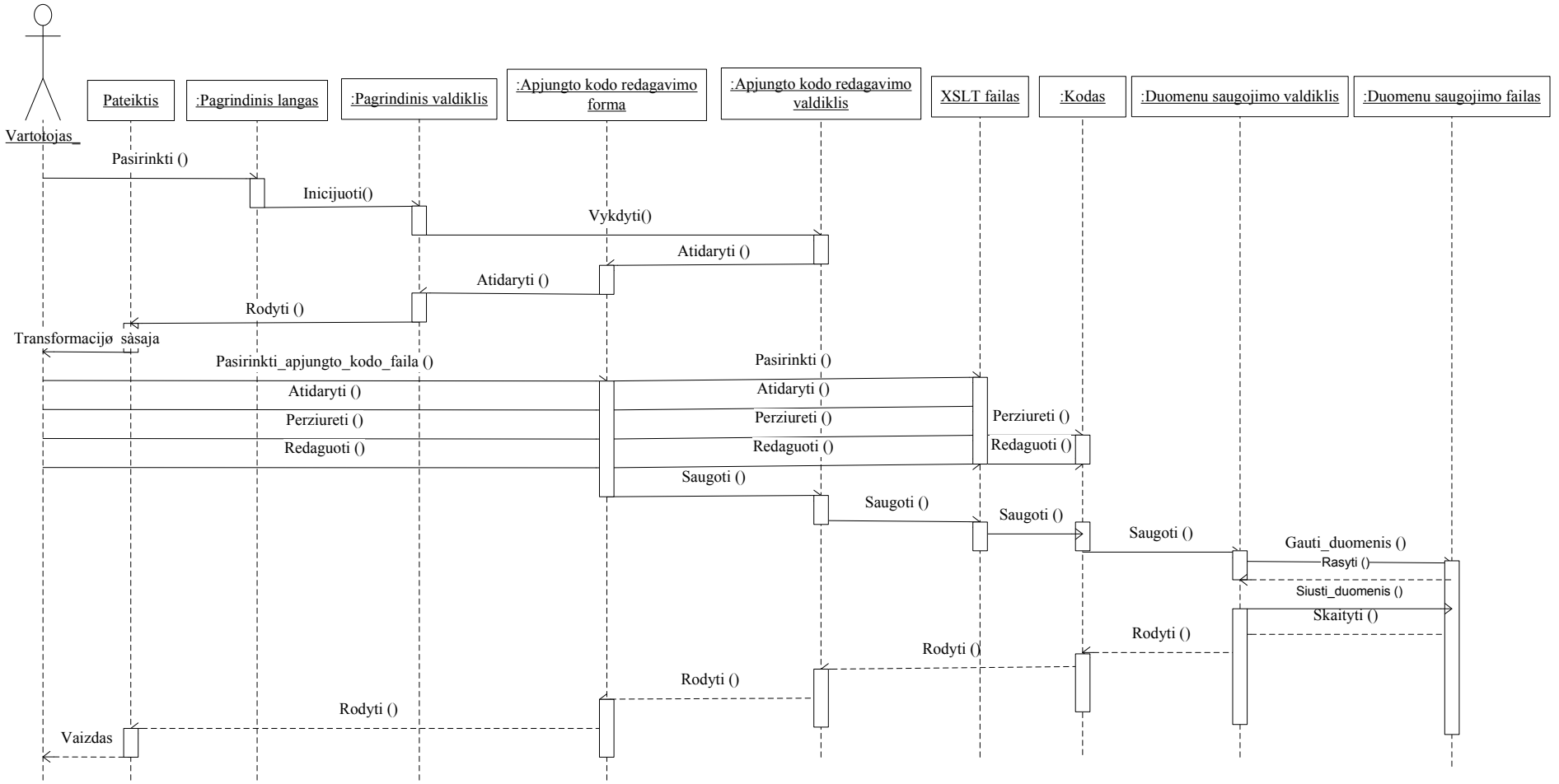
20 pav. PA "Sukurti objektų diagramą" sekos diagrama



21 pav. PA "Sudaryti sistemos specifikaciją" sekos diagrama



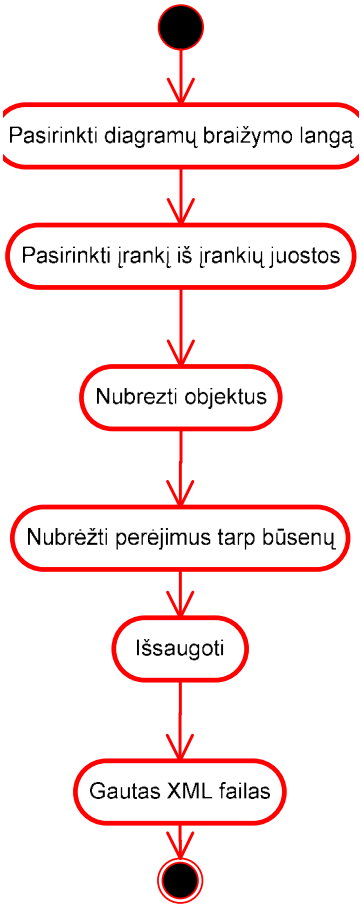
22 pav. PA "Parinkti XSL transformacijas" sekos diagrama



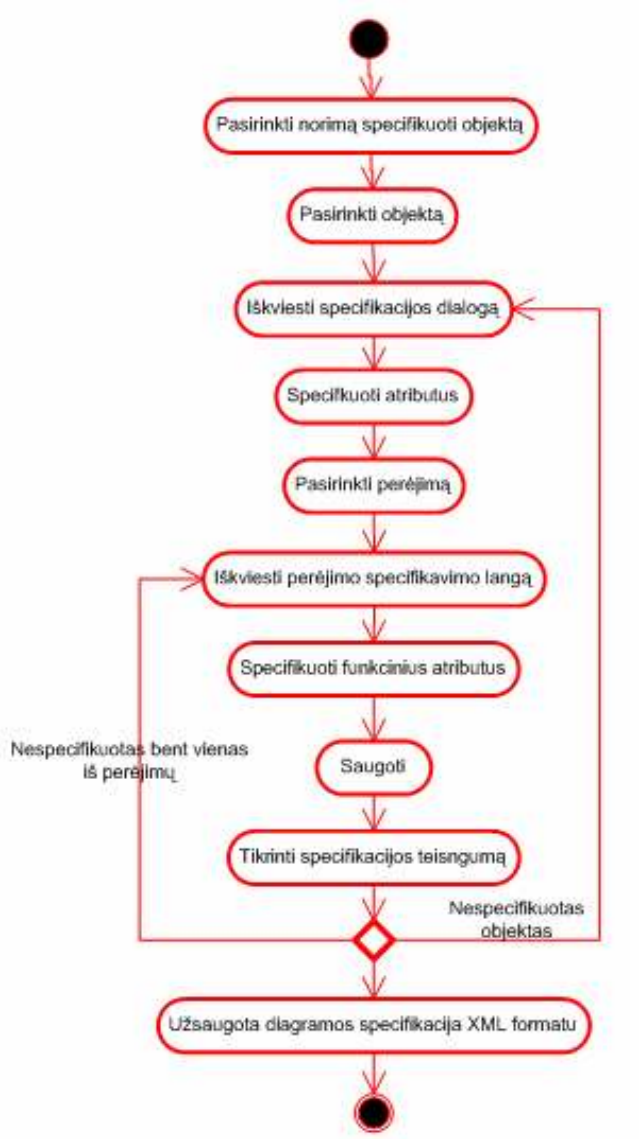
23 pav. PA "Peržiūrėti kodą" sekos diagrama

3.4.2 Panaudojimo atvejų veiklos diagramos

Panaudojimo atvejo „*Sukurti objektų diagramą*“ veiklos diagrama pavaizduota 24 paveiksle.



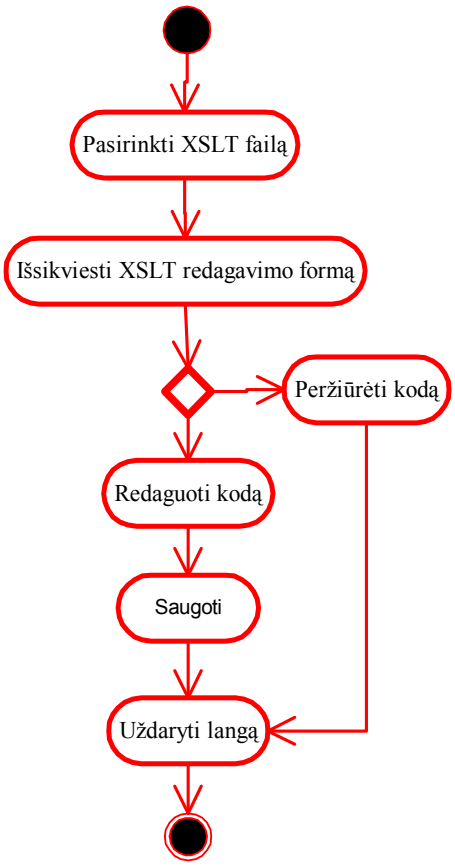
24 pav. PA „Sukurti objektų diagramą“ veiklos diagrama

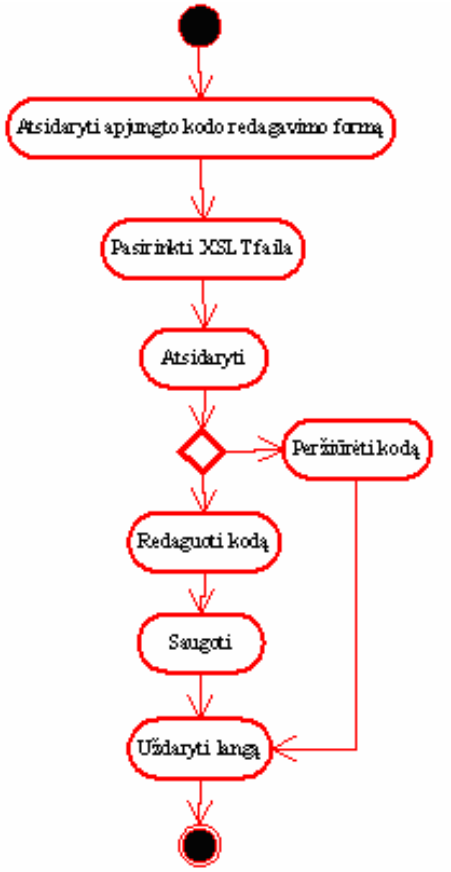


25 pav. PA „Sudaryti sistemos specifikaciją“ veiklos diagrama

Panaudojimo atvejo „*Sudaryti sistemos specifikaciją*“ veiklos diagrama pavaizduota 25 paveiksle.

Panaudojimo atvejo „*Parinkti XSL transformacijas*“ veiklos diagrama pavaizduota 26 paveiksle. Panaudojimo atvejo „*Peržiūrėti kodą*“ veiklos diagrama pavaizduota 27 paveiksle.



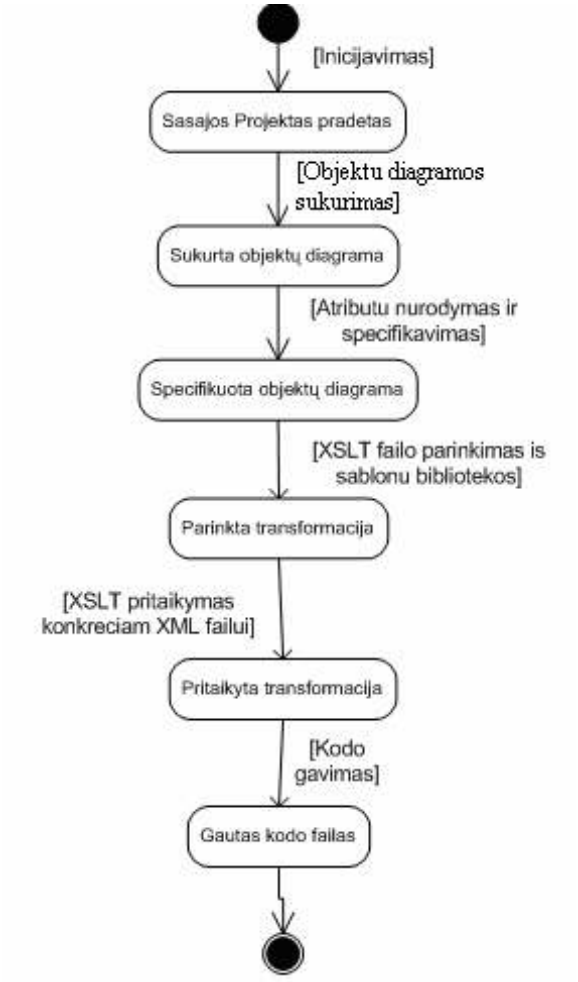


26 pav. PA „Parinkti XSL transformacijas“ veiklos diagrama

27 pav. PA „Peržiūrėti kodą“ veiklos diagrama

3.4.3 Būsenų diagramos

Sąsajos Projekto objekto būsenos viso sąsajos kūrimo proceso metu pavaizduotos 28 paveiksle.

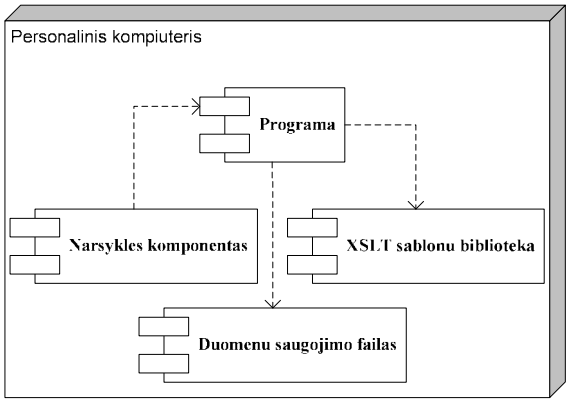


Sąsajos Projekto būsenos

28 pav. Sąsajos projekto būsenų diagrama

3.5. Realizacijos modelis

Kuriama sistema yra suprojektuota veikti pagal kliento architektūrą. Jos komponentų išdėstymo vaizdas pavaizduotas 29 paveiksle.



29 pav. Sistemos išdėstymo vaizdas

3.6. Testavimo modelis bei duomenys, kontrolinis pavyzdys

Testavimo modelis aprašo testinius atvejus, tikrinančius panaudojimo atvejus. Šio skyrelio poskyriuose pateikiami įvairių testavimo modelių, kurie buvo taikomi testuojant sistemą, aprašymai ir gauti rezultatai.

3.6.1 Programinių vienetų testavimo rezultatai

Programinių vienetų testavimas buvo atliekamas programos kodą papildžius testavimo metodu, skirtu skirtingoms klasėms. Vienetų testavimo privalumas – daugkartinis testų panaudojimas.

Buvo generuojami ir naudojami juodos dėžės testai. Juodos dėžės testai buvo generuojami sukuriant testavimo metodą, remiantis klasių metodų aprašais, ir perkompiliuojant programą dar kartą. Po generavimo testiniai atvejai buvo modifikuojami, kad pilnai atitiktų savo paskirtį ir aptiktų galimas klaidas programiniame kode. Taip pat buvo nurodomi testinio atvejo įėjimai ir išėjimai. Kiek įmanoma buvo stengiamasi testais panaudoti vidinius metodų operatorius, pereiti algoritmų šakomis, padengti visus galimus kodo vykdymo kelius.

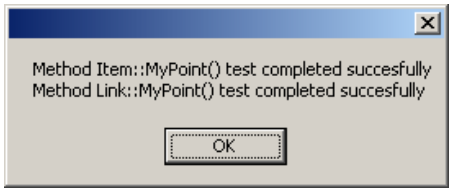
Aptikus klaidas šios buvo taisomos ir testuojama pakartotinai. Šis iteracinis procesas buvo kartojamas tol, kol sukurti testiniai atvejai neberodė klaidų buvimo.

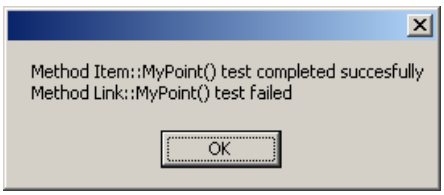
Testavimo metodas:

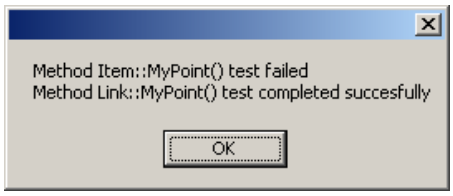
```
   public override bool MyPoint(int x3, int y3)

 private void selfTestToolStripMenuItem_Click(object sender, EventArgs e)
        {
            String bugs = "";
            Item myItem = new Item();
            myItem.CurrentY1 = 1;
            myItem.CurrentX1 = 1;
            myItem.CurrentY2 = 3;
            myItem.CurrentX2 = 3;
            if (myItem.MyPoint(2, 2) && !myItem.MyPoint(4, 4))
                bugs += "Method Item::MyPoint() test completed succesfully\n";
            else
                bugs += "Method Item::MyPoint() test failed\n";

            Link myLink = new Link();
            myLink.CurrentX1 = 2;
            myLink.CurrentY1 = 2;
            myLink.CurrentX2 = 4;
            myLink.CurrentY2 = 4;
            if (myLink.MyPoint(3, 3) && myLink.MyPoint(2, 2) &&
myLink.MyPoint(4, 4) && !myLink.MyPoint(1, 1) && !myLink.MyPoint(5, 5) &&
                !myLink.MyPoint(2, 6) && !myLink.MyPoint(2, 4) &&
!myLink.MyPoint(4, 2))
            {
```

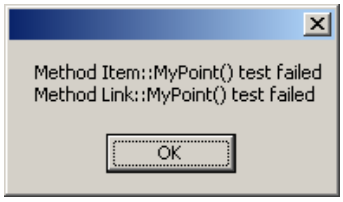
```
                myLink.CurrentX1 = 4;
                myLink.CurrentY1 = 4;
                myLink.CurrentX2 = 2;
                myLink.CurrentY2 = 2;
                if (myLink.MyPoint(3, 3) && myLink.MyPoint(2, 2) &&
myLink.MyPoint(4, 4) && !myLink.MyPoint(1, 1) && !myLink.MyPoint(5, 5) &&
                    !myLink.MyPoint(2, 6) && !myLink.MyPoint(2, 4) &&
!myLink.MyPoint(4, 2))
                {
                    myLink.CurrentX1 = 2;
                    myLink.CurrentY1 = 2;
                    myLink.CurrentX2 = 2;
                    myLink.CurrentY2 = 2;
                    if (!myLink.MyPoint(2, 2) && !myLink.MyPoint(3, 3) &&
!myLink.MyPoint(1, 1) && !myLink.MyPoint(2, 4) && !myLink.MyPoint(4, 2))
                        bugs += "Method Link::MyPoint() test completed
succesfully\n";
                    else
                        bugs += "Method Link::MyPoint() test failed\n";
                }
                else
                    bugs += "Method Link::MyPoint() test failed\n";
            }
            else
                bugs += "Method Link::MyPoint() test failed\n";

            MessageBox.Show(bugs);
        }
```

Testavimo rezultatas programoje:









3.6.2 Scenarijų testavimas

Scenarijų testavimas buvo atliekamas rankiniu būdu. Testavimo scenarijai – vartotojo galimų veiksmų kombinacijų atkartojimas. Šis testavimo etapas padeda testuoti programinės įrangos vartotojo sąsają.

Rankinis testavimo būdas padeda rasti klaidas, kai vartotojas dirba „nestandartiškai“ arba kai veiksmai atliekami ne numatyta ar netikėta tvarka.

3.6.3 Regresinis testavimas

Atliekant sukurto programinio produkto testavimą ir jį bandomąjį diegimą buvo aptinkamos klaidos, kurios vėliau buvo taisomos. Tam kad įsitikinti, kad atlikti pataisymai yra teisingi, ir pataisytas programinis kodas. Regresiniam testavimui buvo atrinktas jau turėtų testavimo atvejų poaibis, pakankamas nustatyti, ar PĮ atitinka reikalavimus po pakeitimų. Testuojant yra galimi keturi rezultatų variantai:

	Yra nauja klaida	**Nėra naujos klaidos**
Sėkmingas pakeitimas	Blogai	Gerai
Nesėkmingas pakeitimas	Blogai	Blogai

Dėl tokios, palyginus mažos, teigiamo rezultato tikimybės reikalingas regresinis testavimas. Jis yra pakankamai imlus laikui, bet dėl galimybės pakartotinai panaudoti jau turimus testinius atvejus.

3.6.4 Testavimo rezultatų apibendrinimas

Atlikus testavimą galima daryti išvadas:

- Testavimas nėra vienkartinė veikla, vykdoma po kodo sukūrimo, bet yra kiekvieno kodo gyvavimo ciklo sudedamoji dalis.
- Testavimas yra daugiau nei vien derinimas. Testavimas yra naudojamas ne vien defektų aptikimui ir taisymui, jis taip pat naudojamas validavimui, patikrai, ir patikimumo įvertinimui.
- Testavimas yra brangus. Testavimo automatizavimas yra geras būdas sumažinti testavimo kaštus ir laiko sąnaudas.
- Visiškas ištestavimas yra neįmanomas, dėl sudėtingumo. Tam tikru momentu programinės įrangos testavimas turi būti nutrauktas ir programinis produktas turi būti atiduotas.
- Daug laiko ir resursų reikalaujantis PĮ kūrimo procesas, tačiau reikalingas, norint užtikrinti tam tikrą PĮ kokybę procesas.
- Baltą dėžę testuoti sunkiau nei juodą, tačiau juodos dėžės testais sunku pasiekti tą patį ištestavimo lygį.

3.7. Sukurto modelio ir jo realizacijos apibendrinimas ir rekomendacijos

Pagal aprašytą technologiją buvo sukurtas įrankio prototipas, leidžiantis generuoti vartotojo sąsają iš dialogo specifikacijos. Įrankis buvo išbandytas generuojant įvairius dialogo scenarijus (pvz: internetinės sąsajos generavimą), taip pat išbandytas kodo generavimas C++ ir JAVA kalboms. Eksperimentai parodė, kad šis įrankis turi plačias taikymo ribas, t.y. galimybė gauti bet kurios programavimo kalbos kodą, priklausomai nuo to, koks XSLT failas pritaikomas, t.y. nuo šio failo turinio, arba norimos internetinės sąsajos vaizdą. Iš vieno dialogo specifikacijos XML duomenų, pritaikius skirtingus XSLT failus, galima gauti klasių aprašus, DB lentelių aprašus, galima sugeneruoti DB procedūras ir sukurti internetinį puslapį. Algoritmą galima taikyti monotoniškiems darbams atlikinėti, t.y. aprašius naują klasę, galima gauti standartinius programinio kodo fragmentus. ProcessDraw, t.y. siūlomą sprendimo realizaciją, galima taikyti tiek mažoms, tiek didelėms sistemoms, tačiau tai yra tik pradinis kodas, toliau reikia programuoti pačiam programuotojui.

Sukurtas metodas, skirtas automatizuotam programos kodo generavimui leidžia:

- Automatizuoti dažnai naudojamų scenarijų ir procedūrų uždavinio kodo generavimą taikant skirtingas XSL transformacijas.
- Pagerinti darbo našumą ir sumažinti klaidų skaičių automatizavus rankiniu būdu atliekamą monotoninį programuotojo darbą.
- Sugeneruoti daugiau tikslinio programos kodo per trumpesnį laiką, taikant konkrečiam tikslui pritaikytas transformacijas, o ne universalius taikomojo uždavinio kodo generavimo įrankius.
- Pagerinti programų sistemos programuotojo ir galutinės suprojektuotos sistemos vartotojo tarpusavio bendradarbiavimą, nes gavus grafinės vartotojo sąsajos vaizdą, vartotojas ir programuotojas gali paprasčiau susikalbėti ir išsiaiškinti sistemos reikalavimus.

Metodą ateityje siūloma tobulinti:

- sukurti atgalinį ryšį tarp programos kodo ir sistemos specifikacijos.
- įrankis galėtų palaikyti klasių diagramas, leidžiančias dalintis vienodais parametrais tarp sistemos objektų.
- sukurti XSLT failų redagavimą pačioje programoje.
- galėtų būti galimybė pasirinkti puslapio dizainą, kai yra projektuojamas grafinės vartotojo sąsajos vaizdas.

4. EKSPERIMENTINIS SISTEMOS TYRIMAS

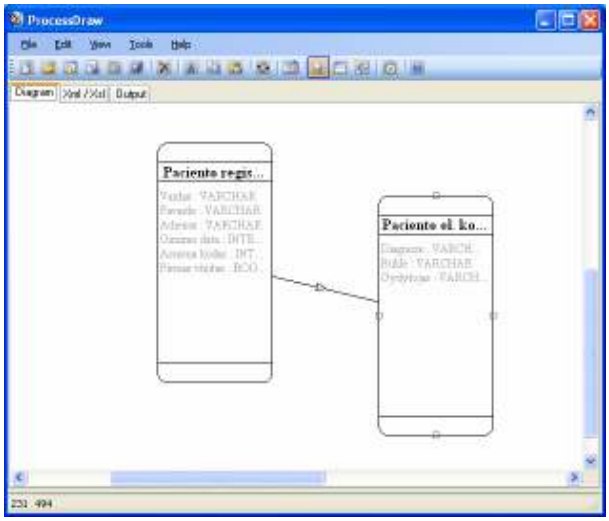
Šioje, eksperimentinėje darbo dalyje yra atliekamas sukurtos ir ištestuotos programinės įrangos bei jos patobulinimų eksperimentinis tyrimas. Pateikiami svarbiausi eksperimentų rezultatai

4.1. Savybių analizė

Norint aiškiai ir detaliai parodyti sistemos galimybes ir savybes, toliau pateikiami skirtingi detalūs pavyzdžiai, suformuluojant uždavinius.

1 uždavinys: Sukurti informacinės sistemos vartotojo sąsajos langus ir sąveikas tarp jų. Gauti grafinį interneto sąsajos vaizdą. Pirmajame lange "Paciento registracija" turėtų būti laukai: vardas, pavardė, adresas, gimimo data, asmens kodas, pirmas vizitas (galima pažymėti lauke) ir perėjimas į kitą interneto sąsajos langą "Paciento el. kortlelė" mygtuku "Registruoti". Antrajame lange turi būti laukai: diagnozė, būklė, gydytojas. Pateikti *.xml ir *.xsl failų kodų pavyzdžius ir grafinius vartotojo sąsajos langus.

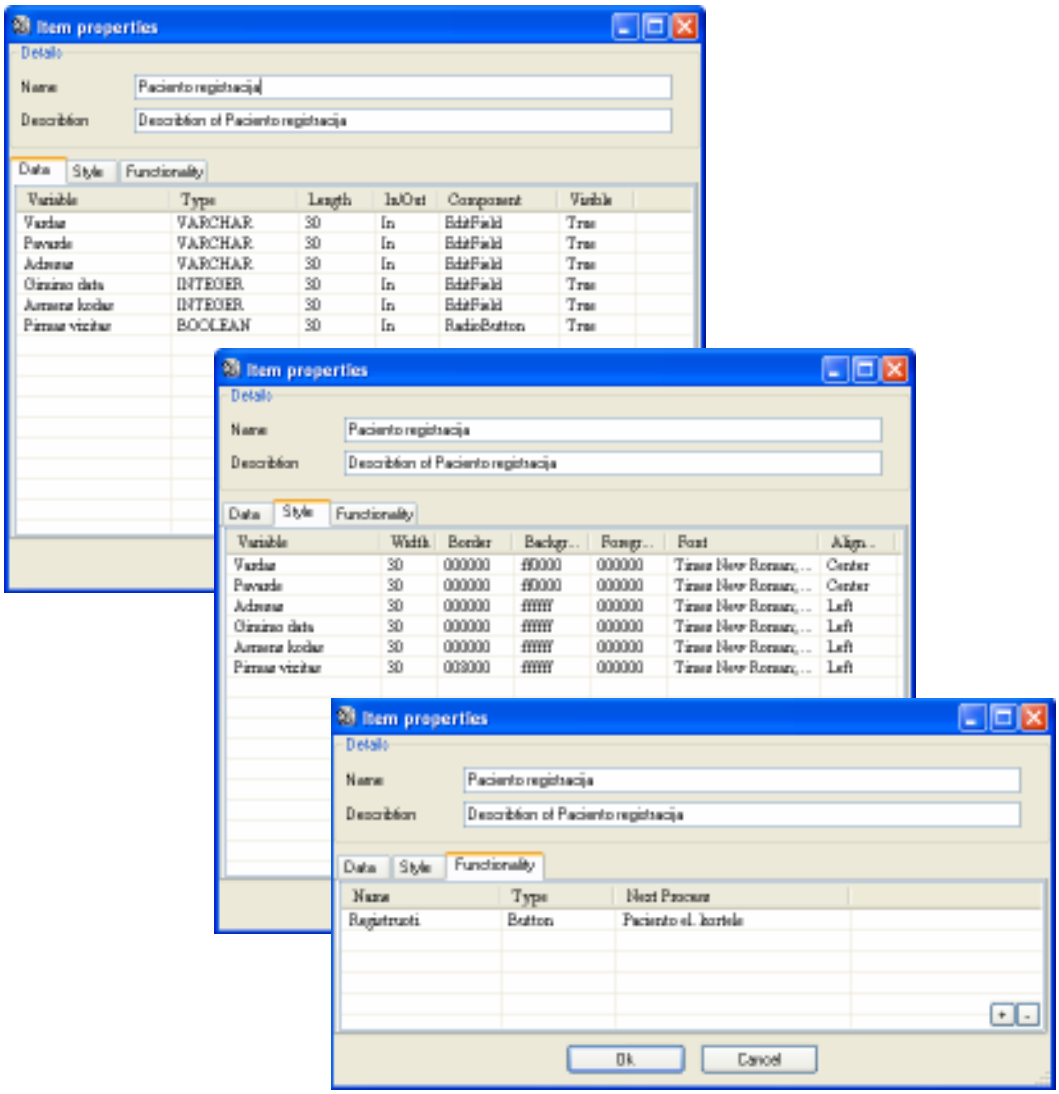
1 žingsnis: objektų diagramos braižymas. Sistemoje, kuri leidžia generuoti internetinę vartotojo sąsają iš dialogo specifikacijos, veikla prasideda paleidus programą ir pirmajame programos lange pradedant braižyti objektų diagramą. Iš įrankių juostos pasirenkamas įrankis skirtas braižyti diagramoms ir ant „balto lapo", matomo programos pirmajame lange braižoma diagrama. Iš karto bus atveriamas ir atributų specifikavimo langas, tačiau nebūtinai reikia įvesti atributus, galite ši langą uždaryti ir įvedinėti atributus tik tuomet, kai būsite nubrėžę objektų diagramą arba bent kelis objektus (30 pav.).



30 pav. Sistemos išdėstymo vaizdas

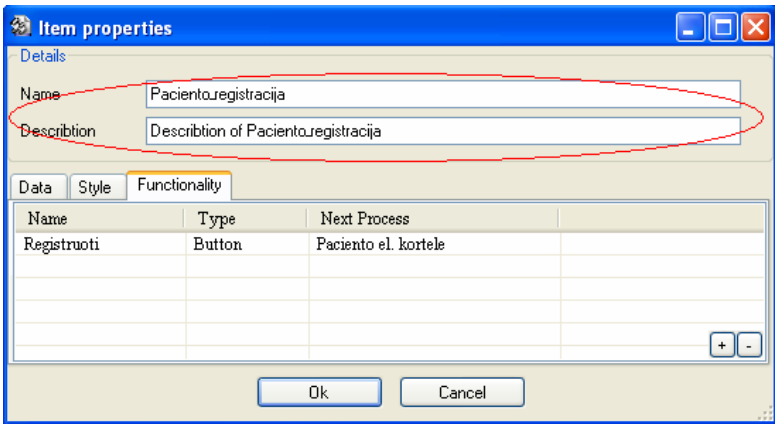
2 žingsnis: specifikacijos sudarymas. Vartotojas pasirinkęs objektą arba perėjimą gali jį specifikuoti.

2a žingsnis: atributų įvedimas. Nubraižius objektą arba perėjimą tarp objektų reikia juos specifikuoti, t.y. įvesti atributus ir juos aprašyti. Atributus galima įvesti tuomet, kai sukuriamas objektas ir tuo pat metu atidaromas specifikavimo langas arba, kai ant nubrėžto objekto spaudžiama du kartus pele, arba įrankių juostoje pasirenkama tam skirta piktograma. Šiame lange galima ne tik įvesti naujus atributus, bet ir juos koreguoti, ištrinti. Taip pat tame pačiame lange galima keisti objekto pavadinimą, įvesti aprašymą kam jis skirtas ir pan. Atliktos operacijos turi būti išsaugomos (31 pav.).



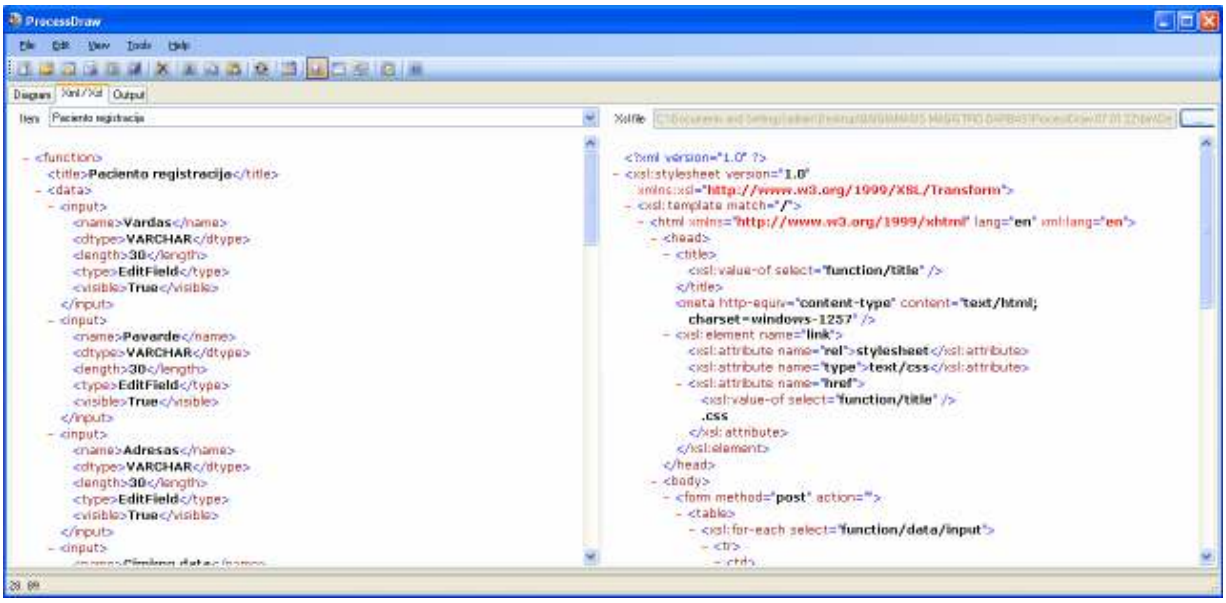
31 pav. Atributų specifikavimo langai

2b žingsnis: objekto savybių koregavimas. Tai forma, skirta koreguoti objekto savybes. Atsidariusiame lange galima koreguoti objekto pavadinimą, įvesti/koreguoti/ištrinti objekto aprašymą (32 pav.).



32 pav. Objekto savybių koregavimo langas

3 žingsnis: duomenų saugojimas xml formatu. Kai brėžiama objektų diagrama pradiniame lange ir įvedami atributai su jų savybėmis, viskas išsaugoma, xml formatu, t.y. sukuriamas *.xml failas, bet taip pat šio failo turinį galima matyti ir programos lange. Norint peržiūrėti xml kodą reikia pasirinkti papildomą meniu punktą „Xml/Xsl“. Kairėje lango pusėje bus matomas xml kodas to objekto, kuris parinktas laukelyje „Item“ (33 pav.). Pavyzdžiui, objektui „Paciento registracija“, kuris turi atributus ir jie yra specifikuoti XML formate išsaugoti duomenys matomi 33 paveiksle.



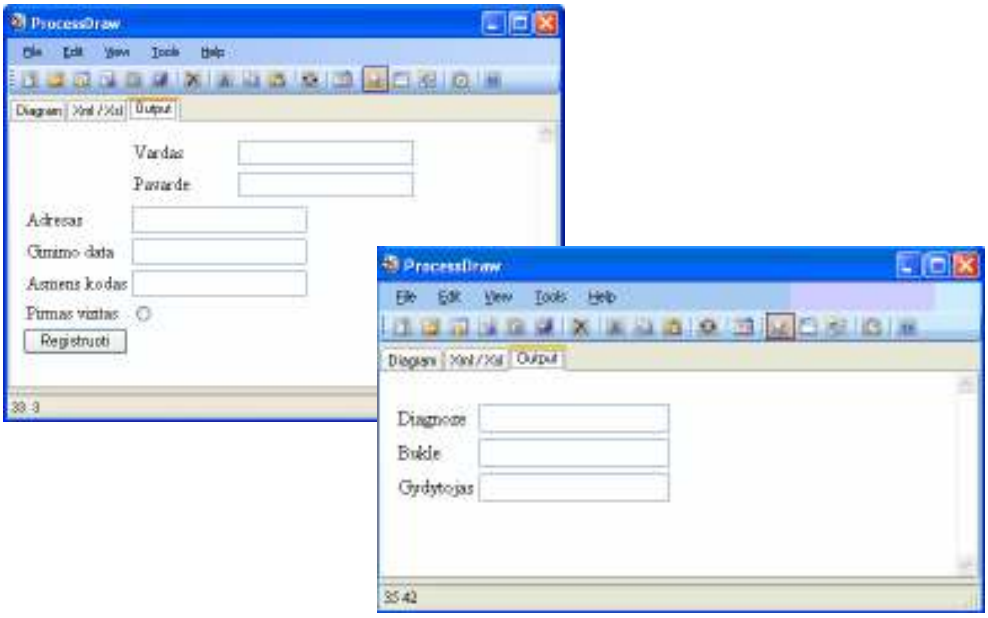
33 pav. XML ir XSLT failų turinio atvaizdavimo langasir matomas kodas

4 žingsnis: transformacijų taikymas. Nepritaikę XSL transformacijų, šiuo atevju gausime HTML kodą. Norint parinkti XSL transformacijas jau prieš tai turi būti sukurtas *.xsl failas. Tada pradiniame programos lange pasirenkamas papildomo meniu punktas „Xml/Xsl“, laukelyje prie „Xsl File“ pasirenkamas jau sukurtas failas ir viskas išsaugoma programoje (34 pav.).

```
<?xml version="1.0" ?>
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
- <xsl:template match="/">
- <html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
- <head>
- <title>
  <xsl:value-of select="function/title" />
  </title>
  <meta http-equiv="content-type" content="text/html; charset=windows-1257" />
- <xsl:element name="link">
  <xsl:attribute name="rel">stylesheet</xsl:attribute>
  <xsl:attribute name="type">text/css</xsl:attribute>
- <xsl:attribute name="href">
  <xsl:value-of select="function/title" />
  .css
  </xsl:attribute>
  </xsl:element>
  </head>
- <body>
- <form method="post" action="">
- <table>
- <xsl:for-each select="function/data/input">
- <tr>
- <td>
- …….
  <xsl:value-of select="caption" />
  </xsl:attribute>
- <xsl:attribute name="onclick">
  location.href=
  <xsl:value-of select="call" />
  ;
  </xsl:attribute>
  </xsl:element>
  </xsl:for-each>
  </form>
  </body>
  </html>
  </xsl:template>
  </xsl:stylesheet>
```

34 pav. XSLT failo turinys (kodo fragmentas)

5 žingsnis: kodo generavimas ir grafinis atvaizdavimas. Kodo generavimui skirtas IE komponentas, kuris iš *.xml ir *.xsl failų sukuria kodą, šie failai yra priskiriami IE komponentui (35 pav.). Sugeneruotą vartotojo sąsajos vaizdą galima pamatyti pasirinkus papildomą meniu punktą „Output“.



35 pav. Grafinis interneto sąsajos atvaizdavimas

<u>2 uždavinys</u>: Sukurti grafinį sąsajos vaizdą, ir gauti JAVA programavimo kalbos kodą. Pateikti *.xsl stilių failo ir galutinio kodo fragmento pavyzdžius.

```
class Paciento_registracija {
String Vardas;
String Pavarde;
String Adresas;
int Gimimo_data;
int Asmens_kodas;
boolean Pirmas_vizitas;

public Paciento_registracija( String Vardas, String Pavarde, String Adresas, int
Gimimo_data, int Asmens_kodas, boolean Pirmas_vizitas) {
this.Vardas = Vardas;
this.Pavarde = Pavarde;
this.Adresas = Adresas;
this.Gimimo_data = Gimimo_data;
this.Asmens_kodas = Asmens_kodas;
this.Pirmas_vizitas = Pirmas_vizitas;
}

public static void main(String args[]) {
Paciento_registracija obj = new Paciento_registracija();
}
}
```

36 pav. Gauto kodo fragmento pavyzdys

```
<?xml version="1.0" ?>
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
- <xsl:template match="/">
  class
  <xsl:value-of select="function/title" />
  {
  <br />
- <xsl:for-each select="function/data/input">
- <xsl:choose>
  <xsl:when test="dtype = 'VARCHAR'">String</xsl:when>
  <xsl:when test="dtype = 'BOOLEAN'">boolean</xsl:when>
  <xsl:otherwise>int</xsl:otherwise>
  </xsl:choose>
  <xsl:text />
  <xsl:value-of select="name" />
  ;
  <br />
  </xsl:for-each>
  <br />
  <br />
  public
  <xsl:value-of select="function/title" />
  (
- <xsl:for-each select="function/data/input">
- <xsl:choose>
  <xsl:when test="dtype = 'VARCHAR'">String</xsl:when>
  <xsl:when test="dtype = 'BOOLEAN'">boolean</xsl:when>
  <xsl:otherwise>int</xsl:otherwise>
  </xsl:choose>
  <xsl:text />
  <xsl:value-of select="name" />
- <xsl:if test="position()!=last()">
  <xsl:text>,</xsl:text>
  </xsl:if>
  </xsl:for-each>
  )
.....
  }
  <br />
  }
  </xsl:template>
  </xsl:stylesheet>
```

37 pav. XSLT failo turinys (kodo fragmentas)

4.2. Kokybės kriterijų įvertinimas

Metodo palyginime galimi du aspektai: kiekybinis ir kokybinis. Pasiūlytąjį metodą įvertinti kiekybiškai buvo neįmanoma, nes:

- sistemos specifikavimas yra subjektyvus dalykas, priklausantis tiek nuo analitiko patirties, žinių, tiek nuo pasirinkto specifikacijos detalumo lygmens, todėl kiekybiškai neįmanoma išmatuoti nei laiko, nei specifikacijos tikslumo.
- neįmanoma kiekybiškai įvertinti bendro sistemos grafinės vartotojo sąsajos sukūrimo.
- atskiriems internetiniams puslapiams sugeneruotas programos kodas priklauso nuo transformacijų tikslumo, kurios kuriamos, atsižvelgiant į darbo specifiką, ir negali būti

lyginamas su standartiškai automatizuoto generavimo priemonėmis gautu programos kodu.

Taigi atsižvelgiant į atlikto darbo specifiką, metodas buvo vertinamas, naudojant kokybinius kriterijus. Metodas buvo įvertintas 6 žmonių, kurie projektuoja ir kuria taikomąsias programas.

Programinės įrangos įvertinimas buvo remiantis ISO/IEC 9126 standarte pateikiamu kokybės vertinimo modeliu. Standarte pateiktoms kokybės charakteristikoms buvo pasirinkta grupė subcharakteristikų iš rekomenduojamų subcharakteristikų sąrašo. Subcharakteristikoms turi būti parenkamos metrikos, pagal kurias atliekamas vertinimas – matavimas.

Pateikiama kokybės vertinimo lentelė (46 lentelė) su subcharakteristikų paaiškinimais.

46 lentelė. Kokybės vertinimas

Subcharakteristika	Įvertinimas	Apibūdinimas
Tinkamumas	10	Produkto teikiamos funkcijos (operacijos) pilnai pateikia funkcionalumą, kuris yra reikalaujamas reikalavimų specifikacijos dokumente.
Tikslumas	10	Produkto pateikiami rezultatai atitinka rezultatus, kurie yra reikalaujami pagal reikalavimų specifikacijos dokumentą.
Saugumas	9	Produkto saugumas užtikrinamas kelių lygių apsaugos sistema.
Suprantamumas	9	Dėl pasirinktos programavimo kalbos, tvarkos, komentarų ir paaiškinimų programos kode, programą nesunku suprasti ir keisti.
Išmokstamumas	10	Programinės įrangos vartotojo sąsaja yra lengvai suprantama ir išmokstama.
Valdymas	10	Programinė įranga nereikalauja didelių vartotojo pastangų vykdant arba valdant pateikiamas operacijas.
Laiko režimas	9	Laikas, skirtas projektavimui, t.y. objektų diagramos braižymui, atsižvelgiant į XML kodo saugojimą, XSLT parinkimą, vartotojo sąsajos pateikimą.
Resursų režimas	10	Programa sukurta C# kalba. Kuriant programą resursų nuotėkis nepastebėtas ir visi įjungti procesai pasibaigia.
Analizuojamumas	9	Visa programinė įranga, reikalinga programos veikimo užtikrinimui yra pagaminta Microsoft. O iškilus sunkumams reiktų vadovautis dažniausiai pasitaikančių klaidų sprendimo būdais kurie pateikiami vartotojo dokumentacijoje.
Keičiamumas	10	PĮ kodas yra parašytas aiškia ir lengvai skaitoma ir suprantama C# kalba.
Testuojamumas	8	PĮ yra lengvai testuojama, kadangi testiniai atvejai yra nesudėtingai išgaunami iš sistemos reikalavimų specifikacijos.
Įdiegimo galimybės	10	Programą įdiegti yra paprasta.

5. IŠVADOS

- Analizės metu buvo nustatyta, kad galima pagerinti vartotojo sąsajos automatizuoto kūrimo procesus grafiškai projektuojant jos veikimą ir jungiant veikimo modelį su kuriamos sąsajos vaizdu.
- Grafinės vartotojo sąsajos veikimo specifikavimui tikslinga naudoti unifikuotosios modeliavimo kalbą UML, nes ji yra lengvai išmokstama ir patogi realizavimui, dokumentavimui, leidžia įvairiapusiškai specifikuoti sistemą dėl savo diagramų gausybės, taip pat yra nepriklausoma nuo konkrečios programavimo kalbos, leidžia aiškiai išreikšti projektuotojo idėjas.
- Pagal objektinę metodologiją vartotojo dialogo modeliavimui geriausiai tinka būsenų diagramos, kadangi jos geriausiai atspindi į vartotojų įvykius reaguojančios sistemos veikimą.
- Iš praktinės patirties ir atliktos sąsajos projektuotojų poreikių analizės, sąsajos specifikavimui pasiūlyta naudoti objektų diagramas, kur kiekviena būsena aprašoma kaip objektas, objektų ryšiai vaizduoja būsenų perėjimus.
- Specifikacijų saugojimui ir apdorojimui buvo pasirinkta XML kalba, modelių transformacijoms į programos kodą - XSL transformacijos. Jų pasirinkimą lėmė suderinamumas tarpusavyje, lengvas palaikymas ir paprastumas.
- Eksperimentai parodė, kad sukurtas metodas ir įrankis, kuris aptariamas šiame straipsnyje, leidžia grafiškai projektuoti grafinės vartotojo sąsajos veikimą ir jungti veikimo modelį su kuriamos sąsajos vaizdu.
- Sukurto įrankio testavimas ir vartotojų apklausa bei kokybinis įvertinimas patvirtino jo tinkamumą sąsajai generuoti.
- Ateityje tikslinga išplėsti įrankį kitos rūšies UML diagramų palaikymu, taip surinkti daugiau informacijos apie sistemą, tuo palengvinant objektų aprašymą ir supaprastinant veiklos modeliavimą. Tikslinga sukurti atgalinį ryšį tarp programos kodo ir sistemos specifikacijos, įrankis galėtų palaikyti klasių diagramas, leidžiančias dalintis vienodais parametrais tarp sistemos objektų, galėtų turėti atgalinį ryšį tarp programos kodo ir sistemos specifikacijos
- 2 straipsniai magistrinio darbo tema buvo pristatyti tarpuniversitetinėse doktorantų ir magistrantų konferencijose (2006 ir 2007m.)

LITERATŪRA

[1] Booch G., and etc. The unified modeling language user guide. Addison Wesley Longman Inc., 1999, 342 p.

[2] Booch G. Object-Oriented Analysis and Design with Applications. Addison-Wesley Publishing company. 1993, 473 p.

[3] Booch G., Rumbaugh J., Jacobson I. The unified modeling language user guide, Addison Wesley Longman Inc. 1999, 218 – 225 p.

[4] Collins D. Designing Object-Oriented User Interfaces. Benjamin/Cummings, Redwood City, CA, 1995.

[5] Erik T. Ray. Learning XML, Second Edition, O'Reilly, Santa Clara, CA. 2003, 27-143 p.

[6] Gamma E., Helm R., Johnson R., Vlissides J. Design patterns: elements of reusable object-oriented software. Addison-Wesley, 1995, 2-7 p.

[7] Yang H. Advances In Uml And Xml-based Software Evolution, idea Group Inc., 2005. ISBN: 1-59140-623-4.

[8] Hans-Erik Eriksson, Magnus Penker. Business Modeling with UML: Business Patterns at Work. John Wiley & Sons, Inc, 2000.

[9] Herrington, J. Code generation in action 2003. Manning Publications Co., ISBN 1-930110-97-9

[10] Hunter D., Cagle K., Dix et K. Beginning XML, 2nd Edition: XML Schemas, SOAP, XSLT, DOM, and SAX 2.0. Wrox Press © 2003

[11] IEEE Standard VHDL Language Reference Manual (Integrated with VHDL-AMS changes), IEEE Std.1076.1.

[12] International Telecommunications Union - Technical section. Functional Specification and Description Language SDL. ITU-T Recommendation Z.100, 1994.

[13] Jorgensen D., Ortiz J. Developing .NET Web Services with XML. Syngress Publishing, Inc. 2002. ISBN: 1-928994-81-4.

[14] Kay M. XSLT 2.0 Programmer's Reference (Programmer to Programmer), Wrox; 3 edition, 2004, 173-493 p., ISBN: 0764569090.

[15] Krasner G. E. and Pope S. T. A cookbook for using themodel-view-controller user interface paradigm in Smalltalk-80. *JOOP*, 1988, 26-49 p.

[16] Pierre-Alain Muller. Instant UML. Wrox Press Ltd.,1997. 343p. ISBN 0-20142-765-6

[17] Rumbaugh J., Baha M., Premerlani W., Eddy F. Object-Oriented Modeling and Design. Englewood Cliffs, New Jersey, Prentice-Hall. 1991, 97 – 102 p.

[18] Swamy S., Molin A., Covnot B. OO-VHDL. Object-oriented extensions to VHDL. Computer, Volume: 28, Issue: 10, Oct 1995, 18-26 p.

[19] Thierry Bodhuin, Enrico Guardabascio and Maria Tortorella, "Migrating COBOL Systems to the WEB by using the MVC design pattern", The 9th Working Conference on Reverse Engineering, 2002, Italy.

[20] Informacijos sistemų katedra. [interaktyvus]. [Žiūrėta 2005 11]. Prieiga internete: <http://www.isd.ktu.lt>

[21] Lietuvos statistikos departamentas. [interaktyvus]. [Žiūrėta 2007 04]. Prieiga internete < http://www.stat.gov.lt/lt/ >

[22] Chauvel F., Jezequel JM. Code generation from UML Models with semantic variation points.[interaktyvus]. [žiūrėta 2006 12]. Prieiga per internetą: <http://www.irisa.fr/triskell/public/2005/Chauvel05a.pdf >

[23] FrontPage 2003 Product Guide. [Interaktyvus]. [Žiūrėta 2006-02]. Prieiga internete: < http://www.microsoft.com/office/frontpage/prodinfo/guide.mspx >

[24] James Hendler Bijan Parsia. XML and the Semantic Web. Oct. 2002. XML Journal. [Interaktyvus]. [Žiūrėta 2007-02], Prieiga internete: <http://www.syscon.com/magazine/?issueid=123&src=false >

[25] Manage your Web site with Visio Professional 2002. [Interaktyvus]. [Žiūrėta 2006 03]. Prieiga internete: <http://office.microsoft.com/en-gb/assistance/HA010352441033.aspx>

[26] Model-View-Controller (MVC). [interaktyvus]. [Žiūrėta 2005 11]. Prieiga internete: <http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html>

[27] No Magic, Inc. MagicDrawTM UML 10.0 User manual. [Interaktyvus]. [Žiūrėta 2005 10]. Prieiga internete: <http://www.magicdraw.com/files/manuals/9.5/MagicDrawUserManual.pdf?NMSESSID=cf04df77b7801bfbe552ad02a66b3271>

[28] Object Management Group. MOF 1.4 specification.. [interaktyvus]. [Žiūrėta 2005 11]. Prieiga internete: <http://www.omg.org/cgi-bin/doc?formal/2002-04-03.pdf>

[29] Object Management Group. XML Metadata Interchange (XMI). OMG Document ad/98-10-05. [interaktyvus]. [Žiūrėta 2005 10]. Prieiga internete: <http://www.omg.org/docs/ad/02-04-10.pdf>

[30] Object Management Group. Request for Proposal: MOF 2.0 Query / Views / Transformations RFP. OMG Document: ad/2002-04-10. [Interaktyvus]. [Žiūrėta 2005-10]. Prieiga internete: <http://www.omg.org/docs/ad/02-04-10.pdf>

[31] Rational Application Developer for WebSphere Software; Eclipse-based UML Visualization and Java Development. [Interaktyvus]. [Žiūrėta 2005-11]. Prieiga internete : <http://www-306.ibm.com/software/awdtools/developer/application/index.html>

[32] Shane Sendall, Gilles Perrouin, Nicolas Guelfi, Olivier Biberstein. Supporting Model-to-Model Transformations: The VMT Approach. [Interaktyvus]. [Žiūrėta 2005-10]. Prieiga internete: <http://se2c.uni.lu/tiki/se2c-bib_abstract.php?id=238>

[33] The code project. VisualClassBuilder - Free Code Generation Tool. By darkoman. [Interaktyvus]. [Žiūrėta 2007-05-02]. Prieiga internete: < http://www.codeproject.com/tools/visualclassbuilder.asp >

[34] Unified Modeling Language. Notation Guide Version 2.1 [Interaktyvus]. [Žiūrėta 2005-10]. Prieiga internete: <http://www.omg.org/>

TERMINŲ IR SANTRUMPŲ ŽODYNAS

C#	Bendros paskirties, aukšto lygio, objektiškai orientuota programavimo kalba sukurta Microsoft.
CASE	Programinė įranga ar programų paketai, skirti supaprastinti programų sistemų kūrimą ir palaikymą. Įvairūs įrankiai skirti skirtingiems programų kūrimo ciklo etapams – projekto valdymui, verslo ar funkcinei analizei, projektavimui, kodo saugojimui ir versijavimui, kompiliavimui, testavimui ir pan. (angl. Computer-aided software engineering)
CORBA	Standartų sistema koordinuotam kelių programų darbui Internete (angl. Common Object Request Broker Architecture)
EJB	angl. Enterprise JavaBeans
IDE	Integruota kūrimo aplinka (angl. integrated development environment). Taip pat žinoma kaip integruota projektavimo aplinka (anlg. integrated design environment) ir integruota klaidų aptikimo aplinka (angl. integrated debugging environment) yra tam tikra programinės įrangos rūšis, kuri padeda programuotojams kuriant programinę įrangą.
IS	Informacinė sistema
JAVA	Bendros paskirties, aukšto lygio, objektiškai orientuota, nuo platformos nepriklausoma programavimo kalba sukurta Sun Microsystems firmoje.
HTML	Hiperteksto žymėjimo kalba (angl. Hypertext Markup Language) - tai kompiuterinė žymėjimo kalba, naudojama pateikti turinį internete. Ji naudojama apibrėžti dokumento struktūrai.
GUI	Grafinė vartotojo sąsają (angl. Graphical User Interface)
MOF	Meta objektų bazė (angl. Meta Object Facility)
MVC	Modelių peržiūros valdiklis (angl. Model-View-Controller)
OO	Objektiškai orientuotas (angl. Object oriented)
QVT	Užklausa/Vaizdas/Transformacija (angl. Query/View/Transform)
PĮ	Programinė įranga
RUP	angl. Rational Unified Process
RAD	angl. Rational Application Developer
UML	Unifikuota modeliavimo kalba (angl. Unified Modeling Language)

VHDL	Labai aukšto lygio projektavimo kalba (angl. Very High-level Design Language). Tai skaitmeninės elektroninės aparatūros funkcionavimo aprašymo kalba. Tai aparatūros tarptautinis standartas. VHDL leidžia aprašyti bet kokį aparatūriškai realizuojamą algoritmą, todėl VHDL kalba labai panaši į programavimo kalbą. Ši kalba tinkama modeliavimui.
W3C	Tarptautinis konsorciumas, kur organizacijos narės, darbuotojai ir visuomenė dirba kartu kurdamos naujus žiniatinklio standartus (angl. World Wide Web Consortium)
WYSIWYG	ką matote, tą ir gaunate (angl. what you see is what you get)
XML	Išplečiama žymių kalba (angl. eXtensible Markup Language)
XSL	Išplečiama stilių kalba (angl. eXtensible Stylesheet Language)
XSLT	Išplečiamos stilių kalbos transformacijos (angl. Extensible Stylesheet Language Transformations)

PRIEDAI

1 priedas. Straipsnis skelbtas 11-ojoje tarpuniversitetinėje doktorantų ir magistrantų konferencijoje, „Informacinės technologijos 2006“ (IT‘06)

2 priedas. Straipsnis skelbtas 12-oje tarpuniversitetinėje doktorantų ir magistrantų konferencijoje, „Informacinė visuomenė ir universitetinės studijos“ (IVUS‘07) “

3 priedas. Straipsnis skelbtas mokslinėje techninėje konferencijoje, „Informacinės technologijos 2007“. KTU.

4 priedas. Straipsnis skelbtas 4-ojoje mokslinėje praktinėje konferencijoje, „Informacinių technologijų taikymas švietimo sistemoje 2006“.