



VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INŽINERINĖS GRAFIKOS KATEDRA

Deividas Celencevičius

VARTOTOJO SĄSAJOS PROTOTIPŲ ANALIZĖ IR JŲ GENERAVIMAS
USER INTERFACE PROTOTYPE ANALYSIS AND ITS GENERATION

Baigiamasis magistro darbas

Informacinių technologijų studijų programa, valstybinis kodas 621E14004

Inžinerinės ir kompiuterinės grafikos specializacija

Informatikos inžinerijos studijų kryptis

Vilnius, 2014

ANOTACIJA

Vartotojo sąsajos prototipų analizė ir jų generavimas

Baigiamajame magistro darbe analizuojamos informacinių sistemų projektavimo procese išsylančios problemos dėl vartotojo sąsajos prototipų kūrimo. Išnagrinėti pagrindiniai principai, kuriais remiantis kuriami prototipai. Pateikti keturių skirtingų UML diagramų ir vartotojo sąsajos elementų modeliavimo įrankiai bei atlikta jų palyginamoji analizė. Baigiamajame darbe pasiūlytas automatizuotas vartotojo sąsajos prototipų generavimo algoritmas. Buvo suprojektuota pastatų renovacijos informacinė sistema, o jos vartotojo sąsajos prototipams pritaikytas automatizuotas generavimo iš UML diagramų algoritmas. Išanalizavus teoriškai ir praktiškai vartotojo sąsajos prototipų kūrimą projektavimo metu, pateikiamos baigiamojo darbo išvados ir siūlymai.

Darbą sudaro: įvadas, literatūros apžvalga ir analizė, rezultatai ir jų aptarimas, projektavimas, išvados ir siūlymai, literatūros sąrašas.

Darbo apimtis - 76 p. teksto be priedų, 39 iliustr., 19 lent., 21 bibliografiniai šaltiniai.

Atskirai pridedami darbo priedai.

Prasminiai žodžiai: Prototipai, projektavimas, modeliavimas, UML, vartotojo sąsaja, projektas, informacinė sistema, modeliavimo įrankiai, būsto renovacija.

ANNOTATION

User Interface Prototype Analysis and its Generation

The aim of this work is to analyze problem that occurred during information system design process. Main problem is user interface prototyping and resources spent on it. The results of the analytical part show that prototyping is a must for qualitative project. The four different UML diagrams and user interface element modeling tools, along with a comparative analysis of them. Thesis suggested automated UI prototype generation algorithm. During design phase author designed building renovation information system and its user interface prototypes were adapted with automatic generation of UML diagrams algorithm. The analysis of the theoretical and practical user interface prototyping during the design process, the final conclusions and recommendations were made.

Structure: introduction, literature review and analysis, results and their discussion, design, conclusions and suggestions, references.

Total 76 pages of text, 39 figures, 19 tables, 21 bibliographic sources.

Keywords: Prototype, design, modeling, UML, user interface, project, information system, modeling tools, renovation.

TURINYS

PAVEIKSLŲ IR LENTELIŲ SĄRAŠAS	6
SANTRUMPŲ SĄRAŠAS	8
IVADAS	9
1. LITERATŪROS APŽVALGA IR ANALIZĖ	11
1.1. Informacinių sistemų kūrimas	11
1.2. Projektavimo svarba projekto vykdymo etapuose	12
1.3. Informacinių sistemų modeliavimas	14
1.3.1. Vieninga modeliavimo kalba	15
1.3.2. Diagramų tipai	16
1.4. Vartotojo sąsajos modeliavimas	21
1.5. Prototipai	22
1.5.1. Prototipų kūrimo procesas	23
1.5.2. Prototipų panaudojimas	24
1.5.3. Projekto kūrimo grupės dalyvių analizė	25
1.5.4. Prototipo kūrimo pradžia	29
1.5.5. Prototipo funkcionalumas	29
1.5.6. Prototipo apimčių nustatymas	30
1.6. Modeliavimo įrankių analizė	33
1.6.1. MagicDraw UML	34
1.6.2. Microsoft Office Visio	39
1.6.3. Oracle Forms Builder	42
1.6.4. Enterprise Architect	44
1.7. Vartotojo sąsajos prototipo generavimo algoritmas	46
1.7.1. Diagramų ryšys su vartotojo sąsaja	46
1.7.2. Diagramų privalomi elementai	49
1.7.3. Modeliavimo diagramų vartotojo sąsajos elementai	50
1.7.4. Vartotojo sąsajos elementai internetinėje sąsajoje	51
1.7.5. Vartotojo sąsajos elementų apjungimas	55
1.7.6. Vartotojo sąsajos generavimo procesas	56
2. REZULTATAI IR JŲ APTARIMAS	57
2.1. Sistemos kūrimas	57
2.2. Prototipų naudojimas	57
2.3. Pasirinktas modeliavimo įrankis	58
2.4. Prototipo generavimo algoritmo apibendrinimas	60
3. PROJEKTAVIMAS	60

3.1.	Sistemos duomenys	60
3.1.1.	Panaudos atvejų diagrama	61
3.1.2.	Klasių diagrama	63
3.1.3.	Veiklų diagrama.....	64
3.2.	Vartotojo sąsajos prototipas	65
3.3.	Vartotojo sąsajos generavimo rezultatai.....	66
	IŠVADOS IR PASIŪLYMAI	68
	LITERATŪROS SĄRAŠAS.....	69
	PRIEDAI	71
	Priedas Nr. 1. Straipsnis, pristatytas 2013 m. gegužės 24 d. Kaune vykusioje konferencijoje „Inžinerinė ir kompiuterinė grafika“	71

PAVEIKSLŲ IR LENTELIŲ SĄRAŠAS

Lentelės:

1 lentelė. IT projektų nesėkmių priežasčių sąrašas	12
2 lentelė. Panaudos atvejų diagramos elementų sąrašas	17
3 lentelė. Klasų diagramos elementų sąrašas	18
4 lentelė. Veiklos diagramos elementų sąrašas	20
5 lentelė. Proceso dalyvių sąrašas ir aprašymas	26
6 lentelė. Sistemos kūrimo proceso žingsniai ir jų aprašymas	27
7 lentelė. Programinės įrangos vertinimo kriterijų sąrašas	33
8 lentelė. <i>MagicDraw UML</i> įrankio keliamų reikalavimų ir galimybių lentelė	35
9 lentelė. <i>MagicDraw UML</i> įrankio privalumai ir trūkumai	39
10 lentelė. <i>Microsoft Office Visio</i> įrankio keliamų reikalavimų ir galimybių lentelė	41
11 lentelė. <i>Microsoft Office Visio</i> įrankio privalumai ir trūkumai	41
12 lentelė. <i>Oracle Forms Builder</i> įrankio keliamų reikalavimų ir galimybių lentelė	43
13 lentelė. <i>Oracle Forms Builder</i> įrankio privalumai ir trūkumai	43
14 lentelė. <i>Enterprise Architect</i> įrankio keliamų reikalavimų ir galimybių lentelė	45
15 lentelė. <i>Enterprise Architect</i> įrankio privalumai ir trūkumai	45
16 lentelė. Privalomų elementų diagramose sąrašas	50
17 lentelė. Vartotojo sąsajos elementų sąrašas	51
18 lentelė. HTML vartotojo sąsajos elementų sąrašas	52
19 lentelė. Vartotojo sąsajos elementų sąrašas	53

Paveikslai:

1 pav. Dinaminė RUP struktūra	13
2 pav. Dinaminė RUP struktūra optimizavus analizės ir projektavimo darbus ir paankstinus realizacijos laiką	14
3 pav. UML diagramų medis	15
4 pav. Pareto dėsnis (20/80)	16
5 pav. Panaudos atvejų diagramos braižymas naudojant modeliavimo objektus	17
6 pav. Klasų diagramos braižymas naudojant modeliavimo objektus	18
7 pav. Veiklų diagramos braižymas naudojant modeliavimo objektus	19
8 pav. OS rinkos dalis nuo 2014 kovo pagal <i>Net Applications</i> atliktą tyrimą	22
9 pav. Prototipo kūrimo proceso diagrama	23

10 pav. Informacinės sistemos kūrimo procesas	27
11 pav. Mažo tikslumo prototipo pavyzdys.....	32
12 pav. MagicDraw PĮ galimybės kuriant informacines sistemas (No Magic 2014).....	35
13 pav. GUI kūrimo elementų grupės.....	36
14 pav. GUI kūrimo elementų visas sąrašas	37
15 pav. Windows operacinės sistemos mygtukų sąrašas	37
16 pav. Grafiniai teksto elementai	38
17 pav. Kiti grafinės vartotojo sąsajos kūrimo elementai.....	38
18 pav. Pavyzdinis grafinės vartotojo sąsajos prototipas sukurtas naudojant <i>MagicDraw UML</i> .	39
19 pav. <i>Microsoft Office Visio</i> standartinių diagramų pasirinkimas.....	40
20 pav. <i>Microsoft Office Visio</i> vartotojo sąsajos prototipo kūrimas naudojant „Wireframe“ papildymą	40
21 pav. <i>Oracle Forms Builder</i> programinės įrangos pradinis langas su elementais	42
22 pav. <i>Oracle Forms Builder</i> vartotojo sąsajos prototipo pavyzdys	43
23 pav. <i>Enterprise Architect</i> darbalaukis.....	44
24 pav. Vartotojo sąsajos kūrimo iš diagramų proceso modelis	47
25 pav. Grafinės vartotojo sąsajos generavimas iš UML diagramų	48
26 pav. Grafinės vartotojo sąsajos generavimas iš UML diagramų naudojant realius duomenis .	49
27 pav. UML diagramų klasių tarpusavio ryšių modelis.....	50
28 pav. Internetinės vartotojo sąsajos elementų modelis.....	53
29 pav. Modeliavimo diagramų ir HTML vartotojo sąsajos elementų klasių apjungimas	55
30 pav. Grafinės vartotojo sąsajos generavimo iš diagramų veiklos procesas	56
31 pav. XML kodo dalis, gauta iš veiklų diagramos, naudojant <i>MagicDraw UML</i>	59
32 pav. UML klasių ir XML elementų sujungimas	59
33 pav. Būsto renovacijos IS panaudos atvejų diagrama	62
34 pav. Būsto renovacijos IS klasių diagrama	63
35 pav. Būsto renovacijos IS tiekėjo administravimo veiklų diagrama	64
36 pav. Būsto renovacijos IS vartotojo sąsajos prototipas	65
37 pav. Naujo vartotojo sąsajos prototipo lango generavimas iš panaudos atvejo.....	66
38 pav. Vartotojo sąsajos prototipo lango užpildymas klasės atributų laukais	66
39 pav. Tiekėjo kortelės vartotojo sąsajos lango prototipas	67

SANTRUMPŲ SĄRAŠAS

CRUD (angl. *Create, Read, Update, Delete*) – skaityti, rašyti, atnaujinti ir trinti.

CSS (angl. *Cascading Style Sheets*) – kalba, skirta nusakyti kita struktūriniu kalba aprašyto dokumento vaizdavimą.

DDL (angl. *Data Definition Language*) – kompiuterinė kalba aprašanti duomenų struktūras.

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

IT (angl. *Information Technology*) – informacinės technologijos.

Java, C#, C++ ir CORBA IDL – programavimo kalbos.

MDA (angl. *Model-driven architecture*) – modeliais grindžiama architektūra.

OMG (angl. *Object Management Group*) – objektinio modeliavimo organizacija.

PHP (angl. *Hypertext Preprocessor*) – programavimo kalba.

RUP (angl. *Rational Unified Process*) – IBM organizacijos sukurtas programų sistemų kūrimo procesas.

SAX (angl. *Simple API for XML*) – įvykiais grindžiamas XML dokumento nuskaitymas.

UML (angl. *Unified Modeling Language*) – modeliavimo ir specifikacijų kūrimo kalba, skirta specifiuoti, atvaizduoti ir konstruoti objektinio programų dokumentus.

URL (angl. *Uniform Resource Locator*) – nuoroda į internetinį puslapį.

WSDL (angl. *Web Services Description Language*) – interneto paslaugas aprašanti kalba.

XML (angl. *Extensible Markup Language*) – duomenų struktūrų bei jų turinio aprašomoji kalba.

IVADAS

Pasaulyje ir Lietuvoje pastaraisiais metais informacinių technologijų (toliau – IT) sritis sparčiai plečiasi, nes išaugo poreikis verslo modernizavimui ir automatizavimui. IT rinkoje gausu informatikos specialistų, kurie nuolatos turi tobulinti savo gebėjimus prisitaikyti prie kasdien besikeičiančios inovacijų rinkos. Tačiau ne tik organizacijų verslo procesai turi būti automatizuojami ir tobulinami, o ir pačių bendrovių, kurios teikia šias paslaugas. Kad suspėti su klientų nustatomais terminais ir atitikti gausybę reikalavimų, IT specialistai nuolatos turi sukti galvas, kaip greičiau ir kokybiškiau teikti paslaugas, todėl yra labai svarbu, kad projekto komandos veiklos procesas būtų kuo labiau optimizuotas, o klientas kuo greičiau pamatytų norimus rezultatus.

Informacinės sistemos (toliau – IS) kūrimo procese neturi didelės vertės, nes klientas pamato tik galutinį produktą arba iš anksto jam pateikiamą būsimos sistemos prototipą. Būtent prototipas yra pagrindinė ir dažnai trūkstama kuriamos IS grandis tarp paslaugų teikėjo ir kliento, nes tik jis iš anksto padeda klientui susidaryti vaizdą apie būsimą sistemą, o kūrėją perspėja apie galimas problemas projekto pabaigoje.

Darbo aktualumas: kasmet sukuriami vis daugiau įvairaus pobūdžio IS, kurios kompiuterizuoja svarbius verslo procesus. Vis daugiau įmonių naudoja IS savo versle, nes jos palengvina darbuotojų užduočių atlikimą, o tai sumažina įmonės sąnaudas bei padidina efektyvumą. Bet koks IS kūrimo proceso optimizavimas šiai dienai sumažintų daugybės organizacijų kaštus.

Darbo tikslas: teoriškai ir praktiškai ištirti IS projektavimo metu kylančias problemas dėl vartotojo sąsajos prototipų kūrimo ir rasti jų sprendimą. Tad iškeliama tokie *uždaviniai:*

1. Išanalizuoti programinės įrangos projektavimo procesą, jo dalyvius, jų funkcijas bei prototipų taikymo svarbą.
2. Pasirinkti modeliavimo kalbą bei jos objektus, kurie geriausiai padėtų atvaizduoti modeliuojamą prototipą.
3. Išanalizuoti rinkoje esančius vartotojo sąsajos prototipams kurti skirtus įrankius ir atlikus jų palyginimą, rasti tinkamiausią generavimui realizuoti.
4. Pasinaudojant mokslinė literatūra ir tyrimo rezultatais, sukurti vartotojo sąsajos prototipų automatizuotą generavimo algoritmą.
5. Atlikti pasirinktos IS projektavimą pasinaudojant tyrimo rezultatais.

Mokslinis naujumas: rinkoje nepavyko rasti panašaus pobūdžio produktų, kurie nesudėtingai automatizuotų vartotojo sąsajos prototipų kūrimą ir atitiktų naudotojų poreikius.

Detalios analizės metu tik užsienio mokslinėje literatūroje rasta susijusių straipsnių, kurie iš įvairių pusių pažvelgia į vartotojo sąsajos prototipų generavimo problemą.

Suprojektuota būsto renovacijos IS, kurios poreikis šiuo metu Lietuvoje yra labai didelis. Dėl užsakovų negalėjimo susitarti su rangovais dėl kainos, stringa renovacijos plėtra Lietuvoje, todėl būtina imtis priemonių ir sukurti sistemą, leidžiančią automatizuoti medžiagų paskirstymą iš tiekėjų pagal nurodytą biudžetą.

Praktinė svarba: projektuojant pastatų apšiltinimo IS, praktiškai atliktas laiko ir pastangų vertinimas atsižvelgiant į rankiniu ir automatizuotu būdu modeliuojamo vartotojo sąsajos prototipo kūrimą.

Darbo aprobavimas: su darbu susijusia tema buvo parengtas straipsnis pavadinimu „Pažangesnis būdas įgyvendinti pikselių pagrįstą atvirkštinės inžinerijos vartotojo sąsajos struktūrą“ ir pristatytas 2013 m. gegužės 24 d. Kaune vykusioje konferencijoje „Inžinerinė ir kompiuterinė grafika“.

Darbo struktūra: įvadas, literatūros apžvalga ir analizė, rezultatai ir jų aptarimas, projektavimas, išvados ir pasiūlymai, literatūros sąrašas, priedai. Darbo apimtis - 76 puslapiai, kuriuose yra 19 lentelių, 39 paveikslėliai. Bibliografinį sąrašą sudaro 21 šaltinis.

1. LITERATŪROS APŽVALGA IR ANALIZĖ

Literatūros apžvalgos ir analizės tikslas – ištirti IS kūrimo procesą ir jo etapus, modeliavimo diagramų galimybes, prototipų svarbą analizės ir projektavimo etapuose, palyginti rinkoje esančius diagramų ir vartotojo sąsajos modeliavimo įrankius bei pasiūlyti vartotojo sąsajos prototipo generavimo algoritmą.

1.1. Informacinių sistemų kūrimas

Turbūt nereikia įrodinėti, kad IT atlieka pagrindinį vaidmenį organizacijos funkcionavime. Sugebantys laiku pasinaudoti žiniomis ir inovacijomis suteikia organizacijai konkurencingumo rinkoje. Sėkmingą organizacijos veiklą lemia: efektyvus klientų aptarnavimas, priimti geresni sprendimai, sisteminis požiūris į informacijos tvarkymą. Visa tai yra susiję su IS naudojimu.

Organizacijoje naudojamos ir plėtojamose IS daro įtaką: struktūrai, tikslams ir darbo planams, priimamiems sprendimams ir nulemia kasdienę elgseną, tačiau tenkindamos organizacijos poreikius ir joms tarnaudamos, sistemos privalo turėti tam tikrą formą, funkcionalumą bei turinį, kurį numato ir įgyvendina sistemų kūrėjai.

IS kūrimo metodika remiasi jos gyvavimo ciklu ir atitinkamomis ciklo stadijomis bei jų metų vykdomais darbais. Rekomenduojama, kad šią metodiką taikytų ir Lietuvoje sistemas kuriančios organizacijos. Pastebima, kad kuriant IS, pagrinde išryškunami tik sistemos kūrimo darbai, o darbų atlikimo dalyviai bei būsimos sistemos naudotojas paliekami antrame plane. IS plėtroje privalo dalyvauti visi organizacijos darbuotojai – ūkinės veiklos dalyviai, o ne tik sistemų kūrimo profesionalai. Tokią nuomonę pabrėžia ne vienas autorius savo mokslinės literatūros šaltinyje (Jeffrey *et. al.* 2007; Satzinger *et. al.* 2007).

Lietuvoje ir užsienyje yra daugybė IS kūrimo, palaikymo ir paslaugų teikimo įmonių, kurios ieško klientų ir bando jiems pasiūlyti geriausias sąlygas bei kompetenciją. Tačiau yra labai sudėtinga parodyti kiekvieno sistemos projekto kūrimo progresą, apimtį bei kokybę. Visą projekto gyvavimą sąlygoja reikalavimai, kurie dažniausiai kinta projekto eigoje. Jie kinta dėl to, kad buvo netinkamai suprasti ar interpretuoti, o dažnai ir pats klientas gauna tai, ko nesitikėjo.

Projekto sėkmė ir nesėkmė priklauso nuo daugybės kriterijų. Nepaisant tobulėjančių projektų valdymo technologijų, daugelis projektų baigiasi nesėkmingai. Čeponienė savo mokymų medžiagoje (2008) pateikė IT srityje atliktą projektų valdymo sėkmės tyrimą, kuris vadinamas CHAOS. Tyrimas pateikė tokius rezultatus:

- 1994 m. nustatyta, kad tik 16% visų nagrinėtų IT projektų įvykdyti laiku ir neviršijant biudžeto;
- 2003 m. sėkmingų IT projektų buvo 34%;
- 2004 m. – 28%, be to, buvo pastebėta, kad IT projektai brangsta.

Projekto sėkmė tiesiogiai priklauso nuo IT projekto dydžio:

- iš projektų, viršijančių 10 mln. dolerių, sėkmingi buvo tik 2%;
- iš projektų, kurių vertė buvo tarp 3 ir 10 mln. dolerių, sėkmingi – nuo 11% iki 23%;
- iš mažiau nei 3 mln. kainuojančių projektų, sėkmingi – nuo 33% iki 46%.

Apjungus keturių skirtingų tyrimų rezultatus (CHAOS, 2004 m.; Standish Group, 1994 m.; ESPITI, 1995 m.; Hallows, 1998 m.), surinktos pagrindinės IT projektų nesėkmių priežastys, kurios pateiktos lentelės pavidalu.

1 lentelė. IT projektų nesėkmių priežasčių sąrašas

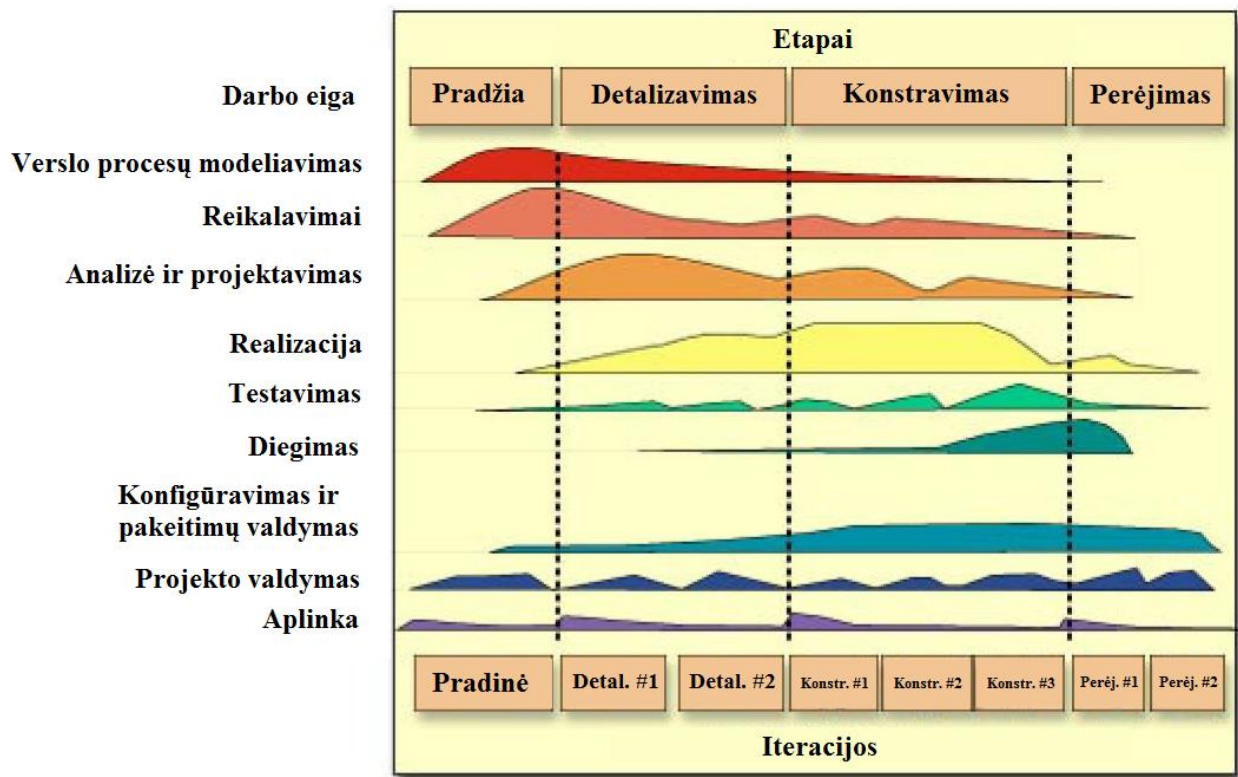
Eil. Nr.	Nesėkmės priežastis
1.	Per mažas galutinio naudotojo įtraukimas ir indėlis į projektą.
2.	Per mažas įmonės administracijos palaikymas.
3.	Prastas projekto valdymas ir/arba planavimas bei metodikos taikymas.
4.	Problemos susijusios su reikalavimais, jų pilnumu, specifikavimu, valdymu, kokybe bei pokyčiais projekto eigoje.
5.	Prastas pradinis projekto apibrėžimas, apimties valdymas, nenumatyti apimties pasikeitimai.

Apibendrinant priežastis, galima daryti išvadą, kad yra svarbu kuo daugiau dėmesio skirti galutiniam sistemos naudotojui, kuris galėtų teikti pastabas kuriamam produktui ir įdėti savo indėlį į projektą. Taip pat labai svarbu projekto kritinėje grandinėje turėti kuo didesnius laiko buferius, kad atsitikus nenumatytiems apimties pasikeitimams, būtų galima išvengti didelių projekto vėlavimų.

1.2. Projektavimo svarba projekto vykdymo etapuose

Projekto gyvavimo cikle IS analizė ir projektavimas užtrunka beveik visą produkto gyvavimo ciklą. Kad pamatyti visų projekto iteracijų, etapų ir darbo eigos sąveiką projekto

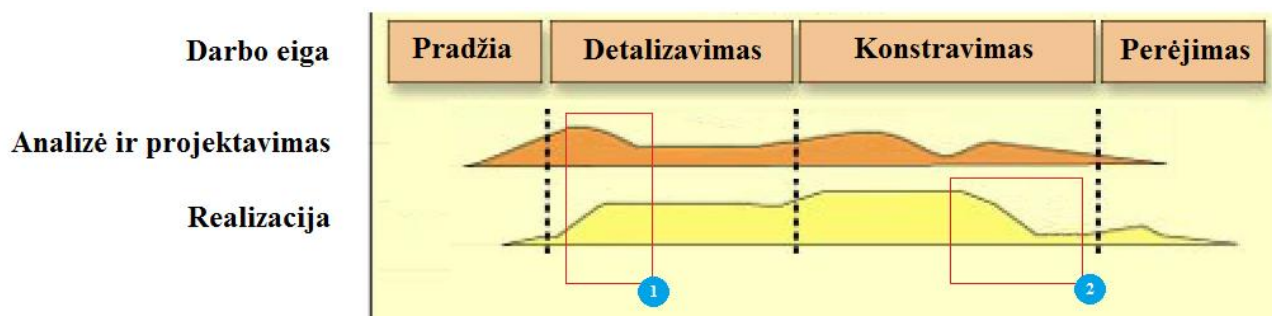
gyvavimo cikle, pasinaudota kartotinio programinės įrangos kūrimo metodika (RUP, angl. *Rational Unified Process*), sukurta įmonės *Rational Software*, nuo 2003 m. priklausančios IBM. Dinaminė RUP struktūra pateikta žemiau esančiame paveikslėlyje (žr. 1 pav.).



1 pav. Dinaminė RUP struktūra

Iš RUP struktūros matoma, kad analizė ir projektavimas detalizavimo etape užima labai daug laiko ir lyginant su kitais darbais, yra vienas svarbiausių projekto pradžioje. Būtent šioje vietoje iškilusios problemos gali stipriai paveikti projekto užbaigimo laiką.

Jei analizės ir projektavimo dalyje esančios didelės laiko sąnaudos būtų kaip nors optimizuotos, lygiagrečiai einantys realizacijos darbai galėtų būti pradėti daug anksčiau, o tai padėtų užtikrinti projekto įgyvendinimą laiku, sutaupytų nemažai laiko ir sąnaudų. Žemiau pateiktame paveikslėlyje imituotas analizės ir projektavimo darbų optimizavimas, po kurio realizacijos pradžia ženkliai ankstesnė (žr. 2 pav.).



2 pav. Dinaminė RUP struktūra optimizavus analizės ir projektavimo darbus ir paankstinus realizacijos laiką

Kadangi realizacija pradedama anksčiau, didelė tikimybė, kad projekto pabaigoje realizacijai skirtas laikas sumažės (žr. 2 pav.). Dėl to bus galima anksčiau pradėti vykdyti testavimą, diegimą bei kitus darbus, kurie priklausomi vienas nuo kito. Kuo anksčiau pradedami darbai, tuo anksčiau klientas pamato rezultatą ir gali teikti jam pastabas. Pats projektas įgauna tam tikrą laiko buferį, kuris užtikrina, kad atsiradus planuotai ar neplanuotai projekto rizikai, pats projekto užbaigimo laikas nepasikeis.

1.3. Informacinių sistemų modeliavimas

Norint sukurti teisingą ir suprantamą IS, iš pradžių reikia ją sumodeliuoti. Nuo senų laikų produktų kūrime buvo braižomi ir modeliuojami eskizai, maketai bei kiti brėžiniai. Joks didelis tiltas ar pastatas nebuvo pastatytas iš anksto jo nesuprojektavus jo architektūrinio brėžinio. To paties reikia ir sistemoms, kuriomis norima sukurti kokį nors funkcionalumą, padėsiantį žmonėms atlikti mechaninį darbą ar kt. Tačiau modeliavime, kaip ir kitose srityse, didžiausia problema išlieka tarpusavio supratimas. Kad brėžinys ar diagrama būtų suprantama visiems, reikia standartizuoto objektų sąrašo, kuris visiems būtų suprantamas ir aiškus.

1997 m. objektų valdymo grupė (OMG, angl. *Object Management Group*) paskelbė vieningą modeliavimo kalbą (UML, angl. *Unified Modeling Language*) kaip standartą, ši populiari modeliavimo ir specifikacijų kūrimo kalba padarė didelį poveikį IS specifikavimui, konstravimui ir projektavimui. Pačioje pradžioje UML buvo naudojama modeliuoti IS, tačiau ją taip pat naudoja ir verslo analitikai, kurie modeliuoja verslo procesus. Ši kalba tinkama apibrėžti verslo struktūrinius, elgsenos ir taisyklių aspektus, o jos naudojimas IS ir verslo modeliavimui gali užtikrinti dokumentacijos stabilumą, vientisumą ir padėti lengviau susikalbėti tarpusavyje projektuojant ir kuriant sistemas (Eriksson *et. al.* 2000).

OMG specifikacijoje nurodoma: „Vieninga modeliavimo kalba (UML) yra grafinė kalba, kuri skirta vizualizuoti, specifikuoti, konstruoti ir dokumentuoti programinę įrangos sistemų artefaktus. UML siūlo standartizuotą būdą kurti sistemos planus, įskaitant ir

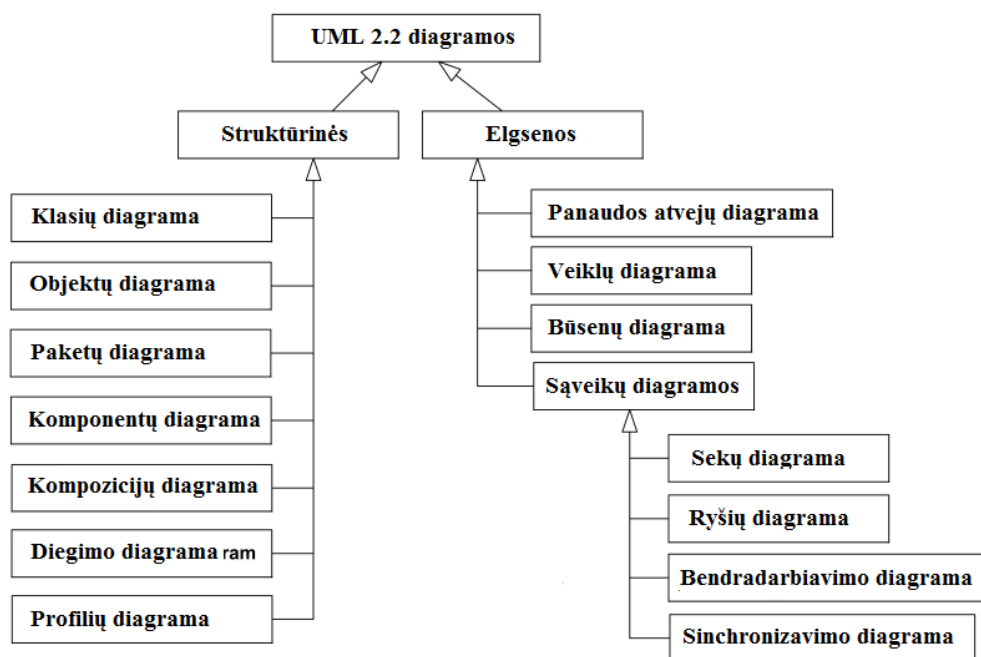
konceptualius dalykus, tokius kaip: veiklos procesus ir sistemos funkcijas, taip pat ir konkrečius dalykus, tokius kaip: programavimo kalbos išrašus, duomenų bazių schemas ir daugkartinio naudojimo programinės įrangos komponentus.“(OMG 2012).

1.3.1. Vieninga modeliavimo kalba

UML – tai vieninga modeliavimo kalba, kuri siūlo standartizuotus informacinės sistemos paveikslėlius, kurie padeda kūrėjų komandai greitai suprasti informacinės sistemos funkcionalumą. Šios modeliavimo kalbos sintaksiniai ženklai įvairioms diagramų rūšims leidžia atvaizduoti visus pagrindinius sistemos vaizdus. Tai standartas, kuris naudojamas informacinių sistemų specifikavimui, projektavimui, atvaizdavimui (OMG 2012).

UML pirmą kartą buvo panaudota 1997 m., kai skirtingų primityvių objektiškai orientuotų modelių žymėjimo būdų: OMT (autorius *James Rumbaugh*), OOSE (autorius *Ivar Jacobson*) ir Booch (autorius *Grady Booch*) kūrėjai susijungė į vieną bendrą modeliavimo kalbą. Šiuo metu UML prižiūri OMG, kuri yra objektinių standartų organizacija (Wikipedia UML 2013).

UML 2.0 versijoje apibrėžiama daugiau nei 10 diagramų, kurias naudojant galima sudaryti bendrą sistemos vaizdą su visais reikalingais jos aspektais. Sistemą galima projektuoti diagramomis taikant skirtingus lygmenis: 1) kaip vartotojas suvokia kuriamą sistemą, 2) kaip sistema bus kuriama, detalizuojant jos struktūrą, procesus ir pan. (Adler *et. al.* 1997).

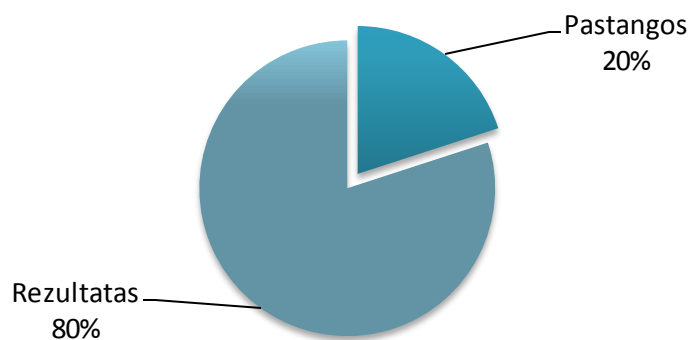


3 pav. UML diagramų medis

UML diagramos skirstomos į dvi šakas: viena jų vaizduoja sistemos struktūrą, kita – elgseną. Struktūrą vaizduojančios diagramos: klasių, objektų, komponentų ir kt. Elgseną vaizduojančių yra kur kas daugiau: panaudos atvejų, veiklų, sekų, būsenų, bendradarbiavimo, laiko ir kt. diagramos.

1.3.2. Diagramų tipai

UML modeliavimo kalba yra standartinė projektavimo priemonė, kuri visuotinai naudojama pasaulyje ir praktiškai neturi rimtesnių atitikmenų. Standartizuojant modeliavimo anotaciją, kuri būtų plačiai suprantama, vienareikšmiškai rekomenduojama pasirinkti UML. Aktualiausia ir plačiausiai naudojama UML 2.0 versija apibrėžia daug tipų diagramų, kurios leidžia specifiškai įvairius architektūros ir projektavimo aspektus. Rezultatui pasiekti nebūtina naudoti visų diagramų. Čia, kaip ir Pareto dėsnyje, galioja 20/80 taisyklė: 80% projekto užtenka 20% UML galimybių.

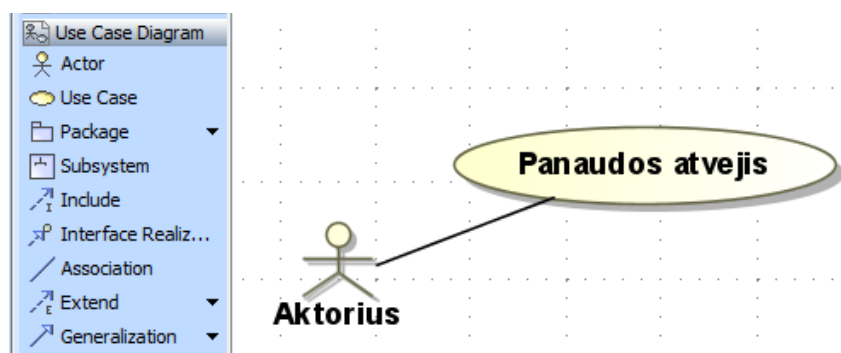


4 pav. Pareto dėsnis (20/80)

Šiame skyriuje pateikiamos pagrindinės UML diagramos, kurios skirtos IS projektavimui ir kūrimui. Detalizuojami kiekvienos diagramos svarbiausi elementai, be kurių diagrama nebūtų prasminga.

Panaudos atvejų diagrama (angl. *use case diagram*) – UML diagrama, aprašanti, ką projektuojama sistema gali atlikti, kartu įtraukiant ir išorinius arba vidinius sistemos veikėjus, vadinamus aktorais. Didelėms sistemoms rekomenduojama šią diagramą skirstyti į posistemius. Pagrindinis šios diagramos elementas – panaudos atvejis, aprašantis aibę panašių sąveikos scenarijų. Kiekvienas panaudos atvejis paprastai turi vieną pagrindinį ir keletą šalutinių scenarijų, kurie aprašomi dinaminėmis UML diagramomis (veiklų, sekų, bendradarbiavimo) (No Magic 2010).

Panaudos atvejų diagramos yra susijusios su sąveika tarp sistemoje vykdomų funkcijų ir aktorių (objektų, kurie tiesiogiai sąveikauja su sistema). Šios diagramos padeda įsivaizduoti bendrą sistemos kontekstą, sistemos ribas ir aktorius, kurie atlieka tam tikrus veiksmus. Dažnai aktoriai atitinka tam tikrus vaidmenis (angl. *roles*). Panaudos atvejų diagrama sudaroma visoms projektuojamoms sistemoms, nes yra lengviausiai suprantama net vartotojui, neturinčiam specialių projektavimo ar programavimo žinių.



5 pav. Panaudos atvejų diagramos braižymas naudojant modeliavimo objektus

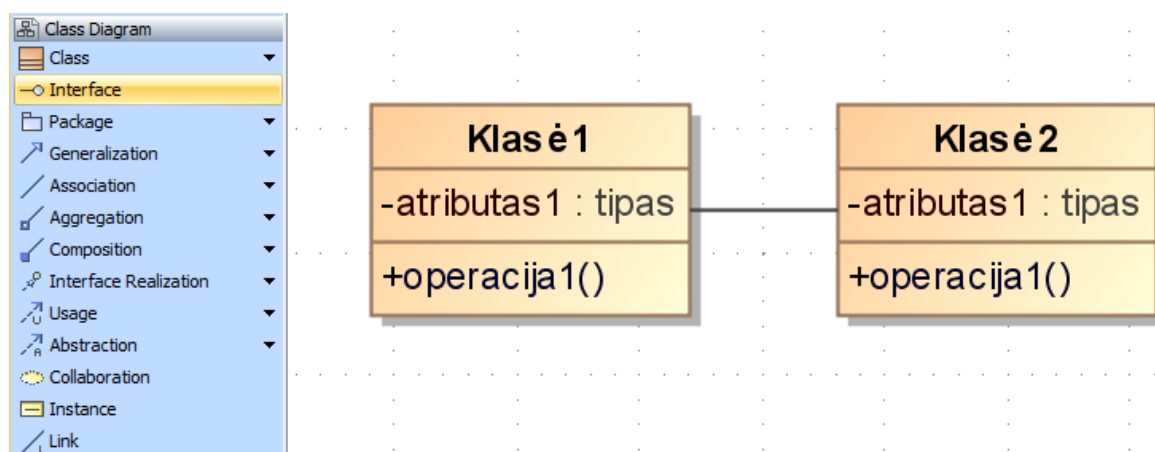
Kiekvienas diagramoje naudojamas aktorius gali būti susijęs su keletu panaudos atvejų, kurie nurodo, kokias funkcijas naudotojas atlieka sistemoje. 2 lentelėje pateiktas išsamesnis ir detalesnis panaudos atvejų diagramos elementų aprašymas.

2 lentelė. Panaudos atvejų diagramos elementų sąrašas

Elementas	Grafinis vaizdas	Aprašymas
Aktorius	 Actor	Sistemos naudotojas, dažniausiai tai būna tam tikra sistemos naudotojo rolė.
Panaudos atvejis	 Use Case	Panaudos atvejis. Funkcija, kurią atlieka aktorius.
Asociacija	 Association	Sąryšis tarp aktoriaus ir panaudos atvejo.
Apibendrinimas	 Generalization	Sąryšis, kuris žymi tam tikros rolės paveldėjimą.
Įterpimo ryšys	 Include	Sąryšis tarp dviejų panaudos atvejų, kurių pirmą privaloma įvykdyti, kad atlikti antrąjį.
Išplėtimo ryšys	 Extend	Sąryšis tarp dviejų panaudos atvejų, kurių vienas papildo kitą.
Sistemos rėmai	 System Boundary	Sistemos rėmai, kurie apibrėžia sistemos ribas. Aktoriai vaizduojami už sistemos rėmų.

Generuojant vartotojo sąsajos prototipą, panaudos atvejų diagrama būtų naudinga apibrėžiant sistemos langus. Pavyzdžiui, kiekvienam panaudos atvejui gali būti sukurtas atskiras vartotojo sąsajos langas, nes dažniausiai kiekviena funkcija reikalauja naujo lango, pavyzdžiui, redagavimo funkcija reikalauja redagavimo formos lango. Kiekviename lange turi būti atitinkami grafiniai elementai: tekstiniai laukai informacijai įvesti, sąrašai informacijai atvaizduoti ir t. t. Šiuos duomenis galima paimti iš klasių diagramų, kuriose esančios klasės užpildomos atributais.

Klasių diagrama (angl. *class diagram*) – tai statinės struktūros diagrama, aprašanti sistemos klases bei ryšius tarp jų. Ši diagrama identifikuoja visas sistemos klases bei apibrėžia kiekvienai klasei jos atributus, operacijas ir ryšius su kitomis klasėmis. Ryšiai apima paveldėjimą, asociaciją, apibendrinimą, klasifikaciją, kompoziciją, agregavimą bei kt. Klasių diagrama viena iš pagrindinių UML diagramų, nes vėliau iš jos kuriamos duomenų bazių diagramos, kuriose saugomi visi sistemos duomenys.

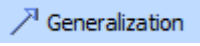
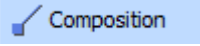
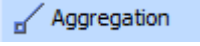


6 pav. Klasių diagramos braižymas naudojant modeliavimo objektus

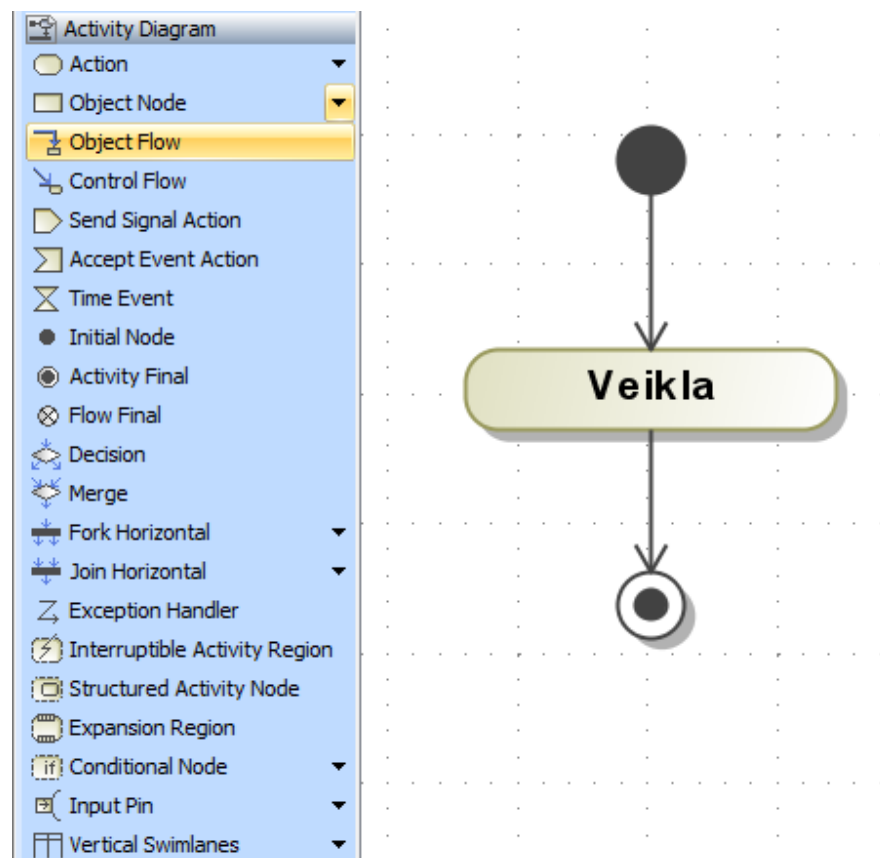
Kiekvienas klasių diagramos elementas su savo grafiniu vaizdu ir aprašymu pateiktas žemiau esančioje lentelėje. Lentelėje pateikti tik pagrindiniai elementai, kurie naudojami klasių diagramų braižymo metu.

3 lentelė. Klasių diagramos elementų sąrašas

Elementas	Grafinis vaizdas	Aprašymas
Klasė	 Class	Objekto klasė, kuri turi savo atributus ir operacijas. Kiekviena klasė turi unikalų pavadinimą.
Asociacija	 Association	Sąryšis tarp dviejų klasių. Šis ryšys sujungia du klasių objektus.
Klasifikacija	 Usage	Sąryšis tarp dviejų klasių, nurodant vienos klasės

Elementas	Grafinis vaizdas	Aprašymas
		priklausomybę kitai.
Apibendrinimas		Sąryšis tarp dviejų klasių, kuri viena paveldi kitos klasės savybes, o tuo pačiu turi savo savybes.
Kompozicija		Sąryšis tarp dviejų klasių, kuris nurodo ryši daug su vienu. Komponentas gali priklausyti tik vienam sudėtiniam objektui ir komponentinio objekto gyvavimas visiškai priklauso nuo sudėtinio objekto.
Agregavimas		Sąryšis tarp dviejų klasių, kuris nurodo, kad sudėtinis objektas susideda iš kitų objektų.

Veiklų diagrama (angl. *activity diagram*) modeliuoja dinaminę sistemos elgseną, t. y. vaizduojami sistemoje vykstantys veiksmai (No Magic 2010). Veiklos šioje diagramoje nurodo, kas atsitinka įvykdžius tam tikrą operaciją, aprašytą klasėse klasių diagramoje. Veiklų diagrama galima atvaizduoti ne tik klasėse esančias operacijas, bet ir kitus sistemoje vykstančius procesus.



7 pav. Veiklų diagramos braižymas naudojant modeliavimo objektus

4 lentelė. Veiklos diagramos elementų sąrašas

Elementas	Grafinis vaizdas	Aprašymas
Pradžia	 Initial Node	Veiklos pradžios taškas. Vienoje diagramoje gali būti tik vienas pradžios taškas.
Pabaiga	 Activity Final	Veiklos pabaigos taškai. Yra galimybė turėti kelis pabaigos taškus vienoje diagramoje.
Veikla	 Action	Veikla, kuri nurodo kokį nors veiksmą, kurį atlieka sistemos naudotojas arba pati sistema.
Srautas	 Control Flow	Srautas, kuris nurodo veiklų eiliškumą ir kryptį.
Sprendimas	 Decision	Sprendimo elementas, kuris su tam tikromis padalina srautą į kelis srautus.
Suliejimas	 Merge	Suliejimo elementas, kuris sujungia kelis srautus į vieną.
Lygiagretus išsišakojimas	 Fork Horizontal	Lygiagretus srauto išsišakojimas, kuris nurodo, jog srautas šakojasi nepriklausomai nuo sąlygų.
Lygiagretus suliejimas	 Join Horizontal	Lygiagretus srauto suliejimas, kuris nurodo, jog srautas susilieja nepriklausomai nuo sąlygų.
Plaukimo juostos	 Vertical Swimlanes	Veiklų „plaukimo juostos“, kurios nurodo tam tikrus veiklos vykdytojus ir jų vykdymo ribas. „Plaukimo juostos“ gali būti horizontalios arba vertikalios.

Veiklos diagrama, generuojant vartotojo sąsajos prototipą, galėtų nustatyti proceso eigą, pagal kurią būtų iškviečiami sugeneruoti sistemos langai. Kiekviena veiklos diagramos veikla atitiktų panaudos atvejį, t. y. langą, į kurį patenkama atlikus tam tikrą veiksmą.

Sekų diagrama (angl. *sequency diagram*) apibūdina dinaminę aktorių ir sistemos objektų sąveiką su tam tikrais veiksmais laike.

Bendradarbiavimo diagrama (angl. *communication diagram*) – iliustruoja ryšius ir sąveikas tarp programinės įrangos objektų.

Būsenų diagrama (angl. *state machine diagram*) apibūdina vieno sistemos objekto dinaminį elgesį kaip būsenų kaitą. Vaizduojamos svarbiausios verslo ar veiklos sistemos būsenos bei tų būsenų kitimas toje pačioje sistemoje. Būsenos diagrama žingsnis po žingsnio atvaizduoja verslo ir operatyvius sistemos komponentų srautus (No Magic 2010).

Komponentų diagrama (angl. *component structure diagram*) – aprašomi sistemoje naudojami komponentai ir jų bendradarbiavimas.

Yra dar daug kitų diagramų, tačiau šios yra pagrindinės norint sukurti būsimos IS projektą. Detaliau apie kiekvieną diagramą aprašyta 2010 m. *MagicDraw* naudotojo vadove.

1.4. Vartotojo sąsajos modeliavimas

Šiame skyriuje aprašomas modeliais pagrįstas vartotojo sąsajos plėtojimas (MDD, angl. *model-driven development of UIs*), kurį sudaro vartotojo sąsajos specifikavimas per vizualius modelius. Vizualūs modeliai apibūdina daug perspektyvų ir artefaktų vartotojo sąsajos kūrimui. MDD paremtas vizualiais įrankiais, kuriuos sistemos projektuotojas naudoja atvaizduoti daug modelių iš skirtingų perspektyvų. Programinio kodo generavimo mechanizmas tokiuose įrankiuose sugeneruoja programinį kodą pagal tai, ką projektuotojas sukūrė modelyje, o tai leidžia lengvai atlikti pakeitimus ir juos perkelti į programinį kodą (Almendros-Jimenez *et. al.* 2008).

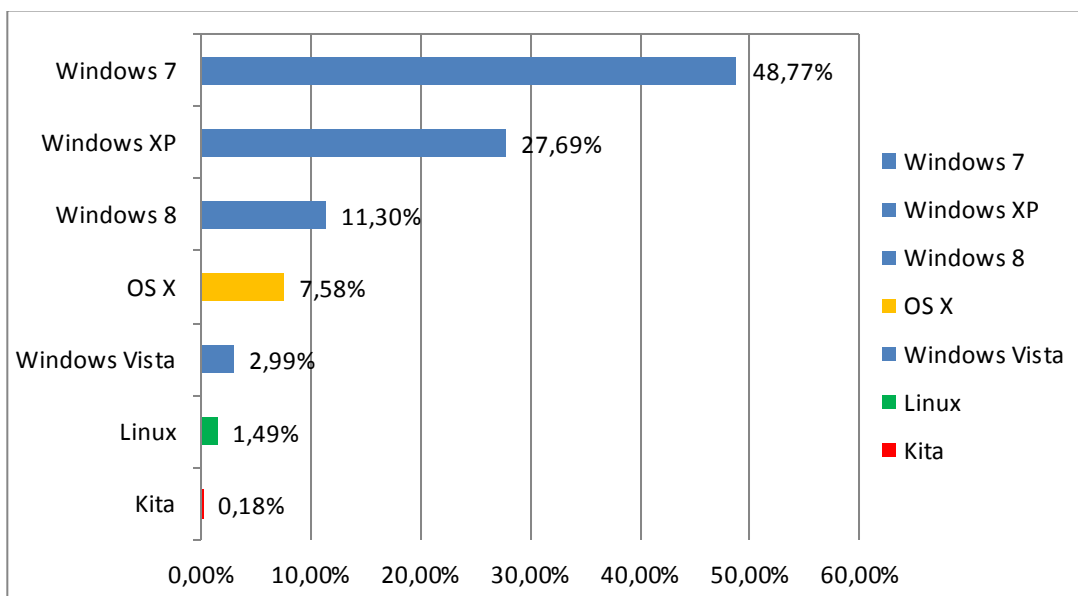
Modeliais pagrįstas vartotojo sąsajos plėtojimas turi tokius teigiamus aspektus (Pinheiro da Silva *et. al.* 2000):

- labiau abstraktus vartotojo sąsajos aprašymas nei tradiciniai metodai;
- vartotojo sąsajos sisteminis projektavimas ir aprašymas:
 - vartotojo sąsajos dizainas iš skirtingų abstrakcijos lygių;
 - didinamos modelių tobulinimo galimybės;
 - pakartotinis specifikacijos vartotojo modelių panaudojimas.
- automatizuojamas projektavimas ir įgyvendinimas iš vartotojo sąsajos.

Tačiau yra ir problemų, kurias reikia spręsti:

- modeliai kartais gali būti sunkiai suprantami (Green *et. al.* 1989): modelių notacija gali būti komplikuota ir semantika gali būti plačiai nežinoma, tačiau tam tikri įrankiai galėtų pagelbėti išspręsti šias abi problemas;
- funkcinių reikalavimų ir vartotojo sąsajos integracijos užduotis dar turi būti sprendžiama;
- nėra bendro nutarimo kokie modeliai turėtų būti naudojami vartotojo sąsajos modeliavimui (Almendros-Jimenez *et. al.* 2008).

Vartotojo sąsajos modeliavime reikia spręsti pagrindinę problemą, kaip apibrėžti vartotojo sąsajos modelius, kad jie būtų suprantami visiems? Kaip sprendimą šiai problemai galima taikyti *Windows* operacinės sistemos (toliau – OS) vartotojo sąsajos prototipą, kuris suprantamas didžiajai daliai informacinių sistemų bei kompiuterių naudotojams, nes pagal 2014 m. *Net Applications* įmonės, kuri atlieka rinkos tyrimus, interneto analizę ir kt., atliktą tyrimą (žr. 8 pav.), net 90,75% kompiuterių naudotojų naudoja *Windows* OS (Usage share of operating systems 2014).



8 pav. OS rinkos dalis nuo 2014 kovo pagal *Net Applications* atliktą tyrimą

Vertinant diagramos rezultatus, jokios kitos OS vartotojo sąsajos langai nėra taip gerai žinomi visame pasaulyje kaip *Windows*. Taikant standartinius langų, mygtukų ir kitus vartotojo sąsajos elementus iš *Microsoft* produkto, tikėtina, kad daugiau nei 90% būsimų sistemos naudotojų supras kas jiems norima parodyti.

1.5. Prototipai

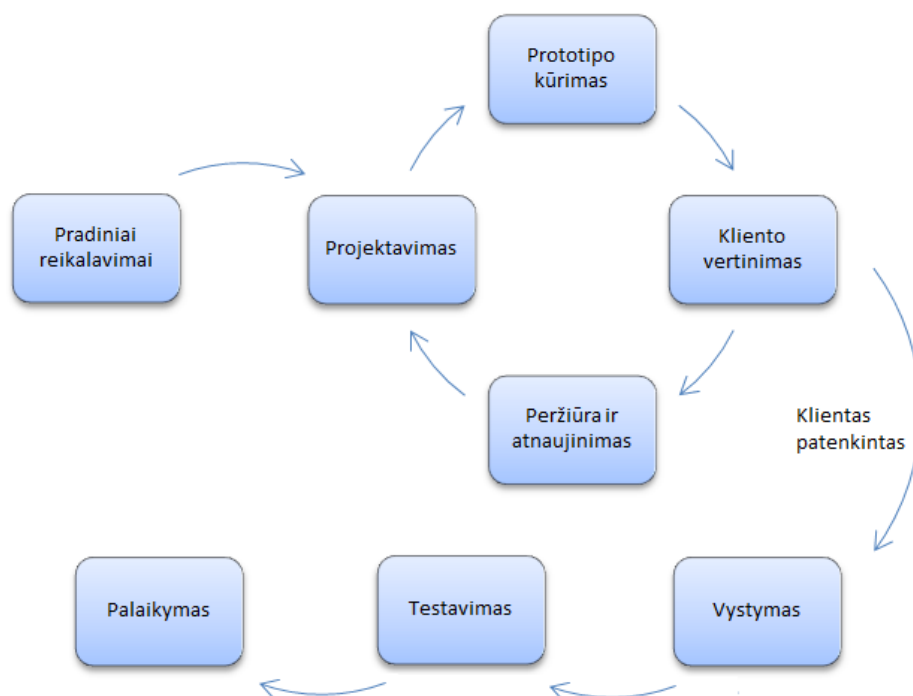
Net ir išsilavinę ir patyrę žmonės daro klaidų. Projekto gyvavimo cikle yra nemažai prielaidų ir blogų sprendimų, kuriuos projekto komanda kaupia ir iš to mokosi. Patyrusios komandos investuoja nemažai laiko, kad anksti atrastų savo blogas prielaidas, tam naudodamiesi vartotojo sąsajos prototipais. Derinant kartu su panaudojimo tyrimais, prototipai veda komandas teisinga linkme (Berkun 2000).

Reikia nesumašyti, kad sistemos beta versijos nėra prototipai, jos yra beveik visiškai užbaigtas inžinerinis rezultatas, kuris išleistas galimų klaidų radimui. Jei išleistoje beta versijoje randama kritinė klaida, vargu ar ji bus išmetama, nors gal tai ir būti pats geriausias sprendimas.

Kūrėjai, testuotojai ir kiti projekto komandos nariai jau investavo savo laiką, o spaudimas gyventi su blogu sprendimu yra labai didelis. Beta versijos padeda rasti klaidas ir trūkumus, bet jos nėra prototipai, kuriuos galima lengvai išmesti.

1.5.1. Prototipų kūrimo procesas

Prototipų modelis yra sistemos kūrimo metodas (SDM, angl. *Systems development method*), kuriame yra kuriamas prototipas (ankstyvas primityvus būsimos sistemos arba produkto pavyzdys), vėliau jis vystomas kaip sistema, testuojamas ir galiausiai palaikomas (žr. 9 pav.).



9 pav. Prototipo kūrimo proceso diagrama

Iš prototipo kūrimo proceso diagramos matoma, kad viskas prasideda nuo pradinių reikalavimų kuriamai sistemai, vėliau pradedamas projektavimo etapas, kurio eigoje sukuriamas prototipas. Prototipas demonstruojamas klientui ir jei yra pastabų, į jas atsižvelgiama ir atliekami atnaujinimai. Procesas sukasi cikle tol, kol klientas patenkintas ir priima būsimos sistemos ar produkto prototipą. Kai prototipas patvirtintas, vystoma pati sistema arba produktas, vėliau ištestuojamas ir, jei testavimo metu neatsirado klaidų, sistema palaikoma jos gyvavimo metu.

Šio metodo privalumai:

- modelis gali būti naudojamas tuomet, kai reikalavimai nėra visiškai aiškūs ar jų visai nėra;

- naudotojas gali eksperimentuoti su sistema ir tobulinti reikalavimus.

Šio metodo trūkumai:

- metodas būna naudojamas kaip pasiteisinimas nedokumentuoti reikalavimų ar projektavimo, nors jie ir aiškūs bei suprantami (Berkun 2000);
- daugiau trūkumų detalizuota 1.5.5 skyriuje.

1.5.2. Prototipų panaudojimas

Prototipų naudojimas reiškia idėjų tyrimą dar prieš tai, kai į jas investuojame. Visi patyrę meistrai ir inžinieriai kuria prototipus: Architektai kuria modelius iš popieriaus, kartono ar virtualios realybės priemonėmis. Aeronautikos inžinieriai naudoja vėjo tunelius. Tiltų statytojai sukuria jėgos spaudimo modelius. Filmų kūrėjai kuria siužetą. O programinės įrangos ir interneto dizaineriai kuria maketus, kurie nusako kaip naudotojai sąveikaus su jų sistemomis (Timah 2011).

Didžiausia priežastis dėl ko yra naudojami prototipai – tai sutaupyti laiko ir resursų. Prototipo vertė yra tokia pati kaip fasado. Kaip ir Holivudo filmavimo aikštelė, kurioje tik priekinės pastatų dalys yra pastatytos. Lyginant su visus produktu, prototipas yra lengvas ir pigus kūrinys. Minimaliomis investicijomis galima rasti panaudojimo ir dizaino problemų bei pakoreguoti vartotojo sąsają prieš tai, kai į galutinį projektą investuojami visi pinigai ir pasirenkamos sprendimą atitinkančios technologijos.

Nagrinėjant konkrečius projekto poreikius, galima rasti ir kitų priežasčių kurti prototipą, nei tik sutaupyti pinigų. Ar tikslas yra iširti naujos sąsajos modelį? Daryti pakeitimus jau egzistuojančiai sąsajos daliai? Išmėginti naują technologiją? Žinant kokius klausimus turi atsakyti kuriamas prototipas, bus sutaupyti ne tik pinigai, bet ir brangus laikas. Pagrindinės priežastys, dėl kurių yra kuriami prototipai:

1. **Koncepcijos įrodymas.** Jei yra nesutarimas dėl projekto vizijos, prototipas gali įrodyti, kad vizija yra įgyvendinama. Prototipas gali parodyti, kad idėja veikia, išreikšdama savo savybes vizualiai ir interaktyviuoju būdu ir/arba motyvuoti komandos narius galvoti apie kylančias problemas iš kitos perspektyvos.
2. **Projektavimo tyrimas.** Jei kuriami interaktyvūs dalykai, vienintelis būdas išsiaiškinti kaip bus naudojamas koks nors objektas, sukurti to objekto maketą ir bandyti juo pasinaudoti. Kartais maketas yra susietas su panaudojimo tyrimu, kai prototipo dalys gali būti vertinamos struktūriškai. Kartais tai tiesiog būdas aiškiai ir vizualiai paaiškinti kūrėjui, kaip kažkas turi veikti arba atrodyti. Daugeliu

atvejų prototipas kuriamas tam, kad dizaino kūrėjas galėtų eksperimentuoti, stengiantis suvokti kaip viskas galėtų veikti ir atrodyti. Bet kuris komandos narys gali naudotis prototipu, kad iš anksto rastų projektavimo problemų, nors dažniausiai paskirti tokių prototipų kūrėjai turi visą kompetenciją aptikti galimas problemas. Dizaino ir projektavimo tyrimas turėtų būti nukreiptas bandant spręsti konkrečias problemas, kurios buvo pastebėtos prototipo kūrimo metu.

3. **Techninis tyrimas.** Programuotojai tirdami kuriamą objektą susiduria su būtinybe priimti sprendimą, kokį kodavimo metodą taikyti. Naudojant HTML ar kitą specifinį kodavimo metodą, kiekvienoje technologijoje galima rasti skirtingus kompromisus. Kartais prototipas leidžia įsigilinti į tai, kokia technologija geriausiai veiks ir palaikys norimą vartotojo sąsajos ar interneto funkciją.

Kartais prototipai kuriami tam, kad apjungti šių priešasčių derinį. Jei projekto komanda planuoja ir suderina darbą tarpusavyje, prototipas gali būti kuriamas tiek dizainerio, tiek programuotojo, tiek projekto vadovo bendromis pastangomis. Tokiais atvejais labai svarbu apsibrėžti apimtis ir darbus, ko iš kiekvieno projekto komandos nario tikimasi padaryti. Svarbu, kad visiems būtų aiškūs tikslai, kurie yra pastatyti ant kortos, ir koks potencialus rezultatus iš viso to išeis.

1.5.3. Projekto kūrimo grupės dalyvių analizė

Prototipus gali kurti visi norintys, neatsižvelgiant į jų profesiją. Tai lengva ir paprasta padaryti, tiesiog turėti paveikslėlį, o tada panaudoti bet kokį interaktyvų įrankį, tokį kaip „PowerPoint“ arba „WordPress“, kad paversti jį interaktyviu. Tokie lengvi prototipai nėra labai praktiški, tačiau puikiai tinka pradiniam vaizdui susidaryti.

Formalesniems ir tinkamiems didesnėms komandoms prototipams, kūrėjas ar projekto vadovas dažnai bendradarbiauja su analitikais ir panaudojimo inžinieriais. Kartu jie generuoja idėjas kaip sukurti prototipą, kuris atspindėtų visus keliamus reikalavimus, o vėliau pereina prie jo kūrimo, kad pamatyti, kaip efektyviai yra sprendžiamos naudotojų problemos. Programuotojai gali prisidėti prie šio prototipo kūrimo jei turi laisvo laiko arba reikia jų, kaip ekspertų, patarimo, tačiau analitikai ir projektuotojai daro didžiausią darbą kuriant būsimo objekto prototipą.

Šiame skyriuje pateikiama informacinės sistemos projekto kūrimo grupės vartotojų analizė, kurie bus proceso dalyviai. Dažniausiai kūrimo proceso dalyvių sąrašą sudaro: klientas, projekto vadovas, reikalavimų inžinierius arba analitikas, projektuotojas, programuotojas, testuotojas, diegėjas bei daug kitų specialistų, kurie atsakingi už tam tikrus darbus kūrimo

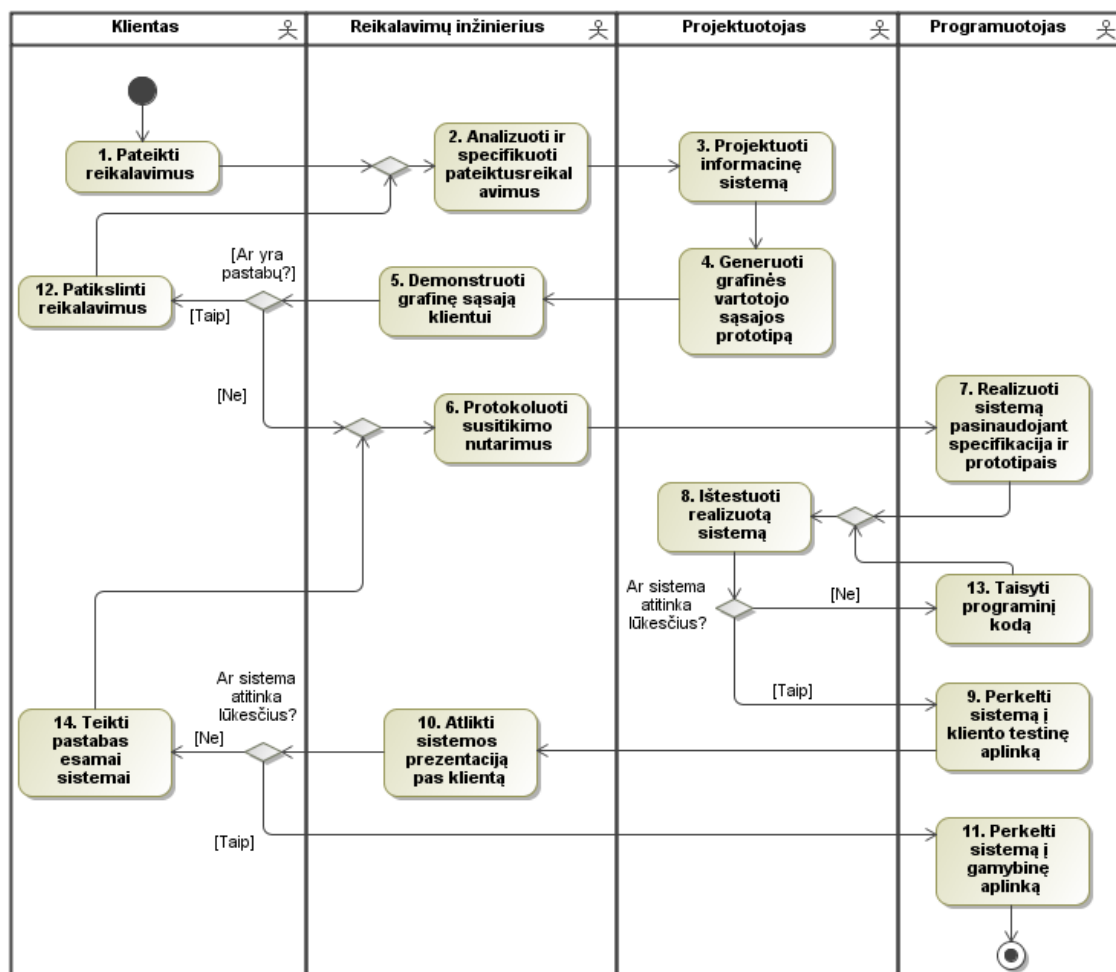
proceso eigoje. Tačiau parinkti 4 pagrindiniai, kurie gali atlikti projekto kūrimą ir be kitų pagalbos (žr. 5 lentelė). Ketvirtu dalyviu parinktas užsakovas, kuris yra neatsiejamas nuo projekto komandos, nes jis kelia reikalavimus kuriamai sistemai. Užsakovą šiuo atveju gali atspindėti klientas ar net projekto komandos įmonės direktorius, kuris iškelė reikalavimus komandai sukurti naują produktą.

5 lentelė. Proceso dalyvių sąrašas ir aprašymas

Proceso dalyvis	Aprašymas
Užsakovas	Užsakovais gali būti tiek fiziniai, tiek juridiniai asmenys, kurie užsako informacinę sistemą. Užsakovu gali būti ir pats projekto komandos narys ar organizacijos darbuotojas, jei sistema kuriama organizacijos viduje. Užsakovai retai išmano informacinių technologijų sferą, todėl svarbu jiems kuo anksčiau pateikti suprojektuotos sistemos vartotojo sąsajos prototipą.
Reikalavimų inžinierius (arba analitikas)	Reikalavimų inžinierius arba analitikas atsakingas už reikalavimų iš užsakovo surinkimą, jų analizavimą ir specifikavimą. Šis specialistas privalo būti stiprus reikalavimų inžinerijoje, veiklos procesų modeliavime ir analizavime. Norint išvengti skirtingų reikalavimų interpretavimų, patartina, kad prie projekto dirbtų vienas reikalavimų inžinierius, tačiau pasiskirsčius darbai, su projektu gali dirbti neribotas kiekis specialistų.
Projektuotojas (arba analitikas)	Projektuotojai arba analitikai suprojektuoja informacinę sistemą pagal užsakovo pateiktus ir reikalavimų inžinieriaus visiškai suformuluotus reikalavimus. Nubraižomos panaudos atvejų, veiklos, klasių, esybių bei kitos diagramos, kuriuos vėliau naudoja programuotojai, kurie programuoja informacinę sistemą. Jei analitikas pakankamai patyręs ir turi projektavimo žinių, tai projektuotojo komandoje gali ir nebūti.
Programuotojas	Programuotojai atsakingi už tikslų ir analizės specifikaciją atitinkantį informacinės sistemos suprogramavimą. Programuotojai turi būti geri specialistai, kad galėtų laiku suprogramuoti tinkamą naudoti informacinę sistemą. Programuotojų privalumas turėtų būti analitinis mąstymas, nes kuriant sistemą gali atsirasti anksčiau nenumatytų kliūčių, kurioms spręsti reikės priimti užtikrintus sprendimus.

Iš lentelės aprašo galima suprasti, kad kiekvienas proceso dalyvis turi tam tikras funkcijas, kurias turi atlikti projekto eigoje. Kai kurios funkcijos gali kartotis, kaip pavyzdžiui, reikalavimų inžinieriui gali tekti pakartotinai daryti analizę, jei projektavimo etape atsirado tam

tikrų pasikeitimų ar neaiškumų. Proceso dalyvių funkcijos informacinės sistemos kūrimo atvaizduotos veiklų diagramoje, kuri pateikta žemiau (žr. 10 pav.).



10 pav. Informacinės sistemos kūrimo procesas

Veiklų diagramoje sužymėtų proceso žingsnių detalesnis aprašymas pateiktas žemiau esančioje lentelėje (žr. 6 lentelę).

6 lentelė. Sistemos kūrimo proceso žingsniai ir jų aprašymas

Eil. Nr.	Proceso žingsnis	Aprašymas	Kas atlieka
1.	Pateikti reikalavimus	Užsakovas pateikia reikalavimus, pagal kuriuos reikės kurti informacinę sistemą.	Užsakovas
2.	Analizuoti ir specifiikuoti pateiktus reikalavimus	Reikalavimų inžinierius analizuoja pateiktus reikalavimus, juos specifikuoja analizės dokumente.	Reikalavimų inžinierius
3.	Projektuoti informacinę sistemą	Projektuotojas, naudodamasis reikalavimų inžinieriaus pateikta specifikacija, atlieka sistemos	Projektuotojas

Eil. Nr.	Proceso žingsnis	Aprašymas	Kas atlieka
		projektavimą.	
4.	Generuoti grafinės vartotojo sąsajos prototipą	Projektuotojas, pasinaudodamas grafinės vartotojo sąsajos prototipų generavimo įrankiu, sugeneruoja sistemos prototipą.	Projektuotojas
5.	Demonstruoti grafinę sąsają klientui	Reikalavimų inžinierius demonstruoja klientui projektuotojo sugeneruotą grafinę vartotojo sąsają.	Reikalavimų inžinierius
6.	Protokoluoti susitikimo nutarimus	Reikalavimų inžinierius protokoluoja susitikimo metu nuspręstus sistemos kūrimo reikalavimus. Kiekvienas kliento reikalavimo įgyvendinimo patvirtinimas leidžia pradėti tos dalies programavimo darbus.	Reikalavimų inžinierius
7.	Realizuoti sistemą panaudojant specifikaciją ir prototipais	Programuotojas suprogramuoja likusią sistemos dalį bei jos logiką. Programavimo metu galima pasinaudoti jau sugeneruota vartotojo sąsaja.	Programuotojas
8.	Ištestuoti realizuotą sistemą	Projektuotojas ištestuoja sistemą, kad joje neliktų klaidų. Taip pat paskutinį kartą patikrinama, ar sistema sukurta pagal reikalavimus.	Projektuotojas
9.	Perkelti sistemą į kliento testinę aplinką	Programuotojas perkelia sistemą į kliento testinę aplinką.	Programuotojas
10.	Atlikti sistemos prezentaciją pas klientą	Reikalavimų inžinierius atlieka sistemos prezentaciją klientui.	Reikalavimų inžinierius
11.	Perkelti sistemą į gamybinę aplinką	Programuotojas įdiegia sistemą pas klientą serveryje. Viskas sukonfigūruojama ir paruošiama naudojimui. Diegimą gali atlikti ir kiti projekto komandos nariai, jei tam turi kompetencijos.	Programuotojas
12.	Patikslinti reikalavimus	Užsakovas tikslina reikalavimus, jei reikalavimų inžinierius neteisingai suprato tam tikrus reikalavimus ar prototipas neatitinka kliento poreikių.	Užsakovas
13.	Taisyti programinį kodą	Programuotojas taiso programinį kodą.	Programuotojas

Eil. Nr.	Proceso žingsnis	Aprašymas	Kas atlieka
14.	Teikti pastabas esamai sistemai	Klientas teikia pastabas esamai sistemai, jei ji neatitinka jo lūkesčių.	Klientas

1.5.4. Prototipo kūrimo pradžia

Kada reikia pradėti kurti prototipą? Prototipas gali būti pradėtas kurti bet kuriuo projekto gyvavimo etapo metu. Dažniausiai jie yra kuriame projekto pradžioje, planavimo metu, prieš tai, kai programuotojai pradeda rašyti būsimos sistemos kodą. Šis laikas yra pats tinkamiausias atlikti tyrimą, o skirtas laikas užtikrina, kad vėliau bus išvengta didelių netikėtumų ir galimų išlaidų. Jei prototipą kuria programuotojai, o ne analitikai ar projektuotojai, labai svarbu, kad šis laikas būtų įskaičiuotas ir suplanuotas kaip kūrimo sąnaudos. Dažnai tokie darbai nebūna vertinami, o skaičiuojamos tik programavimui skirtos valandos.

Projektui įpusėjus, mažesni prototipai gali padėti išspręsti didelius kūrimo ir techninius klausimus. Greitai ir elementarus HTML maketas, kaip dialogo langas ar interneto puslapio maketas turi atrodyti ar elgtis, gali padėti atsakyti į kūrėjo klausimus arba padėti kitiems komandos nariams suvokti, ką turi būti norimas rezultatas. Tam tikrais atvejais, patyręs analitikas ar projektuotojas gali atvaizduoti norimą rezultatą *Microsoft JScript* programinės įrangos pagalba ir suderinti daug programavimo logikos, kurią kūrėjas turėtų apgalvoti.

Laikas, skirtas sukurti prototipą, gali skirtis milžiniškai, priklausomai nuo apimties ir norimo tikslumo, kaip galutinis rezultatas turi atrodyti. Neformalus prototipas gali užtrukti keletą vieno žmogaus darbo valandų, tačiau labiau organizuotam rezultatui gali prireikti kelių savičių visos projekto komandos pastangų (Berkun 2000).

1.5.5. Prototipo funkcionalumas

Ant kiek detalus ir funkcionalus turi būti prototipas? Kuriant prototipą, reikia kurti tik tiek dizaino, kiek iš tiesų bus naudinga jo turėti. Prototipe nesvarbu, kad yra mygtukai, kurie nesispaudžia ir neatlieka jokios funkcijos, arba tekstas, kuris niekada neatsinaujina. Iki tol, kol bus galima patirti ir patyrinti sąveiką su būsimu rezultatu, likusiam funkcionalumui galima naudoti „dūmus ir veidrodžius“, t. y. nesvarbu kokią funkciją atlieka tam tikras mygtukas, svarbu žinoti, kad jis ten yra ir jis padarys tai, ko iš jo tikimasi. Štai keletą priežasčių, kodėl reiktų atsargiai sutelkti savo pastangas:

- **Prototipo kūrimo išlaidos.** Kuriant prototipą yra siekiama sumažinti su jo kūrimu susijusias išlaidas. Prototipo kūrimo uždavinys - pasiekti minimalių investicijų prototipo kūrime, tačiau rezultatas turi atsakyti į visus kūrimo, projektavimo ir dizaino klausimus. Šioje vietoje panaudojimo tyrimai yra labai svarbūs, nes jie tiksliai nusako vartotojo sąsajos vietas, kurioms reikės skirti daugiausia darbo. Net ir be panaudojimo tyrimų, turėtų būti galima nustatyti kokios yra naudotojų problemos, kurias norima išspręsti, ar kokias užduotis norima pagerinti su kuriu prototipu.
- **Ribotas tarnavimo laikas.** Prototipui turėtų būti aiškiai apibrėžtas gyvavimo laikas. Ar jo galutinis tikslas yra demonstracinis pateikimas projekto komandai? Vykdomasis apžvalginis susirinkimas? Speciali apžvalga? Panaudojimo tyrimas? Įsitikinimas patiems, kad šis kūrinys išsprendžia naudotojų problemas? Kai tik patenkinami konkrečių tikslų poreikiai, prototipas turėtų būti atidėtas į šoną. Reikia suvokti, kad prototipe panaudotas kodas ar paveikslėliai bus užmiršti ir nebenaudojami. Gali būti išimčių, kai kodas ar paveikslėliai gyvuoja ir galutiniame produkte, tačiau reikia tikėtis, kad tai nebus šis minėtas atvejis. Kūrimo etape produktą reikės kurti nuo nulio, o prototipas bus naudojamas tik kaip pavyzdys.
- **Rizika išgašdinti komandą.** Prototipo demonstravimas būsimą produkto kūrėjams ir komandos nariams gali būti apgaulingas. Pernelyg sudėtingas ar įmantrus prototipas, demonstruojantis sudėtingus vaizdus ir animacijas, gali sukrėsti žmones. Reikia visada jausti kiek daug ir kiek toli norima pavaizduoti prototipe, kad komandos nariai suprastų teisingai ir neišsigastų būsimų sunkumų.

1.5.6. Prototipo apimčių nustatymas

Prieš pradėdant kurti prototipą, reikia apsvarstyti keletą dalykų, į kuriuos reikia sutelkti savo pastangas ir nustatyti apimtis. Štai keletas pavyzdžių:

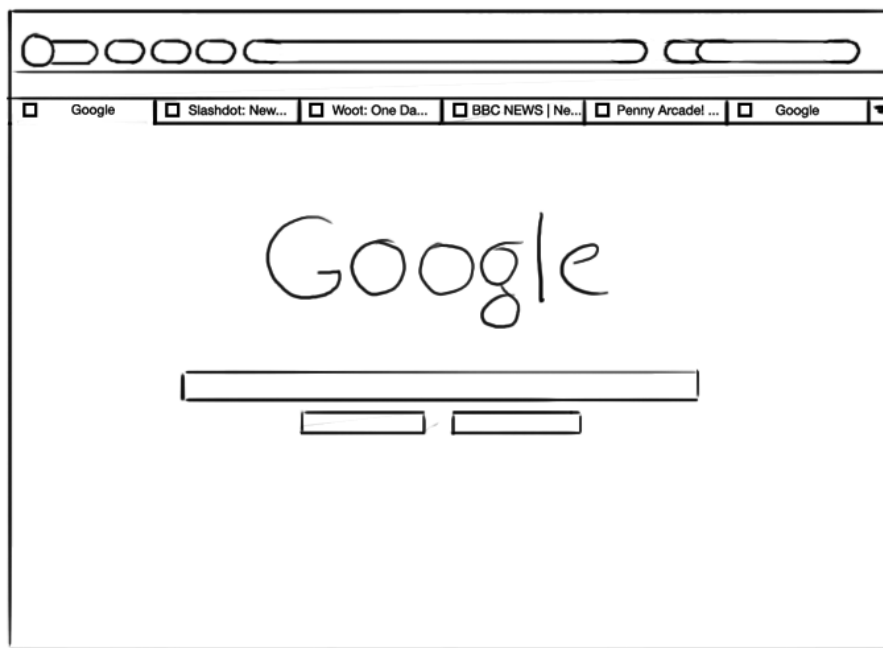
- **Kliento poreikiai.** Jei pradėdama nuo pagrindinių kliento problemų ir poreikių supratimo (poreikių analizę atlieka reikalavimų inžinierius), tuomet žinoma nuo kurių dalių reikia pradėti, kad pateisinti lūkesčius.
- **Panaudos atvejų tyrimas.** Jei prototipas kuriamas tam, kad sužinoti visus galimus panaudos atvejus, reikia viską aptarti su reikalavimų inžinieriumi, kokie bus pagrindiniai panaudos atvejai ir jų išplėtimai.

- **Komandos indėlis.** Pokalbis su pagrindiniais komandos kūrėjais gali atsakyti į daug svarbių klausimų, tokių kaip: ar įmanoma taip padaryti? Ar protinga taip daryti? Kokios gali būti pasekmės? Kartais galima iš komandos pareikalauti daugiau nei tai, ką jie minėjo kaip įmanomą variantą, tokiu būdu išbandyti savo komandą pasiruošimo ir žinių lygyje. Tačiau nereiktų to daryti su kiekviena prototipo dalimi. Yra labai plona linija tarp to, kad stumtelėti savo komandą į priekį, ir tarp to, kad išgąsdinti ją. Jei norima tik įkvėpti komandos narius, galima pasiūlyti kelis galimus variantus ir žiūrėti jų reakcijas.
- **Plotis ir gylis.** Didesnių prototipų kūrime yra svarstoma dėl jo funkcionalumo „pločio ir gylio“. Ar daryti visų funkcijų tik po truputį, bet daug, ar paimti vieną funkciją, tačiau ją padaryti visiškai iki smulkmenų. Jei pasielgiama neatsargiai, bandoma pasiekti abiejų dalykų vienu metu, tuomet gaunamas didžiulis, griozdiškas prototipas, kurį sunku keisti ar išmesti. Praktikoje pasitvirtina sprendimas kurti prototipą į „plotį“, nes tokiu būdu klientui galima parodyti daug funkcionalumo per trumpą laiką.
- **Eskizai prieš vizualų dizainą.** Priklausomai nuo to, kam prototipas bus demonstruojamas, reikia nuspręsti kokią vizualią kokybę prototipas turės. Eskizų arba nubraižytų blokelių gali pakakti žmonėms, su kuriais dirbama, arba tiems, kam nereikia padaryti gero įspūdžio, bandyt įtikti arba suprantantiems kam prototipas skirtas ir ką jis vaizduoja. Tačiau pristatant prototipą mažiau patyrusiam klientui, tam tikram vadovui, daugiau techninei auditorijai, turėtų būti demonstruojamas labiau estetiškas ir sudėtingesnis prototipas, bet tai kainuoja daugiau laiko ir pastangų.
- **IG: investicijų grąža.** Gali būti įvairiausių formų prototipų (panaudojimo, techninių, veiklos, estetinių). Kuo prototipas tikslesnis, tuo labiau atitinka būsimą realybę. O tokie prototipai nekelia nereikalingų klausimų ir kalba patys už save. Kuo mažiau paaiškinimų reikia („Oi, galutiniame produkte taip tikrai nebus.“, „Stilius bus daug gražesnis nei čia.“, „Šitas mygtukas darys tikrai ne tai.“), tuo prototipas laikomas tikslesniu ir atitinkančiu realybę. Žinoma, detalumas ir tikslumas kainuoja laiko, pastangų ir pinigų, todėl reikia nuspręsti iš anksto, koks yra viso šio kūrimo tikslas, ir investuoti tik tiek, kiek reikia.

Investicijų grąža ir prototipo tikslumas yra labai surišti veiksniai. Sąveikos projektuotojai ir panaudojimo projektuotojai dažnai išskiria prototipų kūrime sąvokas didelis ir mažas tikslumas (angl. *high and low fidelity*). Tikslumas reiškia, kaip prototipas reaguoja.

Didelio tikslumo prototipai yra „veikiančios“ programos – paspaudi mygtuką ir kažkas įvyksta. Sąvoka „veikianti“ naudojama todėl, kad pats pasakymas „didelio tikslumo“ nusako, kad kažkas turi veikti. Tai nereiškia, kad tai tikrai atlieka tikrąsias funkcijas, tai tik „dūmų ir veidrodžių“ priedanga. Netikri paveikslėliai, sąrašai ir mygtukų funkcijos yra tik žaidimas. Tikslas yra suteikti realistinį būsimos sistemos skonį bei kaip ji atrodys ir jausis.

Mažo tikslumo prototipai, priešingai, nėra veikiantys. Mažo tikslumo prototipų pavyzdžiu gali būti eskizas ar paprastas paveikslėlis. Radikaliausios formos mažo tikslumo prototipas yra lapas popieriaus su nupieštu paveikslėliu (žr. 11 pav.), o kas yra spausdinama ant popieriaus, žinoma, kad mygtukai neveiks ir kita neveiks, bent jau kol kas (Simon 2005).



11 pav. Mažo tikslumo prototipo pavyzdys

1.6. Modeliavimo įrankių analizė

Rinkoje yra daug organizacijų, kurios siūlo įvairius modeliavimo įrankius, galinčius sukurti objektines diagramas, generuoti iš jų programinį kodą bei atvirkščiai. Tačiau kiekvienas įrankis yra skirtingas, turintis savo privalumus ir trūkumus. Vieni jų skiriasi palaikomų platformų ar operacinių sistemų, skirtingų metodologijų ar programavimo kalbų, į kurias generuojamas programinis kodas, specifika. Dauguma įrankių, kurie teikia programinio kodo generavimo funkciją, turi vienos ar kelių programavimo kalbų sąsajas bei jų interpretavimo metodus, kurie atvaizduoja diagramų elementus į programinio kodo elementus. Norint išsirinkti tinkamiausią įrankį, reikia atlikti jų palyginimą ir išsiaiškinti kiekvieno iš jų privalumus, programinio kodo generavimo principus bei panaudojimo tikslą.

Svarbu, kad įrankis vykdytų kodo projektavimą aukštesniame lygmenyje, nes visi elementai, kurie sudaromi įrankio pagalba, vėliau taps tam tikro kodo elementais. Taigi, pats kodo generavimas iš diagramų turi išsaugoti elementų savybes, ryšius tarp jų bei vėliau suteikti jiems bendrą formą. Po kodo generavimo visi elementai turi būti atvaizduojami nuoseklia struktūra, o tai yra labai svarbu, nes diagramose, o ypač UML diagramose, kiekvienas ryšys, hierarchija ir išdėstymas nulemia sistemos projektavimo tikslumą. Kuo tiksliau įrankis perkels diagramas į programinį kodą, tuo mažiau tikrinimo vėliau reikės daryti programuotojui arba projektuotojui, kuris nori pasinaudoti tuo programiniu kodu.

Būtų sudėtinga išsaugoti diagramų dizainą programiniame kode, nes kiekviena programavimo kalba interpretuoja dizainą skirtingai. Šiuo atveju užtektų nusistatyti standartus, kuriais būtų naudojamosi generuojant kodą, kad vėliau visos programavimo kalbos suprastų vienodai. Tam geriausiai tinka duomenų struktūrų bei jų turinio aprašomoji kalba (XML, angl. *Extensible Markup Language*). Ši programinio kodo aprašomoji kalba naudojama kaip standartas, kurį supranta visos kitos programavimo kalbos.

Šiame skyriuje trumpai apžvelgiami keturi rinkoje populiarūs UML diagramų bei vartotojo sąsajų braižymo įrankiai. Kad atlikti lyginamąją analizę visų keturių įrankių, buvo suformuota programinės įrangos vertinimo kriterijų sąrašas, kuris aktualus šio darbo kontekste. Žemiau pateikta lentelė su kiekvieno iš kriterijų aprašymu.

7 lentelė. Programinės įrangos vertinimo kriterijų sąrašas

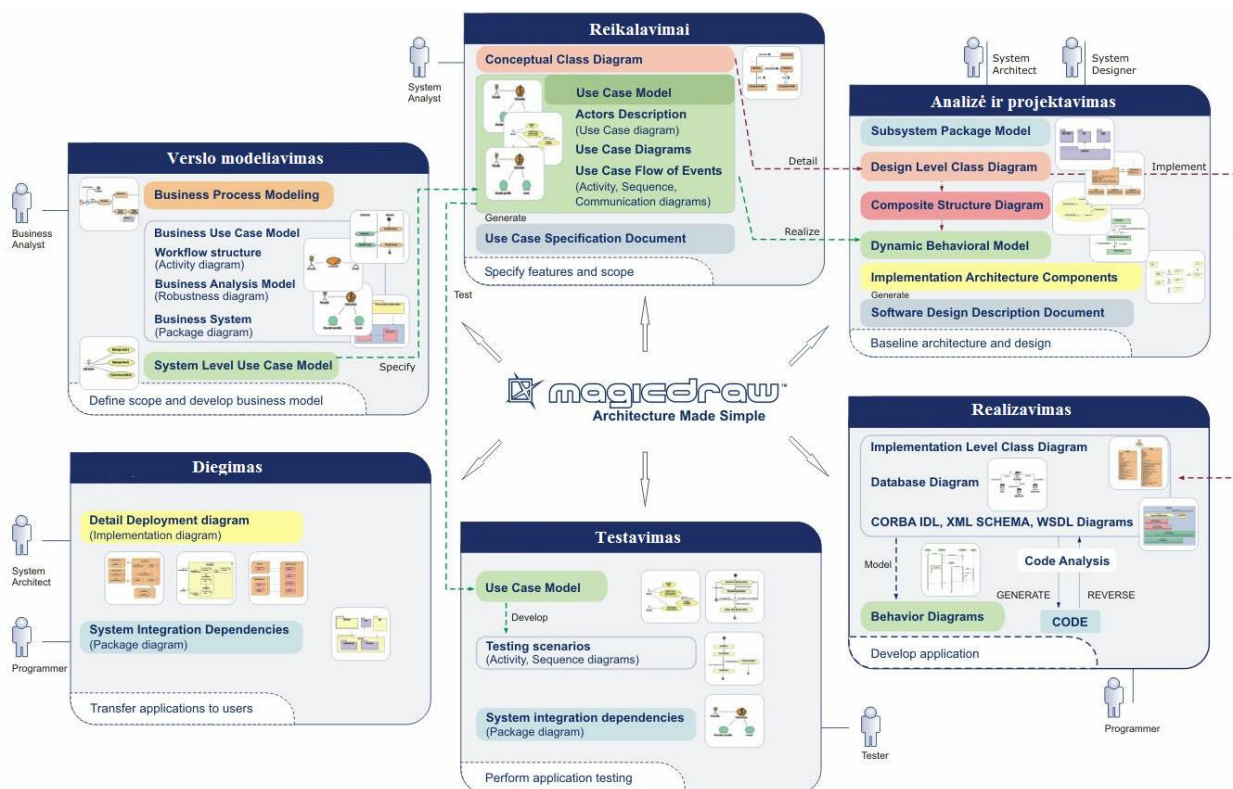
Kriterijus	Aprašymas
UML 1.4 versijos palaikymas	Galimybė naudoti UML 1.4 diagramas
UML 2.0 versijos palaikymas	Galimybė naudoti UML 2.0 diagramas
<i>Robustness</i> diagramos	Galimybė naudoti <i>Robustness</i> diagramas praplečiant UML

BPMN palaikymas	Galimybė naudoti veiklos procesų modeliavimo diagramas (BPMN, angl. <i>Business Process Modeling Notation</i>) – standartinius verslo procesų modelius	
Vartotojo sąsajos braižymas	Galimybė nubraižyti vartotojo sąsajos elementus rankiniu būdu.	
Vartotojo sąsajos generavimas iš modeliavimo diagramų	Galimybė sugeneruoti vartotojo sąsajos prototipą iš modeliavimo diagramų.	
Programavimo kalba	Tiesioginė kodo inžinerija	Atvirkštinė kodo inžinerija
.NET palaikymas	Galimybė generuoti kodą iš diagramų	Galimybė generuoti diagramas iš kodo
Java palaikymas		
C++ palaikymas		
Visual Basic palaikymas		
Verslo procesų įrankių palaikymas	Integracija su kitais verslo modelių įrankiais	

1.6.1. MagicDraw UML

No Magic Europe (UAB "Baltijos programinė įranga") - pirmaujanti Lietuvoje IT įmonė, kurios sprendimais ir patarimais pasitiki didžiausi Lietuvos bankai, telekomunikacijų bendrovės ir didžiosios Lietuvos IT įmonės. *No Magic Europe* atstovauja JAV kompanijai *No Magic Europe* šalyse. Pagrindiniai įmonės tikslai yra plėtoti veiklą programinės įrangos kūrimo, IT konsultacijų ir mokymų srityje Lietuvos ir pasaulio rinkose, patenkinti klientų poreikius ir lūkesčius, gerinančius verslumą (*No Magic* 2014).

Kuriant programinę įrangą (toliau – PĮ), labai svarbu suprantamai ir išsamiai apibrėžti vartotojų poreikius, sukurti efektyvų programinės įrangos modelį, parengti dokumentaciją. Naudojantis *No Magic Europe* sukurtu ir plėtojamu produktu *MagicDraw UML* galima lengvai atlikti šiuos uždavinius ir sutaupyti laiko bei lėšų kituose programinės įrangos kūrimo etapuose.



12 pav. MagicDraw PĮ galimybės kuriant informacines sistemas (No Magic 2014)

MagicDraw UML atitinka naujausius Java bei UML technologijų standartus, turi vieną iš patikimiausių išeities kodų inžinerijos mechanizmų Java, C#, C++ ir CORBA IDL programavimo kalboms bei gali vykdyti šių kalbų kodo atvirkštinę inžineriją, duomenų bazių schemų atvirkštinę inžineriją, kodo bei duomenų bazių schemų generavimą. *MagicDraw UML* panaudoja „roundtrip“ technologiją, leidžiančią keisti tiek objektinį modelį, tiek programos kodą bet koku eiliškumu, juos nuolat sinchronizuojant. *MagicDraw UML* yra vienas iš nedaugelio rinkoje esančių paketų, leidžiančių greičiau nubraižyti UML diagramas bei turintis UML diagramų semantinio teisingumo tikrinimo ir modelio tikrinimo mechanizmą. *MagicDraw UML Teamwork Server* programinė įranga turi komandinio darbo galimybę, leisdamą daugeliui programinės įrangos kūrimo inžinierių dirbti su tuo pačiu objektinio modeliu vienu metu.

MagicDraw UML savybių analizės metu buvo užpildyta įrankio galimybių lentelė (žr. 8 lentelę). Iš lentelės matyti, kad šis įrankis atitinka beveik visus keliamus UML bei programinio kodo inžinerijos reikalavimus. Toks įrankis puikiai tenkina naujausius rinkos reikalavimus ir verslo poreikius.

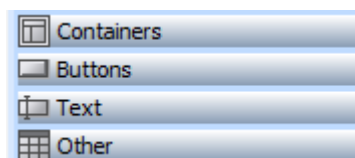
8 lentelė. *MagicDraw UML* įrankio keliamų reikalavimų ir galimybių lentelė

Kriterijus	Aprašymas
UML 1.4 versijos palaikymas	+
UML 2.0 versijos palaikymas	+

Robustness diagramos	+	
BPMN palaikymas	+	
Vartotojo sąsajos braižymas	+	
Vartotojo sąsajos generavimas iš modeliavimo diagramų	-	
Programavimo kalba	Tiesioginė kodo inžinerija	Atvirkštinė kodo inžinerija
.NET palaikymas	+	+
Java palaikymas	+	+
C++ palaikymas	+	+
Visual Basic palaikymas	-	-
Verslo procesų įrankių palaikymas	+	

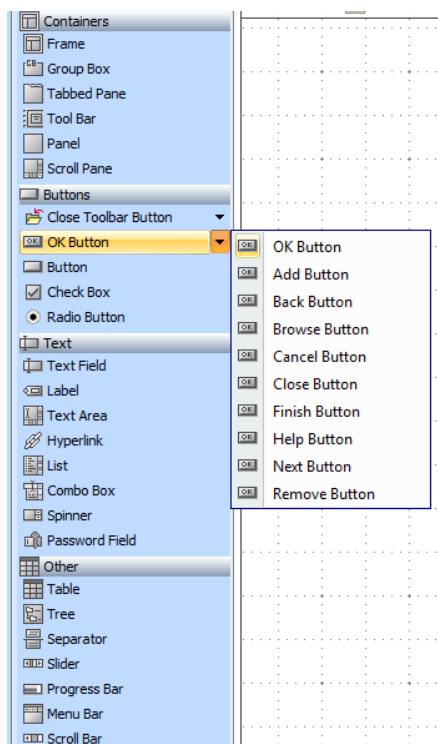
MagicDraw UML visiškai palaiko UML 2 standartus, todėl šis įrankis leidžia kurti visas reikiamas UML diagramas. Nuo 16.6 PĮ versijos leidžiama modeliuoti vartotojo sąsajos prototipus rankiniu būdu, t. y. suteikiama galimybė pasirinkti iš gausaus elementų sąrašo ir kurti GUI langus. Tam yra sukurta atskira diagramos rūšis pavadinimu „Vartotojo sąsajos modeliavimo diagrama“ (angl. *User Interface Modeling Diagram*). Kad būtų paprasčiau rasti norimą elementą, grafiniai elementai suskirstyti į 4 didesnes grupes (žr. 13 pav.):

- blokai;
- mygtukai;
- tekstiniai laukai;
- kita.



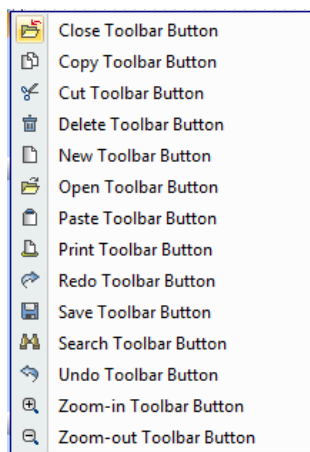
13 pav. GUI kūrimo elementų grupės

Išskleidę visas grupes, gautume gausų pasirinkimą iš grafinių elementų sąrašo (žr. 14 pav.). Pavyzdžiui, jei išskleistume blokų grupę, tai ten galėtume pasirinkti pagrindinio lango rėmelį, langus su slankiosiomis juostomis, ekrano pultą, įrankių juostą ir kt.



14 pav. GUI kūrimo elementų visas sąrašas

Jei reikia kurti mygtukus, tai yra gausus pasirinkimas iš įvairių mygtukų, pvz.: pridėjimo, atšaukimo, uždarymo, pasirinkimo, grįžimo ir kiti mygtukai. Taip pat standartinis žymimasis laukelis, akutė, paprasti mygtukai bei kiti Windows operacinės sistemos įrankių juostos mygtukai (žr. 15 pav.), tokie kaip: uždaryti, kopijuoti, iškirpti, ištrinti, atidaryti ir kt.

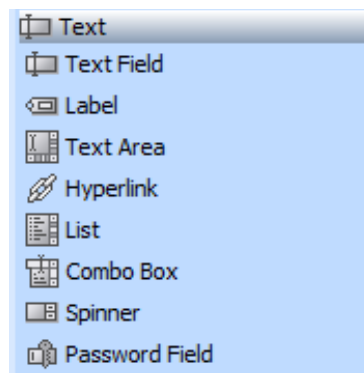


15 pav. Windows operacinės sistemos mygtukų sąrašas

Visi šie mygtukai leidžia sukurti vartotojo sąsajos prototipą, kuris suprantamas didžiajai daliai informacinių sistemų bei kompiuterių naudotojams.

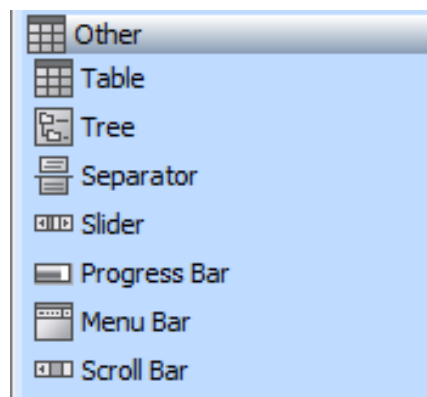
Po mygtukais yra tekstinių laukų grupė (žr. 16 pav.), kurią sudaro: tekstinis laukas, etiketė, saitas, sąrašas bei kt. Naudojant šiuos tekstinius elementus, galima sukurti teksto

įvedimo objektus grafinės vartotojo sąsajos lange. Tai gali būti naudotojo vardo įvedimo laukas, slaptažodžio įvedimo laukas ar kiti reikalingi tekstiniai objektai.



16 pav. Grafiniai teksto elementai

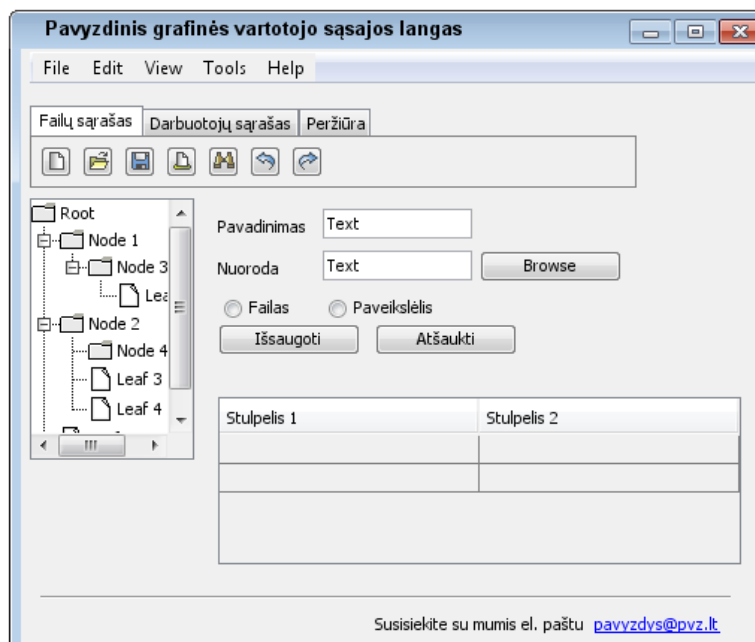
Paskutinę grupę elementų sudaro „Kiti“ elementai. Jų sąrašė galima rasti lenteles, objektų medžius, eigos juostą bei kitus papildomus GUI elementus, kurie nebuvo priskirti ankstesnėms grupėms.



17 pav. Kiti grafinės vartotojo sąsajos kūrimo elementai

Visi šie grafinės vartotojo sąsajos kūrimo elementai skirti tik atvaizdavimui, t. y. iš šių objektų rinkinio galima išsaugoti tik paveikslėlį. Tai viena iš pagrindinių šio įrankio problemų. Taip pat nėra galimybės vartotojo sąsajos elementų generuoti automatiškai pagal prieš tai sukurtą UML diagramą, nors įrankis ir leidžia kurti UML diagramas.

Rezultate *MagicDraw UML* įrankis yra puikus diagramų ir grafinės vartotojo sąsajos langų kūrimui, tačiau jis nėra pakankamai efektyvus, kad būtų galima automatizuoti GUI langų generavimą. Pavyzdinis grafinės vartotojo sąsajos langas pateiktas žemiau esančiame paveikslėlyje (žr. 18 pav.).



18 pav. Pavyzdinis grafinės vartotojo sąsajos prototipas sukurtas naudojant *MagicDraw UML*

Vertinant analizuojamo įrankio privalumus ir trūkumus, buvo sudaryta palyginamoji lentelė, kurioje aprašyti gauti rezultatai (žr. 9 lentelę).

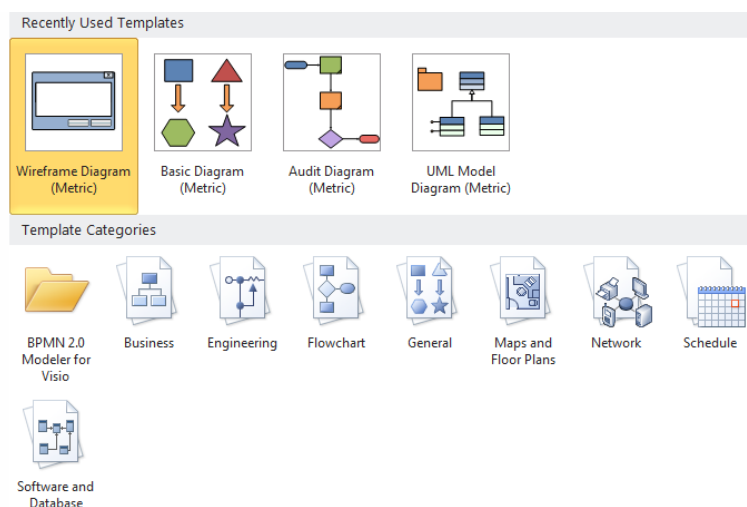
9 lentelė. *MagicDraw UML* įrankio privalumai ir trūkumai

Privalumai	Trūkumai
Palaikoma tiek Windows, tiek Mac OS X operacinių sistemų.	Lyginant su Microsoft Office Visio, nėra integruotas su Office dokumentais, todėl galima įkelti tik paveikslėlį, o ne visa diagramą.
Įrankis siūlo didelį pasirinkimą jau parengtų diagramų ir vartotojo sąsajos prototipų kūrimo elementų, kuriems galima keisti spalvas, šriftą, dydžius bei kt.	Vartotojo sąsajos prototipo negalima sugeneruoti automatiškai, o reikia dėti rankiniu būdu.
Įrankis leidžia sugeneruoti visų galimų paveikslėlių formatų failus (PNG, JPG, TIFF bei kt.)	Nėra toks intuityvus paprastam naudotojui.
Dinaminiai elementai, kurie kinta priklausomai vienas nuo kito.	
Puikus įrankis patyrusiam ir savo profesiją išmanančiam architektui, projektuotojui, analitikui ar dizaineriui.	

1.6.2. Microsoft Office Visio

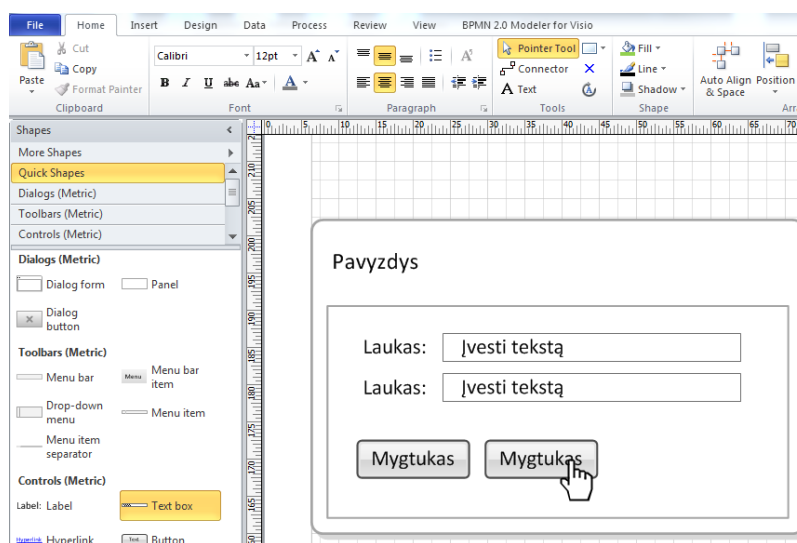
Microsoft produktas *Visio* siūlo patogų ir labai intuityvią diagramų ir vektorinės grafikos braižymo galimybę. Pats produktas buvo sukurtas 1992 m. *Shapeware* bendrovės. 2000 m. *Microsoft* nupirko produktą ir perėmė autorines teises.

Kitaip nei dauguma įrankių, visi *Visio* elementai bet kada gali būti pakeisti ar papildyti kitais. Įrankis siūlo standartinius diagramų elementus, o pačios diagramos pasirenkamos ir galimų pasirinkimų sąrašo (žr. 19 pav.).



19 pav. Microsoft Office Visio standartinių diagramų pasirinkimas

Pasirinkimų sąrašą galima praplėsti atsiųsus tam tikrų įrankio papildymų arba šablonų, kuriuos pridėjus prie standartinių įrankio elementų galima gauti dar geresnių rezultatų. Vartotojo sąsajos prototipų braižymui naudojamas „Wireframe“ papildinys, nes standartiniame *Visio* produkte tokios galimybės nėra. Pasinaudojant papildymu, galima gauti neblogų vartotojo sąsajos prototipo rezultatų, tačiau šis įrankis neleidžia automatiškai generuoti prototipų. Taip pat visi elementai yra statiniai ir nepriklausomi vienas nuo kito. Pavyzdinis vartotojo sąsajos prototipo kūrimas pateiktas žemiau esančiame paveikslėlyje (žr. 20 pav.).



20 pav. Microsoft Office Visio vartotojo sąsajos prototipo kūrimas naudojant „Wireframe“ papildymą

Apibendrinant produkto ypatybes, galima teigti, kad produktas yra labai lankstus ir siūlo labai platų diagramų bei kitų elementų naudojimą. Žinoma, visus papildymus reikia atsisiųsti, tačiau gerai tai, kad visi papildymai yra nemokami.

Microsoft Office Visio savybių analizės metu buvo užpildyta įrankio galimybių lentelė (žr. 10 lentelę). Iš lentelės matyti, kad šis įrankis atitinka beveik visus keliamus UML reikalavimus, tačiau su programinio kodo inžinerija įrankis nesusiduria. Toks įrankis tenkina elementarius verslo poreikius, kai reikia sukurti vizualius brėžinius be tolimesnio darbo su jais.

10 lentelė. *Microsoft Office Visio* įrankio keliamų reikalavimų ir galimybių lentelė

Kriterijus	Tenkina (+), netenkina (-)	
UML 1.4 versijos palaikymas	+	
UML 2.0 versijos palaikymas	+	
Robustness diagramos	+	
BPMN palaikymas	+	
Vartotojo sąsajos braižymas	+	
Vartotojo sąsajos generavimas iš modeliavimo diagramų	-	
Programavimo kalba	Tiesioginė kodo inžinerija	Atvirkštinė kodo inžinerija
.NET palaikymas	-	-
Java palaikymas	-	-
C++ palaikymas	-	-
Visual Basic palaikymas	-	-
Verslo procesų įrankių palaikymas	-	

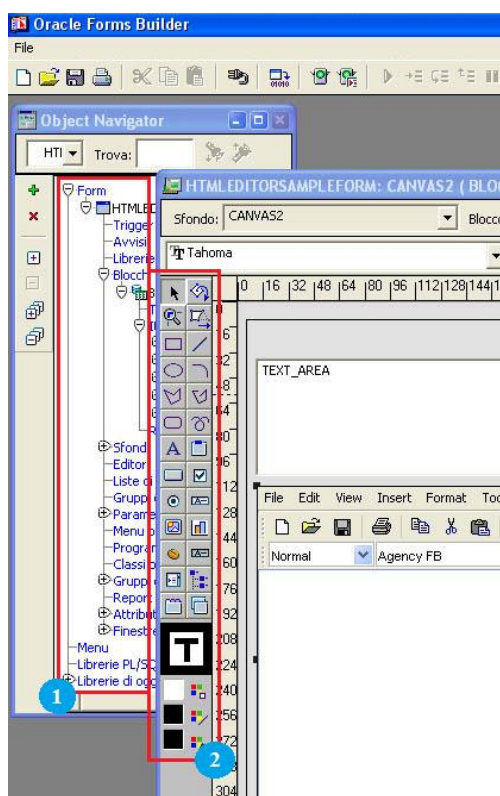
Vertinant analizuojamo įrankio privalumus ir trūkumus, buvo sudaryta palyginamoji lentelė, kurioje aprašyti gauti rezultatai (žr. 11 lentelę).

11 lentelė. *Microsoft Office Visio* įrankio privalumai ir trūkumai

Privalumai	Trūkumai
Labai patogus ir intuityvus įrankis, norint lengvai ir greitai suprojektuoti vartotojo sąsajos prototipą, diagramą ar kitą vaizdą.	Palaikomas tik <i>Windows OS</i> .
Įrankis suderintas su kitais <i>Microsoft Office</i> produktais, todėl <i>MS Word</i> dokumente galima tiesiogiai koreguoti diagramos ar vartotojo sąsajos prototipo elementus.	Nėra galimybės naudotis tiesiogine ar atvirkštine kodo inžinerija. Programavimui visai netinkantis įrankis.
Sudarant diagramas ar vartotojo sąsajos prototipus galima naudotis neribotu kiekiu įvairių elementų, keisti jų dydžius, spalvas, šriftą bei kt.	Elementai labiau statiniai, nei dinaminiai. Turint tuos pačius elementus keliose diagramose, jie tarpusavyje nesusieti ir nekinta kartu.

1.6.3. Oracle Forms Builder

Oracle Forms Builder yra *Oracle* produktas, kuris leidžia sugeneruoti vartotojo sąsajos prototipą. Pati vartotojo sąsaja siejasi su duomenų baze, taip pat yra dinaminė, o ne statinė kaip dauguma kitų vartotojo sąsajos prototipų generavimo įrankių siūlomi variantai. Žemiau esančiame paveikslėlyje matoma, kad visa vartotojo sąsajos prototipo forma saugoma hierarchinės struktūros objektų valdymo medyje, kuris paveikslėlyje pažymas „1“ numeriu. Šiame medyje atvaizduojami trigeriai, kurie iššaukiami atliekant tam tikrus veiksmus formoje.



21 pav. *Oracle Forms Builder* programinės įrangos pradinis langas su elementais

Oracle Forms Builder, standartiškai kaip ir kiti vartotojo sąsajos prototipų kūrimo įrankiai, kairėje lango dalyje siūlo formos elementų pasirinkimo galimybę (žr. 21 pav.-2). Iš pirmo žvilgsnio atrodo, kad sąrašas primena standartinės *Microsoft Windows* programėlės „*MS Paint*“ siūlomus elementus, o tai, patyrusiam vartotojo sąsajos prototipų kūrėjui, gali pasirodyti per daug elementaru ir sunkiai panaudojama.

Sugeneravus pavyzdinį *Oracle Forms Builder* vartotojo sąsajos prototipą, gautame rezultate turime dinaminį prototipą, kur kiekvienas mygtukas gali vykdyti tam tikrą veiksmą, o sąrašai užpildomi pagal duomenų bazės lenteles (žr. 22 pav.).

22 pav. Oracle Forms Builder vartotojo sąsajos prototipo pavyzdys

Pats vartotojo sąsajos prototipas labiau tinkamas realios sistemos kūrimui, nes jis jau iš karto suprogramuojamas taip, kad imtų duomenis iš duomenų bazės ir juos atvaizduotų ekrane. Taip pat naudojantis forma galima redaguoti, ištrinti bei papildyti duomenų bazėje esančius duomenis.

Oracle Forms Builder savybių analizės metu buvo užpildyta įrankio galimybių lentelė (žr. 12 lentelę). Iš lentelės matyti, kad šis įrankis neatitinka UML keliamų reikalavimų, tačiau tinka kurti sudėtingas vartotojo sąsajos prototipų formas, kurios tiesiogiai bendrauja su duomenų baze.

12 lentelė. Oracle Forms Builder įrankio keliamų reikalavimų ir galimybių lentelė

Kriterijus	Tenkina (+), netenkina (-)	
UML 1.4 versijos palaikymas	-	
UML 2.0 versijos palaikymas	-	
Robustness diagramos	-	
BPMN palaikymas	-	
Vartotojo sąsajos braižymas	+	
Vartotojo sąsajos generavimas iš modeliavimo diagramų	-	
Programavimo kalba	Tiesioginė kodo inžinerija	Atvirkštinė kodo inžinerija
.NET palaikymas	-	-
Java palaikymas	+	+
C++ palaikymas	-	-
Visual Basic palaikymas	-	-
Verslo procesų įrankių palaikymas	-	

Vertinant analizuojamo įrankio privalumus ir trūkumus, buvo sudaryta palyginamoji lentelė, kurioje aprašyti gauti rezultatai (žr. 13 lentelę).

13 lentelė. Oracle Forms Builder įrankio privalumai ir trūkumai

Privalumai	Trūkumai
Palaikomas <i>Linux</i> , <i>Windows</i> , <i>Solaris</i>	Vartotojo sąsajos prototipo kūrimas tampa labai

Privalumai	Trūkumai
operacinių sistemų.	sudėtingas paprastam naudotojui.
Tiesiogiai bendrauja su duomenų baze ir atvaizduoja jos lentelių duomenis.	Neturi jokių sąsajų su UML diagramomis.
Yra galimybė pridėti, redaguoti, ištrinti bei kitaip tvarkyti duomenų bazėje esančius duomenis.	Galimybė kodo peržiūrai tik JAVA programavimo kalba.
Vartotojo sąsajos prototipas yra beveik funkcionali sistema, kuria galima atlikti įvairius veiksmus.	Ribotas vartotojo sąsajos prototipo dizainas.

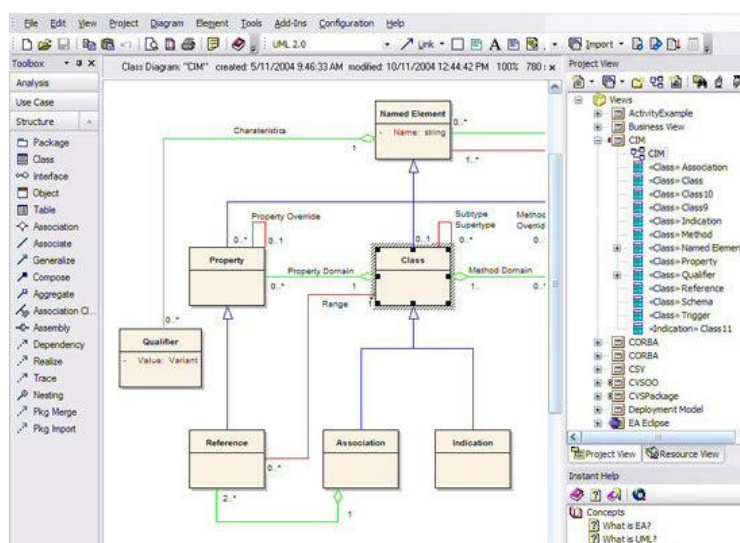
1.6.4. Enterprise Architect

Enterprise Architect kūrėjai Sparx Systems yra pirmaujantis naujoviškų sprendimų tiekėjas, kurių sprendimai paremti unifikauta modeliavimo kalba (UML) ir su ja susijusiais reikalavimais. Bendradarbiaujant su OMG, Sparx Systems yra įsipareigojusi realizuoti potencialią modeliavimo programinę įrangą paremtą atvirais standartais.

Pagrindinis įmonės produktas *Enterprise Architect* gavęs daugybę apdovanojimų nuo jos išleidimo 2000 m. rugpjūčio mėn. Naujausia PĮ versija yra 11, kuri turi 300 000 registruotų naudotojų visame pasaulyje.

Enterprise Architect palaiko šiuos standartus: UML 2, SysML, BPMN, UPDM, TOGAF, Zachman, DDS. Taip pat gali integruotis su Eclipse, Visual Studio, DOORS, Visio, XMI, Version Control Tools. Naujausia 11 PĮ versija palaiko net UML 2.4.1 standartą (Sparx Systems 2014).

Žiūrint į programos darbalaukį jis iš esmės niekuo nesiskiria nuo *MagicDraw UML* (žr. 23 pav.). Kairėje pusėje yra didelis pasirinkimas iš diagramų braižymo elementų. Dešinėje pusėje matomi visi diagramos elementai.



23 pav. Enterprise Architect darbalaukis

Vienas iš pagrindinių trūkumų yra tai, kad įrankis nesuteikia galimybės braižyti vartotojo sąsajos prototipų. Žinoma, įrankyje yra galimybė generuoti objektus iš programinio kodo, bet tai būtų per daug sudėtinga rezultatui pasiekti. Tad šis produktas netinkamas, kad būtų galima lengvai sukurti vartotojo sąsajos prototipus.

Enterprise Architect savybių analizės metu buvo užpildyta įrankio galimybių lentelė (žr. 14 lentelę). Iš lentelės matyti, kad šis įrankis pilnai atitinka UML ir programinio kodo keliamus reikalavimus, tačiau negali braižyti vartotojo sąsajos elementų rankiniu būdu ar generuoti jų automatiškai.

14 lentelė. *Enterprise Architect* įrankio keliamų reikalavimų ir galimybių lentelė

Kriterijus	Tenkina (+), netenkina (-)	
UML 1.4 versijos palaikymas	+	
UML 2.0 versijos palaikymas	+	
Robustness diagramos	-	
BPMN palaikymas	+	
Vartotojo sąsajos braižymas	-	
Vartotojo sąsajos generavimas iš modeliavimo diagramų	-	
Programavimo kalba	Tiesioginė kodo inžinerija	Atvirkštinė kodo inžinerija
.NET palaikymas	+	+
Java palaikymas	+	+
C++ palaikymas	+	+
Visual Basic palaikymas	+	+
Verslo procesų įrankių palaikymas	-	

Vertinant analizuojamo įrankio privalumus ir trūkumus, buvo sudaryta palyginamoji lentelė, kurioje aprašyti gauti rezultatai (žr. 15 lentelę).

15 lentelė. *Enterprise Architect* įrankio privalumai ir trūkumai

Privalumai	Trūkumai
Palaikoma tik <i>Windows</i> operacinės sistemos. Galimybė naudotis <i>Linux</i> ir <i>Mac</i> tik su <i>WINE</i> arba <i>CodeWeavers CrossOver</i> .	Lyginant su <i>Microsoft Office Visio</i> , nėra integruotas su <i>Office</i> dokumentais, todėl galima įkelti tik paveikslėlių, o ne visa diagramą.
Įrankis siūlo didelį pasirinkimą jau parengtų diagramų kūrimo elementų, kuriems galima keisti spalvas, šriftą, dydžius bei kt.	Vartotojo sąsajos prototipo negalima sugeneruoti automatiškai.
Įrankis leidžia sugeneruoti visų galimų paveikslėlių formatų failus (PNG, JPG, TIFF bei kt.)	Nėra toks intuityvus paprastam naudotojui.
Dinaminiai elementai, kurie kinta priklausomai vienas nuo kito.	Nėra galimybės dėti vartotojo sąsajos elementus rankiniu būdu.
Puikus įrankis patyrusiam ir savo profesiją	

išmanančiam architektui, projektuotojui, analitikui ar dizaineriui, tačiau tinkamas tik diagramų braižumui.	
---	--

1.7. Vartotojo sąsajos prototipo generavimo algoritmas

Šiame skyriuje analizuojami galimi vartotojo sąsajos elementų generavimo algoritmai. Vartotojo sąsają turi sudaryti: sistemos langai, langų įvedimo ir peržiūros laukai bei veiksmų mygtukai. Kad sukurti automatizuotą generavimo algoritmą, reikia nustatyti:

- kokios modeliavimo diagramos tam geriausiai tinka;
- kokie tų diagramų elementai yra privalomi norint generuoti vartotojo sąsajos elementus;
- apibrėžti generavimo procesą;
- nustatyti vartotojo sąsajos elementus ir internete naudojamus HTML formato objektų elementus;
- aprašyti algoritmą jungiantį vartotojo sąsajos elementus su HTML elementais.

Visi šie punktai suskaidyti į mažesnius poskyrius, kuriuose atliekamas detalus tyrimas. Apjungus gautus rezultatus galima pasiūlyti vartotojo sąsajos generavimo algoritmą, kurį naudojant automatizuojamas projektavimo etape vykdomas prototipų kūrimas.

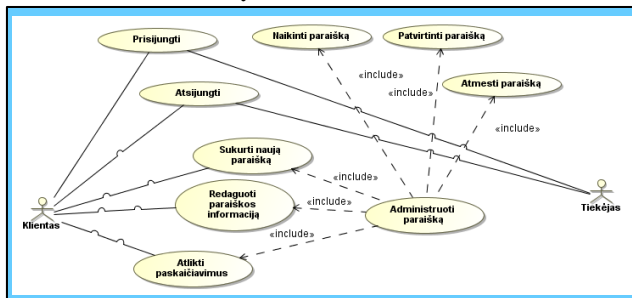
1.7.1. Diagramų ryšys su vartotojo sąsaja

Tiriant galimus UML diagramų ryšius su vartotojo sąsaja, buvo analizuojami galimi variantai. Labiausiai šiam tikslui tinkamos ir panaudojamos 2-3 UML elgsenos diagramos: panaudos atvejų, veiklų ir sekų bei viena UML struktūrą vaizduojanti klasių diagrama (žr. 24 pav.). Sekų diagrama galima ir nesinaudoti, jei grafinės vartotojo sąsajos prototipo funkcijų nenorima atvaizduoti laike. Tad trys pagrindinės UML diagramos, kurių būtinai reikia norint atvaizduoti grafinę vartotojo sąsają:

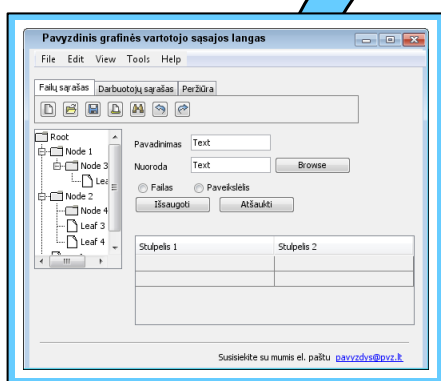
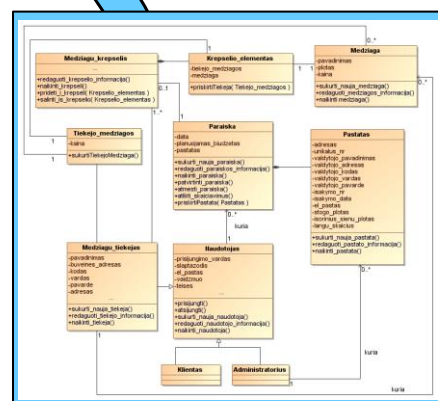
- panaudos atvejų diagrama, kuri apibrėžia funkcinis sistemos reikalavimus ir veiklos procesus;
- klasių diagrama – veiklos objektų sąsajai ir jų atributams apibrėžti;
- veiklų diagrama skirta panaudos atvejų realizavimui bei vartotojo scenarijams (Jesu' s M.Almendros-Jimenez *et. al.* 2008).

Kiekviena iš šių diagramų turi savo atitikmenį vartotojo sąsajos elementams. Iš panaudos atvejų bus generuojami langai, iš klasių bus generuojami languose esantys įvedimo ir peržiūros laukai, o iš veiklų – veiksmų mygtukai.

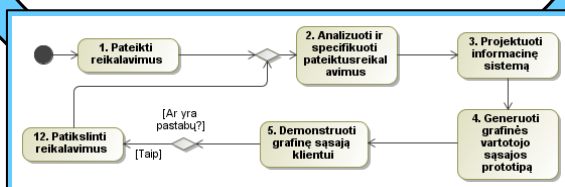
1. ARCHITEKTŪRA. PANAUDOS ATVEJŲ DIAGRAMOS



2. UŽPILDYMAS KLASIŲ DIAGRAMOS



4. PROTOTIPAS. GRAFINĖ VARTOTOJO SĄSAJA



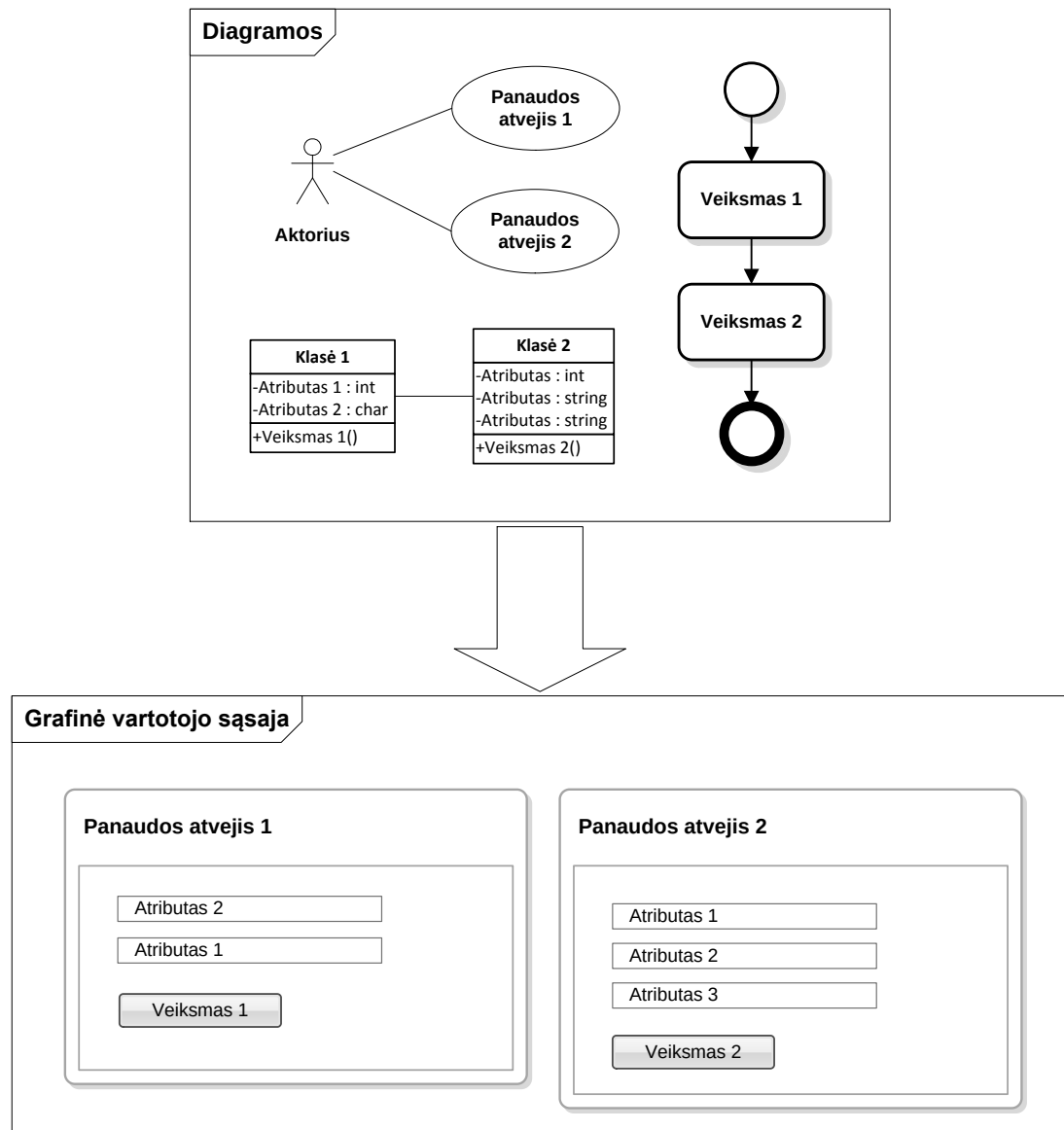
3. ELGSENA. VEIKLŲ DIAGRAMOS

24 pav. Vartotojo sąsajos kūrimo iš diagramų proceso modelis

Paveiksle pateiktas mechanizmas, kaip iš tam tikrų diagramų sugeneruojama vartotojo sąsaja. Mechanizmas padalytas į keturis etapus:

1. Architektūra. Panaudos atvejų diagramos.
2. Užpildymas. Klasių diagramos.
3. Elgsena. Veiklų diagramos.
4. Prototipas. Grafinė vartotojo sąsaja.

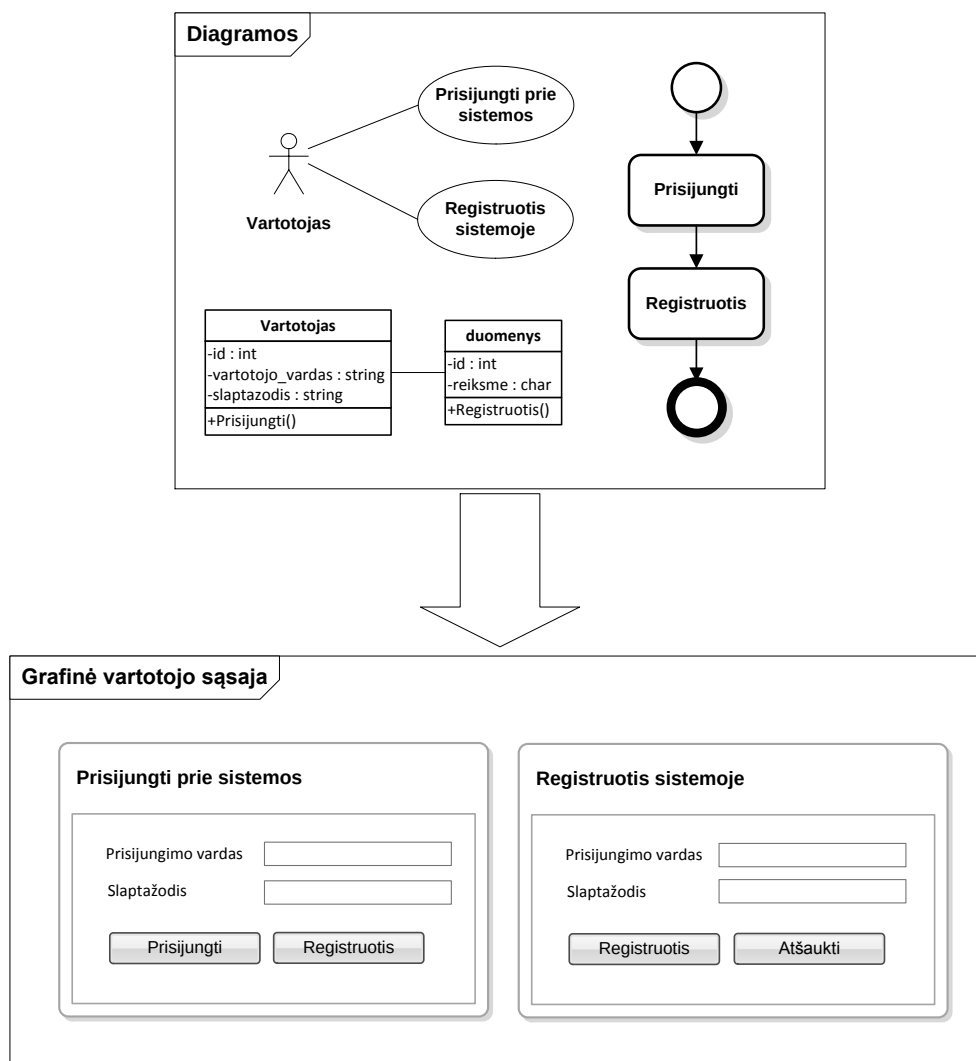
Daroma išvada, kad užtenka turėti bent vieną panaudos atvejį, bent vieną klasę ir vieną veiksmą, kad galima būtų sugeneruoti vieną grafinės vartotojo sąsajos langą su įvedimo galimybe ir tam tikru veiksmu, pvz. išsaugojimo veiksmu (žr. 25 pav.).



25 pav. Grafinės vartotojo sąsajos generavimas iš UML diagramų

Kiekvienas panaudos atvejų diagramoje esantis panaudos atvejis atvaizduoja grafinės vartotojo sąsajos langus, pvz., panaudos atvejis „Registruotis“ kurtų registracijos grafinės vartotojo sąsajos langą, kuriame vartotojas įvestų tam tikrus atributus ir vykdytų tam tikrą veiksmą. Norint vykdyti veiksmą, reikia veiklų diagramos, kurioje kiekvienas veiksmas turėtų savo funkcinį mygtuką, pvz., mygtukas „Registruotis“ atliktų registravimosi veiksmą. Kad vartotojas galėtų atlikti šį veiksmą, jam reikia suvesti tam tikrą informaciją į atributų laukus. Atributų laukai imami iš klasių diagramos, nes joje nurodomi sąryšiai tarp objektų ir lentelėse

saugoma informacija (atributai). Sukūrę klasę „Vartotojas“, joje sukurtume atributus: „identifikacinis numeris“, „vartotojo vardas“ ir „slaptažodis“ bei veiksmus: „registruotis“ ir „atšaukti registraciją“. Galiausiai turėtume visas diagramas, kurių reikia grafinei vartotojo sąsajai sukurti. Žemiau pateiktas paveikslėlis iliustruoja diagramų virsmą į grafinės vartotojo sąsajos langus (žr. 26 pav.).



26 pav. Grafinės vartotojo sąsajos generavimas iš UML diagramų naudojant realius duomenis

1.7.2. Diagramų privalomi elementai

Žemiau pateiktoje lentelėje išvardyti visi privalomi elementai iš kiekvienos diagramos, kurių būtinai reikia norint sugeneruoti vartotojo sąsajos prototipą.

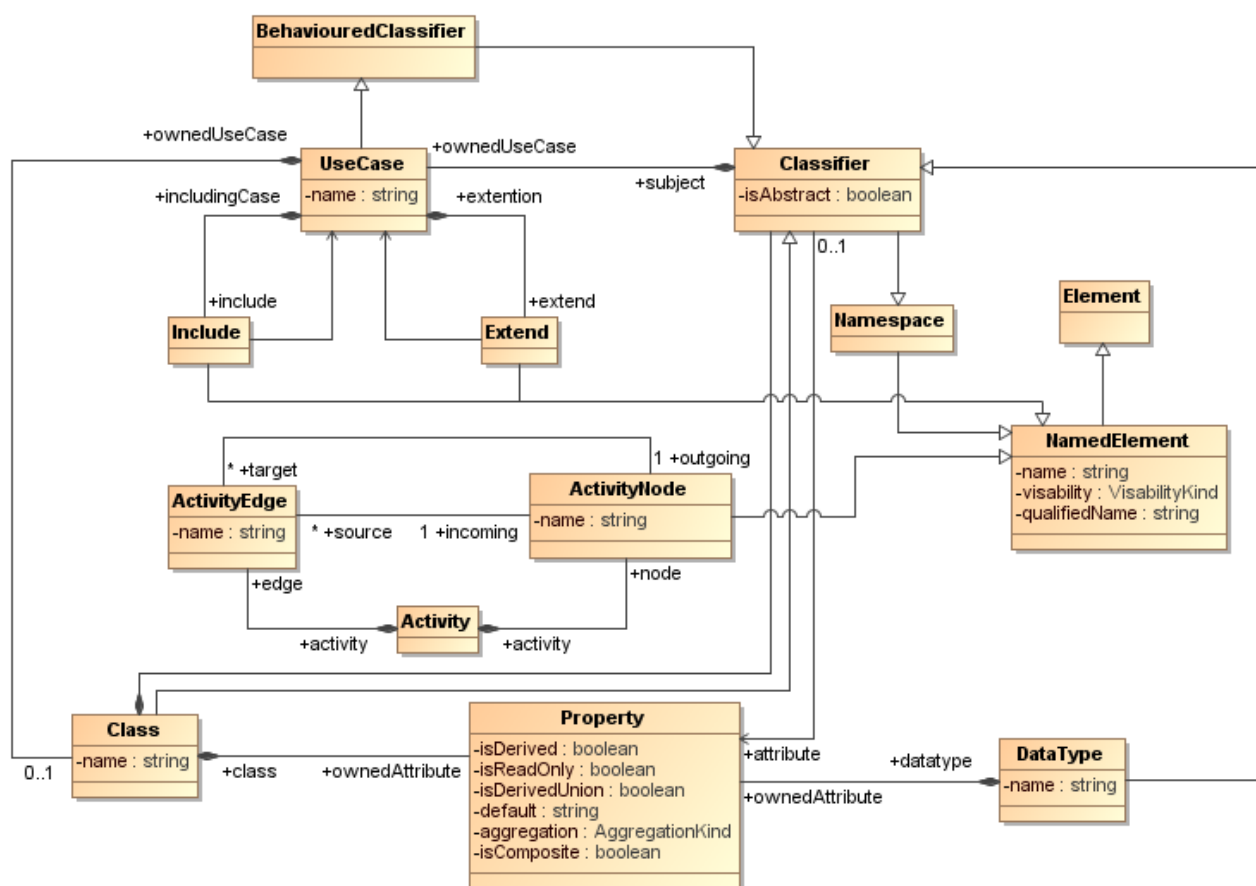
16 lentelė. Privalomų elementų diagramose sąrašas

Diagrama	Elementai	
Panaudos atvejų	Aktorius	Privalo būti bent vienas aktorius.
	Panaudos atvejis	Privalo būti bent vienas panaudos atvejis.
Veiklų	Pradžia	Privalo būti pradžios taškas.
	Pabaiga	Privalo būti pabaigos taškas.
	Veikla	Privalo būti bent viena veikla.
Klasių	Klasė	Privalo būti bent viena klasė su bent vienu atributu. Klasės operacija turi sietis su veikla, o atributai turi būti susieti su panaudos atvejais.

1.7.3. Modeliavimo diagramų vartotojo sąsajos elementai

Analizuojant OMG Unified Modeling Language, Superstructure 2.3 versiją (2010), klasių aprašyme galima rasti visų meta klasių aprašymus bei tarpusavio ryšius. Tyrimo metu apibrėžtos trys UML diagramos, kurių pagalba norima suformuoti vartotojo sąsajos elementus.

Pagrindinės diagramų klasės, kuriomis bus pasinaudota, yra: *Class*, *Property*, *UseCase*, *ActivityNode* ir *ActivityEdge*. Kiekviena iš šių klasių ir tarpusavio ryšiai pavaizduoti modelio diagramoje žemiau (žr. 27 pav.).



27 pav. UML diagramų klasių tarpusavio ryšių modelis

Diagramų tarpusavio ryšius su papildomais priedais, tokiais kaip „ownedAttribute“ ir kt. nagrinėjo Costal Dolors bei kiti jo kolegos darbe „Bendrųjų apribojimų UML apibrėžimų gerinimas“ (Dolors *et. al.* 2008). Lentelėje žemiau aprašytos modelio klasės ir jų atributai bei ryšiai (žr. 17 lentelę).

17 lentelė. Vartotojo sąsajos elementų sąrašas

Klasė	Aprašymas
<i>UseCase</i>	Panaudos atvejo sistemos įvykiai. Klasė aprašo dalyvaujančius veikloje aktorius ir ką jie gali atlikti su sistema.
<i>Include</i>	Panaudos atvejo ryšys, kuris nurodo, kad vienas panaudos atvejis įterpia kitą atvejį. Kad sėkmingai atliktų procesą, reikia atlikti įterpiančią panaudos atvejį.
<i>Extend</i>	Panaudos atvejo ryšys, kuris išplečia (angl. <i>extends</i>) vieną atvejį kitu. Išplėtimas vykdomas su tam tikromis sąlygomis ir nėra privalomas.
<i>Class</i>	Ši klasė apibūdina objektus, kurie turi atributus, operacijas ir ryšius vienas tarp kito.
<i>Property</i>	Klasė skirta atvaizduoti atributus arba jų rinkinius.
<i>DataType</i>	Klasė, kuri aprašo tam tikro atributo duomenų tipą.
<i>Activity</i>	Klasė, kuri aprašo veiklą.
<i>ActivityNode</i>	Klasė, kuri nurodo veiklą žingsnius ir jų eiliškumą.
<i>ActivityEdge</i>	Klasė, kuri aprašo dviejų žingsnių tarpusavio ryšį.
<i>Classifier</i>	Klasė, kuri aprašo įvykių rinkinį su kiekvieno iš jų ypatybėmis.
<i>BehavioredClassifier</i>	Turi vartotojo sąsajos realizacijos elementus.
<i>Namespace</i>	Elementų, turinčių savo pavadinimus, rinkinys.
<i>NamedElement</i>	Elementas, kuris gali turėti savo pavadinimą ir būti pagal jį identifikuotas.
<i>Element</i>	Sudėtinė modelio dalis, galinti turėti kitus elementus.

Klasių gali būti ir daugiau, tačiau šios yra pagrindinės, kurios turėtų dalyvauti vartotojo sąsajos generavime. Kiekviena klasė yra viena nuo kitos priklausoma, tad neturint vienos iš jų, sustabdomas generavimas ir rodoma klaida.

1.7.4. Vartotojo sąsajos elementai internetinėje sąsajoje

Šiais laikais dauguma organizacijų, prieš užsakydamos IS kūrimo darbus, pagalvoja, ar joms kurti lokalią, ar internetinę sistemą. Kadangi internetinė sąsaja labiau centralizuota ir prie jos nereikia specialių priemonių prisitaikyti, dauguma pasirenka internetinę sistemą.

Internetinio puslapio atvaizdavimui dažnai pasirenkamos HTML arba PHP (angl. *Hypertext Preprocessor*) programavimo kalbos. Renkantis iš šių dviejų, labiau yra standartizuota HTML, nes ji gali būti pritaikyta prie bet kokios kitos programavimo kalbos. Kad internetiniame puslapyje sukurti grafinius elementus, reikia iš pradžių juos žinoti. Žemiau pateiktoje lentelėje surašyti visi elementai ir jų tipai, kurie naudojami HTML programavimo kalboje (Hobson 2014).

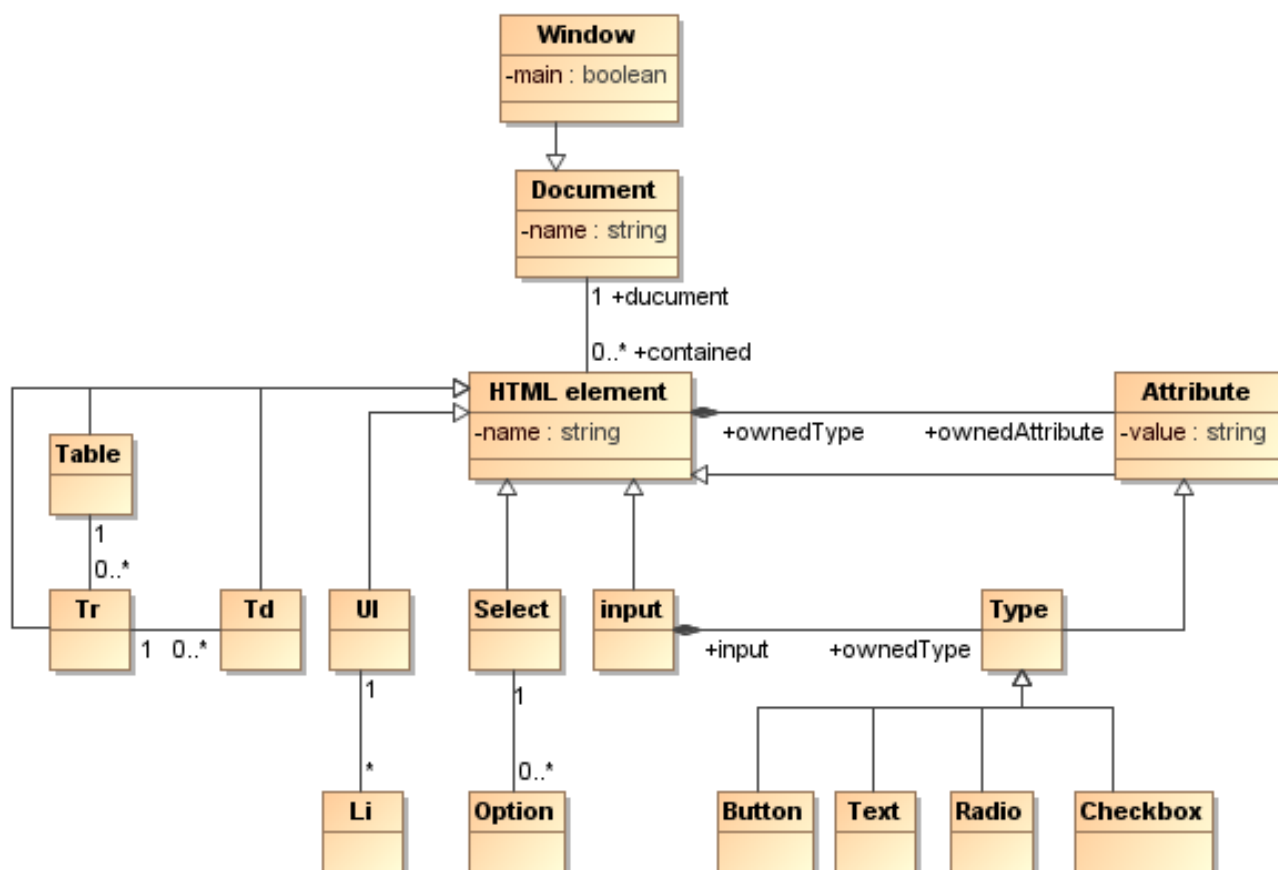
18 lentelė. HTML vartotojo sąsajos elementų sąrašas

Tipas	Elementas	Aprašymas
Šakninis elementas	<code><html></code>	Atstovauja HTML arba XHTML dokumento šaknis. Visi kiti elementai turi būti šio elemento palikuonys.
Dokumento metaduomenys	<code><head></code>	Atstovauja metaduomenų kolekciją apie dokumentą, įskaitant sąsajas arba apibrėžimus, scenarijus ir stilių.
	<code><title></code>	Nurodo dokumento pavadinimą, rodomą naršyklės pavadinimo juostoje arba puslapio skirtuką. Jis gali būti tik tekstinio pavidalo.
	<code><base></code>	Nustato bazinį URL į puslapį.
	<code><link></code>	Naudojamas susieti JavaScript ir CSS su esamu HTML dokumentu.
	<code><meta></code>	Apibrėžia metaduomenis, kurių negalima apibrėžti naudojant kitus HTML elementus.
	<code><style></code>	Naudojamas aprašyti CSS stiliams.
Scenarijai	<code><script></code>	Apibrėžia scenarijus. Scenarijų kalba yra JavaScript.
	<code><noscript></code>	Apibrėžia alternatyvų turinio rodymą, kai naršyklė nepalaiko scenarijų.
	<code><template></code>	Konteineris kliento pusėje.
Kiti elementai		
Žymėjimai	<i>body, section, nav, article, h1, h2, h3, h4, h5, h6, header, footer, address, main.</i>	
Turinio grupavimas	<i>p, hr, pre, blockquote, ol, ul, li, dl, dt, dd, figure, figcaption, div.</i>	
Formos	<i>form, optgroup, option, textarea, button, label, input, select,</i>	
Mygtukai	<i>button</i>	
Paveiksliukai	<i>img</i>	
Objektai	<i>object</i>	

Pasinaudojant šiais elementais, iš modeliavimo diagramų elementų galima nurodyti sąlygas, su kuriomis būtų iškviečiami tam tikri HTML elementai, pvz. jei reikia atlikti veiksmą mygtuko paspaudimu, diagramoje nurodomas mygtuko elementas „*button*“, kuris HTML kode atitinka mygtuką.

Kad sudaryti internetinės vartotojo sąsajos meta modelį, reikia pasinaudoti elementais, kurie buvo aprašyti aukščiau esančioje lentelėje. Kiekvienas HTML elementas yra vienaip ar kitaip susietas vienas su kitu, todėl galima sudaryti šių elementų modelį, t. y. diagramą, kaip jie vienas su kitu bendrauja.

Iš pagrindinių vartotojo sąsajos elementų, kurie naudojami grafinei vartotojo sąsajai kurti, tokių kaip: *button*, *text*, *radio*, *checkbox*, *table* ir kt., sudaromas modelis. Žemiau patikta internetinės vartotojo sąsajos meta-modelio diagrama.



28 pav. Internetinės vartotojo sąsajos elementų modelis

Klasių aprašas pateiktas žemiau esančioje lentelėje (žr. 19 lentelę).

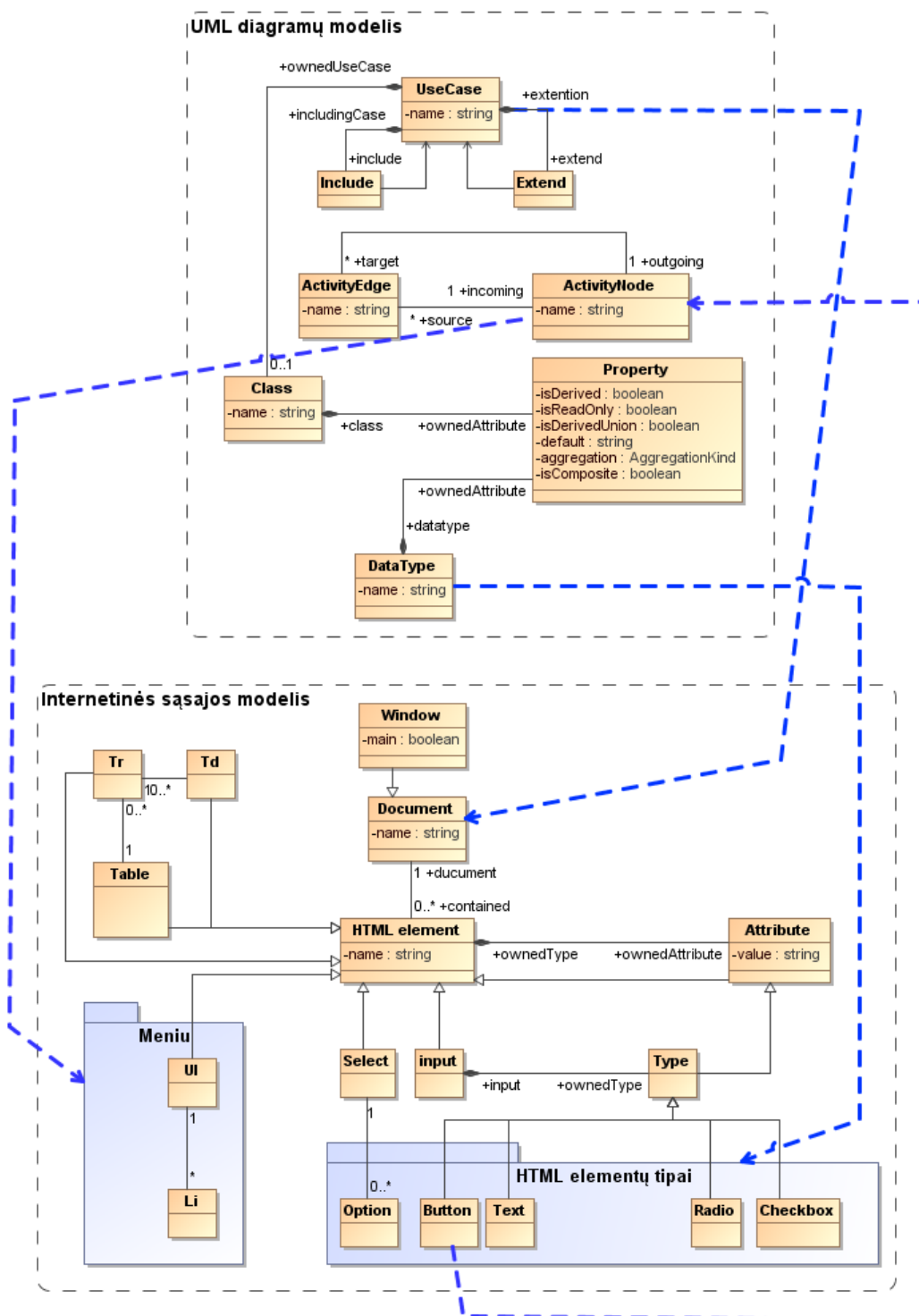
19 lentelė. Vartotojo sąsajos elementų sąrašas

Klasė	Aprašymas
<i>Window</i>	Vartotojo sąsajos langas, kuris savyje turi kitus elementus. Atributas „ <i>main: boolean</i> “ nurodo lango vietą langų hierarchijoje.
<i>Document</i>	Dokumentas, kurio tipas yra HTML.
<i>HTML element</i>	HTML vartotojo sąsajos elementas.
<i>Table</i>	Lentelės tipo grafinis elementas, skirtas atvaizduoti duomenų suvestinei.

Klasė	Aprašymas
<i>Tr</i>	Lentelės stulpelio grafinis elementas.
<i>Td</i>	Lentelės eilutės grafinis elementas.
<i>Ul</i>	Grafinis elementas, naudojamas formuoti sąrašus su ženkliais.
<i>Li</i>	Grafinis elementas, nurodantis sąrašo su ženkliais objektus.
<i>Select</i>	Grafinis elementas, kuris skirtas pasirinkimo iš sąrašo lauko atvaizdavimui. Jei yra galimybė pasirinkti keletą reikšmių, naudojamas „Multi-select“.
<i>Option</i>	Grafinis elementas, kuris naudojamas reikšmių užpildymui „Select“ elementui.
<i>Input</i>	HTML objekto pridėjimo elementas.
<i>Type</i>	HTML elemento atributas.
<i>Button</i>	Mygtuko grafinis elementas, kuris skirtas veiksmui atlikti. Siejamas su „Input“ elementu.
<i>Text</i>	Teksto įvedimo grafinis elementas. Siejamas su „Input“ elementu.
<i>Radio</i>	Pasirinkimo grafinis elementas, kuris leidžia pasirinkti tik vieną reikšmę. Siejamas su „Input“ elementu.
<i>Checkbox</i>	Pasirinkimo grafinis elementas, kuris leidžia pažymėti daugiau nei vieną reikšmę, arba nepažymėti nieko. Siejamas su „Input“ elementu.
<i>Attribute</i>	HTML elemento atributas.

1.7.5. Vartotojo sąsajos elementų apjungimas

Šiame skyriuje aprašomas ir pateikiamas UML diagramų elementų ir HTML vartotojo sąsajos elementų apjungimo modelis. Modelyje punktyrais atskirtos UML ir HTML dalys.

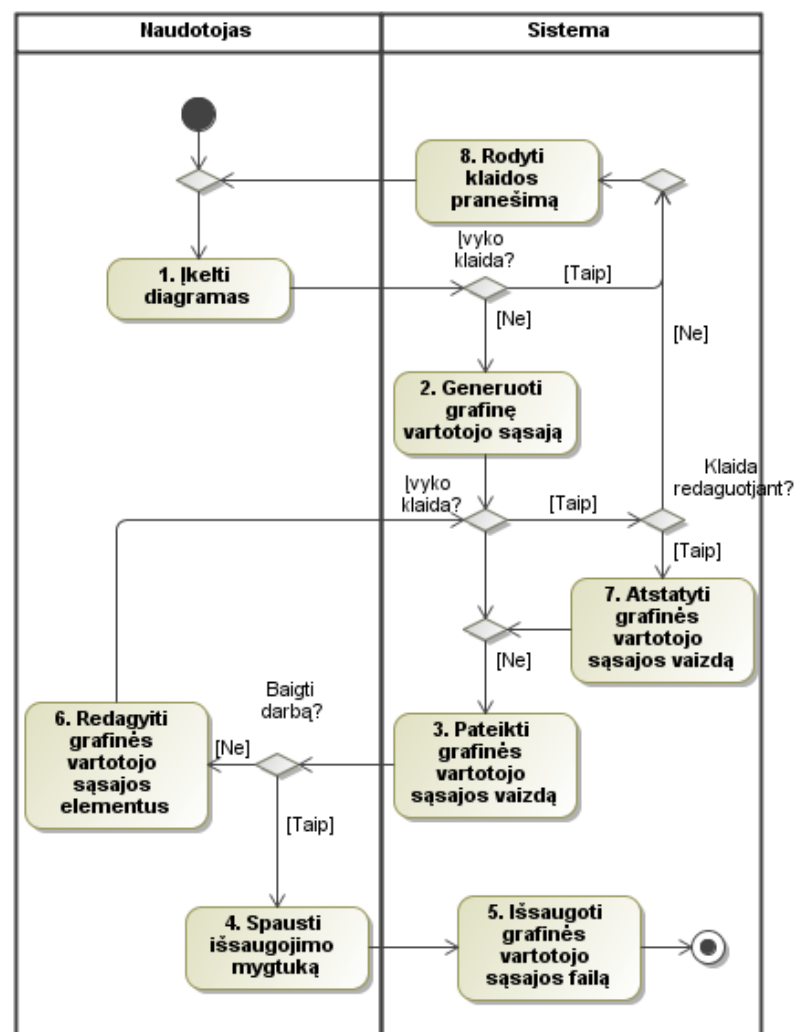


29 pav. Modeliavimo diagramų ir HTML vartotojo sąsajos elementų klasių apjungimas

Modelyje vaizduojama kokios klasės iš UML diagramų generuoja HTML internetinės sąsajos elementus. *ActivityNode* veiklų diagramos klasė generuoja meniu elementus: UI ir Li. Vėliau šie elementai atvaizduojami kaip meniu sąrašas. Pagal duomenų tipą generuojami ir kiti HTML elementai.

1.7.6. Vartotojo sąsajos generavimo procesas

Sukurta sistema arba įrankis, kuris formuotų grafinę vartotojo sąsają, turėtų bendrauti su naudotoju, kuris pateiktų UML diagramas. Šis procesas atvaizduotas diagramoje žemiau (žr. 30 pav.).



30 pav. Grafinės vartotojo sąsajos generavimo iš diagramų veiklos procesas

Iš proceso matyti, kad procesas turi daug pasirinkimo (angl. *decisions*) elementų, kurie su tam tikromis sąlygomis tęsia arba stabdo procesą. Įvykus kokiai nors klaidai, naudotojas turi būti informuojamas, kokia klaida įvyko ir dėl ko.

2. REZULTATAI IR JŲ APTARIMAS

2.1. Sistemos kūrimas

Sistemos kūrimo procesas prasideda nuo reikalavimų. Šie reikalavimai padeda sistemų inžinieriams identifikuoti: būsimos sistemos sudėtingumą, jos funkcinius panaudos atvejus, verslo problemas bei daug kitų dalykų, dėl ko buvo nuspręsta kurti IS. Kuriant tokią sistemą, yra labai svarbu surinkti, išanalizuoti bei specifikuoti visus esamus ir būsimus reikalavimus. Tokius reikalavimus pateikia užsakovas, tačiau tai gali būti ir vidinis projektas, kur reikalavimai kuriami projekto komandos viduje.

Šiame darbe analizuota: IS kūrimo procesas, proceso dalyviai bei jų pareigos. Procesai vaizduojami UML diagramomis, kurios yra sudedamosios IS projektavimo ir kūrimo dalys. Taip pat aprašomas diagramų sąryšis su grafine vartotojo sąsaja, kuri dažniausiai yra sukurtos sistemos rezultatas.

Išanalizavus IS kūrimo aplinką nustatyta, kad joje dalyvauja: užsakovas, reikalavimų inžinieriai, projektuotojai, programuotojai, testuotojai, projektų vadovai ir kt. Labai svarbų vaidmenį kuriant IS atlieka užsakovas, kuris pateikia reikalavimus sistemai, juos derina su projekto komanda, vertina rezultatus ir, atsižvelgdamas į juos, pateikia pastabas ir komentarus apie kuriamą sistemą. Dažniausiai užsakovas į sistemą žiūri iš vartotojo pusės, t. y. iš savo specifinės darbo veiklos procesų, ko dažnai projekto komanda gali ir nesuprasti, nes nėra susidūrusi su kliento darbo aplinka. Todėl prieš kurdama IS, projekto komanda turi įsigilinti į kuriamos sistemos aplinką, kas ir kaip sistema naudosis, tai padeda sukurti tikslingą ir reikalingą produktą, kuriuo patogiu ir naudinga naudotis. Žinoma, pasitaiko atvejų, kai užsakovai patys nežino ko tiksliai nori, nes nėra gerai susidūrę su informacinėmis technologijomis ir jų galimybėmis, todėl projekto komandai dažnai tenka susidurti su abstrakčiais reikalavimais, kartais net ir nelogiškais. Tokiais atvejais į pagalbą ateina reikalavimų inžinieriai, kitaip dar vadinami sistemų analitikais, kurie išanalizuoja pateiktus reikalavimus, suderina jų įgyvendinimą su užsakovais ir pasiūlo sistemos funkcinius sprendimus.

2.2. Prototipų naudojimas

Kai reikalavimai būna apibrėžti ir aiškūs, o reikalavimų inžinieriai būna padarę savo darbą, užsakovai prašo pademonstruoti kaip preliminariai atrodys būsima sistema. Jei sistemos kūrimo procesas dar nepriėjo programavimo darbu etapo, analitikams reikia paruošti būsimos IS prototipą. Norint jį paruošti, reikia ne tik viską išanalizuoti ir dokumentuoti, tačiau ir

suprojektuoti būsimos sistemos architektūrą bei, pasitelkus tam tikrus įrankius, sukurti vizualiai aiškų ir suprantamą sistemos prototipą. Projekto komandos tikslas – kuo greičiau pateikti grafinį vartotojo sąsajos prototipą klientui, nes jis nesupranta sistemos kodo ir kas vyksta jo viduje, todėl negali vertinti projekto eigos progreso. Kuo greičiau užsakovas patvirtins būsimą projekto planą ir reikalavimų atitikimą, tuo greičiau bus galima pradėti realizavimo etapą. Skubama todėl, kad kiekvienas projektas turi savo terminus, o šie dažniais atvejais būna labai trumpi.

Projektavimo proceso etape prisijungia ir projektuotojas, kuris turėtų suprojektuoti būsimą sistemą pagal pateiktus reikalavimus. Tai gali padaryti ir analitikas, jei jo kompetencija pakankama suprojektuoti visą sistemą. Prototipo kūrimui reikalingos papildomos priemonės ir nemažos laiko sąnaudos. Atsižvelgiant į tai, kad projektuotojai ir analitikai kartu turi kurti ir derinti sistemos reikalavimų specifikaciją bei visos sistemos modelį, o projekto plane laiko ir finansiniai ištekliai prototipo kūrimui dažniausiai nebūna numatomi, šį etapą gerokai pagreitinę algoritmas, kuris leistų automatiškai generuoti grafinės vartotojo sąsajos prototipus iš reikalavimų analizėje braižomų UML diagramų. Toks algoritmas būtų naudingas abiem šalims, nes taip būtų sutaupomas laikas, o ir ankstyvojoje projekto kūrimo stadijoje būtų identifikuojamos galimos sistemos klaidos bei trūkumai.

2.3. Pasirinktas modeliavimo įrankis

Buvo atlikta vartotojo sąsają galinčių kurti įrankių lyginamoji analizė, kurios metu paaiškėjo, jog nė vienas nėra tinkamai pritaikytas generuoti grafinės vartotojo sąsajos iš nubraižytų diagramų, todėl šio darbo metu buvo analizuojamos galimybės papildyti tinkamiausią įrankį funkcionalumu, kuris galėtų atlikti grafinės vartotojo sąsajos elementų generavimą iš UML diagramų, nes būtent šios diagramos pripažintos ekspertų tinkamiausiomis IS modeliavimui.

Modeliavimo įrankių analizės metu buvo vertinti ir palyginti keturi skirtingi produktai: *MagicDraw UML*, *Microsoft Office Visio*, *Oracle Forms Builder* ir *Enterprise Architect*. Išsiaiškinus kiekvieno įrankio galimybes ir įvertinus privalumus bei trūkumus, buvo nuspręsta, kad geriausiai vartotojo sąsajos elementų generavimui tinka *MagicDraw UML*. Šis įrankis puikiai pritaikytas dinaminių UML diagramų braižymui, taip pat turi vartotojo sąsajos elementų biblioteką su jau paruoštais grafiniais objektais.

Vienas iš pagrindinių reikalavimų, kad įrankis turi generuoti programinį kodą iš sumodeliuotų diagramų taipogi tenkinamas. *MagicDraw UML* ne tik generuoja programinį kodą iš UML diagramų, bet ir suteikia net kelių programavimo kalbų pasirinkimą. Žemiau pateikiama XML failo kodo dalis, kuri gauta iš veiklų diagramos.

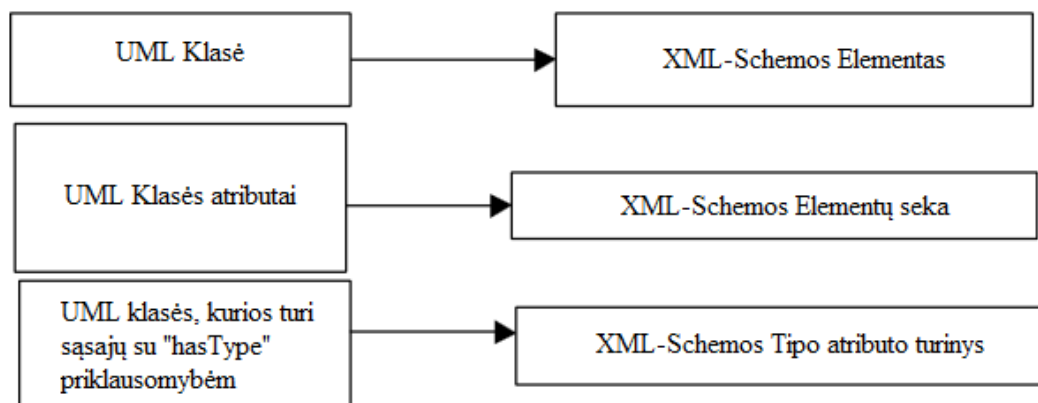
```

- <mdElement xmi:id="_17_0_6_1121036d_1400427678087_692895_13602" elementClass="SwimlaneHeader">
  <elementID xmi:idref="_17_0_6_1121036d_1400427669037_836609_13591"/>
  <geometry>487, 83, 184, 21</geometry>
  <compartment compartmentID="TAGGED_VALUES"/>
  <verticalCenterline>_17_0_6_1121036d_1400428339975_904879_13738</verticalCenterline>
  <rotated xmi:value="false"/>
</mdElement>
- <mdElement xmi:id="_17_0_6_1121036d_1400427669039_991910_13593" elementClass="SwimlaneCell">
  <geometry>487, 104, 184, 584</geometry>
  - <mdOwnedViews>
    - <mdElement xmi:id="_17_0_6_1121036d_1400427782640_461594_13643" elementClass="CallBehaviorAction">
      <elementID xmi:idref="_17_0_6_1121036d_1400427782630_186576_13642"/>
      <geometry>528, 126, 100, 45</geometry>
      <compartment compartmentID="TAGGED_VALUES"/>
      <verticalCenterline>_17_0_6_1121036d_1400427669039_289286_13592</verticalCenterline>
      <horizontalCenterline>_17_0_6_1121036d_1400427703654_89096_13618</horizontalCenterline>
    </mdElement>
    - <mdElement xmi:id="_17_0_6_1121036d_1400427813855_883120_13650" elementClass="CallBehaviorAction">
      <elementID xmi:idref="_17_0_6_1121036d_1400427813844_543895_13649"/>
      <geometry>528, 202, 100, 69</geometry>
      <compartment compartmentID="TAGGED_VALUES"/>
      <verticalCenterline>_17_0_6_1121036d_1400427669039_289286_13592</verticalCenterline>
      <horizontalCenterline>_17_0_6_1121036d_1400427813856_225949_13651</horizontalCenterline>
    </mdElement>
    - <mdElement xmi:id="_17_0_6_1121036d_1400427813869_897110_13654" elementClass="ControlFlow">
      <elementID xmi:idref="_17_0_6_1121036d_1400427813858_601200_13652"/>
      <linkFirstEndID xmi:idref="_17_0_6_1121036d_1400427813855_883120_13650"/>
      <linkSecondEndID xmi:idref="_17_0_6_1121036d_1400427782640_461594_13643"/>
      <geometry>578, 202; 578, 171;</geometry>
      <compartment compartmentID="TAGGED_VALUES"/>
      <nameVisible xmi:value="true"/>
    </mdElement>
  </mdOwnedViews>
</mdElement>

```

31 pav. XML kodo dalis, gauta iš veiklų diagramos, naudojant *MagicDraw UML*

Iš kodo matyti, kad kiekvienas diagramos elementas aprašomas su savo identifikaciniu numeriu, dydžio išmatavimais bei kitais parametrais. Kode pateikta plaukimo juostos (angl. *swimline*) antraštė, veiklos (angl. *CellBehaviorAction*) ir srautas (angl. *ControlFlow*). Turint duomenis apie visus veiklų diagramos elementus, galima atkurti norimą vartotojo sąsajos vaizdą iš sumodeliuotos diagramos. Tam tereikia sudėti ryšius tarp UML klasių bei jų atributų ir XML elementų (Marchetti 2005). Apjungimas pavaizduotas žemiau esančiame paveikslėlyje.



32 pav. UML klasių ir XML elementų sujungimas

2.4. Prototipo generavimo algoritmo apibendrinimas

Analizės metu pateiktas pasiūlymas, kad sukurti grafinę vartotojo sąsają iš diagramų, užtektų 3-4 tipų UML diagramų: panaudos atvejų (angl. *use case*), klasių (angl. *class*) ir veiklos (angl. *activity*). Tais atvejais, kai grafinę vartotojo sąsają reikėtų atvaizduoti laike, būtų pasinaudota sekų (angl. *sequence*) diagrama.

Pasiūlytas vartotojo sąsajos prototipo generavimo algoritmas transformuoja UML diagramų elementų klases ir HTML internetinės sąsajos elementų klases. Generavimo mechanizmas padalintas į keturis etapus: architektūra (panaudos atvejų diagramos), užpildymas (klasių diagramos), elgsena (veiklų diagramos) ir prototipas (grafinė vartotojo sąsaja).

3. PROJEKTAVIMAS

3.1. Sistemos duomenys

Šioje dalyje aprašoma būsto renovacijos IS projektavimo metu gauti duomenys. Būsto renovacijos sistema pasirinkta dėl šiuo metu Lietuvoje aktualios daugiabučių renovacijos problemos, kuri aktuali daugumai Lietuvos miestų.

Susiduriama su problema, dėl kurios stringa daugiabučių namų renovacija. Pasak šiaulielių, labiausiai nuo šio žingsnio sulaiko didžiulės kainos, kurias nustato renovacijos planų rengėjai. Šiaulių miesto savivaldybės vicemeras Juras Andriukaitis aiškino, kad tris kartus skelbė investicinių planų rangovų paiešką, tačiau nepavyko rasti tinkamo varianto dėl aukštos kainos (Jankauskienė 2014).

Atsižvelgiant į susiklosčiusią situaciją, buvo nuspręsta suprojektuoti IS, kuri padėtų bendrijoms bei visiems norintiems susirasti rangovus visoje Lietuvoje, kurie sutiktų atlikti darbus už nustatytą kainą. Kadangi sistema turi būti pasiekiamą internetu, projektuojama sistema turi būti kuriama kaip internetinis puslapis (angl. *web application*).

Kad suprojektuoti tinkamą sistemą, reikia parengti trijų tipų diagramas: panaudos atvejų, klasių ir veiklų. Kiekvienoje iš jų aprašomi duomenys bei atliekami kiti žymėjimai.

3.1.1. Panaudos atvejų diagrama

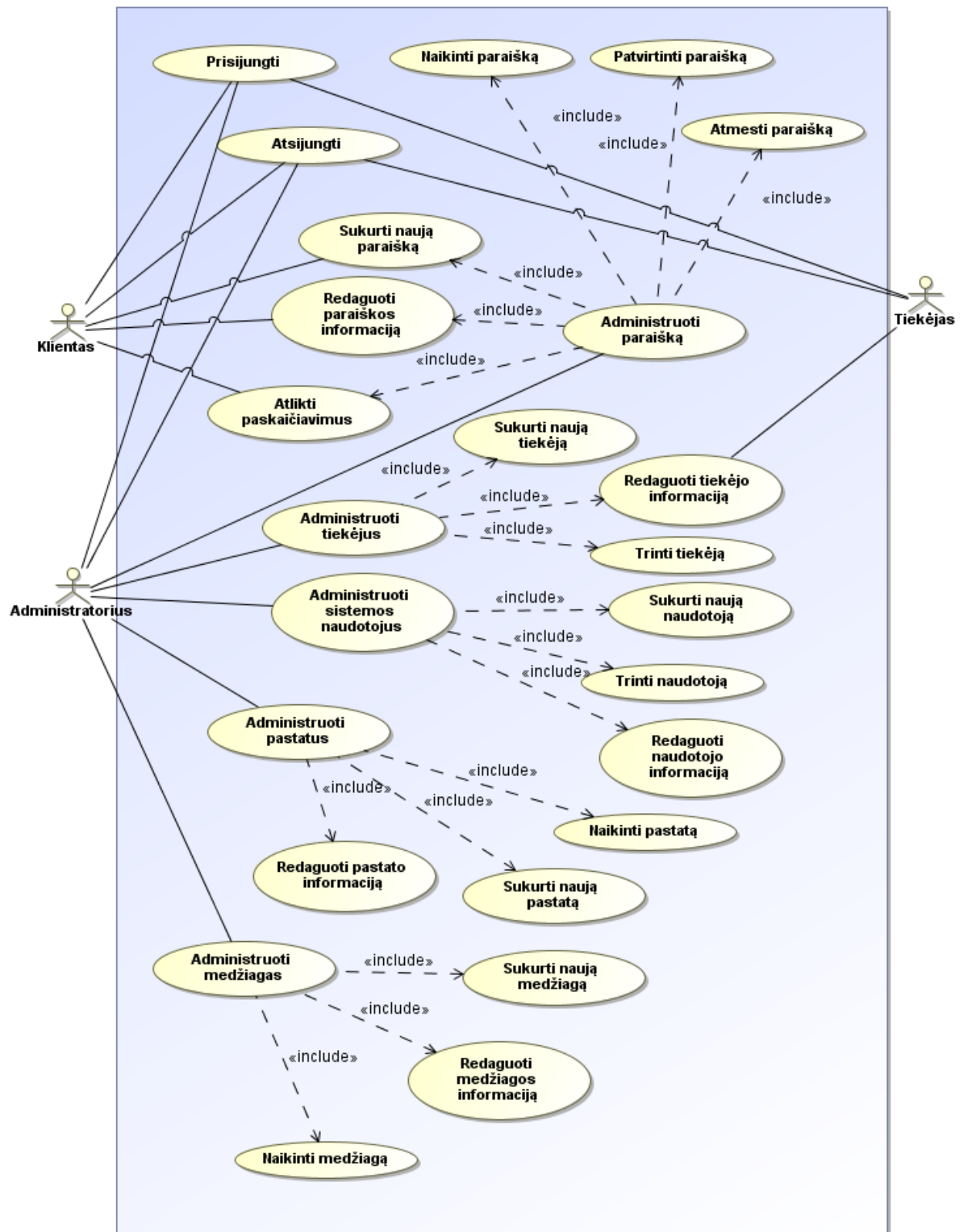
Panaudos atvejų diagrama naudojama tam, kad apibrėžti sistemoje naudojamą funkcijas. Visi panaudos atvejai atitinka tam tikrą funkciją, pvz.: sukurti naują paraišką, sukurti naują naudotoją, redaguoti pastato informaciją ir t. t.

Kiekvienas iš panaudos atvejų turi turėti savo aprašymą. Aprašymas susideda iš: panaudos atvejo pavadinimo, aprašymo, sąlygų „prieš“ ir „po“, pagrindinio scenarijaus, alternatyvių scenarijų (jei tokie yra) ir dar kitų dalių. Visų panaudos atvejų aprašymas gali užimti net kelis šimtus projektavimo dokumento lapų, todėl dėl baigiamojo darbo apimčių aprašymai nebus daromi.

Panaudos atvejų diagramoje (žr. 33 pav.) yra keli pagrindiniai atvejai, kurie apjungia daug mažesnių atvejų. Tai nėra sistemoje vykdomos funkcijos, bet toks žymėjimas supaprastina diagramos atvaizdavimą. Pagrindiniai atvejai yra šie: administruoti paraišką, administruoti tiekėjus, administruoti sistemos naudotojus, administruoti pastatus ir administruoti medžiagas.

Pagrindiniai sistemos naudotojai yra:

- klientas – kuris nori teikti paraišką būsto renovacijai;
- tiekėjas – kuris nori skelbti savo parduodamas medžiagas;
- administratorius – kuris administruoja sistemą ir jos naudotojus.

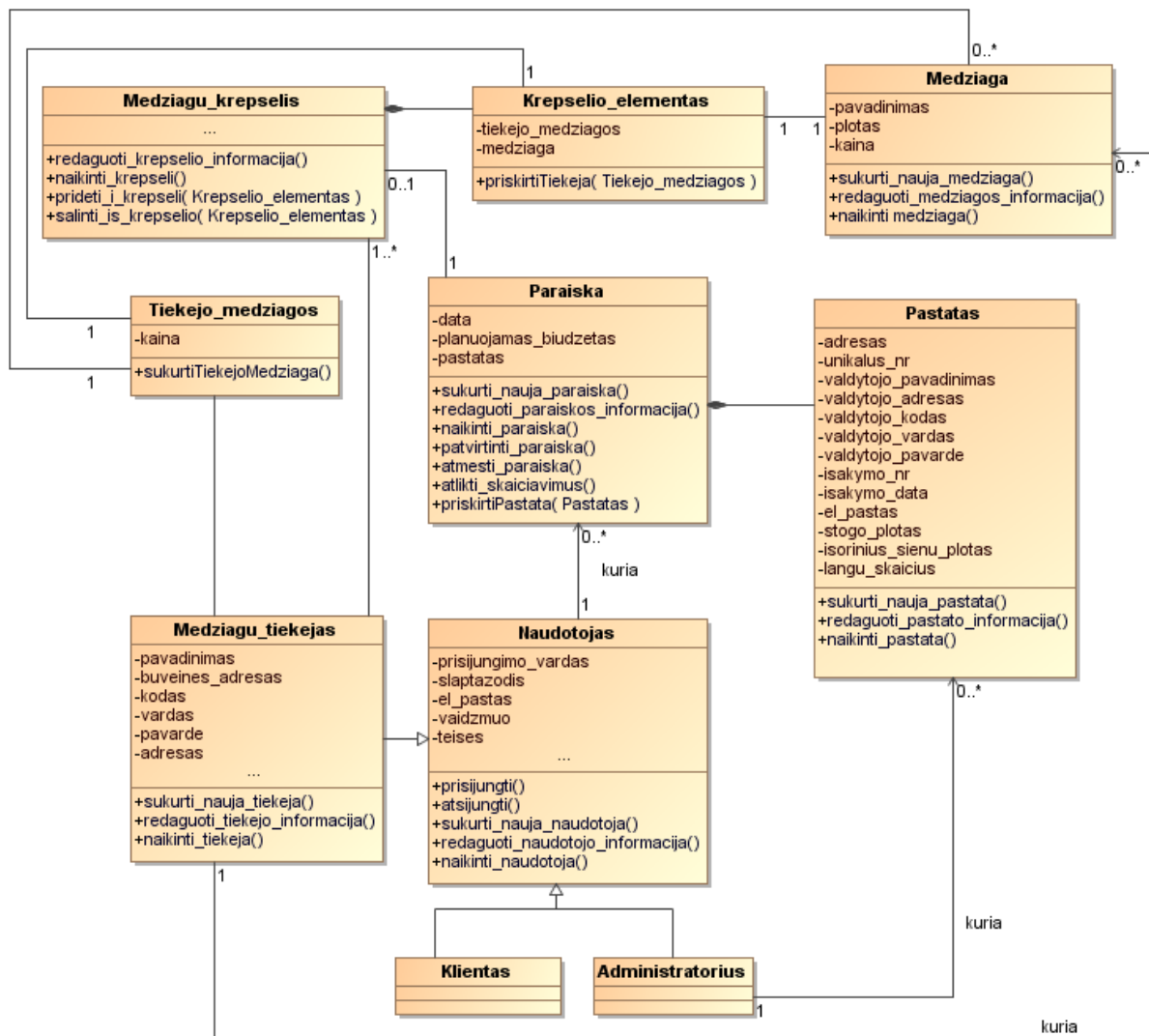


33 pav. Būsto renovacijos IS panaudos atvejų diagrama

Naudojantis šia panaudos atvejų diagrama, galima suprasti visas kuriamos būsto renovacijos IS funkcijas. Tokia diagrama suprantama net ir paprastam vartotojui, kuris neturi informatikos žinių.

3.1.2. Klasių diagrama

Klasių diagrama naudojama tam, kad apibrėžti sistemoje saugomus duomenis. Klasių diagramoje sistemoje esantys objektai aprašomi kaip klasės, kurios turi savo atributus ir operacijas. Žiūrint į būsto renovacijos IS klasių diagramą (žr. 34 pav.), galima matyti, kad sistemos veikloje dalyvauja šie objektai: paraiška, medžiaga, pastatas, naudotojas, medžiagų tiekėjas, tiekėjo medžiagos, medžiagų krepšelis, krepšelio elementas, klientas ir administratorius.



34 pav. Būsto renovacijos IS klasių diagrama

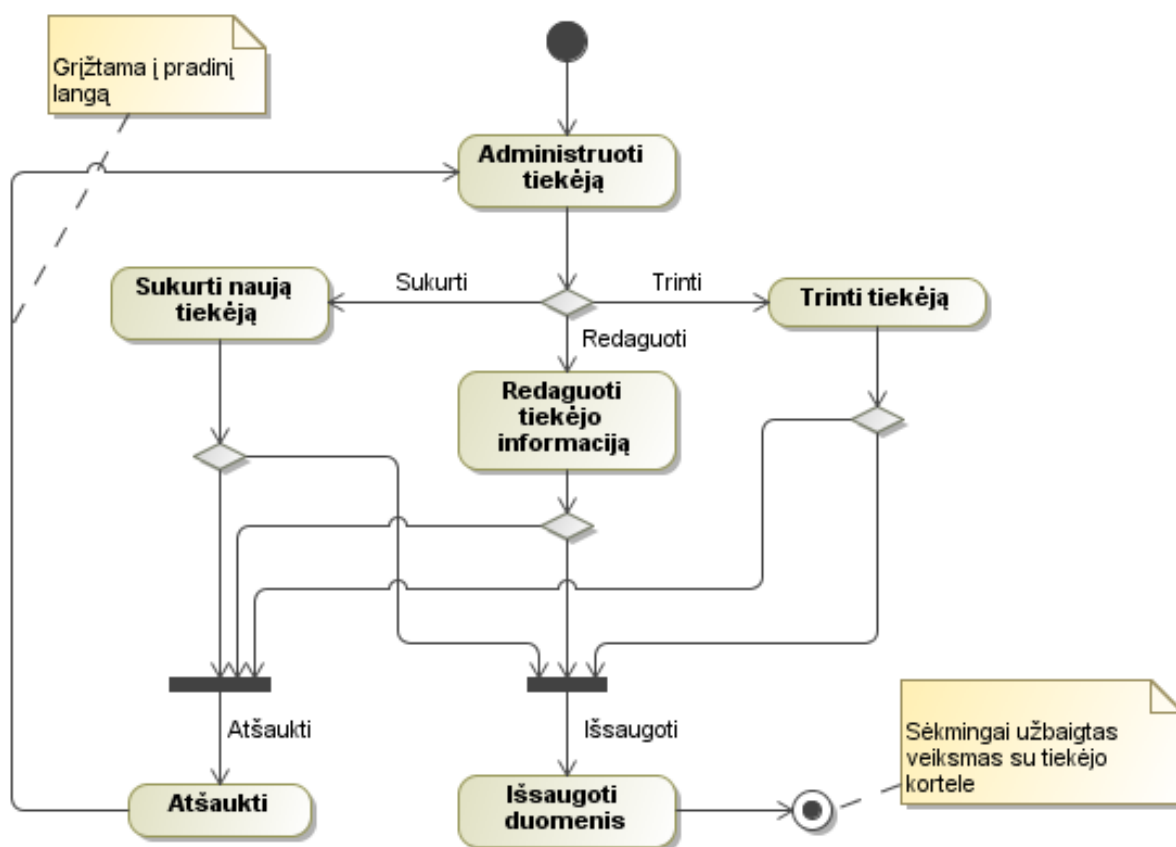
Ryšiai tarp kiekvienos klasės nurodo, kokios klasės sąveikauja tarpusavyje ir kokių būdu. Iš diagramos matyti, kad naudotojas kuria paraišką, o tarp šių klasių ryšys yra „1“ su daug. Tai reiškia, kad vienas naudotojas gali teikti daug paraiškų, o viena paraiška gali turėti tik vieną naudotoją. Tokia pati logika yra ir su kitomis klasėmis, priklausomai nuo jų ryšio, pvz., paraiška

gali turėti nulį arba vieną medžiagų krepšelį, nes kuriant naują paraišką, ji dar neturi parinktų medžiagų.

3.1.3. Veiklų diagrama

Veiklų diagrama nurodo naudotojo arba sistemos veiksmus, kuriuos vykdant galima pasiekti norimo rezultato. Veiklų diagrama gali būti skirtingo detalumo, pvz., veikla gali būti „redaguoti pastato informaciją“, o ši veikla gali būti detalizuota iki žingsnių, kuriuos reikia padaryti norint atlikti redagavimą.

Kadangi veiklų diagramos kiekvienai veiklai braižomos ir aprašomos individualiai, pateikta pavyzdinė diagrama, kuri nurodo veiklas su tiekėjo informacija (žr. 35 pav.).



35 pav. Būsto renovacijos IS tiekėjo administravimo veiklų diagrama

Tiekėjo administravime yra trys pagrindinės veiklos: sukurti naują tiekėją, redaguoti tiekėjo informaciją ir trinti tiekėją. Kiekviena iš šių veiklų turi veiksmus, kurie išsaugo įvestą informaciją arba atšaukia veiklos vykdymą ir neišsaugo jokių duomenų.

3.2. Vartotojo sąsajos prototipas

Demonstraciniams vartotojo sąsajos prototipams, kurie generuojami automatiškai, galima pritaikyti norimą dizainą. Kadangi prototipas kuriamas internetiniam puslapiui, kodas gali būti parašytas hiperteksto žymėjimo kalba (HTML, angl. *Hyper text Markup Language*) – tai kompiuterinė žymėjimo kalba, naudojama pateikti turinį internete. Internetinių puslapių, kurie parengti HTML formatu, dizainai lengvai keičiami naudojant CSS (angl. *Cascading Style Sheets*) – kalba, skirta nusakyti kita struktūrine kalba aprašyto dokumento vaizdavimą. Pritaikius būsto renovacijos IS prototipui žalios spalvos dizainą, o taip pat pakeitus kelis teksto šriftus, galima gauti rezultatą, kuris pateiktas žemiau esančiame paveikslėlyje (žr. 36 pav.).



36 pav. Būsto renovacijos IS vartotojo sąsajos prototipas

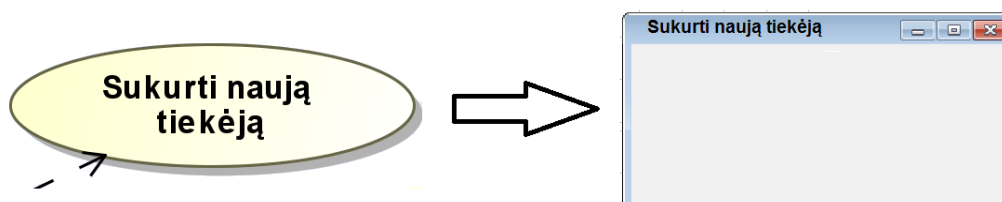
Iš vartotojo sąsajos prototipo matoma, kad jis atitinka pagrindinius išorinių naudotojų (klientas ir tiekėjas) panaudos atvejus, tokius kaip: pridėti įmonę, sukurti naują paraišką, prisijungti prie sistemos bei registruotis. Visas kitas funkcijas atlieka administratorius, o veikimo logika užrašyta sistemos kode.

Tokį IS vartotojo sąsajos prototipą galima teikti klientui kaip paveikslėlį, kuriame aiškiai matomas būsimos sistemos dizainas bei galimos atlikti funkcijos. Analizės metu nustatyta, kad prototipai su jiems pritaikytu dizainu užtrunka daug daugiau laiko sukurti bei patiriama daugiau išlaidų, tačiau klientas tokiu produktu bus daug labiau suinteresuotas nei matydamas pilkus sistemos langus.

3.3. Vartotojo sąsajos generavimo rezultatai

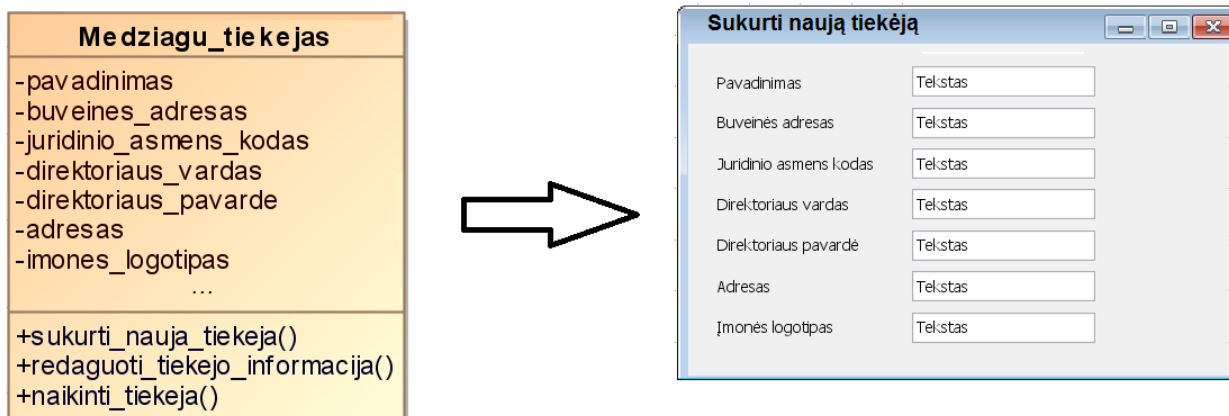
Projektavimo dalyje iš sistemos duomenų galima generuoti rezultatus – vartotojo sąsajos elementus. Kadangi baigiamojo darbo apimtis ribota, pavyzdiniam atvaizdavimui bus naudojamas medžiagų tiekėjo vartotojo sąsajos prototipų atvaizdavimas.

Pagal vartotojo sąsajos prototipų generavimo algoritmą, pirma reikia nustatyti su medžiagų tiekėju susijusius panaudos atvejus. Iš projektavimo dalies sistemos duomenų skyriuje pateiktos panaudos atvejų diagramos galima matyti, kad su medžiagų tiekėjo administravimu yra susiję šie panaudos atvejai: sukurti naują tiekėją, redaguoti tiekėjo informaciją, ištrinti tiekėją. Tiekėjo, kaip objekto, gyvavimas prasideda tuomet, kai į sistemą įvedamas naujas tiekėjas. Taigi, panaudos atvejis „Sukurti naują tiekėją“ atvaizduojamas kaip vartotojo sąsajos langas (žr. 37 pav.).



37 pav. Naujo vartotojo sąsajos prototipo lango generavimas iš panaudos atvejo

Vartotojo sąsajos langas iš pradžių yra tuščias, nes nėra tik antrame algoritmo žingsnyje yra nuskaitomi medžiagų tiekėjo atributai. Medžiagų tiekėjo įvedimo forma turi septynis atributus, kurių kiekvienas pateikiamas iš eilės nuo viršaus į apačią (žr. 38 pav.).



38 pav. Vartotojo sąsajos prototipo lango užpildymas klasės atributų laukais

Trečiame algoritmo žingsnyje imami veiklų diagramos žingsniai, kurie galimi su naujo tiekėjo įvedimo forma. Šie žingsniai yra trys: išsaugoti, atmesti ir pasirinkti. Visi trys yra išplečiantys pagrindinį žingsnį „Inicijuoti naujo tiekėjo sukūrimą“, nes sistemos naudotojas turi

galimybę atlikti bet kurį iš šių žingsnių, kad koku nors būdu užbaigti įvedimą. Išsaugojimas atliks įvestos formos saugojimą, atmetimas – jos uždarymą, o pasirinkimo veikslas leis naudotojui pasirinkti norimą tiekėjo logotipą iš lokalios kompiuterio direktorijos.

Jei naudotojas pasirenka logotipą ir išsaugo naujo tiekėjo įvedimo formą, sistema pateikia tiekėjo kortelės langą, kuris užpildytas visa naudotojo suvesta informacija. Šis langas yra peržiūros režime, todėl su juo galimi redagavimo ir ištrynimo veiksmai.



Tiekėjo kortelė	
Logotipas	
Pavadinimas	UAB "ARGINTA"
Buveinės adresas	Molėtų plentas, Vilnius 14259
Juridinio asmens kodas	120930752
Direktoriaus vardas	Artūras
Direktoriaus pavardė	Mrzauskas
Uždaryti Pridėti medžiagas	

39 pav. Tiekėjo kortelės vartotojo sąsajos lango prototipas

Tokį prototipą galima laikyti baigtiniu, nes dizaino pritaikymas jau kainuoja nemažai laiko ir resursų, o iš funkcionalumo pusės niekas nesikeičia. Dizainas gali būti derinamas vėliau, kai sistema jau atitinka funkcinis reikalavimus.

IŠVADOS IR PASIŪLYMAI

1. IS kūrimo metu yra būtina skirti daug dėmesio galutiniam sistemos naudotojui ir projekto darbų kritinėje grandinėje užtikrinti kuo didesnius laiko buferius.
2. Modeliavimas išlieka pats efektyviausias būdas pateikti norimą informaciją. Geriausiai tai atvaizduoja UML diagramos.
3. Labai svarbu kuo anksčiau supažindinti klientą su projektuojama sistema. Tam geriausia naudoti vartotojo sąsajos prototipus.
4. Net 90,75% žmonių naudoja *Microsoft Windows* operacines sistemas, todėl rekomenduojama, kad vartotojo sąsajos prototipai atitiktų *Windows* standartus.
5. Pastebima, kad IS kūrimo proceso projektavimo etape sugaištama daug brangaus laiko ir resursų modeliuojant vartotojo sąsajos prototipus. Reikia įrankio, kuris leistų generuoti vartotojo sąsaja automatiškai, pasinaudojant modeliavimo diagramomis.
6. Vartotojo sąsajos prototipų generavimui būtina sukurti generavimo algoritmą, kuris naudotų UML diagramas kaip pagrindą prototipo sukūrimui.
7. Siūlomam generavimo algoritmui, kaip įėjimo duomenys, reikalingos trys UML diagramos: panaudos atvejų, klasių ir veiklų. Rezultate gaunamas internetinės vartotojo sąsajos prototipas, kuris parašytas HTML programavimo kalba.
8. Palyginus keturis rinkoje populiarius vartotojo sąsajos modeliavimo įrankius, labiausiai reikalavimus atitiko *MagicDraw UML*, kurio privalumai pranoko kitus produktus.
9. Įrankis, kuris generuotų vartotojo sąsajos prototipus iš UML diagramų, ne tik sutaupyto daug brangaus projektavimo etapo laiko, bet ir esant tam tikriems reikalavimų pakitimams, automatiškai atnaujintų ir patį prototipą.
10. Automatizuotas vartotojo sąsajos prototipų generavimo algoritmas neturi reikalauti iš naudotojo įvesti daug papildomų duomenų, nes kitu atveju, bus paprasčiau kurti prototipą rankiniu būdu.
11. Pasinaudojant tyrimo rezultatais, suprojektuota būsto renovacijos informacinė sistema, kuri sugeneruota naudojantis prototipo generavimo algoritmu.

LITERATŪROS SĄRAŠAS

- Čaplinskas A. 2010. Programų sistemų inžinerija. Paskaitų skaidrės.
- Čeponienė Lina. 2008. Informacinių sistemų projektų vadybos technologijos. [Žiūrėta: 2014-05-24]. Prieiga internetu<http://darbai.jaunareklama.lt/agro/site/files/failai/1_paskaita.pdf>
- Dolors Costal, Cristina Gómez, Anna Queralt, Ruth Raventós, Ernest Teniente. 2008. Improving the definition of general constraints in UML. Springer-Verlag.
- E. Marchetti. 2005. Automatic XML schema generation from UML application profile. E&I elektrotechnik und informationstechnik. ScienceDirect.
- Hans-Erik Eriksson and Magnus Penkeri. 2000. Business Modeling With UML: Business Patterns at Work. John Wiley & Sons Inc.
- Jeffrey Whitten; Lonnie Bentley. 2007. Systems Analysis and Design Methods.
- Jesu' s M.Almendros-Jimenez, Luis Iribarne. 2008. An extension of UML for the modeling of WIMP user interfaces. ScienceDirect. Journal of Visual Languages and Computing 19: 695–720.
- John W. Satzinger, Robert B. Jackson, Stephen D. Burd. 2007. Systems analysis and design in a changing world. Course Technology.
- Jonathan Simon. 2005. Interaction Happens: Prototyping Techniques. [Žiūrėta: 2014-03-24]. Prieiga internetu <<https://today.java.net/pub/a/today/2005/08/23/prototyping.html>>
- Lidija Jankauskienė. 2014. Kodėl stringa daugiabučių renovacija. LRT Televizijos Naujienų tarnyba. [Žiūrėta: 2014-05-24]. Prieiga internetu <<http://www.delfi.lt/verslas/nekilnojamas-turtas/kodel-stringa-daugiabuciu-renovacija.d?id=64900245>>
- M. Adler, A. Bers, C. Broxton, M. Caccamo, K. Chu, F. Rovitto. 1997. „Developer/2000: Guidelines for Building Applications“.
- Mohammed Elkoutbi, Ismail Khriess, Ruolf K. Keller. 2006. Automated Prototyping of User Interfaces Based on UML Scenarios. Springer Science + Business Media, Inc. Manufactured in The Netherlands.
- No Magic. 2014. Apie įmonę. [Žiūrėta: 2014-03-24]. Prieiga internetu <<http://www.nomagic.lt/dispatcher.php?lang=1&item=57&arg=>>>
- OMG „Unified Modeling Language TM“ (OMG UML), Superstructure, Version 2.2. 2012.
- Scott Berkun. 2000. The art of UI prototyping. [Žiūrėta: 2014-05-24]. Prieiga internetu <<http://scottberkun.com/essays/12-the-art-of-ui-prototyping/>>

SparxSystems. 2014. About. [Žiūrėta: 2014-03-24]. Prieiga internetu <<http://www.sparxsystems.com.au/about.html>>

T.R.G. Green, Cognitive dimensions of notations, in: Proceedings of the Fifth Conference of the British Computer Society, Human-Computer Interaction Specialist Group on People and Computers V, Cambridge University Press, New York, NY, USA, 1989, pp. 443–460.

Trevor Hobson. 2014. HTML5 element list. [Žiūrėta: 2014-03-24]. Prieiga internetu <https://developer.mozilla.org/en/docs/Web/Guide/HTML/HTML5/HTML5_element_list>

Wan Timah. 2011. Prototyping model. [Žiūrėta: 2014-05-24]. Prieiga internetu <<http://csebrules.blogspot.com/2011/01/assignment-2-task-2-prototyping-model.html>>

Wikipedia. Unified Modeling Language (UML). [Žiūrėta 2013-10-13] interneto prieiga: <http://lt.wikipedia.org/wiki/Unified_Modeling_Language>.

Wikipedia. Usage share of operating systems. [Žiūrėta: 2014-05-24]. Prieiga internetu <http://en.wikipedia.org/wiki/Usage_share_of_operating_systems>

PRIEDAI

Priedas Nr. 1. Straipsnis, pristatytas 2013 m. gegužės 24 d. Kaune vykusioje konferencijoje „Inžinerinė ir kompiuterinė grafika“

PAŽANGESNIS BŪDAS ĮGYVENDINTI PIKSELIŲ PAGRISTĄ ATVIRKŠTINĖS INŽINERIJOS VARTOTOJO SĄSAJOS STRUKTŪRĄ

Deividas Celencevičius

Vilniaus Gedimino technikos universitetas

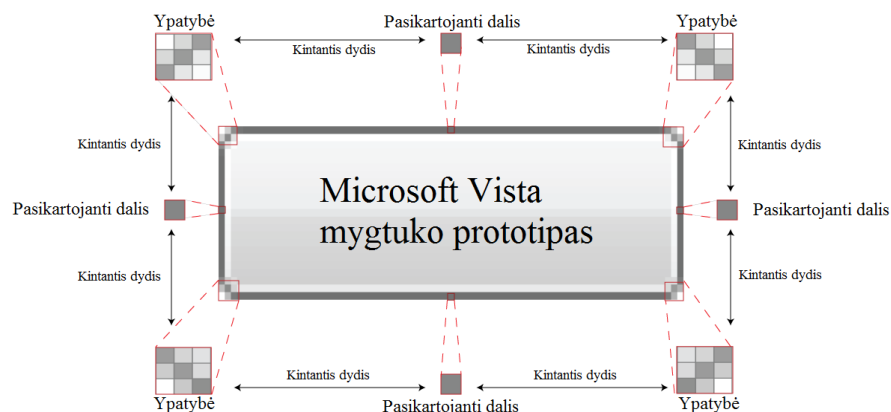
1. Įvadas

Patogus ir lankstus vartotojo sąsajos atvaizdavimas yra beveik kiekvienos informacinės sistemos (toliau – IS) tikslas. Tačiau reta IS suteikia vartotojui galimybę keisti ar kitaip modifikuoti esamą grafinę vartotojo sąsają (GUI, angl. *Graphical User Interface*), dėl to visi vartotojai yra priversti naudotis bendra, gamintojo sukurta, vartotojo sąsaja. Pirmiausia buvo ieškoma būdo, kuris leistų keisti esamą vartotojo sąsają, o vėliau pavyko rasti ir įrankį, kuris atlieka norimus pakeitimus. Analizuojant probleminę sritį buvo padarytos išvados, kad kiekvienoje grafinėje vartotojo sąsajoje objektai atvaizduojami pikselių pagalba, o tam, kad keisti jau sukurtus objektus ar pridėti papildomo funkcionalumo, reikės pasinaudoti atvirkštinės inžinerijos principu. Šiame straipsnyje pateikiami keli pavyzdžiai, kurie galėtų išspręsti grafinės vartotojo sąsajos struktūros keitimo problemas.

2. Objekto atpažinimas pagal pikselius ir jo radimas pagal pelės žymeklį (Bubble cursor)

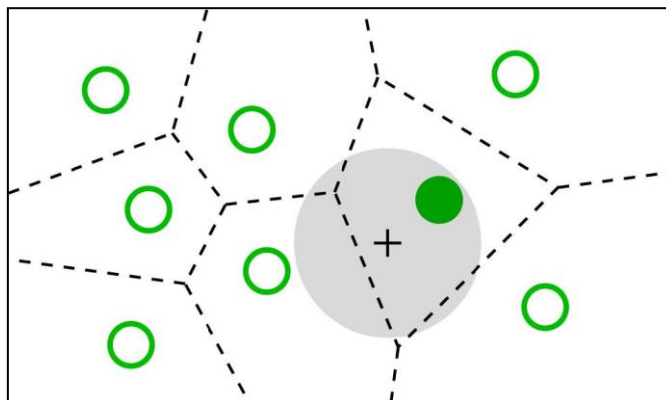
Kiekvienas grafinės vartotojo sąsajos objektas yra atvaizduojamas pikselių pavidalu, kas leidžia kiekvieną iš jų identifikuoti. Dažniausiai IS objektai kuriami pagal iš anksto paruoštus šablonus, todėl juos galima skirstyti į tam tikras grupes, pvz.: kiekvienas Windows operacinės sistemos valdymo mygtukas sukurtas pagal šabloną (skiriasi tik jų dydžiai), todėl identifikavus vieno iš šių mygtukų kraštinių pikselius (I pav.), galima identifikuoti visus tos grupės mygtukus.

Paveikslėlyje pavaizduotas Microsoft Vista operacinės sistemos mygtuko prototipas, kurio visų kraštinių pikselių atvaizdavimo ypatybės yra identifikuotos. Iš paveikslėlio matome, kad mygtukas yra stačiakampis, tai reiškia, kad jis turės 4 kampus ir 4 kraštines. Kiekvienas kampas sudarytas iš skirtingų 9 pikselių mozaikos, o kraštinės yra sudarytos iš vieno pikselio, kuris nuolatos pasikartoja nuo vieno kampo iki kito, kinta tik kraštinės ilgis. Todėl šį mygtuko prototipą galime identifikuoti iš 40 pikselių [2]. Identifikavus objektą pagal jo pikselius, galime sužinoti objekto dydį ir vietą vartotojo sąsajos lange, kas leidžia sukurti papildomo funkcionalumo esamam objektui.



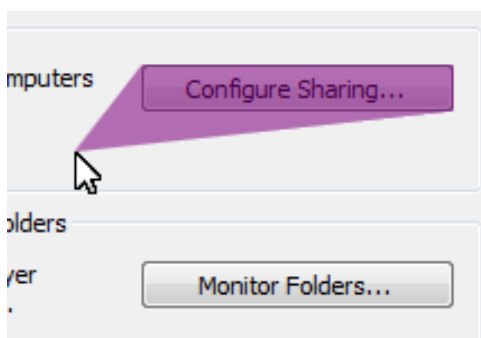
I pav. Microsoft Vista operacinės sistemos mygtuko prototipas su ypatybėmis [2]

Artimiausio objekto radimas, kitaip vadinamas „burbulinis žymeklis“ (angl. *Bubble cursor*), kurio principas yra pagal pelės žymeklį rasti jam artimiausius objektus (II pav.). Kadangi atlikome mygtuko identifikavimą pagal jo pikselių sudėtį, galime pasinaudoti *bubble cursor* metodu, kad rastume kelis objektus, kurie išsidėstę vartotojo sąsajos lange. Pelės žymeklis yra kaip nematomas apskritimas, kuris gali per atstumą užfiksuoti sau artimiausią objektą. Kiekvienas objektas turi savo ribas, kurios negali kirstis su kito objekto ribomis. Kai pelės žymeklio apskritimas patenka į kurio nors objekto ribas, jis jį užfiksuoja. Jei reikia rinktis tarp dviejų ar daugiau objektų, pasirenkamas tas, kurio ribose yra didžiausia apskritimo ploto dalis.



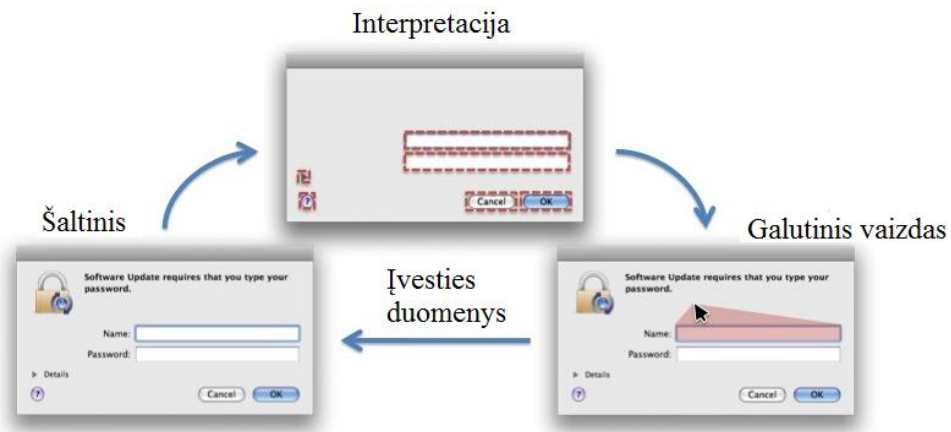
II pav. Grossman'o ir Balakrishnan'o burbulinis žymeklis užtikrina, kad artimiausias objektas būtų pažymėtas [1]

Sudėjus objekto identifikavimą, paieška ir funkcinių objektų atpažinimą vartotojo sąsajos lange, galima gauti rezultatą, kuris iš dalies pakeičia esamos sąsajos atvaizdavimą. Pavyzdyje pateikiamas šių trijų funkcijų gautas rezultatas – artimiausio objekto radimas ir pakeista pikselių pagrįsta atvirkštinės inžinerijos vartotojo sąsajos struktūra (III pav.).



III pav. Pakeistas grafinės vartotojo sąsajos atvaizdavimas [2]

Pavyzdyje pateiktas rezultatas buvo gautas atliekant šiuos žingsnius: 1) Padaroma esamos vartotojo sąsajos ekrano nuotrauka „bitmap“ formatu, 2) atpažistami ekrane esantys objektai, 3) atvaizduojamas pakeistas ekrano vaizdas (realus vartotojo sąsajos vaizdas paslepiamas naudojant virtualaus ekrano metodus), 4) įvesties duomenys susiejami su realia vartotojo sąsaja, 5) sugeneruojami nauji išeities duomenys, kuriuos vartotojas atlieka, ir atnaujinamas vartotojo sąsajos ekrano vaizdas (IV pav.).



IV pav. Esamos grafinės vartotojo sąsajos keitimo mechanizmas

Toks grafinės vartotojo sąsajos pakeitimas galėtų labai palengvinti vartotojų darbą su kompiuteriu, ypač tiems, kurie turi judėjimo negalią, pvz.: Parkinsono ligą, cerebrinį paralyžį ar raumenų distrofiją. Taip pat yra daugybė kitų galimybių, kaip galima pakeisti grafinę vartotojo sąsajos struktūrą naudojant pikselių pagrįstą atvirkštinės inžinerijos principą.

3. Išvados

- Programinių įrankių ir metodų pagalba galima pakeisti bet kokią grafinę vartotojo sąsają.
- Kiekvienas grafinės vartotojo sąsajos objektas atvaizduojamas pikselių pagalba, todėl galima identifikuoti kiekvieną iš tų objektų.
- Informacinių sistemų ir programų objektai dažniausiai kuriami pagal iš anksto paruoštus šablonus, todėl galima sukurti taisykles, pagal kurias būtų galima atpažinti kiekvieną grupę, kuri sudaryta iš tam tikrų objektų.
- Keičiant grafinę vartotojo sąsają galima padėti kompiuterių vartotojams paprasčiau naudotis jiems pateikiama informacija.

Literatūra

1. Grossman, T. and Balakrishnan, R. (2005). The Bubble Cursor: Enhancing Target Acquisition by Dynamic Resizing of the Cursor's Activation Area. *Proc. of the ACM Conference on Human Factors in Computing Systems (CHI 2005)*, 281-290.
2. Dixon, M. and Fogarty, J. (2010). Prefab: Implementing Advanced Behaviors Using Pixel-Based Reverse Engineering of Interface Structure. Computer Science & Engineering DUB Group, University of Washington.

Implementing advanced behavior using pixel-based reverse engineering of interface structure Deividas Celencevičius

Summary

This article analyze graphical user interface problem, when user can't change default GUI for their needs. Article represents a first steps towards pixel-based reverse engineering of interfaces based on models of how those interfaces are painted by applications and user interface toolkits. This paper intentionally presents the simplest description of using a prototype library to reverse engineer an image, focusing on information flow as toolkit locates features, generates hypotheses, and tests those hypotheses.