KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Marius Pečiulaitis

Automatizuotas internetinių IS vartotojo sąsajų kūrimas

Magistro darbas

Darbo vadovas: prof. dr. Rimantas Butleris
Konsultantas: lekt. mag. Tomas Danikauskas

Kaunas, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Marius Pečiulaitis

Automatizuotas internetinių IS vartotojo sąsajų kūrimas

Magistro darbas

Recenzentas
doc. dr. E. Karčiauskas
2007-01-09

Vadovas
prof. dr. R. Butleris
2007-01-09

Atliko
IFM-1/4 gr. stud.
M. Pečiulaitis
2007-01-09

Kaunas, 2007

Turinys

Paveikslų turinys

Lentelių turinys

1. Įvadas

Viena iš programinės įrangos dalių, kuri tiesiogiai įtakoja vartotojo darbo su sistema kokybę ir komfortabilumą yra grafinė vartotojo sąsaja. Dažnai be pagrindinių reikalavimų sistemos funkcionalumui vartotojai iškelia reikalavimus ir sistemos naudojimo patogumui. Vis dar kyla nesusipratimų, kai programinės įrangos kūrėjai pagrindinį dėmesį skiria struktūriškai teisingam ir logiškam programinės įrangos kodo parašymui ir praktiškai mažąja dalį resursų skiria teisingo – sistemos vartotojų poreikius atitinkančios vartotojo sąsajos kūrimui [3]. Realybė yra tokia, kad tinkamai sukurta vartotojo sąsaja leidžia žmonėms daug greičiau perprasti darbo su sistema specifiką.

Grafinės vartotojo sąsajos kūrimas yra svarbus dėl kelių priežasčių. Pirmiausia kuo ji intuityvesnė, tuo vartotojui lengviau ja naudotis, ir kuo lengviau ja naudotis tuo pigiau sistema atsieina, nes tai sumažina išlaidas apmokymams. Kuo suprantamesnė jūsų PĮ vartotojo sąsaja, tuo didesniam vartotojų skaičiui bus malonu ja naudotis, kas didins jų pasitenkinimą darbu, kurį jūs davėte. PĮ, kurią dėl nepatogios vartotojo sąsajos sudėtinga naudoti gali tiesiog neprigyti numatytoje veikloje nepriklausomai nuo jos techninio tobulumo ar teikiamo funkcionalumo. Pagrindiniai aspektai į kuriuos reiktų atkreipti dėmesį kuriant grafinę vartotojo sąsają yra [5]:

- *Pastovumas*. Stengtis mygtukus skirtinguose programos languose išdėstyti pastoviose vietose, jei viename lange galima pavyzdžiui iš sąrašo išsirinkti elementą ant jo spragtelėjus du kartus pele, ta pačią galimybę reik įdiegti ir kituose programos vietose. Taip vartotojas mintyse greičiau sukurs bendrą vaizdą, kaip sistema veikia.
- *Nustatyti standartus ir jų laikytis*. Norint išlaikyti pastovumą sistemoje reikia visoje laikytis apibrėžtų dizaino standartų. Geriausia pritaikyti industrinius, tokius kaip IBM ar Microsoft.
- *Sekti kontrasto taisykles*. Šviesiame fone naudoti tamsų tekstą ir atvirkščiai.
- *Neslėpti neaktyvių kontrolės elementų*. Padaryti juos pilkais. Taip vartotojas greičiau supras kaip sistema veikia. Žinos kokie veiksmai jam leidžiami ir kokie ne.
- Lygiuoti laukus. Kai lange daugiau nei vienas koregavimo laukas, juos reikėtų viena kryptimi.
- *Nekurti sudėtingų langų*. Perkrautais langus sudėtinga suprasti ir jais naudotis.
- *Efektyviai grupuoti elementus*. Tie kurie yra logiškai susiję turėtų būti vienas šalia kito, kiti atskirai.
- *Iššokantys meniu neturėtų būti vienintelis funkcijų vykdymo šaltinis*. Vartotojai negali mokytis naudotis sistema, jei dauguma galimų funkcijų yra paslėpta.

Kiekvienais metais sukuriama daugybė informacinių sistemų, tad toks didelis pasirinkimas tapo pagrindiniu iššūkiu interaktyvios programinės įrangos kūrėjams. Vartotojai reikalauja, kad informacija bei funkcijos būtų pasiekiamos vienodai patogiai ir efektyviai, nepriklausomai nuo sistemos, net jei ir ji

ar aplinka bei duomenys ar jų pobūdis dinamiškai keičiasi. Prie viso to, vartotojams svarbu, kad turima sistema jie galėtų naudotis nepriklausomai nuo naudojamos tiek techninės, tiek programinės įrangos ir jos pokytis neįtakotų programinės įrangos funkcionalumo bei vartotojo sąsajos ypatumų.

Taigi šio projekto tikslas yra sukurti sistemą optimaliai parenkančią multi platforminės grafinės vartotojo sąsajos įgyvendinimą tiek dizaino tiek programinio kodo atžvilgiu atsižvelgiant į apdorojamų duomenų specifikaciją.

2. Vartotojo sąsajos kūrimo metodų analizė

2.1. Tyrimo sritis, objektas ir problema

Kurti programinę įrangą, nors ir naudojamos naujausios technologijas, suteikiančios daug galimybių ir sprendimų, tačiau iškyla problemų – programinės įrangos kūrėjams tai pasireiškia kaip uždavinys sukurti keletą versijų vienos programos norint ją pritaikyti prie konteksto pokyčių. Programuoti skirtingas programos versijas skirtingoms operacinėms sistemoms, reiškia papildomas sąnaudas, tiek IS kūrimui tiek tolimesniam jos palaikymui. Prie viso to, šiuo metu naudojami IS kūrimo įrankiai suteikia mažai galimybių kurti programas, kurios dinamiškai keičiasi priklausomai nuo jų veikimą įtakojančios aplinkos – funkcinių ir nefunkcinių reikalavimų pokyčio, duomenų struktūros bei pobūdžio ir pan. Menkiausias pakeitimas naudojamų duomenų struktūroje reikalauja programuotojų įsikišimo – sistemos perdarinėjimo. To pasekmės - išardoma pradinė PĮ architektūra, perprogramavimas reikalauja papildomų laiko, žmogiškųjų ir finansinių sąnaudų.

Programinės įrangos su grafine vartotojo sąsaja kūrimas šiandiena neįsivaizduojamas be aplikacijų programavimo sąsajos (API) [2]. Kadangi toks kūrimo procesas reikalauja daug programinio kodo, tyrinėtojų pastangos nukrypo į tikslą sukurti deklaratyvų grafinės vartotojo sąsajos tekstinį aprašymą. Deja daugelis bandymų šioje srityje yra apriboti iki sintaksės lygio ir apsiriboja specifiniais įrankiais, todėl reikia tolesnių tyrimų, kurie įrodytų, jog grafinės sąsajos kūrimą galima gėrokai supaprastini ir pagreitinti.

Pastaruoju metu PĮ industrijoje skiriamas vis didesnis dėmesys ne tik standartizuoti informacijos tarp programų apsikeitimo metodą, bet ir suformuoti modelį, kuris įgalintų programinę įrangą sąveikauti interakcijos duomenų lygmenyje t.y. duomenimis, kurie aprašo ir susieja grafikos elementus.[apie XIML].

2.2. Analizės metodų, priemonių parinkimas

Darbo tikslas - surasti tinkamiausią metodą grafinės vartotojo sąsajos automatizuotam kūrimui. Tad analizės metodo esmė – atrinkti tinkamiausią būdą generuoti XML pavidalu pateiktus duomenis į dinamiškai besikeičiančią grafinę vartotojo sąsają.

Pagrindiniai aspektai į kuriuos atsižvelgiama atliekant metodų analizę:

- sugeneruotos grafinės vartotojo sąsajos (ar jos specifikacijos) nepriklausomumas nuo platformos
- programavimo/žymių kalbos paprastumas

- automatiškai sugeneruoto programinio kodo optimalumas (funkcionalumo ir apimties atžvilgiu)
- suformuojamos GUI atvaizdavimo pastovumas skirtingose naršyklėse

2.3. Organizacijos veiklos analizė

Sistema kuriama informacijos sistemų projektavimo mokslo grupės užsakymu. Pradžioje ji bus naudojama eksperimentiniams tikslams, o vėliau mokymo procesui bei realiam taikymui.

2.4. Pasaulio literatūros šaltiniuose pateiktų grafinės vartotojo sąsajos generavimo uždavinio sprendimų lyginamoji analizė

Daugelis metodų skirtų specifikuoti reikalavimus programinei įrangai bei projektuoti PĮ, bet mažai dėmesio skiriama išsamiai vartotojo sąsajos kūrimo metodologijai.

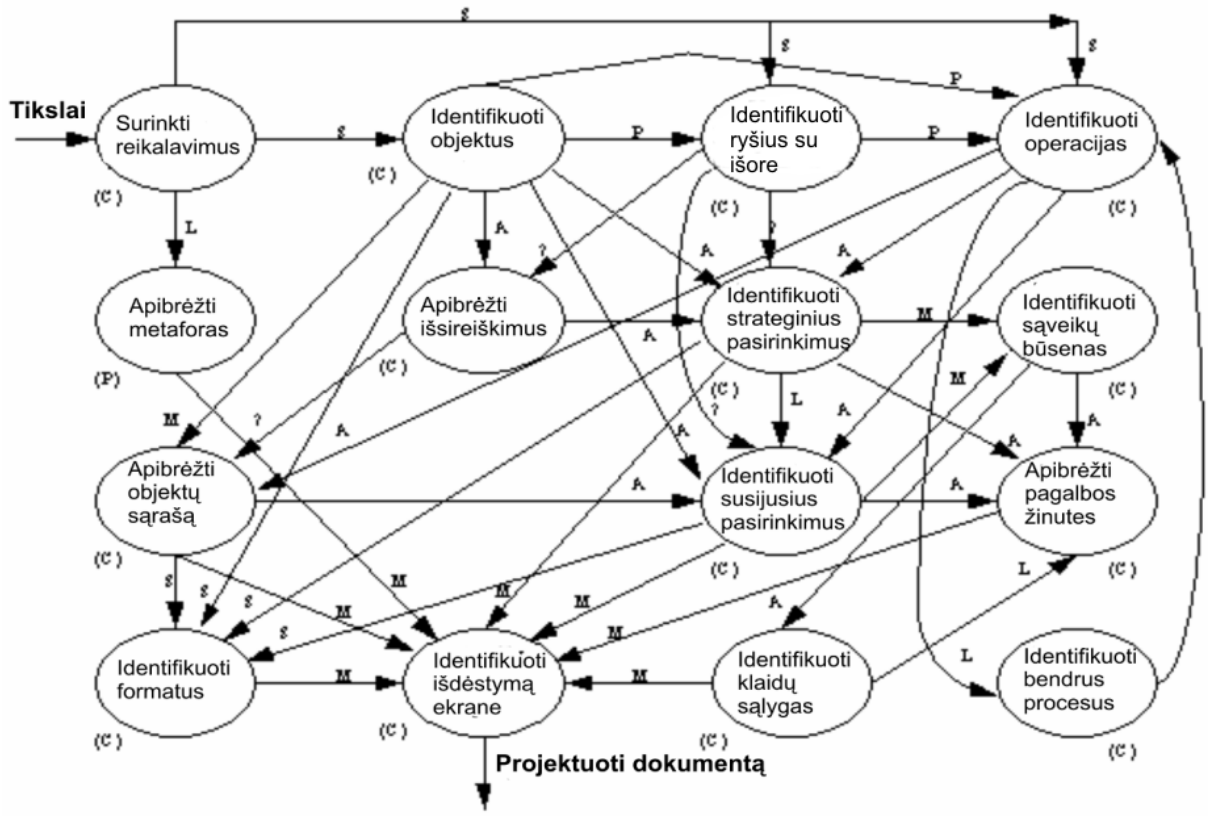
Vartotojo sąsajos projektavimas gali būti suskaidomas į tris pagrindines fazes:

- planavimas ir idėjos iškėlimas,
- projektavimas ir prototipo kūrimas,
- kokybės įvertinimas.

Dauguma projektavimo įrankių palaiko viską išskyrus planavimą bei idėjos iškėlimą, kai tuo tarpu projektavimas reikalauja išradingumo, projektavimo žinių ir patirties, užduočių analizavimo ir visapusiško vartotojų reikalavimo supratimo. Pavyzdžiui GUI projektavimo metodologijoje ***Balasubramanian ir Turoff*** išskiriama 15 užduočių [7]:

- Identifikuoti PĮ reikalavimus.
- Identifikuoti metaforą ar metaforas.
- Identifikuoti objektus, kurie sudaro PĮ.
- Identifikuoti šalutines klasifikacijas ar ryšius tarp objektų.
- Identifikuoti veiksmus ar funkcijas, kurias gali atlikti objektai. Tai gali būti bendri veiksmai, aiškiai nusakyti veiksmai arba valdymo funkcijos.
- Identifikuoti modifikatorius ar filtrus, kurie pažymi objektų posistemes.
- Identifikuoti strateginius pasirinkimus, kurie leidžia vartotojui pritaikyti sąveiką specifinių tikslų atlikimui.

- Identifikuoti objektų formatus, objektų dalis, meniu ir t.t.
- Identifikuoti objektų, kurie yra semantiškai susiję arba paremti specifiniais pasirinkimų kriterijais, sąrašą.
- Identifikuoti reaguojančius pasirinkimus, kurie gali būti atlikti sąrašo ar objektų rinkinio. Tai pasirinkimai, kurie numato pilną specifinių tikslų vykdymą.
- Identifikuoti besidalinančius procesus ar funkcijas.
- Identifikuoti vartojo sąveikos būsenas.
- Identifikuoti būtiną pagalbą per visą sistemą.
- Identifikuoti galimus pranešimus (atgal vartotojui) ir klaidų sąlygas .
- Identifikuoti ekrano schemą – darbo plotą, valdymo sritį, būsenų sritį, pranešimų sritį.

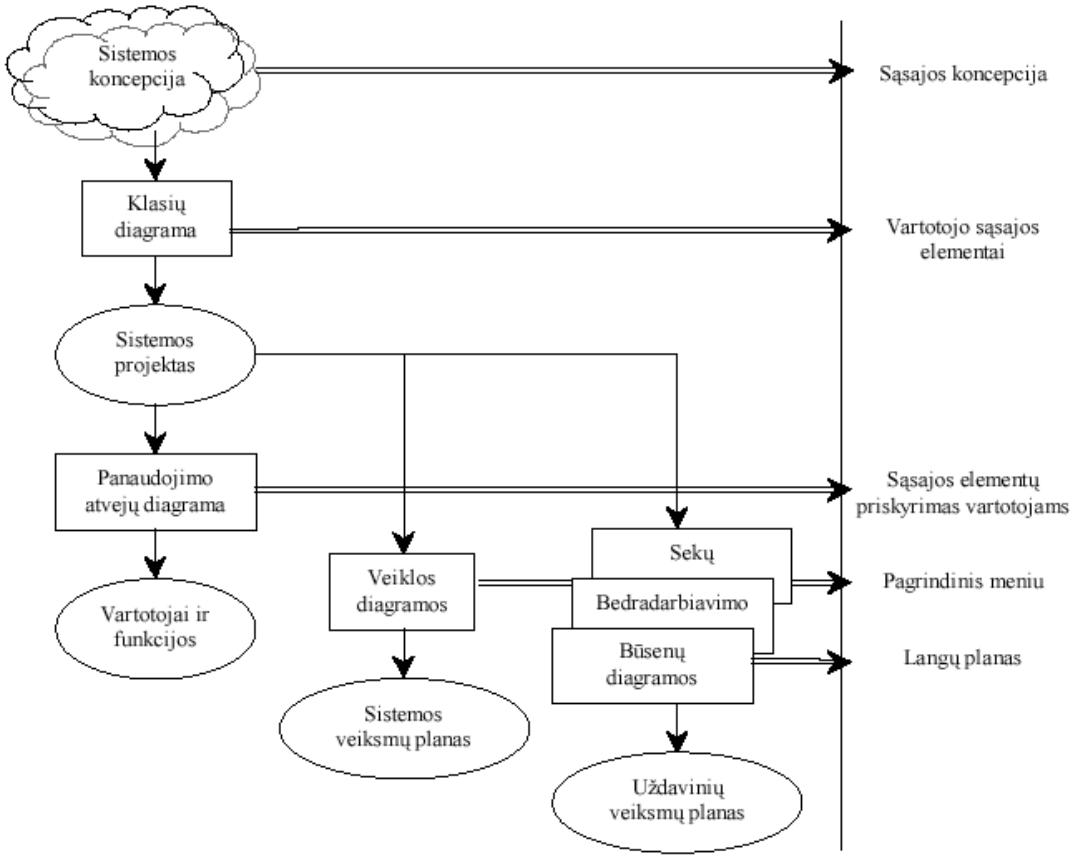


1 pav. Vartotojo sąsajos projektavimo užduotys

Kadangi vartotojo sąsajos projektavimas kūrybiškas procesas, jis negali būti nuoseklus ir užduočių vykdymo tvarka nėra tokia svarbi kaip jų išbaigtumas. Įrankis naudojantis šią metodologiją pateikia projektuotojui galimus aspektus į kuriuos jam derėtu atsižvelgti tam tikrame projektavimo etape ir leidžia sudaryti projektavimo procesą savo nuožiūra tuo pačiu užtikrindamas logišką nuoseklumą.

Dauguma sąsajos generavimo įrankių padeda sukurti prototipą, sukonstruoti konkrečius dizaino elementus, tačiau projektuotojui neleidžia suprasti, koks jo veikimo principas. Bet jeigu būtų skiriama daugiau dėmesio renkant funkcinius ir nefunkcinius reikalavimus - esminę projektavimo informaciją formaliu būdu, generavimo įrankiui būtų galima pateikti aukštesnio lygio specifikacijas.

Kita galima projektavimo metodika yra ***UML*** (Unified Modeling Language)- tai programų sistemų projektavimo rezultatų (artefaktų) vaizdavimo, specifikavimo, konstravimo ir dokumentavimo kalba.[20] UML apibrėžia 12 diagramų tipų [10], suskirstytų į tris kategorijas: keturi diagramų tipai (klasių, objektų, komponentų, įdiegimo diagramos) vaizduoja statinę struktūrą, penki (panaudojimo atvejų, sekų, veiklos, bendradarbiavimo, būsenų diagramos) – skirtingus dinaminės elgsenos aspektus, trys (paketai, posistemės, moduliai) – modulių organizavimo ir valdymo būdus. Šiame formalių uždavinio procesus aprašančių diagramų rinkinyje įžvelgiamas ėjimą nuo uždavinio koncepcijos iki duomenų bazės schemos ir uždavinio sprendimo algoritmo formalaus aprašymo. **2 pav.** parodytas galimas šių diagramų apjungimas projektavimo procese ir išskirti sąsajos projektavimo žingsniai, kurie turėtų būti atlikti bendroje sistemos projektavimo veiksmų sekoje. Klasių diagrama yra pradinė ir pagrindinė, nes jai sudaryti reikia žinoti uždavinio formuluotę ir esminius struktūrinius elementus. Ši diagrama leidžia aiškiai suprasti, ko yra siekiama uždaviniu ir konceptualiai parodyti ryšiai tarp uždavinio struktūrinių elementų (klasių). Klasių diagramą galima laikyti apibendrintu sistemos projektu, todėl ją kuriant paaiškėja atskiri vartotojo sąsajos elementai. Sudarius klasių diagramą, galima vystyti sistemos aprašymą panaudojimo atvejų (angl. *use case, verte aut.*) diagramą, kurioje sistema yra paskirstoma “menamiems” vartotojams įsivaizduojant jų funkcijas sistemoje. Taigi, šitokia diagrama leidžia įsivaizduoti atskirų vartotojų sąsajas, bet tik apibendrintame lygyje. Sistemos projektas, aprašytas klasių diagramose leidžia detalizuoti sprendžiamais uždaviniais (veiklos diagrama) ir atlikti šių uždavinių detalizaciją (būsenų, sekų, bendradarbiavimo diagramos). Šios keturios diagramos vadinamos elgsenos diagramomis. Pirmoji leidžia sudaryti pagrindinį sąsajos meniu, o kitos trys – vartotojo sąsajos langų planą.



2 pav. UML ir vartotojo sąsajos projektavimas

Tačiau naudojant UML vartotojo sąsajos modeliavimas nėra toks paprastas kaip norėtųsi. UML neaprašo tiksliai ryšių tarp panaudojimo atvejų bei veiklos diagramų. Taip pat neturi notacijos apibrėžti abstraktų pateikimą.

2.5. Modeliais grindžiamas vartotojų sąsajos projektavimas

Modeliais grindžiamas vartotojų sąsajos projektavimas yra aiškus konceptualių abstrakcijų žymėjimas. Šis formalus aprašymas gali būti naudojamas valdyti vartotojo sąsajai jos naudojimo metu.

Pagrindinis modelių privalumas yra jų aprašomoji galia bei aukštas abstrakcijos lygis [9]. Kitas privalumas yra tas, jog jie valdo naują aukšto lygio įrankių kartą pavyzdžiui vartotojo sąsajos generatoriai, transformatoriai bei pagalbos generatoriai.

Bet taip pat toks projektavimas turi ir savo trūkumų. Viena iš problemų – modeliais paremtas aukšto lygio abstrakcionizmas įtakoja tai, jog dizaineriai turi suprasti tas abstrakcijas. Tai

reikalauja abstraktaus mastymo, noro praleisti didelę laiko dalį studijuojant tam tikros modeliavimo kalbos abstrakcijas prieš tai, kai bus pradėta kurti vartotojo sąsaja. Tai kontrastas palyginus su vizualiai paremtais žemesnio lygio vartotojo sąsajos kūrimo įrankiais tokiais kaip Dreamweaver ir pan., kuriuos galima įsisavinti ir išmokti valdyti per keletą minučių.

Susijusi problema yra poreikis naudoti formalią kompiuterinę kalbą. Taigi, prie viso sudėtingumo dirbant su labiau abstrakčiu lygiu, projektuotojas turi valdyti reikiamas abstrakcijas reikalaujamas kompiuterio.

Modeliais paremtas Web aplikacijų programavimas gali padėti struktūrizuoti procesą. Pagrindinis modelis gali tarnauti išskaidant projektavimo procesą skirtingais aspektais ir parodyti sąsają tarp vaidmenimis paremto vaizdavimo.

Modeliais valdoma architektūra (MDA) yra požiūris į IT sistemos specifikaciją, kuri atskiria sistemos funkcionalumo specifikaciją nuo sistemos funkcionalumo specifinėje platformoje (OMG, 2001) [8].

Technologinė pažanga išsprendė kaikurias šių problemų projektuojant programinę įrangą skirtingoms platformoms: XML (eXtensible Markup Language – išplečiama žymių kalba) dokumentai palaikomi XSL (eXtensible Stylesheet Language) stilių leidžia sukurti koreguojamą PĮ pateikimą skirtingų įrenginių naudotojams. Žvelgiant į žmogaus ir kompiuterio sąsajos (HCI) subsistemos pusės, dauguma patobulinimų buvo pasiekti iš esmės dėl grafinių naršyklių evoliucijos ir jų galimybių realizuoti nuo platformos nepriklausančias vaizdavimo kalbas [4].

2.6. Vartotojo sąsajos kūrimo technologinių sprendimų analizė

Pagrindinis darbo tikslas yra sukurti abstraktų aprašymą ir terpę, kuri galėtų automatizuotai siūlyti tinkamą grafinę vartotojo sąsają, pagal naudojamą informaciją bei sistemos funkcijas konkretiems atvejams. Tai taip pat dar vadinama vartotojo sąsajos plastiškumu.

Šio projekto galutinis rezultatas turėtų išspręsti anksčiau minėtas problemas:

- Nuo platformos nepriklausantį programinį kodas;
- Dinamiškai besikeičianti vartotojo sąsaja

Grafinė vartotojo sąsaja turi būti kaip įmanoma intuityvesnė ir patogesnė naudoti. Jos generavimas turi užimti kuo mažiau laiko bei išnaudoti kuo mažiau kompiuterio resursų.

HTML problema yra ta, kad tai buvo sukurta interakcijai tarp žmonių ir mašinų. Kai tinklas buvo išrastas vėlyvais 1980 metais, viskas atitiko to meto poreikius. Tačiau kai tinklas persikėlė į

kitus gyvenimo sritis, iš HTML buvo pareikalauta atlikti daugybę neįprastų dalykų. XML yra sukurtas grįžti mums prie struktūrizuotų duomenų pasaulio.

Kaip bebūtų, automatinis vartotojo sąsajos kodo generavimas nėra pagrindinis sprendimas dėl daugelio besikeičiančių faktorių į kuriuos reikia atsižvelgti projektavimo metu. Pusiau automatinis palaikymas yra labiau tinkamas ir lankstus variantas.

2.6.1 GUI generavimas iš komentuoto programinio kodo

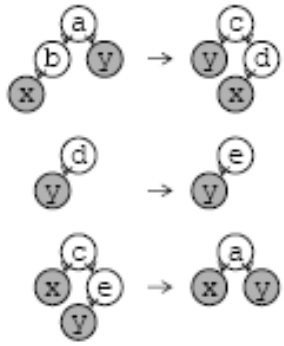
Vienas iš galimų GUI generavimo būdų - programinį kodą papildyti grafinės sąsajos interakcijos elementais. Privalumas toks, jog programuotojams nereikia specifikuoti daug konkrečių grafinės sąsajos aspektų ar bendradarbiauti su dizaineriais. Jiems tereikia kode pažymėti, jog grafinis elementas turi būti naudojamas vartotojo interakcijos procese. Tai yra tas pats, kaip atskirai apibrėžti PĮ logiką ir atitinkamą užduočių modelį. Tik šiuo atveju pastarasis neapibrėžiamas atskirai, o generuojamas iš anotuoto kodo.

Įprastai GUI suformavimui reikia rankiniu būdu apibrėžti kiekvieną elementą arba juos ekrane nupiešti su tam skirtu įrankiu. Abu būdai reikalauja atskiro darbo GUI formavimui. Tačiau kokie grafikos elementai bus naudojami galima sužinoti tiesiog analizuojant programinį kodą. Tai ypač patogu prototipiniame lygmenyje norint įvertinti programinę įrangą – atliekant dažnus pakeitimus nereikia lygiagrečiai keisti ir GUI – tai bus atliekama automatiškai. Tačiau iškyla tokie sunkumai [19]:

- Programavimo kalba negali būti lengvai pasirenkama. Ją turi būti patogu analizuoti, kad nustatyti, kur vartotojo interakcija prasideda. Kalba turėtų būti lengvai praplečiama papildomomis GUI valdymo, kontrolės aprašus.
- Kadangi programuojama su papildomais GUI aprašais, galiausiai PĮ kodą gali būti sudėtinga suprasti tose vietose, kur sudėtingas vartotojo interakcijos procesas.
- Kadangi toks GUI generavimas naudoja užduočių modelį netiesiogiai tai paveldi kažkurias problemas iš modeliais paremto GUI projektavimo pvz. neoptimalus elementų išdėstymas ar jų pasirinkimas.

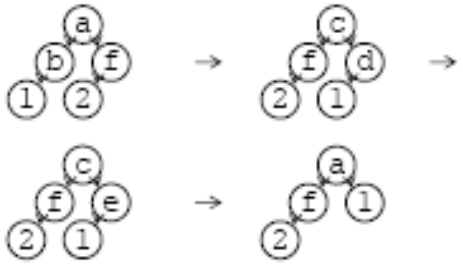
Norint analizuoti anotuotą programinį kodą ir generuoti GUI be papildomos programuotojo pagalbos galima rinktis medžio perrašymu paremtą programavimo kalbą. Medžio perrašymas yra labai paprastas būdas specifikuoti duomenų transformacijas – programavimo kalbos elementai yra struktūrizuoti medžiu t.y. programoje pateikiami taisyklių rinkiniai, kurių kiekvienas gali būti suprantamas kaip : „jeigu kurnors randamas šios formos medis, jį pakeiskite į tokios formos medį“. Turint pavyzdžiui pradinį medį (duomenų struktūrą) `a(b(1),f(2))` ir tam tikras pradines taisykles, tai medį galima transformuoti į `a(f(2),1)`:

Pradinės taisyklės:
`a(b(x),y) → c(y,d(x))`
`d(y) → e(y)`
`c(x,e(y)) → a(x,y)`



a(b(1),f(2)) transformacija:
`a(b(1),f(2)) → c(f(2),d(1)) →`
`c(f(2),e(1)) → a(f(2),1)`

Pradinis medis



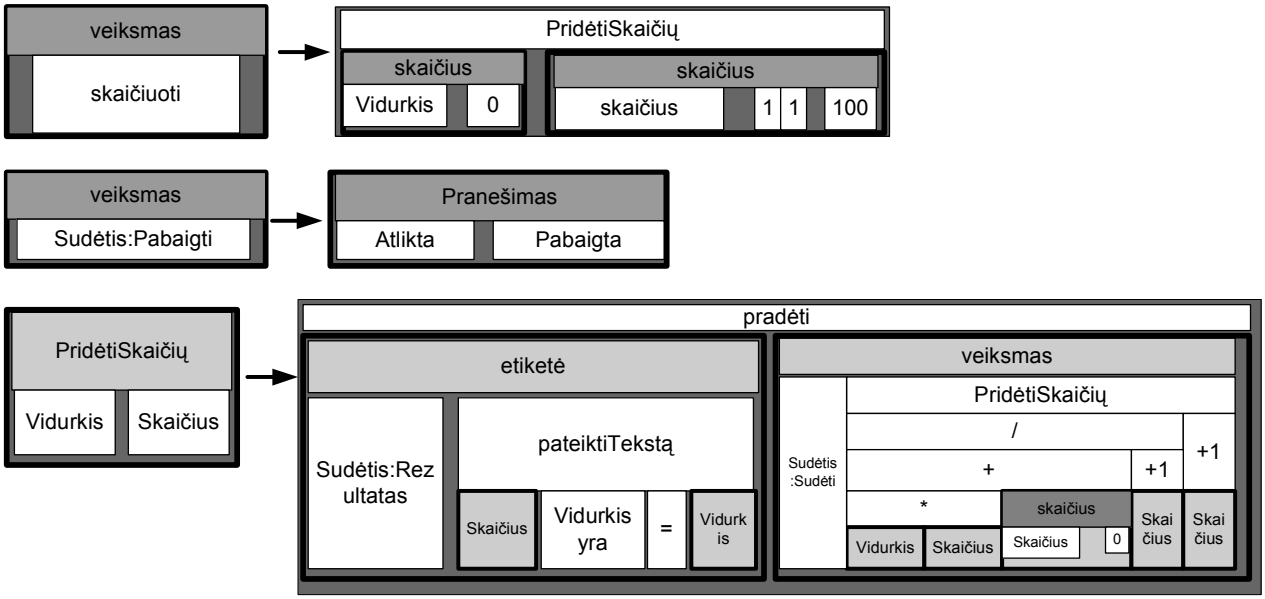
Galutinė forma

Medžių perrašymu paremtos kalbos pasirinkimas turi daug privalumų. Programos vykdymo kelius yra santykinai lengva analizuoti ir tai yra tokiame aukštame abstrakcijos lygyje, jog gali būti anotuojama pastoviu būdu nereikalaujančiu pakeitimų pačioje kalboje. Tokias programas nesudėtinga atvaizduoti naudojant įvairias vizualizacijos technikas leisiančias greičiau ir lengviau suprasti programą.

GUI generavimas suskaidomas į du pagrindinius žingsnius:

- Programinio kodo analizavimas, kad išgauti taškus, kuriuose bus vartotojo interakcija. Išgauta informacija naudojama suformuoti dialogui tarp vartotojo ir programos.
- Generavimas konkrečios grafinės vartotojo sąsajos t.y. visi abstraktūs GUI elementai yra susiejami su nuo platformos priklausančiais elementais.

Tarkime reikia suprogramuoti programą, kuri atliktų skaičiavimą: `(Vidurkis*Skaičius+[įvedamas skaičius])/(Skaičius+1)`. Įvedinėti skaičius galima kiek nori kartų. Grafiškai tai atrodytų:



3 pav. Programos struktūra medžio forma

1 lentelė. Medžio formos struktūrinio programos vaizdavimo elementų paaiškinimai

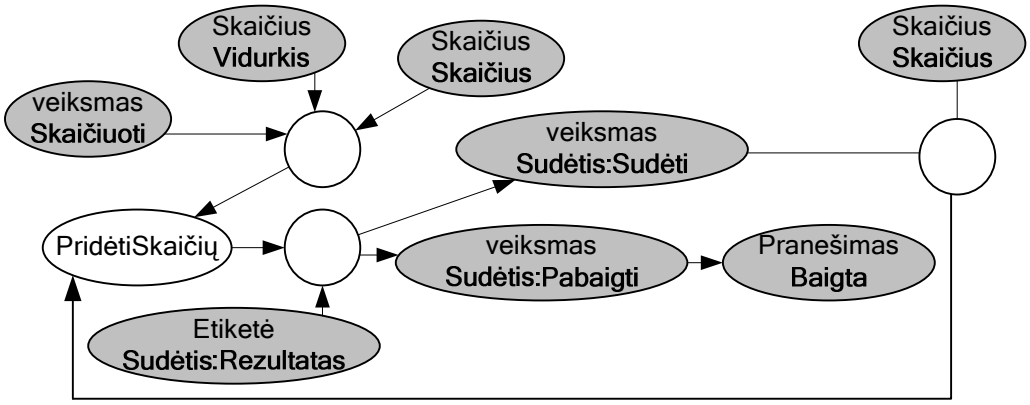
1	Skaičius 1
Tekstas	Tekstas su reikšme „Tekstas“
Kintamasis	Kintamasis, kuris gali įgyti bet kokią reikšmę
tėvas vaikas1 vaikas2 anūkas	Šį medį sudaro du sub medžiai: pirmasis submedis vaikas1 neturi vaikų, antrasis submedis vaikas2 turi vieną vaiką pavadinimu anūkas
GUIAprašas pavadinimas	GUI aprašas, kurio tipas GUIAprašas ir vienas parametro pavadinimas

Laukeliai su tamsiausiu fonu – GUI anotacijos t.y. veiksmas, etiketė ir skaičius.

- Pradiniai **Vidurkis** ir **Skaičius** skaičiai yra gaunami iš vartotojo vykdant pirmą taisyklę aktyvuotą **Skaičiuoti** veiksmo.
- Trečia taisyklė iškviečiama iš pirmosios ir naudojama pakartotiniam naujo skaičiaus pridedamo prie vidurkio įvedimui. Čia yra trys GUI anotacijos**: Sudėtis:Rezultatas etiketės** anotacija naudojama koreguoti **etiketės** tekstą rodomą vartotojui, **Sudėtis:Sudėti veiksmo** anotacija naudojama kaip pasirinkimo taškas tarp šios programos atšakos ir antros taisyklės, **skaičiaus** anotacija naudojama gauti naują skaičių pridedamą prie vidurkio.

- Antroji taisyklė įvykdoma **Sudėtis:Pabaigti veiksmu** yra alternatyva **Sudėtis:Sudėti veiksmo** programos atšakai apibrėžtai trečioje taisyklėje. Ši taisyklė naudojama užbaigti programos darbui. Ryšys tarp šios taisyklės ir **Sudėtis:Sudėti veiksmo** trečioje taisyklėje apibrėžiamas prefiksu **Sudėtis**.

Automatiškai išanalizavus programos kodą gaunamas grafas vaizduojantis kontrolę ir duomenų srautus susietus su GUI anotacijomis. Mazgai – tai programos ir GUI sąveikos taškai, o lapai – jų laikinos priklausomybės



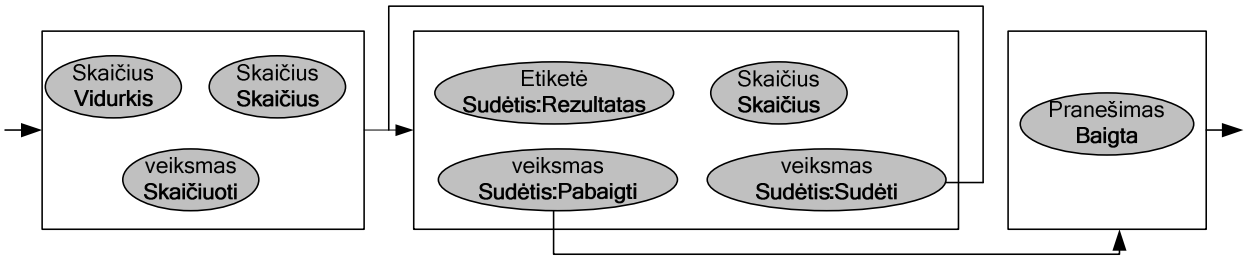
4 pav. Duomenų ir kontrolės srautai programoje

- Nepasiekiamas mazgas **veiksmas skaičiuoti** laikomas programos startiniu tašku GUI interakcijoje.
- Yra vienas atsišakojimas kur priimamas sprendimas tarp **Sudėtis:Sudėti** ir **Sudėtis:Pabaigti** veiksmų ir tai turėtų būti apteikiama vartotojui kaip pasirinkimas.
- Ciklas taikant taisyklę **PridėtiSkaičių**. GUI elementai naudojami cikle turi būti pakartotinai todėl juos reikia talpinti kartu.
- **Pranešimas baigta** yra vienintelis mazgas iš kurio negalima pasiekti kitų mazgų, todėl programa yra nutraukiama jį pasiekus.

Taigi išvados tokios, jog [19]:

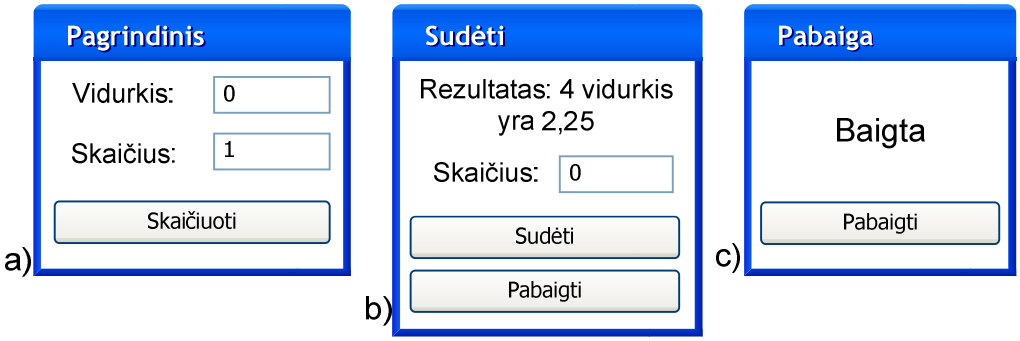
- Visi nepasiekiami mazgai turi būti suprantami kaip pradiniai interakcijos su GUI taškai ir vartotojui turi būti pateikiami kaip veiksmo elementai (mygtukai, meniu ir pan.)
- Visos laikinos priklausomybės ir lygiagrečios GUI interakcijos yra nustatomos tokia tvarka, kaip pateikiami dialogų langai. Elementų vardų prefiksai taip pat nurodo priklausomybes; tai ypač svarbu lygiagretiems veiksmams.
- Visi ciklai ar nepriklausomos ciklų dalys yra apibrėžiamos taip, kokia turėtų būti dialogų hierarchija.

Gautų dialogų seka:



5 pav. Planuojamų suformuoti dialogo langų seka

Kiekvienas dialogo langas atvaizduoja rinkinį tuo pačiu metu vartotojui pasiekiamų sąveikų su GUI. Tai gaunama automatizuotai analizuojant programinio kodo grafą ir naudojama kaip konkretesnis (tačiau vis dar abstraktus) GUI aprašas. Paskutinis žingsnis – sugeneruoti reikiamoje platformoje veikiančią vartotojo sąsają.



6 pav. Automatiškai sugeneruoti dialogo langai

Taigi šis GUI generavimo metodas nereikalauja nei atskiro GUI apibrėžimo specifikuojant konkrečius grafinius elementus nei abstraktaus GUI ar užduočių modelio. Šis būdas taikomas ypač prototipiniame PĮ kūrime, kur svarbus greitis ir iteratyvumas. Tai patogu, kuomet ankstyvose kūrimo stadijose programinis kodas ir GUI dažnai keičiasi. Kad supaprastinti kodo analizę ir pašalinti sunkumus kylančius naudojant kitas programavimo kalbas tikslingiausia rinktis medžio perrašymu paremtą programavimo kalbą.

2.6.2 Automatinis pastovios vartotojo sąsajos generavimas naudojant interpretatoriaus generatorių

Ši idėja kilo kuriant du projektus: MultiJava (MJ) ir Java Modeling Language (JML) [20]. Kadangi šie įrankiai yra susiję ir turi daug panašumų, GUI turi būti apytiksliai panašaus išdėstymo ir elgesio. Programuojant kiekvienam įrankių GUI atskirai būtų sudėtinga jas elgtis vienodai, kas trikdytų vartotoją ir reikalautų didesnių kaštų.

Kad išvengti dvigubo darbo pirmiausiai GUI pilnai suprogramuojama rankiniu būdu. Kitame etape visi laukai, kuriuose buvo specifinė nuo įrankio priklausanti informacija perkeliama į GUI subklases. Jos buvo generuojamos automatiškai t.y. interpretatorius skaitydavo programuotojo paruoštą tekstinę bylą. Pats interpretatorius yra taip pat automatiškai sugeneruotas naudojant gramatiką ir kitą interpretatorių.

Iš programuotojo perspektyvos, GUI generuojamas dviem etapais [20]:

- Surenkant bendrumus tarp visų vartotojo sąsajų į viena superklasę, kurioje saugomi visi GUI komponentai. Ši klasė apibrėžia visiškai pilną išdėstymą ir elgesį kiekvieno GUI.
- Išskiriant skirtumus tarp visų grafinių sąsajų, kurie patalpinami į specialias bylas. Jos naudojamos automatizuotam generavimui subklasių, kurios atskiria GUI duomenis nuo kodo įgyvendinimo.

GUI superklasėje saugoma pagrindinė informacija apie elementų išdėstymą. Kadangi ši superklasė turi ir subklases, superklasė apibrėžia abstrakčius metodus kurie specifikuoja konkrečių metodų vardus ir grąžinamas reikšmes esančius subklasėse.

Pavyzdžiui GUI turi nuorodą į tam tikro projekto tinklapį. Kadangi skirtinguose įrankiuose tinklapio pavadinimas ir vieta skiriasi, GUI turi iškviesti metodą, kuris turi žinoti tą pavadinimą ir vietą.

Subklasės kodas bei interpretatorius generuojamas naudojant ANTLR (ANother Tool for Language Recognition)

. Kad galėtumėme paleisti GUI, subklasė kviečia superklasę, tam, kad galėtų gauti visą informaciją GUI formavimui.

Po šių operacijų sukuriamos visos reikiamos įvesties bylos, kurių reikės grafinėms sąsajoms. Juose saugoma informacija apie visus GUI skirtumus galutiniame programiniame kode.

Toks GUI generavimas išsprendžia problemas apibrėžtas anksčiau [20]:

- Automatiškai sugeneruotas GUI yra **lengviau koreguoti**, nes įprastai reikia keisti tik superklasę. Kiekvienas pakeitimas atliktas superklasei atsispindi visuose įrankiuose, taigi taip sumažinamas bylų, kurias reikia modifikuoti skaičius. Tai taip pat sumažina klaidų skaičių programiniame kode.
- Automatiškai sugeneruotos GUI yra **pastovesnės**, nes kiekviena GUI paveldi vieną pagrindinę superklasę. Ji apibrėžia ir išdėstymą ir elgesį kiekvienos savo subklasės, taigi kiekviena GUI atrodo ir elgiasi vienodai. Tai padeda naujiems vartotojams rasti ir panaudoti visas funkcijas.
- Automatiškai sugeneruoti GUI yra **greičiau** nei rankiniu būdu, nes po superklasės ir generavimo kodo parašymo tereikia aprašyti tik įvesties bylas, kurios apibūdina skirtumus tarp kiekvienos

GUI. Interpretatorius sukuria reikiamas subklases automatiškai, taupydamas laiką kūrimui ir testavimui.

Tokią GUI generavimo metodiką galima taikyti jei:

- Skirtumai GUI languose yra nedideli
- Langų skirtumus galima lengvai aprašyti bylose.

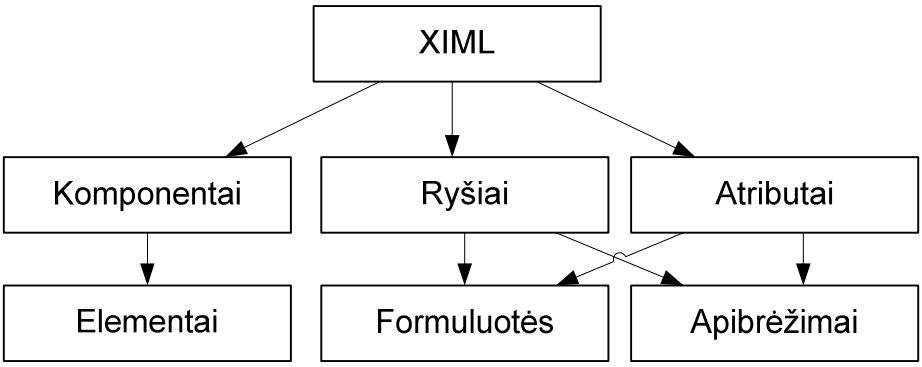
Todėl šis generavimo būdas nebūtų geriausias sprendimas jei GUI turi daug skirtumų programos languose arba tie skirtumai nėra lengvai aprašomi. Tokiu atveju būtų daug subklasių, o bendrumų skaičius superkalsėje toks mažas, kad dauguma klaidų kiltų sudėtingose subklasėse, kad programą darytu sudėtingai tvarkomą.

2.6.3 XIML kalba

XIML (eXtensible Interface Markup Language) tai XML paremta vartotojo universali sąsajos specifikavimo kalba kuriama nuo 1999 metų. XIML gali būti standartinė PĮ technika keistis interakcijos duomenimis ir sąveikauti per integruotus vartotojo sąsajos inžinerijos procesus, kurie apima ne tik sistemos projektavimą, bet ir jos veikimą bei įvertinimą.

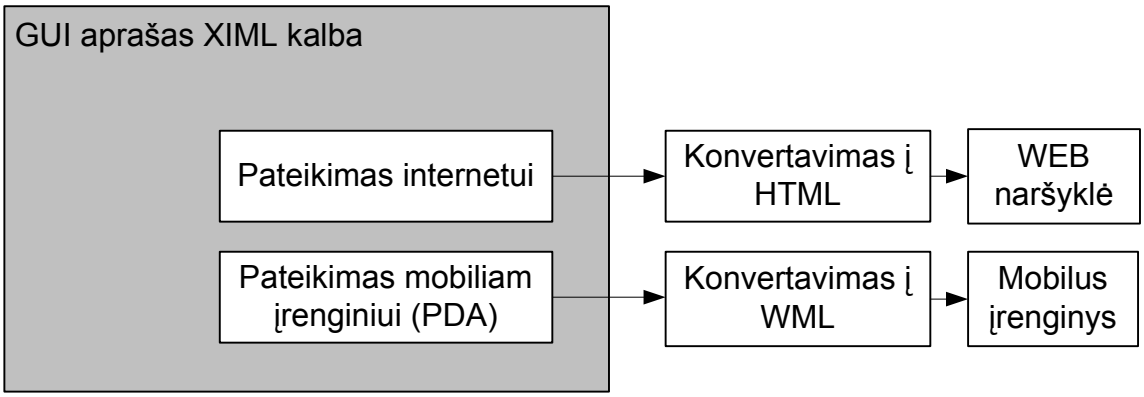
XIML kalbą sudaro šie vaizduojamieji vienetai [18]:

- **Komponentai**. XIML yra sąsajos elementų, kurie kategorizuoti į vieną ar daugiau pagrindinių sąsajos elementų rinkinys. Tai komponentai įprastai naudojami sąsajos modeliuose:
  - vartotojo užduočių - biznio procesai, kurie reikalauja sąveikos su vartotoju,
  - domenų objektų - duomenų objektai, kuriuos vartotojas mato ir gali jais manipuliuoti,
  - vartotojų tipų - vartotojų hierarchija ir charakteristikos,
  - pateikimo elementų - interakcijos elementų hierarchija,
  - dialogo elementų - struktūrizuota kolekcija elementų, kurie nurodo interakcijos veiksmus pasiekiamus vartotojui per GUI pvz. spragtelėjimas, valdymas balsu ar gestas pele.
- **Ryšiai.** Ryšys yra apibrėžimas arba formuluotė, kuri sieja bet kuriuos du ar daugiau XIML elementų arba komponentų. Tai gali padėti žiniomis paremtam projektavimui, vykdymui ir vartotojo sąsajos įvertinimui. XIML palaiko ryšių apibrėžimus, kurie specifikuoja kanoninę ryšio formą taip pat formuluotes (*staitments*), kurios apibūdina naujus atskirus ryšių atvejus.
- **Atributai.** Tai elementų ypatybės kurioms priskiriamos reikšmės. Jos gali būti pagrindinių duomenų tipų arba kitų elementų egzemplioriai.



7 pav. XIML kalbos vaizduojamieji vienetai

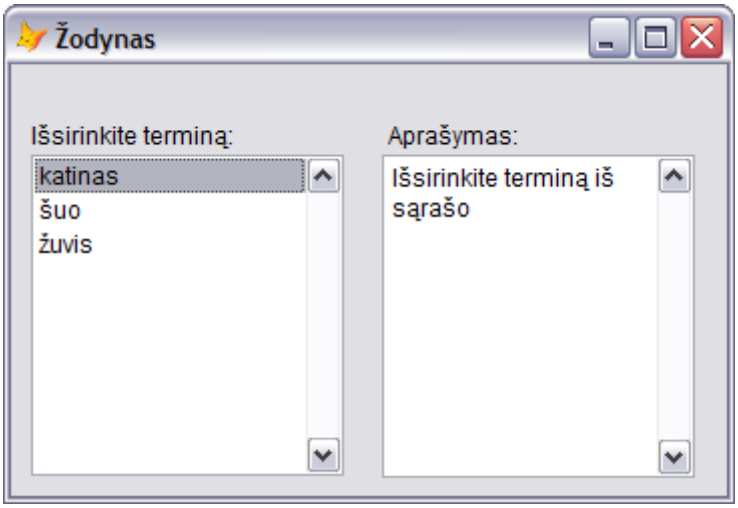
Vienas iš svarbesnių XIML privalumų- ją galima naudoti kuriant vartotojo sąsajas įvairiems įrenginiams. XIML atskiria sąsajos apibrėžimą nuo jos generavimo kiekvienam įrenginiui t.y. GUI generavimu turi atlikti pats įrenginys. Anksčiau daugelis modeliais paremtų sąsajos kūrimo sistemų to neatskirdavo todėl PĮ kūrėjai sumaišydavo sąsajos logiką ir jos apibrėžimą.



8 pav. XIML gali būti naudojama kuriant vartotojo sąsajas skirtingiems įrenginiams

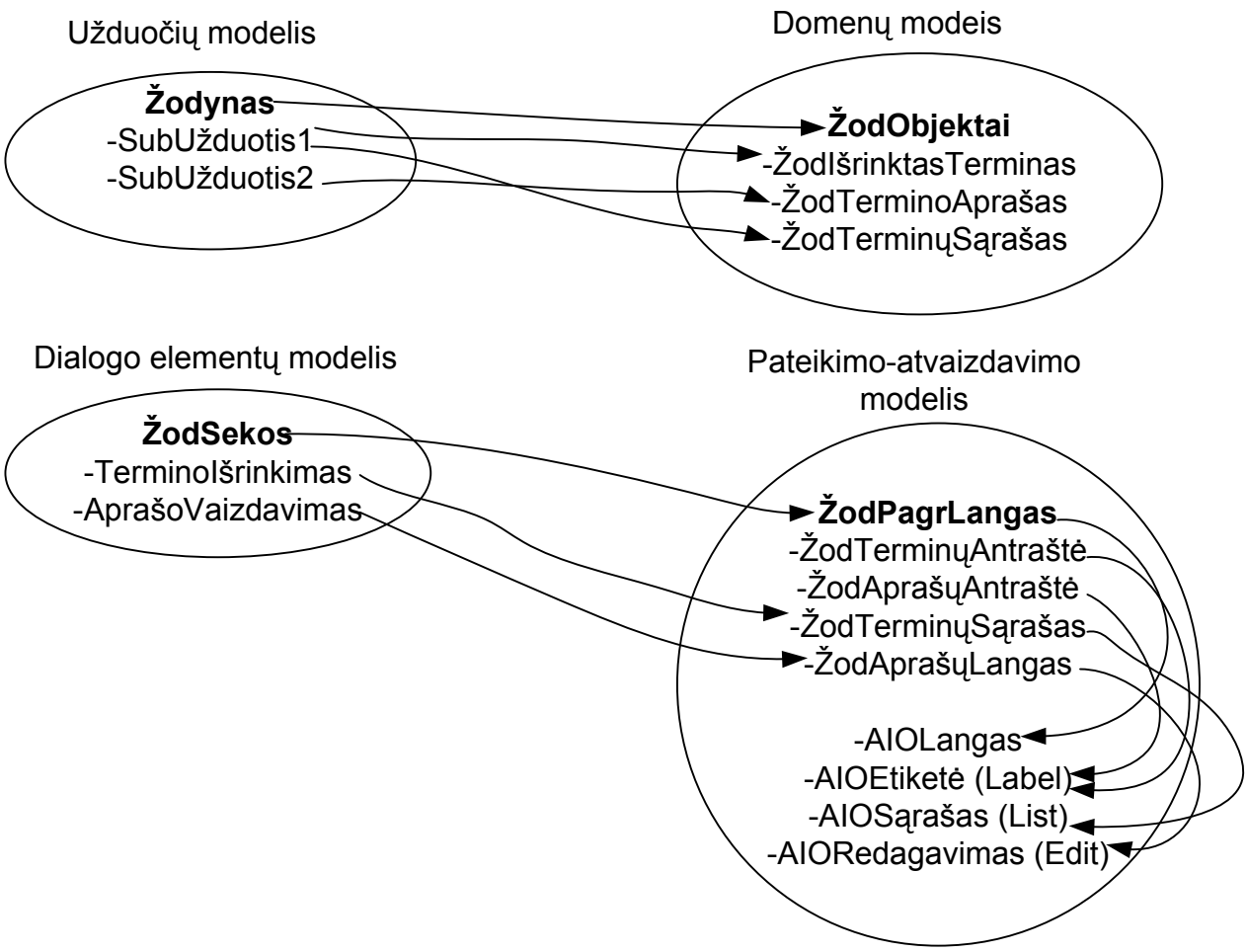
Kitas XIML privalumas – ši kalba palaiko grafinės vartotojo sąsajos išdėstymo perkonfigūravimą. Pvz. kokį elementą naudoti automatiškai parenkama pagal ekrano dydį arba pagal duomenų tipą, kuris bus atvaizduojamas.

Pavyzdys:
Generuojama GUI žodynui. Kairėje esančiame sąraše išsirinkus terminą dešinėje parodomas jo apibūdinimas.



9 pav. Žodyno langas

GUI specifikuojant XIML kalba, bus panaudoti tokie modeliai:



10 pav. Modeliai specifikuojantys žodyno programos GUI

2.6.4 XUL kalba

XUL yra XML kalba, kurioje galima panaudoti tokius kaip XSLT, Xpath, DOM standartus vartotojo sąsajos manipuliavimui .

XUL kalba buvo sukurta atviro kodo projekte Mozilla; projektuotojų tikslas buvo sukurti naują įrankį Web aplikacijų grafinei vartotojo sąsajai, kuris galėtų sąveikauti su plačiai tinkle naudojamomis vaizdavimo kalbomis, tokiomis kaip HTML, CSS ir t.t. XUL paskirtis yra kurti grafines programas vietoj to, kad formuoti dokumentus ir kurti tinklapius; taigi tai yra vaizdavimo formalizmų tokių kaip HTML ar DHTML papildymas [12]. Naudojant variklį Gecko, programuotojai gali įgyvendinti multi platforminę grafinę vartotojo sąsają t.y. tokią sąsają, kurios vaizdavimas nepriklauso nuo įrenginio.

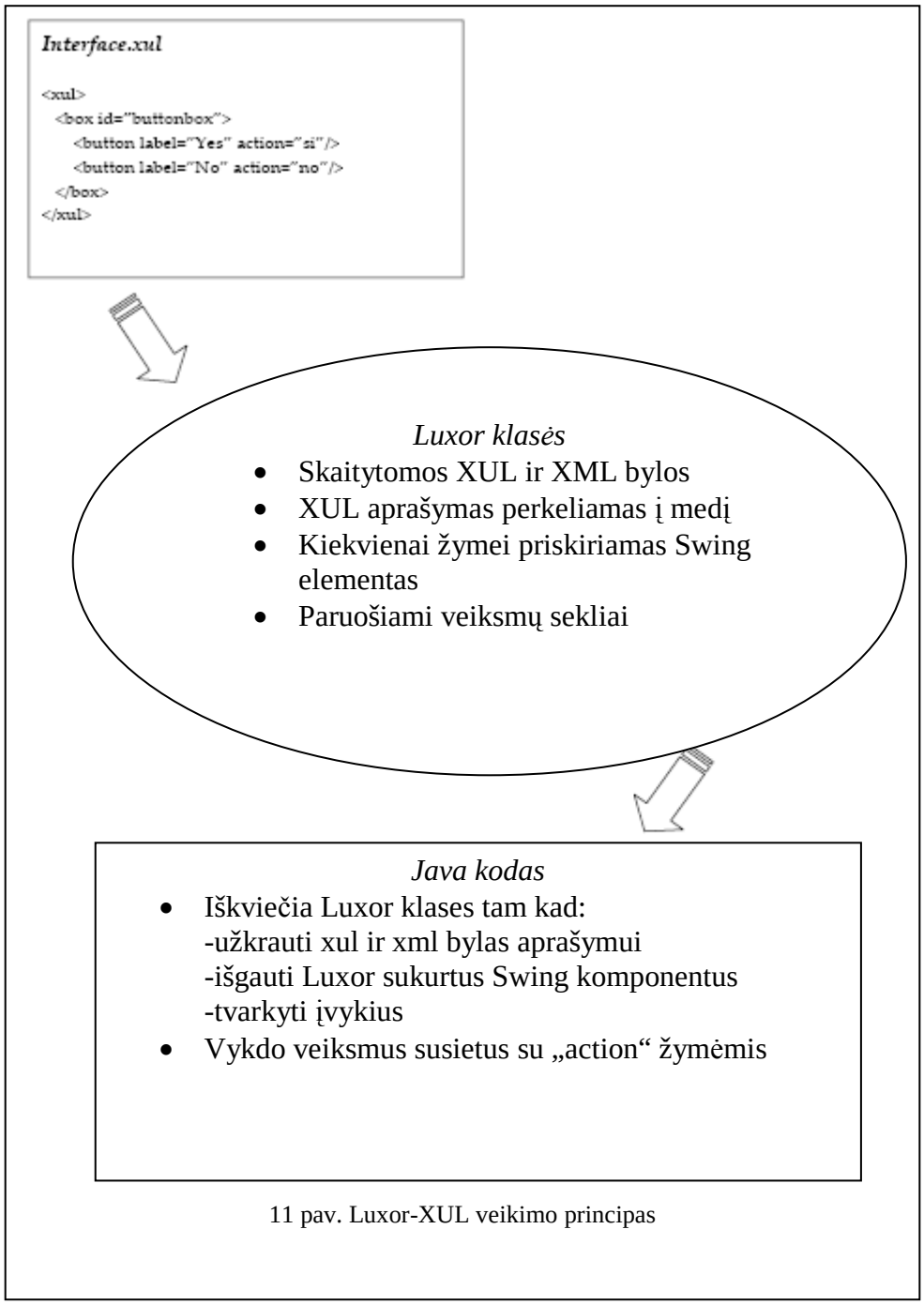
XUL vartotojo sąsajos kūrimo koncepcijoje grafinė vartotojo sąsają yra autonominė esybė programos atžvilgiu ir yra kaip įrankis, kuris gali būti naudingas ir programuotojams ir vartotojams.

Pavyzdžiui naudojant XUL kuriant Netscape 7 ir Mozilla grafines sąsajas leidžia šioms naršyklėms atrodyti vienodai skirtingose operacinėse sistemose ir prie viso to jos gali būti lengvai keičiamos pačių vartotojų [12]. Jei jis nori pakeisti naršyklės išvaizdą, tereikia tiesiog atsisiųsti vieną iš GUI atvaizdavimo šablonų nekeičiant vaizdavimo variklio programinio kodo. Kiekvienas šablonas yra suformuotas iš XUL, XML bylų ir paveikslėlių. Taigi grafinė vartotojo sąsaja gali būti pateikiama aukštame išvaizdos lygyje naudojant hierarchiškai organizuotų elementų rinkiniais panaudojant XML formalizmus.

XUL programos gali būti atidaromos tiesiog tinklapyje arba gali būti atsisiųstos ir instaliuotos (instaliavimas supaprastintas iki minimumo).

2.6.5 Luxor-XUL projektas

Luxor-XUL yra atviro kodo projektas kurio tikslas integruoti Mozillos XUL funkcionalumą su Java portatyvumu, kad būtų galima sukurti nuo platformos nepriklausančią vartotojo sąsają, XUL bylos naudojamos apibūdinti grafiniams elementams, o XML naudojama įgyvendinti tam tikrus objektus, tokius kaip medžiai, sąrašai ir lentelės. XUL dokumente grafinė vartotojo sąsaja apibūdinama hierarchiškai; hierarchiniai ryšiai tarp elementų yra tinkamai pateikiami dokumento medžio struktūros pagalba [12]. Luxor reikalauja, jog identiška struktūra būtų laikinai patalpinama atmintyje. Tai pasiekiama naudojant JDOM paketą (JDOM yra atviro kodo API XML dokumentų kūrimui, analizavimui bei koregavimui). **11 pav.** pavaizduota kaip Luxor sąveikauja su Java programa: kai tik XUL sąsajos aprašymas yra baigtas, turi būti paruošiamos Luxor klasės, kurios galės sugeneruoti grafinę vartotojo sąsają su visomis mygtukų funkcijomis.



11 pav. Luxor-XUL veikimo principas

Taigi naudojant tokią kalbą kaip XUL yra patogu kurti grafines vartotojo sąsajas, nes tai reikalauja santykinai mažai programinio kodo ir taip sutaupome daug laiko. XUL kalba, sukurta Mozillos komandoje yra gana galinga, bet veikia tik Mozillos/Netscape kontekste (naršyklėse paremtose Gecko varikliu). Luxor XUL atviro kodo projektas bando praplėsti XUL naudą kitoms programoms paremtoms Java technologijomis. Deja, Luxor tikslai nėra visiškai pasiekti: dauguma XUL galimybių nėra visiškai suderinama su Java. Tačiau bendru atžvilgiu, XUL kalba yra paprasta ir norint modifikuoti

vartotojo sąsają, nereikia koreguoti Java kodo. Taigi mažesnis programinio kodo kiekis galėtų pagerinti visos IS savybes.

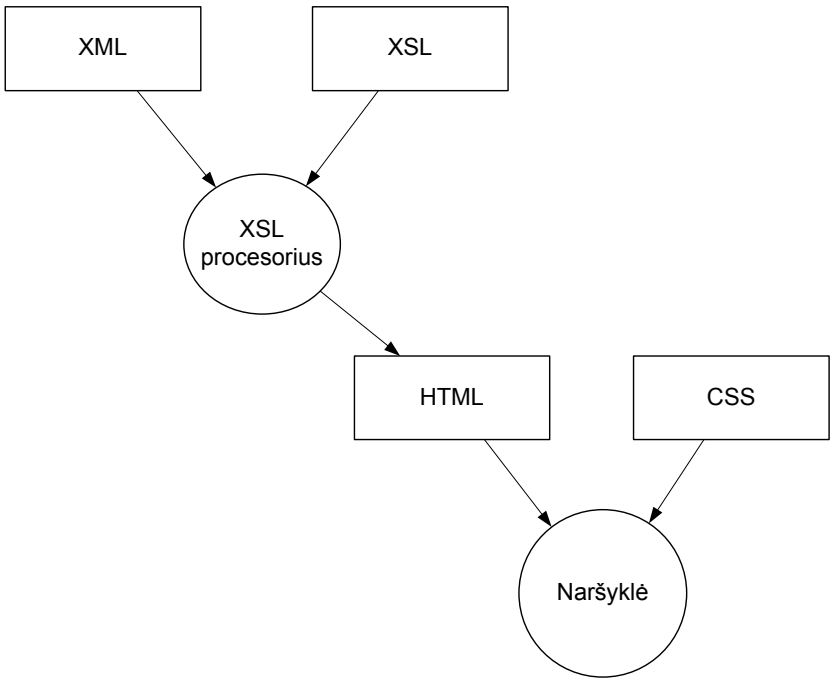
```
//filter_enti.xul

<xul>
<map id="SCELTA_PROP1">
            <entry key="UNO" value="tipoUNO"/>
           <entry key="DUE" value="tipoDUE"/>
</map>
<map id="SCELTA_PROP2">
            <entry key="UNO2" value="specUNO"/>
           <e  ntry key="DUE2" value="specDUE"/>
</map>
<tabbox>
<tabs>
<tab label="Proprieta'"/>
<tab label="Locazione"/>
<tab label="Funzione E.A."/>
<tab label="Servizi Offerti"/>
<tab label="Materiale"/>
<tab label="Offerte Educative"/>
<tab label="Offerte Formative"    />
</tabs>
<tabpanels>
<tabpanel id="proptab">
....//segue descrizione del pannello
</tabpanel>
<tabpanel id="loctab">
...../segue descrizione del pannello
</tabpanel>
<tabpanel id="funtab">
…
…
</tabpanels>
</tabbox>
</xul>
```

12 pav. Programos langas ir jam sugeneruoti reikalingas kodas

2.6.6 XSLT transformacijų šablonai

XSLT yra sukurta kaip dalis XSL, kuri yra XML stilių kūrimo kalba. XSL transformavimo žodynas yra ta pati XML kalba. XSL apibūdina XML dokumento stiliaus keitimą naudodama XSLT, kad apibūdinti kaip dokumentas yra transformuojamas į kitą XML dokumentą. Kaip bebūtų, XSLT nėra universali XML transformavimo kalba [14].

Ši diagrama demonstruoja kaip XML verčiama į HTML:



13 pav. XML transformavimo į HTML procesas [13]

XML transformuojama XSLT pagalba. XSLT procesorius skaito vienu metu XML dokumentą bei XSLT stilių failą, kuriame pateikiami taisyklės nurodančios kaip reikia transformuoti XML bylą. Taip gaunamas HTML kodas.

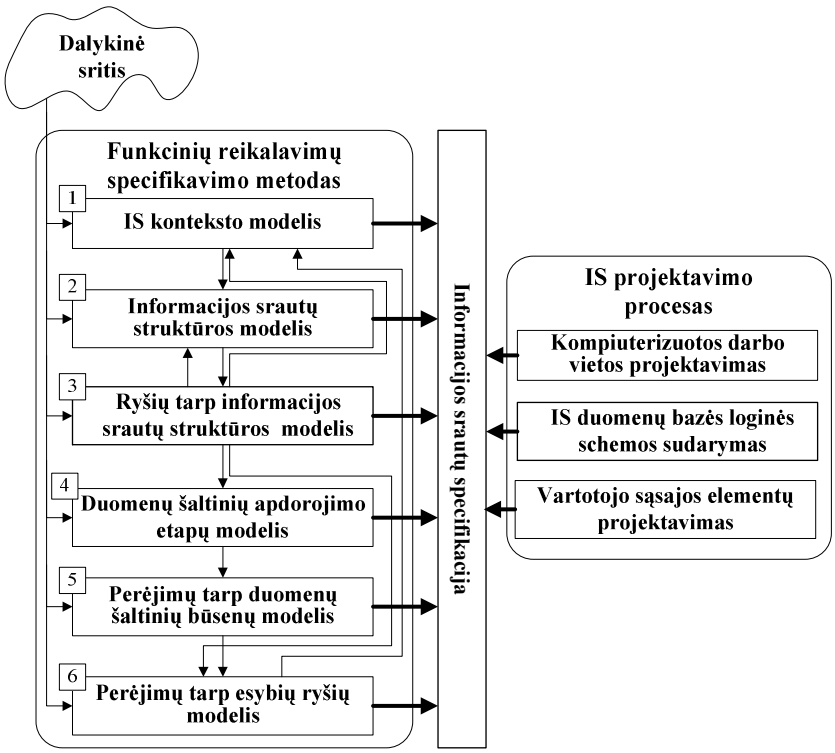
Transformacijos procesas vyksta labai panašiu keliu kaip ir skriptų rašymo kalbos Python ar Perl – pritaikomos taisyklingos išraiškos prie įvedamo duomenų srauto ir rasti elementai transformuojami į išeinančius duomenis. Šiuo atžvilgiu XSLT galėtų būti vadinama skriptų kalba, ypač dėl to jog turi elementus srauto kontrolei. Dauguma XSLT atliekamų funkcijų galima įvykdyti naudojant API tokias kaip SAX (Simple API for XML) ar DOM (Document Object Model). Tačiau kaip bebūtų, XSLT turi pranašumų. Vieną iš jų iliustruoja XML tikslas: duomenų atskyrimas nuo programos logikos ir kodo. Kadangi XSLT neturi nuo platformos priklausančios sintaksės, mūsų duomenys gali būti konvertuojami į bet kokį pasirinktą formatą kombinuotą su programiniu kodu ir pritaikyta bet kokiam įrenginiui. Taigi, „švari“ informacija gali būti nukreipta netgi į dar nesukurtus įrenginius ar sistemas.

2.7. GUI generavimo technologijos parinkimas

Automatizuota informacinių sistemų internetinių sąsajų generavimo sistema yra vienas iš modulių ODRES (Output Driven Requirements Specification Method) metodu paremto CASE įrankio, skirto automatizuotai specifikuoti ir projektuoti IS kūrimo procesų etapams. ODRES metodo, kuriamo

KTU Informacinių sistemų katedroje, paskirtis – kuriamos IS įeinančių ir išeinančių duomenų srautų analizavimas, formalizavimas ir išsaugojimas duomenų saugykloje, taip, kad būtų įvertinama kuo didesnė dalis nefunkcinių reikalavimų, o formalizacija būtų atlikta aukštame lygyje. Taip siekiama sumažinti tikimybę, jog sistemos projektuotojas reikalavimus interpretuos kitaip nei buvo numatęs užsakovas.

ODRES metode informacijos srautų specifikavimas vykdomas šešiais iteraciniais etapais [23]:



14 pav. Konceptuali ODRES metodo schema [23]

Pirmajame etape žemiausio lygio nedalomos funkcijos susiejamos su duomenų šaltiniais ir rezultatais (IS išvedama informacija).

Antrajame - kiekvienam rezultatui sudaromas struktūros ir rezultato esybių-ryšių modelis.

Trečiajame etape specifikuojami informaciniai ryšiai tarp duomenų šaltinių ir rezultatų.

Ketvirtojo etapo rezultatas yra šaltinio apdorojimo etapų ir perėjimų tarp apdorojimo etapo specifikacija

Penktame etape duomenų šaltinio apdorojimas detalizuojamas iki duomenų šaltinio būsenų lygmens.

Paskutinis etapas leidžia patikslinti jau esančius ryšius tarp duomenų šaltinių ir rezultatų.

Nei GUI generavimas iš programinio kodo, nei naudojant interpretuojantį generatorių skaidantį GUI specifikaciją į klases nepasirinktas pirmiausiai dėl to, jog to nebuvo galima pritaikyti pagal ODRES metodu paremto projekto specifiką t.y. GUI čia turi būti generuojama ne iš programinio kodo, o pagal esamų duomenų šaltinių bei jų sąveikų aprašymus. Projekte programinis kodas realizuojamas po vartotojo sąsajos suformavimo. Be to GUI generavimas analizuojant jos specifikacijos

klases tinkamas tik ten, kur varotojo sąsajos langai nedaug skiriasi vienas nuo kito. Mūsų kuriamoje sistemoje tai būtų didelis trūkumas.

Atlikus lyginamąją analizę tarp šių projektavimo metodų, nuspręsta, jog labiausiai atitinkantis poreikius yra XHTML (arba HTML) kodo formavimas naudojant XLST stilių formavimo kalbą. Pagrindinis faktorius nulėmęs šį pasirinkimą – nuo platformos nepriklausanti sintaksė nereikalaujanti specialios programinės įrangos jai apdoroti. Galimybė kurti savo žodynus (XSLT yra sukurta XML kalbos pagrindu). Be to, interpretatoriui, kuris mūsų atveju yra XSLT, nereikalingas joks papildomas variklis t.y. visą vaizdo suformavimo ir atvaizdavimo procesą atlieka naršyklė. Kitas pasirinkto GUI generavimo varianto privalumas – gauto XHTML dokumento „švarumas“ – nereikia rūpintis jo validavimu - priešingu atveju naršyklė suformuotos vartotojo sąsajos nerodytų.

Apie XIML sužinota nesenai, atlikus jau nemažą dalį šio projekto darbų, todėl nėra visiškai aišku, kiek tai gali būti naudinga šiam projektui. XIML suformuota XML pagrindu, o GUI gali būti specifikuojama 5-kiais modeliais. XIML kalba neaprašo konkrečiai, kokie elementai turi būti naudojami vartotojo sąsajoje, tai turi nuspręsti ir sugeneruoti GUI interpretatorius. Jis yra programuojamas atskirai pagal tai, kokiam įrenginiui ir kokia galutine kalba turi būti pateikiama GUI (HTML, WMA, Java ar pan.). Toks grafinės vartotojo sąsajos generavimo procesas panašus į mūsų pasirinktąjį, tačiau XIML kalbą interpretuoti reikėtų daug sudėtingesnio XSLT šablono, dėl to būtų problematiška jį koreguoti rankiniu būdu. Taip prarastume dalį GUI formavimo lankstumo. XIML kūrėjai nepateikė jokio konkretaus interpretatoriaus, todėl neaišku koks yra kitas galimas gilių specifinių žinių nereikalaujant būdas sukurti nuo platformos nepriklausantį interpretatorių, kuris veiktų be jokių papildomų apdorojimo sistemų. Taigi šiuo atžvilgiu, mūsų pasirinktas GUI generavimo kelias turi pranašumų.

2 lentelė. Technologijų palyginimas

Technologija	**PĮ reikalinga GUI atvaizdavimui**	**Nepriklausomumas nuo platformos**	**Papildomai reikalaujamos kalbos**	**GUI išvaizdos pastovumas skirtingose naršyklėse**
XUL	Naršyklė su Gecko valdikliu	+	-	Nėra informacijos
XIML	Interpretatorius ir naršyklė	+	Kalba XIML interpretatoriui sukurti	Visiškai priklauso nuo interpretatoriaus
XSLT	Naršyklė	+	XHTML (HTML)	Dalinai priklauso nuo XSLT taisyklių, tačiau XSL ir XHTML

Technologija	PĮ reikalinga GUI atvaizdavimui	Nepriklausomumas nuo platformos	Papildomai reikalaujamos kalbos	GUI išvaizdos pastovumas skirtingose naršyklėse
				reikalauja griežto kalbos sintaksės standartų laikymosi ir taip leidžia lengviau užtikrinti GUI vaizdavimo pastovumą

2.8. Analizės išvados

1. Aptarti pagrindiniai aspektai į kuriuos reikia atkreipti dėmesį kuriant grafinę vartotojo sąsają.

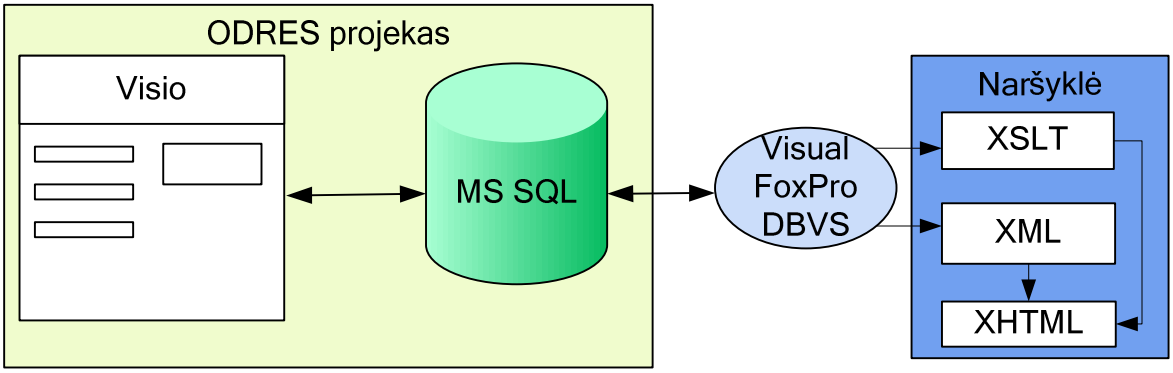
2. Kuriamos sistemos projektavimui bus panaudojamas UML specifikavimo metodas.

3. Palyginti keli galimi kodo automatizavimo procesai. Aptartos XML, XSLT, XUL, XIML kalbų suteikiamos galimybės XML formatu gaunamos informacijos transformavimui į internetinių tinklapių naršyklei suprantamą kodą. Dėmesys buvo kreipiamas į kalbos multi-platforminį pritaikomumą, taip pat į reikiamą papildomą programinę įrangą realizuoti automatinį generavimą ar sugeneruoto kodo atvaizdavimą. Šioje analizės dalyje nuspręsta, jog tinkamiausiai tam tiks XSLT stilių pagalba iš XML kodo formuojamas XHTML ar HTML kodas. Tačiau neatmestas bent dalinis ir XIML kalbos panaudojimas. XUL atsisakyta dėl reikalaujamo papildomo Gecko variklio kalbai atpažinti.

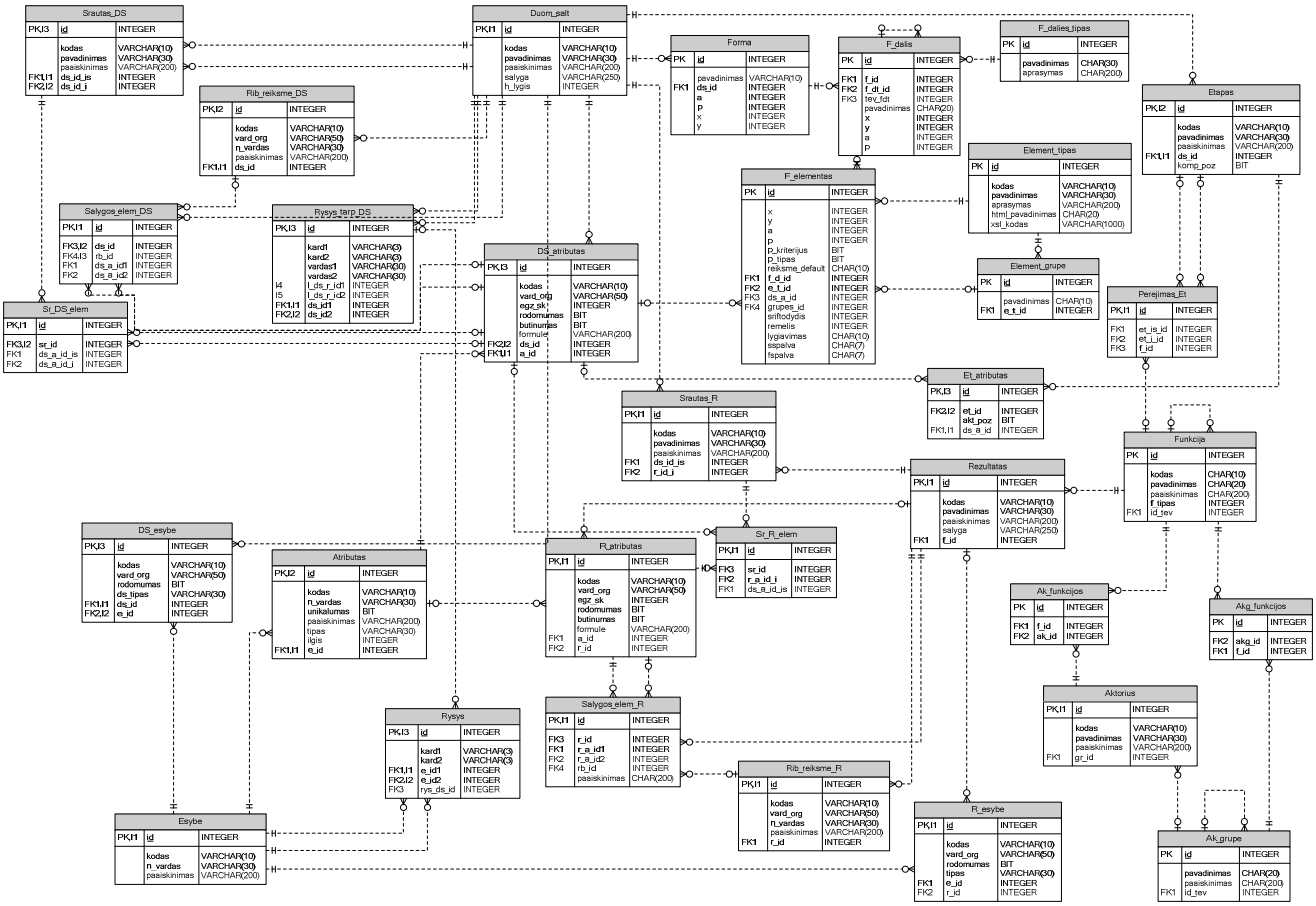
4. Aptartas vieno iš projekto (Luxor-XUL) transformuojančio Mozillos aplikacijų grafinei sąsajai generuoti sukurtos kalbos XUL transformavimą į Java programas veikimo principas bei suteikiamos galimybės transliavimo. Kurti panašų projektą (generuoti iš XUL Java kodą) atsisakyta, nes dauguma XUL galimybių nėra visiškai suderinama su Java.

3. Automatizuoto vartotojo sąsajos kūrimo procesas

Programinės įrangos reikalavimų specifikacija saugoma duomenų bazėje (**16 pav.** Detalesnė schema prieduose). Šiam projektui bus reikalinga informacija nusakanti, kaip turės atrodyti programinės įrangos grafinė vartotojo sąsaja t.y. naudojami grafikos elementai pavadinimai, jų aprašai HTML kalba, interpretavimo taisykles XSL ir XHTML kalba, aukščiai, pločiai, koordinatės ir pan. Šie duomenys Visual FoxPro pagalba konvertuojami į XML formatą tuo pačiu generuojant šio kodo interpretatorių XSL kalba. Kaip atrodys sugeneruota GUI galima išvysti po to kai internetinė naršyklėje XML formatu išsaugota informacija interpretuojama XSLT pagalba ir suformuojamas XHTML kodas kuris iš karto pateikiamas ekrane kaip grafinė vartotojo sąsaja.



15 pav. Grafinės vartotojo sąsajos kūrimo procesas



16 pav. ODRES metodo metabazės schema [17]

Atliekant automatizuotą grafinės vartotojo sąsajos generavimą bus panaudojami šie ODRES metodo reikalavimų specifikacijos metamodelio objektai [17]:

- *Duom_salt* - lentelėje saugoma informacija apie organizacijos objektus (duomenų šaltinius) saugančius duomenis, reikalingus funkcijoms įvykdyti.
- *DS_atributas* – saugoma informacija apie organizacijos objektų (duomenų šaltinius) saugančių duomenis, reikalingus funkcijoms įvykdyti, atributus. Iš čia bus renkama informacija apie grafikos elementų išdėstymą vartotojo sąsajoje
- *F_Elementas* - saugoma informacija apie naudojamus grafinius elementus, skirtus vartotojo sąsajai kurti
- *Forma* – saugoma informacija apie visas kiekvienam DŠ naudojamas formas
- *F_dalis* - Lentelėje saugoma informacija apie kiekvienos formos dalis (epizodus)
- *F_dalies_tipas* - Formos dalies pavadinimas
- *Element_tipas* - Visi galimi grafikos elementų tipai, kuriuos galima panaudoti generuojant vartojo sąsają
- *Element_grupe* - Apjungtos elementų grupės

- *Atributas* - lentelėje saugomi visi nagrinėjamo organizacijos veiklos konteksto atributai.
- *Rysys* - lentelėje saugomi visi nagrinėjamo organizacijos veiklos konteksto ryšiai tarp esybių.
- *Forma* - informacija apie visas kiekvienam DŠ naudojamas formas
- *Etapas* - lentelėje saugoma informacija apie duomenų šaltinio apdorojimo etapus
- *Et_atributas* - lentelėje saugoma informacija apie duomenų šaltinių apdorojimo etapo metu įvedamus arba modifikuojamus atributus.

3.1. XML žymių aprašai

Norint sugeneruoti XML kodą, kiekvienas lentelės atributas (išskyrus išorinius raktus) privalo turėti griežtai apibrėžtą savo atitikmenį – žymę XML kode. Sukurtame PĮ prototipe kol kas atpažįstami ir turi XML žymes 13 atributų. Viena iš žymių (el_id) yra sudėtinė, formuojama iš dviejų atributų – elemento pavadinimo ir jo identifikacinio numerio duomenų bazėje. To imtasi po to, kai nuspręsta, jog CSS klasės bei kiekvienas suformuotas grafinis elementas turės unikalų pavadinimą, o vien tik identifikacinis numeris būtų ne pakankamai informatyvus.

Kiekvienas aukščio, pločio ar dydžio matavimo vienetas pasirenkamas pagal poreikius (pikseliais, procentais ar pan.). XSL kodas jį interpretuos taip, kaip nurodysime.

3 lentelė. Sistemoje naudojamų XML žymių aprašai

XML žymė	Šaltinis	Aprašymas
elementas	`F_elementas`	Žyme prasideda kiekvieno sąsajos elemento aprašymas
Vardas	`f_dalis.pavadinimas`	Formos dalies (epizodo) pavadinimas
Plotis	`f_elementas.p`	Elemento plotis
x_koord	`f_elementas.x`	Viršutinio dešinio elemento kampo koordinatė x ašyje.
y_koord	`f_elementas.y`	Viršutinio dešinio elemento kampo koordinatė y ašyje
Aukstis	`f_elementas.a`	Elemento aukštis
reiksme	`f_elementas.reiksme_default`	Elemente atvaizduojamas tekstas. Tai gali būti mygtuko pavadinimas, stulpelių antraštės ir pan.
sriftodydis	`f_elementas.sriftodydis`	Elemente naudojamo teksto dydis.
remelis	`f_elementas.remelis`	Elemento rėmelio (jei toks yra) plotis

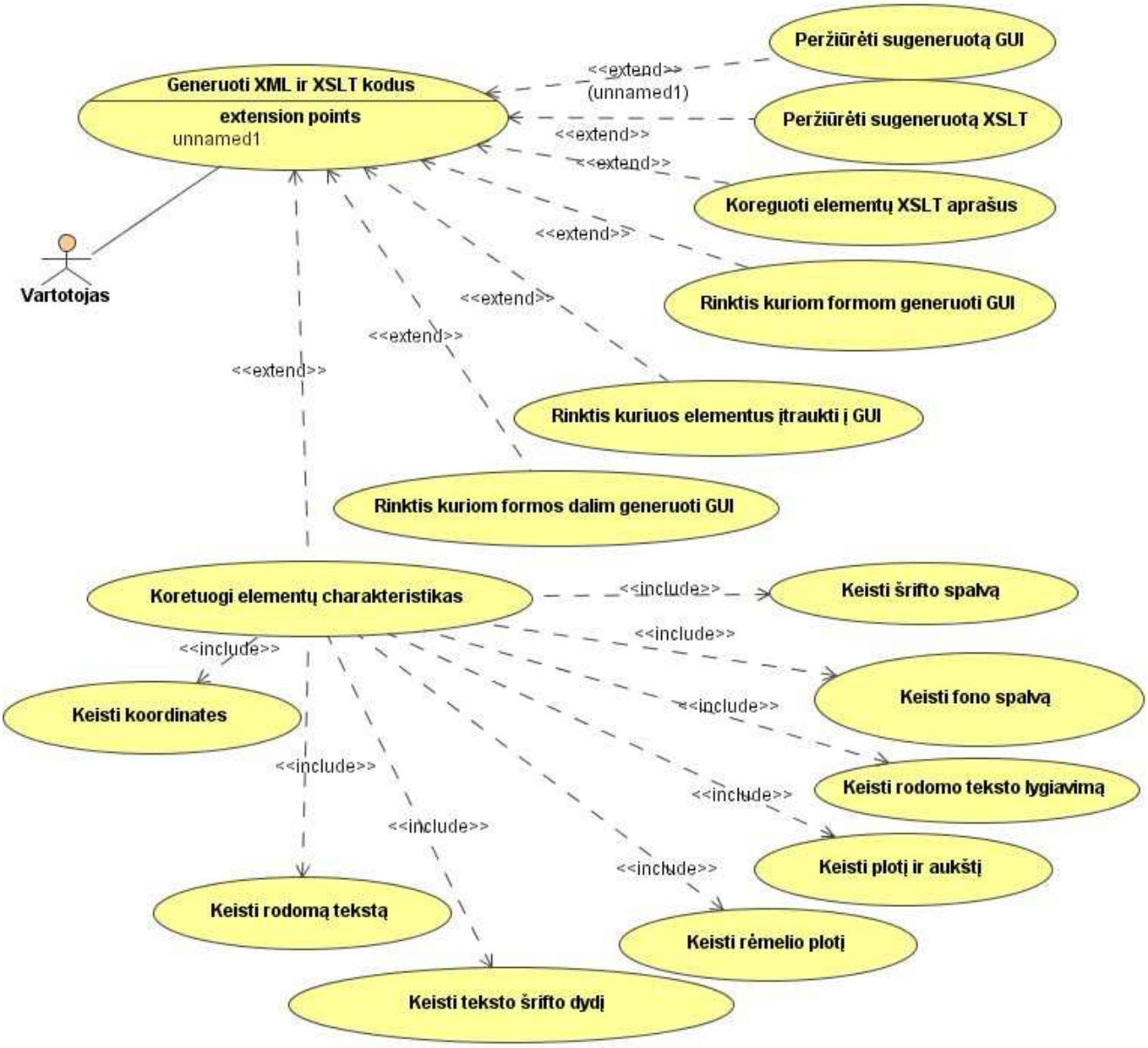
XML žymė	Šaltinis	Aprašymas
lygiavimas	`f_elementas.lygiavimas`	lygiavimas (aprašomas angliškai – left, right, center)
el_id	`element_tipas.pavadinimas+"_"+ f_elementas.id`	Unikalus elemento (CSS klasės) identifikacinis aprašas. Sudaromas iš elemento pavadinimo ir jo identifikacinio numerio duombazės lentelėje (pvz. Mygtukas_12).
Tipas	`element_tipas.pavadinimas`	Elemento tipas. Tai gali būti bet koks elementas, kurį galima atvaizduoti HTML kodu pvz. Mygtukas, lentelė, teksto įvedimo laukelis ir t.t.
Forma	`Forma`	Žyme prasideda kiekvienos formos aprašymas.
srif_spalva	`f_elementas.sspalva`	Šrifto spalva
fon_spalva	`f_elementas.fspalva`	Fono spalva
Ryšys	`f_elementas.link_f_id`	Lango (formos), kuris atidaromas paspaudus elementą ID

3.2. Grafinės vartotojo sąsajos generavimo programinės įrangos funkcijos

Kaip paminėta anksčiau, tikslas – sukurti dinamišką ir lankstų grafinės sąsajos generavimo procesą t.y. sistemos vartotojui suteikti kuo daugiau galimybių koreguoti sąsajos išvaizdą iš karto stebint pasikeitimus gautame XHTML dokumente ir tuo pačiu kaip įmanoma labiau sumažinti poreikį koreguoti generatoriaus programinį kodą.

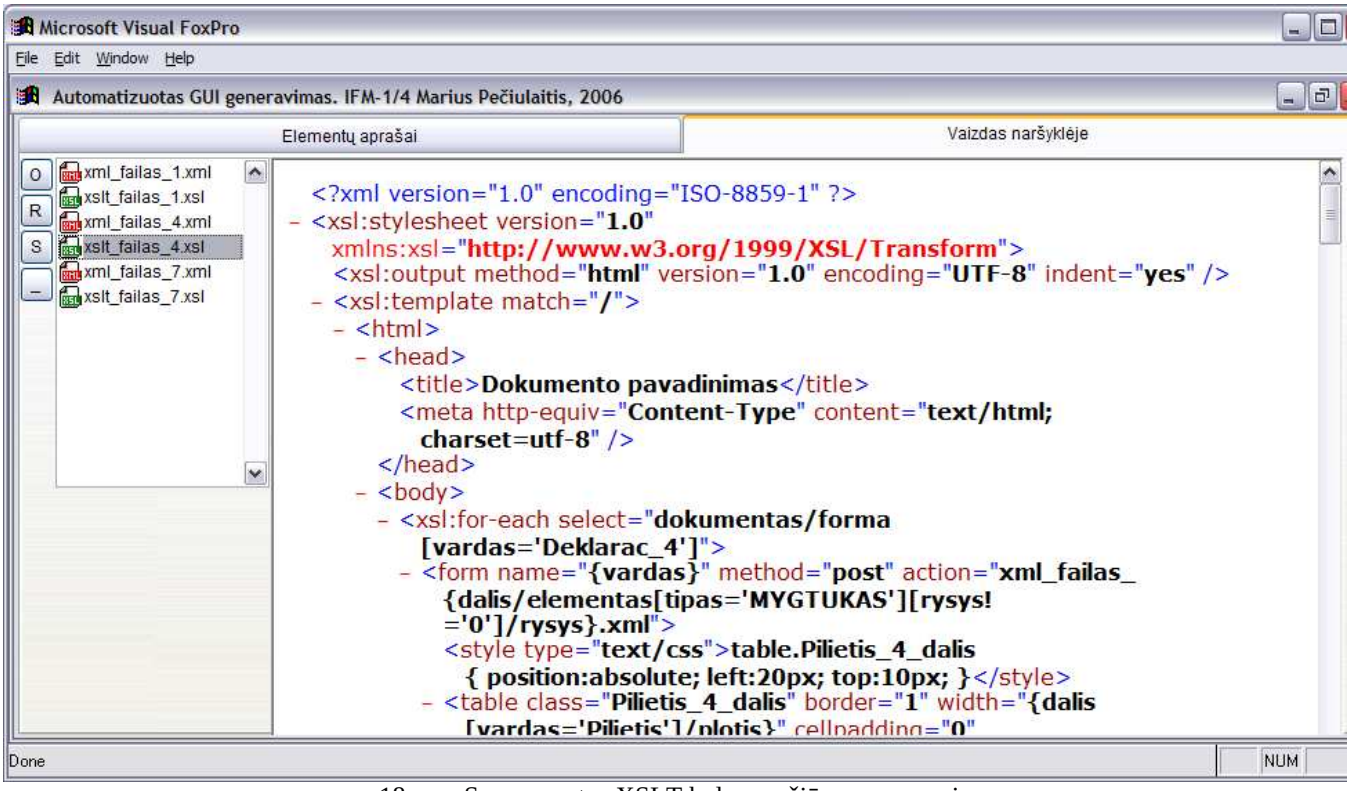
Grafikos elementų rodomumas ir savybės koreguojamos keturiuose lygmenyse:

- Formų generavimas/negeneravimas
- Formų daliu generavimas/negeneravimas
- Konkrečiu grafikos elementų įtraukimas/pašalinimas iš GUI
- Elementų charakteristikos keitimas: koordinatės ekrane, rodoma tekstinė informacija, plotis, aukštis, rėmelio plotis, teksto šrifto dydis, spalva bei lygiavimas, fono spalva.
- Koreguojamos XSLT transformacijos interpretuojančios elementų aprašus XML kode.



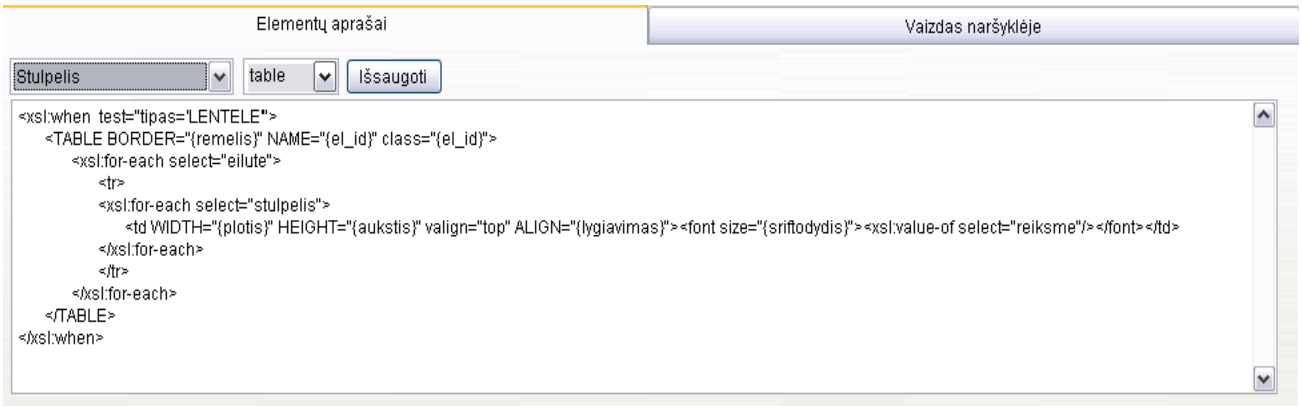
17 pav. GUI generavimo sistemos panaudojimo atvejų diagrama

Prireikus, tame pačiame lange galima atsidaryti peržiūrai XSLT bylas. Patogesniam skaitymui žymės ekrane automatiškai išlygiuojamos, paspaudus kairėje išdėstytus + ar – ženklus elementus galima išskleisti arba sutraukti parodant arba paslepiant jų vaikinius elementus.



18 pav. Sugeneruotos XSLT bylos peržiūra programoje

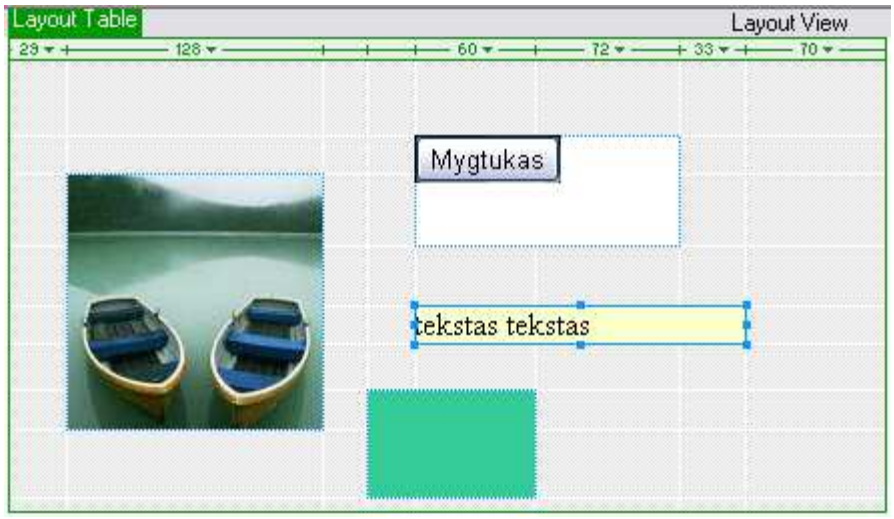
Čia pat, kortelėje „elementų aprašai“, galima koreguojamas kiekvieno galimo grafikos elemento aprašas XSL kalba. Iš šių aprašų vėliau automatiškai formuojamas XSLT šablonas.



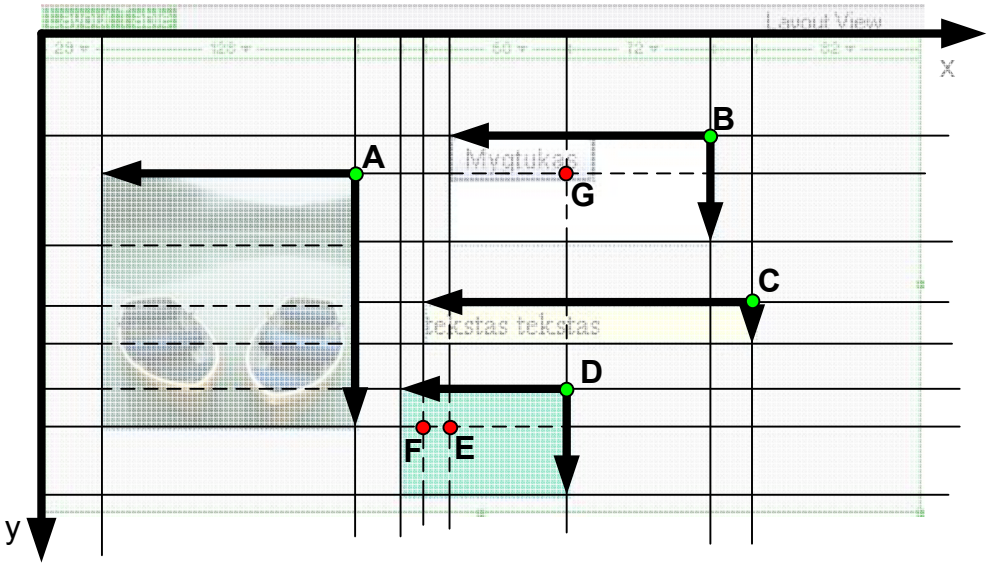
19 pav. Grafikos elementų aprašų koregavimo langas

3.3. Grafikos elementų pozicionavimas erdvėje

Grafinę vartotojo sąsają generuojama XML kodą XSL pagalba transformuojant į HTML kodą. Sąsajos elementai gali būti išdėstomi kiekvieną jų talpinant į lentelės celes (matomas arba nematomas – pasirinktinai) arba panaudojus CSS.



20 pav. Erdvės išskaidymas į celes



21 pav. Erdvės išskaidymas į celes ir jų sujungimas

Kiekvienas elementas kuriamoje sistemoje aprašomas 4 aspektais:

- viršutinio dešinio kampo (pvz. A, B, C ir D) koordinate x ašyje;
- viršutinio dešinio kampo (pvz. A, B, C ir D) koordinate y ašyje;
- aukščiu

- pločiu

Jei elementus ketintume erdvėje išdėstyti juos įkeliant į lentelės celes, reikėtų išspręsti tokius uždavinius:

- Erdvę išskaidyti į celes, kiekvieną grafinio objekto kraštinę pratęsiant į abi puses horizontaliai ir vertikaliai, kol pasieksime kiekvieno vaizduojamosios erdvės kraštus
- Sugrupuoti celes taip, kad jų bendroje užimamoje erdvėje tilptų reikiamas elementas. Šioje dalyje reikia apskaičiuoti, kuriuos taškus iš visų celių kampų koordinačių aibės reikės pašalint (pvz. G, F ir E) taip apjungiant kelias celes į vieną.
- Atrinkti, kurios celės bus užpildomos, o kurios skirtos tik bendros lenteles karkasui išlaikyti.

HTML kodu lentelės aprašomos pirmiausiai eilučių lygyje, o tik po to stulpelių, pvz.:

```
<tr>
  <td width="16" height="54"> </td>
  <td width="99"> </td>
  <td width="157"> </td>
  <td width="42"> </td>
  <td width="27"> </td>
</tr>
<tr>
  <td height="21">E</td>
  <td rowspan="3" valign="top“>A</td>
  <td rowspan="2" valign="top">B</td>
  <td valign="top">C</td>
  <td>D</td>
</tr>
```

Iš visų eilutėje esančių stulpelių (celių) tik vienas aprašomas aukščiu pikseliais-žemiausia celė toje eilutėje. Visos celės eilutėje išskaidomos į skirtingų aukščių lygius. Priklausomai nuo to į kurį lygį patenka celė, taip ir aprašomas jos aukštis. **22 pav.** celės E, C ir D patenka į žemiausią lygį, tad vienai iš jų (pirmai iš kairės, E) nurodomas aukštis pikseliais.

```
<td height="21">E</td>
```

Kitos to paties lygio celės aprašomos nenurodant jokio aukščio matavimo vieneto:

```
<td valign="top">C</td>
```

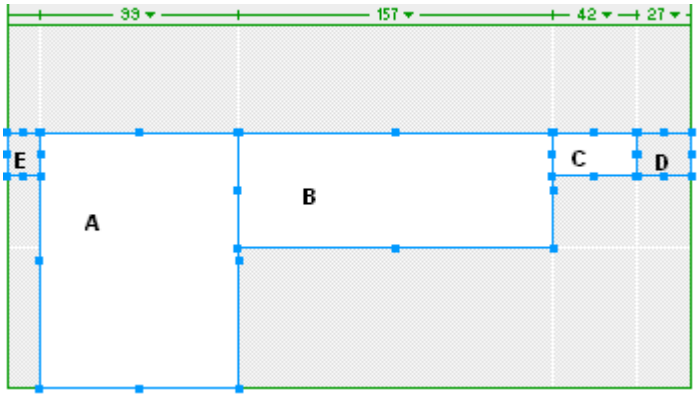
```
<td>D</td>
```

B celė patenka į antrą aukščių lygį, tad jos aprašas yra toks:

```
<td rowspan="2" valign="top">B</td>
```

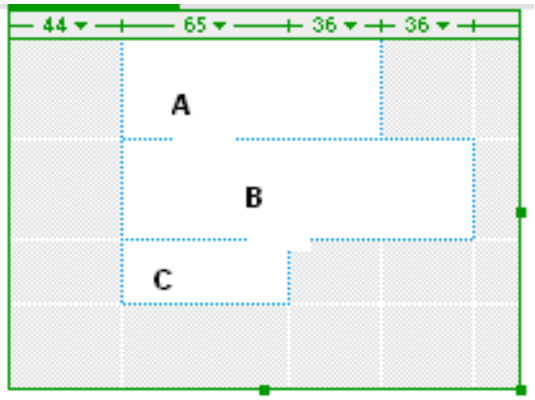
Celė A yra aukščiausia tarp visų toje eilutėje ir patenka į trečią lygmenį:

```
<td rowspan="3" valign="top“>A</td>
```



22 pav. Celių eilutė

Tokia pačia logika aprašomi ir celių pločiai: viename stulpelyje esančiosios celės sugrupuojamos į lygmenis pagal savo plotį.



23 pav. Celių stulpelis

Toks elementų išdėstymo būdas reikalauja ne mažų skaičiavimų ir apkrauna papildomu programavimu bei sudėtingesniu, ilgesniu galutiniu HTML kodu. Kad išvengti to, nuspręsta, jog patogiausiai grafikos elementus pozicionuoti erdvėje CSS pagalba. Tam pakanka nurodyti taškų A, B, C ir D (pav. 10) koordinates x ir y ašyje (reliatyviai, absoliučiai). Pvz.

```
h2.elemento_vardas
{
position:absolute;
left:100px;
top:150px
}
```

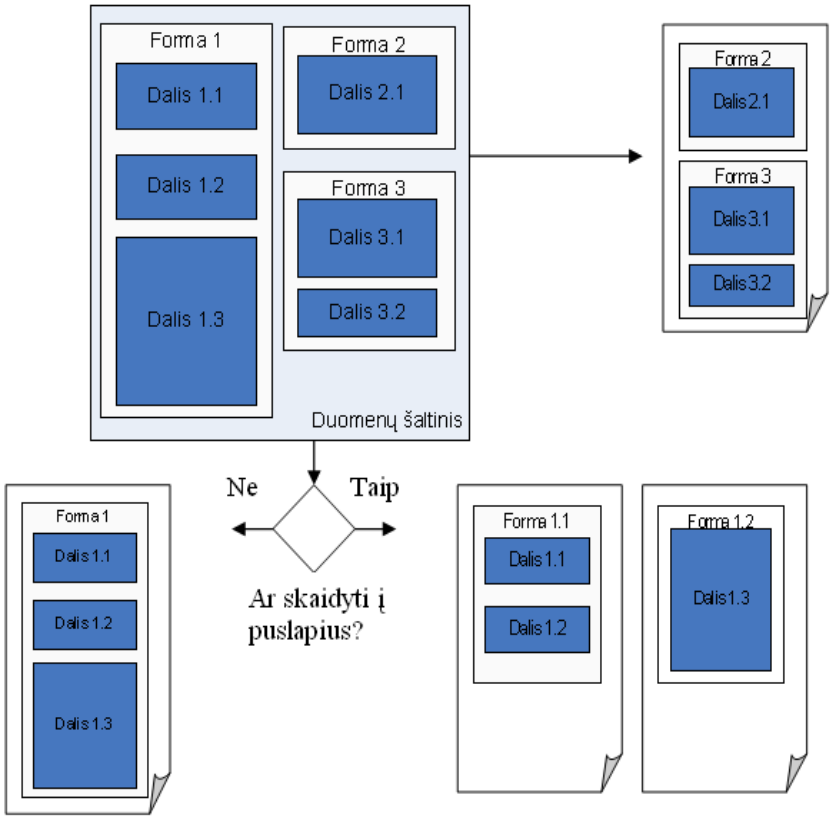
Prie viso to CSS suteikia didesnes ir lankstesnes galimybes keisti grafikos elementų vaizdavimą. Sukurtas klases galima pritaikyti iš karto keletui iš jų, todėl nebereikės kiekvieno elemento HTML kodo koreguoti atkirai, norint pakeisti tinklapio išvaizdą. Ateityje bus galima pasinaudoti ir kitais CSS privalumais.

3.4. XML ir XSLT generavimo žingsniai

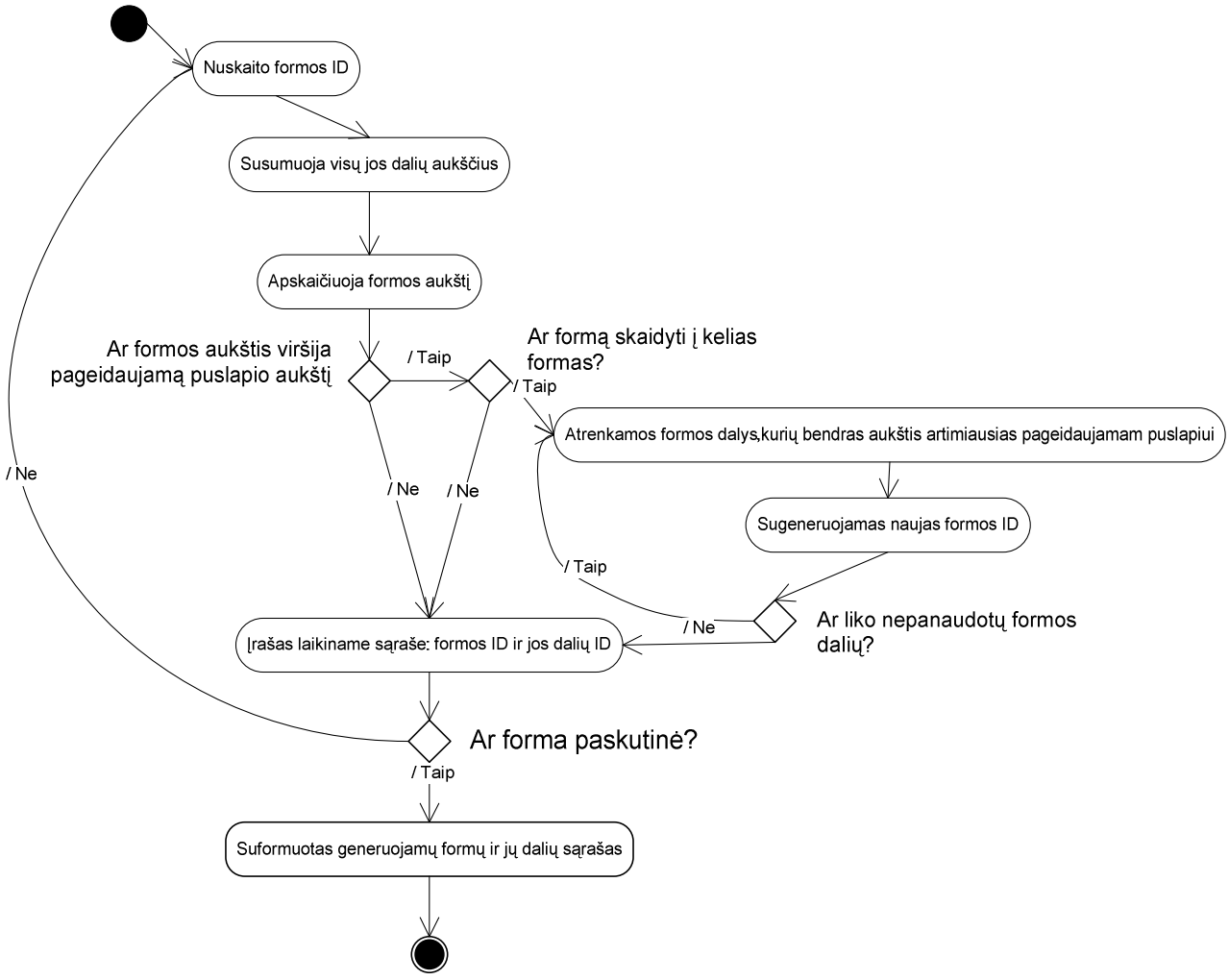
Duomenų šaltiniai, kuriems generuojama vartotojo sąsaja gali susidėti iš daugelio dalių, tad siekiant, kad tinklapiu būtų patogu naudotis tikslinga formas išdėlioti kelėtame puslapių. Kiek jų bus apskaičiuojama pagal kelis parametrus:

- Pageidaujamas puslapio aukštis
- Formos aukštis

Pirmąjį parametrą nurodo sistemos vartotojas, o antrasis apskaičiuojamas pagal formos dalių skaičių, aukštį ir tarpus tarp jų vertikalioje ašyje. Jei forma viršija pageidaujamą puslapio aukštį, vartotojas apie tai perspėjamas ir gali nuspręsti formą skaidyti ar ne į kelias ir talpinti į atskirus puslapius. Formų puslapiavimas projekte dar nerealizuotas, tačiau algoritmas turėtų būti vykdomas pagal **24 pav.** pateiktą schemą.



24 pav. Duomenų šaltinio skaidymas į tinklapio puslapius



25 pav. Formų paskirstymo į puslapius algoritmo veiksmų seka

Lentelėje „f_elementas“, kiekvienas elementas gali turėti „link_f_id“ atributo reikšmę – tai formos ID, kuri turi būti pasiekiama įvykus vartotojo interakcijai su šiuo grafikos elementu. Taigi vartotojo sąsajos langai gali būti susiejami tarpusavyje per bet kurį panaudotą elementą (kiek tai leidžia HTML sintaksė). Sugeneravus GUI šie ryšiai turi būti veiksnūs ir įprastai suformuojami mygtukų arba nuorodų pavidalu.

XML žymių hierarchija turi atitikti informacijos saugomos duomenų bazėje hierarchiją t.y. jei tėvinė lentelė turi vaikinių įrašų, tad ir XML kode tėvinis elementas (suformuotas iš tėvinės lentelės) turės vaikinių elementų.

1. (1 žingsnis sekų diagramoje) Pasirinktos GUI generavimui formos išskirstomos į puslapius. Jei forma netelpa į puslapį, tuomet ji išskaidoma į mažesnes. Pradinės formos grafikos elementai perskirstomi naujoms formoms, taip pat pakeičiamos formos grafinių elementų koordinatės Y ašyje.

2. (2-5 žingsniai sekų diagramoje) Nuskaitoma informacija apie puslapio formą ir pradedamos formuoti XSLT ir XML bylos – sukuriamos jų antraštės.

- **XML:** sukuriama elemento <forma> pradžios žymė. Čia talpinsime informaciją apie visus elementus priklausančius šiai formai.

3. (6 žingsnis sekų diagramoje) Nuskaitomas formos dalies (epizodo) aprašas iš lentelės ‚f_dalis‘

- **XSLT:** Suformuojamas XHTML kodas aprašantis, kaip nagrinėjama formos dalis bus vaizduojama grafinėje sąsajoje – jos plotis, aukštis, pavadinimas ir koordinatės.

4. (7-15 žingsniai sekų diagramoje) Nuskaitoma informacija apie jai priklausantį grafikos elementą iš lentelės ‚f_elementas‘

- **XML:** pagal tai ar elementas yra grupinis ar ne (ar turi aprašą lentelėje ‚element_grupe‘) formuojamas jį apibūdinantis kodas: koordinatės erdvėje, plotis, aukštis, šrifto dydis, rėmelio plotis, lygiavimas, tipas, reikšmė.

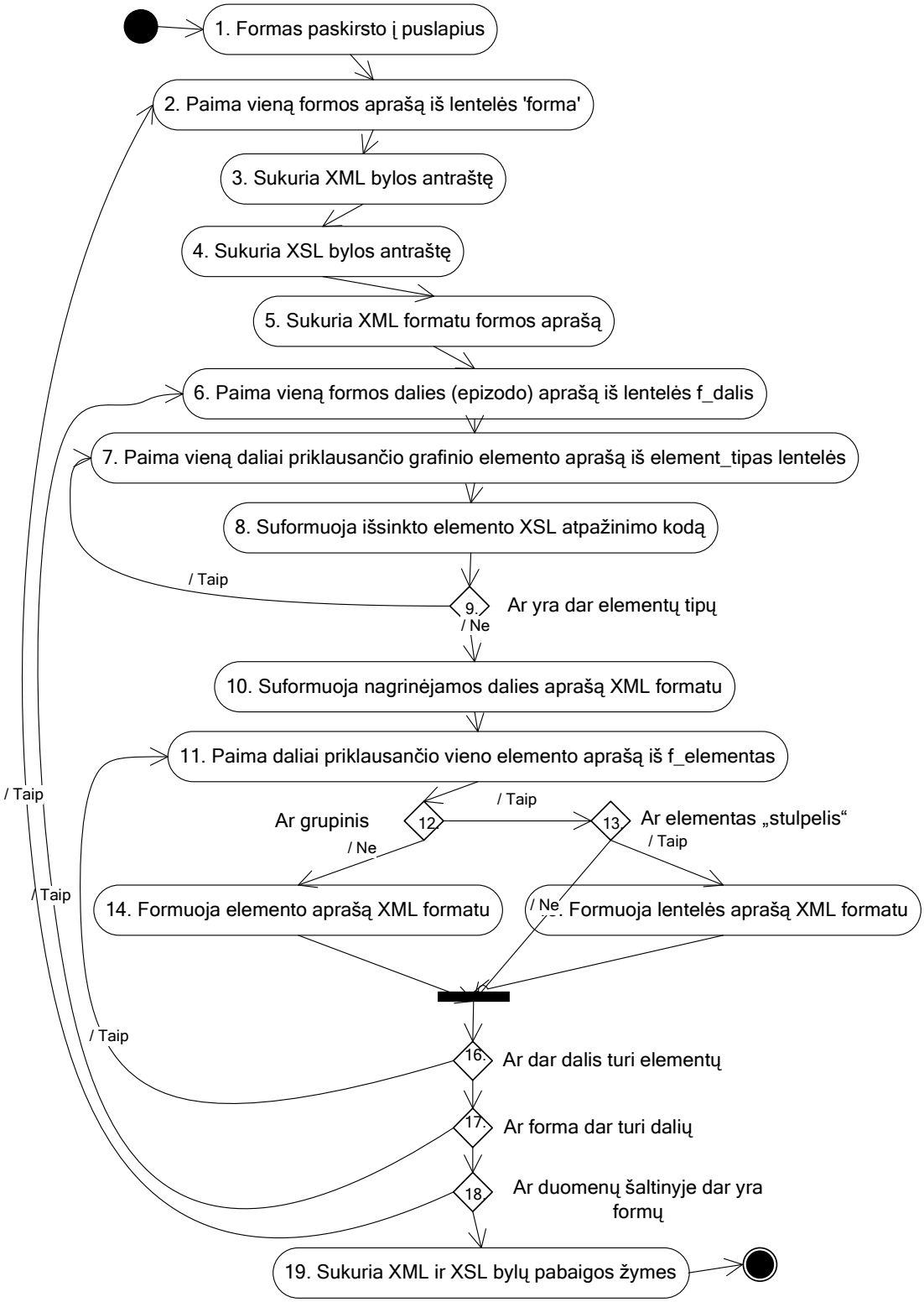
  Ne grupiniams elementams taikomas vienodos XML formavimo sąlygos. Grupiniams elementams (pvz. ‚lentelė‘, kuri formuojama iš elementų ‚stulpelis‘) aprašomos kitos kodo generavimo taisyklės priklausomai nuo elementų grupavimo charakteristikos.

  Pvz. grupinio elemento ‚lentelė‘ formavimas:

  Kaip jau minėta anksčiau, XHTML‘e lentelė formuojama pirmiausiai eilučių lygmenyje. Tačiau duomenų bazėje lenteles apsirašome stulpelių lygmenyje – taip kiekvienas stulpelis gali turėti savo antraštę (header) ir nereikalauja papildomos lentelės jų saugojimui (ką tektų įgyvendinti, jei lenteles aprašytume eilučių lygmenyje). XSL transformacijų rinkmenos pagalba, lentelę suformuoti XHTML kodu yra paprasčiau jei XML kodas ją apibūdina eilučių lygmenyje. Tad šios XML kodo generavimo dalies uždavinys – stulpelius „paversti“ į eilutes. Kadangi duombazėje nėra tiesiogiai aprašomos lentelės koordinatės ekrane, reikia pasinaudoti stulpelių koordinatėmis: lentelės dešinys viršutinis kampas bus nutolęs tiek, kiek yra nutolęs paskutinysis ją sudarantis stulpelis. Atributą ‚a‘ (aukštis) šį kartą naudojame nusakyti eilučių skaičiui stulpelyje, o kiekviena eilutė pagal nutylėjimą suformuojama 25 pikselių aukščio.

- **XSLT:** Kiekvienam galimam elementui suformuojama CSS klasė, aprašanti absoliutines jo koordinates ekrane.

5. (16 žingsnis sekų diagramoje) Tikrinama ar dar yra neaprašytų formos dalies elementų. Jei taip, grįžtama į 4-tą punktą.
6. (17 žingsnis sekų diagramoje) Tikrinama ar dar yra neaprašytų formos dalių. Jei taip, grįžtama į 3-tą punktą.
7. (18 žingsnis sekų diagramoje) Tikrinama ar dar yra neaprašytų formų. Jei yra, grįžtama į 2-trą punktą.
8. Pabaiga
    - **XSLT:** Galutinai suformuotoje XSLT byloje kiekviena formos dalis turi atskiras aprašytų elementų interpretavimo ir atvaizdavimo XHTML kodu taisykles. Pagal nutylėjimą visose dalyse jos vienodos, tačiau sistemos vartotojui suteikiama galimybė nepriklausomai keisti kiekvieno formos epizodo grafinį vaizdą. Kadangi ši XSL transformacijų rinkmena yra šablonas čia aprašomi visi atpažįstami elementai, nežiūrint į tai ar jie panaudoti nagrinėjamoje formoje.



26 pav. Grafinės vartotojo sąsajos generavimo algoritmo veiksmų sekų diagrama

4. GUI generatoriaus programos prototipas

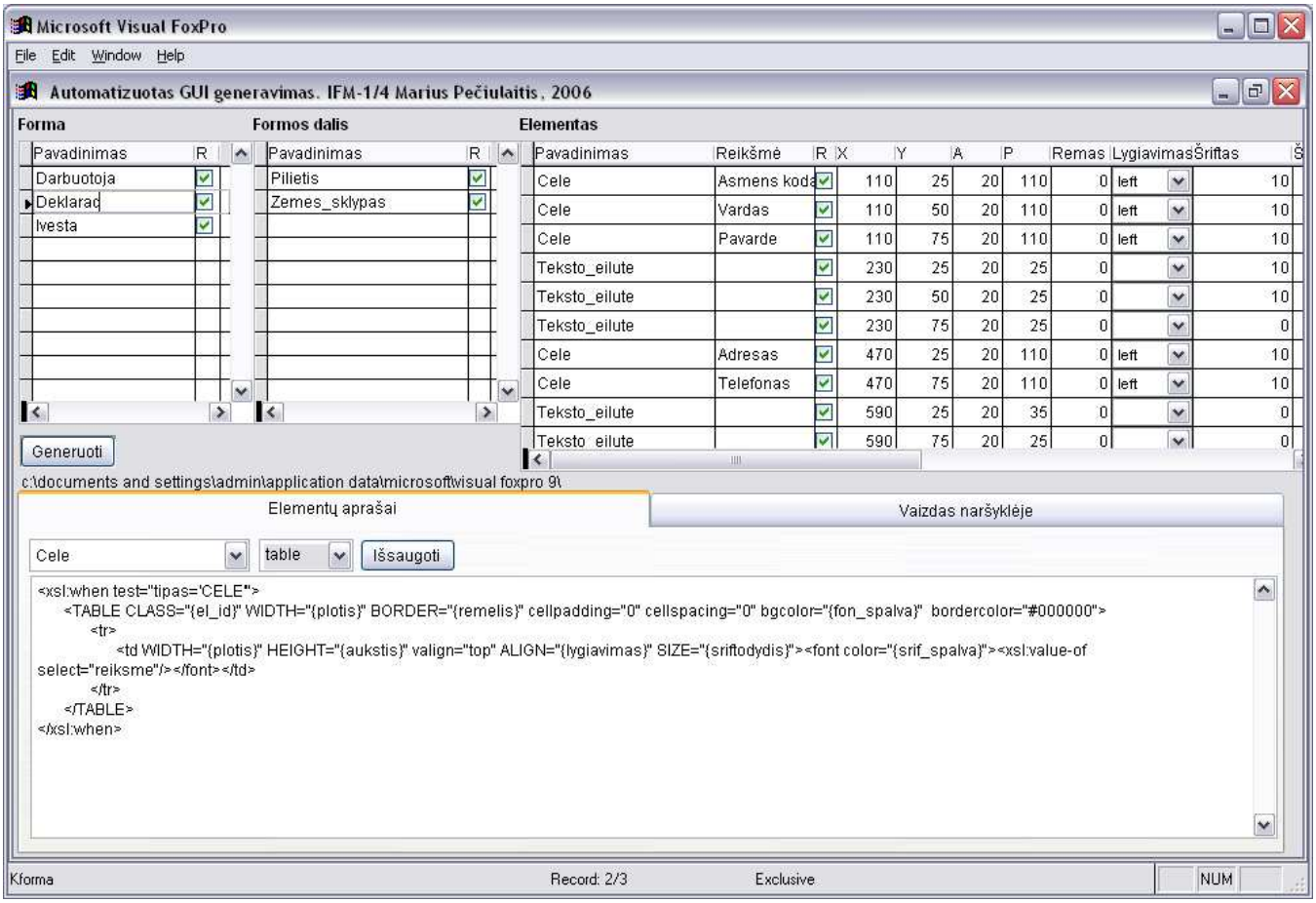
XML ir XSL kodo generavimui pasirinkta Microsoft korporacijos duomenų bazių valdymo sistemų kūrimo priemonė Visual FoxPro 9. Realizuojant generatorių buvo siekiama, kad juo galėtų naudotis kuo platesnis programinės įrangos kūrėjų ratas todėl vienas iš uždavinių – sistemą padaryti kuo lankstesnę, paprastesnę ir lengviau prisitaikomą pagal savo poreikius. Sukurtame įrankyje nekeičiant Visual FoxPro programinio kodo galima operatyviai keisti grafikos elementų pozicionavimo ir fizines charakteristikas. Šis procesas atliekamas rankinių būdu aprašant XML kodui interpretuoti ir elementams atvaizduoti taikomas taisykles XSL kalba ir šią informaciją išsaugant duomenų bazėje. Programinė įranga šiuo atžvilgiu atlieka tik taisyklių apjungimą ir XSLT bylos suformavimą.

Mažiau lanksti pokyčiams sistemos dalis – XML kodo generavimas. Visos taisyklės aprašomos Visual FoxPro kalba ir saugomos programos vidiniame kode.

Realizuojant automatizuotą GUI generavimą, buvo atlikti papildymai ODRES metodo reikalavimų saugyklos metamodelyje:

- Lentelės papildytos atributais aprašyti elemento teksto šrifto dydžiui, spalvai, bei lygiavimui; rėmelio pločiui, fono spalvai.
- Papildomai sukurtos lentelės elementų grupėms, elementų tipams, formų dalims bei jų tipams.
- Lentelė DS_atributas išskaidyta į dvi dalis (DS_atributas ir F_elementas)

GUI generatoriaus prototipas sudarytas iš vieno programinio modulio ir vieno vartotojo sąsajos lango (a). Detali instrukcija, kaip naudotis įrankiu pateikta 10.4.2 priede.



27 pav. pav. GUI generatoriaus prototipo pagrindinis langas

Sistemos prototipe generuojamos XSL transformacijos šablono struktūros epizodas su fragmentų paaiškinimais:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output method="html" version="1.0" encoding="UTF-8" indent="yes" />
  - <xsl:template match="/">
    - <html>
      - <head>
          <title>Dokumento pavadinimas</title>
          <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        </head>
      - <body>
```

Konkrečiai formai taikomų taisyklių rinkinys

```
  - <xsl:for-each select="dokumentas/forma[vardas='Ivesta_7']">
    - <form name="{vardas}" method="post" action="xml_failas_{dalis/elementas[tipas='MYGTUKAS'][rysys!='0']/rysys}.xml">
        <style type="text/css">table.Issaugota_7_dalis { position:absolute; left:20px; top:10px; }</style>
      - <table class="Issaugota_7_dalis" border="1" width="{dalis[vardas='Issaugota']/plotis}" cellpadding="0" cellspacing="0">
        - <tr>
            <td width="{dalis[vardas='Issaugota']/plotis}" height="{dalis[vardas='Issaugota']/aukstis}" />
          </tr>
        </table>
        <style type="text/css">table.Issaugota_7_antraste { position:absolute; left:30px; top:0px; }</style>
      - <table bgcolor="#FFFFFF" class="Issaugota_7_antraste" border="0" cellpadding="0" cellspacing="0">
        - <tr>
            <td>Issaugota</td>
          </tr>
        </table>
```

Formos dalies elementų formavimas

```
    - <xsl:for-each select="dalis[vardas='Issaugota']/elementas">
```

Elemento "Mygtukas" formavimas

```
      - <xsl:choose>
        - <xsl:when test="tipas='MYGTUKAS'">
            <input class="{el_id}" type="submit" name="{el_id}" value="{reiksme}" />
```

CSS kodas mygtukui

```
          - <style type="text/css">
              input.
              <xsl:value-of select="el_id" />
              { position:absolute; left:
              <xsl:value-of select="x_koord" />
              px; top:
              <xsl:value-of select="y_koord" />
              px; }
            </style>
          </xsl:when>
        - <xsl:when test="tipas='LENTELE'">
          - <TABLE BORDER="{remelis}" NAME="{el_id}" class="{el_id}">
            - <xsl:for-each select="eilute">
              - <tr>
                - <xsl:for-each select="stulpelis">
                  - <td WIDTH="{plotis}" HEIGHT="{aukstis}" valign="top" ALIGN="{lygiavimas}">
                    - <font size="{sriftodydis}">
```

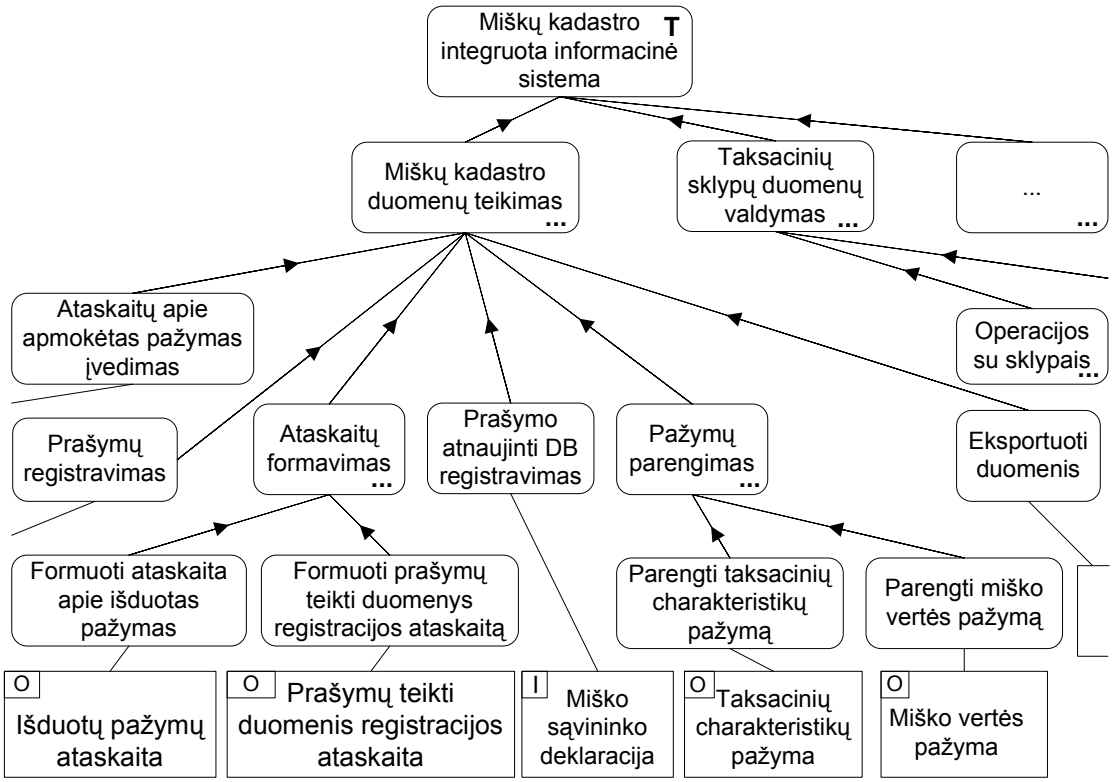
28 pav. XSLT fragmentas

5. Eksperimentinis grafinės vartotojo sąsajos generavimo sistemos panaudojimas

Šiame skyrelyje pateikiamas eksperimentinis pavyzdys, kuriame atliekamas duoto dalykinės srities uždavinio kompiuterizavimas. Reikalavimų specifikavimas ir projektavimo etapai atliekami pagal ODRES metodo procesą.

Eksperimentui pasirinktas Duomenų atnaujinimo pagal miško savininkų pateiktą informaciją apie įvykdytus miško kirtimus uždavinys. Šis uždavinys svarbus šaltinis miškų kadastro duomenų apie privačias miško valdas papildymui. Miško savininkai pateikia užpildytą *Miško savininko deklaraciją* (žr. 3 pav.), kurios duomenis panaudojami išduodant miškų kadastro pažymas.

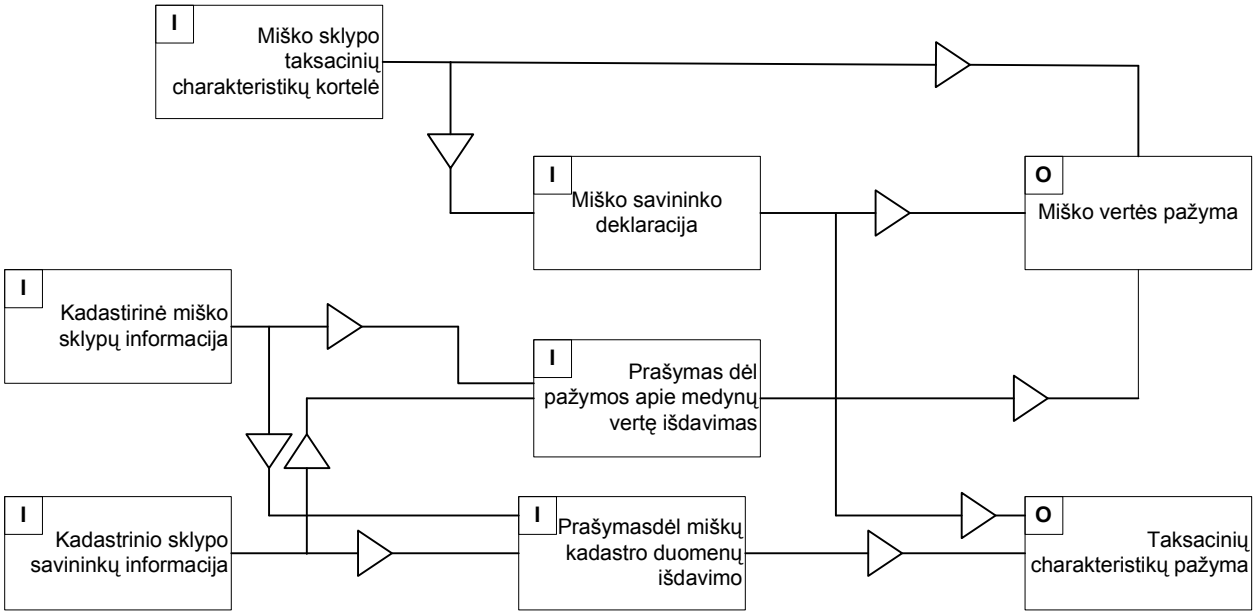
Pagal Odres metodo procesą apibrėžiamas kompiuterizuojamos veiklos kontekstas. Tam sudaromas modifikuotas funkcijų hierarchijos modelis, kuris pateikiamas **29 pav.**.



29 pav. Miškų kadastro integruotos informacinės sistemos konteksto modelio epizodas

Kadangi kompiuterizuoti pasirinkta funciją *Prašymo atnaujinti DB registravimas*, todėl toliau reikalavimų specifikavimo ir projektavimo procesas bus atliekamas tik šiai funkcijai. Ši funkiją susijusi su duomenų šaltiniu *Miško savininko deklaracija*. Sekančiame etape sudarome ryšių tarp informacijos srautų struktūros modelį (žr. **30 pav.**). Jame pateikiame informacija, kuri rodo, kaip analizuojamas duomenų šaltinis susijęs su kitais nagrinėjamos kontakto duomenų šaltiniais ir rezultatais. Šis modelis panaudojamas sudarant navigacijos tarp langų medį arba IS menių struktūra. Iš šio modelio matyti, kad

norint suformuoti nagrinėjamą duomenų šaltinį reikalinga *Miško sklypo taksacinių duomenų kortelės* informacija, o taip pat šis duomenų šaltinis bei *Miško savininko deklaracija* naudojami suformuoti *Miško vertės pažymą*.



30 pav. Ryšių tarp informacijos srautų struktūros modelis

Tolimesnio žingsnelio metu pagal miško savininko deklaracijos šabloną (žr. **31 pav.**) sudaromas lokalus duomenų bazės modelis, kuris pateikiamas **32 pav.**. Šis duomenų bazės modelis vadinamas lokalus, nes jis apima tik vieno nagrinėjamo duomenų šaltinio duomenų sritį. Integruojant visų kompiuterizuojamos dalykinės srities duomenų šaltinių modelius sudaromas galutinis kuriamos IS duomenų modelis.

Toliau sudaromas duomenų šaltinio apdorojimo etapų modelis (žr. **33 pav.**), kuris parodo duomenų šaltinio apdorojimo procesą. Iš sudaryto modelio matyti, kad miško savininkas pateikia deklaracija valstybinės miškotvarkos tarnybos specialistui (darbuotojui), kuris šią deklaraciją užregistruoja informacinėje sistemoje. Iš šio modelio jau galima numatyti, kurie etapai turi būti kompiuterizuojami. Nagrinėjamo modelio atvejų kompiuterizuojamas etapas *Registruoti prašyma atnaujinti duomenis.*

Visa ODRES proceso metu specifikuojamas informacija saugoma saugykloje, kurios struktūros epizodas susijęs su vartotojo sąsajos projektavimu pateikiamas **32 pav.**.

Pilietis (ė): ----------

Asmens kodas: ----------

Adresas: ----------

VALSTYBINĖS MIŠKOTVARKOS TARNYBOS DIREKTORIUI

PRAŠYMAS ATNAUJINTI MIŠKŲ KADASTRO DUOMENIS

Prašau atnaujinti man priklausančiame žemės sklype esančių miškų, įregistruotų Lietuvos Respublikos miškų valstybės kadastre, kadastro duomenis.

Duomenys apie žemės sklypą:

Registro įrašo Nr. ..... Kadastro Nr. .....

Bendras žemės sklypo plotas .....ha, iš jo miškas…….ha.

ANKETA

1. Valdai yra parengtas individualus miškotvarkos projektas? (TAIP / NE)

2. Valdoje yra vykdyti miško kirtimai:

Urėdija: …….....
Girininkija: .....…...

Kvartalo numeris	Taksacinio miško sklypo numeris	Taksacinio miško sklypo dalis	Taksacinio miško sklypo plotas, ha	Kirtimo intensyvumas, procentais	Kirtimo metai	Pastabos

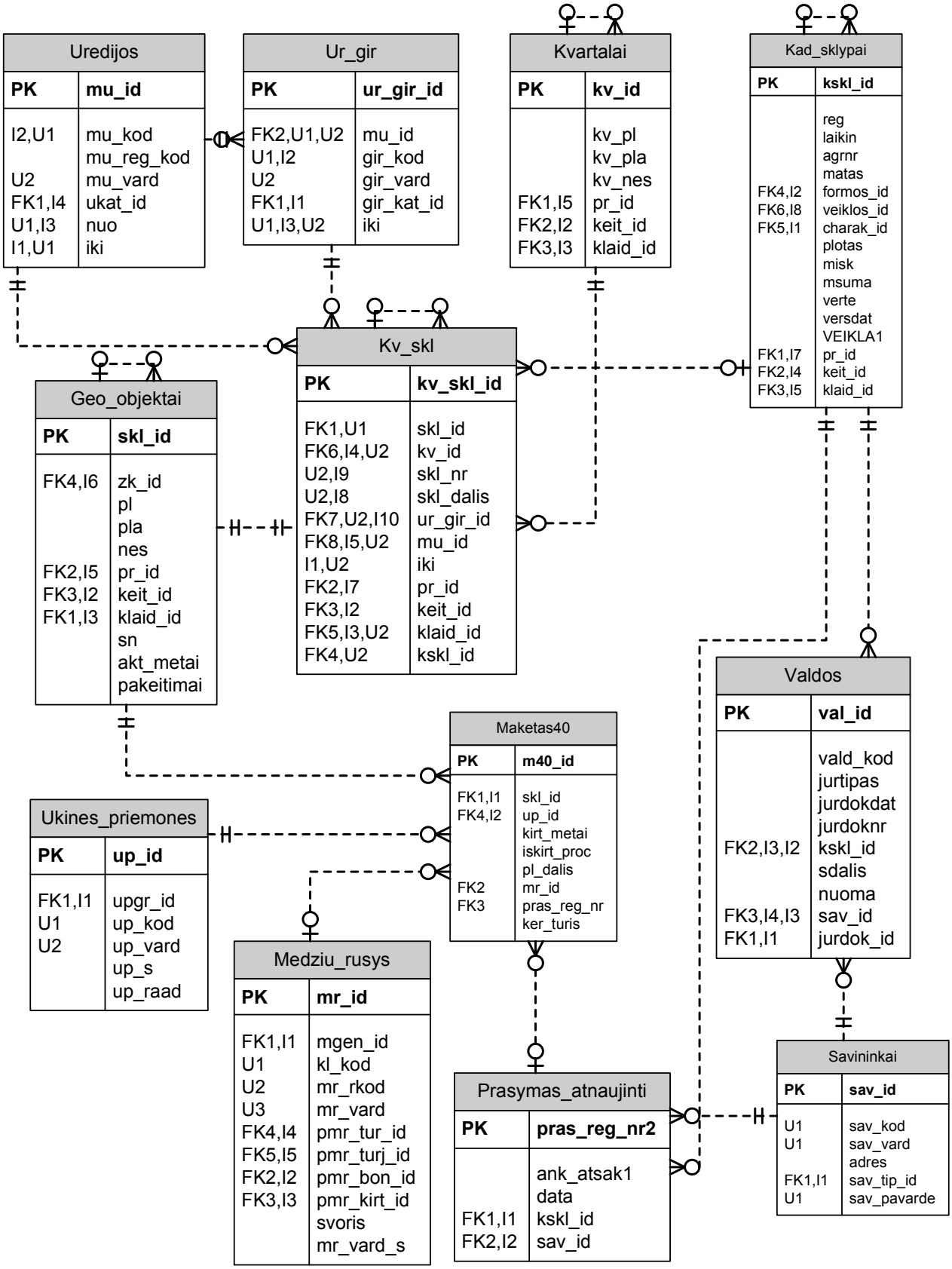
Pastabos:

- Kvartalų ir taksacinių sklypų numeriai surašomi pagal individualų miškotvarkos projektą, o jo nesant – pagal valstybinės miškų inventorizacijos duomenis.
- Plotai pildomi 0,1 ha tikslumu.
- Kirtimo intensyvumas procentais – skaičius, parodantis, kokia tūrio dalis iš medyno yra iškirsta.

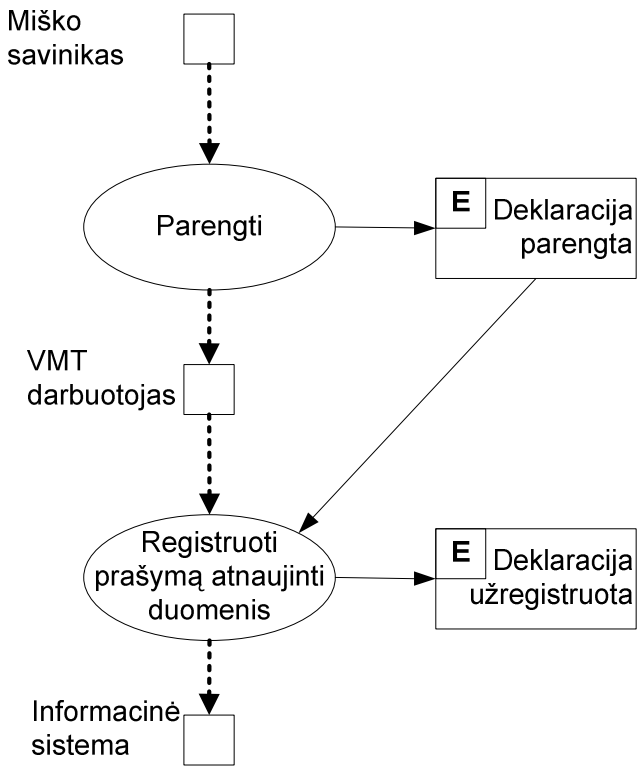
Garantuoju, kad anketoje pateikti duomenys yra teisingi.

....
Data *Parašas*

31 pav. Miško savininko deklaracijos eskizas

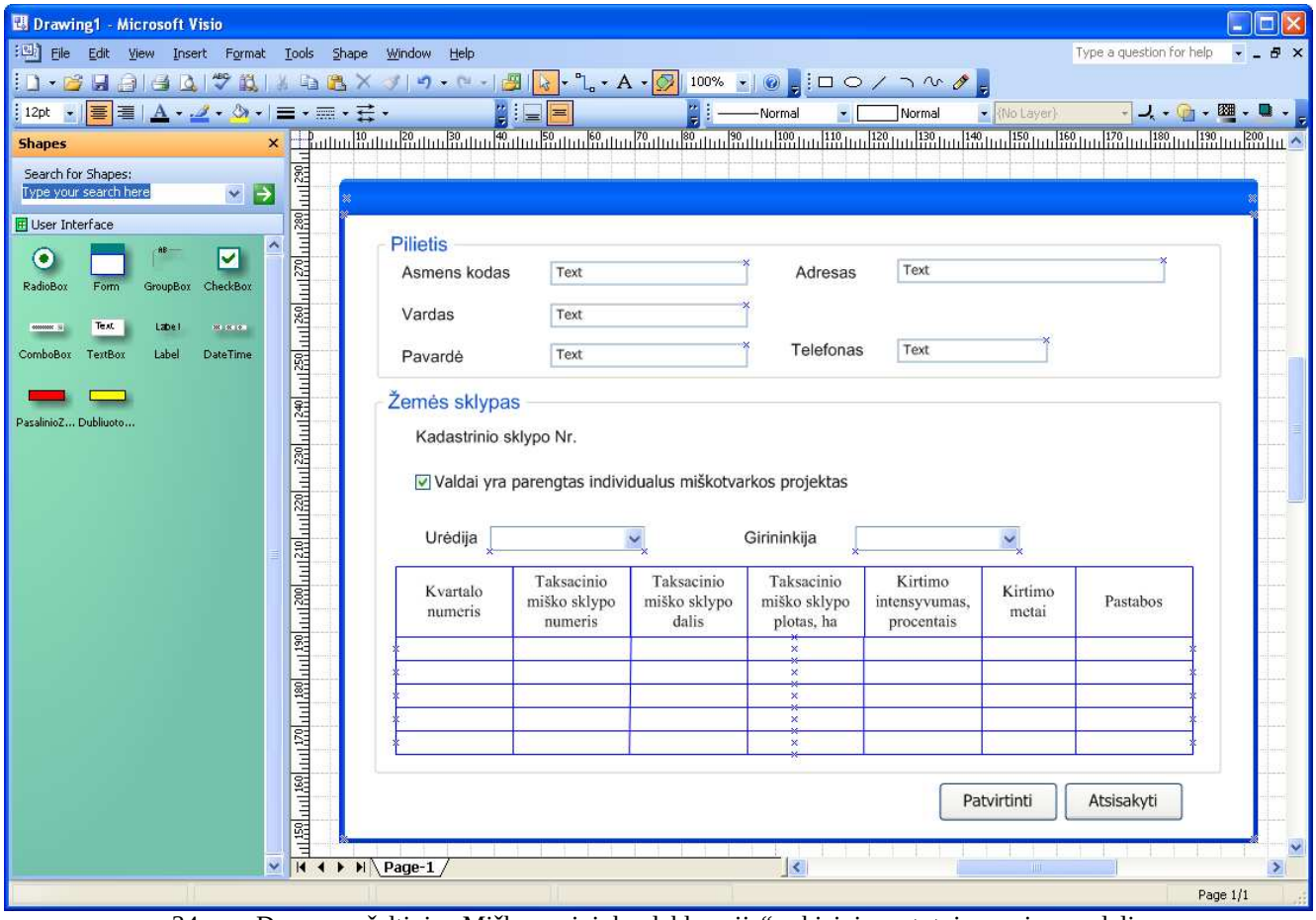


32 pav. Duomenų šaltinio „Miško savininko deklaracija“ duomenų bazės modelis



33 pav. Duomenų šaltinio apdorojimo etapų diagrama

Pasinaudojant vartotojo sąsajos projektavimo įrankių, kuris realizuotas įrankio Visio 2002 pagrindu sudaromas *Miško savininko deklaracija* eskizinis vartotojo sąsajos modelis (žr. **34 pav.**). Sudaryto modelio pirminis variantas sugeneruojamas automatizuotai, o sistemos projektuotojas šį modelį gali pakoreguoti rankiniu būdu. Eksperimento metu vartotojo sąsajos eskizą buvo stengiamasi priartinti prie *Miško savininko deklaracijos* šablono išvaizdos. Sudarytas modelis išsaugomas, kaip atskiras dokumentas, o taip pat visa informacija išsaugoma ir iš ODERS metodo saugykloje. Šio modelio išsaugoti duomenys pateikiami 9 priede.



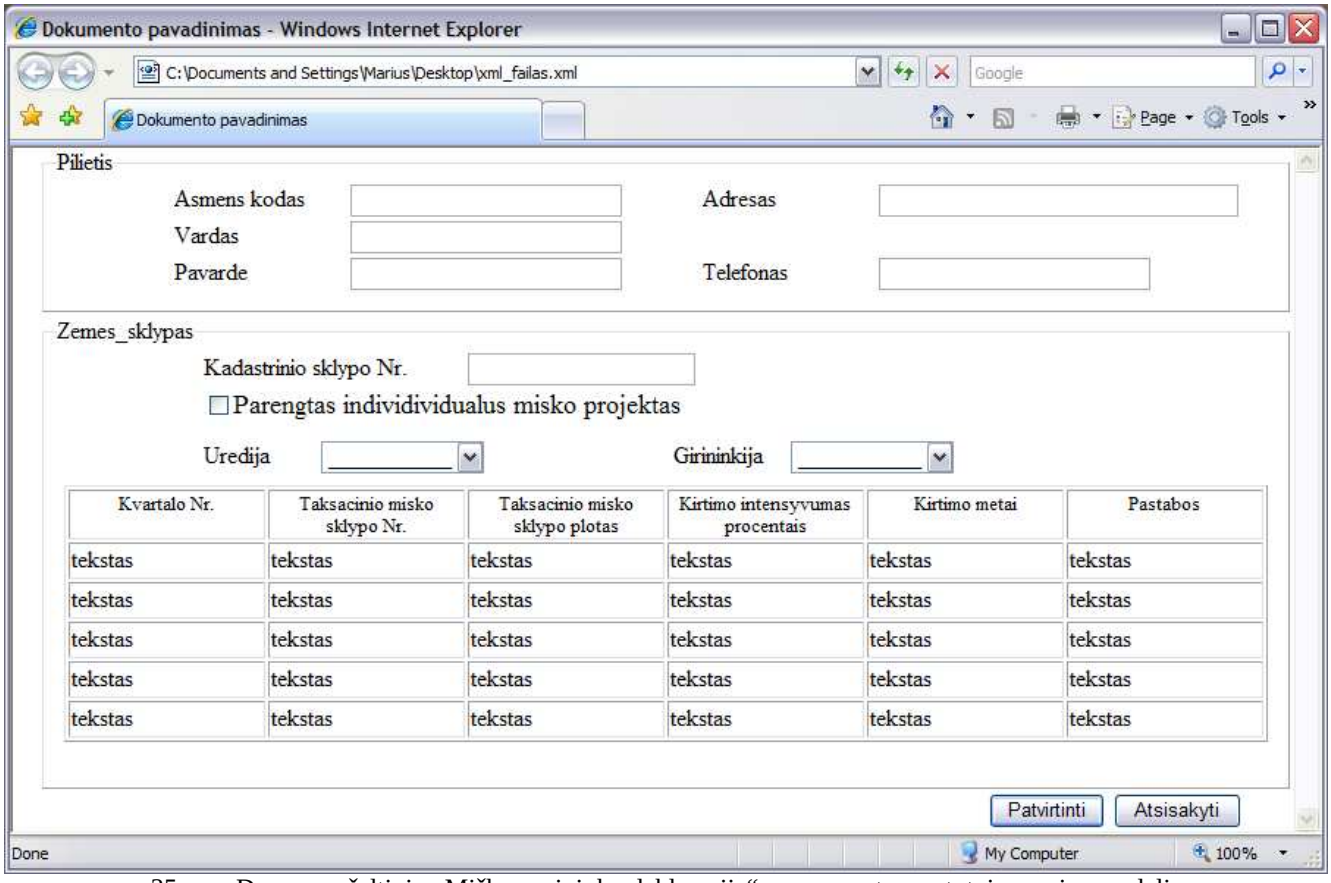
34 pav. Duomenų šaltinio „Miško savininko deklaracija“ eskizinis vartotojo sąsajos modelis

Dabar reikalavimų IS specifikacijos ir projektinių sprendimų saugykloje turima visa informacija iš kurios gali būti sugeneruota vartotojo sąsaja (10.1 priedas). Tam panaudojamas šio darbo metu sukurtas įrankis.

Įjungtas GUI generatorius automatiškai atsirenka iš duombazės visus IS specifikacijos duomenis reikalingus vartotojo sąsajos formavimui ir pateikia juos ekrane.

1. Pagrindiniame lange pažymime varnele formas, kurioms generuosime sąsają:
    - Deklaracija
    - Ivesta
2. Pasirinkus pirmąją formą, dešiniau, lentelėje „Formos dalys“ rodomi jos dalių pavadinimai:
    - Pilietis
    - Zemes_sklypas
3. Ties jais taip pat pažymime varneles, jog abi dalys bus įtraukiamos į GUI. Pasirenkant kiekvieną iš dalių lentelėje „Elementas“ rodomi grafikos elementų, naudojamų dalyje aprašai. Ties visais turi būti sužymėtos varnelės.

4. Du kartus spragtelėję ant po mygtuku „Generuoti“ rodomos nuorodos išsirenkame katalogą, kuriame bus išsaugomos sugeneruotos XML ir XSLT rinkmenos.
5. Paspaudus mygtuką „Generuoti“ pradedamas vykdyti GUI specifikacijos duomenų analizavimas ir grafinės vartotojo sąsajos formavimas.
6. Operacijai pasibaigus internetine naršykle atidarius vieną iš sugeneruotų XML rinkmenų ekrane parodoma galutinai suformuota grafinė vartotojo sąsaja (žr. **35 pav.**).

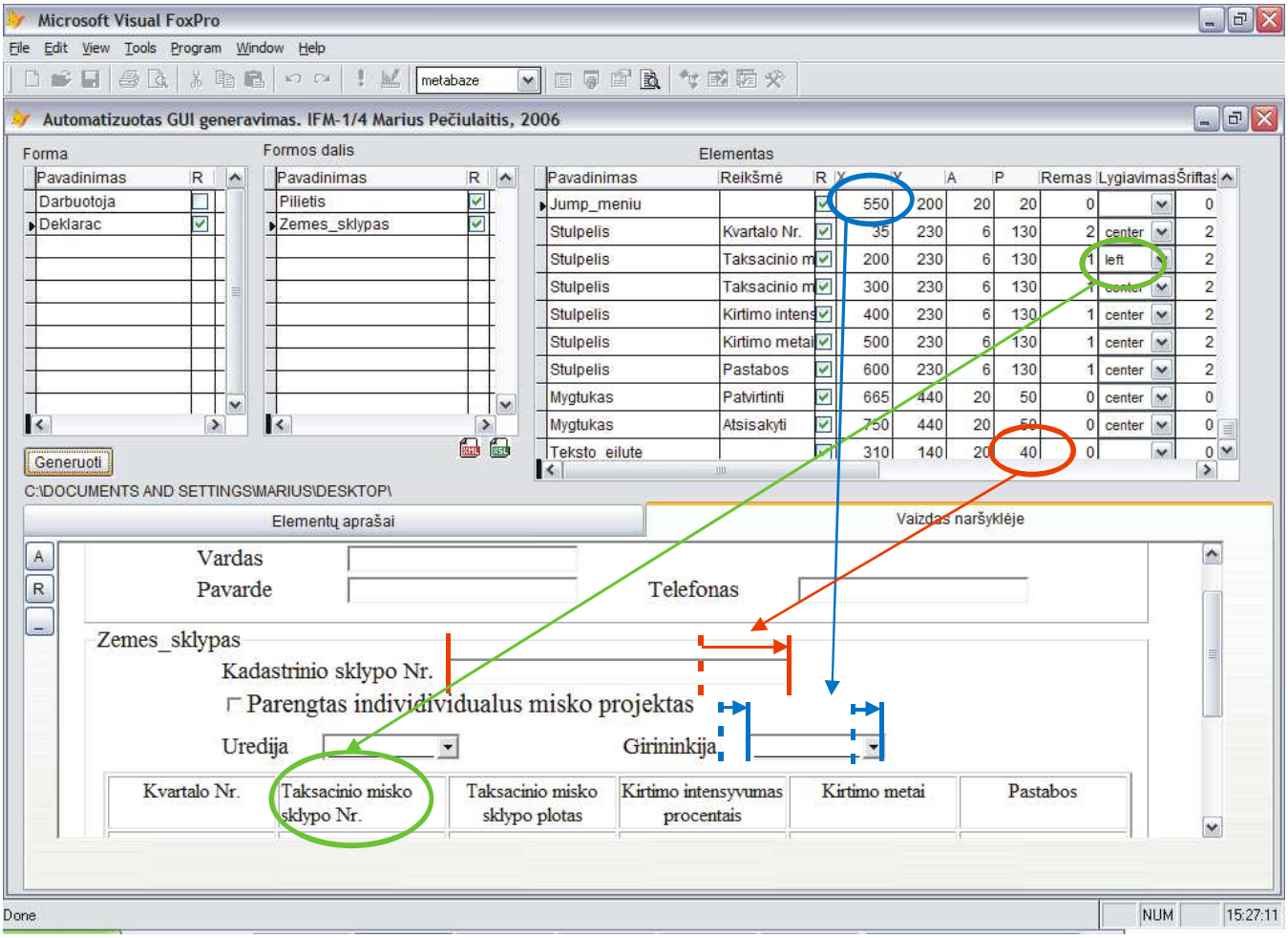


35 pav. Duomenų šaltinio „Miško savininko deklaracija“ sugeneruotas vartotojo sąsajos modelis

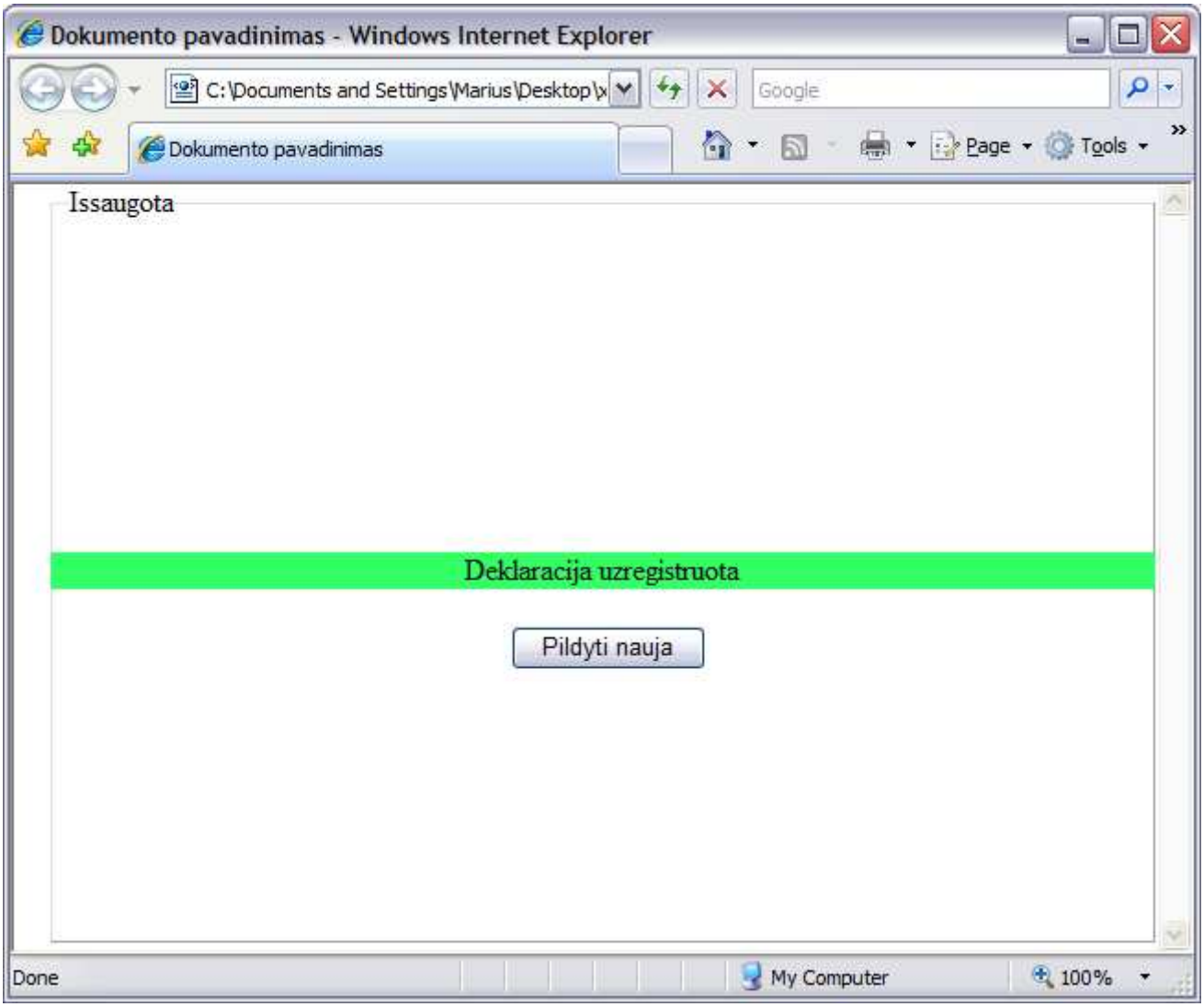
Sugeneruota GUI atitinka suprojektuotąją tiek kiek tai leidžia padaryti HTML elementai t.y. tereikia tinkamai aprašyti jų tipą bei koordinates. Ryšiai tarp GUI langų realizuoti programiškai todėl, mygtukas „patvirtinti“ funkcionalus – jį spragtelėjus atidaromas naujas tinklapio langas, informuojantis, jog duomenys išsaugoti (žr. **37 pav.**). Čia esančiu mygtuko paspaudimu vartotojas gali grįžti atgal į pildytą formą.

Prireikus atlikti smulkius elementų išdėstymo ar kitus jų išvaizdos parametrų pakeitimus, tai atliekama per GUI generatoriaus sąsają keliais mygtukų paspaudimais – pasikeitimus grafinėje sąsajoje galima stebėti tame pačiame programos lange (žr. **36 pav.**). Jeigu XSL taisyklių korekcijos metu nesilaikoma kalbos standartų, pvz. žymė parašoma didžiosiomis raidėmis, internetinė naršyklė generuoja

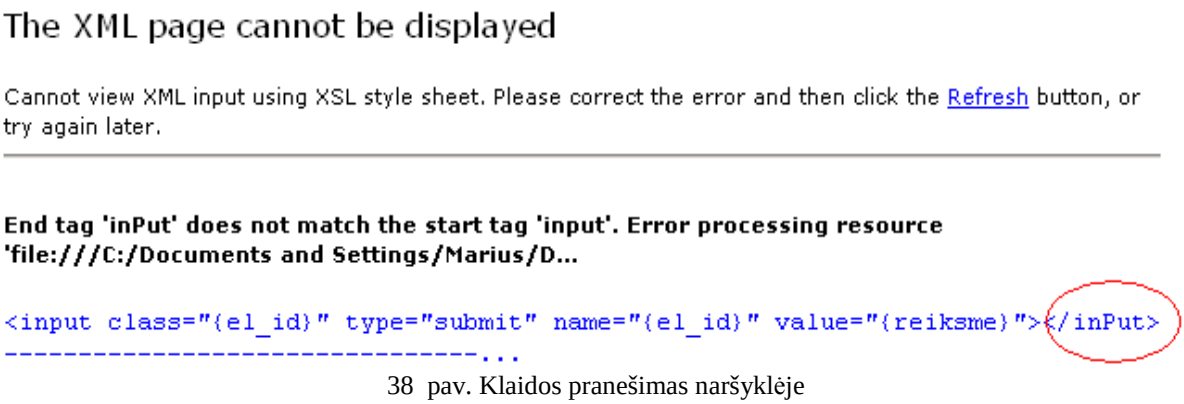
klaidos pranešimą (žr . **38 pav.**) ir vartotojo sąsajos ekrane nepateikia. XLST ir XML bylų kodai pateikti atitinkamai 10.2 ir 10.3 prieduose.



36 pav. Duomenų šaltinio „Miško savininko deklaracija“ generuojamos GUI korekcija programoje



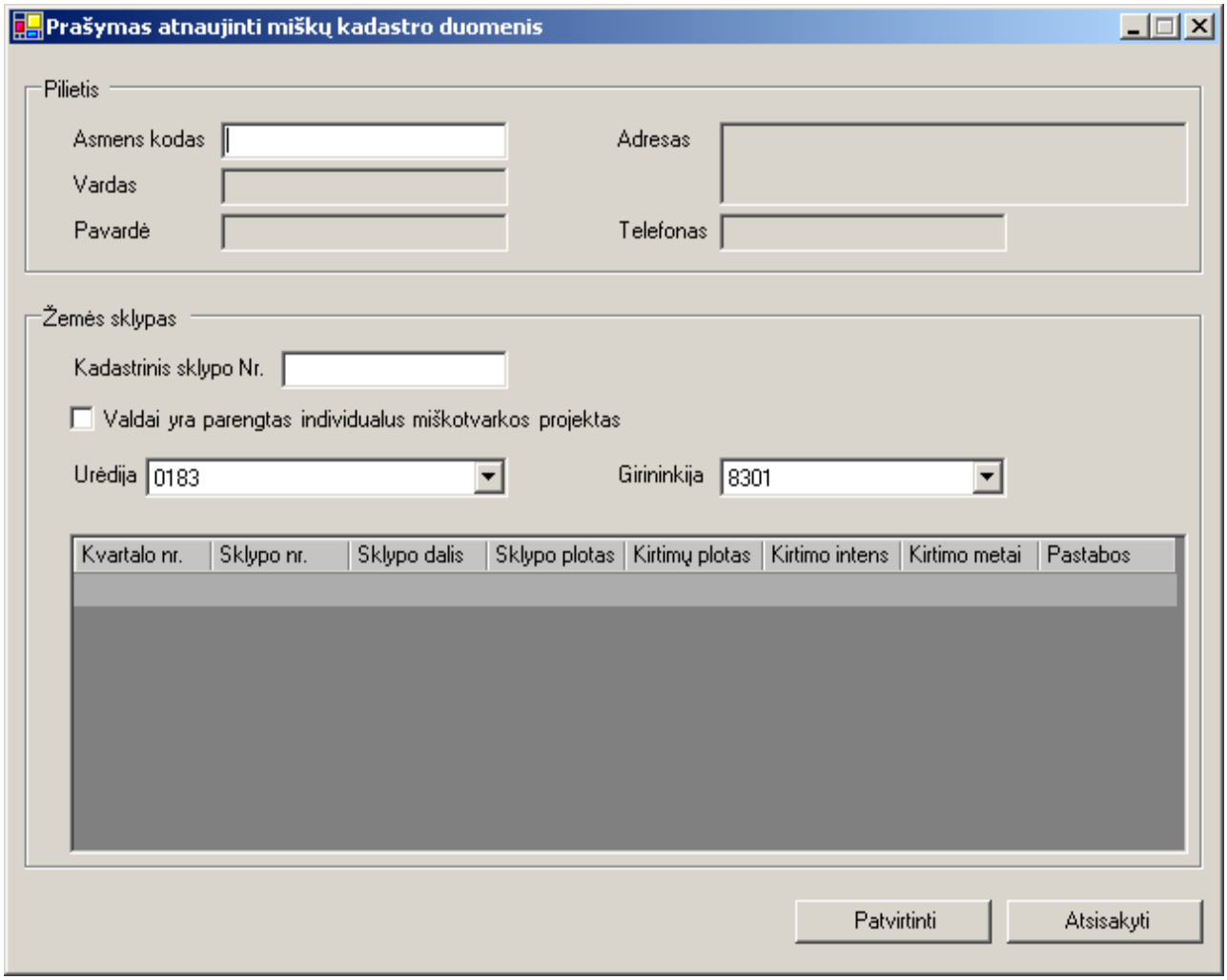
37 pav. Duomenų šaltinio „Miško savininko deklaracija“ sugeneruotas vartotojo sąsajos modelis



38 pav. Klaidos pranešimas naršyklėje

O **39 pav.** pateikiamas realiai sukurto produkto vartotojo sąsajos langas. Šis produktas buvo kuriamas kliento serverio architektūros pagrindu, kurio klientinė dalis realizuota, kaip tradicinė taikomoji programa. Iš paveikslėlio matosi, kad sukurtas produktas ir eksperimento metu sugeneruota vartotojo sąsaja praktiškai nesiskiria.

Tačiau reiktų vėlgi paminėti, kad pateiktas realus produktas yra sukurtas konkrečiai operacinei sistemai, t.y yra priklausomas nuvartotojo sąsajos, todėl šio produkto perkėlimas į kitą platformą reikalautų naujų resursų. Siūlomas internetinės vartotojo sąsajos modelis, be didesnių problemų, gali būti perkeliamas į bet kurią platforma įskaitant ir mobilius įranginius.



39 pav. Duomenų šaltinio „Prašymas atnaujinti MK duomenis“ realizuotos sistemos modulio vartotojo sąsaja

6. **Vertinamoji sukurtos sistemos analizė**

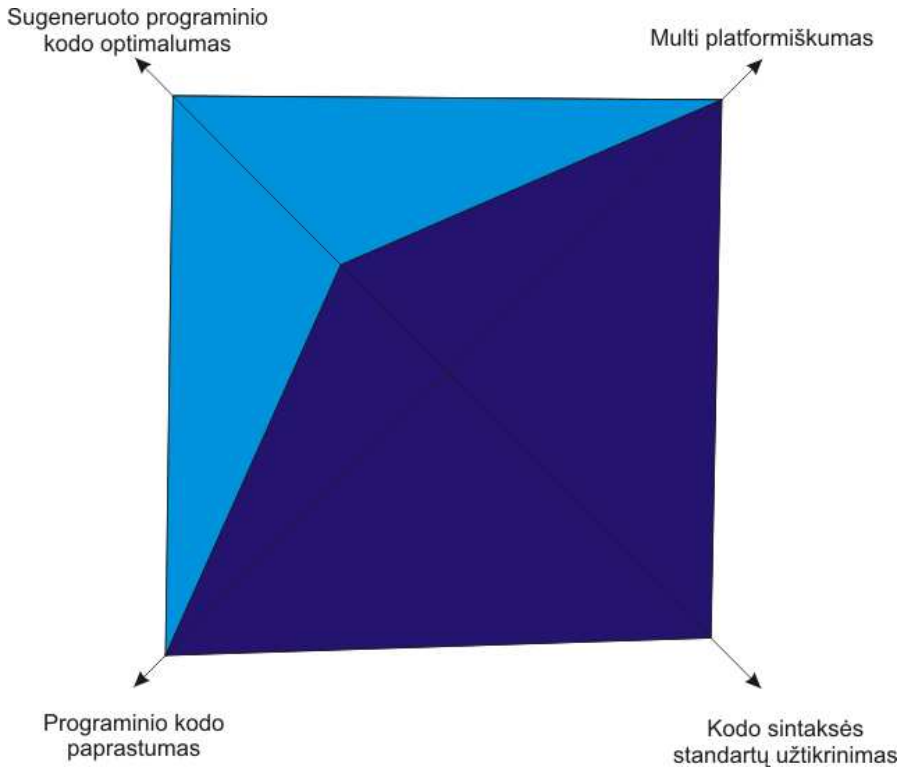
Formuojamas grafinės vartotojo sąsajos programinis kodas paremtas XML kalba bei CSS technologija. XML suteikia galimybę keistis informacija nepriklausomai nuo įrenginio ar programinės įrangos, todėl tai yra vienas iš pagrindinių žingsnių link multi platforminės GUI sukūrimo - suformuotas varotojo sąsajos aprašas bus suprantamas vienodai kiekvienoje sistemoje. Dėl tos pačios priežasties šiam aprašui interpretuoti pasirinkta XML paremta stilių šablonų kalba XSL. Tačiau galutiniam vaizdui išgauti nepakanka vien šių technologijų, tam reikalinga papildomai interneto naršyklė. Dauguma populiariausių šiuolaikinių naršyklių pilnai palaiko CSS, XML ir XSL technologijas, todėl galima teigti, kad sukurta GUI yra bent dalinai multi platforminė.

XSLT šablonas formuoja (X)HTML ir CSS kodą, todėl norint pakoreguoti generuojamą GUI sistemos vartotojui pakaks turėti minimalias žinias apie XML ir XSL. Mokėti dirbti su CSS nebūtina - šis kodas automatiškai sugeneruojamas interpretatoriaus. Didžiausią dėmesį vartotojas turėtų skirti mokymuisi programuoti (X)HTML - nuo to priklausys sugeneruotos GUI išbaigtumas ir patogumas naudoti. Tačiau palyginus su JAVA (kaip viena iš galimų grafinės vartotojo sąsajos kūrimo variantų), tai yra gerokai lengviau ir greičiau išmokti. Prie viso to, norint keisti XML, XSLT, (X)HTML ar CSS kodą nereikia jokio specialaus įrankio, pakanka paprasčiausio teksto redaktoriaus. Kuriant sistemą buvo siekiama, jog GUI generavimo procesą galėtų koreguoti kuo didesnis vartotojų ratas: XML ir XSLT bylos generuojamos pasinaudojus Visual FoxPro kalba, tačiau norint pakoreguoti GUI išvaizdą vartotojui nereikia mokėti šios kalbos, jam pakanka pakeisti grafikos elementų specifikacijas duomenų bazėje arba tiesiog XSLT šablonų byloje. Šias operacijas galima atlikti ir per GUI generatoriaus sąsają. Taigi grafinės vartotojo sąsajos formavimo kodas yra pakankamai paprastas (palyginus su JAVA, XIML ar XUL), o jo sintaksę galima greitai bei be didelių sunkumų išmokti savarankiškai.

Šiame sistemos prototipe kiekvienos formos dalies elementams interpretuoti atkartojamos vis tos pačios XSLT taisyklės. Dėl šios priežasties XSLT byla gerokai išsiplečia ir ją tampa sudėtingiau koreguoti rankiniu būdu, jei forma susideda iš daugelio dalių. Toks XSLT generavimo principas vartotojui suteikia platesnes galimybes nepriklausomai koreguoti tų pačių grafikos elementų vaizdavimą skirtingose formos dalyse, tačiau tai yra trūkumas jei norima kaip įmanoma labiau sumažinti XSLT apimtis ir neketinama tiem patiem grafikos elementam taikyti skirtingas atvaizdavimo formavimo taisykles. Ateityje šioje sistemoje galima įdiegti galimybę pasirinkti, kurį XSLT generavimo būda taikyti. Be to, kaip optimaliai apimties ir funkcionalumo atžvilgiu bus sugeneruotos XSL transformacijų bylos didžiąja dalimi priklauso ir nuo to, kaip pats sistemos vartotojas apsirašys grafikos elementus formuojančias XSLT taisykles.

XML konvertavimas XSLT pagalba į grafinę sąsają, priešingai nei siūlo Luxor-XUL ar XIML projektai, vykdomas kiekvieną kartą kai vartotojas atverčia tinklapį ir tai atliekama kliento pusėje. Tai reikalauja, jog kiekvieno vartotojo interneto naršyklė palaikytu abejas XML ir XSLT kalbas. Tinklapio administratoriaus atžvilgiu tai privalumas – prireikus pakoreguoti sugeneruotos GUI išvaizdą nebūtina vėl pakartotinai naudotis mūsų sukurtu įrankiu pergeneruoti visos vartotojo sąsajos. Pakanka tekstiniu redaktoriumi pakeisti jau suformuotas XML ar XSLT rinkmenas. Tai ypač pravartu, kai GUI koreguojama ne Windows, o kitose operacinėse sistemose, kuriose šis įrankis neveiktų. Tačiau vartotojo sąsajos specifikacijos interpretavimas vartotojo pusėje turi trūkumą. Keliuose tinklapiuose norint pateikti šiek tiek kitokią grafinę sąsają XSLT lygmenyje, su kiekvienu skirtingu transformacijų šablonu turėsime padaryti kopiją bei kartu pateikti GUI specifikaciją XML formatu. Tai aktualu kai norima sutaupyti vietos serverio informacijos kaupikliuose.

Kuriant tinklapių grafines sąsajas, kurios formuojamos kliento pusėje yra svarbu programuojant laikytis kalbos standartų jei norima, jog kiekviena naršyklė GUI atvaizduotu vienodai. XML paremtos kalbos reikalauja griežto sintaksės taisyklių laikymosi (priešingai pvz. nei HTML) ir grafinė vartotojo sąsaja nėra suformuojama iš viso jei kode yra bent viena kalbos sintaksės klaida. Jei GUI buvo išvydome vienoje naršyklėje, galima būti tikram, jog ir kitoje naršyklėje ji atrodys taip pat. Taigi šiame projekte panaudotos XML ir XSLT kalbos leidžia greitai įsitikinti ar sugeneruotas kodas atitinka visus standartus ir taip užtikrina tinklapio išvaizdos pastovumą skirtingose naršyklėse.



40 pav. Tikslų įgyvendinimas

7. Išvados

1. Išanalizavus privalumus ir trūkumus keletos galimų automatizuotų grafinės vartotojo sąsajos kūrimo metodų bei technologijų priimtas sprendimas, jog šiam projektui tinkamiausia yra panaudoti XML bei XSL kalbas ir sugeneruoti GUI pagal funkcinius reikalavimus, kurie suformuoti pagal ODRES metodo siūloma procesą yra saugomi duomenų bazėje. Priimant sprendimą buvo įvertintai tai, jog šios technologijos yra nepriklausomos nuo platformos, nereikalauja papildomos programinės įrangos (išskyrus naršyklę) grafinei vartotojo sąsajai atvaizduoti. Taip suformuota GUI paprasta koreguoti – tam pakanka tekstinio redaktoriaus, o pakeitimus galima atlikti tiek vartotojo sąsajos specifikacijos duomenų (XML) tiek ir jų interpretavimo lygmenyje (XSLT) – tai suteikia sąsajos formavimo procesui lankstumo. Tiek XML ir XSLT, tiek ir XHTML reikalauja griežto sintaksės taisyklių laikymosi, todėl taip lengviau užtikrinama, jog kodas bus interpretuojamas ir GUI atvaizduojama vienodai kiekvienoje naršyklėje. Pasitaikius bent vienam sintaksės standartų neatitikimui grafinė vartotojo sąsaja neformuojama, tokiu būdų pašalinant tikimybę, jog netikslumas vėliau gali būti ne vienodai interpretuojamas skirtingose interneto naršyklėse.
2. Atlikus palyginamąjį tyrimą tarp dviejų galimų grafikos elementų pozicionavimo ekrane būdų (HTML ir CSS) priimtas sprendimas papildomai panaudoti CSS technologiją. Taip gerokai supaprastinome XSLT kodo generatoriaus veikimo principą, sumažinome XSL transformacijų rinkmenų apimtis.
3. Realizuotas grafinės vartotojo sąsajos generatoriaus prototipas duomenų bazėje saugomą GUI specifikaciją transformuojantis į XML bylas, o šios informacijos interpretavimui ir GUI formavimui generuojantis taisyklių rinkinius XSLT. Sukurtame sistemos prototipe buvo siekiama įnešti kuo didesnį lankstumą generuojant vartotojo sąsają. Prototipe įdiegta XSL taisyklių kūrimo ir koregavimo funkcija, taip pat vartotojui suteikta galimybė be papildomų įrankių koreguoti duomenų saugykloje sukauptą vartotojo sąsajos specifikavimo informaciją. Kaip atrodo sugeneruota GUI galima stebėti tame pačiame programos lange papildomai nesinaudojant naršykle.
4. Eksperimentinio sąsajos generatoriaus bandymo metu suprojektuota miško kadastro duomenų pildymo sistema. Šią informaciją formalizavus ir perkėlus į Visual FoxPro duomenų bazę sukurtu įrankiu sugeneruota visiškai pradinius jos reikalavimus atitinkanti grafinė vartotojo sąsają.

5. Atlikus vertinamąją sukurto įrankio analizę paaiškėjo, jog pasirinkta GUI generavimo metodika didžiąja dalimi pateisino projekto tikslus. Vartotojo sąsajos specifikacija bei jos transformavimo šablonai paremti XML kalba todėl gali būti vienodai interpretuojami bet kurioje operacinėje sistemoje. Didelis skaičius šiuolaikinių naršyklių tai pat palaiko [21] ir CSS technologiją, tad galima teigi jog generuojama grafinė vartotojo sąsaja atitinka multi platforminės programinės įrangos reikalavimus [22]. Generuojamo programinio kodo optimalumą (apimties ir funkcionalumo atžvilgiu) gali valdyti pats vartotojas, o panaudotos technologijos griežtai reikalauja laikytis sintaksės standartų, antraip GUI ekrane neatvaizduojama.
6. Šio darbo metu atlikta keletas ODRES metodo specifikacijos saugyklos korekcijų bei papildymų: įnešti papildomi atributai žymintys teksto šrifto dydį, spalvą, rėmelio plotį, teksto lygiavimą, elementų žymes HTML kalboje, elementui taikomas XSL transformacijas. Papildomai sukurtos lentelės grafikos elementų tipams bei grupėms aprašyti, taip pat formos dalims bei jų tipams.
7. Darbo pagrindu parengtas ir perskaitytas straipsnis „Automatizuotas internetinių IS vartotojo sąsajų kūrimas“ 11-toje tarpuniversitetinėje VUKF, KTU ir VDU doktorantų ir magistrantų konferencijoje „Informacinės technologijos 2006“. Straipsnis publikuojamas konferencijos pranešimų rinkinio I dalyje, 110-115 psl.

8. Literatūros sąrašas

[1] Díaz, J. S.; López O. P.; Fons J. *From User Requirements to User Interfaces: A Methodological Approach. Department of Information Systems and Computation Valencia University of Technology*. Valencia, Spain, 2000. [žiūrėta 2006-01-20].

[2] Pfisterer Ch. *Semantic Description Language for Platform-Independent Graphical User Interfaces*. Institut für Visualisierung und Interaktive Systeme Universität Stuttgart. [žiūrėta 2006-01-18].

[3] Savidisa S.; Stephanidisa C. *Unified user interface design: designing universally accessible interactions*. 2003. [žiūrėta 2006-01-18].

[4] Duke D.; Fields B.; Harrison, M. D. *A Case Study in the Specification and Analysis of Design Alternatives for a User Interface*. Human-Computer Interaction Group, Department of Computer Science, University of York, UK, 1999. [žiūrėta 2006-01-18].

[5] Ambler S. W; User Interface Design: Tips and Techniques. President, Ronin International. 2000. [žiūrėta 2006-12-22].

[6] Eidukynaitė V. Projektavimo metodologijos ir technologijų apžvalga. KTU, 2003. [žiūrėta 2006-01-21].

[7] Balasubramanian V. *Supporting the User Interface Design Process with Hypertext Functionality*. E-Papyrus, Inc.Edison, NJ 08817, USA., and Graduate School of Management Rutgers University, Newark, NJ 07102, USA, 2004. [žiūrėta 2006-01-22].

[8] Jespersen J. W.; Linvald J. *Investigating User Interface Engineering in the Model Driven Architecture*. IT-University of Copenhagen, Glentevej 67, 2400 NV, Denmark, 2004. [žiūrėta 2006-01-22].

[9] Trætteberg H. *Model-based User Interface Design*. 2002. [žiūrėta 2006-01-22].

[10] P. P. da Silva; Paton N. W. *User Interface Modelling with UML*. Department of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, UK. [žiūrėta 2006-01-15].

[11] Aloia N.; Concordia C.; Paratore M. T. *Automatic Gui Generation For Web Based Information Systems*. Institute of Information Science and Technologies Italian National Research Council Via Moruzzi, 1 - CNR Research Area, Pisa (56100) Italy. [žiūrėta 2006-01-21].

[12] XUL Tutorial. 2005. [žiūrėta 2005-12-27]. Prieiga per internetą: http://xulplanet.com/tutorials/xultu/.
[13] Transforming XMI to HTML. [žiūrėta 2006-01-17]. Prieiga per internetą: http://www.objectsbydesign.com/projects/xmi_to_html.html

[14] XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999. [žiūrėta 2005-10-14]. Prieiga per internetą: http://www.w3.org/TR/xslt

[15] UML Resource Page. [Žiūrėta 2005 10 20], prieiga per internetą http://www.omg.org/technology/uml/index.htm

[16] HTML and XSLT by Bob DuCharmeAugust 30, 2000. [žiūrėta 2005-12-19]. Prieiga per internetą: http://www.xml.com/pub/a/2000/08/30/xsltandhtml/

[17] Butleris R.; Butkienė R.; Danikauskas T.; Jasiukevičius S. *Straipsniai, metodų aprašymai*. [žiūrėta 2005-10-03]. Prieiga per internetą: ftp://isd.ktu.lt/Isd/Butkiene/Metodo_Magistrantams

[18] XIML. [žiūrėta 2006-10-15]. Prieiga per internetą: http://www.ximl.org/about/basicDocs.asp
[19] J. Jelinek, P. Slavik. GUI Generation from Annotated Source Code. TAMODIA 2004, Prague, Czech Republic. [žiūrėta 2006-10-05]

[20] Boysen K. P.; Leavens G. T. *Automatically generating consistent graphical user interfaces using a parser generator*. 2004. Department of Computer Science, 226 Atanasoff Hall Iowa State University, Ames, IA 50011-1041 USA. [žiūrėta 2006-10-01].

[21] W3C. Cascading Stile Sheets. [žiūrėta 2006-12-11]. Prieiga per internetą: http://www.w3.org/Style/CSS/#browsers

[22] *Cross-platform*.Wikipedia The Free Encyclopedia. [žiūrėta 2006-12-11]. Prieiga per internetą: http://en.wikipedia.org/wiki/Multi-platform

[23] Kapočius K.; Danikauskas T. *The use of business rules for the specification of dynamic aspects of IS*. Information technology and control. Volume 35 number 3a. 2005. [žiūrėta 2006-12-11]

[24] Pečiulaitis M. *Automatizuotas internetinių IS vartotojo sasajų kūrimas*. 11-toji tarptautinė VUKHF, KTU ir VDU doktorantų ir magistrantų konferencijos pranešimų medžiaga. I dalis, 110-115 psl. VU. 2006

9. Santrauka anglų kalba

Automated GUI generation for WEB based Information Systems

In this work the main aspects were discussed that need to be evaluated by establishing graphical user interface. There were discussed some user interface languages and technologies such as XML, XUL, XIML, XSLT, CSS, XHTML, analyzed some used methods for automated user interface generation. On a basis of this analysis, the proper method was chosen to apply for generating of user interface using ODRES method results.

On the next stadeges of the work it was designed and developed prototype of the GUI generation system. The main goals were to develop cross-platform user interface, using as mutch as possible simplyer technologies along with that securing constant design on different web browsers.

Generation process operates using the GUI specification stored in the MS SQL database and converting it to XML and along with the XSLT files used as parser. After these files are opened using web browser, XML and XSLT code is processed to an XHTML code.

The process of the user interface generation is flexible and alows user change GUI's appearance on the fly adjusting it's specification stored on the database or XSLT rooles as well.

During this work were made some ot the modifications to the ODRES database: appended few more attributes, describing graphical elements characteristics and web form relations; added extra tables.

10.Priedai

10.1. Duomenų šaltinio „Miško savininko deklaracija“ GUI specifikacija duombazėje

4 lentelė. Lentelė „Forma“

id	pavadinimas	ds_id	a	p	x	y
1	Darbuotoja	0	800	600	800	600
4	Deklarac	0	800	600	800	600

5 lentelė. Lentelė „F_dalis“

id	f_id	f_dt_id	tev_fdt	pavadinimas	x	y	a	p
1	1	1	0	Darbuotojas	800	600	40	800
2	1	2	0	Darbuotojo duomenys	800	560	560	800
4	4	2	0	Pilietis	20	10	100	850
5	4	2	0	Zemes_sklypas	20	125	310	850

6 lentelė. Lentelė „F_dalies_tipas“

id	pavadinimas
1	Antraste
2	Ivedimo laukai

7 lentelė. Lentelė „Element_tipas“

id	kodas	pavadinimas	html_pavadinimas	xsl_kodas
1		Mygtukas	input	<xsl:when test="tipas='MYGTUKAS'"> <input class="{el_id}" type="submit" name="{el_id}" value="{reiksme}"></input> </xsl:when>
2		Stulpelis	table	<xsl:when test="tipas='LENTELE'"> <TABLE BORDER="{remelis}" NAME="{el_id}" class="{el_id}"> <xsl:for-each select="eilute"> <tr> <xsl:for-each select="stulpelis"> <td WIDTH="{plotis}" HEIGHT="{aukstis}" valign="top" ALIGN="{lygiavimas}"><xsl:value-of select="reiksme"/></td> </xsl:for-each> </tr> </xsl:for-each> </TABLE> </xsl:when>
3		Opcija	div	<xsl:when test="tipas='OPCIJA'"> <input name="{el_id}" type="radio" value="{reiksme}"><xsl:value-of select="reiksme"/></input> </xsl:when>

id	kodas	pavadinimas	html_pavadinimas	xsl_kodas
4		Teksto_eilute	input	<xsl:when test="tipas='TEKSTO_EILUTE'"> <INPUT CLASS="{el_id}" NAME="{el_id}" TYPE="text" SIZE="{plotis}" VALUE="{reiksme}" ></INPUT> </xsl:when>
5		Cele	table	<xsl:when test="tipas='CELE'"> <TABLE CLASS="{el_id}" WIDTH="{plotis}" BORDER="{remelis}" cellpadding="0" cellspacing="0" bgcolor="{fon_spalva}" bordercolor="#000000"> <tr> <td WIDTH="{plotis}" HEIGHT="{aukstis}" valign="top" ALIGN="{lygiavimas}" SIZE="{sriftodydis}"><xsl:value-of select="reiksme"/></td> </tr> </TABLE> </xsl:when>
7		Paveikslas	img	<xsl:when test="tipas='PAVEIKSLAS'"> <img CLASS="{el_id}" src="{reiksme}" ALT="{reiksme}" NAME="{el_id}" WIDTH="{plotis}" HEIGHT="{aukstis}" BORDER="{remelis}"> </img> </xsl:when>
9		Varnele	div	<xsl:when test="tipas='VARNELE'"> <input type="checkbox" name="{el_id}" width="{plotis}"><xsl:value-of select="reiksme"/></input> </xsl:when>
11		Jump_meniu	select	<xsl:when test="tipas='JUMP_MENIU'"> <select name="{el_id}" class="{el_id}" > <option>_____</option> </select> </xsl:when>

8 lentelė. Lentelė „Element_grupe“

id	e_t_id	pavadinimas
1	2	Darbuotojo_info
5	2	Zemes_skl

9 lentelė. Lentelė „F_elementas“

id	x	y	a	p	p_kriterijus	p_tipas	reiksme_default	f_d_id	e_t_id	ds_a_id	grupes_id	sriftodydis	remelis	lygiavimas	fspalva	sspalva	Link_f_id
24	110	25	20	110	0	0	Asmens kodas	4	5	0	0	10	0	left	#FFFFFF		0
25	110	50	20	110	0	0	Vardas	4	5	0	0	10	0	left	#FFFFFF		0
26	110	75	20	110	0	0	Pavarde	4	5	0	0	10	0	left	#FFFFFF		0
28	230	25	20	25	0	0		4	4	0	0	10	0				0
29	230	50	20	25	0	0		4	4	0	0	10	0				0
30	230	75	20	25	0	0		4	4	0	0	0	0				0
32	470	25	20	110	0	0	Adresas	4	5	0	0	10	0	left	#FFFFFF		0
35	470	75	20	110	0	0	Telefonas	4	5	0	0	10	0	left	#FFFFFF		0
36	590	25	20	35	0	0		4	4	0	0	0	0				0
37	590	75	20	25	0	0		4	4	0	0	0	0				0
39	130	140	20	180	0	0	Kadastrinio sklypo Nr.	5	5	0	0	10	0	left	#FFFFFF		0
42	130	165	20	250	0	0	Parengtas individividualus misko projektas	5	9	0	0	4	0		#FFFFFF		0
44	130	200	20	130	0	0	Uredija	5	5	0	0	10	0		#FFFFFF		0
46	210	200	20	20	0	0		5	11	0	0	0	0				0
48	450	200	20	130	0	0	Girininkija	5	5	0	0	10	0		#FFFFFF		0
49	530	200	20	20	0	0		5	11	0	0	0	0				0
51	35	230	6	130	0	0	Kvartalo Nr.	5	2	0	5	2	2	center	#FFFFFF		0
52	200	230	6	130	0	0	Taksacinio misko sklypo Nr.	5	2	0	5	2	1	center	#FFFFFF		0
54	300	230	6	130	0	0	Taksacinio misko sklypo plotas	5	2	0	5	2	1	center	#FFFFFF		0
55	400	230	6	130	0	0	Kirtimo intensyvumas procentais	5	2	0	5	2	1	center	#FFFFFF		0
56	500	230	6	130	0	0	Kirtimo metai	5	2	0	5	2	1	center	#FFFFFF		0
57	600	230	6	130	0	0	Pastabos	5	2	0	5	2	1	center	#FFFFFF		0
60	665	440	20	50	0	0	Patvirtinti	5	1	0	0	0	0	center	#FFFFFF		7
62	750	440	20	50	0	0	Atsisakyti	5	1	0	0	0	0	center	#FFFFFF		0
64	310	140	20	20	0	0		5	4	0	0	0	0				0

10.2. Duomenų šaltinio „Miško savininko deklaracija“ sugeneruoto vartotojo sąsajos modelio pagrindinio lango XSL kodas

```
<?xml version="1.0"
encoding="iso-8859-1" ?>
- <dokumentas>
- <forma>
<vardas>Deklarac_4</vardas>
- <dalis>
<vardas>Pilietis</vardas>
<aukstis>100</aukstis>
<plotis>850</plotis>
<x_koord>20</x_koord>
<y_koord>10</y_koord>
- <elementas>
<tipas>CELE</tipas>
<el_id>Cele_24</el_id>
<aukstis>20</aukstis>
<reiksme>Asmens
kodas</reiksme>
<plotis>110</plotis>
<x_koord>110</x_koord>
<y_koord>25</y_koord>
<sriftodydis>10</sriftodydis>
<remelis>0</remelis>
<lygiavimas>left</lygiavimas>

<fon_spalva>#FFFFFF</fon_s
palva>
<srif_spalva />
<rysys>0</rysys>
</elementas>
- <elementas>
<tipas>CELE</tipas>
<el_id>Cele_25</el_id>
<aukstis>20</aukstis>
<reiksme>Vardas</reiksme>
<plotis>110</plotis>
<x_koord>110</x_koord>
<y_koord>50</y_koord>
<sriftodydis>10</sriftodydis>
<remelis>0</remelis>
<lygiavimas>left</lygiavimas>

<fon_spalva>#FFFFFF</fon_s
palva>
<srif_spalva />
<rysys>0</rysys>
</elementas>
- <elementas>
<tipas>CELE</tipas>
<el_id>Cele_26</el_id>
<aukstis>20</aukstis>
<reiksme>Pavarde</reiksme>
<plotis>110</plotis>
<x_koord>110</x_koord>
<y_koord>75</y_koord>
<sriftodydis>10</sriftodydis>
<remelis>0</remelis>
<lygiavimas>left</lygiavimas>

<fon_spalva>#FFFFFF</fon_s
palva>
<srif_spalva />
<rysys>0</rysys>
</elementas>
- <elementas>
<tipas>TEKSTO_EILUTE</tipas>

<el_id>Teksto_eilute_28</el
_id>
<aukstis>20</aukstis>
<reiksme />
<plotis>25</plotis>
<x_koord>230</x_koord>
<y_koord>25</y_koord>
<sriftodydis>10</sriftodydis>
<remelis>0</remelis>
<lygiavimas />
<fon_spalva />
<srif_spalva />
<rysys>0</rysys>
</elementas>
- <elementas>
<tipas>TEKSTO_EILUTE</tipas>

<el_id>Teksto_eilute_29</el
_id>
<aukstis>20</aukstis>
<reiksme />
<plotis>25</plotis>
<x_koord>230</x_koord>
<y_koord>50</y_koord>
<sriftodydis>10</sriftodydis>
<remelis>0</remelis>
<lygiavimas />
<fon_spalva />
<srif_spalva />
<rysys>0</rysys>
</elementas>
- <elementas>
<tipas>TEKSTO_EILUTE</tipas>

<el_id>Teksto_eilute_30</el
_id>
<aukstis>20</aukstis>
<reiksme />
<plotis>25</plotis>
<x_koord>230</x_koord>
<y_koord>75</y_koord>
<sriftodydis>0</sriftodydis>
<remelis>0</remelis>
<lygiavimas />
<fon_spalva />
<srif_spalva />
<rysys>0</rysys>
</elementas>
- <elementas>
<tipas>CELE</tipas>
<el_id>Cele_32</el_id>
<aukstis>20</aukstis>
<reiksme>Adresas</reiksme>
<plotis>110</plotis>
<x_koord>470</x_koord>
<y_koord>25</y_koord>
<sriftodydis>10</sriftodydis>
<remelis>0</remelis>
<lygiavimas>left</lygiavimas>

<fon_spalva>#FFFFFF</fon_s
palva>
<srif_spalva />
<rysys>0</rysys>
</elementas>
- <elementas>
<tipas>CELE</tipas>
<el_id>Cele_35</el_id>
<aukstis>20</aukstis>
<reiksme>Telefonas</reiksme>
<plotis>110</plotis>
<x_koord>470</x_koord>
<y_koord>75</y_koord>
<sriftodydis>10</sriftodydis>
<remelis>0</remelis>
<lygiavimas>left</lygiavimas>

<fon_spalva>#FFFFFF</fon_s
palva>
<srif_spalva />
<rysys>0</rysys>
</elementas>
- <elementas>
<tipas>TEKSTO_EILUTE</tipas>

<el_id>Teksto_eilute_36</el
_id>
<aukstis>20</aukstis>
<reiksme />
<plotis>35</plotis>
<x_koord>590</x_koord>
<y_koord>25</y_koord>
<sriftodydis>0</sriftodydis>
<remelis>0</remelis>
<lygiavimas />
<fon_spalva />
<srif_spalva />
<rysys>0</rysys>
</elementas>
- <elementas>
<tipas>TEKSTO_EILUTE</tipas>
```

```
    <el_id>Teksto_eilute_37</el_id>
  <aukstis>20</aukstis>
  <reiksme />
  <plotis>25</plotis>
  <x_koord>590</x_koord>
  <y_koord>75</y_koord>
  <sriftodydis>0</sriftodydis>
  <remelis>0</remelis>
  <lygiavimas />
  <fon_spalva />
  <srif_spalva />
  <rysys>0</rysys>
    </elementas>
    </dalis>
- <dalis>

    <vardas>Zemes_sklypas</vardas>
  <aukstis>310</aukstis>
  <plotis>850</plotis>
  <x_koord>20</x_koord>
  <y_koord>125</y_koord>
- <elementas>
  <tipas>CELE</tipas>
  <el_id>Cele_39</el_id>
  <aukstis>20</aukstis>
  <reiksme>Kadastrinio sklypo Nr.</reiksme>
  <plotis>180</plotis>
  <x_koord>130</x_koord>
  <y_koord>140</y_koord>
  <sriftodydis>10</sriftodydis>
  <remelis>0</remelis>
  <lygiavimas>left</lygiavimas>

    <fon_spalva>#FFFFFF</fon_spalva>
  <srif_spalva />
  <rysys>0</rysys>
    </elementas>
- <elementas>
  <tipas>VARNELE</tipas>
  <el_id>Varnele_42</el_id>
  <aukstis>20</aukstis>
  <reiksme>Parengtas individividualus misko projektas</reiksme>
  <plotis>250</plotis>
  <x_koord>130</x_koord>
  <y_koord>165</y_koord>
  <sriftodydis>4</sriftodydis>
  <remelis>0</remelis>
  <lygiavimas />

    <fon_spalva>#FFFFFF</fon_spalva>
  <srif_spalva />
  <rysys>0</rysys>
    </elementas>
- <elementas>
  <tipas>CELE</tipas>
  <el_id>Cele_44</el_id>
  <aukstis>20</aukstis>
  <reiksme>Uredija</reiksme>
  <plotis>130</plotis>
  <x_koord>130</x_koord>
  <y_koord>200</y_koord>
  <sriftodydis>10</sriftodydis>
  <remelis>0</remelis>
  <lygiavimas />

    <fon_spalva>#FFFFFF</fon_spalva>
  <srif_spalva />
  <rysys>0</rysys>
    </elementas>
- <elementas>
  <tipas>JUMP_MENIU</tipas>

    <el_id>Jump_meniu_46</el_id>
  <aukstis>20</aukstis>
  <reiksme />
  <plotis>20</plotis>
  <x_koord>210</x_koord>
  <y_koord>200</y_koord>
  <sriftodydis>0</sriftodydis>
  <remelis>0</remelis>
  <lygiavimas />
  <fon_spalva />
  <srif_spalva />
  <rysys>0</rysys>
    </elementas>
- <elementas>
  <tipas>CELE</tipas>
  <el_id>Cele_48</el_id>
  <aukstis>20</aukstis>
  <reiksme>Girininkija</reiksme>
  <plotis>130</plotis>
  <x_koord>450</x_koord>
  <y_koord>200</y_koord>
  <sriftodydis>10</sriftodydis>
  <remelis>0</remelis>
  <lygiavimas />

    <fon_spalva>#FFFFFF</fon_spalva>
  <srif_spalva />
  <rysys>0</rysys>
    </elementas>
- <elementas>
  <tipas>JUMP_MENIU</tipas>

    <el_id>Jump_meniu_49</el_id>
  <aukstis>20</aukstis>
  <reiksme />
  <plotis>20</plotis>
  <x_koord>530</x_koord>
  <y_koord>200</y_koord>
  <sriftodydis>0</sriftodydis>
  <remelis>0</remelis>
  <lygiavimas />
  <fon_spalva />
  <srif_spalva />
  <rysys>0</rysys>
    </elementas>
- <elementas>
  <tipas>MYGTUKAS</tipas>
  <el_id>Mygtukas_60</el_id>
  <aukstis>20</aukstis>
  <reiksme>Patvirtinti</reiksme>
  <plotis>50</plotis>
  <x_koord>665</x_koord>
  <y_koord>440</y_koord>
  <sriftodydis>0</sriftodydis>
  <remelis>0</remelis>

    <lygiavimas>center</lygiavimas>

    <fon_spalva>#FFFFFF</fon_spalva>
  <srif_spalva />
  <rysys>7</rysys>
    </elementas>
- <elementas>
  <tipas>MYGTUKAS</tipas>
  <el_id>Mygtukas_62</el_id>
  <aukstis>20</aukstis>
  <reiksme>Atsisakyti</reiksme>
  <plotis>50</plotis>
  <x_koord>750</x_koord>
  <y_koord>440</y_koord>
  <sriftodydis>0</sriftodydis>
  <remelis>0</remelis>

    <lygiavimas>center</lygiavimas>

    <fon_spalva>#FFFFFF</fon_spalva>
  <srif_spalva />
  <rysys>0</rysys>
    </elementas>
- <elementas>
  <tipas>TEKSTO_EILUTE</tipas>

    <el_id>Teksto_eilute_64</el_id>
  <aukstis>20</aukstis>
  <reiksme />
  <plotis>20</plotis>
  <x_koord>310</x_koord>
  <y_koord>140</y_koord>
  <sriftodydis>0</sriftodydis>
  <remelis>0</remelis>
  <lygiavimas />
```

```
<fon_spalva />
<srif_spalva />
<rysys>0</rysys>
</elementas>
- <elementas>
<tipas>LENTELE</tipas>
<el_id>Zemes_skl_51</el_id>
<x_koord>35</x_koord>
<y_koord>230</y_koord>
<remelis>2</remelis>
- <eilute>
- <stulpelis>
<reiksme>Kvartalo Nr.</reiksme>
<aukstis>25</aukstis>
<plotis>130</plotis>
<x_koord>35</x_koord>
<y_koord>230</y_koord>
<sriftodydis>2</sriftodydis>
<remelis>2</remelis>
<lygiavimas>center</lygiavimas>
<fon_spalva>#FFFFFF</fon_spalva>
<srif_spalva />
<rysys>0</rysys>
</stulpelis>
- <stulpelis>
<reiksme>Taksacinio misko sklypo Nr.</reiksme>
<aukstis>25</aukstis>
<plotis>130</plotis>
<x_koord>200</x_koord>
<y_koord>230</y_koord>
<sriftodydis>2</sriftodydis>
<remelis>1</remelis>
<lygiavimas>center</lygiavimas>
<fon_spalva>#FFFFFF</fon_spalva>
<srif_spalva />
<rysys>0</rysys>
</stulpelis>
- <stulpelis>
<reiksme>Taksacinio misko sklypo plotas</reiksme>
<aukstis>25</aukstis>
<plotis>130</plotis>
<x_koord>300</x_koord>
<y_koord>230</y_koord>
<sriftodydis>2</sriftodydis>
<remelis>1</remelis>
<lygiavimas>center</lygiavimas>
<fon_spalva>#FFFFFF</fon_spalva>
<srif_spalva />
<rysys>0</rysys>
</stulpelis>
- <stulpelis>
<reiksme>Kirtimo intensyvumas procentais</reiksme>
<aukstis>25</aukstis>
<plotis>130</plotis>
<x_koord>400</x_koord>
<y_koord>230</y_koord>
<sriftodydis>2</sriftodydis>
<remelis>1</remelis>
<lygiavimas>center</lygiavimas>
<fon_spalva>#FFFFFF</fon_spalva>
<srif_spalva />
<rysys>0</rysys>
</stulpelis>
- <stulpelis>
<reiksme>Kirtimo metai</reiksme>
<aukstis>25</aukstis>
<plotis>130</plotis>
<x_koord>500</x_koord>
<y_koord>230</y_koord>
<sriftodydis>2</sriftodydis>
<remelis>1</remelis>
<lygiavimas>center</lygiavimas>
<fon_spalva>#FFFFFF</fon_spalva>
<srif_spalva />
<rysys>0</rysys>
</stulpelis>
- <stulpelis>
<reiksme>Pastabos</reiksme>
<aukstis>25</aukstis>
<plotis>130</plotis>
<x_koord>600</x_koord>
<y_koord>230</y_koord>
<sriftodydis>2</sriftodydis>
<remelis>1</remelis>
<lygiavimas>center</lygiavimas>
<fon_spalva>#FFFFFF</fon_spalva>
<srif_spalva />
<rysys>0</rysys>
</stulpelis>
</eilute>
- <eilute>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
</stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
</stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
</stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
</stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
</stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
```

```
<lygiavimas />
  </stulpelis>
  </eilute>
- <eilute>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
  </eilute>
- <eilute>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
  </eilute>
- <eilute>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
```

```
    </stulpelis>
- <stulpelis>
  <reiksme>tekstas</reiksme>
  <aukstis>25</aukstis>
  <plotis />
  <x_koord />
  <y_koord />
  <sriftodydis />
  <remelis />
  <lygiavimas />
    </stulpelis>
    </eilute>
- <eilute>
- <stulpelis>
  <reiksme>tekstas</reiksme>
  <aukstis>25</aukstis>
  <plotis />
  <x_koord />
  <y_koord />
  <sriftodydis />
  <remelis />
  <lygiavimas />
    </stulpelis>
- <stulpelis>
  <reiksme>tekstas</reiksme>
  <aukstis>25</aukstis>
```

```
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
```

```
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
- <stulpelis>
<reiksme>tekstas</reiksme>
<aukstis>25</aukstis>
<plotis />
<x_koord />
<y_koord />
<sriftodydis />
<remelis />
<lygiavimas />
  </stulpelis>
  </eilute>
  </elementas>
  </dalis>
  </forma>
  </dokumentas>
```

10.3. Duomenų šaltinio „Miško savininko deklaracija“ sugeneruoto vartotojo sąsajos modelio pagrindinio lango XML kodas

```
  <?xml version="1.0"
  encoding="ISO-8859-1" ?>
- <xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.
  org/1999/XSL/Transform"
  >
  <xsl:output method="html"
  version="1.0" encoding="UTF-
  8" indent="yes" />
- <xsl:template match="/">
- <html>
- <head>
  <title>Dokumento
  pavadinimas</title>
  <meta http-equiv="Content-
  Type" content="text/html;
  charset=utf-8" />
  </head>
- <body>
- <xsl:for-each
  select="dokumentas/forma[
  vardas='Deklarac_4']">
- <form name="{vardas}"
  method="post"
  action="xml_failas_{dalis/el
  ementas[tipas='MYGTUKAS'
  ][rysys!='0']/rysys}.xml">
```

```
<style
  type="text/css">table.Pilieti
  s_4_dalis {
  position:absolute; left:20px;
  top:10px; }</style>
- <table class="Pilietis_4_dalis"
  border="1"
  width="{dalis[vardas='Piliet
  is']/plotis}" cellpadding="0"
  cellspacing="0">
- <tr>
<td
  width="{dalis[vardas='Piliet
  is']/plotis}"
  height="{dalis[vardas='Pilie
  tis']/aukstis}" />
  </tr>
  </table>
<style
  type="text/css">table.Pilieti
  s_4_antraste {
  position:absolute; left:30px;
  top:0px; }</style>
- <table bgcolor="#FFFFFF"
  class="Pilietis_4_antraste"
  border="0" cellpadding="0"
  cellspacing="0">
```

```
- <tr>
<td>Pilietis</td>
  </tr>
  </table>
- <xsl:for-each
  select="dalis[vardas='Pilieti
  s']/elementas">
- <xsl:choose>
- <xsl:when
  test="tipas='MYGTUKAS'">
<input class="{el_id}"
  type="submit"
  name="{el_id}"
  value="{reiksme}" />
- <style type="text/css">
  input.
<xsl:value-of select="el_id" />
  { position:absolute; left:
<xsl:value-of select="x_koord"
  />
  px; top:
<xsl:value-of select="y_koord"
  />
  px; }
  </style>
  </xsl:when>
```

```
- <xsl:when
    test="tipas='LENTELE'">
- <TABLE BORDER="{remelis}"
    NAME="{el_id}"
    class="{el_id}">
- <xsl:for-each select="eilute">
- <tr>
- <xsl:for-each select="stulpelis">
- <td WIDTH="{plotis}"
    HEIGHT="{aukstis}"
    valign="top"
    ALIGN="{lygiavimas}">
- <font size="{sriftodydis}">
  <xsl:value-of select="reiksme"
    />
    </font>
    </td>
    </xsl:for-each>
    </tr>
    </xsl:for-each>
    </TABLE>
- <style type="text/css">
    table.
  <xsl:value-of select="el_id" />
    { position:absolute; left:
  <xsl:value-of select="x_koord"
    />
    px; top:
  <xsl:value-of select="y_koord"
    />
    px; }
    </style>
    </xsl:when>
- <xsl:when
    test="tipas='OPCIJA'">
- <div class="{el_id}">
- <input name="{el_id}"
    type="radio"
    value="{reiksme}">
  <xsl:value-of select="reiksme"
    />
    </input>
    </div>
- <style type="text/css">
    div.
  <xsl:value-of select="el_id" />
    { position:absolute; left:
  <xsl:value-of select="x_koord"
    />
    px; top:
  <xsl:value-of select="y_koord"
    />
    px; }
    </style>
    </xsl:when>
- <xsl:when
    test="tipas='TEKSTO_EILUTE'">
  <INPUT CLASS="{el_id}"
    NAME="{el_id}" TYPE="text"
    SIZE="{plotis}"
    VALUE="{reiksme}" />
- <style type="text/css">
    input.
  <xsl:value-of select="el_id" />
    { position:absolute; left:
  <xsl:value-of select="x_koord"
    />
    px; top:
  <xsl:value-of select="y_koord"
    />
    px; }
    </style>
    </xsl:when>
- <xsl:when test="tipas='CELE'">
- <TABLE CLASS="{el_id}"
    WIDTH="{plotis}"
    BORDER="{remelis}"
    cellpadding="0"
    cellspacing="0"
    bgcolor="{fon_spalva}"
    bordercolor="#000000">
- <tr>
- <td WIDTH="{plotis}"
    HEIGHT="{aukstis}"
    valign="top"
    ALIGN="{lygiavimas}"
    SIZE="{sriftodydis}">
- <font color="{srif_spalva}">
  <xsl:value-of select="reiksme"
    />
    </font>
    </td>
    </tr>
    </TABLE>
- <style type="text/css">
    table.
  <xsl:value-of select="el_id" />
    { position:absolute; left:
  <xsl:value-of select="x_koord"
    />
    px; top:
  <xsl:value-of select="y_koord"
    />
    px; }
    </style>
    </xsl:when>
- <xsl:when
    test="tipas='PAVEIKSLAS'">
- <a href="x.jpg">
  <img CLASS="{el_id}"
    src="{reiksme}"
    ALT="{reiksme}"
    NAME="{el_id}"
    WIDTH="{plotis}"
    HEIGHT="{aukstis}"
    BORDER="{remelis}" />
    </a>
- <style type="text/css">
    img.
  <xsl:value-of select="el_id" />
    { position:absolute; left:
  <xsl:value-of select="x_koord"
    />
    px; top:
  <xsl:value-of select="y_koord"
    />
    px; }
    </style>
    </xsl:when>
- <xsl:when
    test="tipas='VARNELE'">
- <div class="{el_id}">
- <input type="checkbox"
    name="{el_id}"
    width="{plotis}">
- <font size="{sriftodydis}">
  <xsl:value-of select="reiksme"
    />
    </font>
    </input>
    </div>
- <style type="text/css">
    div.
  <xsl:value-of select="el_id" />
    { position:absolute; left:
  <xsl:value-of select="x_koord"
    />
    px; top:
  <xsl:value-of select="y_koord"
    />
    px; }
    </style>
    </xsl:when>
- <xsl:when
    test="tipas='JUMP_MENIU'"
    >
- <select name="{el_id}"
    class="{el_id}">

    <option>_____</option>
    </select>
- <style type="text/css">
    select.
  <xsl:value-of select="el_id" />
    { position:absolute; left:
  <xsl:value-of select="x_koord"
    />
    px; top:
  <xsl:value-of select="y_koord"
    />
    px; }
    </style>
    </xsl:when>
    </xsl:choose>
    </xsl:for-each>
    </form>
    </xsl:for-each>
```

```
- <xsl:for-each
    select="dokumentas/forma[
    vardas='Deklarac_4']">
- <form name="{vardas}"
    method="post"
    action="xml_failas_{dalis/el
    ementas[tipas='MYGTUKAS'
    ][rysys!='0']/rysys}.xml">
  <style
    type="text/css">table.Zeme
    s_sklypas_5_dalis {
    position:absolute; left:20px;
    top:125px; }</style>
- <table
    class="Zemes_sklypas_5_da
    lis" border="1"
    width="{dalis[vardas='Zeme
    s_sklypas']/plotis}"
    cellpadding="0"
    cellspacing="0">
- <tr>
  <td
    width="{dalis[vardas='Zeme
    s_sklypas']/plotis}"
    height="{dalis[vardas='Zem
    es_sklypas']/aukstis}" />
    </tr>
    </table>
  <style
    type="text/css">table.Zeme
    s_sklypas_5_antraste {
    position:absolute; left:30px;
    top:115px; }</style>
- <table bgcolor="#FFFFFF"
    class="Zemes_sklypas_5_an
    traste" border="0"
    cellpadding="0"
    cellspacing="0">
- <tr>
  <td>Zemes_sklypas</td>
    </tr>
    </table>
- <xsl:for-each
    select="dalis[vardas='Zeme
    s_sklypas']/elementas">
- <xsl:choose>
- <xsl:when
    test="tipas='MYGTUKAS'">
  <input class="{el_id}"
    type="submit"
    name="{el_id}"
    value="{reiksme}" />
- <style type="text/css">
    input.
  <xsl:value-of select="el_id" />
    { position:absolute; left:
  <xsl:value-of select="x_koord"
    />
    px; top:
  <xsl:value-of select="y_koord"
    />
    px; }
    </style>
    </xsl:when>
- <xsl:when
    test="tipas='LENTELE'">
- <TABLE BORDER="{remelis}"
    NAME="{el_id}"
    class="{el_id}">
- <xsl:for-each select="eilute">
- <tr>
- <xsl:for-each select="stulpelis">
- <td WIDTH="{plotis}"
    HEIGHT="{aukstis}"
    valign="top"
    ALIGN="{lygiavimas}">
- <font size="{sriftodydis}">
  <xsl:value-of select="reiksme"
    />
    </font>
    </td>
    </xsl:for-each>
    </tr>
    </xsl:for-each>
    </TABLE>
- <style type="text/css">
    table.
  <xsl:value-of select="el_id" />
    { position:absolute; left:
  <xsl:value-of select="x_koord"
    />
    px; top:
  <xsl:value-of select="y_koord"
    />
    px; }
    </style>
    </xsl:when>
- <xsl:when
    test="tipas='OPCIJA'">
- <div class="{el_id}">
- <input name="{el_id}"
    type="radio"
    value="{reiksme}">
  <xsl:value-of select="reiksme"
    />
    </input>
    </div>
- <style type="text/css">
    div.
  <xsl:value-of select="el_id" />
    { position:absolute; left:
  <xsl:value-of select="x_koord"
    />
    px; top:
  <xsl:value-of select="y_koord"
    />
    px; }
    </style>
    </xsl:when>
- <xsl:when
    test="tipas='TEKSTO_EILUT
    E'">
  <INPUT CLASS="{el_id}"
    NAME="{el_id}" TYPE="text"
    SIZE="{plotis}"
    VALUE="{reiksme}" />
- <style type="text/css">
    input.
  <xsl:value-of select="el_id" />
    { position:absolute; left:
  <xsl:value-of select="x_koord"
    />
    px; top:
  <xsl:value-of select="y_koord"
    />
    px; }
    </style>
    </xsl:when>
- <xsl:when test="tipas='CELE'">
- <TABLE CLASS="{el_id}"
    WIDTH="{plotis}"
    BORDER="{remelis}"
    cellpadding="0"
    cellspacing="0"
    bgcolor="{fon_spalva}"
    bordercolor="#000000">
- <tr>
- <td WIDTH="{plotis}"
    HEIGHT="{aukstis}"
    valign="top"
    ALIGN="{lygiavimas}"
    SIZE="{sriftodydis}">
- <font color="{srif_spalva}">
  <xsl:value-of select="reiksme"
    />
    </font>
    </td>
    </tr>
    </TABLE>
- <style type="text/css">
    table.
  <xsl:value-of select="el_id" />
    { position:absolute; left:
  <xsl:value-of select="x_koord"
    />
    px; top:
  <xsl:value-of select="y_koord"
    />
    px; }
    </style>
    </xsl:when>
- <xsl:when
    test="tipas='PAVEIKSLAS'">
- <a href="x.jpg">
  <img CLASS="{el_id}"
    src="{reiksme}"
    ALT="{reiksme}"
    NAME="{el_id}"
    WIDTH="{plotis}"
    HEIGHT="{aukstis}"
    BORDER="{remelis}" />
    </a>
- <style type="text/css">
```

```
    img.
  <xsl:value-of select="el_id" />
    { position:absolute; left:
  <xsl:value-of select="x_koord"
    />
    px; top:
  <xsl:value-of select="y_koord"
    />
    px; }
    </style>
    </xsl:when>
- <xsl:when
    test="tipas='VARNELE'">
- <div class="{el_id}">
- <input type="checkbox"
    name="{el_id}"
    width="{plotis}">
- <font size="{sriftodydis}">
  <xsl:value-of select="reiksme"
    />
    </font>
    </input>
```

```
    </div>
- <style type="text/css">
    div.
<xsl:value-of select="el_id" />
    { position:absolute; left:
<xsl:value-of select="x_koord"
    />
    px; top:
<xsl:value-of select="y_koord"
    />
    px; }
    </style>
    </xsl:when>
- <xsl:when
    test="tipas='JUMP_MENIU'"
    >
- <select name="{el_id}"
    class="{el_id}">

    <option>_____</op
    tion>
    </select>
```

```
- <style type="text/css">
    select.
<xsl:value-of select="el_id" />
    { position:absolute; left:
<xsl:value-of select="x_koord"
    />
    px; top:
<xsl:value-of select="y_koord"
    />
    px; }
    </style>
    </xsl:when>
    </xsl:choose>
    </xsl:for-each>
    </form>
    </xsl:for-each>
    </body>
    </html>
    </xsl:template>
    </xsl:stylesheet>
```

10.4. Vartotojo dokumentacija

10.4.1. Sistemos funkcinis aprašymas

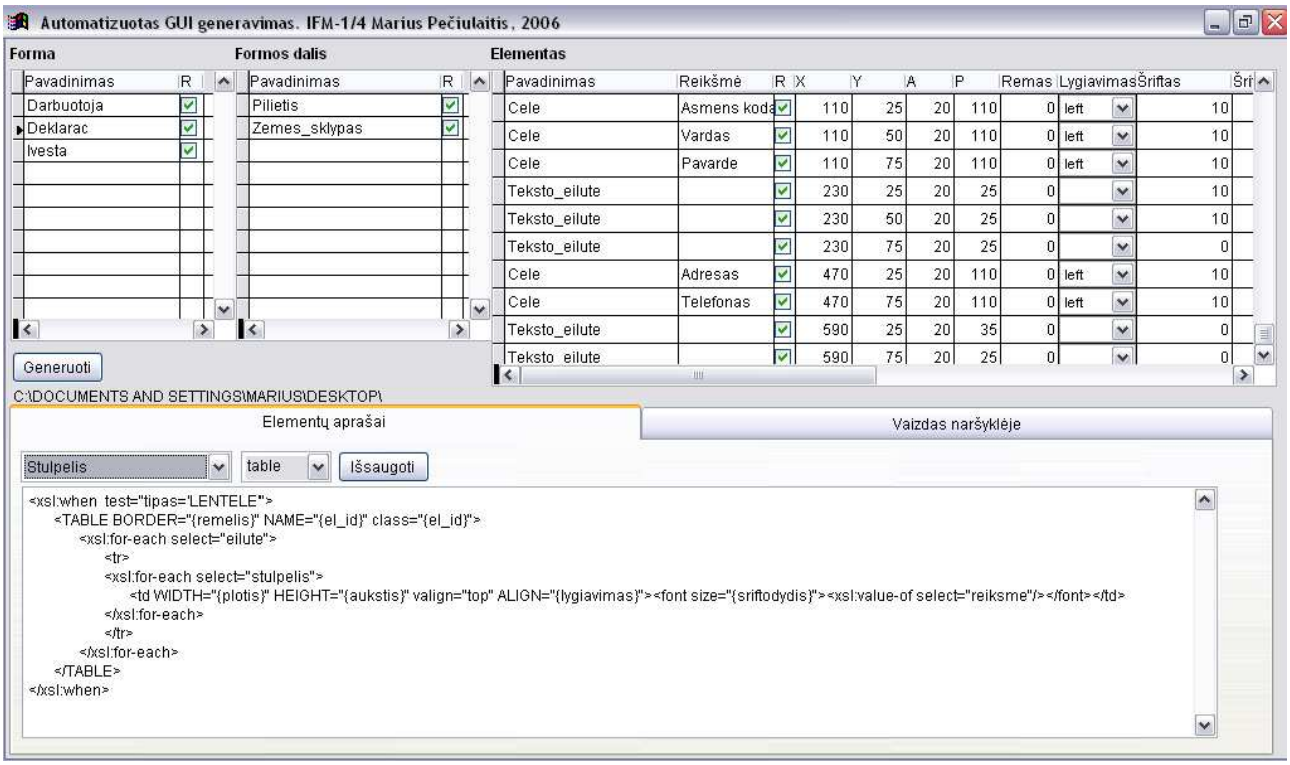
Šio įrankio pagalba automatizuotai generuojama grafinė WEB sistemos vartotojo sąsaja. Įrankis naudojasi duomenų bazėje saugoma GUI specifikacija ir suformuoja dvi rinkmenas:

- XML – vartotojo sąsajos specifikacija XML formatu
- XSLT – XML kodo transformacijų į XHTML šablonas.

GUI generavimo procesą vartotojas gali valdyti keisdamas programoje naudojamų XSLT taisyklių aprašus, taip pat koreguoti duomenų bazėje saugomą informaciją. Visoms operacijoms atlikti bei peržiūrėti suformuotą GUI ar XSLT rinkmenas nereikia jokios kitos papildomos programinės įrangos, tam pakanka šio įrankio.

10.4.2. Sistemos vadovas

Sistemos langas susideda iš dviejų pagrindinių dalių. Viršutinėje pateikiama vartotojo sąsajos specifikacijos aprašas iš duomenų saugyklos. Šie duomenys bus naudojami generuojant XML rinkmenas. Apatinėje programos lango dalyje koregavimui pateikiamos visos sistemoje naudojamos XSL transformacijos (kortelė „Elementų aprašai“). Čiapat galima ir peržiūrėti kaip atrodo galutinai suformuota GUI (kortelė „Vaizdas naršyklėje“)



41 pav. Pagrindinis programos langas

9.4.2.1 GUI specifikacijos duomenų keitimas

Norint pakoreguoti tam tikros formos (duomenų šaltinio) GUI specifikaciją tai galima atlikti trijuose lygmenyse:

- GUI generuojama/negeruojama visai formai – atitinkamai uždedama arba nuimama varnelė stulpelyje R, ties pasirinkta forma. Nuėmus varnelę, šios formos aprašai generavimo metu ignoruojami ir į XML bei XSLT rinkmenas neįtraukiami.

- GUI generuojama/negeneruojama formos daliai (epizodui) - atitinkamai uždedama arba nuimama varnelė stulpelyje R, ties pasirinkta formos dalimi. Nuėmus varnelę, šio formos epizodo aprašai generavimo metu ignoruojami ir į XML bei XSLT rinkmenas neįtraukiami.

- Keičiamos kiekvieno grafikos elemento charakteristikos:

  Reikšmė – gali būti mygtuko pavadinimas, antraštės pavadinimas, teksto eilutė, lentelės celės tekstas ir pan.

  R - rodomumas. Uždėjus arba nuėmus varnelę nurodoma ar įtraukiamas į GUI.

  X – grafikos elemento padėtis X ašyje pikseliais. Jei formuojama lentelė, tuomet aprašoma tik pirmo stulpelio X koordinatė.

  Y – grafikos elemento padėtis Y ašyje pikseliais. Formuojant lentelę aprašoma tik pirmo stulpelio koordinatė.

  A – elemento aukštis. Įprastai tai elemento aukštis pikseliais. Tačiau teksto eilutėms pagal nutylėjimą automatiškai nustatomas 20 px, todėl joms nebūtina nurodyti aukščio. Formuojant lentelių stulpelius, A atributas žymį stulpelio eilučių skaičių.

  P – elemento plotis pikseliais, o jei tai teksto eilutė – jos plotis raidžių skaičiumi.

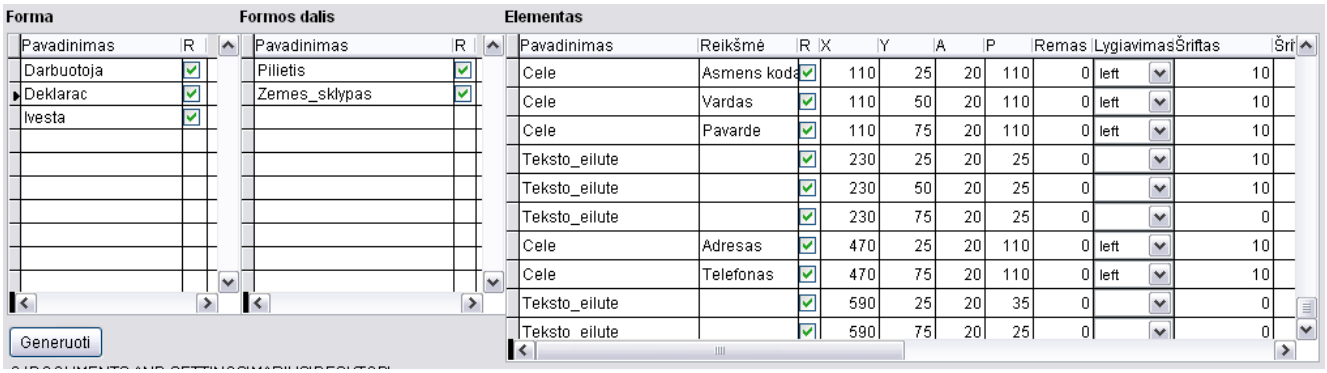
  Rėmas – elemento rėmelio plotis pikseliais. Rėmelius gali turėti celės, stulpeliai bei paveikslai.

  Lygiavimas – nurodoma kaip tekstas elemente lygiuojamas horizontaliai. Galimos reikšmės: left (kairė), center (centre), right (dešinėje).

  Šriftas – teksto dydis pikseliais.

  Šrifto spalva – teksto spalvos kodas HTML formatu. Pvz. šviesiai pilka #CCCCCC.

  Fono spalva – grafikos elemento fono spalva HTML formatu.

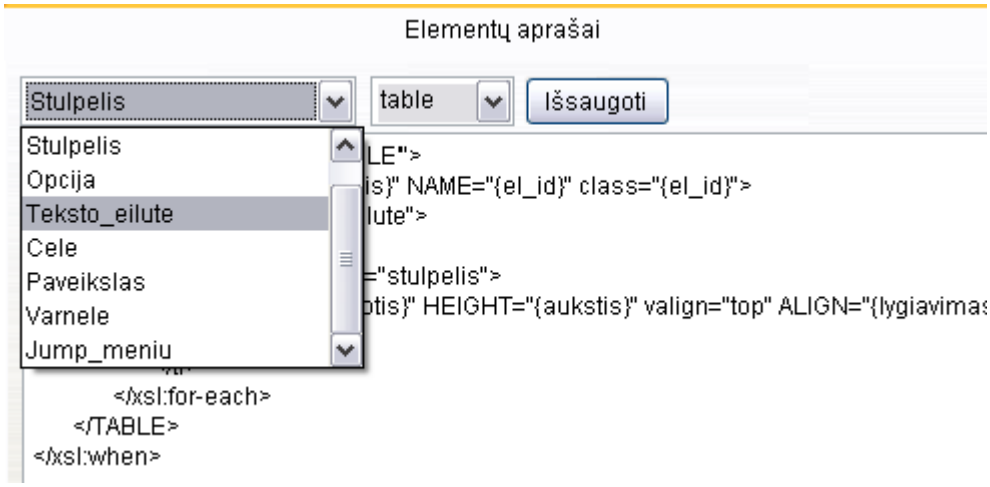


42 pav. Programos lango viršutinė dalis – GUI specifikacijos koregavimas

Formos dalys programoje rodomos tik pasirinktinos formos, o elementai tik pasirinktos formos dalies, todėl norint koreguoti GUI specifikaciją vertėtu pradėti rinktis iš kairės į dešinę: forma, formos dalis, elementas.

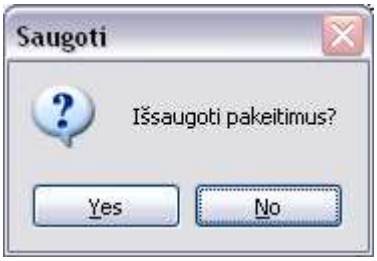
9.4.2.2 XSLT taisyklių koregavimas

Apatinėje programos dalyje, kortelėje „Elementų aprašai“ pirmajame iššokančiame meniu surašyti visų sistemoje naudojamų grafikos elementų pavadinimai (žr. **43 pav.**). Pasirinkus vieną iš jų žemiau pateikiama jo formavimo taisyklė XSL kalba, o šalia pavadinimo rodoma, kokia žymė jam aprašyti naudojama XHTML kode. Nuo to kaip elementas bus atvaizduojamas grafinėje aplinkoje didžiąja dalimi priklauso XSLT taisyklės. Ja nurodoma kaip ir kokios XML žymės bei jų reikšmės turi būti transformuojamos į XHTM kodą.



43 pav. XSLT taisyklių koregavimas

Pakoreguota taisyklė išsaugoma mygtuku „Išsaugoti“ Jį paspaudus iššoka pranešimas, prašantis patvirtinimo (žr. **44 pav.**).

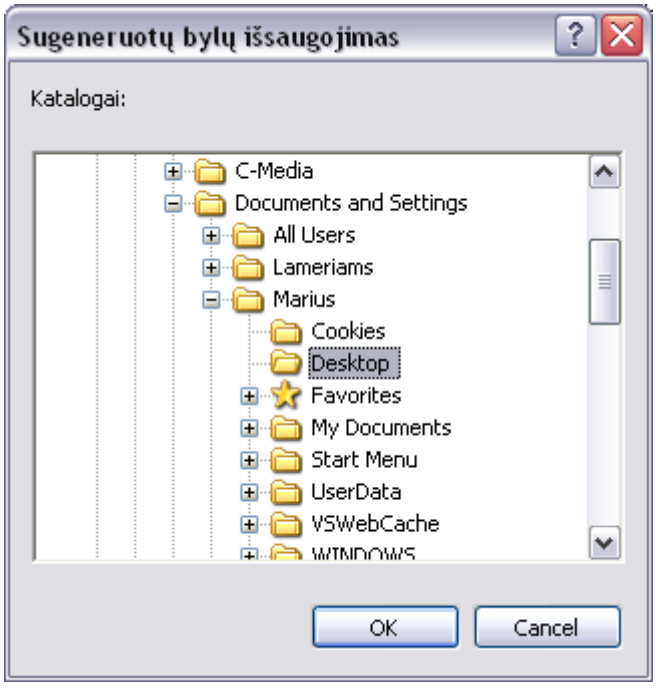


44 pav. XSLT taisyklės išsaugojimo patvirtinimas

9.4.2.3 Generavimas

Atlikus reikiamas GUI specifikacijos ar XSLT taisyklių korekcijas privaloma nurodyti, kur norima išsaugoti sugeneruotas rinkmenas. Tai atliekama du kartus spragtelėjus pele ant po mygtuku „Generuoti“ matomo programos darbinio kelio (**45 pav.**ir išsirinkus norimą katalogą (**46 pav.**).

45 pav. Kelias iki bylos, kur bus išsaugoma sugeneruota GUI

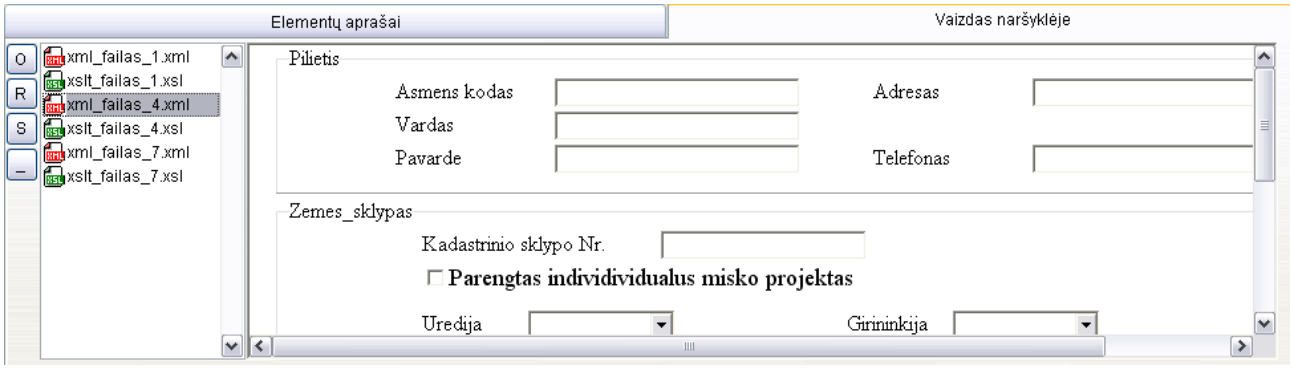


46 pav. Sugeneruotos GUI išsaugojimo vietos pasirinkimas

GUI generavimas vykdomas spragtelėjus mygtuką „Generuoti“. Tol kol vykdoma operacija, šalia jo rodomas raudonas užrašas „Palaukite...“ (**45 pav.**). Operacijai pasibaigus užrašo nebelieka.

9.4.2.4 Sugeneruotų rinkmenų peržiūra

Visos sugeneruotas rinkmenas galima peržiūrėti atidarius kortelę „Vaizdas naršyklėje“ ir pasirinkus vieną iš sąrašo kairėje lango pusėje (**47 pav.**). Jos išrikiuojamos didėjimo tvarka pagal formos identifikacinį numerį duomenų bazėje, kuris matomas ir rinkmenos pavadinime. Kiekviena atskira XML ir XSLT pora – vienas tinklapio puslapis. Pasirinkus XML rinkmeną dešiniau parodomas tinklapio grafinės sąsajos vaizdas.



47 pav. Sugeneruotų rinkmenų peržiūra.

Jei sąraše pažymima XSLT rinkmena, tuomet ekrane pateikiamas jos turinys.

48 pav. XSLT peržiūra

Patogesniai XSLT kodo peržiūrai spragsint ant dešinėje pusėje išdėstytų “-“ ir “+” ženklų atitinkamai sutraukiami arba išskleidžiami elementų aprašai.

Kairėje kortelės pusėje (**47 pav.**) išdėstyti 4 mygtukai:

- **O** – peržiūrai pasirenkama XML ar XSLT rinkmena iš bet kurios disko vietos.
- **R** – peržiūros lango atnaujinimas.

- **S** – rinkmenų sąrašo paslėpimas/parodymas.
- **_** - apatinės programos dalies paaukštinimas/pažeminimas.

10.4.3. Sistemos instaliavimo dokumentas

Įrankis pilnai suderinamas su visomis Windows operacinėmis sistemomis pradedant nuo Windows 98. Sistemos prototipui pakanka 32MB darbinės atminties ir 15 MB vietos diske.

Paleidžiamoji rinkmena: `xmlgener.exe`

Kitos reikalingos darbui rinkmenos:

gdiplus.dll	VFP9r.dll	VFP9t.dll
VFP9ENU.DLL	VFP9renu.dll	

Visual FoxPro duomenų bazė saugoma kataloge „DB foxine“, kuris talpinamas kartu su paleidžiamąja sistemos rinkmena. Duomenų bazės sudėtinės bylos:

duom_salt.BAK	element_tipas.FPT	f_elementas.BAK
duom_salt.CDX	element_tipas.TBK	f_elementas.CDX
duom_salt.dbf	forma.CDX	f_elementas.dbf
element_grupe.BAK	forma.dbf	metabaze.dbc
element_grupe.CDX	f_dalies_tipas.CDX	metabaze.DCT
element_grupe.dbf	f_dalies_tipas.dbf	metabaze.DCX
element_tipas.BAK	f_dalis.BAK	forma.BAK
element_tipas.CDX	f_dalis.CDX	
element_tipas.dbf	f_dalis.dbf	

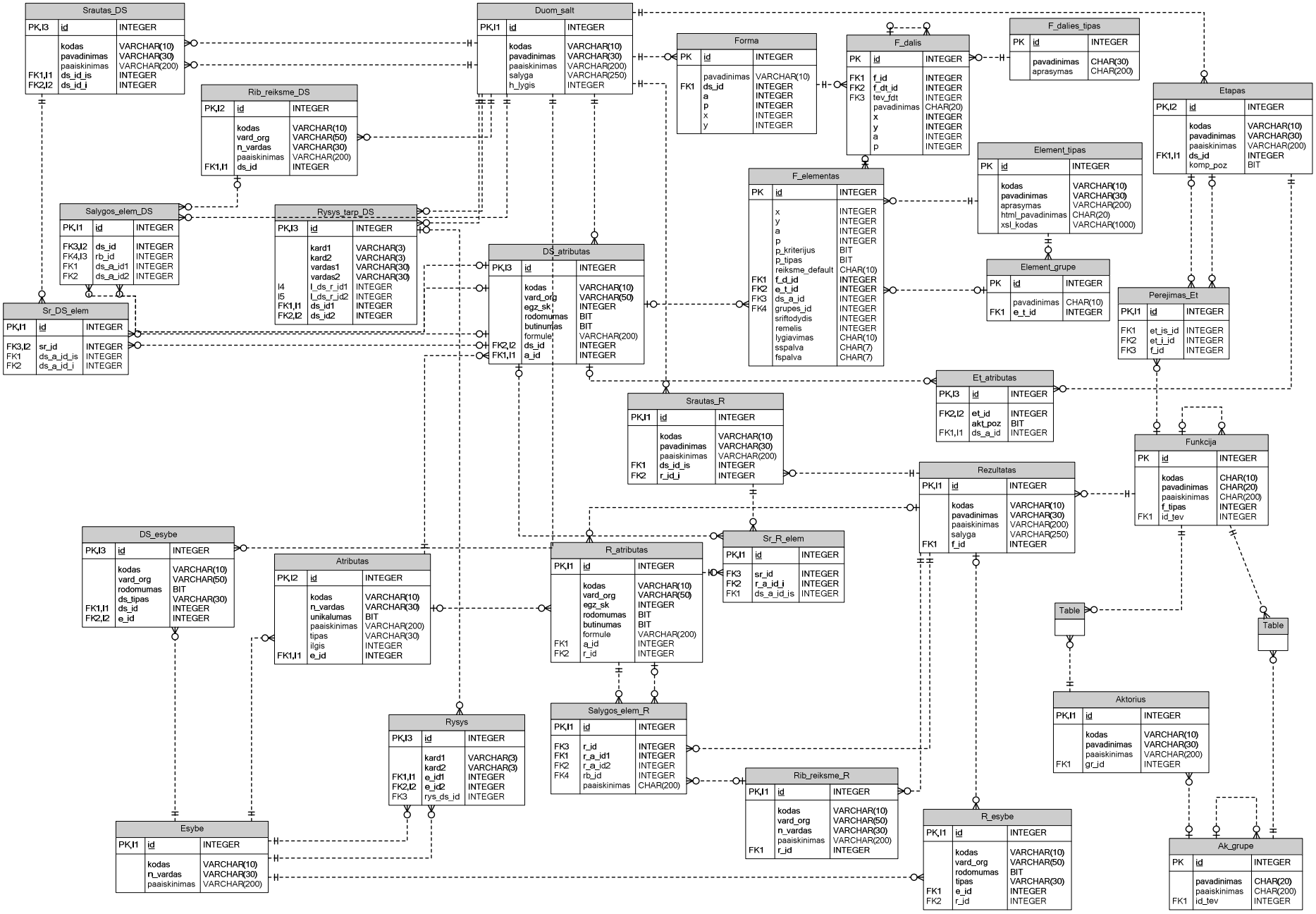
Norint projektą perprogramuoti, reikalingos šios papildomos rinkmenos:

XMLgener.PJT	start.prg	generavimo.prg
XMLgener.pjx	start.BAK	generavimo.BAK
main.SCT	start.FXP	
main.scx	generavimo.FXP	

10.4.4. Sistemos administravimo vadovas

Įrankio prototipas tiesiogiai jungiasi prie Visual FoxPro duomenų bazės, todėl jokių papildomų konfigūravimų nereikia. Ateityje, kai duomenų bazė bus Microsoft SQL, prie jos bus jungiamasi per ODBC.

10.5. ODRES projekto metabazės schema



49 pav. ODRES projekto metabazės schema [17]

Informacijos srautų specifikacijos
metaduomenų bazės objektų aprašymas

Duomenų tipų paaiškinimai:

- **int** – sveikas skaičius;
- **varchar <ilgis>** - nustatyto ilgio simbolių eilutė, kur <ilgis> - eilutės ilgis;
- **bool** – loginis kintamasis įgyjantis reikšmes *true* arba *false*.

Vardas	Duomenų tipas	Pirminis raktas	Būtinas	Unikalus	Paaiškinimas
Atributas					**Lentelėje saugomi visi nagrinėjamo organizacijos veiklos konteksto atributai.**
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
kodas	varchar 10		+	+	Unikalus atributą apibudinantis kodas, kodą sudaro sistemos vartotojas (analitikas/projektuotojas).
n_vardas	varchar 50		+		Atributo vardas, kuris gali būti paimtas iš nagrinėjamo veiklos konteksto arba sukurtas paties vartotojo.
unikalumas	bool		+		Požymis rodantis ar atributo reikšmė turi būti unikali. TRUE – reikšmė turi būti unikali, o FALSE - kad reikšmės unikalumas nėra būtinas.
paaiskinimas	varchar 200				Detali informacija apie atributą.
tipas	varchar 30				Atributo duomenų tipas.
e_id	int				Esybės, kuriai priklauso atributas id.
Esybe					**Lentelėje saugomos visos nagrinėjamo organizacijos veiklos konteksto esybės.**
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
kodas	varchar 10		+	+	Unikalus esybę apibudinantis kodas, kodą sudaro sistemos vartotojas.
n_vardas	varchar 30		+		Esybės vardas, kuris gali būti paimtas iš nagrinėjamo veiklos konteksto arba sukurtas paties vartotojo.
paaiskinimas	varchar 200				Detalizuota informacija apie esybę.
Rysys					**Lentelėje saugomi visi nagrinėjamo organizacijos veiklos konteksto ryšiai tarp esybių.**
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
kard1	varchar 3		+		Ryšio kardinalumas viename ryšio gale. Kardinalumo aprašymo struktūra: <būtinumas>,<kardinalumas>.

Vardas	Duomenų tipas	Pirminis raktas	Būtinas	Unikalus	Paaiškinimas
					Būtinumas gali įgyti šias reikšmes: 0-nebūtinas, 1- būtinas. Kardinalumas gali įgyti šias reikšmes: 1 – vienas, x – daug.
kard2	varchar 3		+		Ryšio kardinalumas antrame ryšio gale. Kardinalumo aprašymo struktūra: <būtinumas>,<kardinalumas>. Būtinumas gali įgyti šias reikšmes: 0-nebūtinas, 1- būtinas. Kardinalumas gali įgyti šias reikšmes: 0 – nulis, 1 – vienas, x – daug.
e_id1	int		+		Esybių, kurias jungia ryšys id.
e_id2	int		+		
rys_ds_id	int				Motyvuojančio ryšio tarp duomenų šaltinių id.
Rezultatas					**Lentelėje saugomi informacija apie KIS funkcionalumo metu formuojamus rezultatus arba žodžiu perduodamus informacijos srautus.**
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
kodas	varchar 10		+	+	Unikalus įrašą apibudinantis kodas, kodą sudaro sistemos vartotojas.
pavadinimas	varchar 30		+		Rezultato pavadinimas, kuris gali būti paimtas iš nagrinėjamo veiklos konteksto arba sukurtas paties vartotojo.
paaiskinimas	varchar 200				Detalizuota informacija apie rezultatą.
salyga	varchar 500				Užrašoma žodinė informacija apie sąlygas, kurias reikia tenkinti formuojant tam tikrą rezultatą.
R_atributas					Lentelėje saugomi informacija apie KIS funkcionalumo metu formuojamų rezultatų atributus.
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
kodas	varchar 10		+	+	Unikalus įrašą apibudinantis kodas, kodą sudaro sistemos vartotojas.
vard_org	varchar 50		+		Rezultato atributo pavadinimas, kuris paimtas iš nagrinėjamo veiklos konteksto, toks koks pateikiamas originaliame rezultate.
egz_sk	int		+		Konkretaus atributo egzempliorių skaičius išvedamų į rezultatą.
rodomumas	bool		+		Ar atributas matomas tik specifikacijos modelyje ar ir originale. FALSE – originale ir modelyje, TRUE - tik modelyje.
butinumas	bool		+		Požymis rodantis ar atributo reikšmės būtinai turi būti pateikiama rezultate. TRUE –

Vardas	Duomenų tipas	Pirminis raktas	Būtinas	Unikalus	Paaiškinimas
					reikšmė turi būti pateikiama, o FALSE - kad reikšmė nebūtina.
formule	varchar 200				Jei atributo reikšmė yra skaičiuojama, išsaugoma formulė, pagal kuria gali būti skaičiuojam atributo reikšmė.
r_id	int		+		Rezultato id, kuriam priklauso atributas.
a_id	int		+		Sistemos atributo id, su kuriuo siejamas rezultato atributas, nes skirtinguose rezultatuose ar duomenų šaltiniuose tas pats sistemos atributas gali būti skirtingai įvardintas.
Rib_reiksme_R					Lentelėje saugoma rezultate esančių atributų egzempliorių ribinių reikšmių informacija.
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
kodas	varchar 10		+	+	Unikalus įrašą apibudinantis kodas, kodą sudaro sistemos vartotojas.
vard_org	varchar 50		+		Reikšmė, kuris paimtas iš nagrinėjamo veiklos konteksto.
n_vardas	varchar 30		+		Reikšmė, kuria KIS siūlo naudoti vartotojas, tai gali būti ir ta pati reikšmė paimta iš nagrinėjamo veiklos konteksto
paaiskinimas	varchar 200				Detalizuota informacija apie reikšmę.
r_id	int		+		Rezultato id, kuriame naudojama ši reikšmė.
R_esybe					Lentelėje saugomi informacija apie KIS funkcionalumo metu formuojamų rezultatų išskirtas esybes.
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
kodas	varchar 10		+	+	Unikalus įrašą apibudinantis kodas, kodą sudaro sistemos vartotojas.
vard_org	varchar 50		+		Esybės pavadinimas, kuris paimtas iš nagrinėjamo veiklos konteksto.
rodomumas	bool		+		Ar esybė matoma tik specifikacijos modelyje ar ir originale. FALSE – originale ir modelyje, TRUE - tik modelyje.
tipas	varchar 30		+		
r_id	int		+		Rezultato id, kuriam priklauso esybė.
e_id	int		+		Sistemos esybės id, su kuria siejama rezultato esybė, nes skirtinguose rezultatuose ar duomenų šaltiniuose ta pati sistemos esybė gali būti skirtingai įvardinta.
Salygos_elem_R					Lentelėje saugoma informacija apie rezultato sudarymo sąlygos elementus.

Vardas	Duomenų tipas	Pirminis raktas	Būtinas	Unikalus	Paaiškinimas
id	int	+	+	+	Unikalus įrašo id, kurį sugeneruoja DBVS.
r_id	int		+		Rezultato, kuriam formuojam sąlyga id.
r_a_id1	int		+		Rezultato atributo id, kuris įtakojamas sąlygos.
r_a_id2	int				Rezultato atributo id, kuris įtakoja rezultato atributą.
rb_id	int				Rezultato ribinės reikšmės id, kuri įtakoja rezultato atributą.
paaiskinimas	varchar 200				Detalizuotas sąlygos aprašas.
Duom_salt					Lentelėje saugomi informacija apie organizacijos objektus (duomenų šaltinius) saugančius duomenis, reikalingus funkcijoms įvykdyti.
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
kodas	varchar 10		+	+	Unikalus įrašą apibudinantis kodas, kodą sudaro sistemos vartotojas.
pavadinimas	varchar 30		+		Duomenų šaltinio pavadinimas, kuris gali būti paimtas iš nagrinėjamo veiklos konteksto arba sukurtas paties vartotojo.
paaiskinimas	varchar 200				Detalizuota informacija apie duomenų šaltinį.
salyga	varchar 500				Užrašoma žodinė informacija apie sąlygas ir logiką, kurias reikia tenkinti formuojant duomenų šaltinį.
h_lygis	int				Duomenų šaltinio hierarchijos lygis.
DS_atributas					Lentelėje saugomi informacija apie organizacijos objektų (duomenų šaltinius) saugančių duomenis, reikalingus funkcijoms įvykdyti, atributus.
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
kodas	varchar 10		+	+	Unikalus įrašą apibudinantis kodas, kodą sudaro sistemos vartotojas.
vard_org	varchar 50		+		Duomenų šaltinio atributo pavadinimas, kuris paimtas iš nagrinėjamo veiklos konteksto.
egz_sk	int		+		Konkretaus atributo egzempliorių skaičius įvedamų į konkretų duomenų šaltinį.
rodomumas	bool		+		Ar atributas matomas tik specifikacijos modelyje ar ir originale. FALSE – originale ir modelyje, TRUE - tik modelyje.
butinumas	bool		+		Požymis rodantis ar atributo reikšmės būtinai turi būti įvedama į duomenų šaltinį. TRUE – reikšmė turi būti pateikiama, o FALSE - kad reikšmė nebūtina.
formule	varchar 200				Jei atributo reikšmė yra skaičiuojama, išsaugoma formulė,

Vardas	Duomenų tipas	Pirminis raktas	Būtinas	Unikalus	Paaiškinimas
					pagal kuria gali būti skaičiuojam atributo reikšmė.
ds_id	int		+		Duomenų šaltinio id, kuriam priklauso atributas.
a_id	int		+		Sistemos atributo id, su kuriuo siejamas duomenų šaltinio atributas, nes skirtinguose rezultatuose ar duomenų šaltiniuose tas pats sistemos atributas gali būti skirtingai įvardintas.
x	int				Vartotojo sąsajos elemento dešinio viršutinio kampo x koordinatė (pikseliais)
y	int				Vartotojo sąsajos elemento dešinio viršutinio kampo y koordinatė (pikseliais)
a	int				Vartotojo sąsajos elemento aukštis (pikseliais)
p	int				Vartotojo sąsajos elemento plotis (pikseliais)
p_kriterijus	bool				Požymis nurodantis ar atributas gali būti naudojamas, kaip paieškos kriterijus duomenų šaltinyje. TRUE - gali būti paieškos kriterijus, FALSE – negali būti paieškos kriterijus .
p_tipas	bool				Požymis nurodantis kokio tipo paieška bus atliekama, pvz., ar nurodoma tik viena atributo reikšmė, ar reikšmių intervalas
elem_id	int				Vartotojo sąsajos elemento tipo ID.
Rib_reiksme_DS					Lentelėje saugoma duomenų šaltinyje esančių atributų egzempliorių ribinių reikšmių informacija.
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
kodas	varchar 10		+	+	Unikalus įrašą apibudinantis kodas, kodą sudaro sistemos vartotojas.
vard_org	varchar 50		+		Reikšmė, kuris paimtas iš nagrinėjamo veiklos konteksto.
n_vardas	varchar 30		+		Reikšmė, kuria KIS siūlo naudoti vartotojas, tai gali būti ir ta pati reikšmė paimta iš nagrinėjamo veiklos konteksto
paaiskinimas	varchar 200				Detalizuota informacija apie reikšmę.
ds_id	id		+		duomenų šaltinio id, kuriame naudojama ši reikšmė.
DS_esybe					Lentelėje saugomi informacija apie esybes išskirtas iš organizacijos objektų saugančius duomenis, reikalingus funkcijoms įvykdyti.
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.

Vardas	Duomenų tipas	Pirminis raktas	Būtinas	Unikalus	Paaiškinimas
kodas	varchar 10		+	+	Unikalus įrašą apibudinantis kodas, kodą sudaro sistemos vartotojas.
vard_org	varchar 50		+		Esybės pavadinimas, kuris paimtas iš nagrinėjamo veiklos konteksto.
rodomumas	bool		+		Ar esybė matoma tik specifikacijos modelyje ar ir originale. FALSE – originale ir modelyje, TRUE - tik modelyje.
tipas	varchar 30		+		
ds_id	int		+		Duomenų šaltinio id, kuriam priklauso esybė.
e_id	int		+		Sistemos esybės id, su kuria siejama duomenų šaltinio esybė, nes skirtinguose rezultatuose ar duomenų šaltiniuose ta pati sistemos esybė gali būti skirtingai įvardinta.
Salygos_elem_DS					Lentelėje saugoma informacija apie duomenų šaltinio sudarymo sąlygos elementus.
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
ds_id	int		+		Duomenų šaltinio, kuriam formuojam sąlyga id.
ds_a_id1	int		+		Duomenų šaltinio atributo id, kuris įtakojamas sąlygos.
ds_a_id2	int				Duomenų šaltinio atributo id, kuris įtakoja duomenų šaltinio atributą.
rb_id	int				Duomenų šaltinio ribinės reikšmės id, kuri įtakoja rezultato atributą.
paaiskinimas	varchar 200				Detalizuotas sąlygos aprašas.
Srautas_R					Lentelėje saugoma informacija apie informacijos srautą tarp duomenų šaltinio ir rezultato.
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
kodas	varchar 10		+	+	Unikalus įrašą apibudinantis kodas, kodą sudaro sistemos vartotojas.
pavadinimas	varchar 30		+		Vardas, kuris gali būti paimtas iš nagrinėjamo veiklos konteksto arba sukurtas paties vartotojo.
paaiskinimas	varchar 200				Detali informacija apie srautą.
ds_id_is	int		+		Duomenų šaltinio id, iš kurio išeina srautas.
r_id_i	int		+		Rezultato id į kurį įeina srautas.
Srautas_DS					Lentelėje saugoma informacija apie informacijos srautą tarp duomenų šaltinio ir duomenų šaltinio.
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
kodas	varchar 10		+	+	Unikalus įrašą apibudinantis kodas, kodą sudaro sistemos vartotojas.
pavadinimas	varchar 30		+		Vardas, kuris gali būti paimtas iš nagrinėjamo veiklos konteksto arba sukurtas paties vartotojo.

Vardas	Duomenų tipas	Pirminis raktas	Būtinas	Unikalus	Paaiškinimas
paaiskinimas	varchar 200				Detali informacija apie srautą.
ds_id_is	int		+		Duomenų šaltinio id, iš kurio išeina srautas.
ds_id_i	int		+		Duomenų šaltinio id į kurį įeina srautas.
Sr_R_elem					Lentelėje saugoma informacija apie informacijos srauto tarp duomenų šaltinio ir rezultato sandarą.
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
sr_id	int		+		Srauto id.
ds_a_id_is	int		+		Duomenų šaltinio iš kurio išeina srautas, atributo id.
r_a_id_i	int		+		Rezultato atributo id, į kurį pereina informacija iš duomenų šaltinio atributo.
Sr_DS_elem					Lentelėje saugoma informacija apie informacijos srauto tarp duomenų šaltinio ir duomenų šaltinio sandarą.
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
sr_id	int		+		Srauto id.
ds_a_id_is	int		+		Duomenų šaltinio iš kurio išeina srautas, atributo id.
ds_a_id_i	int		+		Duomenų šaltinio atributo id, į kurį pereina informacija iš duomenų šaltinio atributo.
Et_atributas					Lentelėje saugoma informacija apie duomenų šaltinių apdorojimo etapo metu įvedamus arba modifikuojamus atributus.
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
ds_a_id	int		+		Duomenų šaltinio atributo id, kuris to etapo metu apdorojamas.
et_id	int		+		Etapo id.
akt_poz	bool		+		Požymis nurodantis ar etape atributas bus Įvedimo/redagavimo ar Peržiūros režime. TRUE - Įvedimo/redagavimo, FALSE – Peržiūros.
Etapas					Lentelėje saugoma informacija apie duomenų šaltinio apdorojimo etapus.
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
kodas	varchar 10		+	+	Unikalus įrašą apibudinantis kodas, kodą sudaro sistemos vartotojas.
pavadinimas	varchar 30		+		Vardas, kuris gali būti paimtas iš nagrinėjamo veiklos konteksto arba sukurtas paties vartotojo.
paaiskinimas	varchar 200				Detali informacija apie etapą.
ds_id	int		+		Duomenų šaltinio id, kuriam priklauso etapas.
komp_poz	bool				Etapo kompiuterizavimo požymis. TRUE – kompiuterizuojamas,

Vardas	Duomenų tipas	Pirminis raktas	Būtinas	Unikalus	Paaiškinimas
					FALSE –nekompiuterizuojamas.
Perėjimas_Et					Lentelėje saugoma informacija apie perėjimus tarp duomenų šaltinio apdorojimo etapų.
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
v_id	int		+	+	Veiksmo id, kurį vykdant atliekamas perėjimas.
et_is_id	int				Etapo id iš kurio įvyksta perėjimas
et_i_id	int				Etapo id į kurį įvyksta perėjimas
Rysys_tarp_DS					Lentelėje saugoma informacija apie ryšius tarp duomenų šaltinių.
id	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
kard1	varchar 3		+		Ryšio kardinalumas viename ryšio gale. Kardinalumo aprašymo struktūra: <būtinumas>,<kardinalumas>. Būtinumas gali įgyti šias reikšmes: 0-nebūtinas, 1- būtinas. Kardinalumas gali įgyti šias reikšmes: 0 – nulis, 1 – vienas, x – daug.
kard2	varchar 3		+		Ryšio kardinalumas antrame ryšio gale. Kardinalumo aprašymo struktūra: <būtinumas>,<kardinalumas>. Būtinumas gali įgyti šias reikšmes: 0-nebūtinas, 1- būtinas. Kardinalumas gali įgyti šias reikšmes: 0 – nulis, 1 – vienas, x – daug.
vardas1	varchar 30		+		Vieno ryšio galo vardas, kuris gali būti paimtas iš nagrinėjamo veiklos konteksto arba sukurtas paties vartotojo.
vardas2	varchar 30		+		Kito ryšio galo vardas, kuris gali būti paimtas iš nagrinėjamo veiklos konteksto arba sukurtas paties vartotojo.
l_ds_r_id1	int				Lanko tarp duomenų šaltinių ryšių id, kuriam priklauso pirmas ryšio galas.
l_ds_r_id2	int				Lanko tarp duomenų šaltinių ryšių id, kuriam priklauso antras ryšio galas.
ds_id1	int		+		Duomenų šaltinio id, iš kurio eina ryšys.
ds_id2	int		+		Duomenų šaltinio id, į kurį eina ryšys.
Vardas	Duomenų tipas	Pirminis raktas	Būtinas	Unikalus	Paaiškinimas
Forma					Lentelėje saugoma informacija apie visas kiekvienam DŠ naudojamas formas
ID	int	+	+	+	Unikalus formos ID, kurį sugeneruoja DBVS.
DS_id	int		+		DŠ ID, kuriam priklauso forma.
Pavadinimas	varchar 10				Trumpas formos pavadinimas.

Vardas	Duomenų tipas	Pirminis raktas	Būtinas	Unikalus	Paaiškinimas
a	int		+		Formos lango aukštis (pikseliais)
p	int		+		Formos lango plotis (pikseliais)
Elementai					Saugoma informacija apie naudojamus grafinius elementus, skirtus vartotojo sąsajai kurti
ID	int	+	+	+	Unikalus vartotojo sąsajos elemento, kurį sugeneruoja DBVS.
Kodas	varchar 10		+	+	Unikalus elemento kodas.
Pavadinimas	varchar 30		+		Elemento pavadinimas
Aprašymas	varchar 200				Elemento aprašymas.
Funkcija					Lentelėje saugoma informacija apie analizuojamos dalykinės srities funkcijų hierarchijos funcijas.
ID	int	+	+	+	Unikalus funcijos id sugeneruoja DBVS.
kodas	varchar 10		+	+	Unikalus įrašą apibudinantis kodas, kodą sudaro sistemos vartotojas.
Pavadinimas	varchar 20		+		Funkcijos pavadinimas
Aprašymas	varchar 200				Detali informacija apie funkciją.
tipas	int		+		Funkcijos tipas: top funkcija - 1, determinuojama funkcija - 2, nedeterminuojama funkcija – 3.
id_tev	varchar 30				Tėvinės funkcijos unikalus ID.
Ak_funkcija					Saugoma informacija apie aktorių atliekamas funkcijas.
ID	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
f_id	int				Funkcijos ID, kurią gali atlikti aktorius.
ak_id	int		+		Aktoriaus ID, kuris gali atlikti funkciją.
Aktorius					Lentelėje saugoma informacija apie visus aktorius esančius nagrinėjamos veiklos kontekste.
ID	Int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
kodas	varchar 10		+	+	Unikalus įrašą apibudinantis kodas, kodą sudaro sistemos vartotojas.
pavadinimas	varchar 30		+		Aktoriaus vardas, kuris gali būti paimtas iš nagrinėjamo veiklos konteksto arba sukurtas paties vartotojo.
paaiskinimas	varchar 200				Detali informacija apie aktorių.
gr_id	int				Grupė kuriai gali priklausyti aktorius
Ak_grupe					Lentelėje saugoma informacijas apie aktorių grupes. Aktorių grupės gali būti sudarytos ir iš aktorių grupių.
ID	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
pvadinimas	varchar 20		+		Vardas, kuris gali būti paimtas iš nagrinėjamo veiklos konteksto arba sukurtas paties vartotojo.
paaiskinimas	varchar 200				Detali informacija apie grupę.
id_tev	int				Tėvinės grupės unikalus id.

Vardas	Duomenų tipas	Pirminis raktas	Būtinas	Unikalus	Paaiškinimas
Akg_funkcija					Saugoma informacija apie aktorių grupių atliekamas funkcijas.
ID	int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
f_id	int				Funkcijos ID, kurią gali atlikti aktorių grupė.
ak_id	int		+		Aktorių grupės ID, kuri gali atlikti funkciją.
F_dalis					Lentelėje saugoma informacija apie kiekviena formos epizodą
ID	int	+	+	+	Formos dalies identifikacinis numeris
F_id	Int		+		Formos, kuriai priklauso epizodas ID
F_dt_id	Int		+		Formos dalies tipo ID
Tev_fdt	Int				
Pavadinimas	Char(50)		+		Dalies pavadinimas
X	Int		+		Dalies viršutinio kampo koordinatė X ašyje
y	Int		+		Dalies kairio kampo koordinatė Y ašyje
a	Int		+		Dalies aukštis
p	int		+		Dalies plotis
F_dalies_tipas					Lentelėje saugomi galimi formų dalių tipai
ID	Int	+	+	+	
Pavadinimas	Char(20)		+		
Aprašymas	Char(200)				
F_elementas					Informacija apie naudojamus grafinius elementus vartotojo sąsajai kurti
ID	int	+	+	+	Identifikacininis numeris
X	Int				Elemento kairio kampo koordinatė x ašyje
Y	Int		+		Elemento viršutinio kampo koordinatė y ašyje
A	Int		+		Elemento plotis
P	Int		+		Elemento aukštis
P_kriterijus	Bit				
P_tipas	Bit				
Reiksme_default	Char(50)				Rodomas tekstas
F_d_id	Int		+		Formos dalies, kuriai priklauso elementas ID
E_t_id	Int		+		Elemento tipo id
Ds_a_id	Int		+		Duomenų šaltinio id
Grupes_id	Int				Grupės, kuriai priklauso elementas id
Sriftodydis	Int				Teksto šrifto dydis
Remelis	Int				Rėmelio plotis
Lygiavimas	Char(10)				Teksto lygiavimas
Sspalva	Char(7)				Teksto spalva
fspalva	Char(7)				Fono spalva
Element_grupe					Apjungtų elementų grupės
ID	Int	+	+	+	
Pavadinimas	Char(20)		+		
E_t_id	int		+		Elemento tipo id
Element_tipas					Grafikos elementų tipai
ID	int	+	+	+	

Vardas	Duomenų tipas	Pirminis raktas	Būtinas	Unikalus	Paaiškinimas
Kodas	Varchar(10)		+		
Pavadinimas	Varchar(30)		+		Grafikos elemento pavadinimas laisva forma
Aprašymas	Varchar(200)				
HTML_pavadinimas	Char(20)		+		Grafikos elemento pavadinimas HTML kodu
XSL_kodas	Varchar(1000)		+		XSL taisyklė transformavimui į XHTML

10.6. Straipsnis magistrantų, doktorantų konferencijoje

AUTOMATIZUOZTAS INTERNETINIŲ IS VARTOTOJO SĄSAJŲ KŪRIMAS

Marius Pečiulaitis
Kauno technologijos universitetas
Informacijos sistemų katedra, Studentų 50-308
[Trumpa straipsnio apžvalga]

Įvadas

Patogi grafinė vartotojo sąsaja yra vienas iš pagrindinių programinės įrangos kokybės vertinimo kriterijų. Tačiau daugelis kūrėjų turi per mažai žinių apie vartotojo sąsajos dizaino įpatumus arba tam skiria per mažai dėmesio. Be to, vartotojai reikalauja, kad informacija bei funkcijos būtų pasiekiamos vienodai patogiai ir efektyviai, nepriklausomai nuo sistemos, net ir dinamiškai kintant sistemos aplinkai, duomenims ar jų pobūdžiui.Tinkamai sukurta vartotojo sąsaja leidžia vartotojams daug greičiau perprasti darbo su sistema specifiką

Taip pat aktualu, kad turima sistema vartotojai galėtų naudotis nepriklausomai nuo naudojamos tiek techninės, tiek programinės įrangos ir jos pokytis neįtakotų programinės įrangos funkcionalumo bei vartotojo sąsajos ypatumų.

Pagrindiniai aspektai į kuriuos reiktų atkreipti dėmesį yra [5]:

- *Pastovumas.* Stengtis mygtukus skirtinguose programos languose išdėstyti pastoviose vietose, pavyzdžiui, jei viename lange galima iš sąrašo išsirinkti elementą ant jo spragtelėjus du kartus pele, tą pačią galimybę reik įdiegti ir kitose programos vietose. Taip vartotojas greičiau įsivaizduos, kaip sistema veikia.
- *Nustatyti standartus ir jų laikytis*. Norint išlaikyti pastovumą sistemoje, reikia joje sekti apibrėžtus dizaino standartus. Geriausia pritaikyti industrinius, tokius kaip IBM ar Microsoft.
- *Sekti kontrasto taisykles*. Šviesiame fone naudoti tamsų tekstą ir atvirkščiai.
- *Neslėpti neaktyvių kontrolės elementų*. Padaryti juos pilkais. Taip vartotojas greičiau supras, kaip sistema veikia ir žinos, kokie veiksmai jam leidžiami.
- *Lygiuoti laukus*. Jei lange daugiau negu vienas koregavimo laukas, juos reikėtų lygiuoti viena kryptimi.
- *Nekurti sudėtingų langų*. Perkrautais langais sudėtinga naudotis ir juos suprasti.
- *Efektyviai grupuoti elementus*. Elementai, kurie yra logiškai susiję, turėtų būti greta vienas kito, o kiti atskirai.
- *Iššokantys meniu neturėtų būti vienintelis funkcijų vykdymo šaltinis.* Vartotojai negali mokytis naudotis sistema, jeigu dauguma galimų funkcijų yra paslėptos.

Šio projekto tikslas - atsižvelgiant į apdorojamų duomenų specifikaciją, sukurti multi platforminę internetinę sistemą, optimaliai parenkančią grafinės vartotojo sąsajos įgyvendinimą tiek dizaino, tiek programinio kodo aspektu.

Technologijos

Programinės įrangos su grafine vartotojo sąsaja kūrimas šiandieną reikalauja įsipareigojimo dirbti su aplikacijų programavimo sasaja. (API) [3]. Kadangi tam reikia daug programinio kodo, tyrinėtojų pastangos nukrypo į tikslą, sukurti deklaratyvų grafinės vartotojo sąsajos tekstinį aprašymą. .

Tad pagrindiniai aspektai į kuriuos atsižvelgiama renkantis tinkamiausią technologiją norint automatizuoti GUI kūrimą yra šie:

- sugeneruotos grafinės vartotojo sąsajos nepriklausomumas nuo platformos;
- programavimo kalbos paprastumas;
- sugeneruoto programinio kodo optimalumas;
- programinio kodo atitikimas standartus.

Vienas iš paprastesnių būdų, padedančių pasiekti užsibrėžtus tikslus, yra rinktis grafinę Web vartotojo sąsają. Taip neapkraunama kliento papildoma programine įranga („plonas“ klientas), kurią jis privalėtų specialiai diegti tam, kad galėtų naudotis viena ar kita sistema. Todėl pakaktų turėti vieną iš populiarių interneto naršyklių. Taip, praktiškai išpildoma pirma sąlyga – daugiaplatformiškumas. Kuriant naršykles, norėta, jog vartotojas internetinį resursą galėtų pasiekti naudodamasis, bet kuria operacine sistema (realizuotų nuo platformos nepriklausančias vaizdavimo kalbas), nereikalaujant iš vartotojo specifinių naudojimosi ja žinių. Tad, grafinės vartotojo sąsajos realizavimas apsiriboja kalbomis, kurias supranta įprastos naršyklės: HTML, PHP, XML, XHTML, XSL, XSLT, Java ir pan.

Tikslas yra sukurti abstraktų aprašymą ir terpę, kuri galėtų automatizuotai siūlyti tinkamą grafinę vartotojo sąsają, pagal naudojamą informaciją bei sistemos funkcijas konkretiems atvejams. Tai taip pat dar vadinama vartotojo sąsajos plastiškumu.

HTML problema yra ta, kad tai buvo sukurta interakcijai tarp žmonių ir mašinų. Kai tinklas buvo išrastas vėlyvais 1980 metais, viskas atitiko to meto poreikius [9]. Tačiau kai tinklas persikėlė į kitus gyvenimo sritis, iš HTML buvo pareikalauta atlikti daugybę neįprastų dalykų. Vėliau, iš dalies, kad praplėsti HTML galimybes, buvo sukurta XML (eXtensible Markup Language) - išplėstinė žymių kalba. Jos pagrindu taip pat sukurtos XSLT bei XSL kalbos. Pastaroji apibūdina XML dokumento stiliaus keitimą naudodama XSLT, nusakydama kaip dokumentas yra transformuojamas į kitą XML dokumentą.

Vienas iš galimų variantų kuriant GUI – panaudoti XUL kalbą [7]. Jos paskirtis yra kurti grafines programas vietoj to, kad formuoti dokumentus ir kurti tinklapius; taigi, tai yra vaizdavimo formalizmų tokių kaip HTML ar DHTML papildymas. Naudojant variklį Gecko, programuotojai gali įgyvendinti multi platforminę grafinę vartotojo sąsają t.y. tokią sąsają, kurios vaizdavimas nepriklauso nuo įrenginio.

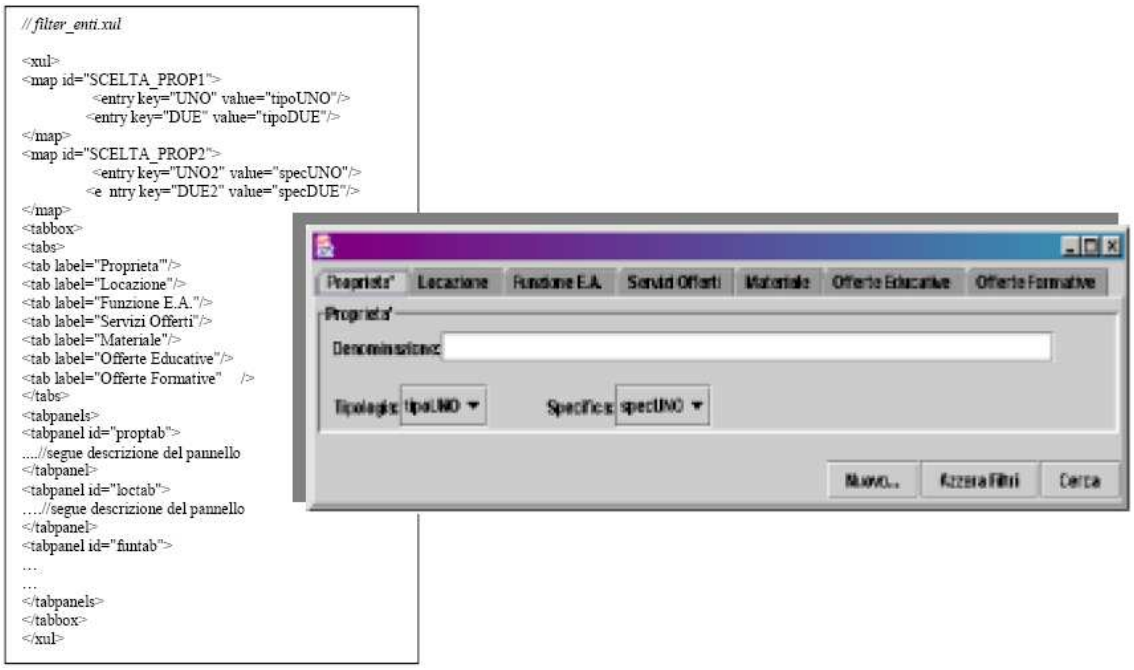
XUL yra paremta konceptu, jog grafinės sąsajos yra autonominės nuo programų esybės.

Taigi, grafinė vartotojo sąsaja gali būti pateikiama aukštame išvaizdos lygyje naudojant hierarchiškai organizuotų elementų rinkinius sudarytus iš XML formalizmų.

XUL programos gali būti atidaromos tiesiog tinklapyje arba atsisiųstos ir instaliuotos (instaliavimas supaprastintas iki minimumo). Tačiau, norint pasinaudoti XUL teikiamomis galimybėmis, visuomet reikalingas papildomas atvaizdavimo variklis Gecko, kurį turės ne kiekvienas sistema besinaudojantis vartotojas.

Luxor-XUL [7] yra atviro kodo projektas, kurio tikslas integruoti Mozillos XUL funkcionalumą su Java portatyvumu, kad būtų galima sukurti nuo platformos nepriklausančią vartotojo sąsają, XUL bylos naudojamos apibūdinti grafiniams elementams, o XML naudojama įgyvendinti tam tikrus objektus, tokius kaip medžiai, sąrašai ir lentelės. XUL dokumente grafinė vartotojo sąsaja apibūdinama hierarchiniu atžvilgiu; hierarchiniai ryšiai tarp elementų yra tinkamai pateikiami dokumento medžio struktūros pagalba. Luxor reikalauja, jog identiška struktūra būtų laikinai patalpinama atmintyje; kad tai būtų pasiekta naudojama JDOM paketas (JDOM yra atviro kodo API XML dokumentų kūrimui, analizavimui bei koregavimui). Paveiksle Nr. 3 pavaizduota, kaip Luxor sąveikauja su Java programa: kuomet XUL sąsajos aprašymas yra baigtas, turi būti paruošiamos Luxor klasės, kurios galės sugeneruoti grafinę vartotojo sąsają su visomis mygtukų funkcijomis.

Taigi, naudojant tokią kalbą kaip XUL, yra patogu kurti grafines vartotojo sąsajas, nes tai reikalauja santykinai mažai programinio kodo ir taip sutaupoma daug laiko. XUL kalba, sukurta Mozillos kompanijoje yra gana galinga, bet veikia tik Mozillos/Netscape kontekste (naršyklėse paremtose Gecko varikliu). Luxor XUL atviro kodo projektas bando praplėsti XUL naudą kitoms programoms paremtoms Java technologijomis. Deja, Luxor tikslai nėra galutinai pasiekti: dauguma XUL galimybių nėra visiškai suderinama su Java. Tačiau bendru atžvilgiu, XUL kalba yra paprasta ir norint modifikuoti vartotojo sąsają, nereikia koreguoti Java kodo. Taigi mažesnis programinio kodo kiekis galėtų pagerinti visos IS savybes.



1 pav. Programos langas ir jam sugeneruoti reikalingas kodas

Priimtiniausias variantas, kurį galima panaudoti kuriant grafinę vartotojo sąsają, yra XSL transformacijos. Informaciją apie elementus bei jų išdėstymą, saugomą duomenų bazėje, patogu išgauti XML pavidalu bei XSLT pagalba formuoti XHTML kodą, kuris būtų „suprantamas“ daugumos populiariausių internetinių naršyklių. XSLT procesorius skaito vienu metu XML dokumentą bei XSLT stilių rinkmeną, kurioje pateikiamos taisyklės nurodančios, kaip reikia transformuoti XML bylą. Taip gaunamas HTML kodas.

Transformacijos procesas vyksta labai panašiu keliu kaip ir skriptų rašymo kalbos Python ar Perl – pritaikomos taisyklingos išraiškos prie įvedamo duomenų srauto ir rasti elementai transformuojami į išeinančius duomenis [9]. Šiuo atžvilgiu XSLT galėtų būti vadinama skriptų kalba, ypač dėl to, jog turi elementus srauto kontrolei. Dauguma XSLT atliekamų funkcijų galima įvykdyti naudojant API tokias kaip SAX (Simple API for XML) ar DOM(Document Object Model). Tačiau, kaip bebūtų, XSLT turi pranašumų. Vieną iš jų iliustruoja XML tikslas: duomenų atskyrimas nuo programos logikos ir kodo. Kadangi XSLT neturi nuo platformos priklausančios sintaksės, mūsų duomenys gali būti konvertuojami, į bet kokį pasirinktą formatą, kombinuotą su programiniu kodu ir pritaikytą, bet kokiam įrenginiui. Taigi, „švari“ informacija gali būti nukreipta netgi į dar nesukurtus įrenginius ar sistemas.

XML, XSLT bei XHTML generavimas

Informacijos sistemų katedroje šiuo metu kuriamame metodas reikalavimų specifikavimui ODRES (Output Driven Requirements Specification) [13] grafinės vartotojo sąsajos kūrimo automatizavimas vyksta tokiu scenarijumi:

Pagal surinktus vartotojų reikalavimus, MS Visio aplinkoje atliekamas grafinės vartotojo sąsajos ir DB schemos projektavimas. Vartotojo reikalavimai bei suprojektuoti kuriamos sistemos elementai (vartotojo sąsaja, DB modelis) saugomi saugykloje. Šiuo metu saugykla realizuota MS Access duomenų bazėje, tačiau vėliau ketinama DB perkelti į MS SQL DBVS, kad įrankiu galėtų naudotis ir nutolę vartotojai.

Informacija iš atitinkamų duomenų bazės lentelių konvertuojama į XML kodą. Tam atlikti galima pasinaudoti pavyzdžiui, duomenų bazių valdymo sistemų kūrimo įrankiu Visual FoxPro. Kadangi XML žymės nėra iš anksto apibrėžtos, galima pasinaudoti šiuo privalumu ir jas susikurti taip, jog lengvai suprastumėme ir galėtumėme atskirti sugeneruoto XML dokumento elementus (tarkim Lietuviškais žodžiais). XSL transformacijų rinkmenas taip pat galima kurti pusiau arba visiškai automatizuotai .

10 lentelė. Elementų aprašai duomenų bazėje.

Tipas	Plotis	Aukstis	Fonas
Lentele	3	2	#CCCCCC

Pvz MS SQL duomenų bazėje turint tokią lentelę (1 lentelė) su galimų grafinių elementų aprašais, būtų galima sugeneruoti tokį XML kodą:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="Sablonas.xsl"?>
<dokumentas>
        <elementas>
                <lentele>
                        <fonas> #CCCCCC</fonas>
                        <eilute>
                                <stulpelis>vienas</stulpelis>
                                <stulpelis>du</stulpelis>
                                <stulpelis>trys</stulpelis>
                        </eilute>
                        <eilute>
                                <stulpelis>keturi</stulpelis>
                                <stulpelis>penki</stulpelis>
                                <stulpelis>sesi</stulpelis>
                        </eilute>
                </lentele>
        </elementas>
</dokumentas>
```

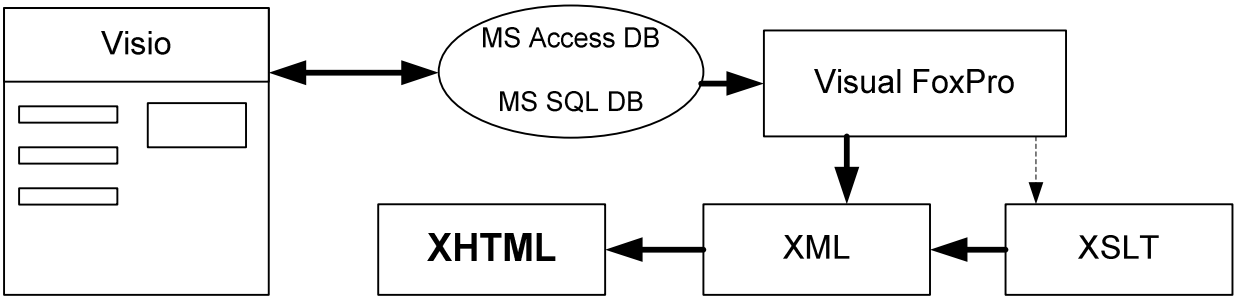
Panaudojus XSL transformacijų rinkmeną:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" version="1.0" encoding="UTF-8" indent="yes"/>
<xsl:template match="/">
<html>
		<body>
			<xsl:for-each select="dokumentas/elementas">
				<xsl:for-each select="lentele">
					<table border="1" bgcolor="{fonas}">
						<xsl:for-each select="eilute">
							<tr>

		<xsl:for-each select="stulpelis">

		<td>

			<xsl:value-of select="."/>

		</td>

		</xsl:for-each>
							</tr>
						</xsl:for-each>
					</table>
				</xsl:for-each>
			</xsl:for-each>
		</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Suformuojamas XHTML dokumentas su grafinės vartotojo sąsajos prototipu:

vienas	du	trys
keturi	penki	sesi

XML kodo generatorius radęs elementą „Lentele“, atributus „Plotis“ ir „Aukstis“ interpretuoja kaip stulpelių bei eilučių skaičių lentelėje. Kitiems grafiniams elementams tai gali būti tiesiogiai interpretuojama kaip aukštis ir plotis.



2 pav. Grafinės vartotojo sąsajos kūrimo procesas.

Išvados

Straipsnyje aptarti pagrindiniai aspektai į kuriuos reiktų įvertinti kuriant grafinę vartotojo sąsają.

Taip pat išanalizuotas vieno iš projekto (Luxor-XUL), transformuojančio Mozillos aplikacijų grafinei sąsajai generuoti sukurtą kalbą XUL į Java programas, veikimo principas bei suteikiamos galimybės transliavimo. Kurti panašų projektą (generuoti iš XUL Java kodą) atsisakyta, nes dauguma XUL galimybių nėra visiškai suderinama su Java.

Palyginti keli galimi kodo automatizavimo procesai. Aptartos XML, XSLT, XUL kalbų suteikiamos galimybės XML formato informacijos transformavimui į internetinių tinklapių naršyklei suprantamą kodą. Dėmesys buvo kreipiamas į kalbos daugiaplatforminį pritaikomumą, taip pat į reikiamą papildomą programinę įrangą, skirtą realizuoti automatinį generavimą ar sugeneruoto kodo atvaizdavimą. Šioje analizės dalyje nuspręsta, jog tinkamiausiai tam tiks XSLT stilių pagalba iš XML kodo formuojamas XHTML ar HTML kodas. XUL atsisakyta dėl reikalaujamo papildomo Gecko variklio kalbai atpažinti.

Siekiat panaudoti ODRES metodo rezultatus IS projektavime. Sekančiuose etapuose bus tiksliai apibrėžtos vartotojo sąsajos meta modelio lentelių struktūras, kuriose bus saugoma informacija apie grafinei vartoto sąsajai kurti reikalingus elementus (teksto laukeliai, paveikslėliai, mygtukai ir pan.), bei sukuriamas algoritmas XML bei XSLT kodo generavimui.

Literatūros sąrašas

[1] J. Sánchez Díaz, O. López, and Juan J. Fons. From User Requirements to User Interfaces: A Methodological Approach. Department of Information Systems and Computation
Valencia University of Technology. Valencia, Spain, 2000. [žiūrėta 2006-01-20].

[2] Ch. Pfisterer. Semantic Description Language
for Platform-Independent
Graphical User Interfaces. Institut für Visualisierung und Interaktive Systeme Universität Stuttgart. [žiūrėta 2006-01-18].

[3] A. Savidisa, C. Stephanidisa . Unified user interface design: designing universally
accessible interactions. 2003. [žiūrėta 2006-01-18].

[4] D. Duke, B. Fields and M. D. Harrison. A Case Study in the Speci_cation and Analysis
of Design Alternatives for a User Interface. Human-Computer Interaction Group, Department of Computer Science, University of York, UK, 1999. [žiūrėta 2006-01-18].

[5] S. W. Ambler. User Interface Design: Tips and Techniques. President, Ronin International, 2000. [žiūrėta 2006-12-22].

[6] J. W. Jespersen* and J. Linvald. Investigating User Interface Engineering
in the Model Driven Architecture. IT-University of Copenhagen, Glentevej 67, 2400 NV, Denmark, 2004. [žiūrėta 2006-01-22].

[7] N. Aloia, C. Concordia, M. Teresa Paratore. Automatic Gui Generation For Web Based Information Systems. Institute of Information Science and Technologies Italian National Research Council Via Moruzzi, 1 - CNR Research Area, Pisa (56100) Italy. [žiūrėta 2006-01-21].

[8] XUL Tutorial. 2005. [žiūrėta 2005-12-27]. Prieiga per Internetą: http://xulplanet.com/tutorials/xultu/.

[9] Transforming XMI to HTML. [žiūrėta 2006-01-17]. Prieiga per Internetą:
http://www.objectsbydesign.com/projects/xmi_to_html.html

[10] XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999. [žiūrėta 2005-10-14]. Prieiga per Internetą: http://www.w3.org/TR/xslt

[11] XML Technology Center.Updated December 27, 2005. [žiūrėta 2005-12-20]. Prieiga per Internetą:
http://www.oracle.com/technology/tech/xml/index.html

[12] HTML and XSLT by Bob DuCharmeAugust 30, 2000. [žiūrėta 2005-12-19]. Prieiga per Internetą:
http://www.xml.com/pub/a/2000/08/30/xsltandhtml/

[13] R. Butleris, R.Butkienė, T. Danikauskas, S. Jasiukevičius. Straipsniai, metodų aprašymai. [žiūrėta 2005-10-03]. Prieiga per Internetą: ftp://isd.ktu.lt/Isd/Butkiene/Metodo_Magistrantams