KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

INFORMACIJOS SISTEMŲ KATEDRA

Ieva Subačiūtė

Vartotojo sąsajos elementų modeliavimas informacijos srautų specifikacijos pagrindu

Magistro darbas

Darbo vadovas

doc. dr. Rimantas Butleris

Konsultantas

doktorantas Tomas Danikauskas

Kaunas
2004

TURINYS

1. Įvadas 4

2. Vartotojo sąsajos kūrimo principų analizė 5

2.1.Tyrimo sritis, objektas, problema 5

2.2. Analizės metodų ir priemonių parinkimas 7

2.2.1. Kaip pasiruošti vartotojo sąsajos kūrimui 7

2.2.2. Iteracinis prototipo kūrimas, pagrįstas integruota funkcinių ir vartotojo sąsajos reikalavimų specifikacija 16

2.2.3. Reikalavimų analizė ir prototipo kūrimas naudojant scenarijus ir būsenų diagramas 19

2.2.4. KIS funkcionalumo rezultatų struktūros specifikavimas 21

2.2.5. KIS funkcionalumo rezultato struktūros specifikavimo procesas 23

2.2.6. Duomenų šaltinių struktūros specifikavimas 23

2.2.7. Duomenų šaltinių apdorojimo etapų specifikavimas 24

2.2.8. Duomenų šaltinių apdorojimo etapų specifikavimo procesas 24

2.3. Projekto tikslas, jo pagrindimas ir kokybės kriterijų apibrėžimas 26

3. Reikalavimų sistemai vartotojo sąsajai modeliuoti specifikacija ir sistemos projektas 28

3.1. Projekto techninė užduotis 28

3.2. Kuriamos sistemos reikalavimų modelio aprašymas 29

3.2.1. Panaudojimo atvejų diagrama 29

3.2.2. Veiklos diagramos panaudojimo atvejų modelyje 30

3.3.3. Dalykinės srities klasių diagrama 32

3.3.4. Vartotojo sąsajos modelis 33

3.3. Sistemos, vartotojo sąsajai kurti, projekto pateikimas 34

3.3.1. Sistemos panaudojimo atvejų bendradarbiavimo diagramos 34

3.3.2. Sistemos elgsenos modelis 34

3.3.3. Duomenų bazės modelis 38

3.3.4. Sistemos architektūros aprašymas 39

3.3.5. Testavimo duomenų aprašymas 45

3.3.6. Reikalavimai sistemos funkcionavimo palaikymui 48

3.3.7. Sistemos naudojimo instrukcija 48

4. Sistemos vartotojo sąsajai kurti eksperimentinis tyrimas 49

4.1. Sukurtos sistemos kokybės tyrimas 49

4.2. Tolimesnio sistemos tobulinimo, plėtojimo galimybės 50
5. Išvados 52
6. Literatūra 54
7. Terminų ir santrumpų žodynas 55
8. Summary 56
9. Priedai.....57

1. Įvadas

Daugelis įmonių, siekdamos sutrumpinti darbų atlikimo laiką, sumažinti kainą ir pagerinti vartotojų poreikių tenkinimą susiduria su poreikiu pertvarkyti savo veiklos procesus. Dėl to diegiamos kompiuterizuotos informacinės sistemos (IS). Kuo daugiau randasi norinčių tai padaryti, tuo mažiau laiko ir galimybių kokybiškai ir efektyviai tai atlikti turi IS inžinieriai. Dėl to, šiuo metu sparčiai plėtojasi kompiuterizuoto IS projektavimo (CASE) sistemos, ieškoma, geresnių būdų probleminei sričiai aprašyti ir organizacijų veiklos procesams atvaizduoti. [2]

IS kompiuterizuotas projektavimas – vienas iš metodų užtikrinti vartotojo reikalavimų analizės ir specifikavimo etapo kokybę. CASE (*Computer-Aided Software Engineering*) priemonės leidžia atsiriboti nuo painaus programinio kodo. Tai priartina IS kūrimą prie vartotojo lygio. Jis (vartotojas – užsakovas) turi galimybę pats dalyvauti kūrimo procese [3].

Integruotos CASE priemonės padeda:

1. Analizavimo ir modeliavimo etapuose (Upper CASE tools);
    - Kuriant analizės diagramas;
        - procesų modelis;
        - duomenų modelis;
    - Įvairiose stadijose generuojant dokumentaciją;
    - Projekto valdymui ir kontrolei;
    - Generuojant modelio diagramas;
        - Programos modelį (struktūros diagramą);
        - Programos loginį modelį (srautų diagramą);
2. Diegimo ir aptarnavimo etapuose (Lower CASE Tools);
    - Automatizuotame kodo generavime (bet kokios kalbos aplinkoj);
    - Automatizuotame schemos generavime (bet kurioj DBVS aplinkoj).

Kodėl visiems šiems etapams naudojamos CASE priemonės?

1. Tam, kad būtų padidintas analitiko produktyvumas;
    - Našus kūrimas ir modifikavimas;
    - Lengvesnis bendradarbiavimas grupiniu darbu pagrįstoje aplinkoje;
    - Galimybė stebėti visus daromus pakeitimus.
2. Kad palengvėtų analitiko ir vartotojo bendravimas (būtų nedviprasmiškas);

- Ataskaitų ir formų schemos gali būti sukurtos ir redaguojamos atsižvelgiant į vartotojo atsiliepimus;
- Vartotojai gali iš karto pamatyti pasikeitimų duomenų srautuose įtaką;

3. Efektyvus techninis aptarnavimas;
   - Leidžia efektyvų sistemos techninį aptarnavimą;
     - Kiekvienas modelio pakeitimas bus automatiškai integruojamas (įjungiamas) į programinį kodą [4].

   Visos minėtos CASE įrankių savybės labai pagreitina IS projektavimą.

   Šiame darbe bus realizuota viena sudėtingo CASE įrankio dalis.

2. Vartotojo sąsajos kūrimo principų analizė

2.1. Tyrimo sritis, objektas, problema

Tyrimo objektas – funkcinių reikalavimų, keliamų kompiuterizuojamai informacinei sistemai (KIS), specifikavimo metodas ir jo panaudojimas informacinės sistemos (IS) projektavime.

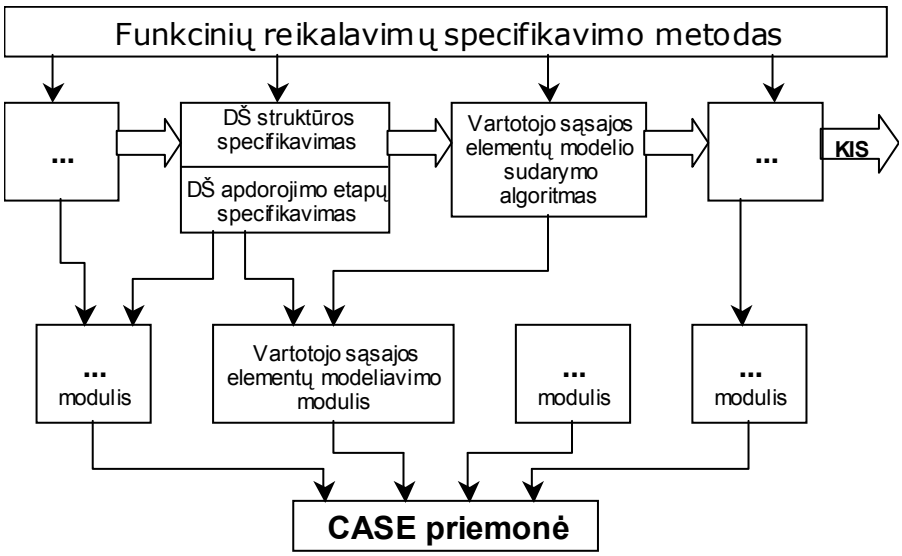
Informacinė sistema (IS) yra kompleksas programinių, techninių bei organizacinių priemonių, padedančių įmonėms įgyvendinti konkrečius savo veiklos uždavinius. Šios priemonės tarpusavyje glaudžiai susijusios – viena programa turi suprasti kitos perduodamus duomenis, kompiuterinė technika turi atitikti programų reikalavimus, darbuotojai turi vykdyti jiems priskirtas funkcijas griežtai nustatyta tvarka. Įmonės informacinės sistemos turi bendrauti ne tik tarpusavyje, bet ir su partneriais bei klientais, valstybinėmis, finansinėmis bei kitomis organizacijomis.

Kompiuterizuojamos informacinės sistemos atliekamas funkcijas nusako sistemos funkciniai reikalavimai, o uždaviniams informacinėje sistemoje atlikti taikomos kompiuterinės technologijos.

Svarbu surinkti ir suprasti funkcinius KIS keliamus reikalavimus ir juos teisingai specifikuoti. Svarbiausias ir pagrindinis funkcinis reikalavimas – išvedamos informacijos (rezultatų) tipas bei struktūra. Nuo šio reikalavimo priklauso kiti funkciniai KIS reikalavimai.

Duomenų šaltinių ir rezultatų struktūros specifikavimas bei DŠ apdorojimo specifikavimas susiveda į vartotojo sąsajos elementų modeliavimo modelio kūrimą (pav. 1). Duomenų šaltinių struktūros ir rezultatų struktūros specifikacija bei duomenų šaltinių apdorojimo specifikacijos yra saugomos specifikacijos saugykloje – meta duomenų bazėje

(priedas pav. 2, 3) Remiantis šiomis specifikacijomis, reikia padaryti modulį, kuris kurtų vartotojų sąsajos elementų prototipus kiekvienam duomenų šaltiniui, t.y. modulis, CASE priemonės dalis, turėtų palengvinti KIS projektavimą.



1 pav. Vartotojo sąsajos elementų modelis – CASE priemonės modulis

Specifikacijų saugyklos – meta duomenų bazės schema ir jos objektų atributų paaiškinimai pateikti priede pav. 2 ir pav. 3 bei 1 lentelėje.

Pagrindinės IS posistemės yra įvedimo ir išvedimo. Vartotojas įveda informaciją (duomenis) įvedimo posistemės pagalba, o išvedimo posistemė juos apdoroja ir vėl pateikia vartotojui. Kuriant IS, įeigos objektai (duomenų šaltiniai) ir išeigos objektai (funkcionalumo rezultatai) traktuojami taip: pirmiausiai išsiaiškinama, kokius išeigos rezultatus reikia gauti, tuomet nustatoma įeigos objektų forma, ir galiausiai sprendžiamas duomenų šaltinių transformavimo klausimas [1].

2.2. Analizės metodų ir priemonių parinkimas

2.2.1. Kaip pasiruošti vartotojo sąsajos kūrimui

Šio darbo tikslas – modulis, kuris analitikui padės kurti informacinę sistemą, o tiksliau – vartotojo sąsają. Prieš ją kuriant labai svarbu suprasti vartotoją, reikalavimus, todėl modulis turėtų suteikti analitikui gana plačias galimybes modeliuojant vartotojo sąsajos elementus.

Vartotojo sąsajos kūrimą galima palyginti su namo statymu: jei blogai sudėti pamatai, nesvarbu kiek bus atlikta apdailos darbų – bendra konstrukcija jau bus sugadinta [5].

Jeigu sistemos interaktyvus bendravimas su žmogumi – vartotoju nėra malonus ir sklandus, – atsiradęs trūkumas sugadins visos sistemos darbą, nors kitais aspektais sistema ir bus visiškai tiksli[5].

Reikia stebėti, klausytis ir kalbėtis su žmonėmis, kurie naudos produktą. Suprasti duomenis, paversti juos vertinga informacija projektavimui ir panaudoti ją vartotojo sąsajos modeliavimui [4].

Vartotojo sąsaja – tai vaizdas, kurį mato ir kurio pagalba vartotojas naudojasi produktu. Sąsaja gali pagelbėti arba supainioti, būti efektyvi arba neefektyvi. Svarbu žinoti, kaip gauti ir pasinaudoti informacija, reikalinga tam, kad būtų galima sukurti naudingą, efektyvią vartotojo sąsają [4].

Neįmanoma atsitiktinai sukurti efektyvios vartotojo sąsajos. Tai galima padaryti, tik kai projektuotojas gerai supranta tiek žmones – vartotojus, tiek technologijas. Labai svarbu žinoti, kas naudosis sistema, vartotojams būdingus įpročius, fizines galimybes, jų atliekamų uždavinių apribojimus. Reikia žinoti, kokį rezultatą vartotojas nori matyti ir kaip ji įsivaizduoja savo atliekamus uždavinius, t.y. konceptualų darbo ir įrankių modelį. Projektuotojas taip pat turi suprasti aplinkybes, kurioms esant vartotojai dirba ir kaip jie bendradarbiauja siekdami tikslo [4].

Gera sąsajos išvaizda yra labai svarbi viso projekto dalis. Ji nuteikia vartotojus produktyviau dirbti, didina jų pasitenkinimą produktu, o tai yra pagrindinis ir didžiausias daugelio kompanijų tikslas. Graži išvaizda didina našumą ir mažina pagalbos skambučių galiotojams skaičių, tokiu būdu mažėja išlaidos [4].

Yra daug priežasčių, kodėl kyla problemos naudojantis vartotojo sąsaja. Viena jų tai, kad projektuotojų prašoma įdėti vis daugiau ir daugiau funkcijų į sistemą. Tačiau pagrindinė priežastis yra ta, kad dažnai programinės įrangos produktai kuriami koncentruojantis ties

technologinėm galimybėm, o ne ties vartotojo poreikiais. Problemos esmė – tai tiesioginio, kruopščiai apsvarstyto vartotojų ir jų užduočių bei aplinkos supratimo trūkumas [4].

Prieš kuriant vartotojo sąsają labai svarbu tam gerai pasiruošti, pasirinkti strategiją, kuri leistų sukurti patogią vartotojo sąsają. Svarbu suprasti, ką reikia žinoti apie vartotojus, jų atliekamas užduotis bei juos supančią aplinką. Visuomet reikia stengtis kurti projektą orientuotą į vartotoją, t.y. surinkti visą reikiamą informaciją prieš projektuojant ir panaudoti ją projektuotojų grupės dirbančios vartotojui reikmėms [4].

Vartotojo sąsaja – tai tiltas tarp informacinės sistemos ir vartotojo pasaulių. Tai būdas, kuriuo vartotojas bendrauja su produktu tam, kad pasiektų savo tikslą. Tai sistemos būdas save parodyti vartotojui ir jos elgesys vartotojo poreikių atžvilgiu [4].

Vartotojo sąsaja, kurią lengva naudoti turi savybes, kurios pateiktos 1 lentelėje.

1 lentelė. Lengvai naudojamos vartotojo sąsajos savybės.

Atspindi darbų sekas, kurios yra gerai žinomos ar ypač patogios vartotojui.
Palaiko vartotojo mokymosi būdą (stilių).
Yra suderinama su vartotojos darbo aplinka.
Apima projekto idėją, kuri yra suprantama vartotojui.
Pateikiama koncentruotai, kad būtų patikimesnė ir lengviau suprantama.
Joje naudojama tokia kalba ir paveikslėliai, kad vartotojams būtų lengviau išmokti ja naudotis.

Trumpiau tariant, lengvai naudojama vartotojo sąsaja paprastai ir elegantiškai įsiterpia į vartotojo gyvenimo ir darbo poreikių aplinką. Jei ji nėra iš karto akivaizdžiai lengvai vartojama, tai ją galima gretai perprasti ir išmokti [4].

Prieš kuriant sistemą, nepakanta paklausti vartotojo, kokios sąsajos jam reikia, nes tai, ką jis papasakos gali pasirodyti labai nekonkretu ir be galo nutolę nuo iš tikrųjų jo atliekamų užduočių. Vartotojo ir jo užduočių analizė yra pagrįsta giliu supratimu, kaip vartotojas šiandien atlieka užduotis. Šis supratimas apima:

- Vartotojų tikslus, tai ką jie bando pasiekti.
- Ką konkrečiai daro tiems tikslams pasiekti.
- Kokias asmenines socialines ir kultūrines savybes vartotojai suteikia atliekamoms užduotims.
- Kaip jie yra veikiami savo fizinės aplinkos.
- Kaip vartotojų patirtis ir žinios veikia jų darbų sekas, kurias jie atlieka, kad įvykdytų užduotis.

- Ką jie labiausiai įvertins naujojoje sąsajoje – greitį, tikslumą, atsistatymą po klaidų, žmogiškąjį kontaktą, linksmumą, iššūkį?

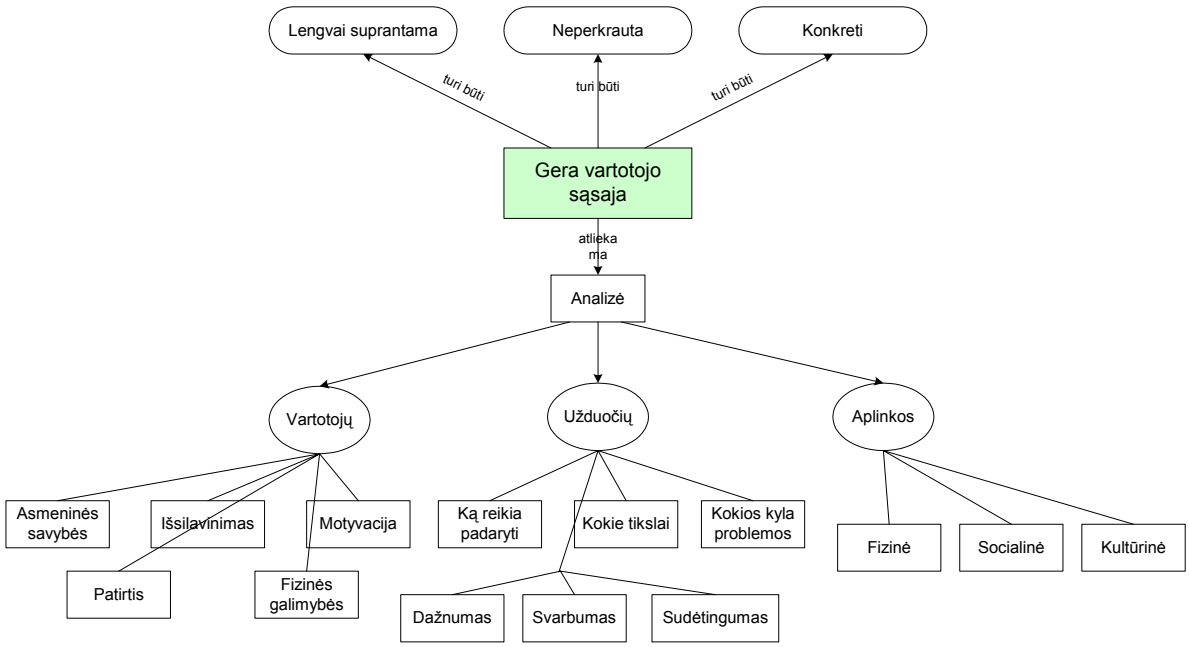
Atlikus vartotojų ir jų užduočių analizę vienam projektui, gali pasirodyti, kad šią patirtį galima perkelti ir kitiems, būsimiems produktams kurti, tačiau negalima visiškai ir išskirtinai pasikliauti tik seniau atliktais tyrimais. Kiekvienam naujam produktui, naujam kūriniui, kiekvienai naujai vartotojų grupei, kiekvienam išplėtimui į kitą rinką ar kita kultūrą, reikia atlikti naują vartotojų ir jų užduočių analizę [4].

Netgi stebint vartotoją jau kūrimo, projektavimo procese, galima pastebėti vis naujų ypatybių, gauti vis naujos informacijos. Tačiau ji jau tikriausiai nepakeis sąsajos projekto krypties. Be to, kuo vėliau bus pastebėtos problemos, tuo brangiau kainuos pakeitimai. Todėl labai svarbu iš karto viską atlikti kuo tiksliau. Ir informacija, apie kurią sužinote vėliau turėtų būti skirta mažiems, smulkiems pataisymams, ne dideliems pakeitimams [4].

Tai, kada jūs pirmą kartą stebite, kaip vartotojai susiduria su produkto sąsaja yra taip pat labai svarbu. Tai neturėtų būti testavimo prieš pat produkto išleidimo į rinką etapas. Rekomenduojama, kad vartotojų ir jų užduočių analizė būtų atlikta kuo anksčiau, prieš priimant svarbius projekto įgyvendinimo sprendimus tam, kad analizės rezultatai galėtų būti panaudoti pirmosioms projekto idėjoms realizuoti [4].

Kuomet vartotojo ir užduočių analizė įeina į ankstyvąją gyvavimo ciklo fazę – prieš projektavimą, atsiranda galimybė sukurti nebrangius projekto modelius nesugaišus daug laiko diegiant produktą techninėje aplinkoje. Tokiu būdu yra ir galimybė užtikrinti, kad pradinės programinė įranga suteiks efektyvų bendravimą tarp vartotojo ir sąsajos objektų [4].

2 pav. struktūrizuotai pavaizduota, nuo ko priklauso sistemos apipavidalinimo sėkmė.



2 pav. Kaip sukurti gerą vartotojo sąsają?

Labai dažnai galutinio produkto sukūrimas atidedamas, patiriama daug nenumatytų išlaidų, nes projektavimo metu iš vartotojų išgirstame, kad sistema atitinka tai, "ką jie sakė", tačiau ne tai, "ką jie iš tiesų turėjo omeny". Vartotojų ir jų užduočių analizė parodo mums, kad būtinai reikia rinkti informaciją apie vartotojų poreikius stebint jį bedirbantį savo aplinkoje, o ne tik išklausant, kas jam, jo nuomone, yra reikalinga [4].

Štai tokios yra priežastys, prieštaravimai, kodėl tokia analizė nėra visuomet atliekama:

- Rinka jau pažįsta savo vartotoją.
- Kuriamas naujas produktas, todėl nėra vartotojų, kuriuos galima būtų stebėti.
- Per daug skirtingų vartotojų naudosis sistema – nėra galimybės ištirti jų visų.
- Darbų tvarkaraštyje ir taip jau trūksta laiko.
- Projekto biudžetas tam per mažas.
- Vienas iš projektuotojų grupės buvo vartotoju 25 metus.
- Turima daugybė nuostabių idėjų ir nenorima, kad vartotojas kritikuotų atliekamą darbą.
- Juk vartotojai nieko nenusimano apie projektavimą.
- Projektuoja projektuotojai, ne vartotojai.
- Patys esam vartotojai ir žinom, ko mums reikia.
- Esamą procesą bet kokiu atveju pakeisime nauju – tai ką mums gali pasakyti dabartiniai vartotojai?
- Niekada nesame atlikę vartotojų ir jų užduočių analizės, ir nežinome, kaip tai daryti.
- Vartotojus ištyrėme, ir jie pasakė, ko jie nori.

Tačiau kiekvienam iš šių prieštaravimų galima rasti kontrargumentų. Tai nėra patirtimi pagrįsti “atsikalbinėjimai”, tiesiog tai kas nauja, dažnai yra sunkiai priimama.

Vartotojų ir jų užduočių analizė apima tris aspektus: vartotojus, jų uždavinius bei aplinką, kurioje jie dirba [4].

Analizuojant vartotojus reikėtų atsakyti į tokius bendrus klausimus [4]:

- Kaip jie mąsto apie savo santykį su darbu? Ar jie profesionalai? Ar siekia karjeros? Jie administracijos ar aptarnavimo sektoriaus darbuotojai? Tiesiog stengiasi atidirbti savo aštuonias valandas? Dirba namuose? Sistema skirta darbui ar laisvalaikiui?
- Ar kuriamas produktas susijęs su pirmaeiliu darbu ar bus naudojamas tik retkarčiais? Ar jie norės sugaišti daug laiko besimokydami ar tai tik maža jų atliekamo darbo dalis?
- Ką ir kiek jie žino apie temos, kuriai projektuojate sistemą aktualumą? Ar yra jie šios temos ekspertai? Ar iš jų bus tikimasi tapti ekspertais? Galbūt jie tik vieną kartą atliks užduotį?
- Kokios patirties jie turi atliekant panašius darbus ar užduotis? Ar jau daug metų dirbą tą patį darbą? Ar tai, kam projektuojamas sistema yra visai nauja ir nepažįstama vartotojui?
- Kokius įrankius jie moka naudoti? Ar jie spausdina profesionaliai ar visiškai negrabiai? Ar juo supa naujausios technologijos ar jie jų vengia? Ar jie pripratę pildyti popierines formas ar jie koma naudotis kompiuteriu standartinėms verslo ar namų operacijoms atlikti?
- Kokios jų motyvacijos atlikti darbą? Ar atlieka užduotis už atlygį? Ar jų darbo aplinkoje daug bendravimo su žmonėmis? Ar jie bando pakilti karjeros laipteliu aukštyn?
- Kokios motyvacijos verčia panaudoti savo asmeninė laiką namuose? Koks jų mokymosi galimybių įvertinimas?
- Kokias technines žinias jie turi savo darbui atlikti? Galbūt jie įgudę technikai, kurie taiso viską, kas sugenda? Ar jie kada nors naudojosi pele, Windows taikomąja programa, operacine sistema?
- kaip gerai jie moka skaityti? Ar jie vengia rašytinės informacijos ir geriau pasiklausia kitų? Gal jie tūri universiteto diplomus ir patirtį skaitant sudėtingus tekstus? Ar jie turi noro skaityti naudodamiesi tokiais kaip jūsų produktais?

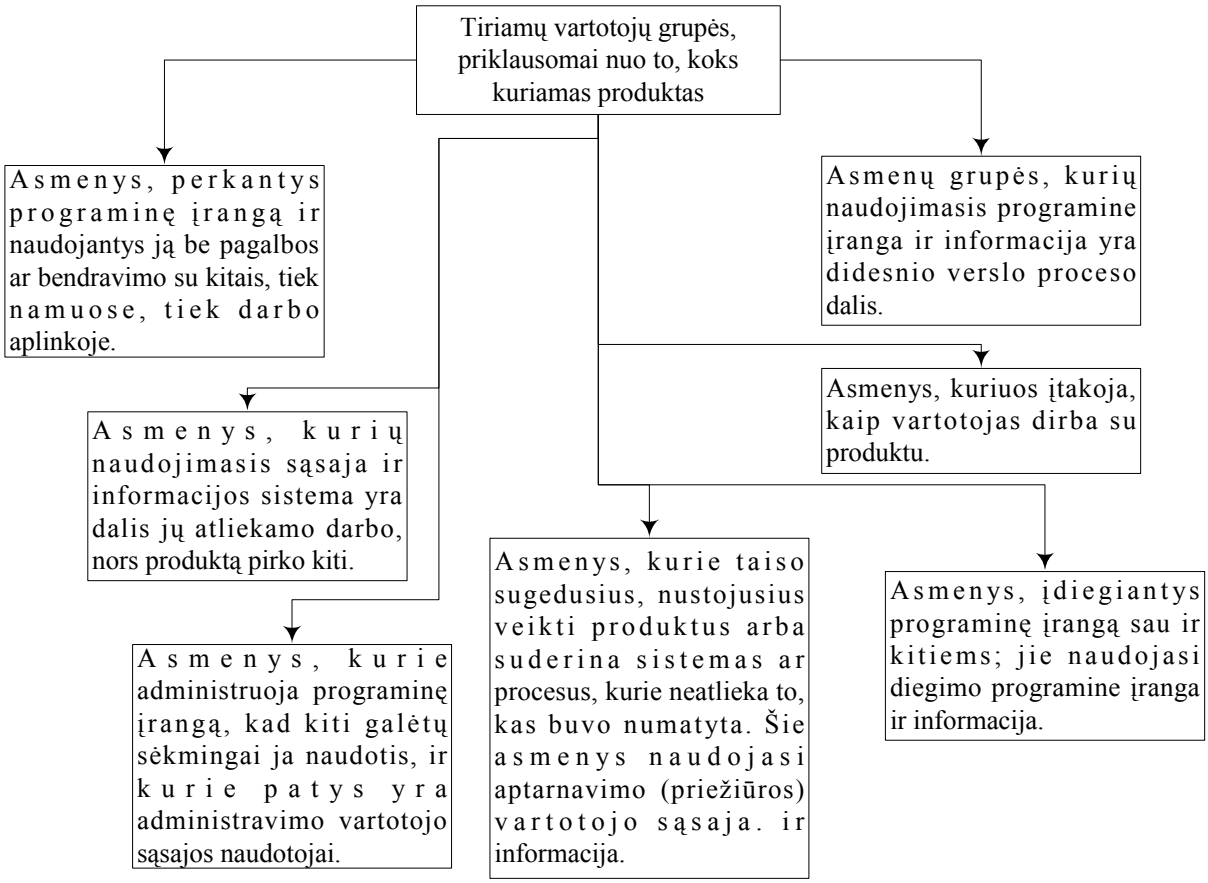
- Kokias kalbas jie gali lengvai vartoti? Ar jie laisvai kalba jūsų kalba, ar tik tokiomis, kurių jūs negalite pasiūlyti? Galbūt jie vartoja specifinį profesinį žargoną, kylantį iš atliekamo darbo aplinkos, ar žargoną tokios socialinės grupės, kuris yra susijęs su jūsų projektu?
- Ar jiems patinka rizikuoti ir išbandyti naujus būdus tam pačiam darbui atlikti? Ar jie vengia naujų išbandymų, suteikdami pirmenybę tam kad išbandyta ir gerai veikia? Ar jie mokosi žaisdami? Ar jiems reikalingas kažkas, kas paaiškintų kiekvieną žingsnį, kol jie mokysis ką nors nauja?

Tai tik bendri, abstraktūs klausimai. Kiekvienai projektuojamai sistemai tenka giliau, nuosekliau panagrinėti skirtingus klausimus. Svarbu atrinkti, kurie iš jų ypač susiję su jūsų produktų ir jūsų vartotojais. Taip pat galima išskirti ir konkretesnius, bendrus kiekvienam projektui klausimus, į kuriuos atsakymų ieškome tyrinėdami vartotojus [4]:

- Kokios asmeninės savybės gali paveikti jų darbą su programine įranga?
- Koks jų supratimas apie darbe atliekamas užduotis?
- Kokias vertybes jie įneša į darbo procesą? Ar jie entuziastingai mokosi? Ar tikisi linksmai, įdomiai bendrauti su sistema? Ar jiems svarbus taupyti pinigus, laiką, tapti ekspertu, paprastai atlikti užduotis?
- Ką jie žino apie probleminę sritį ir įrankius, kuriuos dabar naudoja arba tuos, kuriuos turėtų naudoti naujoje sąsajoje?
- Kokios patirties jie turi naudojant panašius įrankius ir vartotojo sąsajas?
- Kokios yra tikrosios jų užduotys? Ir dėl kokių priežasčių jie naudosis produktu?

Svarbu nagrinėti vartotojų poreikius, nes būtent jie nusprendžia ar naudos produktą, o ne projektuotojai ar analitikai. Svarbu kuo daugiau žinoti apie vartotoją dar ir dėlto, kad tuomet galima suprojektuoti geresnę jiems pritaikytą sistemą [4].

Didžiausias dėmesys turėtų būti teikiamas tiems žmonėms, kurie iš tiesų naudosis jūsų produktu, t.y. tiesioginiams vartotojams. Kaip vartotojai gali būti skirstomi į tiriamų vartotojų grupes, pavaizduota 3 pav.



3 pav. Tiriamų vartotojų grupės

Visi šie vartotojai gali suteikti reikiamų žinių vartotojo sąsajos ir informacinės sistemos projektavimo ir kūrimo procesui.

Visai nesvarbu, kokią įtaką projektavimo sprendimams turės sudėtingos asmeninės vartotojų savybės, reikia pripažinti, kad kuo daugiau turima duomenų sprendimams priimti, tuo sėkmingesni jie bus. Per daug dažnai projektai yra sudaromi remiantis prielaidomis apie vartotojus, prielaidomis, kurios dažnai yra nepripažintos ir nepatikrintos. Vartotojų ir užduočių analizė leidžia projektuojant pasikliauti duomenimis, o ne prielaidomis. Ar bent jau padeda tirti prielaidas ir numatyti jų įtaką projekto sprendimų priėmimui.

Sėkmingam vartotojo sąsajos sukūrimui ir dokumentacijos parašymui, reikia žinoti ne tik apie vartotojus, bet ir apie jų darbą. Ką jie daro? Kokie jų tikslai? Kaip jų siekiama? Kokius darbus padės atlikti projektuojama sistema?

Kokios problemos kyla atliekant užduotis? Ką galima supaprastinti, kad užduotis būtų galima atlikti greičiau ir lengviau? Kaip susiję vartotojų atliekami uždaviniai tarpusavyje? Kuo skiriasi vartotojai vienas nuo kito pagal savo atliekamas užduotis, arba pagal tai, kaip jie naudoja produktą, t.y. jie “naujokai”, pradedantieji ar jau ekspertai?

Norint atsakyti į šiuos ir kitus klausimus reikia atlikti vartotojų užduočių analizę. Šios analizės naudojimas kuriant sąsają, ne visada padeda pereiti tiesiogiai prie struktūrinės schemos ir tvarkingų sąrašų, kuriuos gauname sukūrę produktą. Užduočių analizė kur kas labiau netvarkinga, tačiau lemiamas etapas norint sėkmingai sukurti sistemą ir jos vartotojo sąsają.

Pirmiausiai reikėtų suprasti vartotojų tikslus, ir kaip nuo jų pereinama prie užduočių ir prie veiksmų. Vartotojų tikslai paprastai būna labai paprasti ir trumpi (pvz., naudotis kalendoriumi planuojant darbą), nes jei jie būtų sudėtingesni – juos būtų daug sunkiau pasiekti. Todėl labai svarbu, kad kompiuterines sistemas būtų paprasta naudoti, t.y. vartotojo sąsaja leistų dirbti produktyviai ir lengvai pasiekti tikslus. Sėkmingi produktai sukuriami tik suprantant tiek vartotojo, tiek kompanijos tikslus. Tikslai leidžia suprasti vartotojų vertę. Ir vienintelis būdas sukurti sėkmingą produktą – atsižvelgti į vartotojų vertę projektuojant.

Labai svarbu žinoti, kaip vartotojas pasielgs, jei jam iškils problemų atliekant užduotis. Gali pakisti jo tikslai, bet jei ir tada jis neišvengs jam neišsprendžiamų problemų – jo užduotis gali būti visai neįvykdyta, o tikslas nepasiektas. Tik vartotojas sprendžia, kada ir kaip naudoti dokumentus, programinę įrangą ir technikos priemones. Todėl reikia suprasti ir numatyti vartotojo toleranciją darbų atlikimo greičiui ir pastangoms – tai yra dalis užduočių analizės.

Kuomet užduotį atlieka keletas asmenų, labai svarbu atsižvelgti į kiekvieno jų atliekamus konkrečius darbus ir suderinti jų bendrą darbą su sistema.

Kitas užduočių analizės būdas – ištirti, ką kiekvienas asmuo konkrečioje pozicijoje padaro per dieną, savaitę ar mėnesį. Tai pareigų analizės būdas. Šis būdas padeda:

- Rasti naujų rinkos ir plėtojimo galimybių. Ką veikia žmonės, kuriems galima būtų suprojektuoti naują produktą, palengvinsiantį jų darbą?
- Suprasti specifines savybes, kurias galima įtraukti į produktą. Jei dirbant vartotojai yra dažnai pertraukiami, reikėtų numatyti sistemos galimybes, leidžiančias lengvai nutraukti darbą ir vėl pradėti dirbti su produktu..
- Sužinoti, kas juos slegia, ir ką jie vertina.

Atliekant tokio tipo analizę, reikėtų pasidomėti ne tik tuo, ką žmonės daro atlikdami užduotis, bet ir faktoriais, pateiktais 2 lentelėje.

2 lentelė. Pareigų analizėj naudojami faktoriai.

Dažnumas	Kaip dažnai atliekama kiekviena užduotis?
Svarbumas	Kiek įtakos kiekviena užduotis turi jų atliekamam darbui?
Sudėtingumas	Kaip sudėtinga yra atlikti konkrečią užduotį?
Atsakomybės pasidalijimas	Ar visi asmenys tokiose pareigose atlieka tą pačią užduotį ar iš viso atlieka skirtingas užduotis?

Atliekant bet kokio tipo užduočių analizę, reikėtų stebėti, klausytis ir kalbėtis su keliais vartotojais, turinčiais skirtingą patirtį ir atsakomybę tam, kad būtų galima susidaryti kuo geresnį dominančio darbo vaizdą.

Reikia nepamiršti, kad vartotojas – tai ne projektuotojas. Jų asmeninės savybės gali versti elgtis visiškai skirtingai, kitaip reaguoti į tas pačias situacijas. Negalima tapatinti savęs su vartotoju.

Siekiant sužinoti kuo daugiau apie vartotoją ir jo atliekamas užduotis, reikia nepamiršti atsižvelgti į fizinę, socialinę ir kultūrinę aplinką, kurioje jie dirba. Kuomet vartotojai stebimi ir tiriami jų aplinkoje, ši būtinai turės įtakos susidaromai nuomonei. Tuomet reikėtų išsiaiškinti, kurie aplinkos aspektai labiausiai veiks projektavimo savybes.

Analizuoti vartotojų aplinką svarbu todėl, kad jie nedirba izoliuotai nuo aplinkos. Juos veikia veikla vykstanti aplink, fizinė darbo vietos aplinka, įranga, kurią naudoja, darbo santykiai su kitais individais. Jei kuriamas produktas bus netinkamas konkrečioje aplinkoje, jį gali būti sudėtinga naudoti. Gali atsitikti netgi taip, kad vartotojas kategoriškai atmes produktą ir grįš prie senųjų, įprastų darbo metodų, arba ras būdų kaip sunaikinti jiems primestą sistemą.

3 lentelėje pateikti klausimai, pagal kuriuos reiktų atlikti tyrimą atskirais aplinkos aspektais.

3 lentelė. Tyrimo aplinkos aspektu pagrindiniai klausimai.

Fizinės aplinkos tyrimo aspektai:	**Socialinės aplinkos tyrimo aspektai:**	**Kultūrinės aplinkos tyrimo aspektai:**
Kiek vartotojai turi erdvės?	Ar vartotojų aplinkoje būtina užduotis atlikti tiksliai ir greitai?	Ar vartotojų nacionalinė kultūra turės įtakos jų darbui?
Ar turi savo asmeninę įrangą, ar dirba su skolinta?	Ar aplinkui yra informacijos šaltinių, kurie padėtų atsakyti į kylančius klausimus ir spręsti problemas?	Ar vartotojai dirba nutolę vieni nuo kitų? Ar skirtinguose rajonuose yra kultūrinių skirtumų?
Ar jiems reikia ieškoti informacijos toliau nuo savo darbo vietos?	Ar asmenys, besidalinantys informacija dirba toje pačioje vietoje, šalia vienas kito?	Ar vartotojai priklauso specifinei profesinei kultūrai, kuri turi ypatingų bruožų galinčių paveikti kuriamo produkto savybes?
Ar darbo aplinka triukšminga?	Kaip vartotojai dirba kartu ir dalinasi informacija?	Ar vartotojai priklauso socialinėms ekonominėms grupėms, kurios galėtų paveikti jų santykį su produktu? Galbūt kai kurios metaforos jiems bus nepriimtinos?

Ar pakankamai apšviesta darbo vieta?	Kokia socialinė hierarchija organizacijoje? Kaip vyresnieji bendrauja su žemesnio lygio darbuotojais?	
Ar nešvara, dulkės ir kt. trukdo dirbti?	Ar siejasi tarpusavyje fizinė ir socialinė aplinka? Ar dirba bendroje erdvėje ar visi savo kabinetuose? Ar daug bendrauja tarpusavyje dirbdami?	
Ar slėgis, drėgmė, temperatūra veikia vartotojų gebėjimą naudotis įranga?	Ar daug vartotojų dirba namuose? Ar jie dirba vieni?	
Ar lengvai prieinama elektros srovė, kad būtų galima prijungti visą reikalingą aparatūrą?	Kokie santykiai tarp produkto vartotojų ir jų klientų? Kokiu būdu jie bendrauja tarpusavyje?	

Baigus rinkti visą informaciją apie vartotojus, jų užduotis ir supančią aplinką, reikia ją struktūrizuoti, išanalizuoti ir nuspręsti kokios įtakos ji turės kuriamos sistemos vartotojo sąsajai, jos specifinėms savybėms. Tuomet ateina laikas priimti svarbius projektavimo sprendimus ir galima pradėti kūrimo procesą. [4].

2.2.2. Iteracinis prototipo kūrimas, pagrįstas integruota funkcinių ir vartotojo sąsajos reikalavimų specifikacija

Vienas iš metodų tiksliam reikalavimų vartotojo sąsajai suvokimui yra metodas SCORES [6]. Šio metodo išplėtimo ir sustiprinimo privalumas remiantis [6], susideda iš trijų dalių. Pirmiausia tai, kad vartotojo sąsajos reikalavimų specifikavimas yra pagrįstas UML ir integruotas į funkcinių reikalavimų specifikaciją. Antra, galimybė pusiau automatiškai sugeneruoti prototipą validacijai atlikti. Trečia, modeliu pagrįstas prototipo generavimas leidžia generuoti prototipą taip, kad rankiniu būdu atliekami prototipo pakeitimai valiavimo metu būtų automatiškai perkelti į reikalavimų specifikaciją [6].

Programinės įrangos reikalavimų specifikacija ypač svarbi moderniuose elektroninio verslo srityse. Joje turėtų būti ypatingai efektyviai atsižvelgiama į vartotojo sąsajos reikalavimus [6].

Yra svarbu, kad teorinio lygio ir modeliavimo požiūris į funkcinių ir vartotojo sąsajos reikalavimų specifikacijas būtų kuo panašesnis, kad juos būtų galima sukomponuoti į

integruotą visa apimančią specifikaciją. Tokia specifikacija turi palaikyti validaciją, verifikaciją ir testavimo procesą [6] Šiems tikslams pasiekti autoriai praplečia SCORES metodą [7] FLUID metodo [8] elementais.

Reikalavimų informacinėms sistemoms inžinerijoje programinės įrangos (PI) reikalavimų valiavimui naudojami tiriamieji-pristatomieji arba funkciniai prototipai [10,11]. Pristatomieji prototipai atspindi, kaip taikomoji programa pritaiko gautus reikalavimus vartotojo požiūriu, o funkciniai prototipai įgyvendina svarbias vartotojo sąsajos ir taikomosios programos funkcionalumo dalis.

Kad nekiltų klausimas, kuo remtis, kai prototipas ir programinės įrangos reikalavimų specifikacija nesutampa, autoriai [6] išplečia SCORES metodą [7] pasinaudodami FLUID metodo [8] vartotojo sąsajos modeliavimo elementais suderinamais su UML. Gaunamas Integruotas metodas SCORES+ [6] leidžia ne tik suderinti funkcinių ir vartotojo sąsajos reikalavimų specifikacijas, bet FLUID dėka ir pigiai, pusiau automatiškai konstruoti tiriamuosius, pristatomuosius prototipus įvairiems įrenginiams, platformoms bei sąveikos tipams.

Funkcinių reikalavimų etapo pagrindiniai ir svarbiausi SCORES metodo modeliavimo elementai:

1. Aktoriai (actors), panaudojimo atvejai (use cases) ir veiklos diagramos (activity graphs). Aktoriai charakterizuoja roles atliekama išorinių objektų, kurie sąveikauja su sistema, kaip dalimi susijusio veiksmo vieneto. Panaudojimo atvejai apibrėžia aukšto lygio uždavinius, kurie turi būti susiję su jame dalyvaujančių aktorių tikslu. Veiklos diagramos pagal UML – tai procesų, apimančių vieną ir daugiau klasifikatorių, specifikacija [12].

SCORES metode veiklos (actions) apibrėžiamos trimis tipais: kontekstinis veiksmas, sąveika; makro-veikla (santykiai tarp panaudojimo atvejų); aktorius veikloje (nežinomas aktorius) [6].

2. Srities klasių modelis (pagal [9]) apima svarbiausiu objektų tipus, kurie atvaizduoja detales esančias arba įvykius įvykstančius sistemos darbo kontekste.

Vartotojo sąsajos reikalavimų etape FLUID metodas naudoja tik pačius svarbiausius sprendimus, susijusius su vartotojo sąsaja ir nenurodo (neapibrėžia) nei konkretaus sąveikavimo būdo, nei detalaus grafinės vartotojo sąsajos projekto. Pirmiausia reikia patobulinti FLUID metodo elementai, kad jie atitiktų SCORES metodo abstrakcijos lygios

elementus ir padaryti juos suderinamus su UML. Po to, patobulintus FLUID elementus integruoti į SCORES, pritaikant juos panaudojimo atvejams ir veiklos diagramoms [6].

Statinę vartotojo sąsajos analizės (VSA) struktūrą sąlygoja scenos (scenes) kartu su sąryšiais ir klasių vaizdais (class view). Klasės vaizdas – tai abstrakti atskirų atvejų ir sąryšių iš srities susijusioje vartotojo veikloje pateiktis. Scena be klasės vaizdo neatspindi srities objektų informacijos ir dažnai yra tik pagalba navigacijoje [6].

VSA dinamiką nusako vartotojo operacijos. SCORES+ metode scenose gali vykti dviejų tipų vartotojo operacijos: navigacija (iš vienos scenos į kitą; gali būti pažymėta kaip scenos sukūrimas arba pašalinimas) ir manipuliacija (pažymi pasirinkimą, redagavimą tų srities klasių atskirų atvejų, kurie yra kuriame nors scenos klasės vaizde [6].

Į tokius vartotojo veiksmus kaip lango perkėlimas ar jo išmatavimų keitimas ar slankiosios juostos slankiojimas, nėra kreipiama dėmesio. Taip pat ir į žemo lygio sąveikas kaip pelės spragtelėjimas ar paprastas redagavimas [6].

Vartotojo sąsajos analizės elementai gali būti modeliuojami kaip klasių diagramos elementai, ir įtvirtinti į veiklos diagramas kaip objektų sekos būsenos [12].

Prototipo generavimo etape pagrindinis SCORES ir VSA elementų sujungimo tikslas – integruota funkcinių ir vartotojo sąsajos reikalavimų specifikacija. Srities objekto kintamasis (SOK) – tai VSA klasė, kuri nurodo vieną srities objektą arba rinkinį srities objektų, kurių savybės yra pateiktos klasės vaizde susijusiame su SOK [6].

Srities objektų kintamieji gali būti naudojami kaip parametrų tipai. remiantis UML, kiekvienas manipuliacijos (SOK) parametras gali būti charakterizuotas kaip „į“ (read – skaityti), „iš“ (create – sukurti) arba „į-iš“ (update – atnaujinti) operacijos tų srities objektų, kuriuos nurodo SOK. Srities objektų kintamieji taip pat gali būti susiję vienas su kitu [6].

Prototipų generavimas iš SCORES+ metodo programinės įrangos reikalavimų specifikacijos ir srities objektų kintamųjų vykdomas dviem žingsniais. Pirmuoju programinės įrangos reikalavimų specifikacija ir srities objektų kintamieji automatiškai transformuojami į detalų vartotojo sąsajos projekto modelį. Antruoju iš pastarojo automatiškai generuojamas vykdomasis prototipas reikalingai platformai [6].

Validavimo sesija apima rinkinį vykdomų veiklos scenarijų, kuriuos galima įvykdyti prototipo pagalba. Valiavimo metu prototipas gali būti modifikuotas. Jei jis modifikuojamas, prototipo generavimo įrankis perduoda pakeitimus atgal vartotojo sąsajos projekto modeliui [6].

2.2.3. Reikalavimų analizė ir prototipo kūrimas naudojant scenarijus ir būsenų diagramas

Scenarijai yra geras pagrindas bendravimui su klientais, kurie neturi daug techninių žinių. Jie taip pat palaiko validaciją ankstyvose reikalavimų įgyvendinimo fazėse su nedideliais nukrypimais. Sistemų kūrimas remiantis scenarijais yra gana sudėtingas ir sukelia nemažai neaiškumų, todėl yra gana mažai išvystytas [13].

Reikalavimų modeliavimas generuojant scenarijus iš būsenų diagramų, leidžia išvengti daugelio problemų ir dar turi keletą privalumų [13]:

- Scenarijus ir būsenų mechanizmus galima bet kada vykdyti, o tai leidžia stebėti animuotą vykdymą su sistemos srities objekto struktūros dinaminiais vaizdais.
- Neformalių ir formalių veiksmų specifikavimo elementai sukuria ekspresyvią ir nedviprasmišką veiklos semantiką.
- Šis požiūris įgalina atlikti vientisą perėjimą į pilnai funkcionuojantį prototipą, kad galutinis vartotojas galėtų jį valiuoti, įvertinti.
- Tiksliai palaiko reikalavimų pasikeitimą.

Naudojant scenarijus, tam, kad juos būtų lengviau suprasti, modeliuojama: sąveikavimas (tarp vartotojo ir sistemos), vidiniai sistemos veiksmai (ne techniniai, bet susiję su sritimi) ir kontekstiniai veiksmai atliekami už sistemos ribų. Tiksliai modeliuojama ir vartotojo įtaka naudojant sistemos duomenis [13].

Scenarijų naudojimo tikslas – modeliuoti sistemos reikalavimus kaip pagrindą prototipui generuoti ir įgyvendinti (diegti). Jie taip pat naudojami trejopai validacijai: analitikų, testuotojų ir galutinių vartotojų. Scenarijų semantikai apibūdinti naudojami neformalūs ir formalūs apibrėžimai [13].

Tam, kad nereikėtų įgyvendinti visų galimų scenarijų, kurių sistemoje egzistuoja gana daug, modeliuojama būsenų diagramų diagramą kiekvienam panaudojimo atvejui, kuri gali sugeneruoti visus scenarijus tam panaudojimo atvejui [13].

Scenarijus rašomas papildant panaudojimo atvejo, kuriam priklauso scenarijus, būsenų mechanizmą keletu būsenų ir būsenų perėjimų, taip kad reikiamas scenarijus tampa vykdomu kaip kelias per būsenų diagramų diagramą [13].

Norint sugeneruoti scenarijų iš būsenų diagramos, reikalinga tokia papildoma informacija[13]:

- kokie vartotojo veiksmai pakeičia būseną, t.y. kaip paplitę su tuo susiję įvykiai;

- kokie vartotojo veiksmai įveda naujus duomenis;
- objektų aibę, kuri nusako visos sistemos būseną prieš vykdant scenarijų.

Reikalavimų pasikeitimo problemos sprendimas yra vienas sudėtingiausių, tačiau ypač svarbių išbandymų visuose reikalavimų modeliavimo metoduose, ypatingai tuose, kurie naudoja scenarijus reikalavimams užfiksuoti. Pasikeitimai gali būti dvejopi: reikia pridėti naujų funkcionavimo galimybių arba panaikinti esamas [13].

Po kiekvieno būsenų modelio pakeitimo, scenarijai yra generuojami iš naujo. Tačiau padaryti pakeitimai gali sukelti problemas, į kurias dėmesį turi atkreipti sistemos analitikas [13]:

- Netinkami būsenų modelio pakeitimai gali padaryti nepageidaujamų pakeitimų kai kuriems scenarijams. Tam, kad būtų galima tai išsiaiškinti, reikia nedelsiant vizualizuoti scenarijus. Tuomet analitikas gali pasirinkti: arba toliau vykdyti pakeitimus, arba grąžinti buvusią būsenų modelio būseną.
- Jei pakeitimo metu įvedamos būsenos, reikalaujančios vartotojo informacijos įvedimo, analitikas gali pasirinkti: arba įvesti skirtingas reikšmes kiekvienam scenarijui, arba naudoti iš anksto numatytas reikšmes.
- Būsenų modelio pakeitimai gali priversti scenarijų eiti kita būsenų diagramos šaka nei anksčiau, tuo pakeisdami ir kitus susijusius scenarijus. Tokiu atveju analitikas turi: arba įvesti reikalingas reikšmes, arba leisti sistemai naudoti iš anksto nustatyta reikšmes.

Naudojamos trijų tipų scenarijų pakeitimo vėliavėlės: rezultato pasikeitimas, kelio pasikeitimas ir vartotojo įvesties pasikeitimas. Jei analitikas nenori kiekvieną kartą rūpintis įvedamomis reikšmėmis, jis gali sistemai leisti visada naudoti iš anksto numatytas reikšmes [13].

Skelbiamosios specifikacijos (*declarative specifications*) yra labai tinkamos testavimui ir tarifikavimui, nes tiek išankstinės, tiek vėlesnės sąlygos gali būti testuojamos ir naudojamos tam, kad įrodytų klaidų nebuvimą. Operacijų specifikacijos (*operational specification*) naudoja sakinius (*statements*) arba pranešimus (*messages*) tiksliam semantikos aprašymui [13].

Ne visuomet automatinis darbas yra tinkamas, norint sukurti gerą projektą. Remiantis [7] naudojami panaudojimo atvejai kaip dalinimo (skirstymo) kriterijus būsenų diagramoms. „Include“ (apimti) ir „Extend“ (praplėsti) priklausomybės tarp panaudojimo atvejų pateikia priklausomybės ryšius tarp susijusių būsenų diagramų [13].

Sub-scenarijus yra scenarijaus dalis, kuri realizuojamas viena būsenos diagrama. Sub-scenarijaus vykdymas transformuoja sistemą iš vienos taikomosios programos būsenos kitą. Taikomosios programos būsenos ir perėjimai atliekami sub-scenarijų iš taikomosios programos būsenų diagramos aprašančios galimus sistemos panaudojimo atvejus. Pilnas sistemos testavimas gali būti apibrėžtas visų kelių per taikomosios programos būsenų diagramą rinkiniu [13].

Čia iš FLUID požiūrio yra panaudotas aukšto lygio klasių karkasas vartotojo sąsajos specifikacijai, kas leidžia automatiškai generuoti prototipą kiekvienai reikalingai platformai.

2.2.4. KIS funkcionalumo rezultatų struktūros specifikavimas

KIS funkcionalumo rezultatai (FR) – tai vartotojo reikalavimu kuriamosios KIS išvedamos ataskaitos, rodikliai ir pan. Jų struktūra apibrėžia organizacijos veiklos dalį, kuri analizuojama: analizuojami ir įtraukiami į reikalavimų specifikaciją tik tie procesai, kurių metu suformuojami duomenys, kurių reikia išvedant KIS funkcionalumo rezultatų reikšmes.

Rezultato struktūra aprašoma bent vienu atributu. Rezultatų specifikavimas leidžia numatyti, kokius duomenis turės pateikti KIS. Rezultato modelis yra analogiškas dokumentų formų modeliui, bet papildytas naujais elementais, ir dokumentų formų pavyzdžiai jo formavimui neįvedinėjami. Toks modelis yra dviejų sluoksnių: skirtų vartotojui ir projektuotojui. Tuo būdu vartotojas gali konkrečiai apibrėžti sistemos trūkumus, o projektuotojui lengva jį suprasti ir padaryti pakeitimus.

Atributas – tai organizacijos vieneto savybė. Ji gali būti kiekybinė, kokybinė, klasifikuojanti, identifikuojanti. Jei jis įtrauktas bent į vieną rezultatą ar duomenų šaltinį – jis įtraukiamas ir į specifikaciją.

Atributų grupė – tai atributai, aprašantys tą patį objektą, bet ne duomenų šaltinį ar rezultatus. Tas pats atributas gali būti keliose grupėse. Apibūdinti objektą galima kitu objektu.

Rezultato atributas – aprašo rezultatą arba yra į jį įtrauktas.

Rezultato tipas – kiekvienam rezultatui specifikuotas tik vienas tipas – rezultato struktūros elementas. Ir tipas gali identifikuoti tik vieną rezultatą.

Rezultato potipis – kiekvienam rezultato tipui gali būti specifikuoti keli potipiai – rezultato struktūros elementai. Potipis gali identifikuoti tik vieną rezultatą.

Ribojantis elementas – tai atributo reikšmė, ribojanti rezultato atributų reikšmes. Rezultatui gali būti keli ribojantys elementai.

Neklasifikuotas elementas – bet kuris rezultato struktūros elementas, bet nei vienas iš anksčiau išvardintų. Rezultatui gali būti specifikuoti keli tokie elementai.

Rezultato sąlyga – rezultato formavimo sąlyga. Ji privaloma, kai specifikuotas bent vienas ribojantis elementas.

Sąlygos elementas – parodo, koks rezultato elementas įtakoja kitą rezultato elementą [1].

Komponento pavadinimas	**Grafinis simbolis**
Atributas	<kodas>.<pavadinimas>
Atributų grupė	<kodas>.<pavadinimas>
Atributų reikšmes ribojantis elementas	<kodas>.<pavadinimas>
Elementas, identifikuojantis rezultato/duomenų šaltinio tipą	<kodas>.<pavadinimas>
Neklasifikuotas elementas	<kodas>.<pavadinimas>
Sąlyga	Nėra, tekstas užrašomas prie rezultato ar duomenų šaltinio

[1]

Sąryšio pavadinimas	**Grafinis simbolis**
Ne kartotinio atributo/grupės komponavimo į grupę ryšys	⟶
Kartotinio atributo/grupės komponavimo į grupę ryšys	⟶⟶
Atributo reikšmės įtakojimo ryšys	⟶

[1]

KIS funkcionalumo rezultato specifikavimo pavyzdys pateikiamas priede (Pav. 1).

2.2.5. KIS funkcionalumo rezultato struktūros specifikavimo procesas

Rezultatų, kuriuos suformuos KIS, struktūros specifikavimas. Tai atliekamas nustačius, kokius rezultatus KIS turės formuoti.

Pirmiausiai specifikuojami rezultato struktūros elementai. Turint bent vieną elementą galima: 1)specifikuoti sekantį elementą; 2)klasifikuoti elementą; 3)pašalinti specifikuotą elementą. Jei yra specifikuoti du atributai arba vienas atributas ir vienas ribojantis elementas galima specifikuoti *sąlygą*, kuri įtakoja rezultato formavimą. 4)Turint sąlygą, ją galima koreguoti. 5)Jei pašalinamas vienintelis rezultato struktūrą aprašantis elementas, tai galima tik specifikuoti kitą [1].

2.2.6. Duomenų šaltinių struktūros specifikavimas

Specifikavus KIS funkcionalumo rezultatus dar neturime visų KIS kūrimo procese reikalingų funkcinių reikalavimų, tačiau juos galima išgauti remiantis turimais.

Kuriant funkcionalumo rezultatą, turi būti nurodyta, iš kur ir kokius suformuotus duomenis jis naudos, t.y. turi būti nurodyti duomenų šaltiniai. Duomenų šaltiniai (DŠ) – tai informaciją apie organizacijos veiklą saugantys objektai. Tai įvairūs dokumentai, pranešimai, elektroninės formos ir duomenų paketai.

Duomenų šaltinių ir jų struktūrai specifikuoti naudojamas tas pats metodas kaip ir specifikuojant funkcionalumo rezultatus, t.y. rezultatų/duomenų šaltinių struktūros modelis.

Kiekvienam neišvestiniam KIS FR atributui turi būti specifikuotas bent vienas duomenų šaltinio atributas, naudojamas šio rezultato atributo reikšmėms formuoti. Kiekvieno DŠ bent vienas atributas turi būti naudojamas bent vieno KIS FR atributo reikšmėms formuoti. Analizuojami ir specifikuojami tik tie procesai, kurie atsispindi specifikuotuose duomenų šaltiniuose.

Ar formuojant KIS FR reikalingas specifikuotas duomenų šaltinis pagrindžia duomenų šaltinio atributų ryšys su rezultato atributų formavimų, nurodant tai specifikuotiems srautams. Duomenų šaltinio struktūra yra analogiška rezultato struktūrai (tie patys elementai). Jeigu rezultato atributo reikšmė neišvedama iš kitų atributų, tai jis turi būti susietas su DŠ atributu.

Duomenų šaltinio specifikavimo procesas yra analogiškas KIS FR struktūros specifikavimo procesui.

2.2.7. Duomenų šaltinių apdorojimo etapų specifikavimas

Duomenų šaltinių apdorojimo procesų specifikavimo tikslas – reikalavimų sistemos dinamikai išgavimas. Jie priklauso nuo kompiuterizuodamos organizacijos veiklos, t.y. kokias užduotis atlieka organizacijos darbuotojai. Duomenų šaltiniai atspindi šių užduočių atlikimo rezultatus, todėl analizuojant užduotis, siekiant sužinoti apie organizacijoje vykstančius procesus, reiktų naudoti duomenų šaltinių specifikaciją.

Išsiaiškinus, koks specifikuotas dokumentas (DŠ), kiek laiko ir kieno yra pildomas, sužinomi ir kompiuterizuotini organizacijos veiksmai. Taip pat, į veiksmą įtrauktų objektų būsenos, sąlygos perėjimui iš vienos būsenos į kitą, sąlygos tenkinamos įvykdžius veiksmą.

Duomenų šaltinių apdorojimo etapų specifikavimo proceso metu specifikuojama duomenų šaltinio apdorojimo technologija ir jos gyvavimo ciklas. Organizacijoje procesai vyksta tiek nuosekliai, tiek lygiagrečiai, todėl duomenų šaltiniai, didele dalim atspindintys organizacijoje vykstančia veiklą, taip pat pildomi ir nuosekliai, ir lygiagrečiai. Veiksmus organizacijoje atlieka skirtingi aktoriai, reiškia DŠ pildo skirtingi aktoriai. DŠ gali būti pildomas keliais etapais, skirtingų aktorių. Specifikuojant DŠ apdorojimo etapus, reikia turėti DŠ struktūros specifikaciją.

Duomenų šaltinio specifikavimo etapo metu užpildomi duomenų šaltinio egzempliorių aprašantys atributai (dalis arba visi).

Perėjimas tarp duomenų šaltinio apdorojimo etapų parodo, agento atlikto veiksmo ir jo rezultatų perdavimo gavėjui įtaką DŠ apdorojimo etapų pasikeitimą [1].

2.2.8. Duomenų šaltinių apdorojimo etapų specifikavimo procesas

Duomenų šaltinių užpildymo etapus ir perėjimus tarp jų galima specifikuoti, kai žinoma, kokius DŠ turės formuoti kuriama KIS. Toliau išvardinti veiksmai, kuriuos reikia atlikti, specifikuojant DŠ apdorojimo etapus ir perėjimus tarp jų.

1)Specifikuoti naują dokumentų šaltinio apdorojimo etapą: iš pradžių jis nėra susietas su jokiu DŠ apdorojančiu veiksmu; parenkamas atributas, kuris bus įvedamas etape (gali būti dar neįtrauktas atributas arba įtrauktas į tokį etapą, iš kurio nepereinama į specifikuojamą); sukuriamas ryšys tarp naujo DŠ ir to atributo.

2)Pašalinti specifikuotą duomenų šaltinio apdorojimo etapą. Šalinti etapą galima, jei jis klasifikuotas, kaip nesusijęs su nei vienu DŠ apdorojančiu veiksmu. Jei susijęs – pirmiausiai reikia atsieti nuo jį apdorojančių veiksmų.

Žingsniai: Pašalinti DŠ apdorojimo etapą; pašalinti sąryšį tarp etapo ir atributo; jei tame etape apdorojamas atributas: a)neįtrauktas į kitą etapą – klasifikuoti jį kaip neįtrauktą; b)jei įtrauktas bent į vieną – kaip įtrauktą.

3)Papildyti atributu galima tokį specifikuotą duomenų šaltinio apdorojimo etapą, kuris dar nėra susietas su jokia DŠ būsena.

Žingsniai: pirmiausiai pasirenkamas DŠ apdorojimo etapas, paskui DŠ atributas (neįtrauktą į jokį etapą arba įtrauktas į alternatyvų), kuris bus pildomas (apdorojamas) tame etape. Po to sukuriamas sąryšis tarp to DŠ apdorojimo etapo ir atributo.

4)Pašalinti atributą iš specifikuoto duomenų šaltinio apdorojimo etapo galima tik tuomet, kai pastarasis specifikuojamas kaip nesusietas su jokia DŠ būsena. Priešingu atveju reikia etapą atsieti nuo visų būsenų.

Žingsniai: Pasirenkamas su DŠ būsenomis nesusijęs DŠ apdorojimo etapas, ir pašalinamas ryšys tarp to etapo ir pasirinkto atributo. Paskui reikia klasifikuoti tą atributą: kaip neįtrauktą į jokį kitą DŠ apdorojimo etapą, arba kaip įtrauktą kitą DŠ apdorojimo etapą.

5)Specifikuoti perėjimą tarp duomenų šaltinio apdorojimo etapų.

Specifikavimas atliekamas tarp DŠ apdorojimo etapų, nesusietų su nei viena DŠ būsena. Kuomet etapai susieti su bent viena DŠ būsena – reikia atsieti nuo visų būsenų.

Žingsniai: tais atvejais, kai veiksmai, atliekantys specifikuojama perėjimą nėra kuriantys ar pašalinantys, reikia parinkti etapą, kuris bus atitinkamai esamasis ar sekantysis. Parinkti veiksmo agentą ir gavėją, kuriais gali tapti aktorius (nesusietus su nei vienu arba susietus su bent vienu veiksmu). Specifikuoti agento atliekamą veiksmą, kuris atlieka specifikuojamą perėjimą (naujas veiksmas arba jau specifikuotas, atliekamas pasirinkto agento). Specifikuoti atitinkamo tipo perėjimą tarp DŠ apdorojimo etapų: esamasis apdorojimo etapas nepasirinktas – perėjimą atliekantis veiksmas bus kuriantysis, nepasirinktas sekantysis – bus pašalinantis;

jei pasirinktas esamasis ir sekantysis etapai sutampa, perėjima atliekantis bus pakeičiantis; jei sutampa – bus modifikuojantis.

6)Pašalinti perėjimą tarp duomenų šaltinių apdorojimo etapų galima tik tuomet, kai jie nesusieti su jokia DŠ būsena. Kai yra susieti, reikia pirmiausiai atsieti nuo tų būsenų.

Žingsniai: pirmiausiai pašalinamas pasirinkto tipo perėjimas tarp DŠ apdorojimo etapų. Tai gali būti perėjimas, kurį atliekantis veiksmas yra kuriantis, pašalinantis, pakeičiantis arba modifikuojantis. Agento veiksmą, jei jis neatlieka jokių kitų perėjimų reikia pašalinti iš specifikacijos, jei atlieka – palikti. Po to, klasifikuoti veiksmo agentą: kaip nesusietą arba kaip susietą su veiksmu. Klasifikuoti aktorių – veiksmo gavėją: kaip nesusietą su jokių veiksmu arba kaip susietą su veiksmu. Jei veiksmas, atlikęs pašalinantį perėjimą nebuvo kuriantysis, tai peržiūrėti pašalintojo perėjimo esamąjį etapą: klasifikuoti kaip neįtrauktą arba kaip įtrauktą į kitą perėjimą. Jei veiksmas, atlikęs pašalinantį perėjimą nebuvo pašalinantysis, tai peržiūrėti pašalintojo perėjimo sekantįjį etapą: klasifikuoti kaip neįtrauktą arba kaip įtrauktą į kitą perėjimą [1].

2.3. Projekto tikslas, jo pagrindimas ir kokybės kriterijų apibrėžimas

Darbo tikslas – pateikti programiškai realizuotą vartotojo sąsajos elementų modeliavimo algoritmą informacijos srautų specifikacijos pagrindu. Tai bus dalis CASE priemonės, skirtos KIS projektavimui. Šį modulį planuojama realizuoti su *MS Visual Basic for Applications* programavimo kalba, aplinka ir elementai paimti iš *MS Visio 2000*.

Realizuotas modulis turės galimybę pasinaudoti anksčiau realizuotų modulių rezultatais, saugomais funkcinių reikalavimų specifikavimo saugykloj, ir išsaugoti ten savo rezultatus. Pagal naujus rezultatus modulis turės sukurti vartotojo sąsajos elementus, turinčius dalinį funkcionalumą, t.y. vartotojas (testuotojas, užsakovas) galės matyti formų kaitą ekrane, įvesti duomenis, gauti atsakymus, rezultatus.

Šio darbo užsakovas yra Informacijos sistemų projektavimo mokslinė grupė, nagrinėjanti informacijos sistemų projektavimą plačiame reikalavimų inžinerijos, procesų ir duomenų struktūrų specifikavimo kontekste.

Užbaigtą CASE priemonę numatoma panaudoti mokymo tikslams Kauno Technologijos Universiteto Informacijos sistemų katedroje, gilinti žinias KIS keliamų funkcinių reikalavimų inžinerijoje, bei KIS projektavimui.

Siekiant užsibrėžto tikslo bus įgyvendinami šie uždaviniai:

1. Atlikta funkcinių kompiuterizuojamos IS reikalavimų – informacinių srautų specifikacijos – analizė;
2. *MS Visual Basic for Applications* pagalba įsisavintos *MS Visio 2000* praplėtimo galimybės;
3. Sukurtas modelis, generuojantis vartotojo sąsajos elementus.

Įgyvendinto produkto kokybę numatyta įvertinti pagal šiuos kriterijus:

1. Išbaigtumas – sistema turi atitikti sudarytą specifikaciją;
2. Tolerancija klaidoms – vartotojui įvedus duomenis, sistema turi patikrinti, ar jie teisingi ir juos ištaisyti pati arba duoti pataisyti vartotojui;
3. Modulis turi būti “draugiškas” vartotojui, t.y. būti patogus ir lengvai suprantamas;
4. Efektyvumas – reikalingų kompiuterio resursų, t.y. programinės įrangos aprašymas;
5. Panaudojamumas – kaip interpretuojami įvedami duomenys, ar juos įvedant nesusidaro dviprasmiškos situacijos, ar pasirinktos operacijos atliekamos tiksliai, ar gaunami norimi rezultatai.
6. Integralumas – ar sistema gali būti sujungta su kitais moduliais.

3. Reikalavimų sistemai vartotojo sąsajai modeliuoti specifikacija ir sistemos projektas

3.1. Projekto techninė užduotis

Suderinta:.....

Vadovas: dr. doc. R. Butleris

Data: 2003–02–10

Tema:

Vartotojo sąsajos elementų modeliavimas informacijos srautų specifikacijos pagrindu.

Užsakovas:

Informacijos sistemų projektavimo mokslo grupė.

Reikalavimai projektui, realizacijai ir naudojimo dokumentams:

1) sudaryti vartotojo sąsajos modeliavimo algoritmą;
2) sudaryti programinės modulio realizacijos projektą;
3) pasirinkti ir paruošti testinius duomenis;
4) atlikti programinę modulio realizaciją ir testavimą, naudojant aprašytus testinius duomenis.
5) paruošti sistemos naudojimo instrukciją;
6) paruošti sistemos palaikymo instrukciją.

Reikalavimai projektavimui, programinei ir techninei įrangai:

1) programinės modulio realizacijos projektui numatoma naudoti *Rational Rose* programinį paketą, taikant *RUP*;
2) modulis bus realizuotas su *Visual Basic for Applications* programavimo kalba, grafinei vartotojo sąsajai naudojant *MS Visio 2000* įrankius.
3) minimalūs reikalavimai, užtikrinantys sėkmingą sistemos funkcionavimą, programinei ir techninei įrangai tūri būti šie:
   a. operacinė sistema – Windows 98 ar naujesnė;
   b. programiniai paketai: *MS Visio 2000* ir *MS Access* iš *MS Office* programinio paketo;
   c. procesoriaus greitis – 350MHz, operatyvinė atmintis (RAM) – 96MB, laisvos vietos kietajame diske – 500MB.

IFM–8/4 gr. stud.

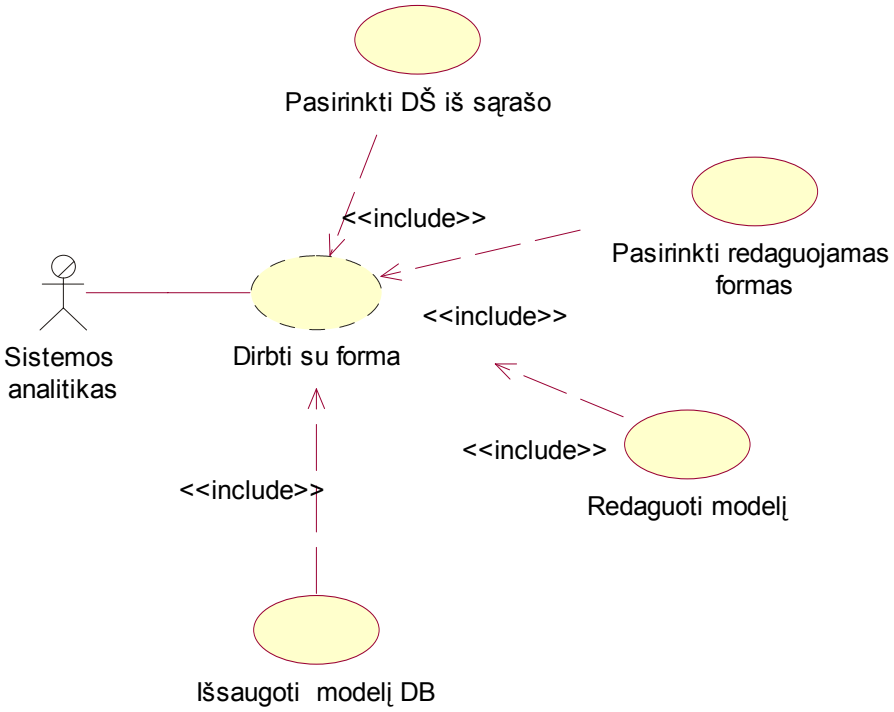
Ieva Subačiūtė

3.2. Kuriamos sistemos reikalavimų modelio aprašymas

3.2.1. Panaudojimo atvejų diagrama

Panaudojimo atvejai aprašo sistemos elgesį, bet ne jos realizaciją. Dėl to vartotojams, projektuotojams, ekspertams yra lengviau suprasti sistemą.

Kuriamą sistemą galima panaudoti darbui su DŠ formomis. Darbas su forma apima funkcijas, kurias galima atlikti sistemos pagalba. Visi tie veiksmai priklauso vienas nuo kito, t.y. negalima pasirinkti redaguojamų formų, prieš tai nepasirinkus DŠ, negalima redaguoti formos modelio, prieš tai nepasirinkus, kurios formos bus redaguojamos.

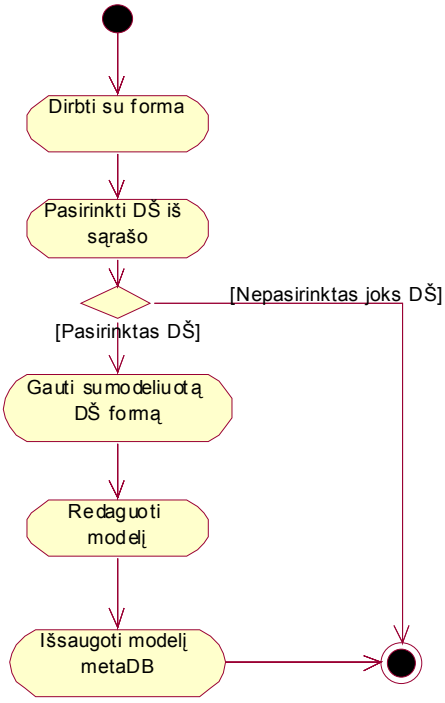


4 pav. Sistemos panaudojimo atvejų diagrama

3.2.2. Veiklos diagramos panaudojimo atvejų modelyje

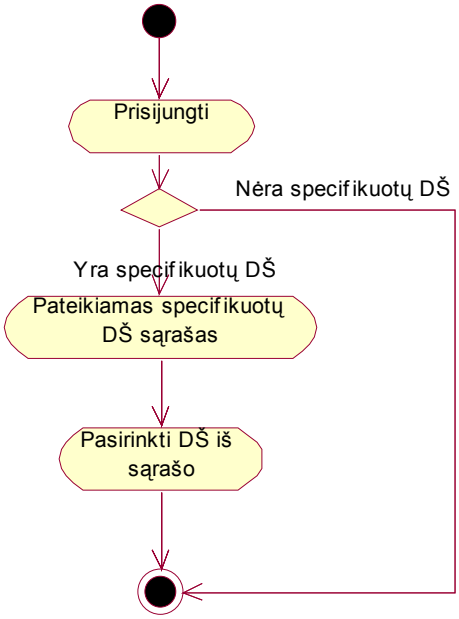
Pagrindinė veiklos diagramos paskirtis – aprašyti veiksmus bei procesus, į kuriuos įtraukiami vienas ar daugiau objektų. Toliau pateikiamos trys veiklos diagramos panaudojimo atvejams aprašyti. Jos visos yra glaudžiai susijusios viena su kita.

Veiklos diagrama „Dirbti su forma“ (pav. 5) apima kitas dvi diagramas, t.y. darbas su forma reiškia, kad bus pasirenkamas DŠ, redaguojama jo forma, o paskui ji išsaugoma meta duomenų bazėje.



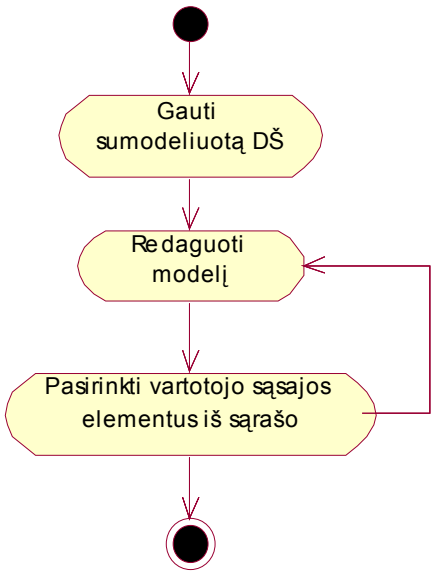
5 pav. Diagrama panaudojimo atvejui “Dirbti su forma”

Jei prisijungus prie duomenų bazės paaiškėja, jog specifikuotų duomenų šaltinių nėra, tuomet DŠ pasirinkti negalima (pav. 6).



6 pav. Diagrama panaudojimo atvejui "Pasirinkti DŠ iš sąrašo"

Jau pasirinkus pageidaujamą DŠ gauname naują pasirinkimą – jo apdorojimo etapus arba formas, t.y. redaguojamas modelis. Tai daroma pasirenkant norimus vartotojo sąsajos elementus (pav. 7).

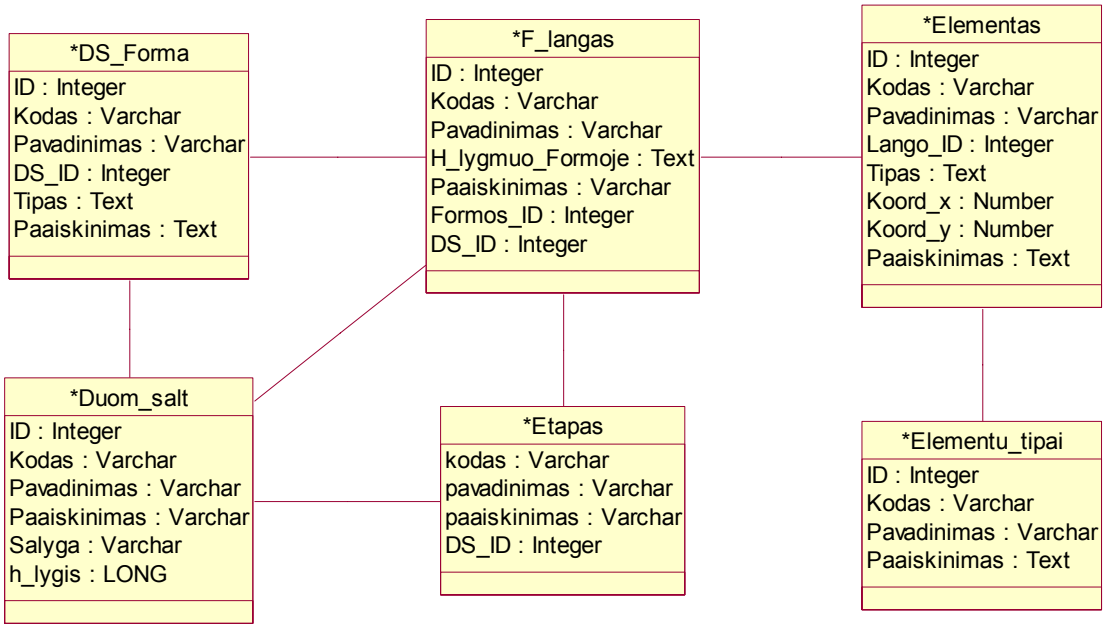


7 pav. Diagrama panaudojimo atvejui "Redaguoti modelį"

3.3.3. Dalykinės srities klasių diagrama

Dalykinės srities klasių diagrama – tai koncepcinis duomenų modelis. Klasė – tai objektų aibė, kurios elementai turi vienodą struktūrą, elgseną, ryšius ir semantiką.

Šiame projekte naudojamos šešios pagrindinės klasės (8 pav.): Duom_salt, DS_Forma, F_langas, Etapas, Elementas ir Elementų_tipai. Duom_salt aprašo visus esamus duomenų šaltinius – dokumentus. Elementas – tai įvairūs WUI elementai. WUI – Windows User Interface – tai šablonas elementų, kuriuos galima naudoti vartotojo sąsajos modeliavimui su *MS Visio 2000*. F_langas – pagrindinė lentelė, reikalinga sistemos funkcionavimui, ji saugo informaciją apie visų duomenų šaltinių apdorojimo etapų langų išvaizdą ir struktūrą. Pačius etapus galima surasti lentelėje Etapas. F_langas ir Elementas – tai dvi svarbiausios lentelės, modeliuojant vartotojo sąsają. Lentelėje Elementu_tipai saugomi visi galimi vartotojo sąsajos elementų tipai iš WUI (Windows User Interface) šablono, esančio *MS Visio 2000* pakete.

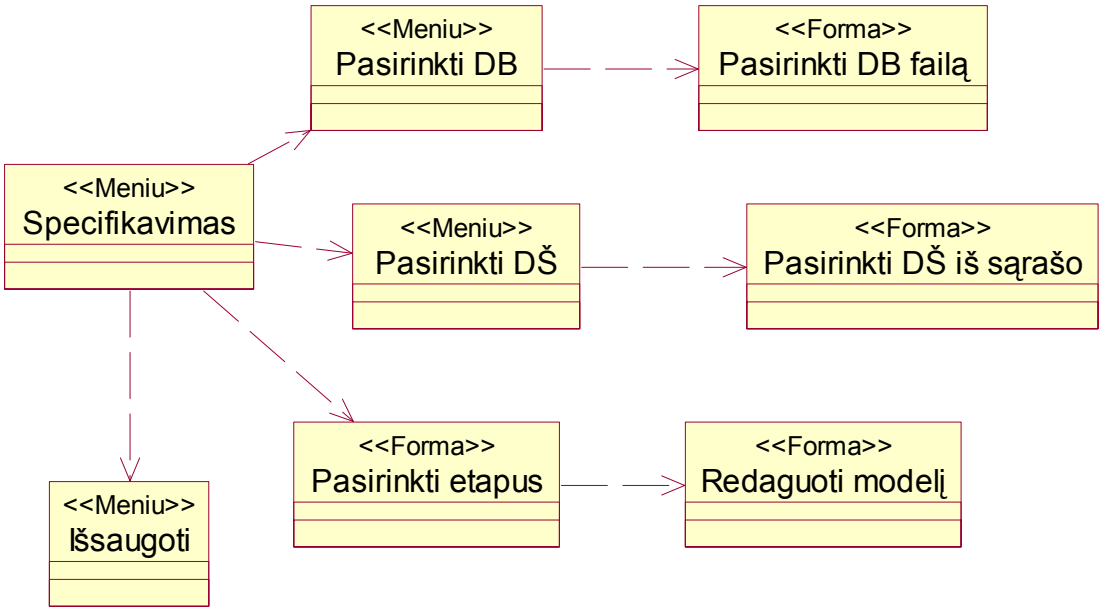


8 pav. Dalykinės srities klasių diagrama

Klasių diagrama parodo, kokios lentelės duomenų bazėje naudojamos duomenims apie duomenų šaltinius bei jų modeliavimo rezultatams saugoti. Duomenų bazėje saugomi duomenys apie suprojektuotą vartotojo sąsają.

3.3.4. Vartotojo sąsajos modelis

Sistemos vartotojo sąsaja – tai įprasta *MS Visio 2000* vartotojo sąsaja papildyta nauju meniu – „Specifikavimas“. Šiame meniu yra keturi punktai, iš kurių tik vienas yra aktyvus darbo pradžioje. Atitinkamai iš eilės pasirenkant meniu punktus, ir kiti tampa aktyvūs.

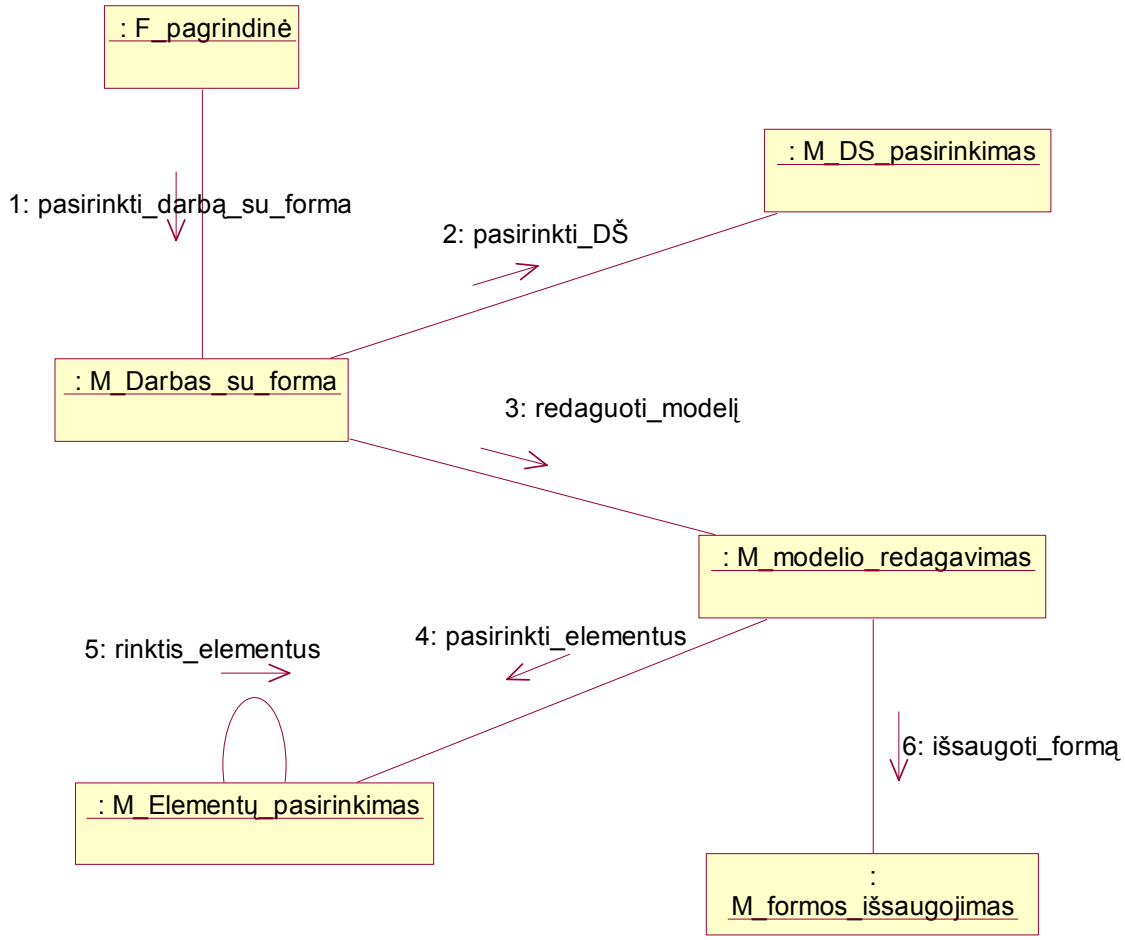


9 pav. Sistemos vartotojo sąsajos modelis

3.3. Sistemos, vartotojo sąsajai kurti, projekto pateikimas

3.3.1. Sistemos panaudojimo bendradarbiavimo diagrama

Sistemos darbas realizuotas pagal tokią bendradarbiavimo diagramą, kurios funkcijų ar formų pavadinimai nebūtinai sutampa su pateiktais 10 pav.

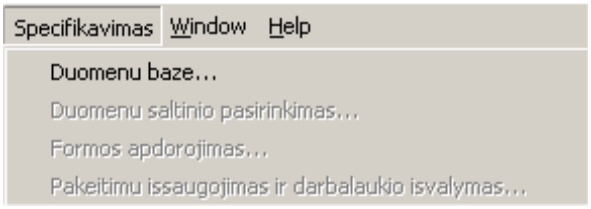


10 pav. Sistemos panaudojimo bendradarbiavimo diagrama

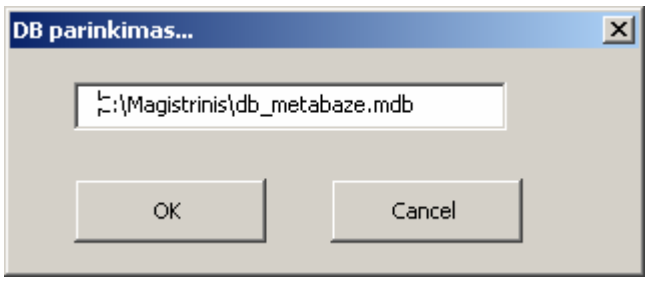
3.3.2. Sistemos elgsenos modelis

Sistema pradeda darbą atidarius *MS Visio 2000* dokumentą. Sukuriamas meniu su keturiais punktais. Sugeneruoto meniu vaizdas pateiktas 11 pav. Pradžioje yra aktyvus tik

pirmasis meniu punktas. Renkantis meniu punktus vieną po kito, aktyvuojami sekantys punktai. Pasirinkus pirmąjį punktą išvedamas dialogo langas (12 pav.), pilnam keliui iki duomenų bazės nurodyti. Tuo pačiu keliu bus saugomi ir pakeitimai, atlikti dirbant su sistema.

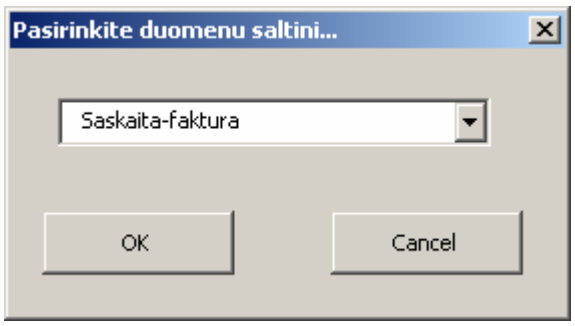


11 pav. Sistemos meniu punktai



12 pav. Sistemos dialogo langas keliui nurodyti.

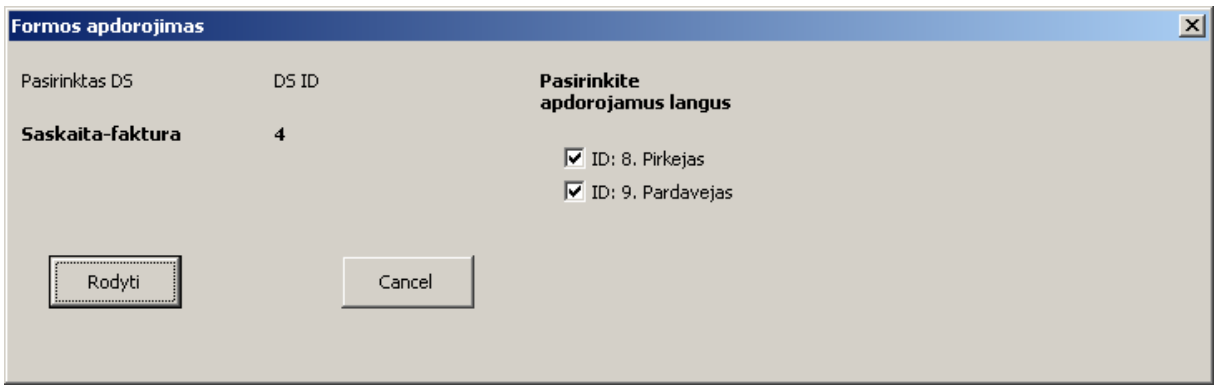
Pasirinkus duomenų bazės failą išrenkami duomenų šaltiniai iš lentelės Duom_salt ir aktyvus pasidaro antrasis meniu punktas – „Duomenu saltinio pasirinkimas...“. Pasirinkus šį punktą gaunamas dialogo langas su išrinktų duomenų šaltinių pavadinimų sąrašu (13 pav.).



13 pav. Duomenų šaltinio pasirinkimo dialogo langas.

Pasirinkus norimą apdoroti duomenų šaltinį iš sąrašo, aktyvuojamas trečias meniu punktas – „DS (pavadinimas) apdorojimas...“ (čia: pavadinimas – duomenų šaltinio pavadinimas iš duomenų bazės). Iš pradžių šis meniu punktas buvo vadinamas „Formos apdorojimas“. Tuo pat metu iš lentelės F_langas įrašų išrenkami pasirinkto DŠ apdorojimo etapai – DŠ formos. Pasirinkus meniu punktą, atveriamas dialogo langas (14 pav.), kuriame reikia pasirinkti,

kuriuos DŠ apdorojimo etapus norima redaguoti, t.y. pateikiamas sąrašas įrašų iš lentelės F_langas, kurie yra susiję su anksčiau pasirinktu duomenų šaltiniu (dokumentu).

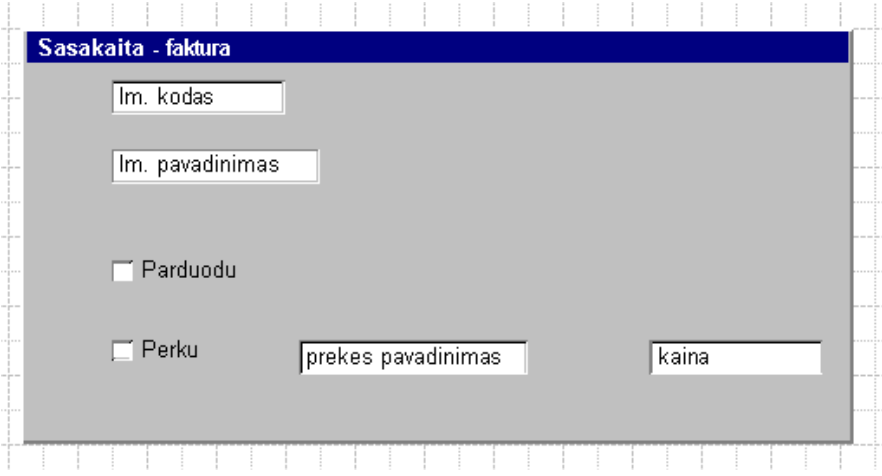


14 pav. Apdorojamų formų pasirinkimo dialogo langas

Paspaudus mygtuką „Rodyti“, aktyvuojamas paskutinysis meniu punktas. Jis leidžia kiekvienoj formoj atlikus pakeitimus išsaugoti duomenų bazėje. Pakeitimai saugomi trimis žingsniais:

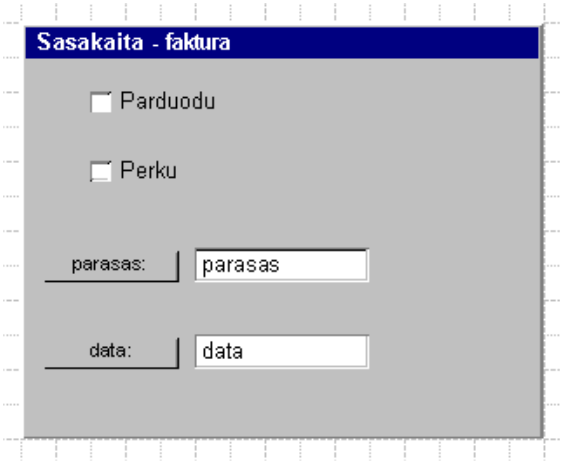
- sukuriami nauji įrašai, jei atsirado elementų, kurių formoje anksčiau nebuvo;
- įrašai atnaujinami, jei buvo pakeista elemento koordinatės formoje ar pavadinimas;
- įrašai pašalinami, jei jie buvo išmesti iš DŠ formos.

Tuo pačiu iš lentelės Elementas išrenkami su pasirinktais duomenų šaltinio apdorojimo etapais susiję GUI elementai, kurie vaizduojami susijusiuose formų languose. Duomenų šaltinio apdorojimo etapai vaizduojami atskirais *MS Visio 2000* dokumento langais. 15 pav. pavaizduota pradinė duomenų šaltinio „Sąskaita – faktūra“ forma, kuri yra aprašyta duomenų bazėje. Šia formą galima redaguoti, keisti. Tai forma, kurią vėliau turėtų pildyti pirkėjas.



15 pav. DŠ „Sąskaita – faktūra“ – pirmas etapas

16 pav. matome pradinę to paties duomenų šaltinio antro apdorojimo etapo formą. Tai formą, kurią vėliau turėtų pildyti pardavėjas.



16 pav. DŠ „Sąskaita – faktūra“ – antras etapas

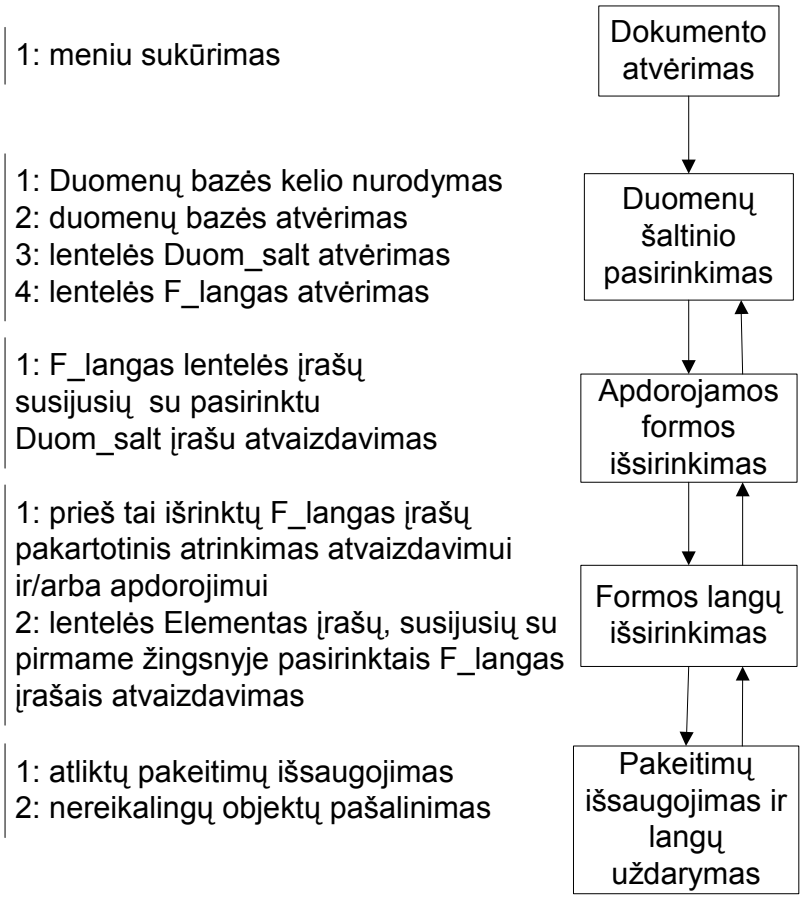
Abi pateiktas formas galima redaguoti, papildyti elementais, kurių gali prireikti įvedant informaciją apie pirkimą – pardavimą. Tai turėtų atlikti sistemos analitikas, kuris modeliuoja vartotojo sąsają projektuojamai kompiuterizuojamai informacijos sistemai.

Pasirinkus paskutinį meniu punktą, t.y. norint išsaugoti pakeitimus, jie išsaugomi, ir formos uždaromos. Darbas su sistema baigiamas uždarant dokumentą.

Duomenų šaltinio aprašas:

Paprastumo dėlei, pirkėjo pildomoje dokumento dalyje rašoma: įmonės kodas, įmonės pavadinimas, prekės pavadinimas ir kaina bei pažymima varnelė, prie perku; pardavėjas pildo tik du laukus: užrašo datą ir parašą bei pažymi varnelę prie parduodu.

Aukščiau aprašytas sistemos elgesys struktūriškai pavaizduotas 17 pav.



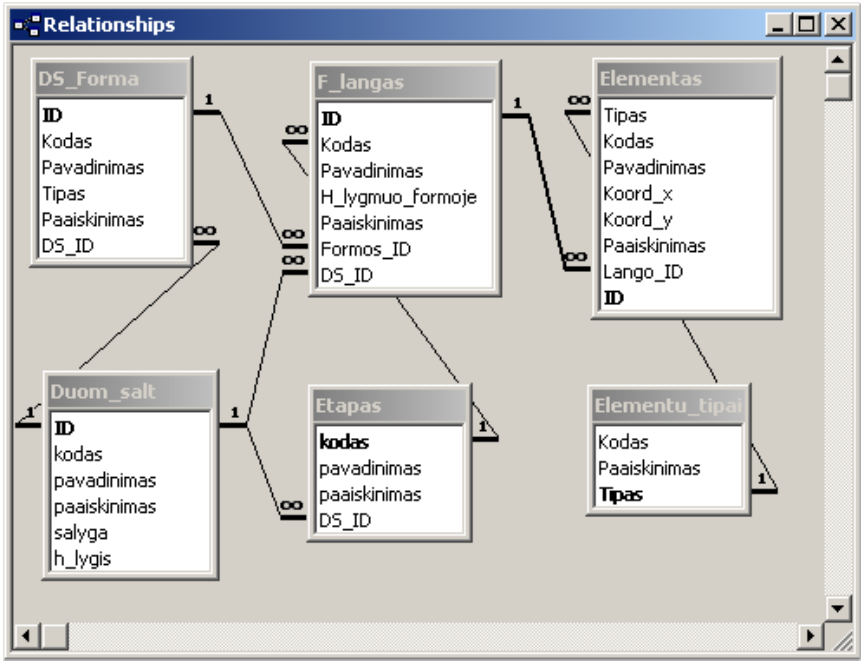
17 pav. Sistemos elgsenos modelis

3.3.3. Duomenų bazės modelis.

Duomenų bazė naudojama duomenims apie duomenų šaltinius bei jų modeliavimo rezultatams saugoti. Duomenų bazėje saugomi duomenys apie suprojektuotą vartotojo sąsają.

Informacijos srautai, kuriais remiantis vėliau modeliuojami DŠ vartotojo sąsajai, nėra pilnai specifikuoti. Saugoma tik pagrindinė informacija apie duomenų šaltinius, t.y. pavadinimas, apdorojimo etapų skaičius (kiek kartų dokumentas turėtų būti pildomas), bei keli pradiniai elementai (tokie kaip „textbox“, „listbox“). Informacija apie tai, kas ir kokiame duomenų šaltinyje turi būti panaudota, kas kiekvieną DŠ pildys, ir kaip nuspręsti, kokius elementus naudoti, nėra specifikuota. Visa tai sprendžia sistemos analitikas, kurdamas sistemą.

18 pav. pateiktas duomenų bazės, naudojamos sistemoje, esybių – ryšių (ER) modelis. Duomenų bazė buvo suprojektuota MS Visio 2000 priemonėmis, o paskui sugeneruota į MS Access duomenų bazę.



18 pav. Suprojektuotos sistemos duomenų bazės modelis.

3.3.4. Sistemos architektūros aprašymas.

Sistemos vartotojo sąsajai kurti architektūra jau pradėta aprašinėti skyriuose „3.3.2. Sistemos elgsenos modelis“ bei „3.3.3. Duomenų bazės modelis“. Čia konkrečiau pasigilinsime į patį programos tekstą ir funkcijas, kurios realizuoja sistemos darbą.

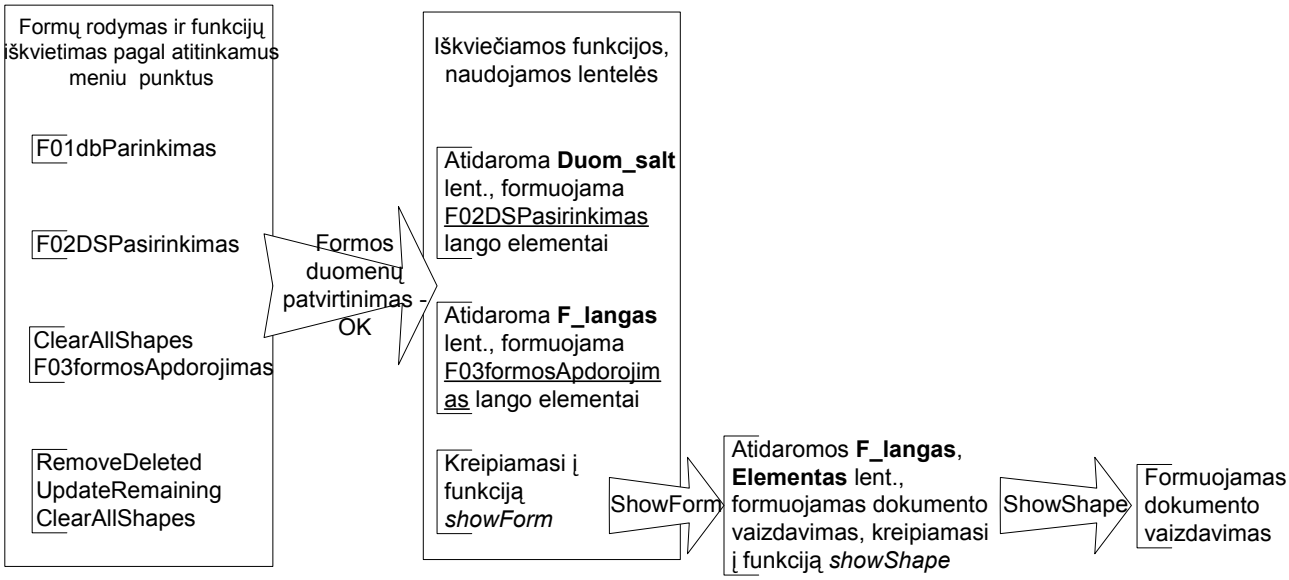
Sistemoje realizuoti du moduliai – Meniu ir Formų. Kiekviename jų realizuotos atskiros funkcijos, susijusios su skirtingais programos elementais ir atliekančios skirtingus veiksmus. Meniu modulis sukuria meniu, ir jo pagalba generuojami dialogo su sistemos vartotoju langai – sistemos formos. Formų modulis realizuoja pačių vartotojo sąsajos elementų kūrimą ir modeliavimą. Abu moduliai yra glaudžiai susiję tarpusavyje ir kai kuriose vietose netgi persipynę vienas su kitu.

Kiekvienas meniu (Meniu modulis), pavaizduoto 3.3.2. skyriaus 11 pav., punktas atlieka tam tikras operacijas remdamasis Formų moduliu. Abu moduliai realizuoti *MS Visual Basic for Applications* programavimo kalba. 4 lentelėje pateiktos funkcijos ir sistemos formos, jų

atliekami programiniai veiksmai. 19 pav. pavaizduotas tų modulių tarpusavio ryšys ir integruotumas sistemos darbo metu.

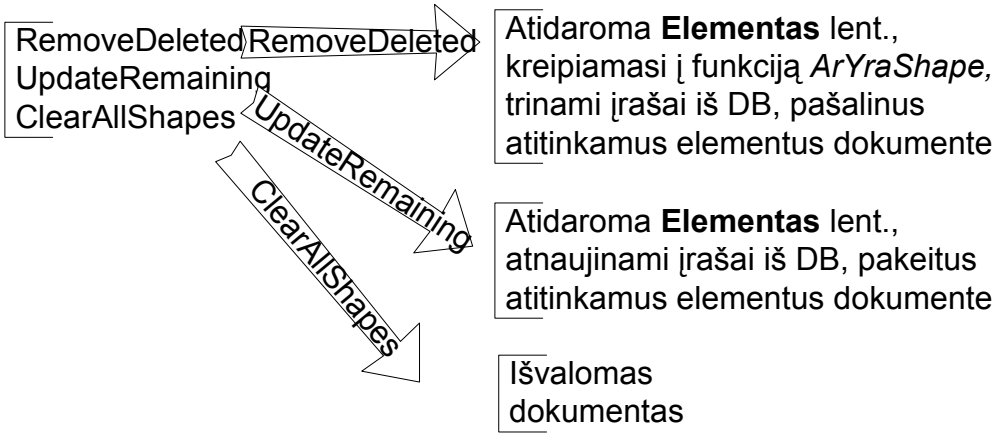
4 lentelė. Sistemos funkcijos ir jų atliekami veiksmai.

Funkcija (F) arba forma (FO)		Atliekami veiksmai
m00meniu	F	Realizuoja sistemos meniu
m01do	F	Realizuoja meniu punktą duomenų bazei pasirinkti
F01dbParinkimas	FO	Sugeneruojamas dialogo langas, kuriame vartotojas nurodo kelią iki duomenų bazės. Atidaroma duomenų bazė ir lentelė Duom_salt. Suformuojami antro meniu punkto formos lango elementai.
m02ado	F	Realizuoja meniu punktą duomenų šaltiniui pasirinkti
F02DSPasirinkimas	FO	Atidaroma lentelė F_langas.
m03ameniu	F	Suformuojami trečio meniu punkto formos lango elementai. Šio punkto antraštė pakeičiama pagal DŠ vardą.
m03do	F	Realizuoja meniu punktą formų apdorojimui pasirinkti
F03formosApdorojimas	FO	Kreipiamasi į funkciją *showFormsPage*.
showFormsPage	F	Atvaizduojami lentelės F_langas įrašai susiję su pasirinktu Duom_salt.
showShape	F	Atvaizduoja vartotojo sąsajos elementą, kuris nurodomas kreipinyje. Tai atliekama kiekvienam elementui, susijusiam su vaizduojama DŠ forma.
m04do	F	Realizuoja meniu punktą pakeitimų išsaugojimui
ClearAllShapes	F	Ištrinami dokumente atvaizduoti elementai – išvalomas darbo laukas. Naudojama kiekvieną kartą renkantis antrą ir trečią menių punktus, tam kad dokumente neliktų nereikalingų elmentų. Parodomas pranešimas apie tai.
UpdateRemaining	F	Tikrinami egzistuojančių elementų dokumente pakeitimai lyginant su duomenimis DB.
RemoveDeleted	F	
ArYraShape	F	Jei elementas su norodytu ID dokumente nerandamas, jis ištrinamas iš DB.



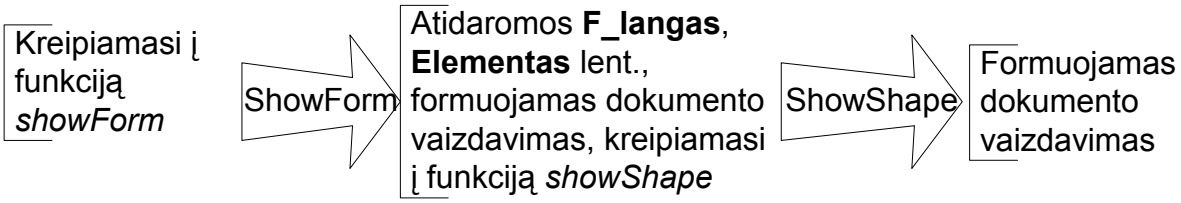
19 pav. Sistemos meniu iškviečiamos formos, funkcijos ir DŠ formų modeliavimas

Norint išsaugoti duomenų bazėje dokumento formoje ar formose atliktus pakeitimus, pasirenkamas ketvirtas meniu punktas. Išsaugojimas vykdomas keliais etapais: pirmiausiai pagal elementų ID tikrinama, ar jie vis dar yra dokumente, jei nėra – ištrinama iš DB. Po to, tikrinami elementų egzistuojančių dokumente pasikeitimai lyginant su duomenimis esančiais DB, ir įrašai DB atitinkamai atnaujinami. Ir galų gale, iš darbalaukio – Visio dokumento – pašalinami visi elementai, kad būtų galima tęsti darbą pasirinkus kitą DŠ ar kitą jo apdorojimo etapą. 20 pav. pavaizduota, kada panaudojamos funkcijos *RemoveDeleted, UpdateRemaining* ir *ClearAllShapes*.



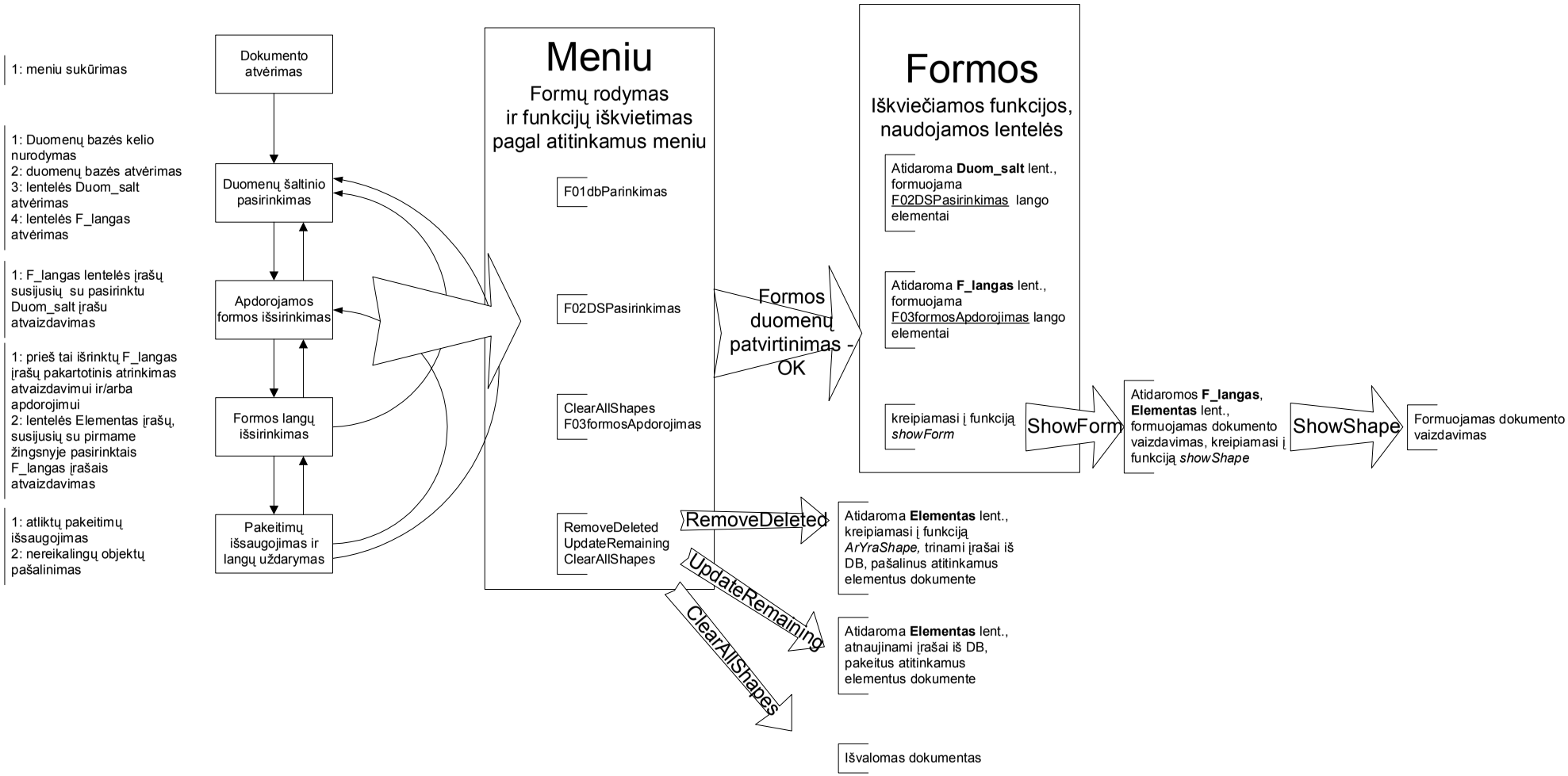
20 pav. Paskutinio meniu punkto detalizavimas panaudojant funkcijas

Funkcija *showForm* (sistemos elgsenos algoritmo funkcijų panaudojimo detalizavimas, 21 pav.) naudoja lenteles F_langas ir Elementas ir kreipiasi i funkciją *showShape* dokumento vaizdavimo formavimui.

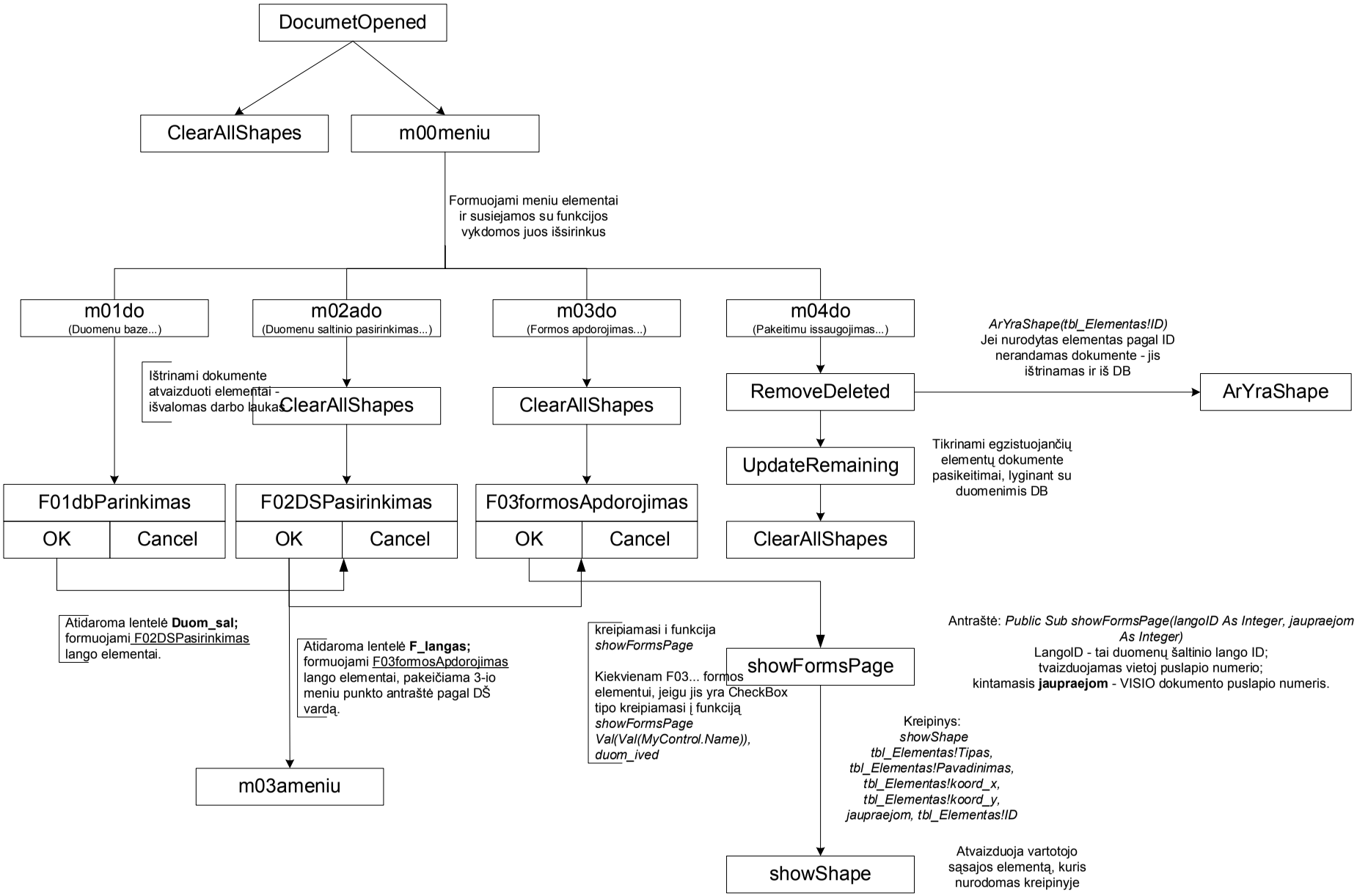


21 pav. Funkcijos *showForm* ir *showShape*.

Aprašytos sistemos elgsenos struktūrizuotas modelis pateiktas 22 pav. Kompiuterizuotos informacinės sistemos architektūros modelis pateiktas 23 pav.



22 pav. Struktūrizuotas sistemos elgsenos algoritmas



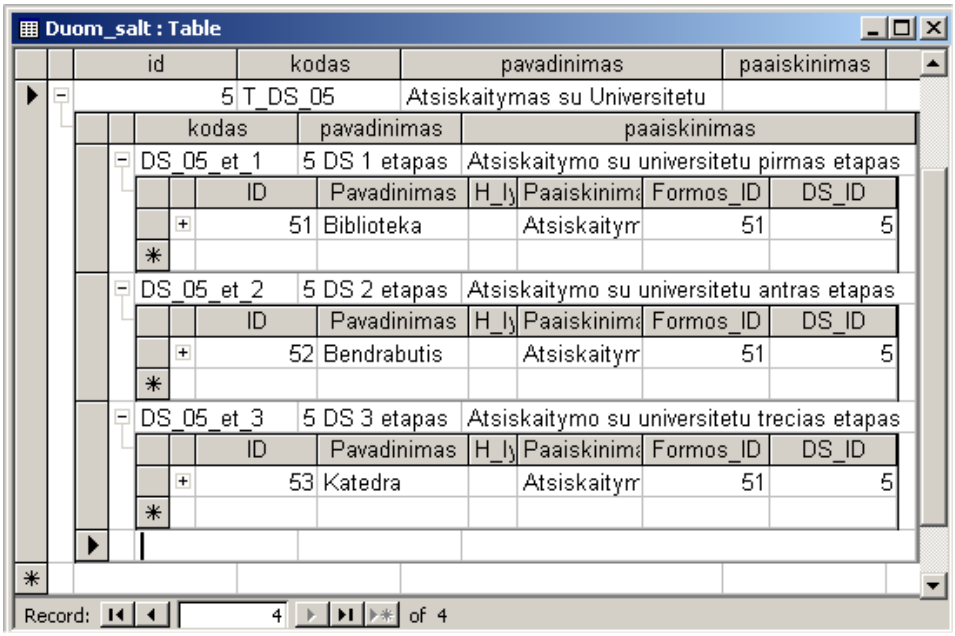
23 pav. Kompiuterizuotos sistemos architektūros modelis.

3.3.5. Testavimo duomenų aprašymas

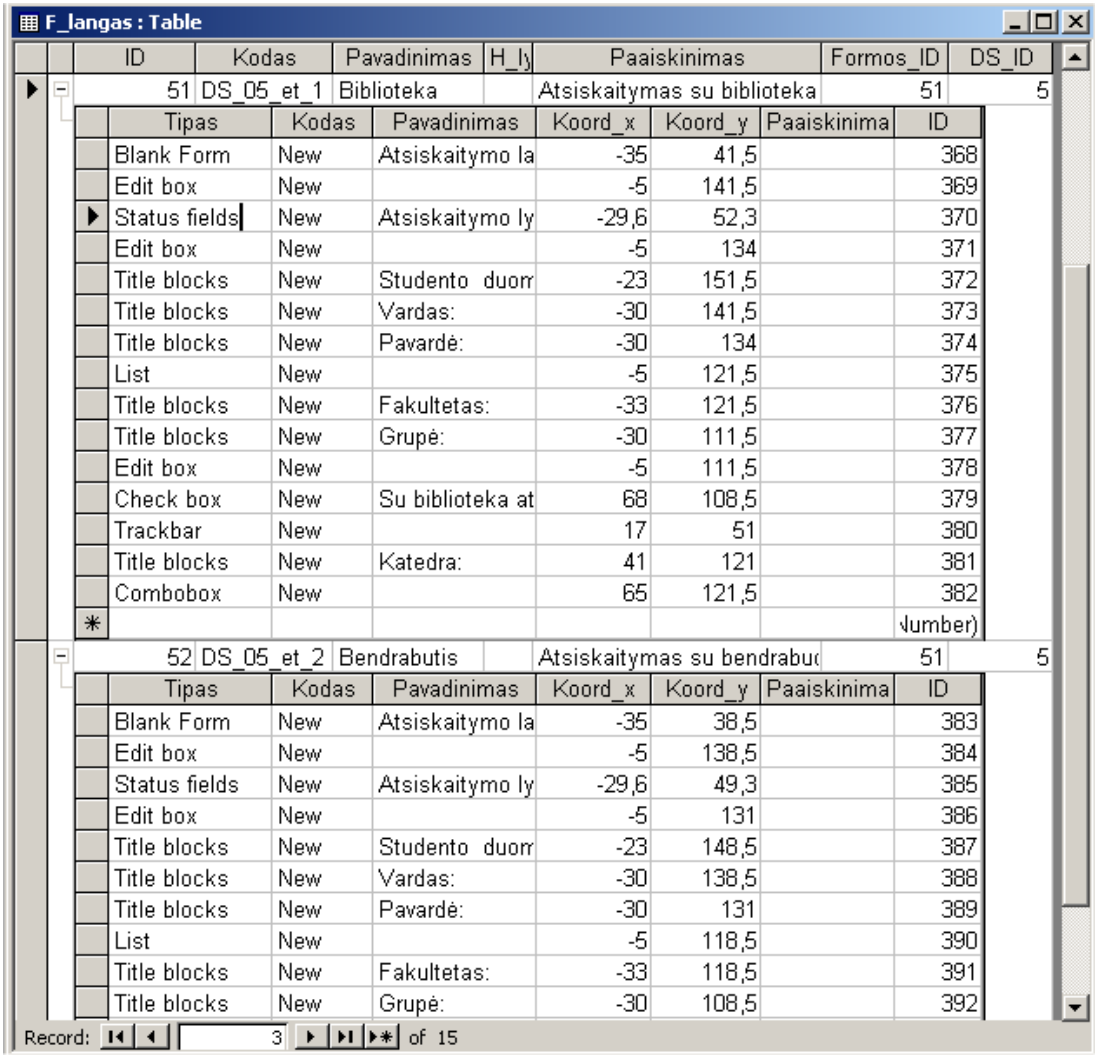
Testavimui pasirinktas paprastas dokumentas „Atsiskaitymo lapelis“, kuriame studentas, prieš gaudamas diplomą, turi surinkti parašus, kad atsiskaitė su universitetu.

Kiekviename duomenų šaltinio – „Atsiskaitymo lapelio“ – pildymo etape užpildomi tokie laukai: studento vardas, pavardė, fakultetas, katedra ir grupė, kuriems jis priklauso. Atitinkamai pastumiamas atsiskaitymo lygio rodiklis, ir pažymima varnelė prie atitinkamo lauko: su biblioteka atsiskaityta, su bendrabučiu atsiskaityta, su katedra atsiskaityta.

Tokio DŠ aprašymas meta duomenų bazėje pavaizduotas 24 ir 25 pav.



24 pav. Testinio DŠ „Atsiskaitymo lapelis“ apdorojimo etapai DB lentelėse Duom_salt, Etapas ir F_langas.



F_langas : Table

ID	Kodas	Pavadinimas	H_ly	Paaiskinimas	Formos_ID	DS_ID
51	DS_05_et_1	Biblioteka		Atsiskaitymas su biblioteka	51	5

Tipas	Kodas	Pavadinimas	Koord_x	Koord_y	Paaiskinima	ID
Blank Form	New	Atsiskaitymo la	-35	41,5		368
Edit box	New		-5	141,5		369
Status fields	New	Atsiskaitymo ly	-29,6	52,3		370
Edit box	New		-5	134		371
Title blocks	New	Studento duom	-23	151,5		372
Title blocks	New	Vardas:	-30	141,5		373
Title blocks	New	Pavardė:	-30	134		374
List	New		-5	121,5		375
Title blocks	New	Fakultetas:	-33	121,5		376
Title blocks	New	Grupė:	-30	111,5		377
Edit box	New		-5	111,5		378
Check box	New	Su biblioteka at	68	108,5		379
Trackbar	New		17	51		380
Title blocks	New	Katedra:	41	121		381
Combobox	New		65	121,5		382
						Number)

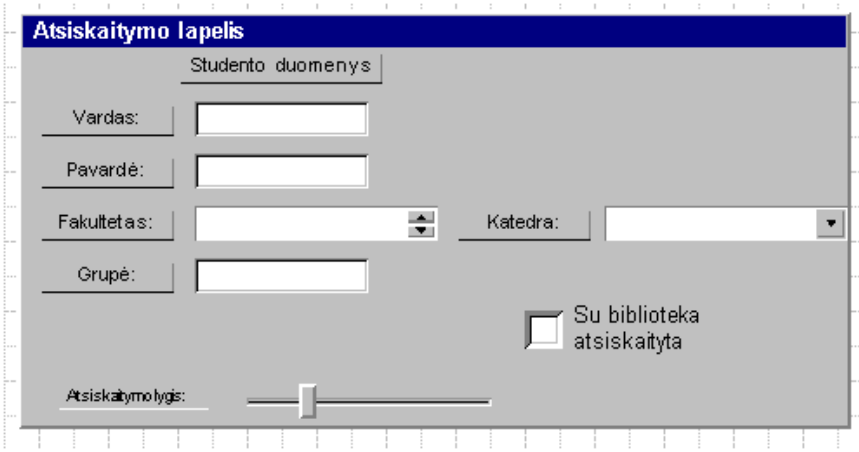
52	DS_05_et_2	Bendrabutis	Atsiskaitymas su bendrabuc	51	5

Tipas	Kodas	Pavadinimas	Koord_x	Koord_y	Paaiskinima	ID
Blank Form	New	Atsiskaitymo la	-35	38,5		383
Edit box	New		-5	138,5		384
Status fields	New	Atsiskaitymo ly	-29,6	49,3		385
Edit box	New		-5	131		386
Title blocks	New	Studento duom	-23	148,5		387
Title blocks	New	Vardas:	-30	138,5		388
Title blocks	New	Pavardė:	-30	131		389
List	New		-5	118,5		390
Title blocks	New	Fakultetas:	-33	118,5		391
Title blocks	New	Grupė:	-30	108,5		392

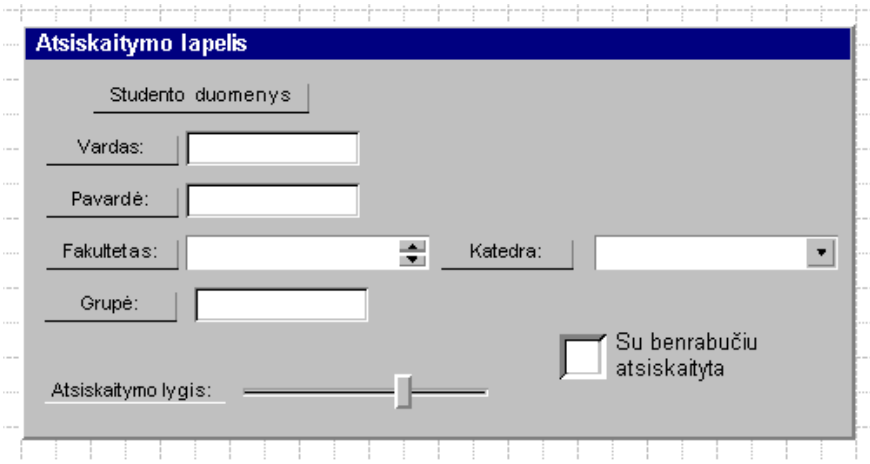
Record: 3 of 15

25 pav. Testinio DŠ apdorojimo etapų specifikacija DB lentelėse F_langas ir Elementas.

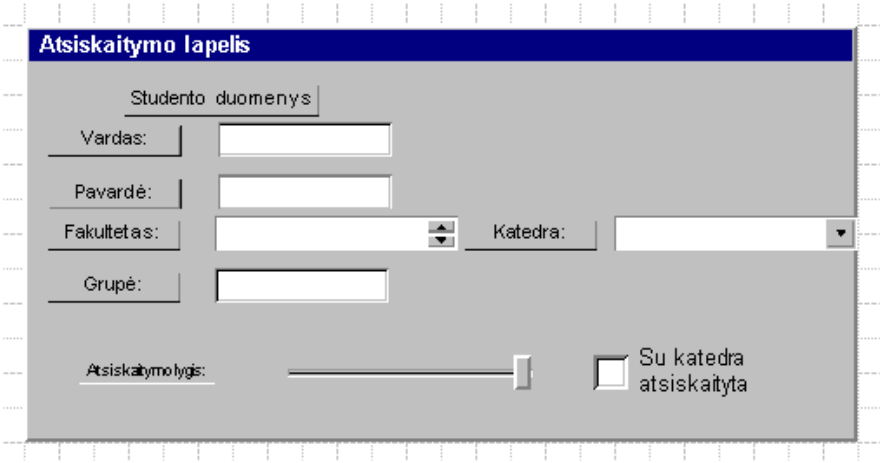
Trys duomenų šaltinio apdorojimo etapai reiškia, kad vartotojui bus pateikti trys *MS Visio 2000* dokumento langai, kurių pavadinimas (puslapio numeris) – tai duomenų šaltinio lango ID. Kaip atrodys vartotojui pateikiamos sugeneruotos *MS Visio 2000* dokumento formos pagal minėtą DŠ aprašymą, pateikta 26, 27 ir 28 pav.



26 pav. Duomenų šaltinis „Atsiskaitymo lapelis“ – pirmas etapas



27 pav. Duomenų šaltinis „Atsiskaitymo lapelis“ – antras etapas



28 pav. Duomenų šaltinis „Atsiskaitymo lapelis“ – trečias etapas

Gavus šiuos modelius, galima juos redaguoti, keisti, išsaugoti pakeitimus duomenų bazėje. Tokiu būdu atliekamas testavimas leidžia patikrinti programiškai realizuoto algoritmo teisingumą.

3.3.6. Reikalavimai sistemos funkcionavimo palaikymui

Kad sistema tinkamai funkcionuotų, reikalinga tenkinti tokias technines ir programines sąlygas:

Operacinė sistema – Windows 98 ar vėlesnė;

terpė, kurioje veikia sistema – *MS Visio 2000*;

MS Access iš MS Office paketo;

procesoriaus greitis – 350MHz, operatyvinė atmintis (RAM) – 96MB, laisvos vietos kietajame diske – 500MB.

3.3.7. Sistemos naudojimo instrukcija

Sistemai startuoti reikia turėti tokią programinės įrangos terpę: operacinę sistemą Windows 98 ar naujesnę versiją, MS Access 2000 arba vėlesnę versiją, MS Visio 2000.

Bylos reikalingos sistemai startuoti: _mag.vsd, wui.vss, db_metabaze.mdb.

Startuojant reikia atidaryti bylą _mag.vsd, tuomet atidaromas paketas MS Visio 2000. Lango kairėje pusėje atsidaro šablonas WUI su savo elementais. Viršutiniame meniu reikia pasirinkti „Specifikavimas“, o iš jo – „Duomenu baze...“. Atsidariusiame dialogo lange reikia nurodyti pilną kelią iki duomenų bazės db_metabazė.mdb ir paspausti „OK“. Tada viršutiniame meniu pasirinkti „Specifikavimas“ ir „Duomenu saltinio pasirinkimas...“. Atsivėrusiame dialogo lange iš pasirinkti duomenų šaltinį, kurį norima modeliuoti.

Toliau vėl reikia eiti į meniu „Specifikavimas“ -> „DS (pavadinimas) apdorojimas...“. Gautame dialogo lange reikia pasirinkti norimus matyti ir redaguoti DŠ langus (apdorojimo etapus) ir paspausti „OK“. Tuomet iš DB išrenkami su tais etapais susiję elementai ir

vaizduojami atitinkamuose MS Visio 2000 languose. Čia galima įkelti ar išmesti elementus iš kairėje pusėje esančio šablono WUI.

Baigus rikiuoti elementus, ir gavus norimą rezultatą, reikia jį išsaugoti. Tam reikia pasirinkti „Specifikavimas“ -> „Pakeitimu išsaugojimas ir darbalaukio išvalymas...“. Tuomet galima vėl rinktis DŠ arba jo formas redagavimui.

Sistema prieš daugelį veiksmų praneša, kad tai, kas tuo metu matoma MS Visio 2000 dokumento lange, bus pašalinta. Tai reiškia, kad bus matomas tuščias dokumentas, bet duomenys niekur nedings – jie saugomi duomenų bazėje.

Norint baigti darbą su sistema, reikia paprasčiausiai uždaryti MS Visio 2000 dokumentą. Jei sistema klausia, ar išsaugoti pakeitimus, reikia išsaugoti.

Į sistemos, skirtos vartotojo sąsajos modeliavimui, naudojimo instrukciją taip pat įeina ir 3.3.2. skyrius, kuriame pateikta sistemos darbo iliustracinė medžiaga.

4. Sistemos vartotojo sąsajai kurti eksperimentinis tyrimas

4.1. Sukurtos sistemos kokybės tyrimas

Tiriant sukurtos sistemos kokybę pastebėta, jog ji tenkina kriterijus, pagal kuriuos buvo numatyta ją vertinti 2.3 skyriuje. Sistema yra pakankamai išbaigta, t.y. ji atitinka sudarytą projektą.

Sistemos toleranciją klaidoms galima pastebėti jau pirmame žingsnyje, kurį reikia atlikti pradedant darbą su sistema. Nepasirinkus teisingo duomenų bazės failo, sistema neveiks, būtina pasirinkti teisingai. Sistema leidžia tai daryti daug kartų, tačiau darbas su sistema neįmanomas, nenurodžius teisingo duomenų bazės failo.

Modulis yra gana draugiškas vartotojo atžvilgiu. Tai reiškia, kad jį gali suprasti ir juo naudotis asmuo anksčiau dirbęs *MS Visio 2000* aplinkoje. Tačiau reikalingos kai kurios papildomos žinios apie konkrečius duomenų šaltinius ir jų savybes, apie informacijos srautus, kurie nulemia vienokį ar kitokį vartotojo sąsajos modeliavimo stilių.

Sistema yra pakankamai integrali ir gali būti sujungta su kitais moduliais, realizuotais *MS Visual Basic for Applications* programavimo kalba.

5 lentelėje pateiktas kelių sistemų – modulių savybių įvertinimas, iš ko galima matyti, kiek, palyginus su plačiai žinomomis sistemomis, sukurtas modulis pasižymi savo išbaigtumu, integralumu ir kitomis svarbiomis savybėmis. Savybių išvystymo lygis, remiantis asmenine patirtimi, vertinamas ir žymimas trimis lygiais: „+“ – gerai, „–“ – blogai, „+/–“ – vidutiniškai.

5 lentelė. Sistemos įvertinimo kriterijų palyginimas su kitomis sistemomis.

Savybė / **Modulis**	Netolerancija klaidoms	„Draugiškumas“ vartotojo atžvilgiu	Integralumas	Panaudojamumas	Išbaigtumas
Vartotojo sąsajos modeliavimo	+/–	+	+	+/–	+/–
Visio	+	+	+	+	+
Rational Rose	+	+/–	+/–	+	+

4.2. Tolimesnio sistemos tobulinimo, plėtojimo galimybės

Sukurtą sistemą yra galimybė išplėsti ir tobulinti, galima taip pat ir modifikuoti jos elgesį tam tikrais atvejais.

Toliau plėtojant sistemos galimybes, reikėtų apimti daugiau duomenų šaltinių, daugiau jų charakteristikų, giliau panaudoti informacijos srautų specifikacijas. Pavyzdžiui, dabar sistemai visai nesvarbu, kas ir ką atlieka, t.y. funkcijos ir duomenų šaltiniai nesusieti su konkrečiais aktoriais, nors tokia galimybė yra ir gali būti įgyvendinta. Dabar visai nesvarbu, kas pradeda pildyti dokumentą ir kas pabaigia tai daryti. Tai galima pakeisti, pavyzdžiui, suteikiant skirtingas formų antraštes toms dokumentų dalims, kurios turėtų būti pildomos skirtingų aktorių.

Sistemoje nėra galimybės atsispausdinti ataskaitos apie egzistuojančias, naujai specifikuotas ir jau sumodeliuotas dokumentų formas – vartotojo sąsają. Tai vienas iš galimų

sistemos plėtojimo būdų. Taip pat galima būtų fiksuoti ir saugoti daugiau elementų savybių. Tokių kaip: spalva, dydis, teksto parametrai.

Tačiau svarbiausias sistemos plėtojimo akcentas – tai duomenų šaltinių detalesnis, nuoseklesnis specifikavimas. Čia turima galvoje, kad reikėtų duomenų bazėje nurodyti daugiau informacijos, pagal kurią būtų modeliuojamos duomenų šaltinių formos. Pavyzdžiui, kas apsprendžia, kokį elementą panaudoti konkrečiam dokumento (duomenų šaltinio) lauko vaizdavimui – textbox ar listbox ir pan.

Taigi, sistema turi labai plačias plėtojimo galimybes, ją daug kur galima keisti ir modeliuoti. Sistema yra tik pradinis modelis, kuriant kitą, daugiau informacijos srautų specifikacijų naudojančią sistemą.

5. Išvados

1. Šiame darbe atlikta vartotojo sąsajos kūrimo principų ir metodų analizė šiais aspektais:
   a) pasirengimas vartotojo sąsajos kūrimui;
   b) prototipo kūrimo metodai: iteracinis, pagrįstas integruota funkcinių ir vartotojo sąsajos reikalavimų specifikacijas, bei kūrimas panaudojant scenarijus ir būsenų diagramas;
   c) išnagrinėtas kompiuterizuojamų informacijos sistemų funkcionalumo rezultatų ir duomenų šaltinių struktūrų specifikavimo metodas.
2. Atliktos analizės pagrindu nustatyti reikalavimai būsimai vartotojo sąsają modeliuojančiai sistemai bei parengta šios sistemos projektinė specifikacija.
3. Sistemos reikalavimų modelyje sudarytos panaudojimo atvejų, veiklos ir dalykinės srities klasių diagramos bei sistemos vartotojo sąsajos modelis, užtikrino reikalavimų sistemai specifikacijos parengimą bei pilnaverčių projektinių sprendimų priėmimą.
4. Sistemos, skirtos vartotojo sąsajai modeliuoti, projekte paruoštos sistemos panaudojimo atvejų bendradarbiavimo diagramos, apibrėžtas sistemos elgsenos modelis, sukurtas DB modelis, aprašyta sistemos architektūra sudarė pilną projektinę specifikaciją, pakankamą jos programinei realizacijai.
5. Sukurtai sistemai aprašyti testavimo duomenys, t.y. pateiktas tekstinis duomenų šaltinio aprašymas bei jo specifikacija DB-ėje. Aprašyti reikalavimai sistemos funkcionavimo palaikymui bei sudaryta sistemos naudojimo instrukcija.
6. Sukurtas prototipas yra tiriamojo lygio, t.y. jį gali naudoti sistemos analitikas KIS modeliavimui. Prototipo darbo rezultatas – tai sumodeliuota vartotojo sąsaja, duomenys apie kurią saugomi DB-ėje.
7. Sukurtas modulis leidžia modeliuoti vartotojo sąsajos elementus iš duomenų bazės pasirinktam duomenų šaltiniui. Duomenų šaltinio apdorojimo etapai vaizduojami atskirais *MS Visio 2000* dokumento langais, t.y. grafinė sistemos aplinka integruota į *MS Visio 2000* programinį paketą. Reikalingų programinių modulių sudarymui bei jų integravimui naudojama *MS Visual Basic for Applications* programavimo kalba.

8. Sukurta sistema bus integruojama su kitais programiniais moduliais, atliekančiais kuriamos taikomosios informacinės sistemos projektavimo funkcijas (įvedamų ir išvedamų informacinių srautų specifikavimą, DB schemos sudarymą ir pan.), , kurie kartu atliks CASE priemonės, apimančios reikalavimų sistemai specifikavimo bei projektavimo funkcijas.

6. Literatūra

1. **Butkienė R**., Informacijos sistemai keliamų funkcinių reikalavimų specifikacijos metodas. Daktaro disertacija. Kaunas, 2002.

2. **Gudas S., Sabaliauskaitė G.**, Organizacijų veiklos modeliavimas valdomų procesų metodu. Informacijos mokslai 2001; 19.

3. **Butleris R., Danikauskas T.**, Reikalavimo specifikavimo *Oracle* CASE terpėje plėtra. Informacijos mokslai 2001; 19.

4. **Hackos J.T., Redish J.C.**, User and Task Analysis for Interface Design. John Wiley & Sons, Inc., 1998.

5. **Raskin J.**, The Humane Interface. An Imprint of Addison Wesley Longman, Inc., 2000.

6. **Homrighausen A., Six H-W., Winter M.**, Round-Trip Prototyping Based on Integrated Functional and User Interface Requirements Specifications, University of Hagen, Germany, 2002.

7. **Kösters G., Six H-W, Winter M**., Coupling use cases and class models as a means for validation and verification of requirements specifications. Requirements Eng 2001; 6(1):3–17.

8. **Kösters G., Six H-W, Voss J**., Combined analysis of user interfaceand domain requirements. In: Proceedings of the second IEEE international conference on requirements engineering, Kolorado Springs, 1996.

9. **Jacobson I, Booch G, Rumbaugh J.**, The unified software development process. Addison-Wesley/ACM Press, Reading, MA, 1999

10. **Bäumer D, Bischofberger W, Lichter H, Züllighoven H**., User interface prototyping: concepts, tools, and experience. In: Proceedings of ICSE-18, Berlin, Germany, 25–29 March 1996, pp 532-541.

11. **Davis A.**, Software prototyping. In: Yovits M, Zelkowitz M. (eds), Advances in computers, Vol 40. Academic Press, San Diego, 1995, pp 39-63.

12. Object Management Group. Unified Modeling Language specification. Version 1.3, OMG, June 1999.

13. **Behrens H.**, Requirements Analysis and Prototyping using Scenarios and Statecharts, Germany, 19-25 May, 2002.

7. Terminų ir santrumpų žodynas

CASE – Computer-Aided Software Engineering

DŠ – duomenų šaltinis

ER – esybių-ryšių (modelis, diagrama)

FR – funkcionalumo rezultatas

GUI – Graphical User Interface

IS – informacinė sistema

KIS – kompiuterizuojama informacinė sistema

SOK – srities objekto kintamasis

UML – Unified Modeling Language

VSA – vartotojo sąsajos analizė

WUI – Windows User Interface

8. Summary

The target of this project was to generate a prototype, which would model the elements of the user interface based on the information flow specification. The system is programmed using *MS Visual Basic for Applications*, its' graphical environment is *MS Visio 2000* and the elements of GUI for modeling are taken from *MS Visio 2000* stencil WUI (wui.vss).

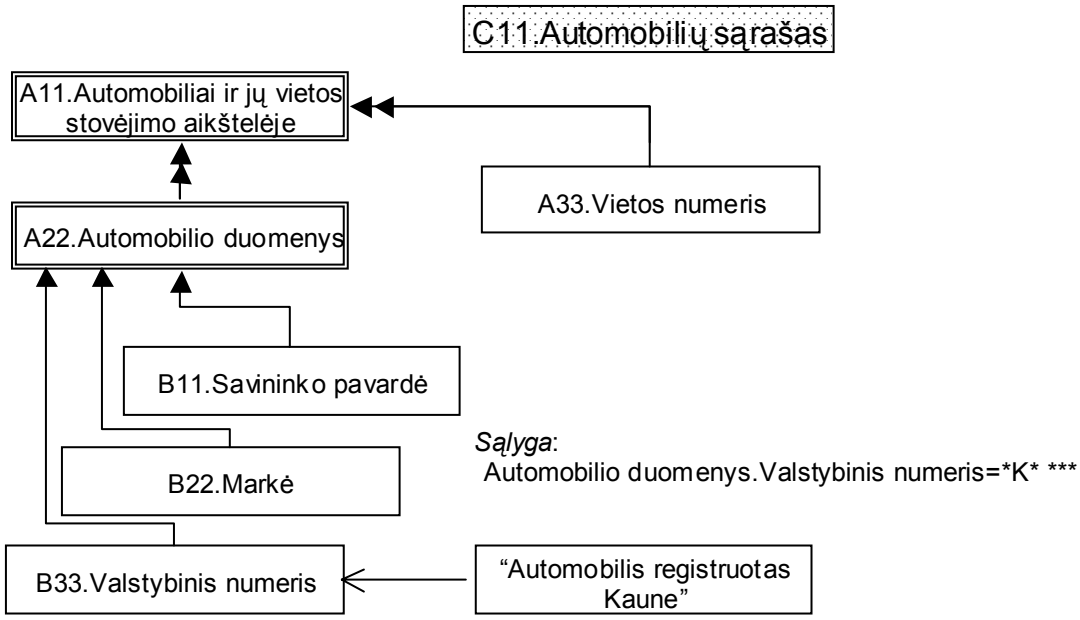
The created module allows for easy modeling of graphical user interface elements for the data source, which is selected from the database. The processing stages of the data sources are illustrated using *MS Visio 2000* documents' windows, – one window for each stage. In the database, the elements of GUI are associated with the data source processing stages they are related to, i.e. the ones that they are present in. It means, that when working with the system, after selecting the desired data source and its processing stages one wants to model, the system generates the forms with associated GUI elements.

This work was being done in the order of the scientific group of Information systems design, which analyses the design of the Information systems in the wide context of requirements engineering, process and data structures specification.

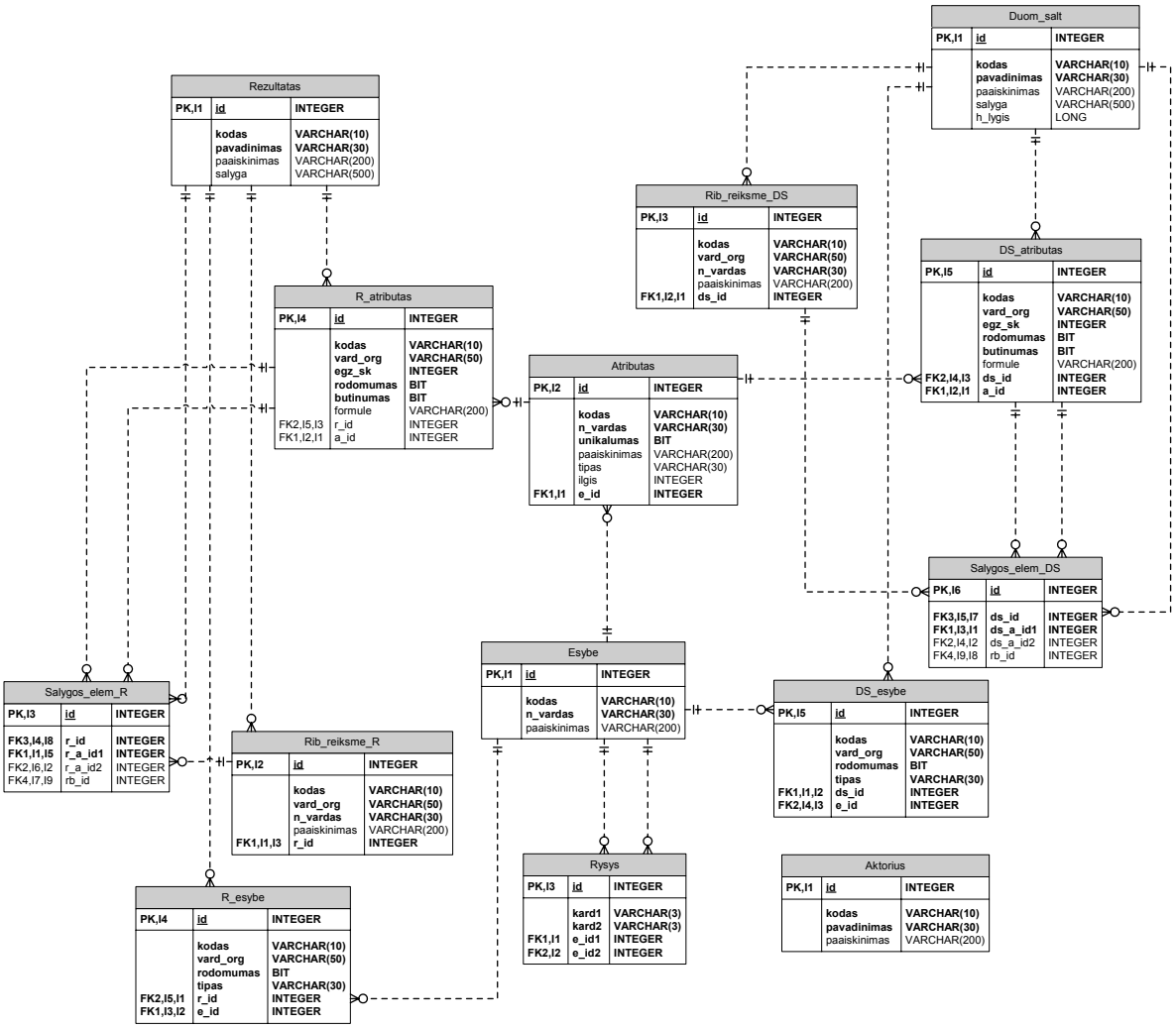
This designed and realized model is going to be a part of a CASE tool meant for designing computerized information systems. Once the CASE tool is finished, it is planned to use it for educational purposes in Kaunas University of Technology in the department of Information systems. It is going to help widen the knowledge in engineering functional requirements raised for computerized information systems. And of course it also will be used for designing computerized information systems.

It is also very important to notice, that *MS Visio 2000* has quite a space for development, it can be used to integrate other modules and applications.

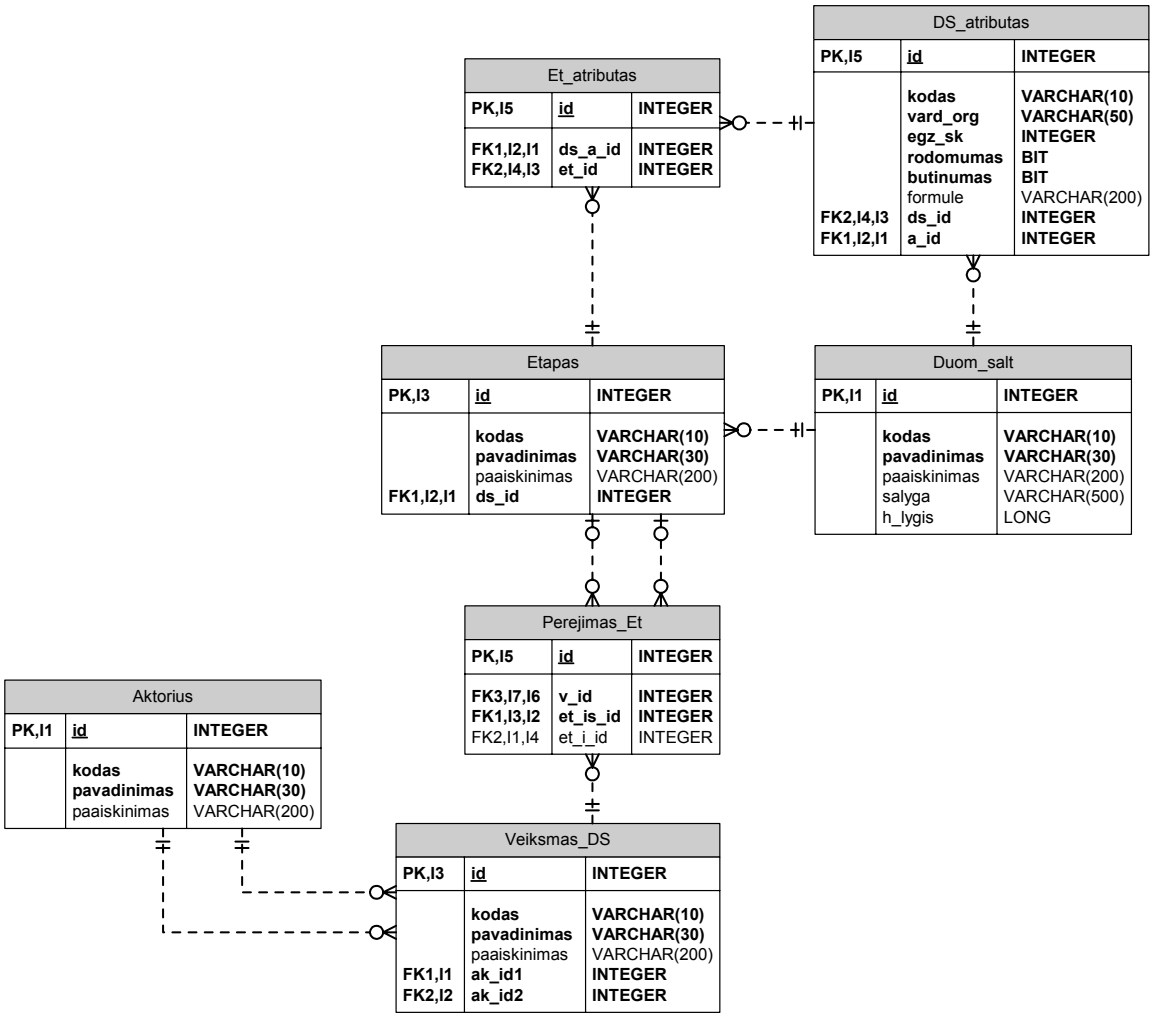
9. Priedai



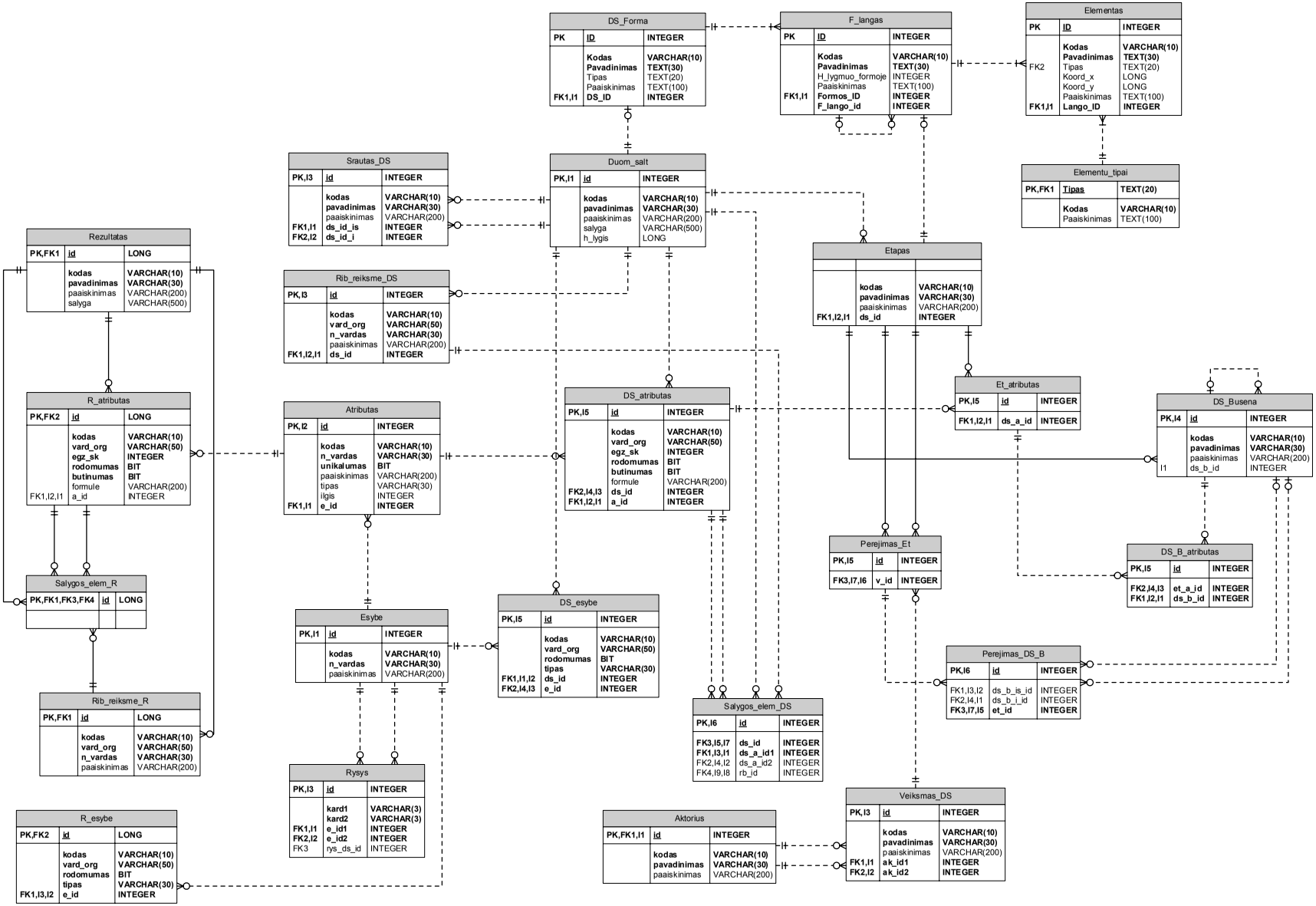
1 pav. KIS funkcionalumo rezultato specifikavimo pavyzdys.



2 pav. Duomenų šaltinių ir rezultatų struktūros specifikacijos saugyklos schema.



3 pav. Duomenų šaltinių apdorojimo specifikacijos saugyklos schema.



4 pav. Visa meta duomenų bazė.

Lentelė nr. 1. Meta duomenų 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		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.
7e_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>. 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: 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 – 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.
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.
id	int	+	+	+	Unikalus įrašo id, kuri 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.
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, 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.
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.
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.
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.

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.
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.
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
Veiksmas_DS					Lentelėje saugoma informacija apie visus galimus duomenų šaltinių apdorojimo veiksmus.
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 veiksmą.
ak_id1	int		+		Aktoriaus id, kuris yra agentas t.y. atlieka veiksmą arba jį inicijuoją.
ak_id2	int		+		Aktoriaus id, kuris yra veiksmo rezultatų (srauto) gavėjas.
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		+		Vardas, kuris gali būti paimtas iš nagrinėjamo veiklos konteksto arba sukurtas paties vartotojo.
paaiskinimas	varchar 200				Detali informacija apie aktorių.