KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Birutė Misevičiūtė

Kompiuterizuotos darbo vietos funkcionalumo specifikavimo modelis

Magistro darbas

Darbo vadovas
doc. R. Butleris

Kaunas, 2004

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. R. Butleris
2004 05 25

Kompiuterizuotos darbo vietos funkcionalumo specifikavimo modelis

Magistro darbas

Kalbos konsultantė
Lietuvių kalbos katedros lektorė
dr. J. Mikelionienė
2004 05 25

Vadovas
doc. dr. R. Butleris
2004 05 25

Konsultantas
Informatikos fakulteto doktorantas
T. Danikauskas
2004 05 25

Recenzentas
doc. S. Maciulevičius
2004 05 25

Atliko
IFM-8/2 gr. stud.
B. Misevičiūtė
2004 05 25

Kaunas, 2004

TURINYS

1 Įvadas

Didėjant informacijos sistemų (IS) atliekamam funkcionalumui, kuris apima vis daugiau organizacijų veiklos, bei įvertinant tai, kad organizacijos padaliniai vienas nuo kitos geografiškai yra vis labiau nutolę, išryškėja, kad atliekant IS projektavimą reikia atskirai atlikti kompiuterizuotų darbo vietų projektavimą [14]. Šiuo metu egzistuojančių technologijų gausoje atrodo neturėtų būti sunku surasti metodą ar įrankį leidžiantį atlikti kompiuterizuotų darbo vietų sudėties specifikavimą.

Dalis siūlomų šios problemos sprendimų dažniausiai susiveda į bendrą vartotojo sąsajos projektavimą. Ir šiuo atveju kur kas didesnis dėmesys skiriamas vartotojo ekraninių formų modeliavimui, jų funkcionalumo aprašymui ir tik maža dalimi paliečia kompiuterizuotos darbo vietos projektavimą. O jei ir paliečiamas šis klausimas, tai dažniausiai sprendimas susiveda į vienos universalios darbo vietos suprojektavimą. Turint šį projektą didžioji dalis reikalavimų vis dar paliekami sistemos programuotojo vertinimui, kad šis savo nuožiūra padarytų sprendimą. Šis sprendimas, nors dažniausiai to tiesiogiai ir neįvardinant, tampa prototipiniu sistemos vartotojo sąsajos variantu ir tik atlikus korekcijas pagal vartotojo išsakytas pastabas sistemą galima įdiegti.

Šio darbo tikslas – išanalizuoti kompiuterizuotos informacinės sistemos (KIS) konteksto ir kompiuterizuotos darbo vietos (KDV) funkcionalumo specifikavimo principus ir apribojimus, suprojektuoti KDV funkcionalumo specifikavimo modelio sudarymą informacijos srautų specifikacijos pagrindu ir patikrinti CASE įrankyje realizuoto modelio teisingumą eksperimentiniu būdu. Sukurtą sistemą įvertinti kitų duomenų modeliavimo sistemų kontekste.

Analizės dalyje išnagrinėti veiklos konteksto specifikavimui ir kompiuterizuotos darbo vietos projektavimui naudojami metodai ir įrankiai, pateiktas šių metodų įvertinimas pagal pasirinktus kriterijus. Kadangi nagrinėjamas modelis yra funkcinių reikalavimų specifikacijos metodo sudedamoji dalis, pateiktas trumpas šio metodo aprašymas ir jo pagrindiniai privalumai.

Trečiojoje dalyje aprašytas pasiūlytas kompiuterizuotos darbo vietos funkcionalumo specifikavimo modelis, jo sudarymo principai ir taisyklės, naudojama notacija, vieta funkcinių reikalavimų specifikavimo metodo kontekste.

KDV funkcionalumo specifikacijos modelio realizacijos skyriuje pateikti fragmentai iš programinio produkto kūrimo metu sudarytų dokumentų. Plačiau pateiktos reikalavimų ir architektūros specifikacijos, kurios sudarytos reikalavimų išgavimo, bei sistemos projektavimo metu.

Eksperimentinėje dalyje pateiktas realizuotos programinės įrangos kokybės įvertinimas, bei palyginimas su kitomis panašaus pobūdžio sistemomis.

Prieduose pateikiama straipsnio, paskelbto konferencijos pranešimo medžiagoje, kopija, taip pat antras straipsnis, parengtas publikavimui, funkcinių reikalavimų specifikavimui naudojama metaduomenų bazė ir jos aprašas, realizuoto CASE įrankio modulio vartotojo dokumentacija.

2 KDV projektavimo metodų ir įrankių analizė

Vienas iš svarbiausių taikomosios programos kūrimo etapų yra vartotojo sąsajos kūrimas. Dažniausiai nuo jos priklauso ar sistemos vartotojai priims, išmoks naudotis ir efektyviai dirbti su pačia sistema [2,15]. Taigi kuriant taikomąsias programas svarbu ne tik jos teikiamas funkcionalumas, bet ir tai, kaip jis pateikiamas vartotojui.

Programinės įrangos vartotojo sąsajos kūrimas yra ne tik santykinai didelės apimties ir sunkus, bet ir sudėtingas projektuoti, įdiegti bei modifikuoti procesas [4]. Todėl šiuo metu vartotojo sąsajos kūrimui ir projektavimui yra siūloma visa eilė įrankių, pvz., greito taikomųjų programų kūrimo (angl. RAD – *rapid application developmet*), integruoto kūrimo aplinkos (angl. IDE – *integrated development environment*), vartotojo sąsajos kūrimo modelių pagrindu (angl. MB-UID – *model based user interface development*).

Kompiuterizuotos darbo vietos (KDV) projektavimas yra traktuojamas kaip viena iš vartotojo sąsajos projektavimo fazių. KDV yra apibrėžiama kaip savarankiška informacinė sistema arba didesnės IS dalis. Ji dažniausiai susijusi su tam tikru žmogaus organizacijoje atliekamu vaidmeniu, kurio funkcijas ji padeda vykdyti. KDV dažnai būna įdiegta tam tikroje vietoje, nors pastaruoju metu pereinama prie virtualių darbo vietų, pasiekiamų per internetą.

Ir nors vartotojo sąsajos projektavimui yra siūloma daug įvairių sprendimų ir įrankių, tačiau KDV funkcionalumo specifikavimui nėra skiriamas didelis dėmesys [14]. Analizės dalyje bus apžvelgti KDV projektavimui naudojami sprendimai, pvz., unifikuotos modeliavimo kalbos (UML) pagrindu siūlomi KDV sudarymo metodai, IDEF metodologija, taip pat aptartas ir Oracle CASE metodas.

2.1 Vartotojo sąsajos kūrimo technologija modelių pagrindu

Vartotojo sąsajos (angl. UI - *user interface*) kūrimo technologija modelių pagrindu siekia pateikti aplinką, kurioje PĮ kūrėjai galėtų paprasčiau nei tradiciniuose vartotojo sąsajos(VS) kūrimo įrankiuose suprojektuoti ir įdiegti vartotojo sąsają [3]. Naudojant šį būdą, sąsajos yra sugeneruojamos iš deklaratyvių specifikacijų (modelių), kurios aprašo vartotojų atliekamas užduotis, turinį, ekranų struktūrą ir išdėstymą bei ekrano elementų roles vartotojui vykdant užduotis [5].

Trys pagrindiniai deklaratyvių vartotojo sąsajos modelių privalumai :

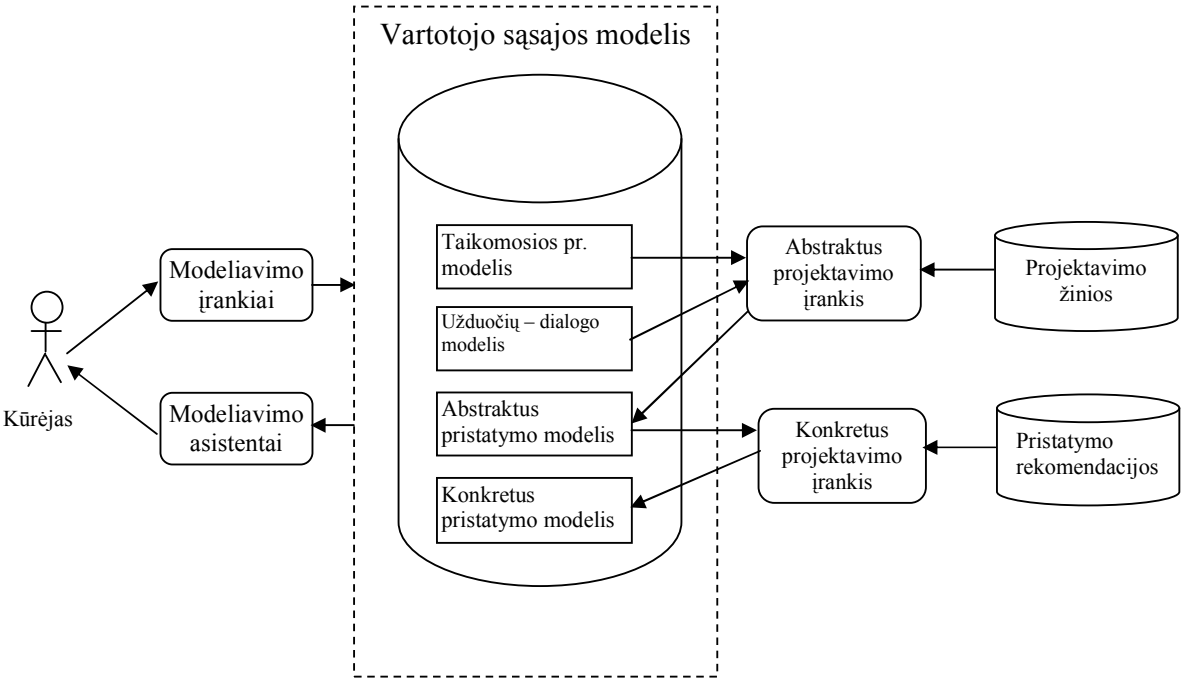
- Jie gali pateikti labiau abstraktų, nei suteikiantį kitų VS kūrimo įrankių, VS aprašą.

- Jie palengvina VS projektavimo ir įdiegimo metodų sukūrimą, nes leidžia :
    - modeliuoti vartotojo sąsają naudojant skirtingus abstrakcijos lygius;
    - tobulinti modelius;
    - pakartotinai naudoti VS specifikacijas.
- Jie suteikia reikalingą infrastruktūrą užduočių, reikalingų VS projektavimo ir įdiegimo procesams automatizuoti.

Pagrindinis vartotojo sąsajos kūrimo technologijos modelių pagrindu trūkumas yra modelių sudėtingumas ir jų notacijos, kurias dažnai būna sunku suprasti ir naudoti [20]. Taip pat dar nėra susitarimo, kuris modelių rinkinys yra tinkamiausias vartotojo sąsajos aprašui, bei kurie vartotojo sąsajos aspektai turėtų būti modeliuojami.

Vartotojo sąsajos projektavimas ir įdiegimas gali būti kartojamas daug kartų, siekiant pagerinti VS specifikaciją arba išeities tekstą. Realiai, kai kurie modeliais pagrįsti vartotojo sąsajos kūrimo įrankiai nėra labai lankstūs tobulinant išeities tekstą. VS kūrimo procese galima išskirti du subprocesus. Pirmasis – vartotojo sąsajos projektavimas, kurio rezultatas yra sukurtas VS modelis. Antrasis – VS įdiegimas, kuris baigiasi vykdomąja VS.

Vartotojo sąsajos projektavimo, naudojant modeliais pagrįstą vartotojo sąsajos kūrimo aplinką, schema yra pateikta 1 pav.



1 pav. Vartotojo sąsajos projektavimas naudojant modeliais grįstą VS kūrimo aplinką

Modeliavimo įrankiai – paprastai tai grafinė aplinka, kuri palengvina sudėtingų VS modelių sudarymą bei leidžia juos redaguoti.

Modeliavimo asistentai – pagrindinė jų funkcija yra modelio tikrinimas, kuris suteikia projektuotojui grįžtamąjį ryšį apie projektavimo procesą.

Taikomosios programos modelis – aprašo taikomosios programos, susijusios su vartotojo sąsaja, savybes.

Užduočių dialogo modelis – aprašo užduotis, kurias vartotojas gali atlikti naudodamasis taikomąja programa, bei ryšius tarp tų užduočių.

Abstraktus pristatymo modelis - VS aprašoma abstrakčių objektų sąvokomis. Pateikia konceptualų struktūros aprašymą ir matomų vartotojo sąsajos dalių elgesį.

Konkretus pristatymo modelis – detaliai aprašomos matomos VS dalys, taip pat pateikiama, kaip sudaryta vartotojo sąsaja.

Vartotojo sąsajos modeliai gali pateikti VS įvairiais abstrakcijos lygiais. Taigi, iš pradžių VS yra aprašoma labai abstrakčiu modeliu, kuris galiausiai yra transformuojamas į konkretų modelį. Kūrėjai bet kada, netgi po VS sugeneravimo, gali sugrįžti į vartotojo sąsajos kūrimo aplinką ir tobulinti modelius.

2.2 UML panaudojimas KDV projektavimui

2.2.1 Spalvoto UML panaudojimas KDV projektavimui

Peteris Kodas(Peter Coad) sukūrė taip vadinamąjį 4 spalvų modelį [16]. Pirmiausia šis modelis buvo pasiūlytas kaip greita intuityvaus interneto programų vartotojo sąsajos modeliavimo technologija, nors ji gali būti pasiūlyta ir tradicinėms IS projektuoti. Ši technologija yra objektiškai orientuota. 4 spalvų modelis užtikrina greitą dalykinės srities analizę, sudarant korektišką klasių diagramą, kurioje galima greitai atskirti ir perprasti dalykinės srities objektų sluoksnius.

Sistemos branduolį sudaro 4 archetipų objektai arba, kitaip tariant, klasės, turinčios keturis stereotipus. Kiekvienas stereotipas turi savo spalvą, todėl sudarytame spalvotame dalykinės srities klasių modelyje lengvai išsiskiria dalykinės srities sluoksniai. Išsiskiriantys dalykinės srities sluoksniai vaizdžiai atskleidžia sistemos elgseną, o spalvinis modelis, anot autorių, patyrusiam projektuotojui leidžia lengvai patikrinti jo pakankamumą, korektiškumą bei pilnumą.

Egzistuoja 4 spalvų modelio objektų stereotipai:

- Atliekama rolė (geltona spalva);
- Laiko momentas/Intervalas (rožinė spalva);
- Asmuo/Vieta/Daiktas/ (žalia spalva);
- Aprašymas (melsva spalva);

Spalvos, anot kūrėjų [16], parinktos tokios, kad būtų galima dirbti su darbo grupe, ant lentos naudojant kanceliarinius užrašų lapelius. Kiekvienas stereotipas turi jau iš anksto žinomų metodų ir atributų aibę. Todėl sudarytas klasių modelis dėl savo spalvinės charakteristikos ir iš anksto žinomos dalinės elgsenos analitikui informatyviai parodo, kokios modeliuojamos sistemos elgsenos galima tikėtis.

Atliekama rolė – šiuo atveju tai yra rolė, kurią atlieka objektas, priklausantis asmuo/vieta/daiktas stereotipui, tam tikru momentu atlikdamas tam tikrą veiksmą su tam tikru tikslu. Pavyzdžiui galėtų būti įmonė, turinti daug klientų. Klientas – tai rolė, kurią atlieka asmuo arba organizacija. Šis stereotipas dažniausiai gali turėti rolės įvertinimo metodą. Pavyzdžiui, gaminio surinkėjo rolė gali turėti metodą *SurinktuGaminiuKiekis*(), taip pat gali būti metodai, užklausiantys, ar tas pats objektas gali atlikti šiuo metu kitą rolę. Pavyzdžiui, ar dabar dirbantis asmuo prie vienos konvejerio linijos gali pereiti dirbti prie kitos linijos.

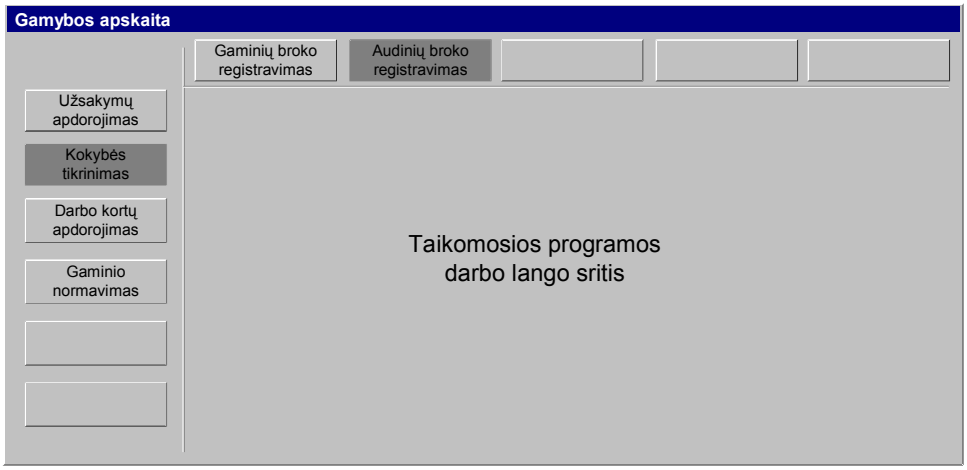
Laiko momentas/intervalas – laiko momentas reiškia tai, kas turi būti atlikta ar įvykti veikloje tam tikru laiko momentu ar per tam tikrą laiko intervalą. Taip netiesiogiai įvardijami veiksmai, užduotys ir kita vykstanti dalykinėje srityje veikla. Vėl galima pateikti paprasčiausius pavyzdžius: užsakymo atlikimas - tai laiko periodas per kurį tiekėjas įsipareigoja atlikti užsakyme numatytus darbus ar pateikti užsakovui užsakytą produktą. Laiko momento stereotipas dažniausiai turi metodus, leidžiančius jam sužinoti jo būseną, jo įvykimo laiką ar jį charakterizuojančias savybes. Pavyzdžiui, metodas, nusakantis ar užsakymas yra skubus, ar užsakymas atliktas.

Asmuo/vieta/daiktas – šiam stereotipui gali priklausyti visi daiktavardžiai, kuriuos galima įvardinti nagrinėjamoje dalykinėje srityje. Jam būdingi metodai, leidžiantys nusakyti paties objekto savybes, jo būseną tam tikrais laiko momentais.

Aprašymas – aprašymas gali būti suprantamas kaip objektų klasifikatorius. Tai gali būti gaminio kodas, kuris savyje saugo informaciją apie gaminio modelį, spalvą ir kitas savybes. Kliento rekvizitų rinkinys taip pat gali būti įvardijamas kaip aprašymas. Aprašymo objektas gali atlikti užduotis tokias, kaip gamybos ataskaitas, t. y. pateikti visos pagamintos produkcijos kiekį per tam tikrą laiko momentą.

Paties spalvoto klasių modelio sudarymą geriausia pradėti nuo identifikuotų veiklos rolių. Toliau galima identifikuoti realius daiktus, kas tas roles atlieka. O rolės tarpusavyje bendradarbiauja, kai tarp jų tam tikru metu ar laiko intervalais vyksta atitinkama veikla. Aprašymo klasės padeda aprašyti ir detalizuoti reikiamu aspektu vykstančią veiklą. Prisilaikant šios modelio sudarymo strategijos bei klasikinių UML klasių modelio sudarymo taisyklių ir jam taikomų apribojimų bus galima pilnai atlikti detalų dalykinės srities spalvoto klasių modelio sudarymą.

Šią KDV technologiją kūrėjai siūlo naudoti tik interneto taikomųjų programų meniu turiniui formuoti. Navigacinis meniu išdėstymas yra perimtas iš IMB Lotus įrankio aplinkos. Be to, pasirinkta dviejų matmenų meniu elementų išdėstymo struktūra (matricinė). Kairėje lango pusėje vertikaliai išdėstomi pirmos eilės elementai, o viršutinės lango dalies dešinėje pusėje išdėstomi antros eilės elementai (žr. 2 pav.).



2 pav. Matricinė meniu elementų išdėstymo struktūra

Šis meniu išdėstymas pilnai gali būti naudojamas ir standartinėse vartojo darbo vietose. Siūloma, kad meniu neturėtų daugiau nei 10 elementų tiek vertikalioje, tiek horizontalioje pasirinkimo juostoje, nes teigiama, kad geras meniu vartotojui suteikia pasirinkimą iš 7+/-2 elementai. Esant didesniam elementų skaičiui meniu traktuojamas kaip perkrautas ir apsunkinantis vartotojo darbą. Nors kartais teigiama ir taip, kad didesnis meniu pasirinkimo skaičius yra geriau, nei sudaryti meniu, kurio pasirinkimų gylis yra didesnis nei 3 lygiai. Taigi pasiūlytas meniu išdėstymas pavadintas „Šachmatinė lenta", nes jis vidutiniškai siūlo meniu , kurio pasirinkimų dydis yra 8x8 [17].

Koncepcija siūlo keletą strategijų, kaip pagal spalvotą klasių modelį sudaryti KDV struktūrą:

- *Į pirmos eilės meniu išdėstyti rolių užduočių etapus*, o į antros eilės meniu etapus – veiksmus arba pasirenkamus duomenis. Ši strategija pasiteisina, jei nagrinėjama dalykinė sritis turi nedaug užduočių. Didėjant užduočių skaičiui, meniu struktūra bus perkrauta dideliu etapų kiekiu, o vartotojas gali susipainioti, koks etapas kokiai užduočiai priklauso.
- *Į pirmos eilės meniu išdėstyti rolių užduotis*, o į antros eilės meniu užduočių žingsnius arba etapus. Ši strategija taip pat tinka tik nedidelei dalykinei sričiai, nes didėjant užduočių skaičiui meniu struktūra bus nepatogi. Be to, jei bus norima į šį etapą įtraukti

ir keleto rolių užduotis, meniu bus dar labiau apsunkintas. Žinoma, dalį problemos galima išspręsti jei KDV būtų formuojamas kiekvienai rolei atskirai.

- *Į pirmos eilės meniu išdėstyti roles,* o į antros eilės meniu išdėstyti rolių atliekamas užduotis, kas gerokai supaprastina meniu struktūrą. Vėl gi gali iškilti problemų, nes organizacijoje užduotis gali būti atliekama kelių vartotojų, t.y. vienam fiziniam asmeniui gali tekti atlikti darbus, priklausančius kelioms rolėms. Kaip pavyzdį galima pateikti siuvyklos sandėlio darbuotoją, kuris gali būti atsakingas už produkcijos išsiuntimą užsakovui – klientų aptarnavimo vadybininko rolė, ir sandėlio apskaitos vedimą – sandėlininko rolė (pirminė sandėlio darbuotojo rolė), bei gautų audinių mėkalo fiksavimas – kokybės kontrolieriaus rolė. Esant tokiam atvejui sistemos vartotojui gali būti sunku identifikuoti jo rolę.
- *Į pirmos eilės meniu išdėstyti Asmenį/vietą/daiktą,* o į antros eilės meniu išdėstomi visa tai, kas vyksta dalyvaujant šiam objektui nagrinėjamoje veikloje. Šią strategiją galima pasirinkti, jei objektai dalyvaujantys veikloje turi sąlyginai daug ryšių su sistemoje esančiomis rolėmis. Tai parodo kokia yra objekto svarba nagrinėjamoje dalykinėje srityje. Taigi į pirmos eilės meniu siuvimo įmonėje būtų galima įdėti, pavyzdžiui, Fiziniai asmenys, Juridiniai asmenys, Įmonė. Tiek fiziniai, tiek juridiniai asmenys gali atlikti keletą rolių: tiekėjai, užsakovai, paslaugų teikėjai, darbuotojai, o įmonė turėdama savo vidinius padalinius, gali atlikti įvairias roles: apskaitos, gamybos, sandėliavimo.
- *Į pirmos eilės meniu išdėstyti Laiko momentas/intervalas objektus,* o antro eilės meniu išdėstyti laiko momentas/intervalas objektus, kurie detalizuoja pirmos eilės meniu objektus. Ši strategija tinkama, jei padarome prielaidą, kad visi veiklos dalyviai žino savo roles ir vykdomą veiklą. Norint naudoti šią strategiją sudaromas spalvotas UML klasių modelis, kurio laiko objektai turi būti detalizuoti į smulkesnius to paties stereotipo objektus. Kūrėjai pristato bendrus laiko momentų/intervalų objektų detalizavimo principus [11].

Apžvelgtos technologijos pagrindiniai privalumai yra tai, kad iš sudaryto klasių modelio galima sudaryti KDV struktūrą. Be to ši koncepcija pasiūlo net 5 KDV sudarymo strategijas, nors kiekviena iš jų turi savų teigiamų ir neigiamų bruožų.

2.2.2 Panaudojimo atvejais grįstas KDV projektavimas

Vienas pagrindinių UML kalbos modelių yra panaudojimo atvejų (Use case) modelis [5,8]. Kaip sudarinėti panaudojimo atvejų modelį yra keletas strategijų. Priklausomai nuo to, kokiu tikslu sudarinėjamas modelis, gali būti sukurtos [9]:

Į vartotoją orientuotas modelis – modelio akcentas yra charakterizuoti visus aktorius ir jų grupes.

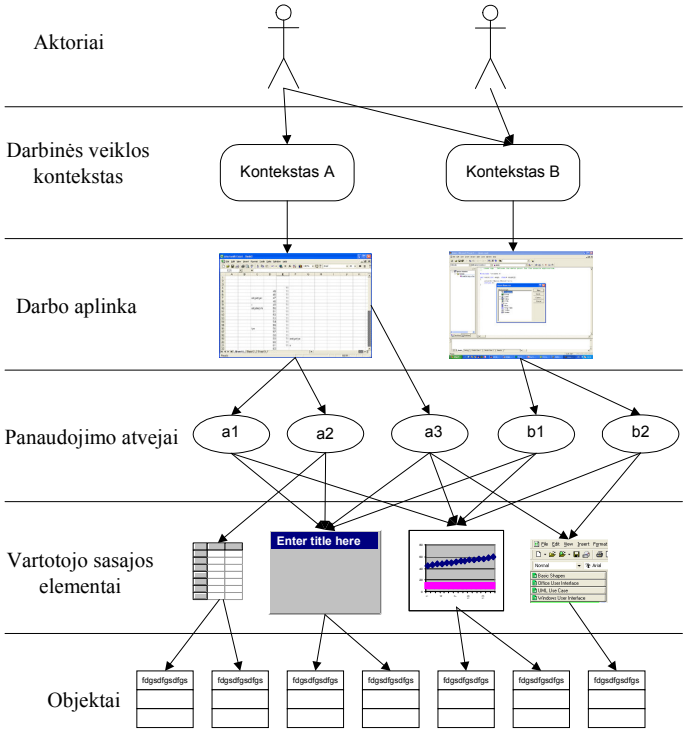
Į tikslus orientuotas modelis – pateikiami aukščiausio prioriteto tikslai siekiant, kad kuriama sistema tenkintų vartotojo keliamus tikslus.

Į veiklą orientuotas modelis – šiame modelyje specifikuojama darbuotojo veikla, jos aplinka.

KDV galima projektuoti naudojant panaudojimo atvejų (PA) diagramą, nes joje išskiriami funkciniai IS objektai (panaudojimo atvejai), taip pat kiekvienam vartotojui priskiriami jo atliekami panaudojimo atvejai. 3 paveikslėlyje pateikiama PA modelio ir darbo aplinkos elementų sąveikos struktūra. Kiekvienas aktorius gali turėti vieną ir daugiau savo darbo situacijų. Kiekviena darbo situacija ryšiu 1:1 turi sąsają su darbo aplinka, kuri jau kompiuterizuota arba siekiama kompiuterizuoti. Kiekvienas vartotojas savo darbo aplinkoje gali dirbti ir naudoti visą informaciją, kuri reikalinga jo veiklai, t. y. kiekvienas aktorius per darbo aplinką sąveikauja su aibe panaudojimo atvejų. Tas pats panaudojimo atvejis gali būti pasiekiamas iš skirtingų darbo aplinkų. Kiekvieno panaudojimo atvejo aplinka naudoja vieną ar keletą vartotojo sąsajos elementų, kurie savo ruožtu naudoja informacijos sistemos objektus. Panaudojimo atvejį galima detalizuoti į keletą smulkesnių panaudojimo atvejų, naudojant dekompozicijos principus [9,10]. Dekompozicijos panaudojimas suteikia keletą privalumų. Pirmiausia dekomponavus sąlyginai didelius panaudojimo atvejus išauga suprantamumas.

Dekomponuotiems panaudojimo atvejams gali būti pritaikytas daugkartinio panaudojimo principas ir tai gali būti panaudota kuriant pačią IS.

KDV projektavimui tinkamiausias būtų į veiklą orientuotas panaudojimo atvejų modelis, nes jis padengia vartotojo darbo sritis, kurias siekiama kompiuterizuoti. Turint sudarytą panaudojimo atvejų modelį kiekvienam vartotojui galima sudaryti atskirą KDV struktūrą. Meniu punktus atitiktų panaudojimo atvejai. Jei panaudojimo atvejams buvo pritaikyta dekompozicija, galima įvesti ir meniu hierarchiją. Panaudojimo atvejų modelis nėra pritaikytas hierarchijos atvaizdavimui. Jei vartotojo veikloje yra keletas nepriklausomų veiklos kontekstų, tai į pirmąjį meniu hierarchijos lygį gali būti iškelta kiekvieno veiklos konteksto darbo aplinka.



3 pav. PA modelio ir darbo aplinkos elementų sąveikos struktūra

Taigi KDV projektavimą, pasinaudojant panaudojimų atvejų modeliu, galima įvardinti kaip klasikinį UML teikiamų priemonių pritaikymą.

2.3 KDV projektavimas pagal Oracle CASE metodą

Oracle CASE metodas kaip pagrindinę priemonę KDV sudarymui naudoja procesų diagramą, kurioje kiekvienam organizacijos vienetui ar aktoriui, yra išskiriamas atskiras veiksmų takelis apibrėžti veiksmams, kurios ji atlieka. O KDV sudaroma kiekvienam takeliui skiriant atskirą meniu punktą, kuris turi dvi submeniu dalis: įvedimo formos, ataskaitos.

Procesų diagramos pagrindu yra sugeneruojama ir funkcijų hierarchija, kurios žemiausio lygio funkcijos bus realizuotos ekraninių formų ir ataskaitų pavidalu. Funkcijų hierarchiją galima sudaryti ir neturint procesų diagramos. Oracle CASE metodo apraše R. Barker yra pateikęs pilną funkcijų hierarchijos sudarymo dokumentaciją [13].

Oracle Designer įrankis taip pat leidžia atlikti KDV projektavimą neturint procesų diagramos, o tiesiogiai sudarant KDV iš funkcijų hierarchijos [10]. Siūlomas Designer sprendimas nėra tobulas, nes jis meniu siūlo sudaryti tik suskirstant žemiausio lygio funkcijas į įvedimo formas ir ataskaitas.

Funkcijų hierarchiją R.Barker siūlo formuoti naudojant „Iš viršaus į apačią“ *(Top-down*) plius „Iš apačios į viršų“ (*buttom-up*) principus[13]. Pirmiausia yra atliekamas „Iš viršaus į apačią“ funkcijų hierarchijos modeliavimas. Pirmoji funkcija – tai sakinys (išreikštas

kaip veiklos funkcija), atspindintis visos veiklos esmę – veiklos strategiją arba misiją. Toliau yra išskiriami tarpusavyje nepriklausomi komponentai, kurie realizuoja viršutinę funkciją (paprastai siūloma išskirti 7-8 funkcijas žemesniame lygyje). Šis procesas kartojamas kiekvienam iš apibrėžtų komponentų. Kiekviename lygyje, siekiant užtikrinti funkcijos įvykdymo pilnumą ir neperteklišKumą, yra užduodami tokie klausimai:

- ar nereikia nieko daugiau, išskyrus identifikuotas funkcijas, įvykdyti tėvinę funkciją?
- ar reikalingos visos ir tik šios funkcijos įvykdyti tėvinę funkciją?

Hierarchija yra dekomponuojama tol, kol pasiekiamas reikalingas detalumo lygis. Atlikus „Iš viršaus į apačią“ modeliavimą, reikia atlikti modelio tikrinimą, naudojant hierarchijos sudarymą „Iš apačios į viršų“.

„Iš apačios į viršų“ technika naudoja veiklos tikslus. Kiekvienam tikslui yra identifikuojamos jį įvykdančios funkcijos, kurios detalizuoja aukščiausio lygio funkciją. Viršutinė, top funkcija, apibrėžiama kaip esminis veiklos tikslas. Suformavus šią hierarchiją, peržiūrima, ar nėra neidentifikuotų akivaizdžių organizacijos tikslų (pavyzdžiui, tarp siuvimo įmonės gamybos apskaitos identifikuotų tikslų nėra pasiūtų gaminių registravimo tikslo). Suradus tokius tikslus, hierarchija papildoma reikalingomis funkcijomis.

Griežtai neapibrėžiama, kada turi būti baigtas hierarchijos dekomponavimas, tačiau siūloma detalizuoti toliau, jeigu funkciją gali atlikti daugiau nei vienas aktorius arba jos pavadinimą sudaro keli sakiniai, daug jungtukų „ir“ bei keli veiksmažodžiai. Jeigu identifikuota funkcija yra aiški ir trumpa, siūloma sustoti aukštesniame lygyje.

2.4 IDEF metodologija

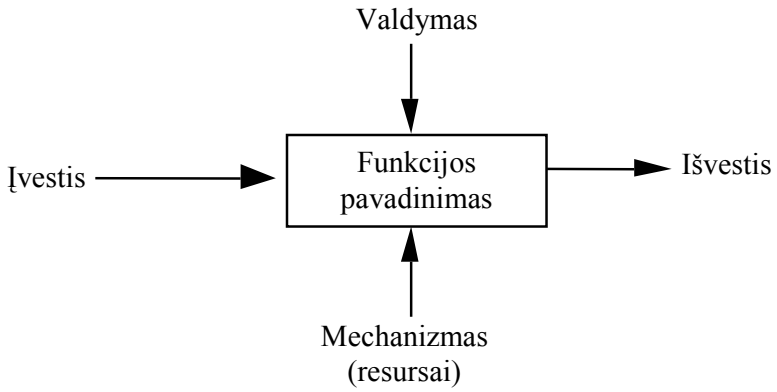
IDEF (*Integration DEFinition language*) yra grupė modeliavimo metodų, naudojamų organizacijos procesams modeliuoti. IDEF sukūrė JAV oro pajėgos integruotų kompiuterių gamybai. IDEF sudaro 16 metodų, pradedant IDEF0 ir baigiant IDEF14(įtraukiant IDEF1 ir 1X). Šie metodai skirti įvairiai informacijai modeliuoti: objektiškai orientuotam projektavimui (IDEF4), simuliacijai (IDEF2), vartotojo sąsajos projektavimui (IDEF8), organizacijos modeliavimui (IDEF12) ir kiti.

Praktiškai yra naudojami tik trys šios grupės metodai: IDEF0 – funkciniam modeliavimui, IDEF3 – procesų modeliavimui ir IDEF1X – duomenų modeliavimui.

2.4.1 Funkcijų modeliavimas, naudojant IDEF0 metodą

IDEF0 yra pagrįstas struktūrinės analizės ir projektavimo technika (*SADT – structured Analysis and Design Technique*)[18]. IDEF0 apima tiek grafinio modeliavimo kalbos (sintaksės ir semantikos), tiek visapusišką modelių kūrimo metodologijos aprašymą. Kuriant naujas sistemas, šis būdas pirmiausiai gali būti naudojamas nustatyti reikalavimams ir specifikuoti funkcijoms, o po to suprojektuoti įdiegimą, kuris atitinka reikalavimus ir vykdo nurodytas funkcijas.

IDEF0 panaudojimo sistemos aprašymui rezultatas yra modelis, kuris susideda iš eilės hierarchinių diagramų, teksto ir terminų žodyno susietų tarpusavyje. Du pagrindiniai modeliavimo komponentai yra funkcijos(diagramose vaizduojami kaip stačiakampiai) ir duomenys bei objektai susiję su šiomis funkcijomis(vaizduojami kaip rodyklės) žr. 4 pav.



4 pav. Konceptualusis IDEF0 modelis

Funkcija – veikla, procesas arba transformacija įvardinta veiksmažodžiu arba fraze su veiksmažodžiu, kuris apibrėžia, kas turi būti atlikta.

Įvesties rodyklė – rodyklių klasė išreiškianti IDEF0 įvestį, pvz., duomenys arba objektai kurie yra funkcijos transformuojami į išvestį. Įvesties rodyklės yra susietos su kairiąja IDEF0 stačiakampio puse.

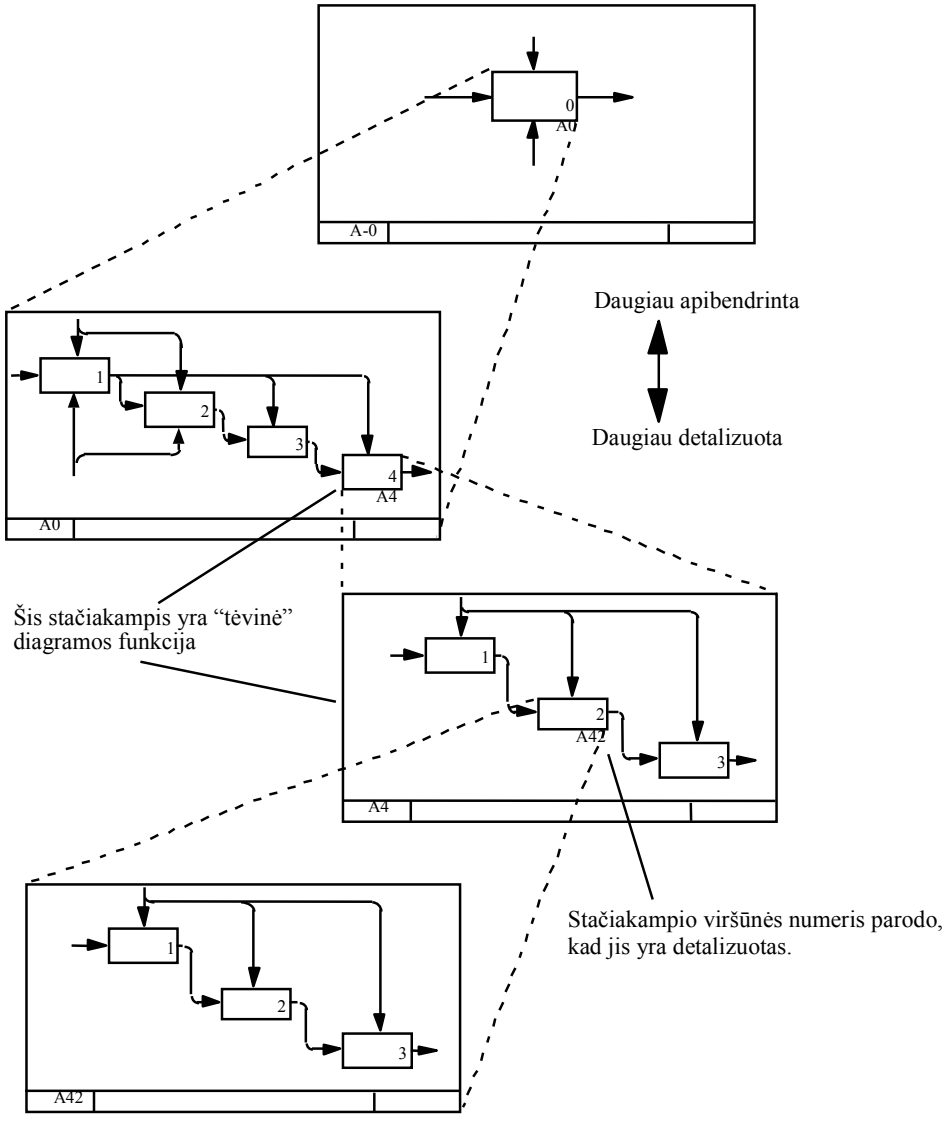
Išvesties rodyklė – rodyklių klasė, kuri išreiškia IDEF0 išvestį, pvz., duomenys arba objektai, kurie yra funkcijos rezultatai. Išvesties rodyklės yra susietos su dešiniąja IDEF0 stačiakampio puse.

Mechanizmų rodyklė – rodyklių klasė, kuri išreiškia IDEF0 mechanizmą, pvz., funkcijos atlikimui naudojamos priemonės. Mechanizmo rodyklės susietos su IDEF0 stačiakampio apačia.

Valdymo rodyklė – rodyklių klasė, kuri išreiškia IDEF0 valdymą, pvz., sąlygas, reikalingas gauti teisingą išeigą. Duomenys ar objektai, modeliuojami kaip valdymas, gali būti

transformuojami funkcijos sukuriant išvestį. Valdymo rodyklės yra susietos su IDEF0 stačiakampio viršumi.

Naudojantis IDEF0 metodu detalizavimas vyksta sukuriant naujas diagramas. Pirmiausia sukuriama diagrama, kurioje yra tik viena Top funkcija ir apibrėžia visos veiklos kontekstą. Ši funkcija gali būti detalizuojama sukuriant naują diagramą („vaikinę“), kurioje specifikuojamos detalizuojančios funkcijos. Kiekvienai „vaikinės“ diagramos funkcijai gali būti sukurta nauja diagrama (5 pav.).



5 pav. IDEF0 modelio struktūra

IDEF0, kaip funkcijų modeliavimo kalbos, charakteristikos :

1. Išsamus ir išraiškingas, galintis reikiamu detalumo lygiu grafiškai pateikti plataus spektro įvairių biznio, gamybos ir kitų tipų organizacijų procesus.
2. Tai aiški ir paprasta kalba, suteikianti griežtas ir tikslias išraiškas, bei naudojimo neprieštaringumą ir interpretavimą.
3. Hierarchinis detalizavimo išdėstymas ir modeliavimo kalbos paprastumas pagerina bendravimą tarp sistemų analitikų, kūrėjų ir vartotojų.
4. Ji gerai ištestuota ir patvirtinta, daug metų naudojant oro pajėgų ir kituose vyriausybės kūrimo projektuose.
5. Ji gali būti generuojama įvairių grafinių įrankių, daug komercinių produktų palaiko IDEF0 diagramų ir modelių analizę ir kūrimą.

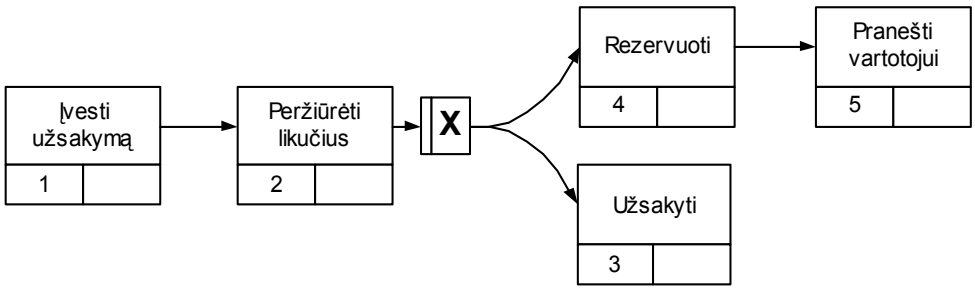
IDEF0 metodologija turi tiek privalumų, tiek trukumų [6]. Pagrindinis privalumas, ypač jeigu naudojama projektavimą palaikanti programa, yra tai, kad ji turi aiškias taisykles ir pateikia vienareikšmiškus procesų aprašus. Pagrindinis trukumas yra tai, kad taisyklės kartais per daug diktuoja sąlygas (apriboja), ir gali būti labai sunku atlikti pakeitimus modeliuose, esant dideliam skaičiui hierarchijos lygių. Taip pat IDEF0 modeliai turi tendenciją greitai sudėtingėti sudarydami daugybę ryšių tarpusavyje.

2.4.2 Vartotojo sąsajos modeliavimas, naudojant IDEF8 metodą

Trijų lygių specifikacijos yra naudojamas IDEF8 metode projektuojant vartotojo – sistemos sąveiką[19]. Pirmajame lygyje yra apibrėžiama bendra sistemos veikimo koncepcija. Sukuriama aibė modelių ir tekstinių aprašų apie sistemos procesus. Antrame projektavimo lygyje specifikuojami vartotojų scenarijai naudojant sistemą. Trečiasis lygis skirtas vartotojo – kompiuterio sąveikos projektavimo detalizavimui ir tobulinimui.

Pirmajame lygyje bendra sistemos veikimo koncepcija yra aprašoma naudojant abstrakčius IDEF3 procesų aprašus, IDEF0 funkcijų modelius ir struktūrizuotus tekstus.

Antrajame lygyje pirmiausiai yra specifikuojami sistemos aktoriai ir jų atliekamos rolės. Vėliau, kiekvienai rolei, naudojant IDEF3 metodą, aprašoma sąveikos su sistema scenarijai(6 pav.). Taip pat atliekama funkcijų ir užduočių analizė, bei pateikiami pradiniai meniu ekranai, meniu juosta ir t. t.



6 pav. IDEF3 diagrama

IDEF8 nėra grafinis vartotojo sąsajos kūrimo metodas. Jis gali būti naudojamas su GUI kūrimo sistemomis, bet nėra griežtai skirtas vartotojo sąsajos kūrimui. Šis metodas naudojamas aprašyti sąveikoms tarp vartotojo ir sistemos, bet ne apibrėžti konkrečią sąsają. IDEF8 naudojamas surinkti informacijai, kokius veiksmus ir užduotis vartotojas atlieka su sistema. Tačiau KDV struktūros sudarymas paliekamas sistemos projektuotojui.

2.5 Informacijos sistemai keliamų funkcinių reikalavimų specifikavimo metodas

Reikalavimų specifikavimas, naudojantis tokiomis priemonėmis kaip *Oracle CASE[13], Rational Unified Process* [8] informacijos sistemai keliamų reikalavimų specifikacijos sudarymo eiga yra nenatūrali. Pavyzdžiui, duomenų modelio elementai į specifikaciją įtraukiami anksčiau, nei išvedamos informacijos vieneto elementai, kurie motyvuoja duomenų modelio elementų atsiradimą specifikacijoje. Informacijos sistemų katedros pasiūlytas informacijos sistemai keliamų funkcinių reikalavimų specifikavimo metodas yra artimas natūraliai reikalavimų analizės eigai [1]. Taikant šį metodą, sistemos kūrimo procesas yra suskirstomas į 4 fazes (7 pav.):

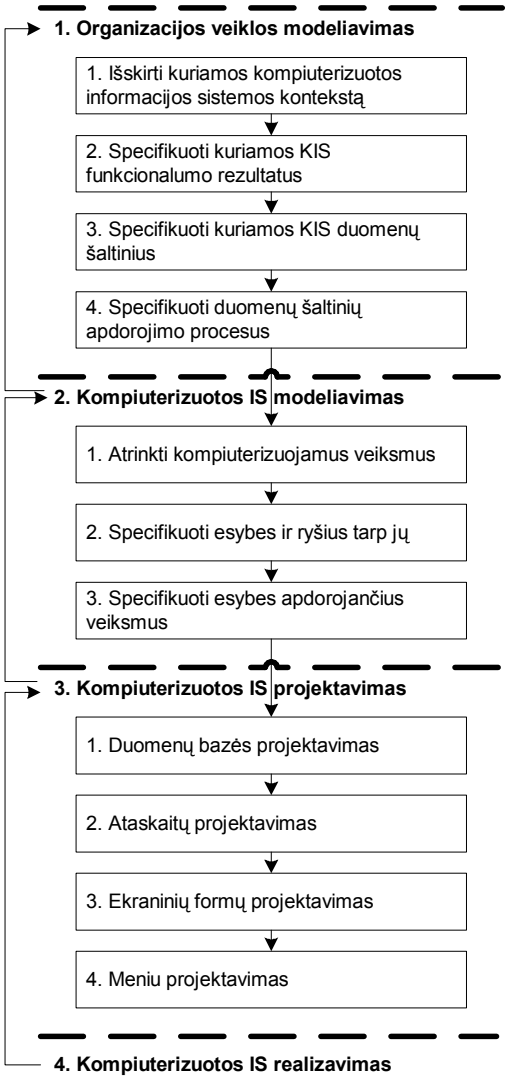
I. Organizacijos kompiuterizuojamos veiklos modeliavimas.
II. Kompiuterizuotos IS modeliavimas.
III. Kompiuterizuotos IS projektavimas.
IV. Realizavimas.

Šis specifikavimo metodas apima pirmąsias dvi sistemos kūrimo proceso dalis, tačiau numatoma, kad metodo taikymo rezultatai bus naudojami ir projektuojant bei realizuojant sistemą.

Modeliavimo procesas, taikant nagrinėjamą metodą yra iteracinis. Į kiekvieną vystymo etapą pereinama nuosekliai, tačiau iš bet kurio etapo galima grįžti į ankstesnį etapą.

Pagrindinė reikalavimų specifikavimo idėja yra tokia:

1. Specifikuojamos organizacijos veiklos funkcijos, apibrėžiančios kuriamos KIS kontekstą.
2. Specifikuojami KIS funkcionalumo rezultatai(FR), kurie susiejami su funkcijomis, t. y. specifikuojama, kokią informaciją turės išvesti KIS.
3. Kiekvienam KIS FR specifikuojami duomenų šaltiniai(DŠ), kurie naudojami FR formuoti, t. y. specifikuojama informacija, kurios reikia KIS išvedamai informacijai gauti.
4. Kiekvienam DŠ specifikuojami juos apdorojantys veiksmai.
5. Kiekvienam veiksmui specifikuojami jį atliekantys aktoriai bei aktoriai, perimantys veiksmo rezultatus.



7 pav. KIS kūrimo, taikant informacijos sistemai keliamų funkcinių reikalavimų specifikavimo metodą, procesas

Informacijos sistemai keliamų funkcinių reikalavimų specifikavimo metodą galima įvertinti kaip metodą, palengvinantį analitiko darbą tiek išgaunant reikalavimus iš vartotojo, tiek atliekant šių reikalavimų specifikavimą. Nuosekli ir natūralizuota metodo etapų atlikimo technologija padidina sudaromos specifikacijos kokybę, padeda geriau patikrinti, ar specifikacija pilna, nes modelis lengvai suprantamas vartotojui.

2.6 Metodų ir įrankių savybių įvertimas

Ankstesniuose skyriuose buvo apžvelgti šiuo metu naudojami metodai bei įrankiai funkcijų hierarchijos sudarymui, bei KDV projektavimui. 1 lentelėje pateiktas IDEF, Oracle CASE, UML palyginimas, pagal pasirinktus kriterijus.

1 Lentelė IDEF, Oracle CASE, UML palyginimas

Kriterijus	IDEF	Oracle CASE	UML
Programinė įranga	+	+	+
Grafinių modelių suprantamumas	+	+/-	+/-
Konteksto specifikavimas	+	+	+
Funkcijų hierarchinė struktūra	+	+	-
KDV sudarymas	-	+	+

Įvairūs programiniai paketai leidžia modeliuoti naudojantis IDEF metodu ir UML, tačiau Oracle CASE metodas yra naudojamas tik Oracle Designer įrankyje.

IDEF metode sudaromi modeliai yra paprasti ir lengvai suprantami, tačiau Oracle CASE ir UML kai kurie modeliai (pvz., duomenų srautų, esybių ryšių, klasių, sekų) yra sunkiai suprantami ir reikalauja ilgesnio apsimokymo laiko jų sudarymui ir interpretavimui.

Konteksto specifikavimą leidžia visi metodai (IDEF metodologijoje - IDEF metodas, Oracle CASE – funkcijų ir procesų diagramos, UML – panaudojimo atvejų diagramos). Tačiau UML nesuteikia priemonių funkcijų hierarchijos specifikavimui.

IDEF metodologija nors ir turi IDEF8 metodą vartotojo sąsajos specifikavimui, tačiau neteikia priemonių kompiuterizuotų darbo vietų sudarymui. Oracle CASE metodo pagrindu realizuotas CASE įrankis leidžia sudaryti tik vieno tipo KDV. Spalvotojo UML pagrindu pasiūlytas KDV sudarymas yra lankstus, tačiau gana sudėtingas dalykinės srities klasių modelio sudarymas.

2.7 Analizės išvados

1. Išanalizuoti kompiuterizuojamos sistemos konteksto specifikavimo metodai, aptarti jų privalumai ir trukumai, į kuriuos tikslinga atsižvelgti sudarant KIS konteksto specifikavimo modelį ir realizuojant CASE įrankį.
2. Aptarti kompiuterizuotos darbo vietos funkcionalumo specifikavimo ir struktūros sudarymo principai ir rekomendacijos, kurios bus įvertintos KDV struktūros projektavime.
3. Apžvelgtas vartotojo sąsajos projektavimas modelių pagrindu, jo panaudojimo problemos, kurių bus siekiama išvengti sudarant KDV funkcionalumo specifikavimo modelį.
4. Aprašytas informacijos sistemai keliamų funkcinių reikalavimų specifikavimo metodas, kurio kontekste bus pateiktas KDV funkcionalumo specifikavimo modelis. Pateikti jo pagrindiniai privalumai kitų funkcinių reikalavimų specifikavimo metodų atžvilgiu.

3 KDV funkcionalumo specifikavimo modelis

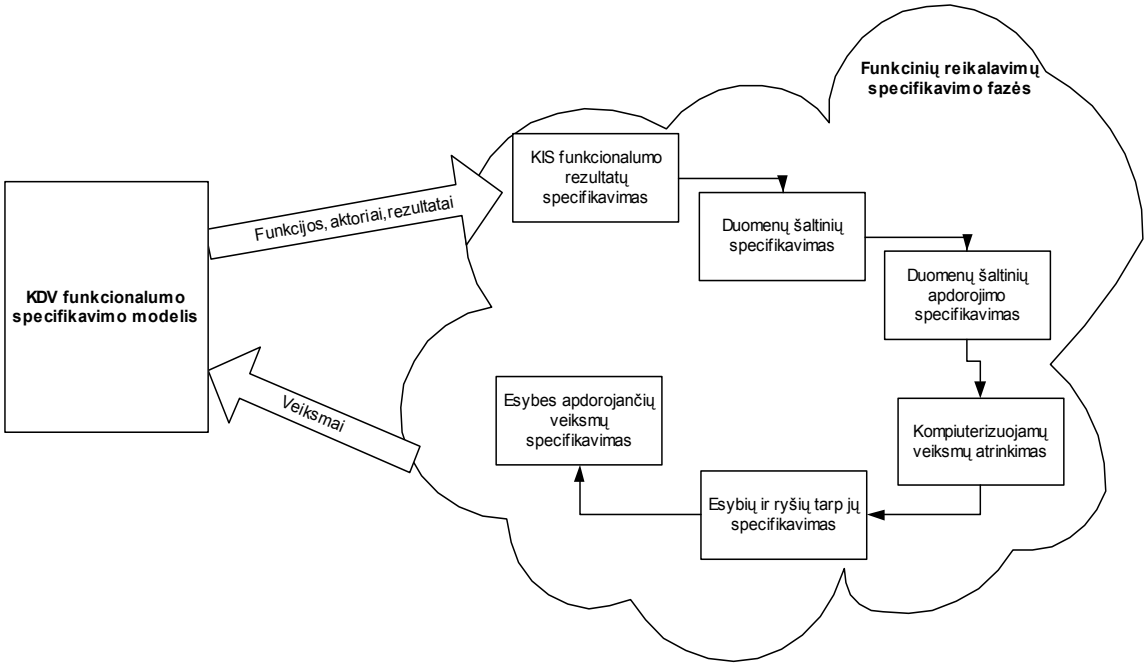
Išanalizavus KDV projektavimo metodus ir įrankius buvo pasiūlyta metodika, informacijos srautų specifikacijos pagrindu, KDV funkcionalumo specifikavimo modelio sudarymui [14]. Šis modelis yra funkcinių reikalavimų specifikavimo (FRS) metodo sudedamoji dalis. FRS metodo pagrindinė idėja: priklausomai nuo KIS aplinkos, pirmiausia reikia nustatyti, ką kuriamoji sistema turi pateikti kaip išeigą, o po to nustatyti, ką paduoti kaip įeigą ir kaip tą įeigą transformuoti į išeigą.Taikant šį metodą, sistemos kūrimo procesas yra suskirstomas į keturias fazes, kurias detalizuoja tam tikri etapai:

I. Organizacijos kompiuterizuojamos veiklos modeliavimas.
    1. Išskirti kuriamos KIS kontekstą;
    2. Specifikuoti kuriamos KIS funkcionalumo rezultatus;
    3. Specifikuoti kuriamos KIS duomenų šaltinius;
    4. Specifikuoti duomenų šaltinių apdorojimo procesus;
II. Kompiuterizuotos IS modeliavimas.
    1. Atrinkti kompiuterizuojamus veiksmus;
    2. Specifikuoti esybes ir ryšius tarp jų;
    3. Specifikuoti esybes apdorojančius veiksmus;
III. Kompiuterizuotos IS projektavimas.
    1. Duomenų bazės projektavimas;
    2. Ataskaitų projektavimas;
    3. Formų projektavimas;
    4. Meniu projektavimas
IV. Realizavimas.

KDV funkcionalumo specifikavimo modelis apima du šio metodo etapus: KIS konteksto specifikavimą ir meniu projektavimą. 8 paveikslėlyje pateikiama principinė nagrinėjamo modelio ir funkcinių reikalavimų specifikavimo metodo sąsajos schema.

Pirminis funkcinių reikalavimų specifikavimo etapas yra konteksto specifikavimas, kurio metu yra specifikuojamos veiklos funkcijos. Taip pat šiame etape sistemos analitikas gali specifikuoti nagrinėjamoje veikloje dalyvaujančius aktorius, jų grupes bei KIS funkcionalumo rezultatus. Atlikus pirminį funkcijų specifikavimo etapą, vykdomi kiti FRS metodo etapai, po kurių sugrįžtama į KDV funkcionalumo specifikavimo modelį. Tuomet modelis papildomas funkcionalumo rezultatus formuojančiais, ir duomenų šaltinius apdorojančiais veiksmais. Baigus formuoti KIS konteksto modelį, bei atlikus jo pilnumo ir

pertekliškumo patikrinimą, suformuojami kompiuterizuotų darbų vietų sudėties, pagal sudarytą specifikaciją, variantai.



8 pav. KDV specifikavimo modelis FRS metodo kontekste

3.1 KIS konteksto specifikavimas

Norint pradėti kurti ir projektuoti sistemą, pirmiausia turi būti specifikuotas kompiuterizuojamos sistemos kontekstas. KIS konteksto specifikavimui buvo pasirinkta funkcijų hierarchijos diagrama, kuri apibrėžia, kokios funkcijos turi būti atliktos, norint pasiekti modeliuojamos organizacijos keliamus tikslus. Šiame modelyje siūloma specifikuoti tik kompiuterizuojamos veiklos kontekstą.

Funkcijų hierarchija bus sudaroma remiantis R. Barker ir C. Longman pasiūlytu funkcijų hierarchijos modeliavimo būdu [13]. Hierarchija sudaroma jos viršūnėje apibrėžiant pagrindinę, visą veiklą apimančią, funkciją, ir toliau ją detalizuojant. Modeliuojant funkcijų hierarchiją kiekviename žemesniame lygyje turi būti specifikuotos visos „tėvinę“ funkciją realizuojančios funkcijos. Kitaip tariant, turi būti identifikuotos visos funkcijos, kurias įvykdžius „tėvinė“ funkcija bus taip pat įvykdyta. Vieno lygio funkcijos turi būti apytikriai vienodai svarbios įgyvendinant organizacijos tikslus. Funkcijų hierarchija siūloma detalizuoti iki vieno duomenų šaltinio arba rezultato apdorojimo transakcijos.

Ši funkcijų hierarchija (FH) yra praplėsta kitais FRS metode naudojamais komponentais: veiksmais, rezultatais, aktoriais ir todėl bus vadinama modifikuota FH

diagrama. Rezultatų ir aktorių specifikavimas įvestas siekiant suteikti projektuotojui galimybę jau veiklos konteksto specifikavimo etape specifikuoti funkcionalumo rezultatus, juos formuojančius veiksmus ir rezultatais besinaudojančius aktorius, taip pat identifikuotus, veikloje dalyvaujančius, aktorius/aktorių grupes. Tačiau platesnis šių elementų specifikavimo aprašymas nebus pateiktas, nes jis atliekamas ir kituose etapuose, bei nėra svarbus KDV sudėties projektavimui. Modifikuotame funkcijų hierarchijos modelyje naudojamų komponentų notacija yra pateikiama 9 paveiksle. Pagrindinių elementų sąrašą sudaro tokie komponentai:

Funkcija – tai veikla, kurią turi įvykdyti organizacinė sistema, kad pasiektų tam tikrą užsibrėžtą tikslą.

Top funkcija – tai funkcija, esanti funkcijų hierarchijos viršūnėje. Ji apibrėžia visą kompiuterizuojamos veiklos kontekstą. Hierarchijoje gali būti tik viena Top funkcija.

Detalizuota funkcija – tai funkcija, kuri susideda iš kelių detalizuotų arba nedetalizuotų funkcijų.

Nedetalizuota funkcija – tai žemiausio lygio hierarchijos funkcijos, kurios realizuoja vieno duomenų šaltinio arba rezultato apdorojimo transakciją.

Funkcijų komponavimo sąryšis - tai ryšys, kuris susieja dvi funkcijas sudarydamas hierarchinę struktūrą. Ryšio rodyklė nurodo „tėvinę“ funkciją.

Funkcijų realizavimo ryšys – tai ryšys, naudojamas susieti funkciją, su ją realizuojančiais veiksmais.

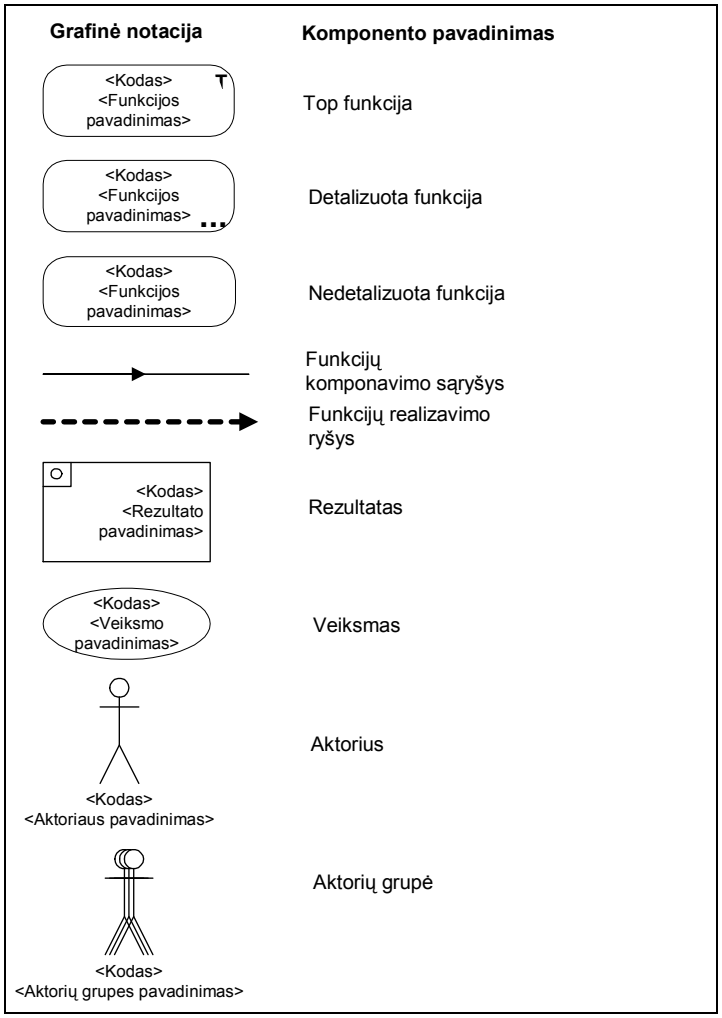
Rezultatas – KIS funkcionalumo metu formuojamas rezultatas (ekraninė forma, ataskaita, duomenų srautas), kuris iš dalies pakeičia iki KIS įdiegimo cirkuliavusius popierinius dokumentus ar žodžiu perduodamus informacijos srautus.

Veiksmas – tai organizacijos veiklą sudarantis veiksmas, kuris sukuria, pašalina, pakeičia tam tikrus organizacijoje cirkuliuojančius informacijos srautus.

Aktorius – tai asmuo, rolė, organizacijos padalinys turintis įgaliojimus atlikti tam tikrus veiksmus.

Aktorių grupė – tai grupė, sudaryta iš aktorių, turinčių įgaliojimus atlikti tam tikrą veiksmą.

Visi naudojami notacijos komponentai, išskyrus ryšius, turi pavadinimą ir santrumpą. Santrumpa naudojama supaprastinti manipuliavimą komponentais. Funkcijų ir veiksmų pavadinimams siūloma naudoti veiksmažodžius arba kitas trumpas, bet tikslias frazes, kuo tiksliau atspindinčias komponento prasmę.



9 pav. Modifikuotos funkcijų hierarchijos notacija

Funkcijų hierarchijos matematinis modelis gali būti apibrėžtas ketvertu

$$Fh=<F, V, R_F, R_V>,$$

čia F – funkcijų aibė, kurią sudaro trys nepersikertantys poaibiai:

$$F = F_N \cup F_D \cup F_T$$

$$F_N \cap F_D \cap F_T = \varnothing$$

F_N - nedetalizuotų funkcijų aibė;

F_D – detalizuotų funkcijų aibė;

F_T – top funkcijų aibė;

V – kompiuterizuojamų veiksmų aibė, ji susideda iš dviejų nepersikertančių aibių:

$$V = V_R \cup V_D$$

$$V_R \cap V_D = \varnothing$$

V_R – rezultatus formuojančių kompiuterizuojamų veiksmų aibė;

V_D - duomenų šaltinius apdorojančių kompiuterizuojamų veiksmų aibė;

$R_V \subseteq F \times V$ - sąryšis, nusakantis, kokios funkcijos yra realizuojamos tam tikrų veiksmų.

R_F – sąryšis, nusakantis, kokios funkcijos detalizuoja aukštesnio lygio funkcijas.

Pagal sąryšį R_v funkcijų aibę galima suskirstyti į du nepersikertančius poaibius:

$$F = F_R \cup F_{NR}$$

$$F_R \cap F_{NR} = \varnothing$$, čia

F_R - realizuojamų funkcijų aibė;

F_{NR} – nerealizuojamų funkcijų aibė.

Toliau yra pateikiami funkcijų hierarchijos sudarymo apribojimai:

1. Funkcijų hierarchijos modelis x turi turėti tik vieną top funkciją y, bent dvi nedetalizuotas funkcijas y' ir y'', nors vieną rezultatą formuojantį veiksmą z ir duomenų šaltinį apdorojantį veiksmą q.

$$\forall x\,[Fh(x) \Rightarrow \exists! y \exists y' \exists y'' \exists z \exists q\,[F(y) \wedge F(y') \wedge F(y'') \wedge V_R(z) \wedge V_D(q) \wedge yra\_top\_funkc(y) \wedge yra\_nedet\_funkc(y') \wedge yra\_nedet\_funkc(y'') \wedge priv\_turėti(x,y) \wedge priv\_tureti(x,y') \wedge priv\_tureti(x,y'') \wedge priv\_tureti(x,z) \wedge priv\_tureti(x,q)]] \quad (1)$$

2. Kiekvienai nedetalizuotai funkcijai x vienas ir tik vienas rezultatą formuojantis veiksmas y ar bent vienas duomenų šaltinį apdorojantis veiksmas z turi būti specifikuotas.

$$\forall x\,[F(x)) \wedge yra\_nedet\_funkc(x) \Rightarrow \exists! y \exists z\,[V_R(y) \wedge realizuoja(x,y) \vee V_D(z) \wedge realizuoja(x,z)]] \quad (2)$$

3. Kiekvienas rezultatą formuojantis veiksmas x turi realizuoti vieną ir tik vieną funkciją y.

$$\forall x\,[V_R(x) \Rightarrow \exists! y\,[F(y) \wedge yra\_nedet\_funkc(y) \wedge realizuoja(y,x)]] \quad (3)$$

4. Kiekvienas duomenų šaltinį apdorojantis veiksmas x turi realizuoti vieną ir tik vieną funkciją.

$$\forall x\,[V_D(x) \Rightarrow \exists! y\,[F(y) \wedge yra\_nedet\_funkc(y) \wedge realizuoja(y,x)]] \quad (4)$$

5. Kiekvienas ryšys x, jungiantis rezultatą formuojantį veiksmą/DŠ apdorojantį veiksmą su funkcija, gali sieti tik vieną rezultatą formuojantį veiksmą y su tik viena nedetalizuota funkcija q arba tik vieną duomenų šaltinį apdorojantį veiksmą z su tik viena nedetalizuota funkcija q.

$$\forall x\,[R_V(x) \Rightarrow \exists! y \exists! z \exists! q\,[F(q) \wedge yra\_nedet\_funkc(q) \wedge (V_R(y) \wedge jungia(x,y,q)) \vee V_D(z) \wedge jungia(x,z,q))]] \quad (5)$$

6. Kiekvienas funkcijų sąryšis x sieja tik dvi skirtingas funkcijas y' ir y'', kur y' yra detalizuota arba nedetalizuota "vaiko" tipo funkcija, o y'' yra top arba detalizuota "tevine" funkcija.

$$\forall x\,[R_F(x) \Rightarrow \exists! y' \exists! y''\,[F(y') \wedge yra\_vaiko\_funkc(y') \wedge (yra\_nedet\_funkc(y) \vee yra\_detaliz\_funkc(y'')) \wedge F(y'') \wedge yra\_tevine\_funkc(y') \wedge (yra\_top\_funkc(y) \vee is\_detaliz\_funkc(y'')) \wedge jungia(x,y',y'')]] \quad (6)$$

Funkcijų hierarchijos modeliavimas, kaip ir funkcinių reikalavimų specifikavimo metodas yra iteracinis procesas, todėl iš kiekvienos analizės ar projektavimo fazės galima sugrįžti koreguoti hierarchiją. Modifikuotos funkcijų hierarchijos sudarymas turi tris etapus:

1. Veiklos konteksto specifikavimas. Šio etapo metu specifikuojamos veiklos funkcijos apbrėžiant kompiuterizuojamos sistemos kontekstą. Taip pat gali būti specifikuojami ir susiejami su funkcijomis rezultatus formuojantys veiksmai, bei patys rezultatai.
2. Funkcijas realizuojančių veiksmų atrinkimas. Šio etapo metu funkcijoms priskiriami kituose FRS metodo fazėse specifikuoti duomenų šaltinius apdorojantys ir rezultatus formuojantys veiksmai. Taip pat gali būti atliekamas tolesnis hierarchijos detalizavimas, kuris reikalingas specifikuotų duomenų šaltinių ar rezultatų formavimui.
3. Verifikavimas. Atliekamas sudaryto modelio neperteklišKumo ir pilnumo tikrinimas: ar visi specifikuoti veiksmai priskirti funkcijoms, ar visos funkcijos turi jas realizuojančius veiksmus.

Modeliuojant organizacijos veiklą funkcijų hierarchija gana sparčiai didėja, sudarydama daug elementų apimančią hierarchiją, kurią gali būti nepatogu peržiūrėti, todėl siūloma FH skaidyti į subdiagramas. Šis skaldymas gali būti atliekamas pagal skirtingas veiklos sritis, padalinius ir t. t.

3.2 KDV sudėties formavimas

KDV sudėties formavimas yra paskutinis funkcinių reikalavimų specifikavimo metodo etapas. Pagal suformuotą ir verifikuotą funkcijų hierarchiją sistemos projektuotojui yra siūlomos dvi KDV sudarymo strategijos:

- Pagal funkcijų hierarchijos šakas;
- Pagal aktorių ir jų grupių atliekamus veiksmus.

Antrojo tipo KDV formuojama aktoriams, kurie buvo specifikuoti duomenų šaltinių apdorojimo arba rezultatų struktūros specifikavimo etapuose. Abiejose strategijose KDV sudarymui yra naudojami veiksmai, kurie yra:

- duomenų šaltinius (DŠ-dokumentus, žodinius pranešimus, kompiuterizuotų sistemų ekranines formas, elektroniniu būdu perduodamus duomenų paketus) apdorojantys veiksmai;
- IS funkcionalumo rezultatus (dokumentus, rodiklius, kitą išvedamą vartotojui informaciją) formuojantys veiksmai.

Sukurtas modelis remiasi 4 spalvų modelio[16] rekomendacijomis, kad meniu neturėtų daugiau nei 10 meniu punktų. Teigiama, kad geras meniu vartotojui suteikia pasirinkimą iš 7+/-2 elementų. Esant didesniam elementų skaičiui meniu traktuojamas kaip perkrautas ir apsunkinantis vartotojo darbą.

Žemiau pateikiami kompiuterizuotos darbo vietos sudėties formavimo principai pagal abi strategijas.

KDV formavimas pagal funkcijų hierarchijos šakas

10 paveikslėlyje pateikta meniu formavimo pagal pasirinktas funkcijų hierarchijos šakas veiksmų seka.

KDV sudarymo pagal hierarchijos šakas modelis yra dvejetas

$$K_F = <Fh, M>$$

Fh – funkcijų hierarchijos modelis, aprašytas 3.1 skyriuje.

M – meniu elementų aibė, kuri susideda iš trijų nepersikertančių poaibių:

$$M = M_{PG} \cup M_{PK} \cup M_{TR}$$

$$M_{PG} \cap M_{PK} \cap M_{TR} = \varnothing$$

M_P – pagrindinio meniu elementų aibė,

M_{PK} – paskutinio submeniu lygio elementų aibė;

M_{TR} – tarpinio submeniu lygio elementų aibė.

Kompiuterizuotos darbo vietos funkcionalumo atrinkimui ir suformavimui taikomi pagrindiniai apribojimai ir sudarymo eiga:

1. Pasirenkama funkcija, kuriai suformuojama autonominė kompiuterizuota darbo vieta. Kiekviena kompiuterizuota darbo vieta x yra sudaroma pasirinktai funkcijų hierarchijos z funkcijai y, kurios tipas gali būti tik detalizuojama arba top.

$$\forall x[K_F(x) \Rightarrow \exists! y[Fh(z) \wedge (F_D(y) \vee F_T(y)) \wedge yra\_sudaroma(x,y) \wedge priklauso(y,z)]]$$

(1)

2. Pasirenkamas pasirinktos funkcijos šakos hierarchijos lygis iš kurio bus formuojamas KDV pagrindinis meniu.
3. Sudaromas pagrindinis meniu x iš funkcijų hierarchijos diagramos atrenkant nurodyto lygio funkcijas y, kurios detalizuoja 1 punkte pasirinktą funkciją z. Jeigu nors viena šaka yra trumpesnė nei nurodytas hierarchijos lygis, pasirenkama žemiausio lygio nedetalizuota funkcija.

$$\forall x[M_{PG}(x) \Rightarrow \exists y[F(y) \wedge F(z) \wedge \det alizuoja(y,z) \wedge (yra\_nurodyto\_lygio(y) \vee žemiausio\_lygio(y)) \wedge sudaro(x,y)]]$$

(3)

4. Atrenkami funkciją x, kuri detalizuoja 1 punkte pasirinktą funkciją z, realizuojantys veiksmai y.

$$\forall x[M_{PG}(x) \Rightarrow \exists y[V(y) \wedge F(x) \wedge F(z) \wedge \det alizuoja(y,z) \wedge (real\_funkc(x,y)]]$$

(4)

5. Jeigu atrinktų veiksmų skaičius vienai 3 punkte nurodytai funkcijai yra didesnis nei dešimt, tai vykdomas 6 punktas, priešingu atveju – 7 punktas.
6. KDV meniu x sudaro pagrindinis meniu y, I lygio submeniu z, bei II lygio submeniu q.

$$\forall x[K_F(x) \Rightarrow \exists y \exists z \exists q[M_{PG}(y) \wedge M_{PS}(z) \wedge M_{TR}(q) \wedge sudaro(x,y,z,q)]]$$

(6)

Čia:

a) I lygio submeniu x sudaro 3 punkte atrinktos funkcijos y žemiausio lygio (nedetalizuotos) funkcijos q.

$$\forall x[M_{TR}(x) \Rightarrow \exists q[F(q) \wedge F(y) \wedge yra\_ne\det\_funkc(q) \wedge \det alizuoja(y,q) \wedge sudaro(x,q)]]$$

(6 a.)

b) II lygio submeniu x sudaro I lygio submeniu y funkcijas z realizuojantys veiksmai q.

$$\forall x[M_{PS}(x) \Rightarrow \exists q[V(q) \wedge F(z) \wedge M_{TR}(y) \wedge realiz\_funkc(y,q) \wedge sudaro(x,q)]]$$

(6 b.)

7. Kompiuterizuotos darbo vietos meniu x sudaro pagrindinis meniu y ir I lygio submeniu z.

$$\forall x[K_F(x) \Rightarrow \exists y \exists z[M_{PG}(y) \wedge M_{PS}(z) \wedge sudaro(x,y,z)]]$$

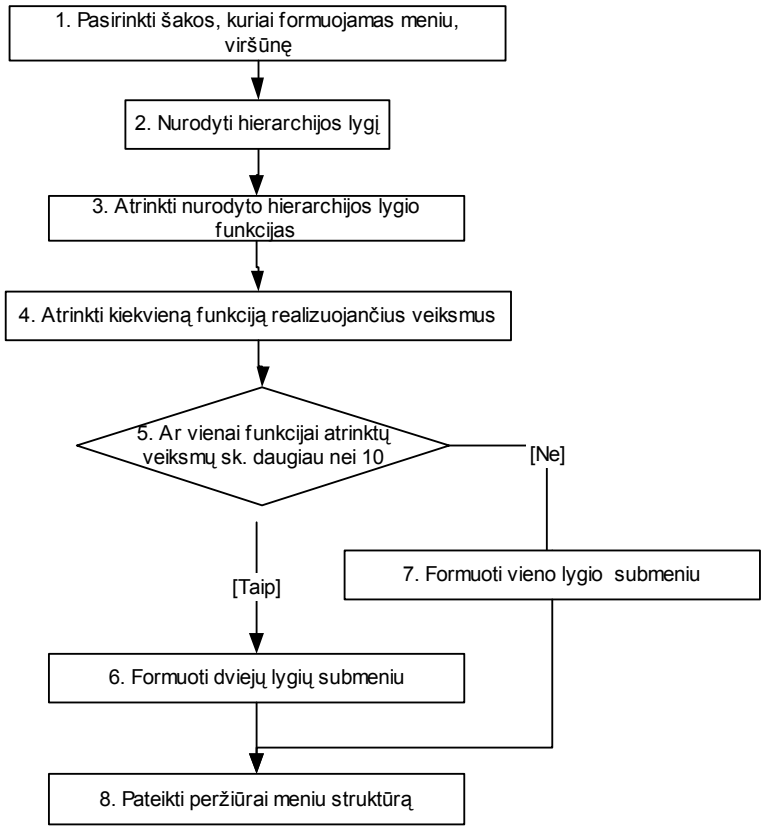
(7)

a) I lygio submeniu sudaro 3 punkte atrinktą funkciją y realizuojantys veiksmai q.

$$\forall x[M_{PS}(x) \Rightarrow \exists q[V(q) \wedge F(y) \wedge realiz\_funkc(y,q) \wedge sudaro(x,q)]]$$

(7 a.)

8. Sistemos projektuotojo peržiūrai pateikiamas suformuotas meniu eskizas.

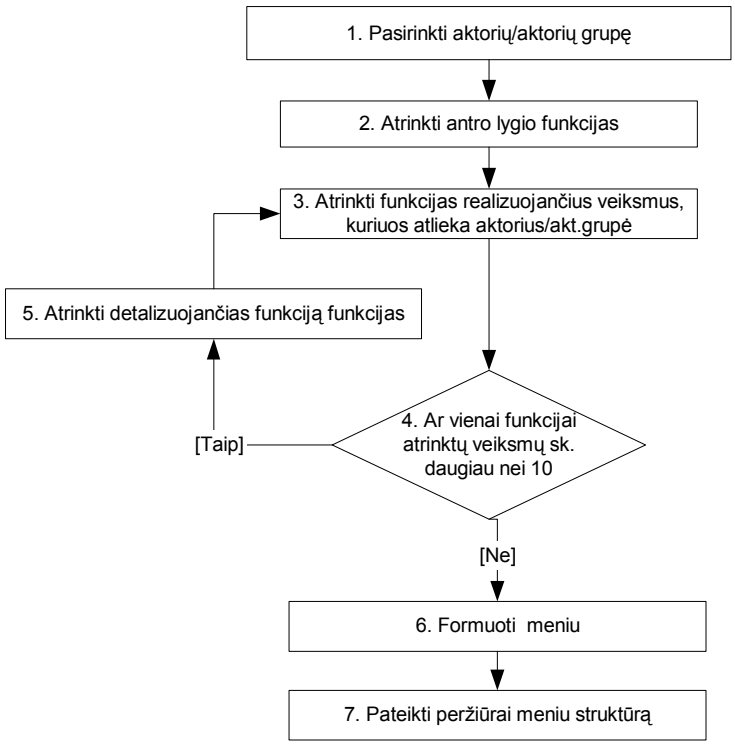


10 pav. Meniu formavimo pagal hierarchijos šakas veiksmų seka

Jeigu funkcijų hierarchija buvo suformuota skirtingose šakose modeliuojant atskirų padalinių veiklą, tai ši KDV sudėties formavimo strategija gali pasiūlyti pilnai funkcionalumą padengiančią KDV sudėtį pageidaujamam padaliniui.

KDV formavimas pagal aktorių ir jų grupių atliekamus veiksmus

11 paveikslėlyje pateikta meniu formavimo pagal aktorių arba jų grupių atliekamus veiksmus vykdymo eiga.



11 pav. Meniu formavimo pagal aktorius ir jų grupes veiksmų seka

KDV sudarymo pagal hierarchijos šakas modelis yra trejetas

K_A = <Fh, M, A, AG>

Fh – funkcijų hierarchijos modelis, aprašytas 3.1 skyriuje.

M – meniu elementų aibė, kuri susideda iš dvijų nepersikertančių poaibių:

$M = M_{PG} \cup M_{PK} \cup M_{TR}$

$M_{PG} \cap M_{PK} \cap M_{TR} = \varnothing$

M_P – pagrindinio meniu elementų aibė,

M_{PK} – paskutinio submeniu lygio elementų aibė.

A – aktorių, atliekančių veiksmus, aibė;

AG –aktorių grupių aibė.

Kompiuterizuotos darbo vietos funkcionalumo atrinkimui ir suformavimui taikomi pagrindiniai apribojimai ir sudarymo eiga:

1. Kompiuterizuota darbo vieta x sudaroma pasirinktam aktoriui y arba aktorių grupei z.

$$\forall x[K_A(x) \Rightarrow \exists y \exists z[A(y) \wedge AG(z) \wedge (sudaro(x,y) \vee sudaro(x,z))]] \quad (1)$$

2. Sudaromas pagrindinis meniu x iš funkcijų hierarchijos diagramos y atrenkant antro lygio funkcijas z.

$$\forall (x)[M_P(x) \Rightarrow Fh(z) \wedge F(y) \wedge priklauso(y,z) \wedge antro\_lygio(y,z) \wedge sudaro(x,y)] \quad (2)$$

3. I lygio submeniu sudaromas kiekvienai 2 punkte atrinktai funkcijai x surandant ją realizuojančius veiksmus y, kuriuos atlieka 1 punkte pasirinktas aktorius/aktorių grupė z.

$$\forall x[M_{PS}(x) \Rightarrow F(v) \wedge V(y) \wedge A(z) \vee AG(z) \wedge realizuoja(x,y) \wedge atlieka(y,z) \wedge sudaro(x,y)] \quad (3)$$

4. Jeigu atrinktų veiksmų skaičius yra didesnis nei 15, vykdyti 5 punktą, priešingu atveju 6 punktą.
5. Atrinkti 3 punkte naudojamai funkcijai ją detalizuojančias funkcijas ir vėl vykdyti 3 punktą.
6. KDV meniu x sudaro pagrindinis meniu y, sudarytas 2 punkte, I lygio submeniu z, sudarytas 3 punkte.

$$\forall x[K_A(x) \Rightarrow M_{PG}(y) \wedge M_{PS}(z) \wedge sudaro(x,y,z)] \quad (6)$$

Jeigu nėra funkciją realizuojančių veiksmų, kuriuos atlieka 1 punkte pasirinktas

7. Sistemos projektuotojui peržiūrai pateikiamas sudarytas meniu eskizas.

Ši strategija yra patogi naudoti norint suformuoti kompiuterizuotą darbo vietą pasirinktam organizacijos veikloje dalyvaujančiam aktoriui ar jų grupei.

4 KDV funkcionalumo specifikavimo modelio realizacija

4.1 Reikalavimų specifikacija

4.1.1 Projekto kūrimo pagrindimas

Nesuprasti vartotojų reikalavimai yra viena iš pagrindinių programinės įrangos (PĮ) projektų žlugimo priežasčių. Vartotojo reikalavimų supratimas įgyjamas reikalavimų inžinerijos procese. Šiuo metu egzistuoja keletas metodų reikalavimams specifikuoti bei perkelti į projekto specifikaciją. Tai *Oracle CASE* metodas [13], *Rational Unified* procesas [10] ir kiti. Tačiau CASE (*Computer Aided System Engineering*) priemonėse siūloma specifikavimo technologija neatitinka natūraliu būdu naudojamos vartotojo reikalavimų bei projektinių sprendimų registravimo technologijos. Todėl buvo pasiūlyta realizuoti CASE įrankį, naudojantis sukurtu organizacijos veiklos modeliavimu paremtu metodu, kuris leidžia specifikuoti kompiuterizuotai IS keliamus funkcinius reikalavimus natūraliai ir sklandžiai[1].

Šio darbo praktinės dalies tikslas – suprojektuoti ir realizuoti CASE įrankio modulį, pasiūlyto KDV funkcionalumo specifikavimo modelio pagrindu. Sukurtas įrankis turi leisti specifikuoti elementus sudarant grafinį modelį, o taip pat išsaugoti specifikavimo rezultatus duomenų metabazėje.

Sukurtas CASE įrankis ir jo moduliai galės būti panaudoti kaip mokymo priemonė informacijos sistemų katedroje (ISK), gilinant žinias IS keliamų funkcinių reikalavimų inžinerijoje. Išbaigtą CASE priemonę galės naudoti informacijos sistemų projektuotojai.

4.1.2 Apribojimai reikalavimams

4.1.2.1 Apribojimai sprendimui

Sistemos kūrimo eigai ir charakteristikoms keliami tokie apribojimai:

- sistema turi funkcionuoti Windows 2000/XP operacinės sistemos pagrindu;
- sistema turi veikti nešiojamame kompiuteryje;
- sistema naudojasi MS Visio 2000/2002 programiniu paketu;
- sistema naudojasi MS Access duomenų bazių valdymo sistema.

4.1.2.2 Diegimo aplinka

Diegimo aplinkos charakteristikos :

- Windows 2000/XP operacinės sistemos platforma;
- MS Visio 2000/2002 tekstinis redaktorius;
- MS Access duomenų bazių valdymo sistema.

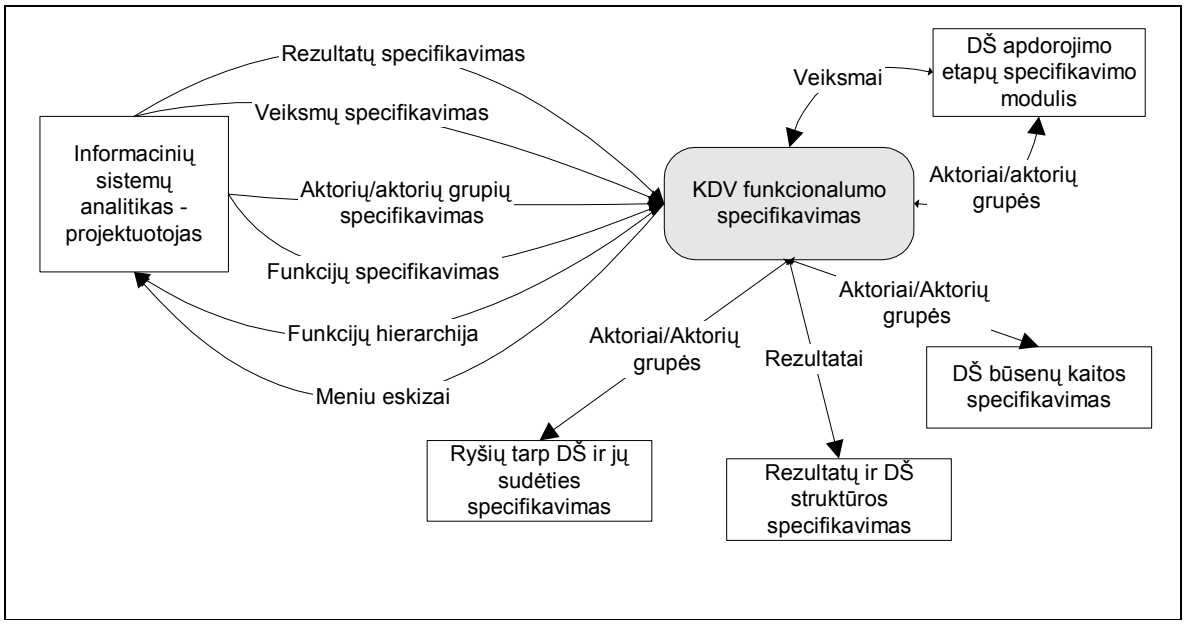
4.1.2.3 Bendradarbiaujančios sistemos

Kuriamas programinis modulis yra dalis CASE įrankio, realizuojančio IS kūrimą funkcinių reikalavimų specifikacijos pagrindu. Šio modulio rezultatais naudosis :

- Rezultatų ir DŠ(duomenų šaltinių) struktūros specifikavimo modulis;
- Ryšių tarp DŠ ir jų sudėties specifikavimo modulis;
- DŠ apdorojimo etapų specifikavimo modulis;
- Ryšių tarp duomenų šaltinių ir DŠ būsenų specifikavimo modulis;
- DŠ būsenų kaitos specifikavimo modulis.

4.1.3 Veiklos kontekstas

Pateikiama kuriamo CASE įrankio modulio veiklos konteksto diagrama (12 pav.), kuri apibrėžia šios sistemos ribas, išorines esybes, kurios bendrauja su sistema, bei pagrindinius informacijos srautus tarp sistemos ir išorinių esybių.



12 pav. Konteksto diagrama

Pateiktas veiklos įvykių sąrašas, kuris apima visus veiklos įvykius, už kuriuos atsakinga nagrinėjama veikla.

2 Lentelė Veiklos įvykių sąrašas

Eil. Nr.	***Įvykio pavadinimas***	***Įeinantys/Išeinantys informacijos srautai***
1	Informacinių sistemų analitikas identifikuoja kompiuterizuojamas funkcijas.	Funkcijų specifikavimas(in)
2	Sistemų analitikas identifikuoja rezultatus.	Rezultatų specifikavimas (in)
3	Informacinių sistemų analitikas specifikuoja aktorius/aktorių grupes	Aktorių/aktorių grupės specifikavimas (in)
4	Sistemų analitikas specifikuoja rezultatus formuojančius veiksmus.	Veiksmų specifikavimas (in)
5	Informacijos sistemų analitikas peržiūri funkcijų hierarchijos diagramą	Funkcijų hierarchija (out)
6	Informacijos sistemų analitikas peržiūri siūlomus meniu sudarymo eskizus	Meniu eskizai (out)
7	Specifikuotų aktorių/aktorių grupių panaudojimas sudarant ryšių tarp DŠ ir jų sudėties specifikavimo, DŠ apdorojimo etapų specifikavimo, DŠ būsenos kaitos modulius.	Aktoriai/aktorių grupė (out)
8	DŠ apdorojimo etapų specifikavimo modulyje specifikuotų veiksmų panaudojimas	Veiksmai (in)
9	Specifikuotų rezultatų panaudojimas rezultatų ir DŠ struktūros specifikavimo modulyje.	Rezultatai (out)

4.1.4 Panaudojimo atvejų sąrašas

Panaudojimo atvejų diagrama (13 pav.) nusako ribas tarp sistemos ir vartotojo. Toliau yra pateikiamas reikalavimų metu suformuotų panaudojimų atvejų aprašas.

1. Panaudojimo atvejis: Veiklos funkcijų hierarchijos formavimas

Aktorius: Sistemų analitikas

Aprašymas: Įvairių funkcinių reikalavimų metu identifikuotų elementų specifikavimas, hierarchinės struktūros diagramos sudarymas ir jos verifikavimas.

2. Panaudojimo atvejis: Specifikuoti funkciją

Aktorius: Sistemų analitikas

Aprašymas: KIS konteksto funkcijų specifikavimas

3. Panaudojimo atvejis: Specifikuoti sąryšį tarp funkcijų

Aktorius: Sistemų analitikas

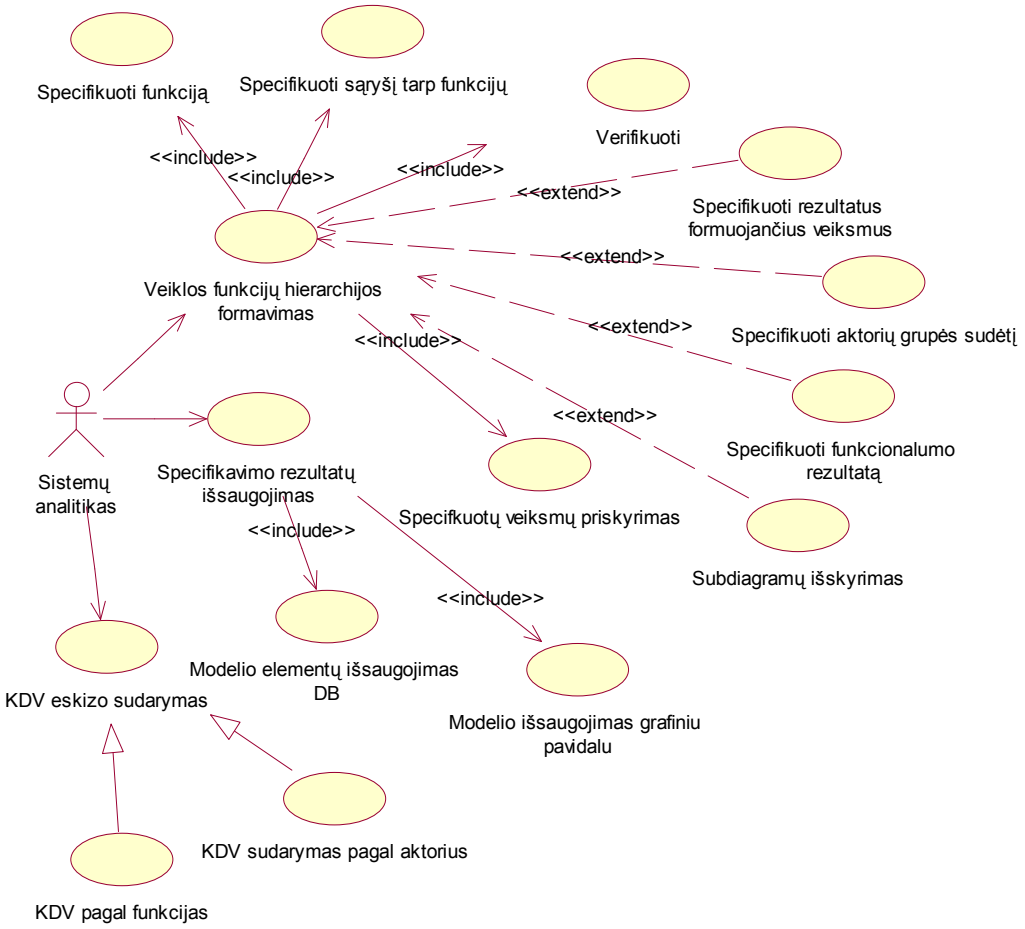
Aprašymas: Ryšio tarp dviejų funkcijų sudarymas, sudarant hierarchinę struktūrą ir

priklausomai nuo funkcijos padėties, jos vaizdavimo keitimas.

4. Panaudojimo atvejis: Verifikuoti

Aktorius: Sistemų analitikas

Aprašymas: Sudarytos diagramos neprieštaringumo, pilnumo ir pertekliškumo tikrinimas.



13 pav. Panaudojimo atvejų diagrama

5. Panaudojimo atvejis: Specifikuoti rezultatus formuojančius veiksmus

Aktorius: Sistemų analitikas

Aprašymas: Veiksmų, formuojančių rezultatus, specifikavimas, bei susiejimas su nedetalizuotomis funkcijomis.

6. Panaudojimo atvejis: Specifikuoti aktorių grupės sudėtį

Aktorius: Sistemų analitikas

Aprašymas: Aktorių grupių sudarymas iš jau specifikuotų sistemoje aktorių.

7. Panaudojimo atvejis: Specifikuoti funkcionalumo rezultatą

Aktorius: Sistemų analitikas

Aprašymas: Identifikuotų funkcionalumo rezultatų specifikavimas, jų susiejimas su veiksmais bei aktoriais.

8. Panaudojimo atvejis: Subdiagramų išskyrimas

Aktorius: Sistemų analitikas

Aprašymas: Funkcijų hierarchijos modeliavimo pratęsimas naujoje diagramoje.

9. Panaudojimo atvejis: Specifikuotų veiksmų priskyrimas

Aktorius: Sistemų analitikas

Aprašymas: Duomenų šaltinius apdorojančių veiksmų atrinkimas iš DB ir priskyrimas funkcijai, kurią jie realizuoja.

10. Panaudojimo atvejis: Specifikavimo rezultatų išsaugojimas

Aktorius: Sistemų analitikas

Aprašymas: Specifikuotų rezultatų išsaugojimas (grafinio modelio išsaugojimas arba specifikuotų rezultatų saugojimas IS saugykloje.)

11. Panaudojimo atvejis: Modelio elementų išsaugojimas DB

Aktorius: Sistemų analitikas

Aprašymas: Modeliavimo metu specifikuotų elementų išsaugojimas IS saugykloje.

12. Panaudojimo atvejis: Modelio išsaugojimas grafiniu pavidalu

Aktorius: Sistemų analitikas

Aprašymas: Sudaryto funkcijų hierarchijos modelio išsaugojimas.

12. Panaudojimo atvejis: KDV eskizo sudarymas

Aktorius: Sistemų analitikas

Aprašymas: Kompiuterizuotos darbo vietos sudarymo eskizų pateikimas, naudojant

funkcijų hierarchijos modelį (pagal aktorius, pagal funkcijas).

12. Panaudojimo atvejis: KDV eskizo sudarymas pagal funkcijas

Aktorius: Sistemų analitikas

Aprašymas: KDV eskizo pateikimas pagal nurodytą funkcijų hierarchijos šaką.

12. Panaudojimo atvejis: KDV eskizo sudarymas pagal aktorius

Aktorius: Sistemų analitikas

Aprašymas: KDV sudarymas nurodytam aktoriui.

4.1.5 Funkciniai reikalavimai

Reikalavimai grafinės diagramos sudarymui:

Reikalavimas 1: Sistema turi keisti funkcijos grafinį žymėjimą, priklausomai nuo jos padėties hierarchijoje (top, detalizuotos, nedetalizuotos funkcijos grafiniai žymėjimai).

Reikalavimas 2: Sistema turi leisti naudoti tik hierarchijos modeliavimui skirtus grafinius žymėjimus.

Reikalavimas 3: Sistema turi neleisti keisti grafinės funkcijų hierarchijos komponentų notacijos.

Reikalavimas 4: Sistema turi pakeisti funkcijos, kurios modeliavimas tęsiamas kitoje diagramoje, spalvą.

Reikalavimas 5: Sistema turi leisti sujungti tik tuos grafinius elementus, kurie gali būti susiejami pagal modelio sudarymo taisykles.

Reikalavimas 6: Sistema turi leisti išsaugoti grafinį sudaryto modelio vaizdą.

Reikalavimai duomenų mainams su IS saugykla:

Reikalavimas 7: Sistema turi sudaryti galimybę naudotis jau specifikuotais elementais, kurie išsaugoti IS saugykloje.

Reikalavimas 8: Sistema turi leisti išsaugoti sudaryto specifikacijos modelio duomenis IS saugykloje.

4.1.6 Nefunkciniai reikalavimai

Reikalavimai sistemos išvaizdai

- Turi būti naudojama grafinė vartotojo sąsaja.
- Sąsaja turi būti lengvai skaitoma, suprantama.

- Sąsaja turi būti sąveikaujanti.
- Turi būti išlaikomi realizuojamo CASE įrankio grafinės vartotojo sąsajos standartai.

Reikalavimai panaudojamumui

- Paprasta naudotis IS projektuotojams- analitikams.
- Turi būti lengvai išmokstama dirbti su sistema.

Reikalavimai vykdymo charakteristikoms

- Interaktyvaus bendravimo su vartotoju metu programa turi veikti pakankamai greitai, užtikrinti efektyvų darbą.
- Sistema neturi reikalauti daugiau resursu nei to maksimaliai gali prireikti Microsoft aplinkai.

4.2 Architektūros specifikacija

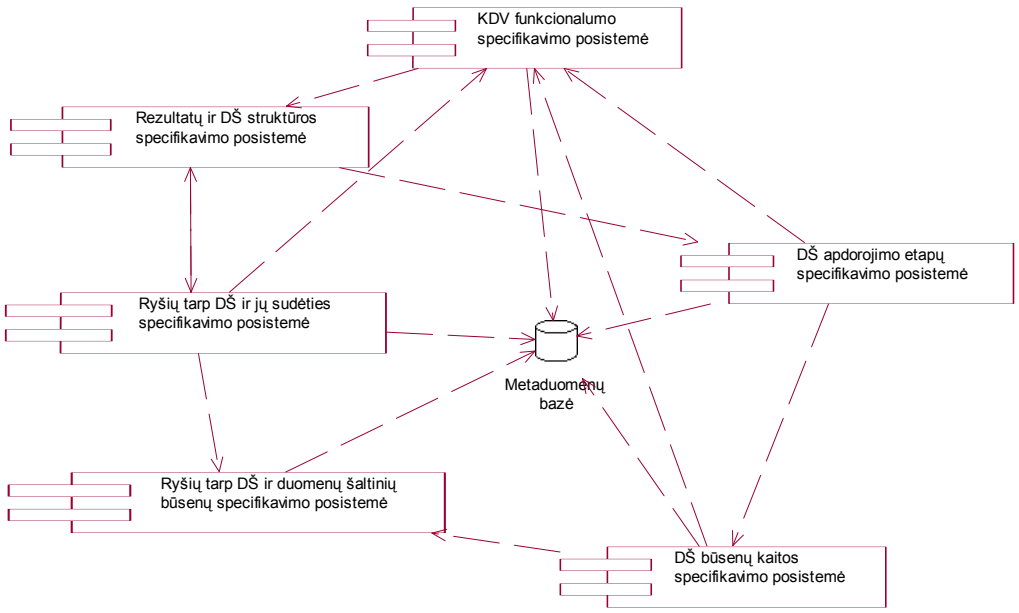
Šiame skyriuje yra specifikuojama "KDV funkcionalumo specifikavimo modelio" projekto architektūra, kurios pagrindu realizuotas CASE įrankio modulis. Kuriamos programų sistemos architektūros parinkimas – svarbus projektinis sprendimas – atliekamas vadovaujantis reikalavimais gautais analizės etape. Architektūros grafinėms pateiktims bus naudojama UML (*Unified Modeling Language)* notacija.

Esminiai reikalavimai, kurie įtakoja architektūros specifikavimą:

- Kuriama CASE įrankio aplinka bus integruota į MS Visio 2000/2002 paketą, todėl kiek įmanoma turi būti naudojami standartiniai Visio sąsajos komponentai.
- KDV funkcionalumo specifikavimo posistemė turi būti integruota į kuriamą sistemą, užtikrinant korektišką CASE įrankio funkcionavimą.

4.2.1 KDV funkcionalumo specifikavimo sistemos sąveika su kitomis sistemomis

Komponentų diagrama (14 pav.) atvaizduoja projektuojamos sistemos vietą kitų sistemų kontekste, parodo, kad funkcijų hierarchijos specifikavimo posistemė yra viena iš CASE įrankio posistemių. Šis modulis kaip ir visi CASE įrankio moduliai naudojasi iš anksto paruošta ir nuolat papildoma bei atnaujinama informacijos srautų specifikacijos metaduomenų baze.

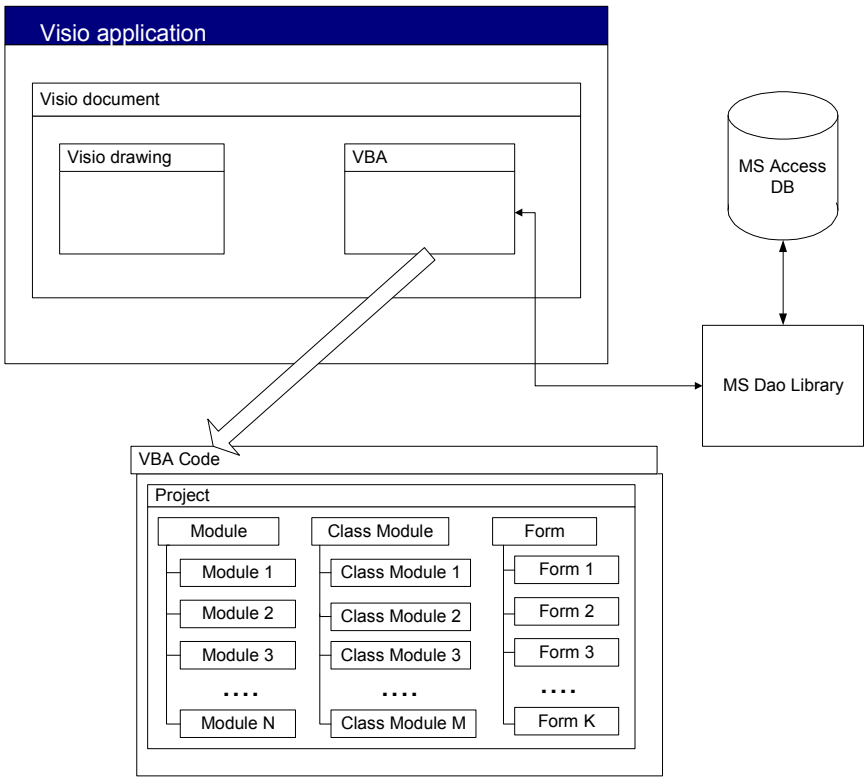


14 pav. KDV funkcionalumo specifikavimo posistemės vieta IS kompiuterizavimo CASE posistemių kontekste

4.2.2 CASE įrankio principinė schema

KDV funkcionalumo specifikavimo įrankiui realizuoti pritaikytas VISIO/2000 paketas. Specifikacijai sudaryti ir saugoti grafinėje notacijoje naudojamas paruoštas failas-šablonas. Vienas Visio failas – vienos funkcijų hierarchijos šablonas. Taip pat visa specifikacija yra išsaugoma specialiai sukurtoje MS Access duomenų bazėje. Principinė įrankio struktūros schema pateikta 15 paveiksle. Žemiau pateiktas šios schemos komentaras:

- Visio aplinka.
- Visio dokumentas. Skirtinguose dokumentuose saugomi atskyri CASE įrankio moduliai.
- Microsoft Access duomenų bazė. Šioje bazėje saugomi specifikuoti elementai.
- Microsoft DAO biblioteka, skirta programinės kalbos priemonėmis pasiekti ir valdyti lokalios arba nutolusios duomenų bazę duomenis, objektus ir struktūrą.
- Visio dokumento grafinė dalis (*drawing*). Kiekvieną grafinę dalį gali sudaryti keli lapai, jie naudojami skaidyti funkcijų hierarchijos diagramą, ir jos detalizavimą pratęsti naujame lape.
- Visio dokumento programinė dalis. Programinę dalį sudaro VBA kodas. Su Visio dokumentu susijęs vienas projektas, kurį sudaro moduliai, klasės moduliai ir ekraninės formos.



15 pav. Įrankio principinė struktūros schema

4.2.3 Sistemos modulių diagrama

Pagrindiniai įrankyje naudojami moduliai yra pavaizduoti 16 pav. Toliau pateikiamas trumpas kiekvieno modulio aprašas:

Duomenu valdymas (modulis). Šis modulis skirtas veiksmams su duomenų baze. Šiame modulyje tikrinama, ar egzistuoja duomenų bazė, pasirenkama duomenų bazė, realizuotas saugojimas duomenų bazėje bei informacijos išgavimas, įterpiamas papildomas meniu punktas standartiniame Visio lange.

Tikrinimas (modulis). Šis modulis yra skirtas loginiam tikrinimui prieš įrašant duomenis į duomenų bazės lenteles. Jame realizuotos procedūros, tikrinančios, ar visi objektai diagramoje yra sujungti, ar visi objektai specifikuoti, ar teisingos ryšių kryptys, ar duomenų bazėje dar nėra specifikuotų kitų elementų, turinčių tokius pačius kodus ar pavadinimus. Taip pat atliekamas specifikacijos pertekliškumo tikrinimas, ar nėra niekur nenaudojamų nedetalizuotų funkcijų, aktorių, aktorių grupių, veiksmų.

Funkcijų hierarchija (modulis). Šiame modulyje realizuotas funkcijų hierarchijos grafinis sudarymas. Priklausomai nuo funkcijos pozicijos modelyje, keičiamas jos grafinis žymėjimas. Taip pat realizuotas funkcijos hierarchijos sudarymo pratęsimas kitame Visio dokumento lape.

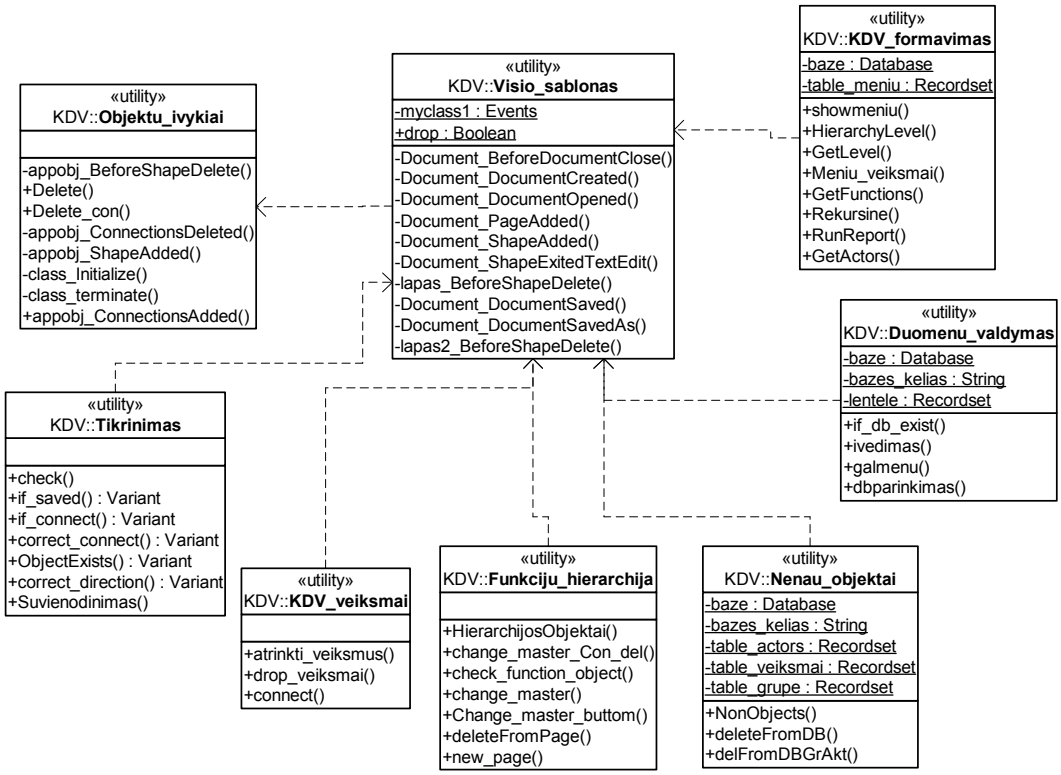
Objektų_įvykiai (klasės modulis). Šiame modulyje realizuotos visos procedūros, kurios turi būti atliktos, priklausomai nuo objektų elgesio: objektų įkėlimo, šalinimo, sujungimo, ryšių pašalinimo.

KDV_veiksmai (modulis). Šiame modulyje realizuotos DŠ apdorojančių ir rezultatus formuojančių veiksmų specifikavimas ir atrinkimas, bei automatinis susiejimas su nedetalizuotomis funkcijomis.

Nenau_objektai (modulis). Šiame modulyje yra atliekamas specifikacijos nepertekliškumo tikrinimas. Pateikiami visi nenaudojami objektai, ir suteikiama galimybė juos šalinti.

KDV_formavimas (modulis). Šiame modulyje atliekamas nurodytos šakos ar aktoriaus KDV struktūros suformavimas. Realizuotas aktoriaus ar funkcijos atrinkimas, funkcijų hierarchijos giliausio lygio apskaičiavimas, sudarytos struktūros pateikimas MS Access ataskaitos forma.

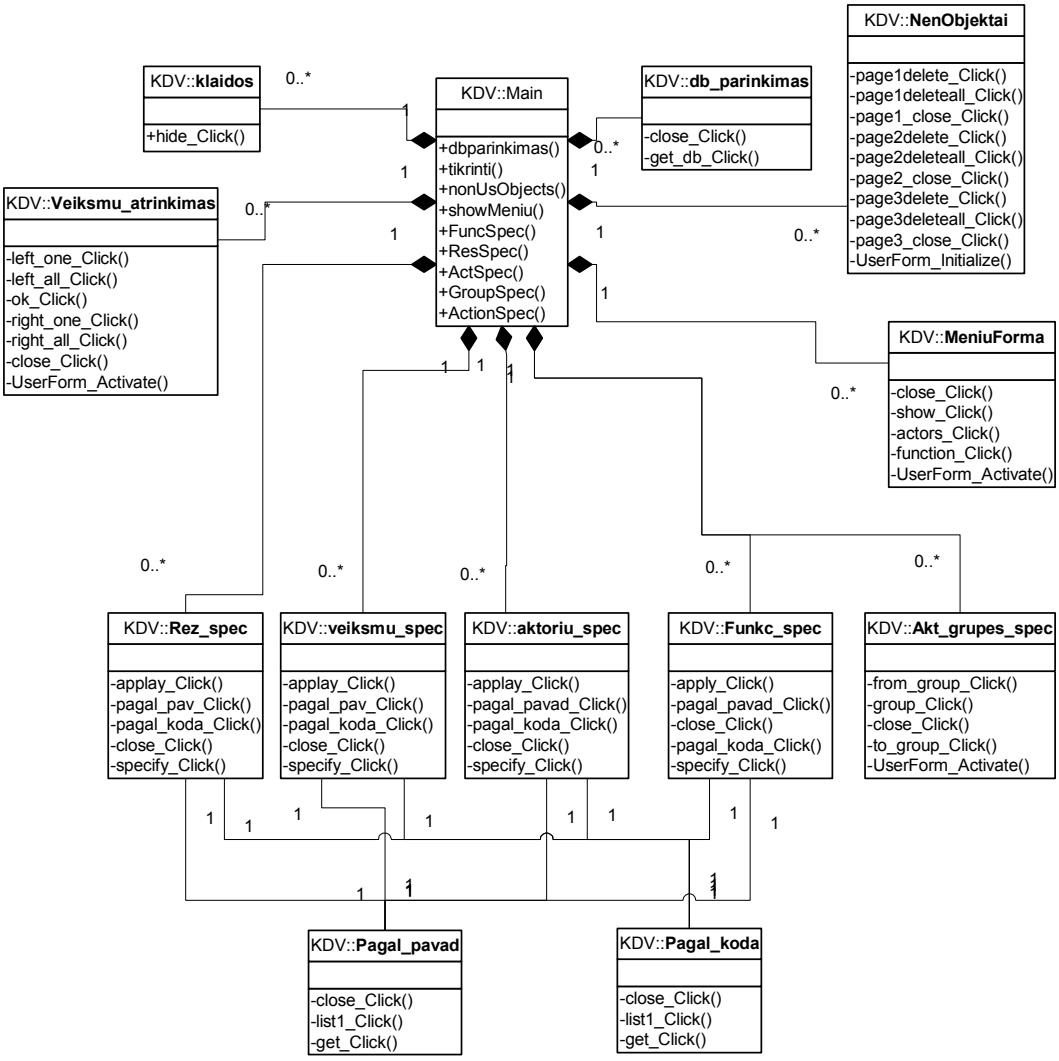
Sablonas (modulis). Šiame modulyje yra realizuoti veiksmai su Microsoft Visio darbo dokumentu, kuriame sudaroma funkcijos diagrama.



16 pav. CASE įrankio moduliai

4.2.4 Sistemos formų diagrama

CASE įrankio posistemę sudarančios formos pateiktos 17 pav. Beveik visos formos yra iškviečiamos naudojant pagrindinę „*Main*“ formą.



17 pav. CASE įrankio formos

Formų, pateiktų 17 pav., trumpas aprašymas:

Main. Tai Visio dokumente integruotas meniu punktas, kurio pagalba yra iškviečiamos visos kitos ekraninės formos.

Klaidos. Forma naudojama pateikti tikrinimo metu aptiktų klaidų sąrašui.

Veiksmu_atrinkimas. Ši forma naudojama kompiuterizuojamų veiksmų iš metabazės atrinkimui ir susiejimui su funkcijomis.

Db_parinkimas. Forma naudojama nurodyti specifikacijos metu naudojamą metaduomenų bazę.

NenObjektai. Forma naudojama nenaudojamų specifikuotų objektų sąrašo pateikimui ir jų pašalinimui iš metabazės.

MeniuForma. Forma skirta KDV sudarymo tipo, priklausomai nuo nurodyto tipo aktoriaus arba funkcijos pasirinkimui.

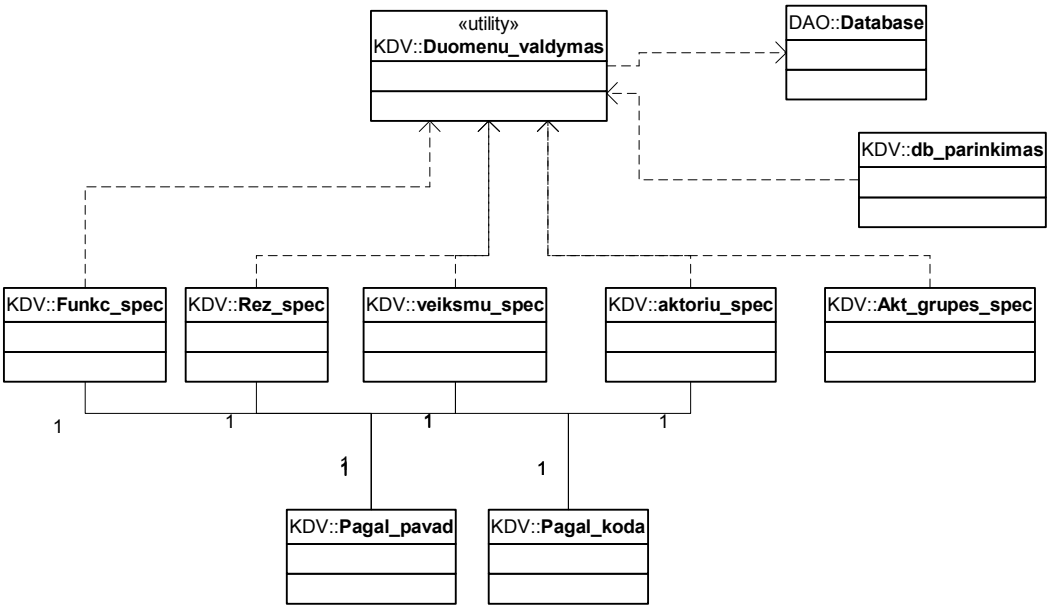
Rez_spec. Ši forma naudojama specifikuojamo funkcionalumo rezultato informacijai įvesti bei priskirti šią informaciją grafiniams diagramos elementams. Analogiška paskirtis yra *veiksmu_spec, aktoriu_spec, funkc_spec* formų. Visos šios formos naudojasi formomis *pagal_pavad* ir *pagal_koda.*

Pagal_koda. Ši forma naudojama surūšiuotai pagal elemento kodą informacijai iš metaduomenų bazės pateikti. Analogiška forma yra *pagal_pavad,* kuri pateikia informaciją pagal elemento pavadinimą.

Akt_grupes_spec. Forma skirta specifikuoti aktorių grupės sudėtį iš jau metaduomenų bazėje specifikuotų aktorių.

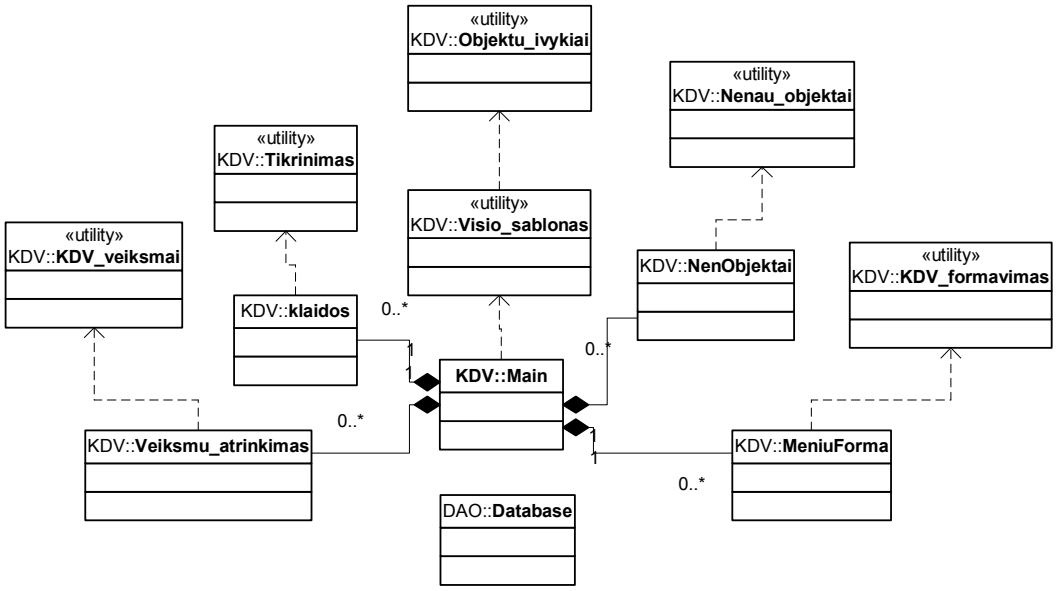
4.2.5 Modulių ir formų sąveikos diagrama

Modulių ir formų sąveikai pavaizduoti buvo sudarytos dvi diagramos. 18 pav. pateikta duomenų valdymo modulio ir elementų specifikavimo formų sąveikos diagrama.



18 pav. Duomenų valdymo modulio ir specifikavimo formų diagrama

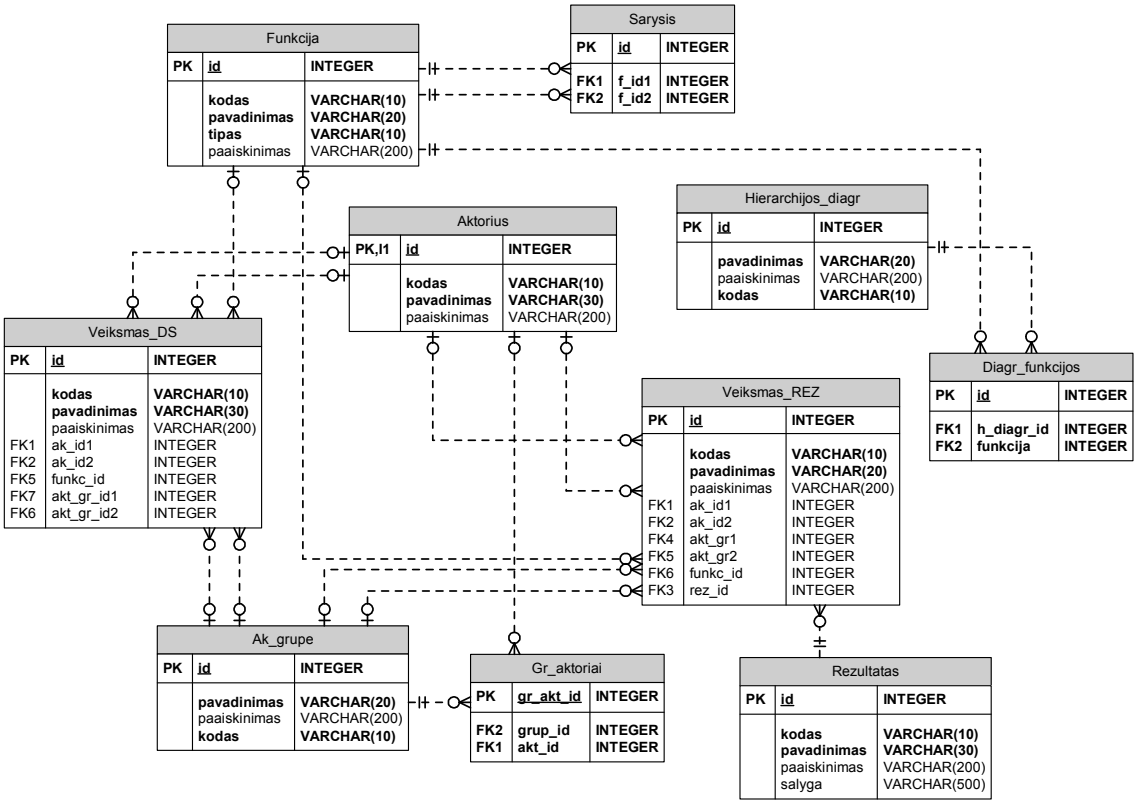
19 pav. pateikta visų likusių modulių ir formų sąveikos diagrama. Šioje diagramoje visi moduliai naudoja *DAO::Database*, todėl ji nėra susieta ryšiais.



19 pav. Formų ir modulių sąveikos diagrama

4.2.6 Duomenų bazės schema

IS saugyklai naudojama metaduomenų bazės schema yra pateikiama 20 pav.



20 pav. Metaduomenų bazės schema

Pilna funkcinių reikalavimų specifikacijos metaduomenų bazė ir lentelių aprašymai pateikti prieduose (Priedas Nr.1).

4.2.7 Kokybės kriterijai

Atliekant architektūros projektavimą bus remiamasi šiais kriterijais :

- efektyvumas – kokių kompiuterio resursų reikia sistemai veikti;
- panaudojamumas – ar sistema pakankamai tiksliai interpretuoja vartotojo įvedamus duomenis, ar yra priimtina vartotojui;
- integralumas – ar sistema gali būti sujungta su kitais moduliais;
- teisingumas – ar sistema pilnai atitinka specifikaciją ir veikia logiškai;
- patikimumas – ar sistema yra tolerantiška klaidoms;
- pakartotinis panaudojimas – ar sistema yra nepriklausoma nuo operacinės sistemos, ar gali būti pakartotinai panaudoti jos komponentai;
- Pernešamumas – ar sistema gali būti lengvai perkeliama į kitą platformą.

5 KDV funkcionalumo specifikavimo modulio eksperimentinis įvertinimas

Realizuotas CASE įrankio modulis ištirtas pagal architektūrinio projektavimo metu nustatytus kokybės kriterijus ir palygintas su panašaus pobūdžio sistema: *Rational Rose Enterprise Edition*.

5.1 Sukurtos sistemos kokybės tyrimas ir įvertinimas

Realiai atlikto darbo kokybės įvertinimo tikslai

Sukurtos sistemos funkcionalumo analizavimas, atitikimas funkcinei specifikacijai ir vartotojo įvertinimas ar jam tinka galutinis rezultatas – visa tai atliekama kokybės įvertinimo procese. Pagrindiniai kokybės vertinimo tikslai:

- Patikrinti, ar realizuota programų sistema atitinka reikalavimų specifikacija. Neatitikimas reikalavimų specifikacijai gali nulemti, kad sukurta sistema nebus naudojama.
- Funkcionavimo, logikos klaidų aptikimas, kurios nebuvo rastos testavimo metu. Paieškos tikslas surasti dar nesurastas klaidas ir surinkti galimų papildymų siūlymus sistemos tolimesniam vystymui.

5.1.1 Kokybės vertinimo procesas

Peržiūros

Realizuotos sistemos kokybė buvo tikrinama įvairiais būdais. Vienas iš jų – peržiūros. Ši sistema buvo analizuota :

- Statiniu būdu. Projekto konsultantas peržiūrėjo sistemos programos išeities tekstą ieškodamas galimų funkcionalumo problemų.
- Interviu su užsakovu. Sistema buvo pristatoma užsakovui evoliuciniu modeliu, todėl jis dalyvavo sistemos projektavimo ir realizavimo etapuose. Sistema buvo keičiama pagal užsakovo pageidavimus.

Interviu su užsakovu

Sistema buvo kuriama evoliuciniu modeliu, todėl nuolat vyko interviu su užsakovu. Pirmiausia buvo surinkti kuriamos sistemos reikalavimai ir parašyta reikalavimų specifikacija. Po reikalavimų surinkimo pradėtas projektavimo ir realizavimo etapas. Pristačius užsakovui

prototipą, kuriame buvo realizuotos pagrindinės reikalavimų dokumente specifikuotos funkcijos, buvo pareikštas pageidavimas papildyti naujomis funkcijomis:

- Diagramos sudarymo pratęsimas naujame lape;
- Rezultatų specifikavimą ir apdorojimą pateikti funkcijų hierarchijos diagramoje.

Taip pat buvo pateiktas pageidavimas pakeisti funkcijų specifikavimo elementų žymėjimą, labiau atitinkantį įprastinį šio elemento žymėjimą. Po pirmosios peržiūros buvo papildytas reikalavimų ir architektūros dokumentas. Sekančių peržiūrų metų nebuvo pateikti papildomi reikalavimai, tačiau pateikti pasiūlymai, kad sukurtas įrankis būtų patogesnis ir priimtinesnis sistemų analitikams.

Sistemos atitikimas specifikacijai

Buvo atliktas tikrinimas, ar sistema atitinka specifikacijoje pateiktus reikalavimus. Nebuvo rasta netenkinamų reikalavimų, todėl vertinama, kad sistema atitinka reikalavimų specifikaciją.

5.1.2 Vertinimo rezultatai

Architektūros dokumente apibrėžti kokybės reikalavimai yra įvertinti 3 lentelėje.

3 Lentelė Programinės įrangos kokybės įvertinimas

Įvertinimas ***Kriterijus***	***l. aukštas***	***aukštas***	***vidutiniškas***	***žemas***	***l. žemas***
Efektyvumas	V				
Panaudojamumas			V		
Integralumas			V		
Teisingumas	V				
Patikimumas	V				
Palaikomumas	V				
Suprantamumas	V				
Pakartotinas panaudojimas		V			
Pernešamumas				V	

Paaiškinimai:

- Pernešamumas – šis kriterijus įvertintas žemu, nes sukurtas produktas yra realizuotas naudojant Microsoft Visio 2000 ir veikia tik Microsoft platformoje.
- Integralumas įvertintas vidutiniškai, nes kiti CASE įrankio moduliai yra dar realizavimo fazėje ir buvo išbandytas darbas tik vienu realizuotu moduliu.

- Panaudojamumas įvertintas vidutiniškai, nes šis modulis dar bus tobulinimas, siekiant jį padaryti patogesniu vartotojui.

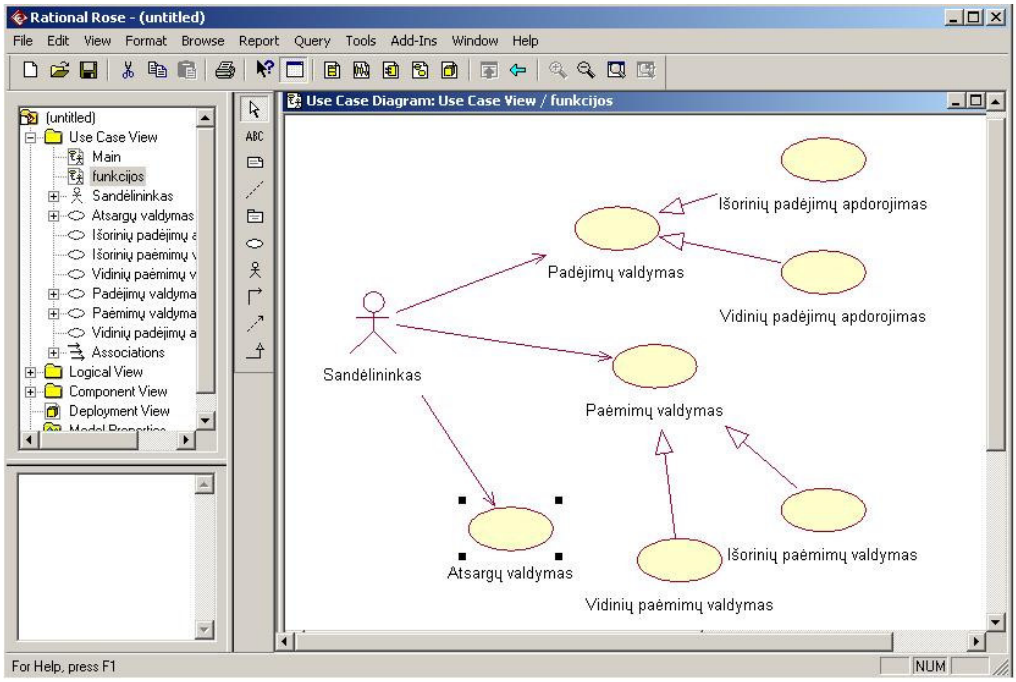
5.1.3 Rational Rose 2000 paketo eksperimentinis tyrimas

Unifikuotas Rational procesas (RUP) yra vienas iš labiausiai paplitusių programinės įrangos kūrimo metodų. RUP nuosekliai ir pakankamai formaliai aprašo visą programinės įrangos kūrimo procesą, pradedant veiklos modeliavimu, baigiant sukurtos sistemos pateikimu vartotojui. Modelių sudarymui naudojama UML notacija.

Rational Rose 2000 nėra realizuotas automatizuotas KDV struktūros projektavimas, tačiau juo naudojantis galima specifikuoti KIS kontekstą, bei atlikti KDV projektavimą, rankiniu būdu suformuojant KDV sudėtį.

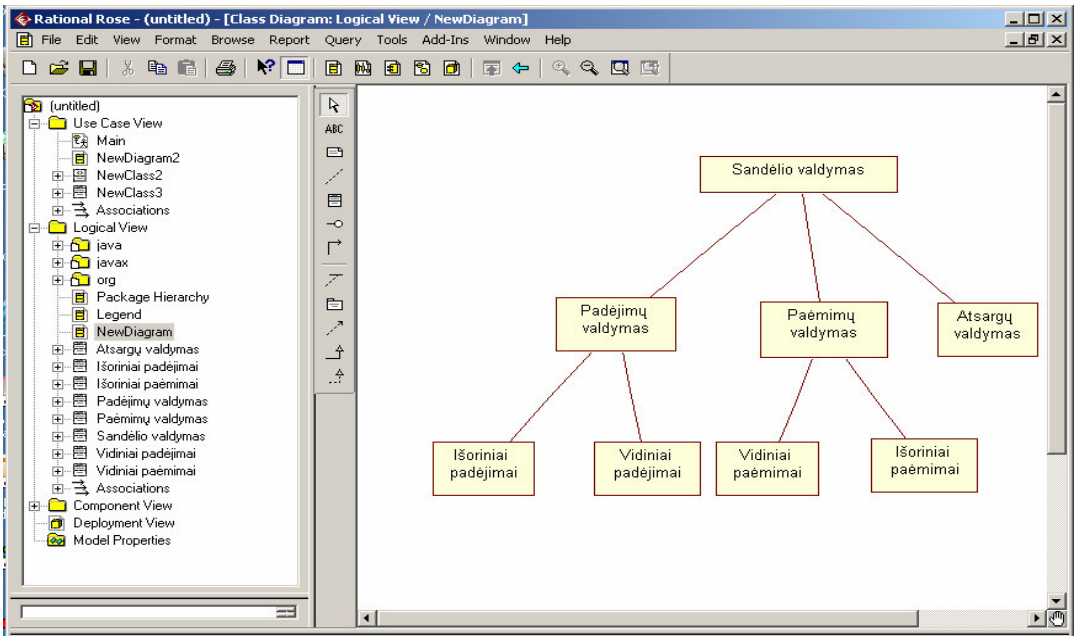
Šiame pakete nėra diagramos, kuri būtų skirta funkcijų hierarchijos sudarymui, todėl bus sudaromos įvairios diagramos skirtinguose sistemos pjūviuose.

Panaudojimų atvejų vaizde (21 pav.) funkcijas galima pavaizduoti panaudojimo atvejų diagrama. Tačiau šioje diagramoje daugiau modeliuojamos kiekvieno vartotojo atliekamos funkcijos, o ne visos organizacijos.



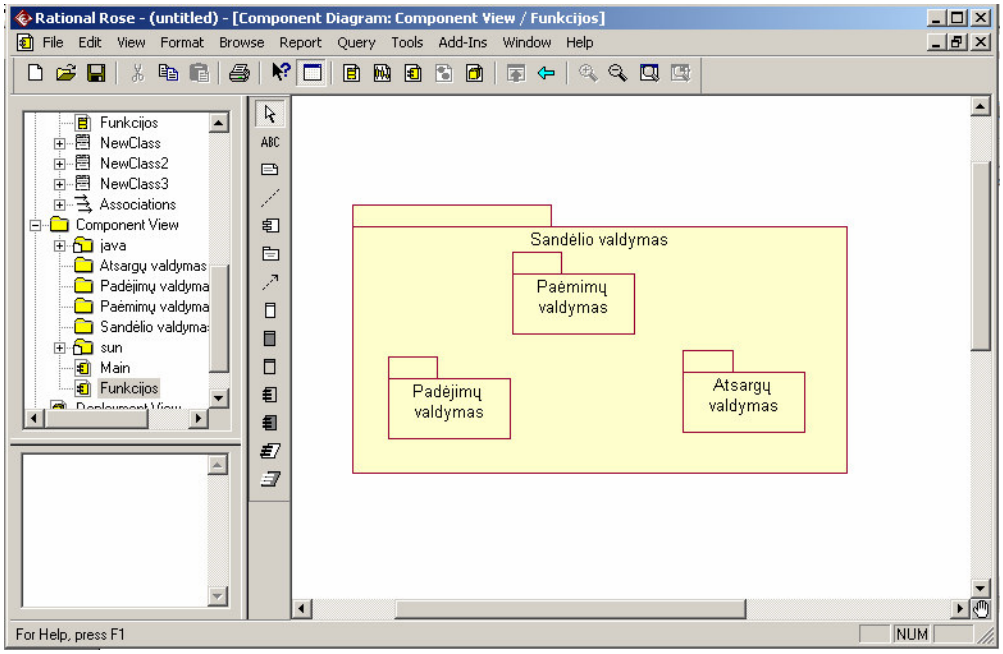
21 pav. KIS konteksto specifikavimas, naudojant panaudojimų atvejų diagramą

Loginiame vaizde funkcijoms modeliuoti galima naudoti klasių diagramą. Šioje diagramoje galima sudaryti hierarchinę struktūrą (22 pav.).



22 pav. Konteksto specifikavimas, naudojant klasių diagramą

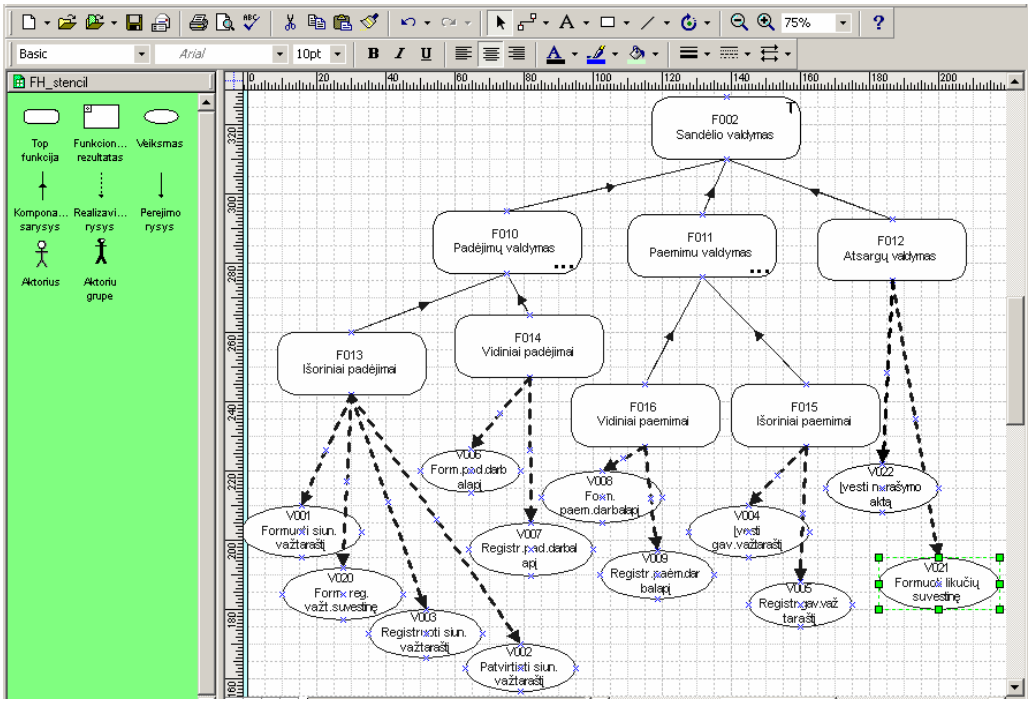
Komponentų vaizde funkcijoms modeliuoti galima panaudoti paketus (23 pav.). Toks modeliavimas nėra labai patogus, nes kiekvieną paketą detalizuojantys paketai vaizduojami jo viduje.



23 pav. Funkcijų modeliavimas, naudojant komponentų diagramą

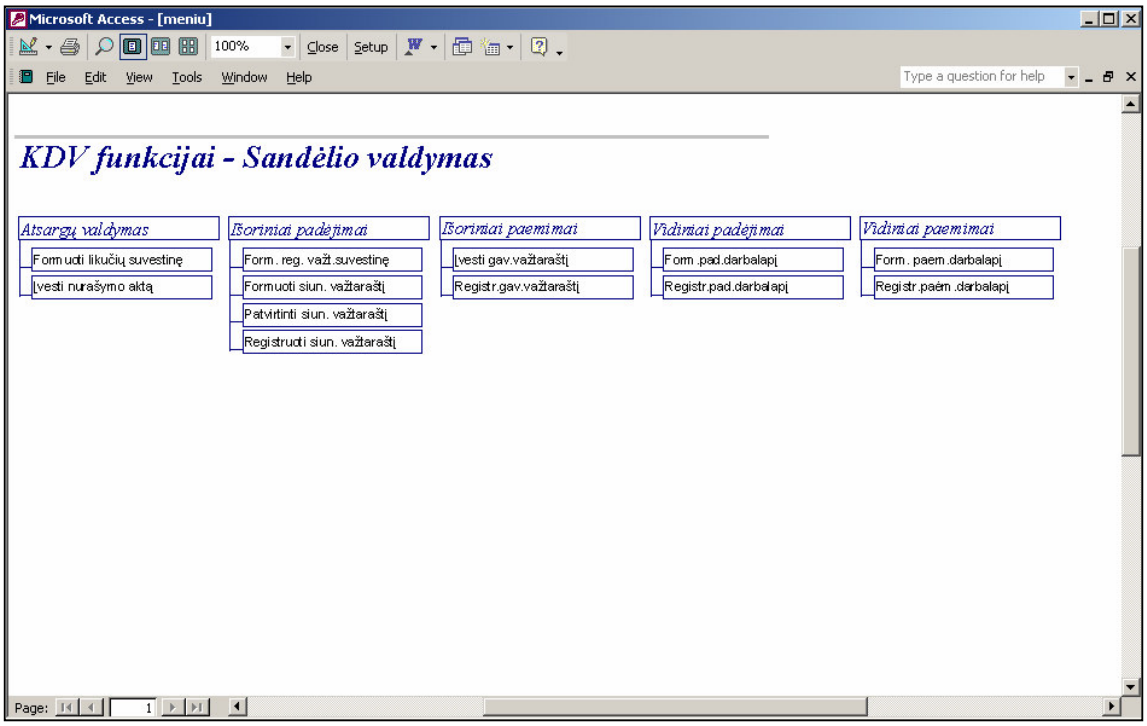
5.1.4 Sukurto įrankio panaudojimas KDV funkcionalumo specifikavimui

Realizuotame įrankyje KIS konteksto specifikavimui naudojama funkcijų hierarchijos diagrama (24 pav.).



24 pav. KIS konteksto specifikavimas, naudojant realizuotą CASE įrankio modulį

Specifikavimo metu išgauti duomenys išsaugomi metaduomenų bazėje tolesniems projektavimo etapams.



25 pav. Automatiškai sugeneruotas KDV projektinis variantas

5.1.5 Eksperimentinio tyrimo įvertinimas

Tokie paketai kaip *Rational Rose* turi gerai išvystytų priemonių rinkinį dalykinės srities analizei ir modeliavimui bei kitiems sistemų projektavimo etapams. Tačiau jis neteikia priemonių automatizuotam KDV struktūros projektavimui, taip pat nepalaiko tiesioginio darbo su metaduomenų baze.

Realizuotas KDV funkcionalumo specifikavimo modelis nėra dar tiek išvystytas kaip komerciniai tokio pobūdžio paketai. Tačiau modelis palaiko automatizuotą KDV projektavimą, taip pat yra skirtas tiesioginiam darbui su informacijos srautų specifikacijos saugykla, t.y. metaduomenų baze. Naudojama paprasta ir aiški notacija, funkcijų hierarchijos modelis lengvai suprantamas ir palengvina analitiko ir vartotojo bendravimą.

6 Išvados

1. Išanalizuoti kompiuterizuojamos sistemos konteksto specifikavimui naudojami Oracle Case ir IDEF0 metodai, aptarti jų privalumai ir trūkumai, į kuriuos buvo atsižvelgta sudarant KIS konteksto specifikavimo modelį.
2. Apžvelgti KDV funkcionalumo specifikavimo ir struktūros sudarymo principai ir rekomendacijos, kurios įvertintos formuojant kompiuterizuotą darbo vietą.
3. Pristatytas KDV funkcionalumo specifikavimo modelis funkcinių reikalavimų specifikavimo metodo kontekste, kuris pateikia KIS specifikavimo ir lankstaus kompiuterizuotos darbo vietos formavimo būdą. Pateikti formalūs funkcijų hierarchijos ir KDV formavimo apribojimai.
4. Realizuojant CASE įrankio modulį, pagal pasiūlytą KDV funkcionalumo modelį, buvo įvykdyti ir dokumentuoti visi programinės įrangos kūrimo etapai, sudarant reikalavimų, architektūros, detalios architektūros specifikacijas ir vartotojo vadovą.
5. Sukurtas CASE įrankio modulis pateiktam modeliui realizuoti, integruotas į Visio 2000 aplinką praplėtus jo funkcionalumą Visual Basic programavimo kalba. Šis modulis leidžia sudarytą specifikaciją išsaugoti grafiškai ir metaduomenų saugykloje, kuriai realizuoti buvo panaudota MS Access DBVS. Informacija, išsaugota duomenų bazėje, naudojama kituose funkcinių reikalavimų specifikavimo etapuose, o grafiškai išsaugota informacija palengvina sistemų analitiko darbą su vartotoju.
6. Realizuotas CASE įrankio modulis palengvina sistemos analitiko – projektuotojo darbą, nes automatizuoja KDV projektavimą, bei atlieka dalinį sudaryto modelio pilnumo ir nepertekliškumo verifikavimą.
7. Atliktus programinės įrangos kokybės vertinimą pagal užduotus kriterijus paaiškėjo, kad sukurta sistema veikia nepažeisdama apribojimų. Eksperimentinio tyrimo metu, lyginant realizuotą įrankį su Rational Rose paketu, nustatyta, kad šis CASE įrankio modulis neprilygsta komerciniams paketams, tačiau leidžia dirbti su metaduomenų baze bei palaiko automatinį KDV struktūros projektavimą.
8. Magistrinio darbo tematika pristatyti straipsniai 2004m. sausio 28-29d. KTU vykusioje konferencijoje “Informacinės technologijos 2004” ir 2004m. gegužės 28d. VUKHF vykusioje konferencijoje “Informacinės technologijos verslui – 2004”.

7 Literatūra

1. **Butkienė R**. Informacijos sistemai keliamų funkcinių reikalavimų specifikavimo metodas: daktaro disertacija. KTU – Kaunas: Technologija, 2002.
2. **Ambler. S. W.** The Object Primer 2nd edition, building object applications that work, and Proccess paterns. Cambridge University Press, 2000.
3. **Souchon N., Limbourg Q., Vanderdonct J.,** Task Modelling in Multiple Contexts of Use. Lecture Notes on Computer Sciences, Vol. 2545, pp. 55-73, Springer – Verlag Berlin Heidelberg 2002.
4. **Savidis A., Stephanidis C**., Unified user interface design: designing universally accessible interactions. Interacting with computers, 2004.
5. **Hay D.C.,** Requirements Analysis: From Business Views to Architecture. Prentice Hall PTR, 2002.
6. **Raskin J**., The humance interface, new direction for designing interactive systems. Addison-wesley, 2000.
7. **Hackos J.T., Redish J.C.,** User and task anlysis for interface design. Wiley, 1998.
8. **Maciaszek L. A.,** Requirements analysis and system design developing information systems with UML. Addison-wesley, 2001.
9. **Harmelen M. V.**, Object modeling and user interface design. Addison-wesley, 2001.
10. **Muller P.,** Instant UML. Wrox press Ltd.,1997.
11. **Butkienė R**., Informacijos sistemų projektavimas Oracle Designer/2000 priemonėmis.Kaunas:Technologija, 1998.
12. **Butkienė R., Butleris R.,** The Approach for the User Requirements Specification. 5th East-European conference ADBIS'2001, Research Communications, Ed. by A.Čaplinskas, J.Eder. Vilnius, 2001, p.p. 225-240.
13. **Barker R., Longman C.,**. Case Method, function and process modelling. Addison-wesley, 1992.
14. **Butleris R., Danikauskas T., Misevičiūtė B.,** Informacijos sistemos kompiuterizuotų darbo vietų projektavimas. Iš: Informacinės technologijos 2004: konferencijos pranešimų medžiaga, Kaunas 2004 sausio 28 - 29 d.. Kaunas, 2004, p. 533-538.Informacinės technologijos 2004
15. **Puerta A. E.,** Supporting User-Centered Design of Adaptive User Interfaces Via Interface Models[interaktyvus], Standford, USA, 1998.[žiūrėta 2004.01.20]. Priega per internetą: www-smi.stanford.edu/pubs/SMI_Reports/SMI-98-0747.pdf
16. Developing a UI Design from a UML Color Model. www.uidesign.net, [žiurėta 2004.01.15]. Prieiga per Internetą: http://www.uidesign.net/1999/papers/UML_UI.html
17. Chessboard Layout Pattern. www.uidesign.net, [žiurėta 2004.01.15]. Prieiga per Internetą: http://www.uidesign.net/1999/papers/Chessboard.html

18. Integration Definition For Function Modeling (Idef0). Draft Federal InformationProcessing Standards Publication 183, 1993 December 21.
19. R.J. Mayer, J.H. Crump, R. Fernandes, A. Keen, M.K. Painter. Information integration for concurrent engineering(ICEE) compendium of method report, 1995.[žiūrėta 2004.04.11]. Prieiga per internetą: http://www.idef.com/Downloads/pdf/compendium.pdf
20. **Danikauskas T., Misevičiūtė B.,** Reikalavimų specifikacijos modeliais grindžiamas IS vartotojo sąsajos projektavimas. Iš: Informacinės technologijos verslui 2004: konferencijos pranešimų medžiaga, Kaunas 2004 gegužės 28. Kaunas, 2004.Informacinės technologijos verslui 2004.

8 Terminų ir santrumpų žodynas

Kompiuterizuota darbo vieta(KDV) - savarankiška informacinė sistema arba didesnės IS dalis. Ji dažniausiai susijusi su tam tikru žmogaus organizacijoje atliekamu vaidmeniu, kurio funkcijas ji padeda vykdyti.

Informacinė sistema (IS) – tai sistema, kuri surenka, apdoroja, saugo ir platina informaciją, padedančią kontroliuoti ir koordinuoti organizacijos poreikius, priimant valdymo sprendimus, analizuojant problemas, kuriant naujus produktus

Informacijos srautų specifikacija – tai KIS keliamų funkcinių reikalavimų specifikacija, aprašanti, kokio tipo ir kokios struktūros turi būti KIS išvedama informacija, t.y. KIS funkcionalumo rezultatai, bei juos formuojantys duomenų šaltiniai ir duomenų srautai. Ši specifikacija yra saugoma metaduomenų saugykloje.

Kompiuterizuota informacinė sistema (KIS) – informacijos sistema, kurioje visiems arba tam tikrai daliai uždavinių atlikti taikomos kompiuterinės technologijos.

9 Abstract

The workspace design is usually treated as a part of the user interface design without paying a lot of attention to a workspace design. Such approach works rather well when small IS, which consists of tens of functions and only few actors, is being designed. However, in such case finding optimal and fast solutions can become problematic when IS workspace is being designed for large scale IS.

The purpose of this work is to present a model of computerised workspace functionality specification, using information flows specification. The creation of workspace structure is based on function hierarchy model. After making analysis of computerized workspace design and functions modeling, the function hierarchy modeling technique and process of building computerized workspace should be presented, implemented and verified. Also the CASE tool for function modeling and structure of workspace design is described.

10 Priedai

10.1 Pirmasis priedas. Metabazės schema ir jos aprašymas.

10.2 Antrasis priedas. Realizuoto įrankio vartotojo vadovas.

10.3 Trečiasis priedas. Parengtas publikavimui straipsnis.

Straipsnis, parengtas publikavimui konferencijos pranešimų medžiagoje "Informacinės technologijos verslui - 2004".

10.4 Ketvirtasis priedas. Straipsnio kopija.

Straipsnis "Informacijos sistemos kompiuterizuotų darbo vietų projektavimas"