KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Zenonas Šleinius

Automatizuotas kompiuterizuotos IS prototipo kūrimas informacijos srautų specifikacijos pagrindu

Magistro darbas

Darbo vadovė
doc. R. Butkienė

Kaunas, 2004

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

TVIRTINU
Katedros vedėjas
(parašas) doc. dr. R. Butleris
2004 05 24

Automatizuotas kompiuterizuotos IS prototipo kūrimas informacijos srautų specifikacijos pagrindu

Informatikos mokslo magistro baigiamasis darbas

Kalbos konsultantė
Lietuvių kalbos katedros lektorė
(parašas) dr. J. Mikelionienė
2004 05 24

Vadovė
(parašas) doc. dr. R. Butkienė
2004 05 21

Recenzentas
(parašas) doc. dr. P.Kanapeckas
2004 05 21

Atliko
IFM-8/4 gr. stud.
(parašas) Z. Šleinius
2004 05 21

Kaunas, 2004

Turinys

SUMMARY

The domain of this research are CASE tools, designated for requirements engineering, which are able to generate computerized information system prototypes. The main goal of this project is to design software, which would enable to generate an IS prototype from requirements specification, created re specification method, suggested by the Information Systems department. The software is a part of a CASE tool, designed by the IS department. The CASE tool is designated for creating requirements specification and the software will provide additional functionality to it.

The opportunities of computerized IS prototyping using *Oracle Designer, Visual FoxPro* and *MS Access* were analyzed. In case of *Oracle Designer,* in order to create a computerized IS prototype we have to go through the whole design stage. In case of DBMS *Visual FoxPro* and *MS Access* we must have a database designed. And the software for generating computerized IS prototype enables you to generate a system prototype at an early design stage and without having the system project.

1. Įvadas

Sistemų inžinerijos patirtis per paskutiniuosius 30 metų parodė, kad yra daug didesnė tikimybė, jog kompiuterinė sistema elgsis pagal kliento norus, bus pabaigta laiku ir biudžeto ribose, jei jos kūrimui naudojamas specialus metodas. „Krioklio modelis" buvo pirmasis programinės įrangos kūrimo metodas. Pagal šį metodą reikalingos sistemos kūrimas vykdomas, konkrečia tvarka atliekant tam tikrą fazių skaičių. Paprastai šios fazės apima reikalavimų analizę, sistemų specifikaciją, aukšto lygio projektavimą, žemo lygio projektavimą ir kt.

Pastaruoju metu pasiūlyti ir išbandyti kiti metodai. Vienas iš jų, žinomas kaip evoliucinis prototipų kūrimas, siūlo sistemos prototipą sukurti kiek įmanoma greičiau iš pirmo reikalavimų projekto. Prototipas vartotojams parodomas kaip realizuojamos sistemos išvaizdos ir elgesio imitacija, o vartotojai kviečiami pakeisti savo pradinius reikalavimus, spręsdami apie imitacijos ir tikros jų norimos sistemos atitikimo laipsnį. Šis prototipų kūrimo ir validavimo ciklas tęsiasi, kol abi pusės tampa patenkintos prototipu. Kai tai įvyksta, tuometiniai reikalavimai įvedami į formalesnio sistemos kūrimo proceso pirmą fazę.

Reikalavimų inžinerijoje naudojami metodai ir CASE įrankiai daugiausia skirti viduriniams ar vėlesniems informacinės sistemos (IS) kūrimo etapams. Dabartiniai CASE įrankiai negali pradiniuose projektavimo etapuose sukurti kompiuterizuojamos informacinės sistemos prototipo. Jie nepadeda projektuotojui reikalavimų validavimo procese.

Šio darbo tikslas yra sukurti programą, kuri kurs kompiuterizuotos informacinės sistemos (KIS) prototipą pagal sudarytą reikalavimų specifikaciją. Tai bus priedas prie sistemos reikalavimų specifikacijai sudaryti ir kuriamam įrankiui suteiks papildomo funkcionalumo.

Šis įrankis padės sumažinti atotrūkį tarp vartotojo ir informacinės sistemos kūrėjo. Jis leis vartotojui ankstyvose IS kūrimo stadijose per sukurtą prototipą pamatyti, ar tiksliai vykdomi jo reikalavimai. Tai pagerins ir pagreitins tolesnį IS kūrimą.

2. Analizės dalis

2.1. Tyrimo sritis, objektas ir problema

Tyrimo sritis – CASE įrankiai, skirti reikalavimų inžinerijai, kuriantys kompiuterizuotų IS prototipus.

Tyrimo objektas – Informacijos sistemų katedroje kuriamas CASE įrankis, skirtas reikalavimų inžinerijai.

Nauja informacijos sistema kuriama tam tikrai problemai (ar jų aibei), su kuria susiduria organizacija, spręsti. Sukurtoji informacijos sistema yra tam tikrų įvykių, vadinamų sistemos vystymu, rezultatas. Sistemų vystymo procesas susideda iš tokių veiklų:

1. Sistemos analizė – problemų, kurios bus sprendžiamos panaudojant kuriamą IS, analizė.
2. Sistemos projektavimas – detalizuojama, kaip IS tenkins sistemos analizės metu nustatytus informacinius reikalavimus.
3. Programavimas – procesas, pervedantis sistemos specifikaciją, gautą projektavimo metu, į programos kodą.
4. Testavimas – procesas, kurio metu nustatoma, ar sistema duoda norimus rezultatus nurodytomis sąlygomis.
5. Transformacija – procesas, pakeičiantis seną sistemą nauja.
6. Eksploatavimas – IS vartotojas ir techninis specialistas patikrina, kaip naujoji sistema tenkina pirminius jos tikslus.
7. Palaikymas – techninės įrangos, programinės įrangos, dokumentacijos, procedūrų pakeitimas gamybos sistemoje, norint pataisyti klaidas, tenkinti naujus reikalavimus ar pagerinti gamybos efektyvumą.

Informacijos sistemos dalykinės srities analize vadinamas informacijos apie dalykinę sritį rinkimo, apibendrinimo ir suvokimo procesas.

Analizės metu siekiama nustatyti, kokias užduotis turi vykdyti kuriamoji IS, t. y. siekiama išsiaiškinti užsakovo pageidavimus bei naudotojo poreikius, suformuluoti kuriamos sistemos reikalavimus ir sukaupti dalykinės srities specialistų naudojamų žinių bazę. Nė viena sistema negali būti suprojektuota nesukaupus ir nepanaudojus atitinkamos dalykinių žinių bazės. Pavyzdžiui, neįmanoma sukurti kompiuterizuotos buhalterinės apskaitos sistemos nesusipažinus su pačia buhalterine apskaita.

Vienas sunkiausių sistemos analitiko uždavinių – reikalavimų informacijos sistemai nustatymas, pradedamas spręsti ankstyvojoje IS vystymo stadijoje. Tai sritis, kuri kelia daug sunkumų sistemos tyrėjui ir kurioje daug pastangų nueina veltui. Informacijos reikalavimai –

informacinių poreikių, kuriuos turi tenkinti naujoji sistema, formuluotė, išdėstymas. Jie nurodo, kam reikalinga toji informacija, taip pat kada, kur ir kaip toji informacija yra reikalinga. Reikalavimų analizės metu rūpestingai apibrėžiami naujos ar modifikuojamos sistemos tikslai, ir išsamiai aprašomos naujos sistemos funkcijos. Specifikuojant reikalavimus turi būti atsižvelgta į ekonominius, techninius ir laiko apribojimus, taip pat į organizacijos tikslus, procedūras ir sprendimų procesus. Klaidos reikalavimų analizėje yra pagrindinė kuriamos sistemos nesėkmės ir didelių sistemos vystymo išlaidų priežastis. Sistema sukurta remiantis klaidingais reikalavimais bus atmesta arba turės būti iš esmės peržiūrėta. Todėl reikalavimų analizės svarba turi būti įvertinta.

Reikalavimų inžinerija yra iteratyvus procesas, kurį sudaro tokios fazės: reikalavimų išgavimas, reikalavimų analizė, reikalavimų specifikacijos sudarymas, reikalavimų validavimas ir reikalavimų panaudojimas. Darbas bet kurioje fazėje gali baigtis būtinybe sugrįžti prie vartotojo, arba norint išgauti iš jo naujus reikalavimus, arba siūlant egzistuojančius reikalavimus pakeisti ar pašalinti.

Prieš pradedant sistemos kūrimo procesą, reikia apibrėžti sistemos reikalavimų aibę. Taigi pirma reikalavimų inžinerijos pagrindinė funkcija yra surinkti visus aktualius reikalavimus. Reikalavimų išgavimo fazės tikslas – apibrėžti pilną reikalavimų aibę, kurie laikomi būtinais sistemos kūrimui, iš vartotojų, klientų, analitikų ir t. t. ir išreikšti juos tinkama forma. Iš pradžių reikalavimai išreiškiami neformaliai, natūralia kalba. Vėliau, sudarant reikalavimų specifikaciją, jie gali būti išreikšti formalesnėmis notacijomis.

Kita reikalavimų inžinerijos pagrindinė funkcija – išanalizuoti surinktus reikalavimus. Šios fazės tikslas – susisteminti reikalavimus į logiškai susijusias grupes (pavyzdžiui, kartu surinkti visus reikalavimus našumui) ir juos kritiškai apžvelgti. Apžvalgos metu bandoma nustatyti, ar reikalavimai neprieštarauja vieni kitiems, ar nėra neaiškių reikalavimų, ar nėra reikalavimų, pareiškiamų daugiau kaip vieną kartą, ar netrūksta reikalavimų tam tikroms sritims. Šios analizės pagrindu vartotojams ir klientams gali būti surengta akistata su analitiku, kad kartu jie pabandytų išspręsti prieštaravimus, išsiaiškinti nesuprantamus teiginius, pašalinti dubliavimus ir t. t.

Pradžioje išgauti reikalavimai išreiškiami neformaliai, natūralia kalba, pvz., anglų. Kai kuriems reikalavimų inžinerijos metodams tai galutinis jų išreiškimo būdas. Tačiau kitiems metodams pradinė neformali reikalavimų išraiška gali būti paversta, galbūt per vieną ar daugiau tarpinių pateikimų, į formalesnę reikalavimų išraišką.

Aktualu patikrinti, ar sudaryta reikalavimų specifikacija teisingai atspindi vartotojo bei užsakovo išsakytus reikalavimus, t. y. sudarytą reikalavimų specifikaciją būtina validuoti. Tai padaryti galima dviem būdais: neautomatizuotu ir automatizuotu (panaudojant animaciją diagramose, sukuriant IS prototipą). Neautomatizuoto validavimo trūkumas yra toks, kad

sudarytos reikalavimų specifikacijos skirtingų tikrintojų gali būti interpretuojamos skirtingai. Animacijos trūkumas – animuojamą specifikaciją gerai supranta inžinierius, o vartotojas ar užsakovas be specialaus pasiruošimo jos gali ir nesuprasti. Sistemos prototipas šiuo atžvilgiu yra pranašesnis, nes vartotojas ir užsakovas gali pamatyti ir patikrinti, kaip jo reikalavimus suprato inžinierius.

Po sėkmingo validavimo reikalavimai naudojami vienam iš šių tikslų:

- rankiniam projektavimui,
- automatiniam projektavimui (įskaitant procedūrų generavimą),
- vykdymui (kaip produkto prototipui),
- ryšio priemonės tiekimui (pvz., tarp projekto narių),
- sistemos ir vienetų testų generavimui,
- produkto realizacijos valdymui,
- buvimui sutartimi tarp kliento ir tiekėjo,
- atliekant vaidmenį konkurencingos kainos pasiūlymo procese.

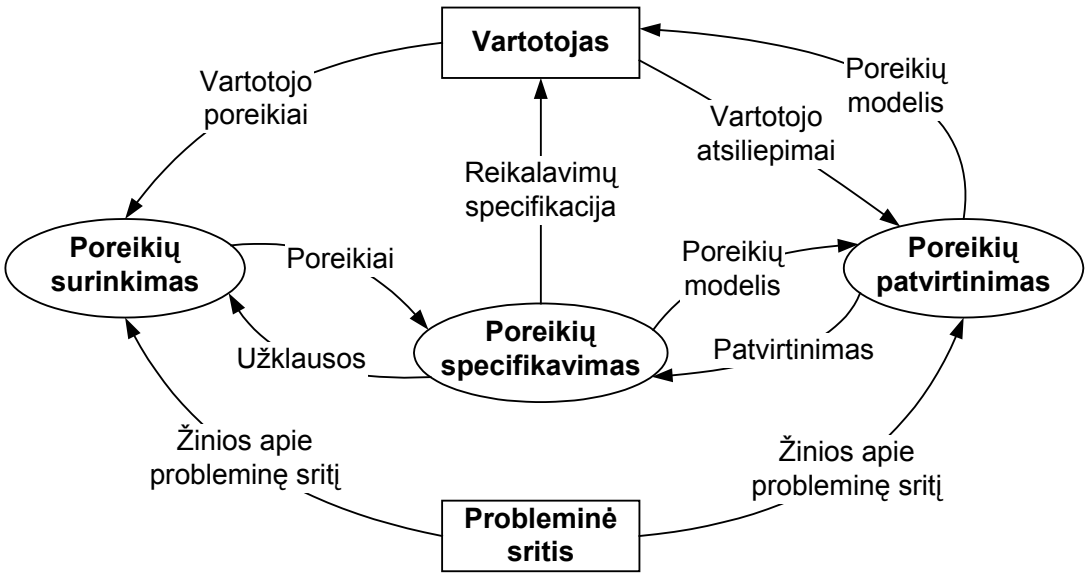
Daugelis esamų įrankių neturi galimybės sukurti būsimos sistemos prototipo arba reikia įdėti daug darbo rašant programinį kodą. Pavyzdžiui, *Rational Rose* paketas sugeneruoja klasių aprašus, bet metodus reikia rašyti pačiam. *Oracle Designer* paketo trūkumas yra tas, kad jis daugiau yra skirtas sistemos projektuotojams. Naudojantis juo sistemos prototipą galima sukurti tik sudarius jos projektą. Tačiau yra poreikis patikrinti reikalavimų specifikaciją dar neperėjus prie sistemos projektavimo. *Oracle Designer* pakete nėra apgalvotos programos struktūros.

Be to, kiekvienas CASE įrankis automatizuoja tam tikrą informacinių sistemų kūrimo metodą. Pavyzdžiui, *Rational Rose* paketas automatizuoja *Rational Unified Process* (RUP) metodą, *Oracle Designer* – *Oracle* CASE metodą [2]. KTU kuriamas CASE įrankis, automatizuojantis Informacijos sistemų katedroje sukurtą reikalavimų informacinei sistemai specifikavimo metodą, nes šiai metodikai automatizuoti netinka nei vienas iš sukurtų CASE įrankių.

2.1.1. Reikalavimų inžinerijos procesas

Vienas iš svarbiausių analizės resursų yra žmonės. Pats vienas informacijos sistemų reikalavimų negali suformuoti nei užsakovas, nei tiekėjas. Tiekėjas nepajėgia pakankamai giliai suprasti užsakovo problemų ir suvokti visų niuansų, o užsakovas yra nesusipažinęs su sisteminės analizės metodais ir dažniausiai nesugeba susisteminti reikalavimų ir perteikti jų projektuotojams suprantamais terminais. Todėl tiekėjo ir užsakovo atstovai reikalavimus turi formuluoti kartu. Tiekėjui atstovauja vadinamieji sisteminiai analitikai, o užsakovui – dalykinės srities specialistai.

Sisteminiai analitikai – tai specialistai, susipažinę su sisteminės analizės metodais ir turintys praktinius tų metodų taikymo įgūdžius. Jie yra tarpininkai tarp būsimų sistemos naudotojų ir jos projektuotojų. Pagrindinis sisteminių analitikų uždavinys – kartu su užsakovais ir naudotojais suformuluoti informacijos sistemos reikalavimus ir pateikti juos sistemos



1 pav. Reikalavimų inžinerijos procesas

projektuotojams. Kita vertus, sisteminiai analitikai privalo išaiškinti užsakovams ir naudotojams projektuotojų priimamų sprendimų esmę. Todėl sisteminiai analitikai turi palaikyti glaudžius ryšius ir su užsakovais, ir su projektuotojais.

Reikalavimų inžinerijos procesas pavaizduotas 1 pav.

Informacijos šaltiniai yra trys:

1) užsakovai, dalykinės srities ekspertai bei asmenys, vykdantys dalykinėje srityje vykstančius procesus,
2) rašytiniai šaltiniai,
3) tiesioginis vykstančių procesų stebėjimas.

Sisteminės analizės sėkmė priklauso nuo ją atliekančio asmens sugebėjimų, intuicijos ir praktinių įgūdžių. Nėra tikslių rekomendacijų, kaip rinkti ir apibendrinti reikalingą informaciją.

Pagrindiniai informacijos rinkimo metodai:

1) struktūrinis ir nestruktūrinis interviu,
2) tikslų analizė,
3) scenarijai,
4) formų analizė,
5) natūralios kalbos metodai.

Pagrindinė informacijos rinkimo etapo problema – tarpusavio bendravimas. Analitikams, būsimiems kuriamos programos naudotojams ir dalykinės srities ekspertams vieniems kitus suprasti yra gana sudėtinga. Jie kalba skirtingų profesijų žargonais, naudoja skirtingas mąstymo schemas, dažniausiai neturi psichologinių žinių, būtinų darbui kolektyvuose, sudarytuose iš skirtingas specialybes turinčių asmenų.

Labai svarbi terminų problema. Daugelyje dalykinių sričių nėra griežtos, nusistovėjusios terminijos, terminai neturi vienareikšmės interpretacijos – skirtingi asmenys juos skirtingai aiškina.

Bendrauti trukdo ir skirtingos mąstymo schemos. Kiekviena profesija formuoja tam tikrą mąstymo būdą. Pavyzdžiui, dauguma sisteminių analitikų mąsto algoritmiškai, o dauguma valdininkų – remiasi situacijomis. Skirtingus mąstymo įgūdžius ir skirtingą patirtį turintiems žmonėms bendraujant trumpą laiką, susidaro įspūdis, kad partneris yra nekompetentingas. Analitikas teigia, kad užsakovas nežino, ko jam iš tikrųjų reikia ir apskritai nesugeba formuluoti reikalavimų, o užsakovas aiškina, kad analitikas yra visiškas neišmanėlis, nepajėgia suvokti paprasčiausių, „visiems žinomų" dalykų.

2.2. Analizės ir projektavimo metodų, priemonių parinkimas

Srities reikalavimų analizė ir sistemos projektavimas atliekamas naudojant *RUP* metodą, kuris remiasi UML.

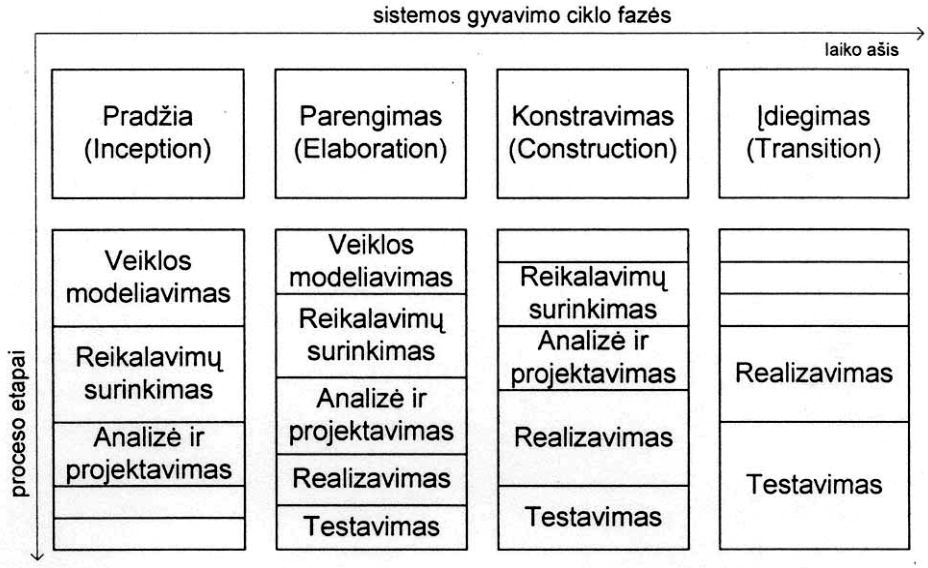
UML suteikia daug daugiau galimybių nei Oracle CASE metodas [2], kuris neužtikrina vartotojo reikalavimų pilnos specifikacijos sudarymo. Pirmiausia ši specifikavimo kalba pasižymi daug didesniu išraiškingumu bei notacijos įvairove, tuo užtikrindama galimybes aprašyti įvairiausias dalykinės srities (DS) charakteristikas bei vartotojo reikalavimų niuansus. UML naudojamas klasių modelis suorientuoja specifikaciją objektiniu aspektu, tuo tarpu Oracle CASE priemonėse naudojamas klasių modelis nėra visiškas UML klasių modelio atitikmuo, nes jame trūksta pilnos klasių paveldėjimo realizacijos. UML kalbos priemonėmis galima adekvačiai specifikuoti didžiąją dalį dalykinės srities semantinių aspektų.

Tačiau naudojant UML, vartotojo funkcinius reikalavimus betarpiškai tenkinantys rezultatai – formos, ataskaitos, meniu, kurie dažniausiai išreiškiami per sąsajos elementus, nėra pilnai specifikuojami, o specifikacijos forma neadekvati vartotojo turimam jų įvaizdžiui. Būtent šiems reikalavimams specifikuoti yra paranku naudoti Oracle CASE [2] priemones, nes jos turi tam reikiamas priemones – modulių diagramas.

UML notacija realizuota viename iš labiausiai paplitusių ir palaikomų programinės įrangos kūrimo metodų – vieningo *Rational* proceso (RUP).

RUP aprašomas dviejose dimensijose: laiko (išskiriamos gyvavimo ciklo fazės) ir proceso komponenčių (kiekvienoje fazėje išskiriami proceso etapai). PĮ gyvavimo ciklas (2 pav.) susideda iš 4 nuoseklių fazių: pradžios, parengimo, konstravimo, įdiegimo. Kiekvienoje fazėje gali būti atliktos kelios iteracijos.

Pradžios fazėje nustatoma, kokią organizacijos veiklos dalį turi palaikyti kuriama IS. Tai atliekama identifikavus išorines esybes (aktorius, kurie bendrauja), ir šio bendravimo prigimtį, t. y. identifikuojami visi sistemos panaudojimo atvejai.



2 pav. RUP fazės ir etapai

Paruošimo fazės tikslas – išanalizuoti dalykinę sritį, parinkti pagrindinę architektūrą, sudaryti projekto planą ir pašalinti rizikingiausius elementus iš projekto.

Konstravimo fazėje palaipsniui sukuriamas visas produktas, kuris gali būti įdiegtas.

Diegimo fazėje produktas įdiegiamas. Čia nusprendžiama, ar produktas yra tinkamas, ar reikia pradėti naują jo gyvavimo ciklą.

Kiekviena gyvavimo ciklo fazė susideda iš kelių etapų: veiklos modeliavimo, reikalavimų surinkimo, analizės ir projektavimo, realizavimo, testavimo.

RUP funkciniams reikalavimams specifikuoti naudoja Ivaro Jacobsono sugalvotus panaudojimo atvejus (angl. *use cases*). Taikant RUP metodą, sistemos reikalavimų specifikacija susideda iš dviejų dalių: panaudojimo atvejų modelio ir papildomų specifikacijų (angl. *supplementary specification*). Panaudojimo atvejai specifikuoja dinaminę funkcinių reikalavimų dalį, o papildomos specifikacijos aprašo likusius funkcinius ir nefunkcinius reikalavimus.

Rational Rose paketas leidžia braižyti tokias UML diagramas:

Panaudojimo atvejų diagrama (angl. *Use-Case*). Į ją perkeliamos sistemos teikiamos paslaugos kartu su aktoriais, kurie šiomis paslaugomis naudosis. Dalis reikalavimų perimama iš

sistemos tikslų aprašo. Sistemos tikslai yra tarsi paslaugas verifikuojantys reikalavimai, t. y. parodo pagrindinius procesus, kuriuos turėtų kompiuterizuoti sistema. Nefunkciniai veikimo reikalavimai nurodo išorinius sistemos aktorius: partnerines ir bendradarbiavimo sistemas. Veikimo reikalavimai taip pat padeda iš dalies nustatyti panaudojimo atvejų diagramų ribas.

Darbų sekų diagrama (angl. *Sequence*). Ji detalizuoja panaudojimo atvejus, todėl šis modelis aprašo sistemos atliekamų darbų eigą, sąveiką tarp objektų. Šios diagramos parodo darbų eigos, sąveikos tarp objektų tęstinumą laiko atžvilgiu, nusako, kada ir kokie veiksmai turi būti atliekami. Panaudojant pseudokodą, galima įvesti sąlyginę darbų atlikimo seką, tai leidžia specifikuoti tokius reikalavimus:

- sistemos elgsena;
- reakcija į įėjimus;
- ko sistema neturėtų daryti.

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

Klasių diagrama (angl. *Class*). Joje specifikuojamas sistemos duomenų modelis, panašus į Oracle ER modelį. Pagrindiniai reikalavimai šiam modeliui gaunami su įėjimo ir išėjimo srautų sudėtimi, kuriuos prieš tai turi apdoroti sistemos analitikas. Nedidelė dalis reakcijos į įėjimus reikalavimų, apdorotų analitiko, taip pat persikelia į duomenų modelį. Diagramoje modeliuojamos objektų klasės, jų atributai, su jomis atliekamos operacijos ir ryšiai tarp klasių.

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

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

Veiklos diagrama (angl. *Activity*). Ji skirta atvaizduoti elgsenai sistemos viduje. Nors ši diagrama gali būti panaudojama modeliuojant visos organizacijos veiklą. Esant didelei sistemai, jos veiklą patogu nagrinėti skaidant ją pagal panaudojimo atvejų diagramas. Veiklos diagramose specifikuojami reikalavimai iš sistemos tikslų, sistemos teikiamos paslaugos. Įėjimo ir išėjimo srautai šioje diagramoje specifikuojami siekiant parodyti juos veiklos kontekste.

Paskirstymo diagrama. Ji labiau skirta projektavimui, nei poreikiams specifikuoti. Tačiau analizės dalyje galima panaudoti veikimo nefunkciniams reikalavimams specifikuoti. Būtent šie reikalavimai nusako veikimo principus ir sąsajas su išorinėmis sistemomis bei resursų pasiskirstymą tarp aparatūrinės įrangos.

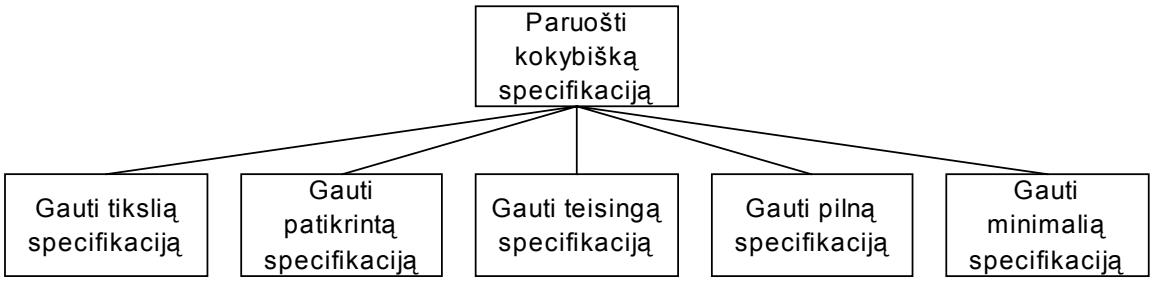
Stiprioji *Rational Unified Process* metodo pusė yra jo inžinerinė realizacija. *Rational* kompanijos pateikiamas CASE priemonių rinkinys leidžia automatizuoti nemažai RUP veiklų, reikalavimų inžinerijai palengvinti naudojamos šio paketo programos.

RUP yra sudėtingas metodas ir jo išmokimo kreivė yra pakankamai gulsčia (investuotas laikas atsiperka negreitai). Panaudojimo atvejų modelio sudarymas yra pakankamai sudėtingas, nes pati panaudojimo atvejų idėja, iš pradžių atrodanti paprasta, reikalauja daug mokymosi, norint ją sėkmingai pritaikyti.

2.3. Reikalavimų inžinieriaus veiklos analizė

2.3.1. Reikalavimų inžinieriaus veiklos tikslų modelis

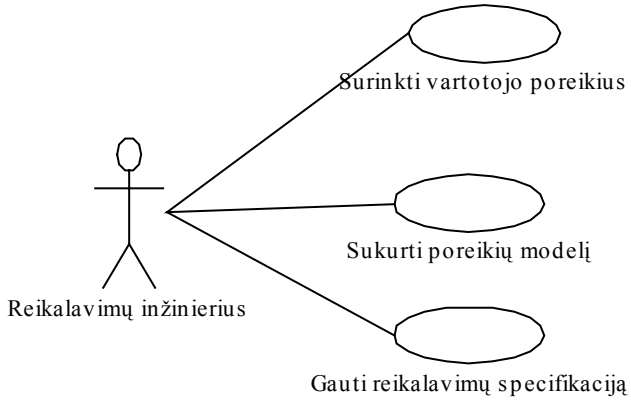
Pagrindinis reikalavimų inžinieriaus veiklos tikslas yra paruošti kokybišką reikalavimų specifikaciją. 3 pav. esančiame reikalavimų inžinieriaus veiklos tikslų modelyje šis tikslas detalizuojamas žemesnio lygio reikalavimais.



3 pav. Reikalavimų inžinieriaus veiklos tikslų modelis

2.3.2. Reikalavimų inžinieriaus veiklos panaudojimo atvejai

Pagrindiniai reikalavimų inžinieriaus veiklos panaudojimo atvejai yra „surinkti vartotojo poreikius", „sukurti poreikių modelį" ir „gauti reikalavimų specifikaciją " (4 pav.).



4 pav. Reikalavimų inžinieriaus veiklos panaudojimo atvejai

2.4. Pasaulio ir Lietuvos literatūros šaltiniuose pateiktų sprendimų problemai spręsti lyginamoji analizė

Nagrinėsime tokias duomenų bazių valdymo sistemas (DBVS), kurios savo automatizuoto projektavimo priemonėmis leidžia sukurti IS prototipus.

Oracle Designer [2] sistemos prototipo kūrimas atliekamas tokia tvarka. Sukuriamas dalykinės srities modelis: specifikuojamos funkcijos ir esybės bei ryšiai tarp jų. Esybių atributus susiejame su funkcijomis. Pasinaudojus sudaryta specifikacija apie dalykinę sritį (DS), paleidžiamos automatizuoto generavimo priemonės. Jos sukuria ekraninių formų, ataskaitų projektus. Vykdant generavimą sukuriamos veikiančios ekraninės formos. Norint sukurti ekraninę formą su Oracle Designer, reikia suprojektuoti duomenų bazę. Tai darant su Visual FoxPro ar MS Access taip pat turi būti suprojektuotos bent jau duomenų bazės lentelės (1 lentelė).

1 lentelė

CASE įrankių ir DBVS, kuriančių kompiuterizuotos IS prototipą, palyginimas

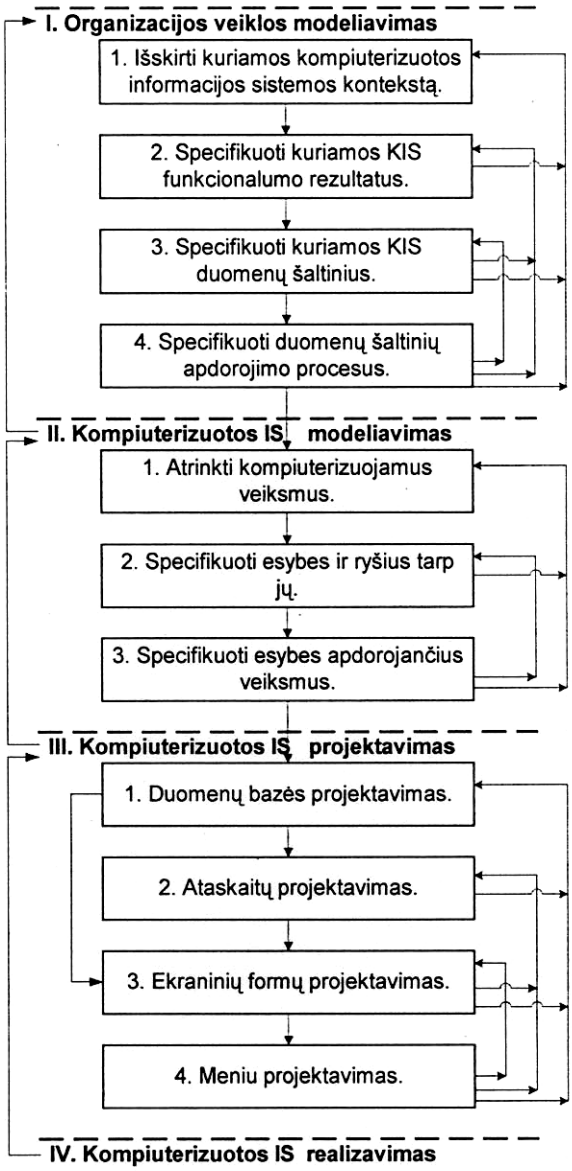
	Oracle Designer (automatizuoto projektavimo įrankis)	**Visual FoxPro (DBVS)**	**MS Access (DBVS)**
Ką reikia turėti norint sugeneruoti ekraninę formą (prototipą)	Specifikuojamos funkcijos ir esybės. Funkcijos susiejamos su esybėmis, nurodant kaip su jomis elgiamasi. Analogiškai ir esybių naudojamus atributus susiejame su funkcijomis. Iš funkcijų gaunami	Reikia turėti suprojektuotą DB. Turime deklaruoti lenteles ir susieti jas ryšiais. Tai leis generuoti prototipus, bet ekr. formas turėsime pataisyti taip, kad atitiktų veiklos taisykles.	Reikia turėti suprojektuotą DB. Turime deklaruoti lenteles ir susieti jas ryšiais. Tai leis generuoti prototipus, bet ekr. formas turėsime pataisyti taip, kad atitiktų veiklos taisykles.

	programinių modulių projektai, o iš esybių – DB projektas. Tada galima generuoti veikiantį ekraninės formos prototipą.		
Prototipo funkcionalumas	Prototipas bus jau beveik veikianti sistema su tam tikrais neišbaigtumais, bet jį galime sukurti tik atlikę sistemos projektavimą.	Prototipas leis įvedinėti duomenis į DB, bet be garantijų, kad tai atitiks sistemos veiklos taisykles.	Prototipas leis įvedinėti duomenis į DB, bet be garantijų, kad tai atitiks sistemos veiklos taisykles.

2.4.1. IS keliamų funkcinių reikalavimų specifikavimo metodas, išskiriant kompiuterizuojamus veiksmus

Sumažinti atotrūkį tarp vartotojo (reikalavimų teikėjo) ir projektuotojo nėra paprastas uždavinys. Tuo tikslu turi būti sudaroma vartotojo reikalavimų specifikacija, reikalavimus atvaizduojant notacijoje, kurią be papildomo apmokymo gali suprasti ir vartotojas. Tokio tikslo laikomasi IS keliamų funkcinių reikalavimų specifikavimo metode [1].

Taikant informacijos sistemai keliamų funkcinių reikalavimų specifikavimo metodą, KIS kūrimo procesas yra suskirstomas į 4 fazes (5 pav.):



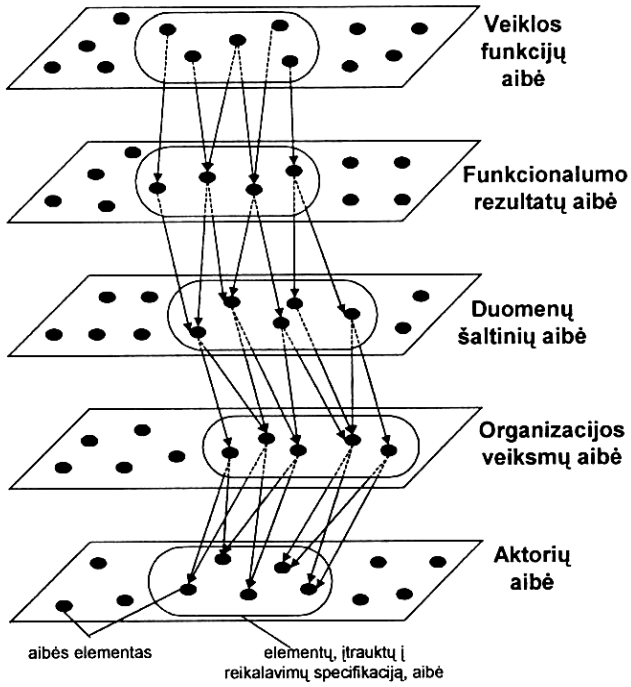
5 pav. IS kūrimo procesas, taikant IS keliamų funkcinių reikalavimų specifikavimo metodą

1. Organizacijos kompiuterizuojamos veiklos modeliavimas.
2. Kompiuterizuotos IS modeliavimas.
3. Kompiuterizuotos IS projektavimas.
4. Sistemos realizavimas.

Pirmoji metodo dalis apima organizacijos kompiuterizuojamos veiklos modeliavimą, nes pagrindiniai funkciniai reikalavimai kuriamai IS specifikuojami šioje fazėje. Organizacijos kompiuterizuotinos veiklos modeliavimas skirstomas į 4 etapus [1]:

1. Išskirti kuriamos kompiuterizuotos informacijos sistemos kontekstą.
2. Specifikuoti kuriamos KIS funkcionalumo rezultatus.
3. Specifikuoti kuriamos KIS duomenų šaltinius.
4. Specifikuoti duomenų šaltinių apdorojimo procesus.

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



6 pav. Principinės metodo idėjos iliustracija

Pagrindinė reikalavimų specifikavimo idėja yra tokia (6 pav.):

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

Pateiktas modelis IS naudojamų duomenų srautų struktūrai specifikuoti. Šis modelis sudarytas iš 2 lygių:

1. Į vartotoją orientuotas lygis.
2. Į analitiką orientuotas lygis.

Duomenų šaltinių apdorojimo etapams, ryšiams bei perėjimams tarp duomenų šaltinių būsenų specifikuoti panaudota R. Gusto pasiūlyta komunikacinių veiksmų ir perėjimų tarp objektų būsenų grafinė notacija bei modeliavimo principai.

Bendrai vertinant IS keliamų funkcinių reikalavimų specifikavimo metodą, jį galima įvardinti kaip metodą, palengvinantį analitiko darbą ir išgaunant reikalavimus iš vartotojo, ir atliekant jų specifikavimą. Nuosekli ir natūralizuota metodo etapų atlikimo technologija padidina sudaromos specifikacijos kokybę, padeda geriau patikrinti, ar specifikacija pilna, nes modelį nesunku suprasti ir vartotojui, tačiau sulėtina visos analizės tempą.

Nagrinėtas funkcinių reikalavimų, keliamų IS, specifikavimo metodas pasižymi tokiomis savybėmis [1]:

1. Leidžia formalizuotai (grafiškai) specifikuoti reikalavimus KIS išvedamos informacijos struktūrai anksčiau, nei specifikuojamas dalykinės srities duomenų modelis. Tokios galimybės neturi nė vienas žinomų metodų. Ši savybė reikalavimų specifikavimo eigą daro natūralią.
2. Reikalavimai specifikuojami sklandžiai pereinant nuo vieno modelio prie kito. Tokiu būdu išvengiama chaotiško specifikavimo, suartinamas specifikavimo procesas su natūralia reikalavimų analizės eiga, diktuojama iteratyvaus vartotojo ir analitiko vienas kito mokymo. Kiekvienas kitas modelis formuojamas ankstesnių modelių pagrindu. Gaunama vientisa specifikacija, kurioje išvestiniai elementai yra motyvuojami anksčiau įvestais ar išvestais elementais. Kituose metoduose vientisą specifikaciją įmanoma gauti, tačiau sklandus perėjimas tarp modelių nepalaikomas.
3. Informacijos struktūrai specifikuoti naudojamas šiam tikslui sukurtas dviejų lygių (vienas lygis yra skirtas vartotojui, kitas – analitikui) modelis, leidžiantis neformaliai specifikuotus reikalavimus informacijos struktūrai susieti su formalizuotais struktūros elementais. Tokiu būdu pagrindžiamas duomenų struktūros elementų atsiradimas specifikacijoje. Be to, sumažinamas tarp vartotojo ir analitiko esantis supratimo atotrūkis. Nagrinėtuose metoduose duomenų struktūrą atitinkantys esybių-ryšių (naudojamas *Oracle* CASE metode), klasių (RUP metode) ar konceptų modeliai yra pirminiai, sudaromi nebūtinai pagal konkrečią metodiką, bet išanalizavus gautą informaciją apie dalykinę sritį.
4. Informacijos struktūros (DŠ struktūros) modelis pritaikytas reikalavimams, keliamiems KIS dinamikai, nustatyti ir specifikuoti. Tokiu būdu specifikuojant vartotojo reikalavimus, sistemos modelyje neatsiejama dinamika nuo statikos, tuo tarpu naudojant kitų metodų specifikavimo priemones šie aspektai visiškai arba iš dalies atskiriami.
5. Naudojamos formalizuotos priemonės, leidžiančios kontroliuoti specifikacijos pakankamą išsamumą, vidinį darnumą bei minimalumą. Šiam tikslui pritaikyti ir išplėtoti R. Gusto pasiūlyti semantinių diagramų kokybės analizės ir įvertinimo principai. Kiti nagrinėti metodai formalizuotų priemonių patikrinti, ar sudaryta specifikacija yra

pakankamai išsami bei minimali, organizacijos veiklos modeliavimo fazėje neturi. Viena iš priežasčių – nenatūrali specifikacijos sudarymo eiga.

6. Metodas ištirtas konceptualaus pavyzdžio pagrindu bei išbandytas eksperimentine CASE įrankio realizacija.

2.5. Projekto tikslas ir jo pagrindimas, kokybės kriterijų apibrėžimas

Šio projekto tikslas – sukurti programinę įrangą, kuri leistų sukurti informacinės sistemos prototipą iš reikalavimų specifikacijos, sudarytos remiantis Informacijos sistemų katedroje pasiūlytu reikalavimų specifikavimo metodu.

Sukurtoji programinė įranga turės būti Informacijos sistemų katedroje kuriamo CASE įrankio, skirto reikalavimų specifikacijai sudaryti, dalimi. Ji suteiks papildomą funkcionalumą prie šio įrankio.

Pagrindinis projekto kokybės kriterijus – sukurtos programinės įrangos funkcionalumas. Pagrindinė sukurtoji programinės įrangos funkcija yra – sukurti IS prototipą iš duomenų bazėje pasirinktos specifikacijos. Duomenų bazėje saugoma tik vienos sistemos reikalavimų specifikacija. Programa ją turi išanalizuoti, sugeneruoti programinį kodą, kurį bus galima įvykdyti, saugoti jo versijas, šalinti. Turės būti galimybė sukurti tik tam tikros IS dalies prototipą, t. y., tik tam tikrą ekraninę formą ar ataskaitą.

Kitas kokybės kriterijus galėtų būti – sukurtos sistemos tinkamumas reikalavimų specifikacijai, sudarytai Informacijos sistemų katedroje kuriamu CASE įrankiu, validuoti.

Taip pat turi būti patikrinta, ar sukurtoji programinė įranga:

- integruojasi į Informacijos sistemų katedroje kuriamą CASE paketą,
- yra išbaigta, t. y., ar atitinka visus jai iškeltus funkcinius reikalavimus,
- yra tolerantiška klaidoms,
- yra perkeliama, t. y., ar ją galima perkelti kartu su reikalavimų specifikacijos duomenų baze į bet kurią *Microsoft* aplinką.

2.6. Kompiuterizuojamos sistemos varianto parinkimas

Pagrindinis kompiuterizuojamas uždavinys – IS prototipo automatizuotas kūrimas.

2.7. Analizės išvados

1. Išanalizuotos kompiuterizuotos IS prototipo kūrimo galimybės su automatizuoto projektavimo įrankiu *Oracle Designer* ir DBVS *Visual FoxPro* bei *MS Access*.

2. Išanalizuotas reikalavimų inžinerijos procesas.

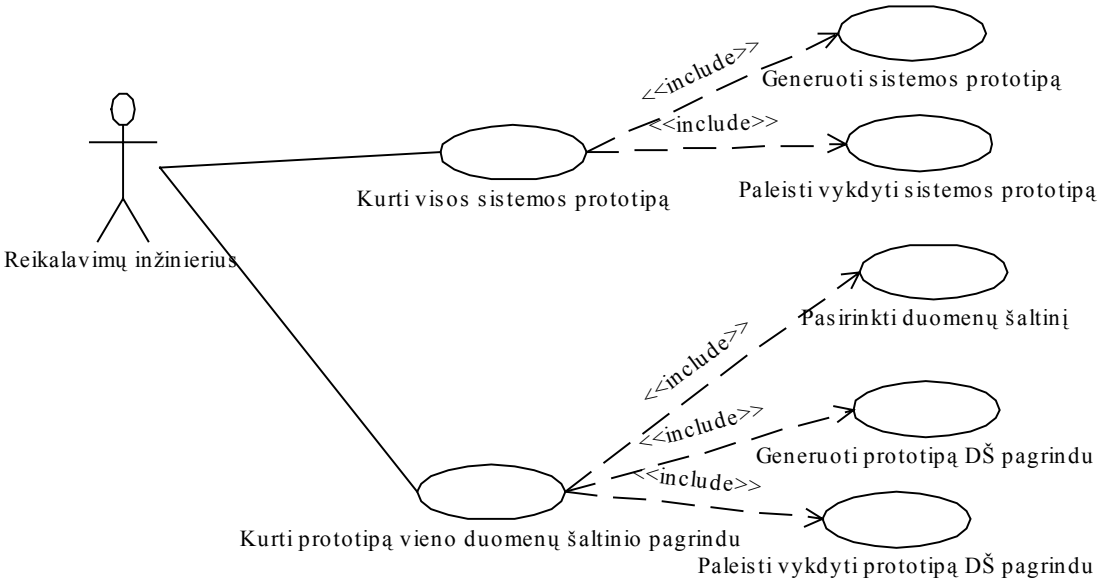
3. *Oracle Designer* atveju tam, kad sukurti kompiuterizuotos IS prototipą, reikia praeiti visą projektavimo etapą. DBVS Visual FoxPro ir MS Access atveju reikia turėti suprojektuotą DB. Tai leis generuoti IS prototipus, bet veiklos taisyklių nebus paisoma. Todėl reikia kurti naują sistema, kuri leistų ankstyvoje kūrimo stadijoje sukurti sistemos prototipą, nežiūrint į tai, kad neturime sistemos projekto.

3. Projekto dalis

3.1. Reikalavimų modelis

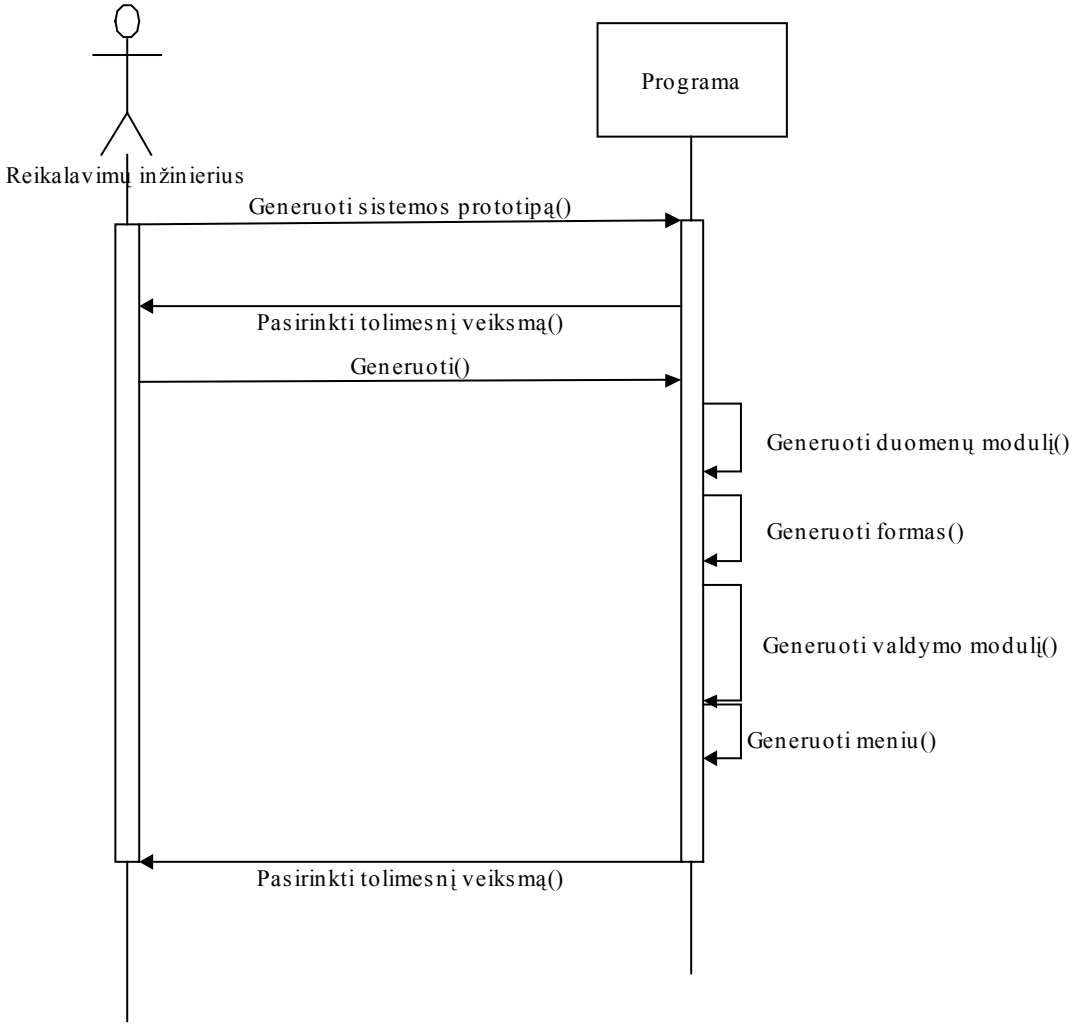
3.1.1. Vartotojų panaudojimo atvejų modelis

Vartotojų panaudojimo atvejų modelio pagrindiniai panaudojimo atvejai yra ,,kurti visos sistemos prototipą" ir ,,kurti prototipą vieno duomenų šaltinio pagrindu". Šie panaudojimo atvejai detalizuojami 7 pav. Panaudojimo atvejis ,,kurti visos sistemos prototipą" susideda iš panaudojimo atvejų ,,generuoti sistemos prototipą" ir ,,paleisti vykdyti sistemos prototipą". O panaudojimo atvejis ,,kurti prototipą vieno duomenų šaltinio pagrindu" susideda iš panaudojimo atvejų ,,pasirinkti duomenų šaltinį", ,,generuoti prototipą DŠ pagrindu" ir ,,paleisti vykdyti prototipą DŠ pagrindu".

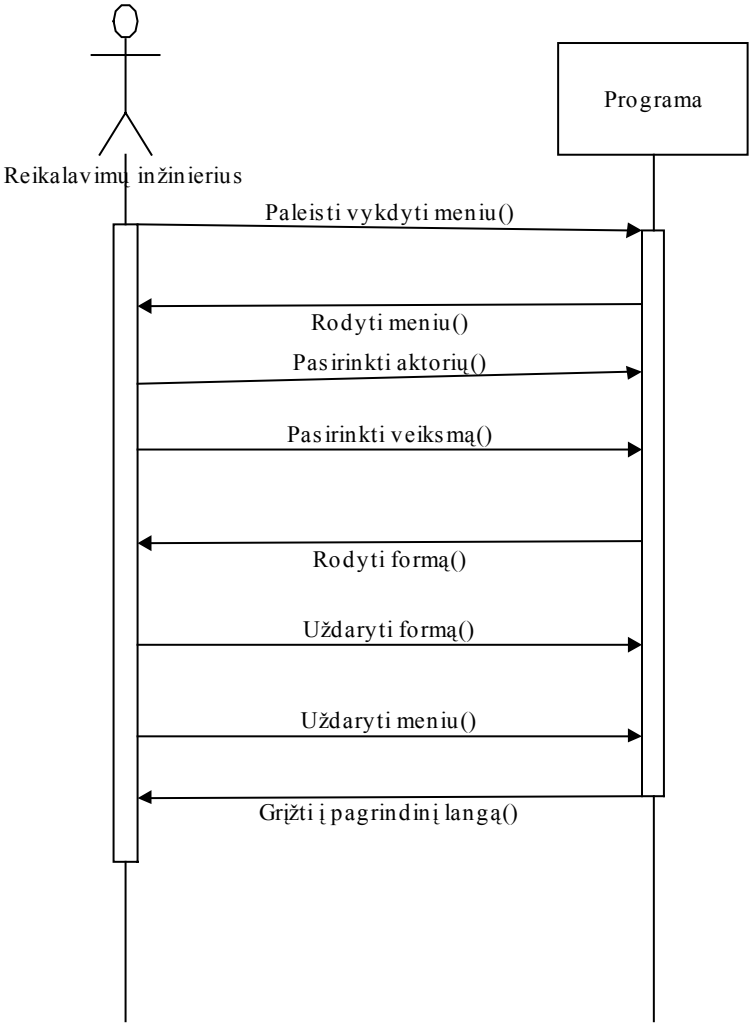


7 pav. Vartotojų panaudojimo atvejų modelis

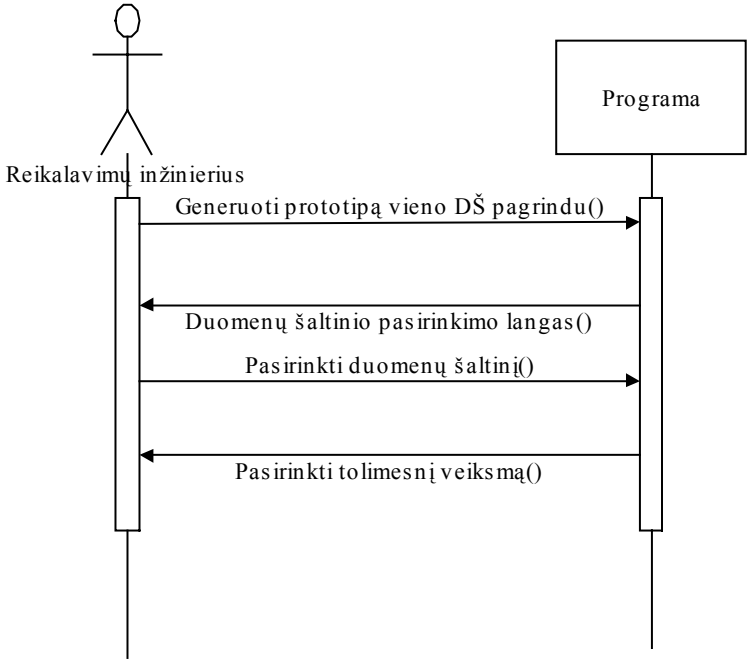
Kiekvienas vartotojų panaudojimų atvejis yra specifikuojamas vartotojo ir sistemos sekų diagrama (8-12 pav.).



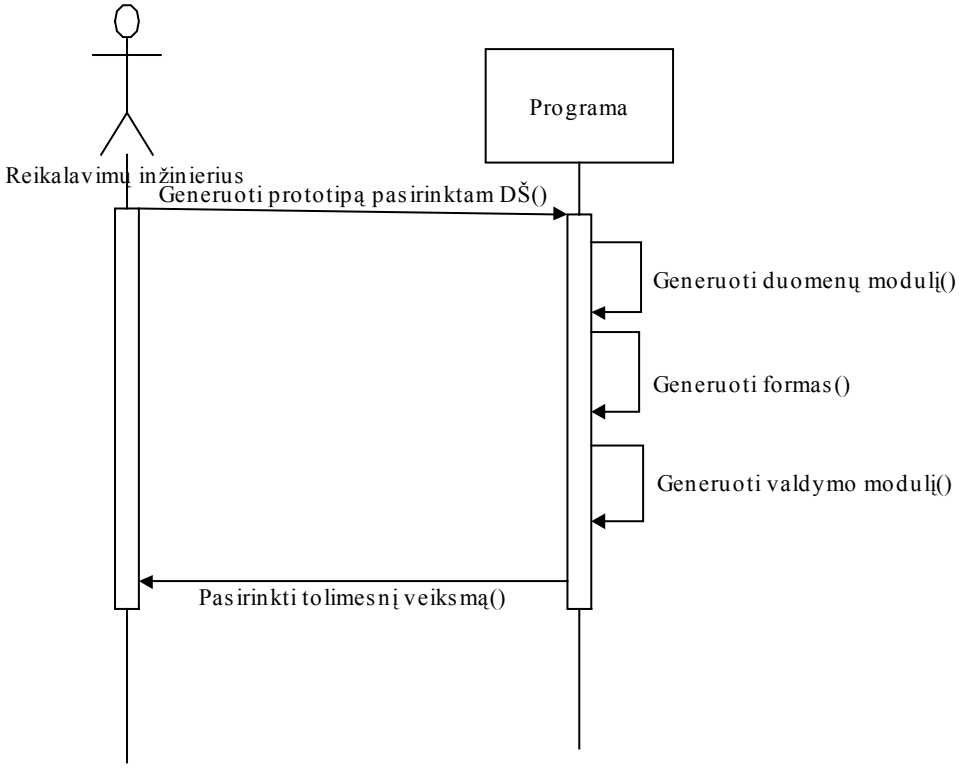
8 pav. Sekų diagrama panaudojimo atvejui ,,generuoti sistemos prototipą”



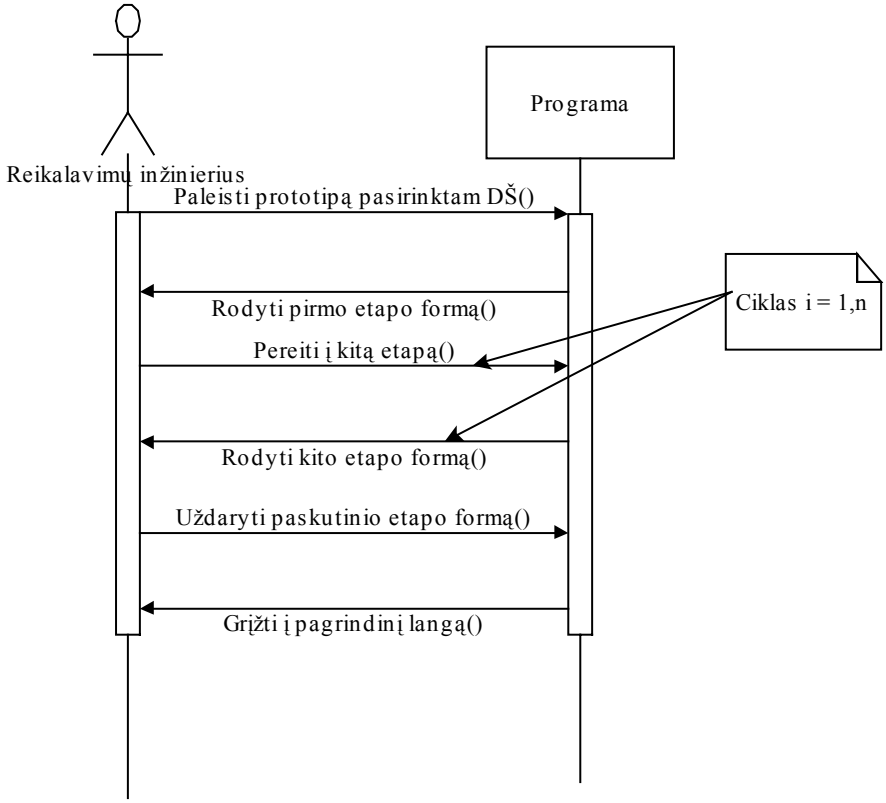
9 pav. Sekų diagrama panaudojimo atvejui „paleisti vykdyti sistemos prototipą”



10 pav. Sekų diagrama panaudojimo atvejui „pasirinkti duomenų šaltinį”



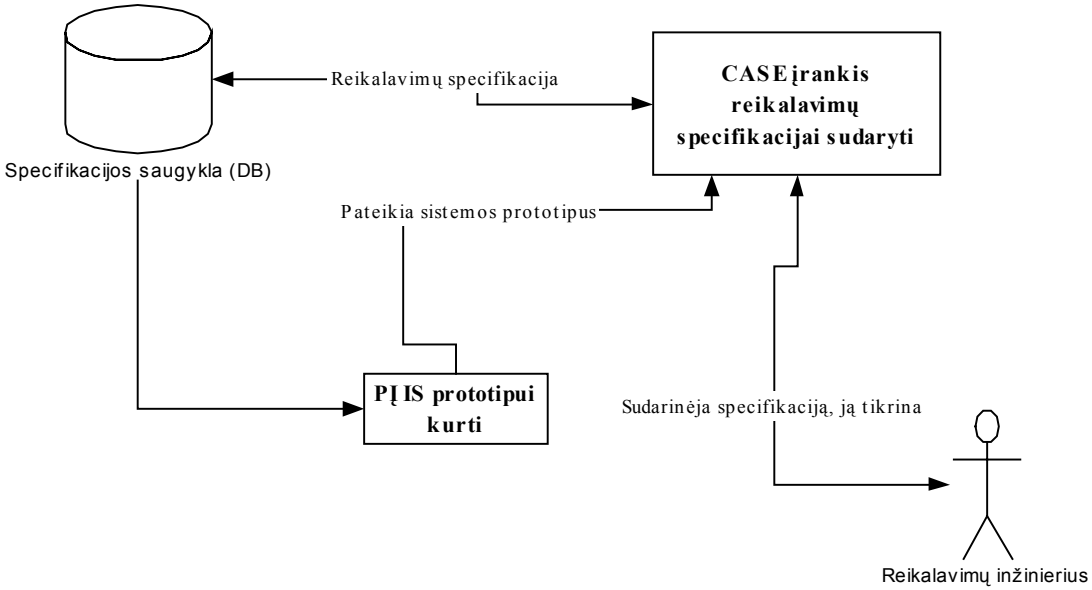
11 pav. Sekų diagrama panaudojimo atvejui ,,generuoti prototipą DŠ pagrindu”



12 pav. Sekų diagrama panaudojimo atvejui ,,paleisti vykdyti prototipą DŠ pagrindu”:

3.1.2. Sistemos kontekstinė diagrama

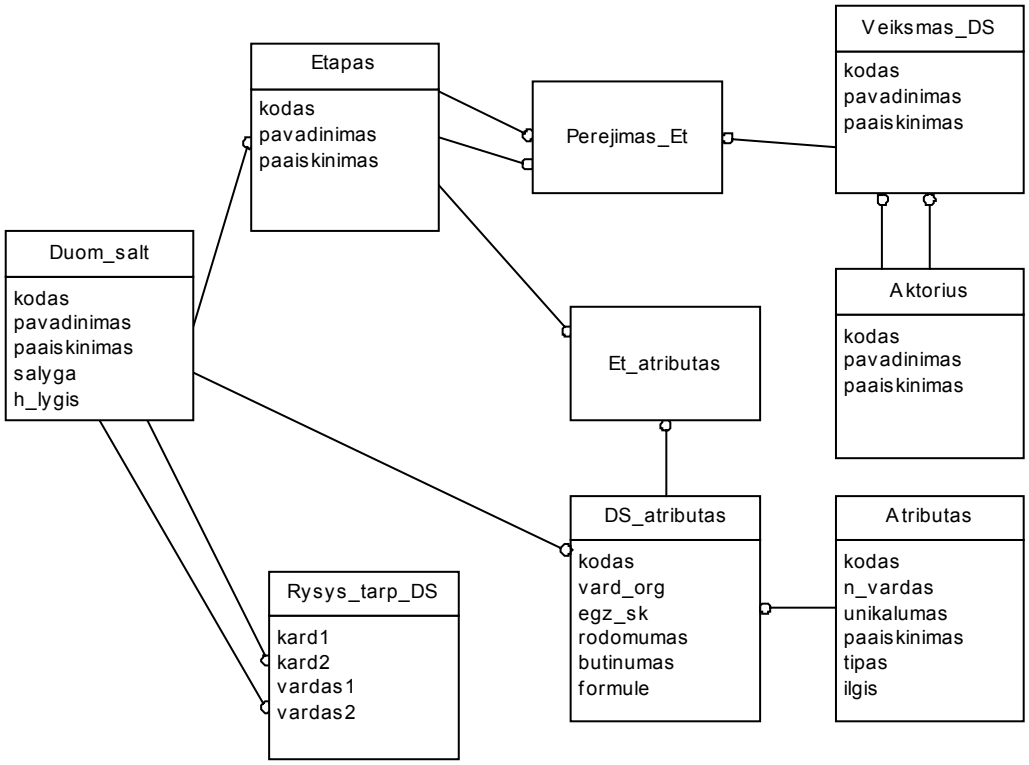
Sistemos kontekstinėje diagramoje vaizduojama sąveika tarp kuriamos PĮ IS prototipui kurti, specifikacijos saugyklos, CASE įrankio reikalavimų specifikacijai sudaryti ir reikalavimų inžinieriaus (13 pav.).



13 pav. Sistemos kontekstinė diagrama

3.1.3. Dalykinės srities klasių diagrama

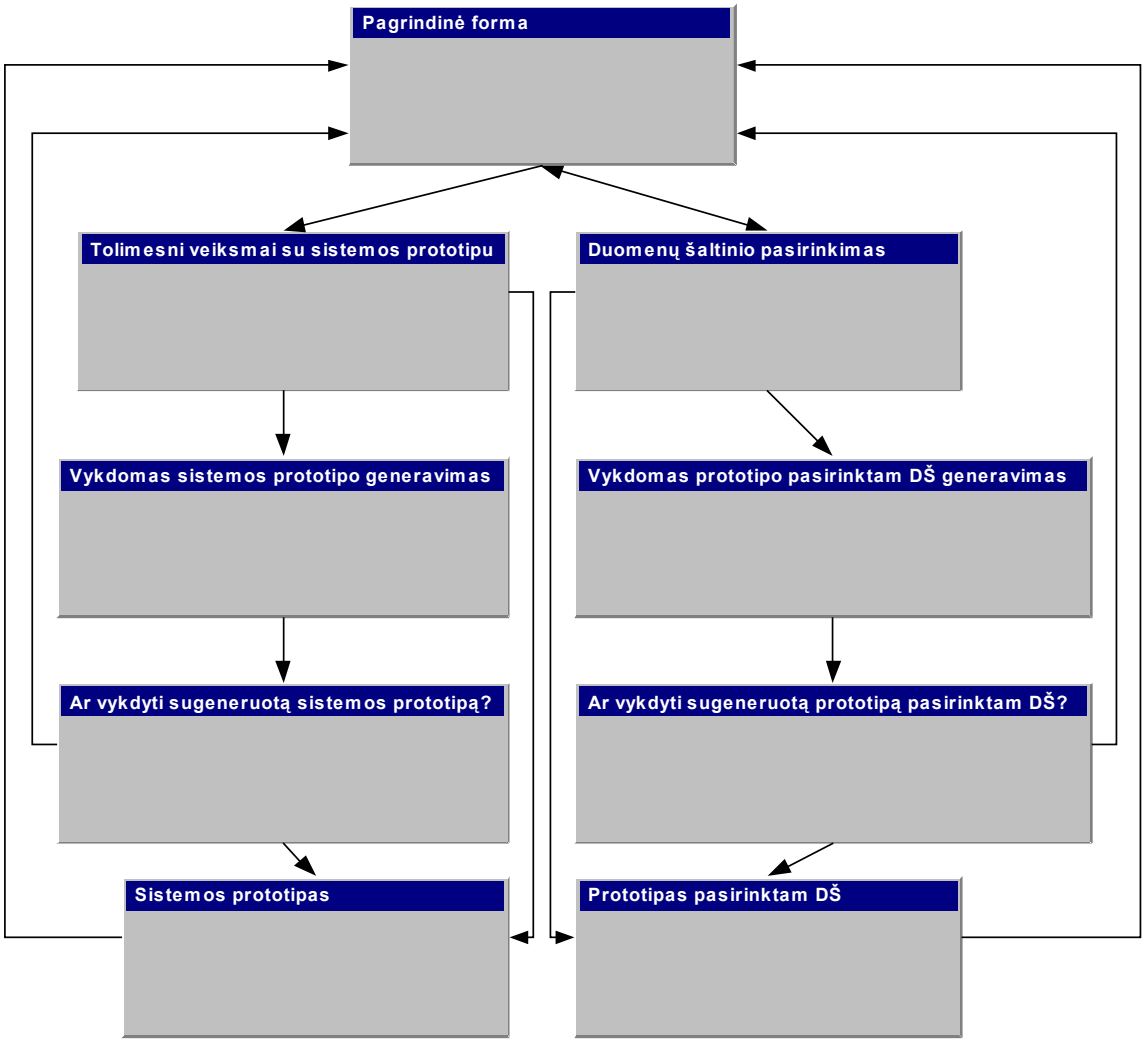
14 pav. vaizduojama DS klasių diagrama.



14 pav. Dalykinės srities klasių diagrama

3.1.4. Vartotojų interfeiso modelis

Vartotojų interfeiso modelyje matome kokia yra galima navigacija tarp programos ekraninių komponentų (15 pav.).

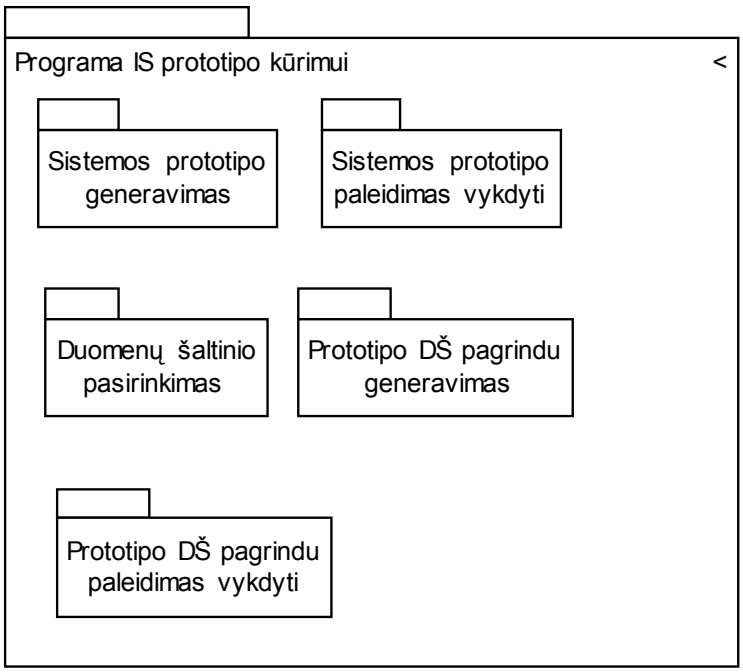


15 pav. Vartotojų interfeiso modelis

3.2. Sistemos projektas

3.2.1. Sistemos architektūra

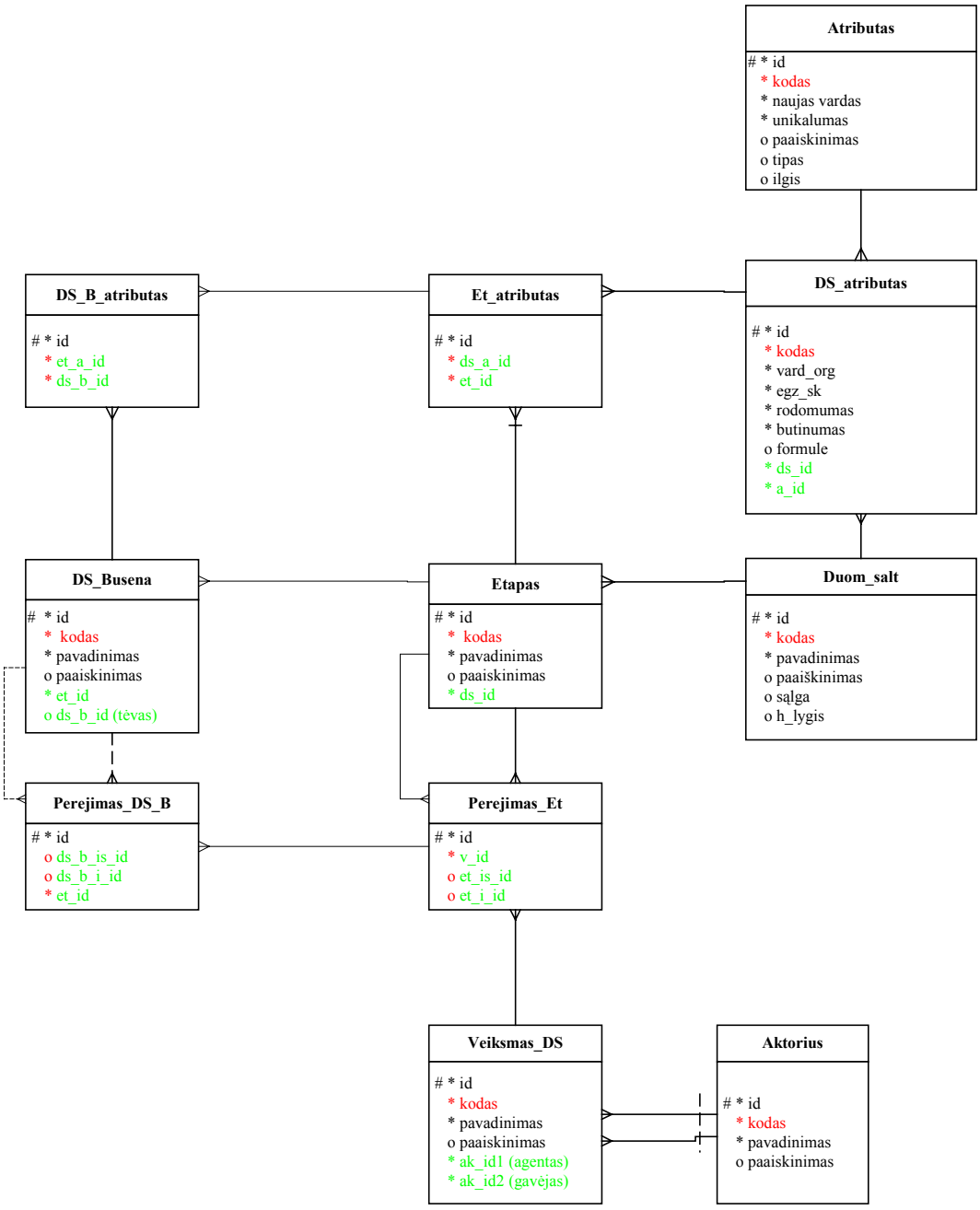
Sistemos architektūrą vaizduoja paketų diagrama (16 pav.):



16 pav. Sistemos architektūros diagrama

3.2.2. Duomenų bazės loginė schema

Kuriamai programinei įrangai reikalingos šios specifikacijos saugyklos (DB) lentelės (17 pav.):



17 pav. Duomenų bazės loginė schema

Duomenų bazės objektų aprašymas

DB objektų aprašymą matome (2 lentelėje).

Duomenų tipų paaiškinimai:

- **int** – sveikas skaičius;
- **varchar <ilgis>** - nustatyto ilgio simbolių eilutė, kur <ilgis> - eilutės ilgis;

- **bool** – loginis kintamasis įgyjantis reikšmes *true* arba *false*.

2 lentelė

Vardas	Duomenų tipas	Pirminis raktas	Būtinas	Unikalus	Paaiškinimas
Atributas					Lentelėje saugomi visi nagrinėjamo organizacijos veiklos konteksto atributai.
Id	**Int**	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
kodas	varchar 10		+	+	Unikalus atributą apibudinantis kodas, kodą sudaro sistemos vartotojas (analitikas/projektuotojas).
n_vardas	varchar 50		+		Atributo vardas, kuris gali būti paimtas iš nagrinėjamo veiklos konteksto arba sukurtas paties vartotojo.
unikalumas	Bool		+		Požymis rodantis ar atributo reikšmė turi būti unikali. TRUE – reikšmė turi būti unikali, o FALSE - kad reikšmės unikalumas nėra būtinas.
paaiskinimas	varchar 200				Detali informacija apie atributą.
Tipas	varchar 30				Atributo duomenų tipas.
e_id	Int				Esybės, kuriai priklauso atributas id.
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.

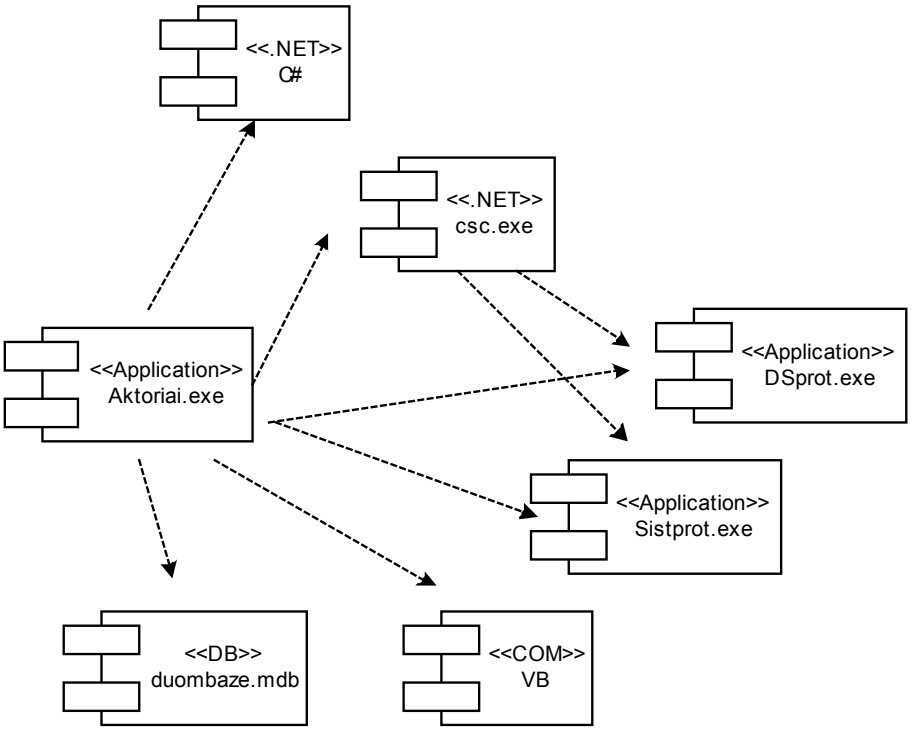
Id	Int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
Kodas	varchar 10		+	+	Unikalus įrašą apibudinantis kodas, kodą sudaro sistemos vartotojas.
vard_org	varchar 50		+		Duomenų šaltinio atributo pavadinimas, kuris paimtas iš nagrinėjamo veiklos konteksto.
egz_sk	Int		+		Konkretaus atributo egzempliorių 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.
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.
Rysys_tarp_DS					Lentelėje saugoma informacija apie ryšius tarp duomenų šaltinių.
Id	Int	+	+	+	Unikalus įrašo id, kuri sugeneruoja DBVS.
kard1	varchar 3		+		Ryšio kardinalumas viename ryšio gale. Kardinalumo aprašymo struktūra:

					<būtinumas>,<kardinalumas>. Būtinumas gali įgyti šias reikšmes: 0-nebūtinas, 1-būtinas. Kardinalumas gali įgyti šias reikšmes: 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.
Vardas1	varchar 30		+		Vieno ryšio galo vardas, kuris gali būti paimtas iš nagrinėjamo veiklos konteksto arba sukurtas paties vartotojo.
Vardas2	varchar 30		+		Kito ryšio galo vardas, kuris gali būti paimtas iš nagrinėjamo veiklos konteksto arba sukurtas paties vartotojo.
L_ds_r_id1	int				Lanko tarp duomenų šaltinių ryšių id, kuriam priklauso pirmas ryšio galas.
L_ds_r_id2	int				Lanko tarp duomenų šaltinių 100 ryšių id, kuriam priklauso antras ryšio galas.
ds_id1	int		+		Duomenų šaltinio id, iš kurio eina ryšys.
ds_id2	int		+		Duomenų šaltinio id, į kurį eina ryšys.

3.2.3. Realizacijos modelis

Sistemos realizacijos modelis vaizduojamas komponentų diagrama (18 pav.):



18 pav. Sistemos realizacijos modelis

3.2.4. Testavimo modelis bei duomenys, kontrolinis pavyzdys

Duomenų bazėje turime tokius testavimo duomenis:

Duom_salt : Table

id	kodas	pavadinimas
0	D06	VS pastatymo-keitimo aktas
1	SKM02	Uzduotis VS keisti
2	SKM01	Orderis VS priduoti i remonta ir grazinti is
0		

19 pav. Lentelė Duom_salt.

id	kodas	pavadinimas	paaiskinimas	ds_id
0	E01	Sudaryta		1
1	E02	Neatliktas keitimas		1
2	E03	Atliktas keitimas		1
3	E04	Atliktas ivertinimas		1
4	E05	Atiduotas		2
5	E06	Priimtas		2
6	E07	Perduotas		2
7	E08	Atsiimtas		2
8	E09	Grazintas		2
9	E10	Sustatytas		0
0				0

20 pav. Lentelė Etapas.

id	kodas	vard_org	egz_sk	rodomumas	butinumas	formule
0	DSA01	Nr	1	☑	☑	
1	DSA02	Pastatymo data	1	☑	☐	
2	DSA03	Abonento. Kodas	1	☑	☐	
3	DSA04	Abonento. Pavadinimas	1	☑	☐	
4	DSA05	Adresas. Gatves pavadinimas	1	☑	☐	
5	DSA06	Adresas. Pastato Nr.	1	☑	☐	
6	DSA07	Pastatytas skaitiklis. Nr	1	☑	☐	
7	DSA08	Pastatytas skaitiklis. Marke	1	☑	☐	
8	DSA09	Pastatytas skaitiklis. Diametras	1	☑	☐	
9	DSA10	Pastatytojo parodymai	1	☑	☐	
10	DSA11	Uzplombuota	1	☑	☐	
11	DSA12	Nuimtas skaitiklis. Nr	1	☑	☐	
12	DSA13	Nuimtas skaitiklis. Marke	1	☑	☐	
13	DSA14	Nuimtas skaitiklis. Diametras	1	☑	☐	
14	DSA15	Nuimtojo parodymai	1	☑	☐	
15	DSA16	Statytojas	1	☑	☐	
16	DSA17	Nr	1	☑	☑	
17	DSA18	Data	1	☑	☐	
18	DSA19	Vykdytojas1	1	☑	☐	
19	DSA20	Vykdytojas2	1	☑	☐	
20	DSA21	Keliavimo budas	1	☑	☐	
21	DSA22	Punkto Eil.Nr.	1	☑	☐	
22	DSA23	Abonento. Kodas	1	☑	☐	
23	DSA24	Abonento. Pavadinimas	1	☑	☐	
24	DSA25	Abonento. Telefonas	1	☑	☐	

21 pav. Lentelė DS_atributas.

Et_atributas : Table

id	ds_a_id	et_id
1	0	9
2	1	9
3	2	9
4	3	9
5	4	9
6	5	9
7	6	9
8	7	9
9	8	9
10	9	9
11	10	9
12	11	9
13	12	9
14	13	9
15	14	9
16	15	9
17	16	0
18	17	0
19	18	0
20	19	0
21	20	0
22	21	0

Record: 1 of 62

22 pav. Lentelė Et_atributas.

Perejimas_Et : Table

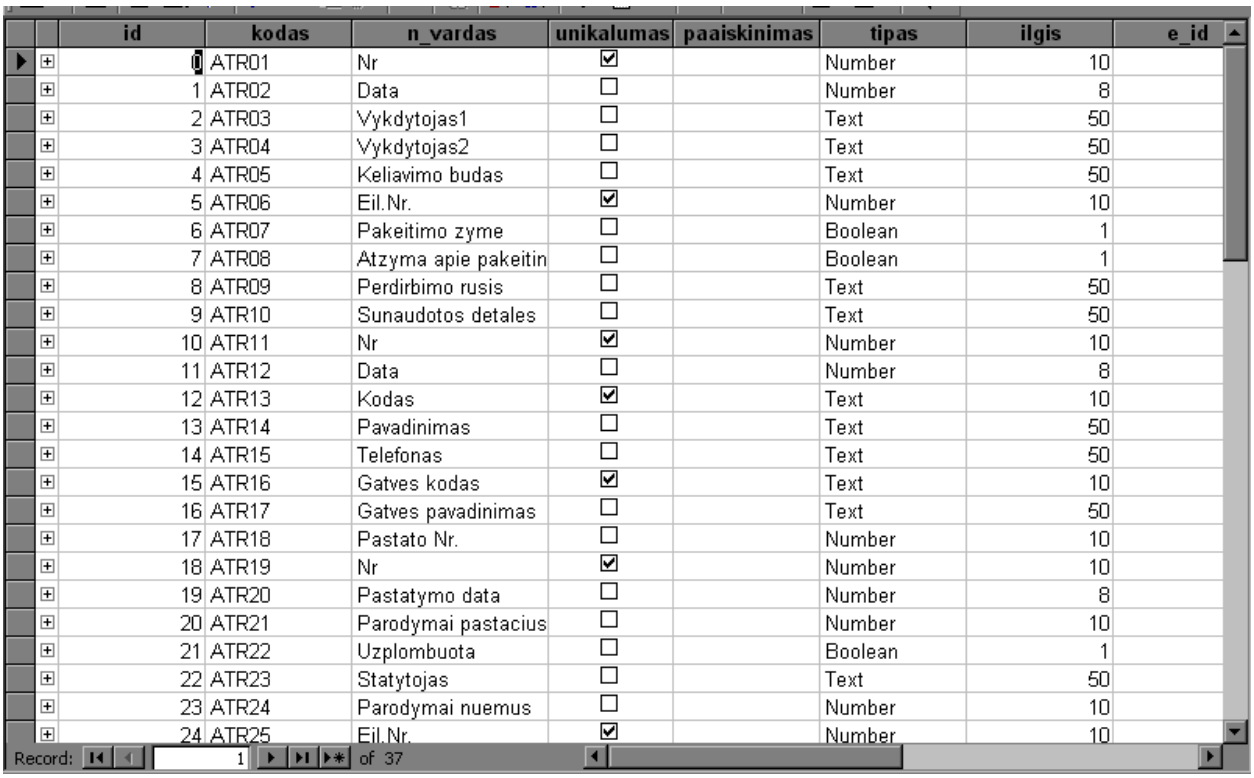
id	v_id	et_is_id	et_i_id
1	1		9
2	5		4
3	6	4	5
4	7	5	6
5	8	6	7
6	9	7	8
7	2		0
8	3	0	1
9	1	0	2
10	4	2	3
0	0	0	0

23 pav. Lentelė Perejimas_Et.

Veiksmas_DS : Table

id	kodas	pavadinimas	paaiskinimas	ak_id1	ak_id2
1	V11	Pastatyti-Keisti		3	2
2	V12	Sudaryti		2	3
3	V13	Nekeisti		3	2
4	V14	Ivertinti nuimta skaitikli		4	2
5	V15	Priduoti i remonta		2	5
6	V16	Priimti remontuoti		5	8
7	V17	Perduoti remontuoti		8	7
8	V18	Atsiimti is remonto		7	6
9	V19	Grazinti		6	2
0				0	0

24 pav. Lentelė Veiksmas_DS.



id	kodas	n_vardas	unikalumas	paaiskinimas	tipas	ilgis	e_id
0	ATR01	Nr	☑		Number	10	
1	ATR02	Data	☐		Number	8	
2	ATR03	Vykdytojas1	☐		Text	50	
3	ATR04	Vykdytojas2	☐		Text	50	
4	ATR05	Keliavimo budas	☐		Text	50	
5	ATR06	Eil.Nr.	☑		Number	10	
6	ATR07	Pakeitimo zyme	☐		Boolean	1	
7	ATR08	Atzyma apie pakeitin	☐		Boolean	1	
8	ATR09	Perdirbimo rusis	☐		Text	50	
9	ATR10	Sunaudotos detales	☐		Text	50	
10	ATR11	Nr	☑		Number	10	
11	ATR12	Data	☐		Number	8	
12	ATR13	Kodas	☑		Text	10	
13	ATR14	Pavadinimas	☐		Text	50	
14	ATR15	Telefonas	☐		Text	50	
15	ATR16	Gatves kodas	☑		Text	10	
16	ATR17	Gatves pavadinimas	☐		Text	50	
17	ATR18	Pastato Nr.	☐		Number	10	
18	ATR19	Nr	☑		Number	10	
19	ATR20	Pastatymo data	☐		Number	8	
20	ATR21	Parodymai pastacius	☐		Number	10	
21	ATR22	Uzplombuota	☐		Boolean	1	
22	ATR23	Statytojas	☐		Text	50	
23	ATR24	Parodymai nuemus	☐		Number	10	
24	ATR25	Eil.Nr.	☑		Number	10	

Record: 1 of 37

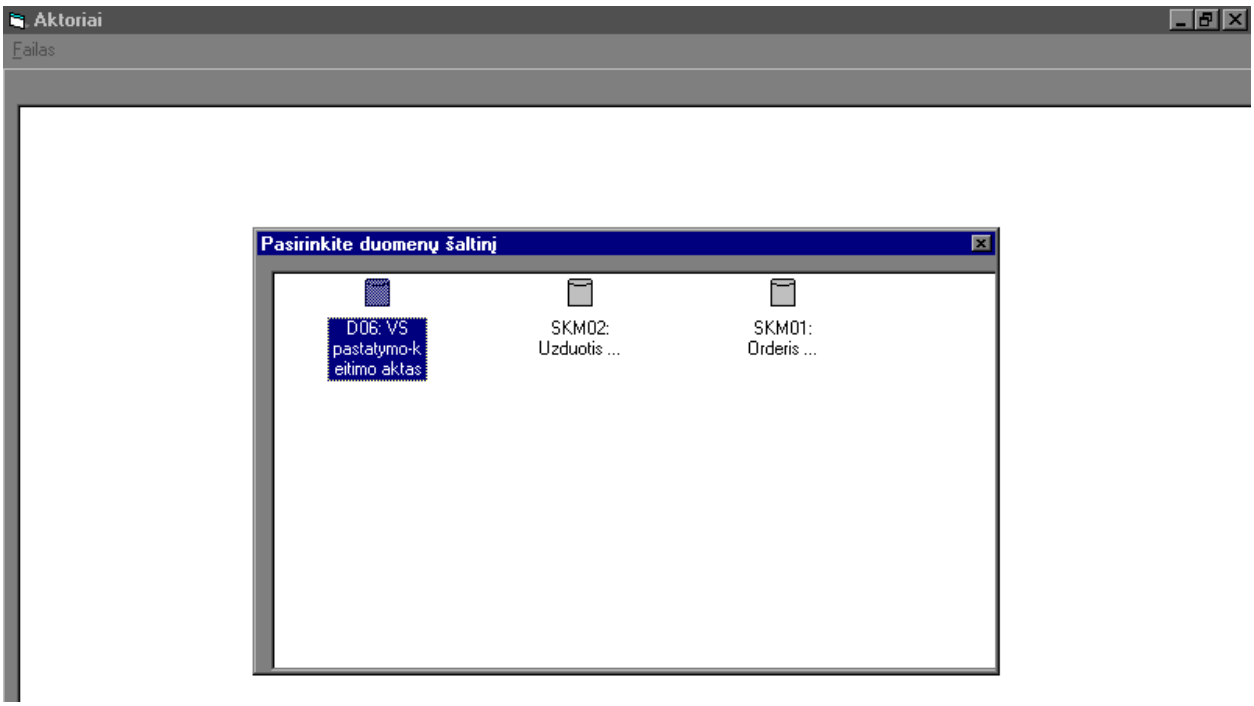
25 pav. Lentelė Atributas.

Aktorius : Table

id	kodas	pavadinimas
1	A01	Skaitikliu baro virsininkas
2	A02	Dispeceris
3	A03	Skaitiklio statytojas-keitejas
4	A04	Skaitikliu baro virsininkas
5	A05	Priimantis i remonta
6	A06	Grazinantis is remonto
7	A07	Atsiimantis is remonto
8	A08	Remonto baro darbuotojas
0		

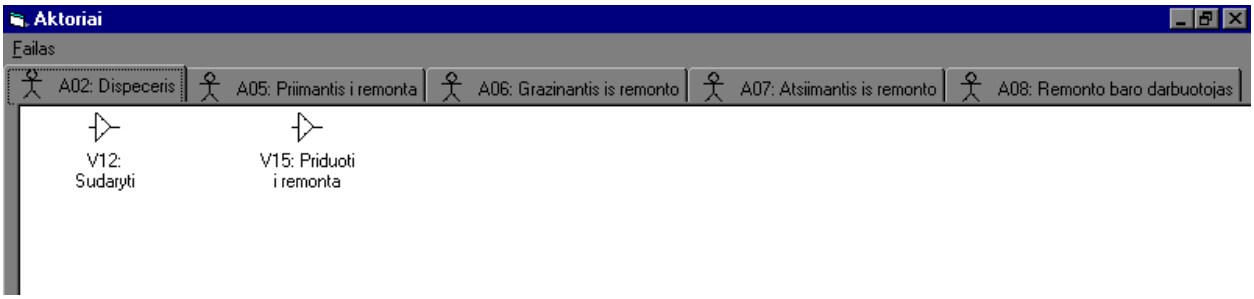
26 pav. Lentelė Aktorius.

Įvykdę programą, ekrane matome duomenų šaltinio pasirinkimo ekraninę formą:

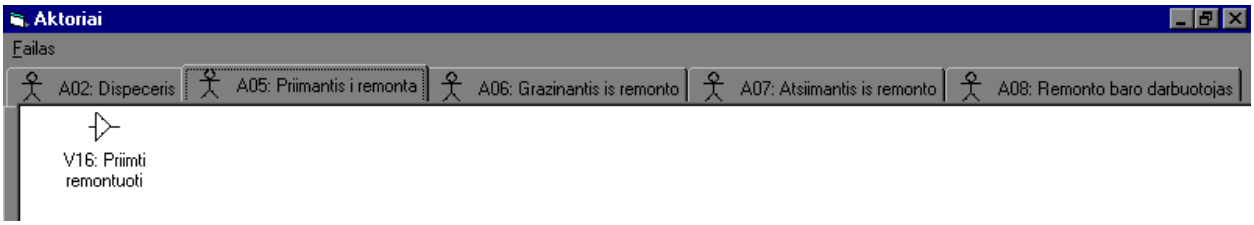


27 pav. Duomenų šaltinio pasirinkimo ekraninė forma.

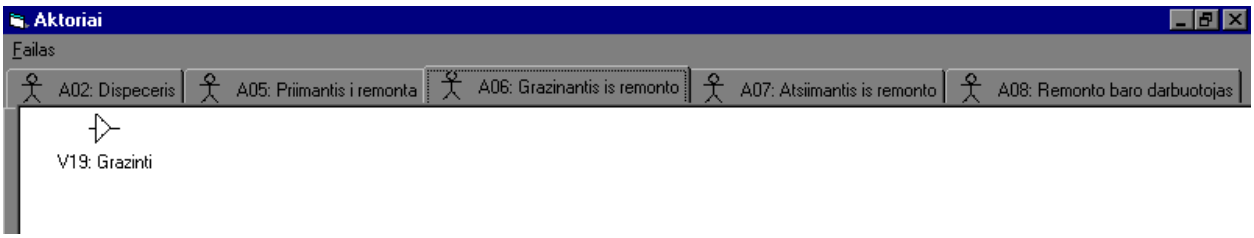
Pasirinkę duomenų šaltinį “Orderis VS priduoti į remontą ir grąžinti iš remonto”, matysime tokius aktorius ir jų atliekamus veiksmus:



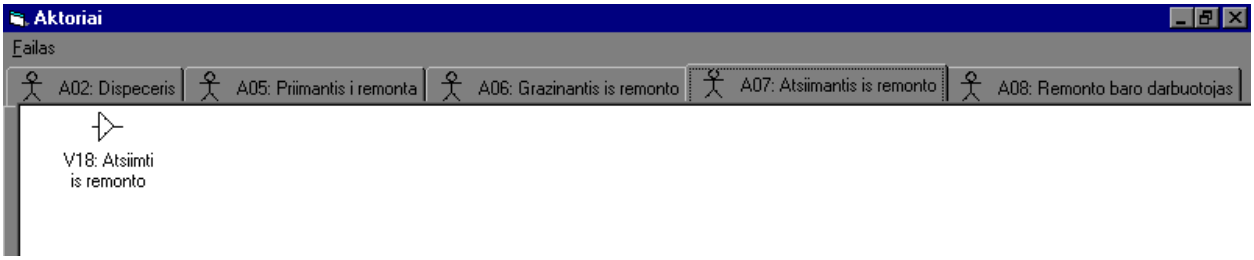
28 pav. Dispečerio veiksmai.



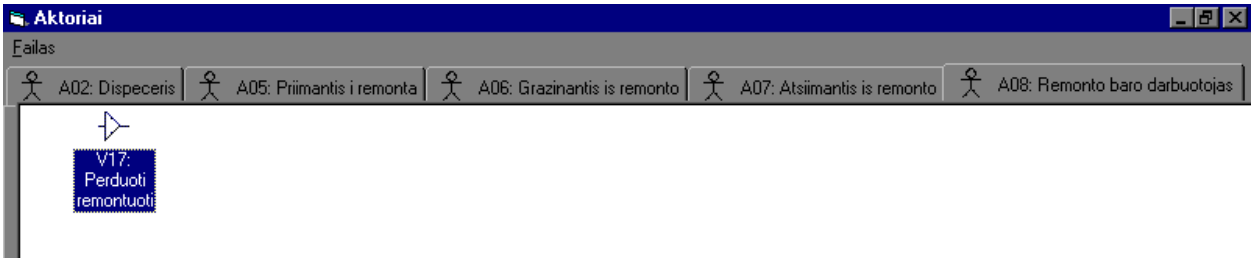
29 pav. Priimančiojo į remontą veiksmai.



30 pav. Grąžinančiojo iš remonto veiksmai.

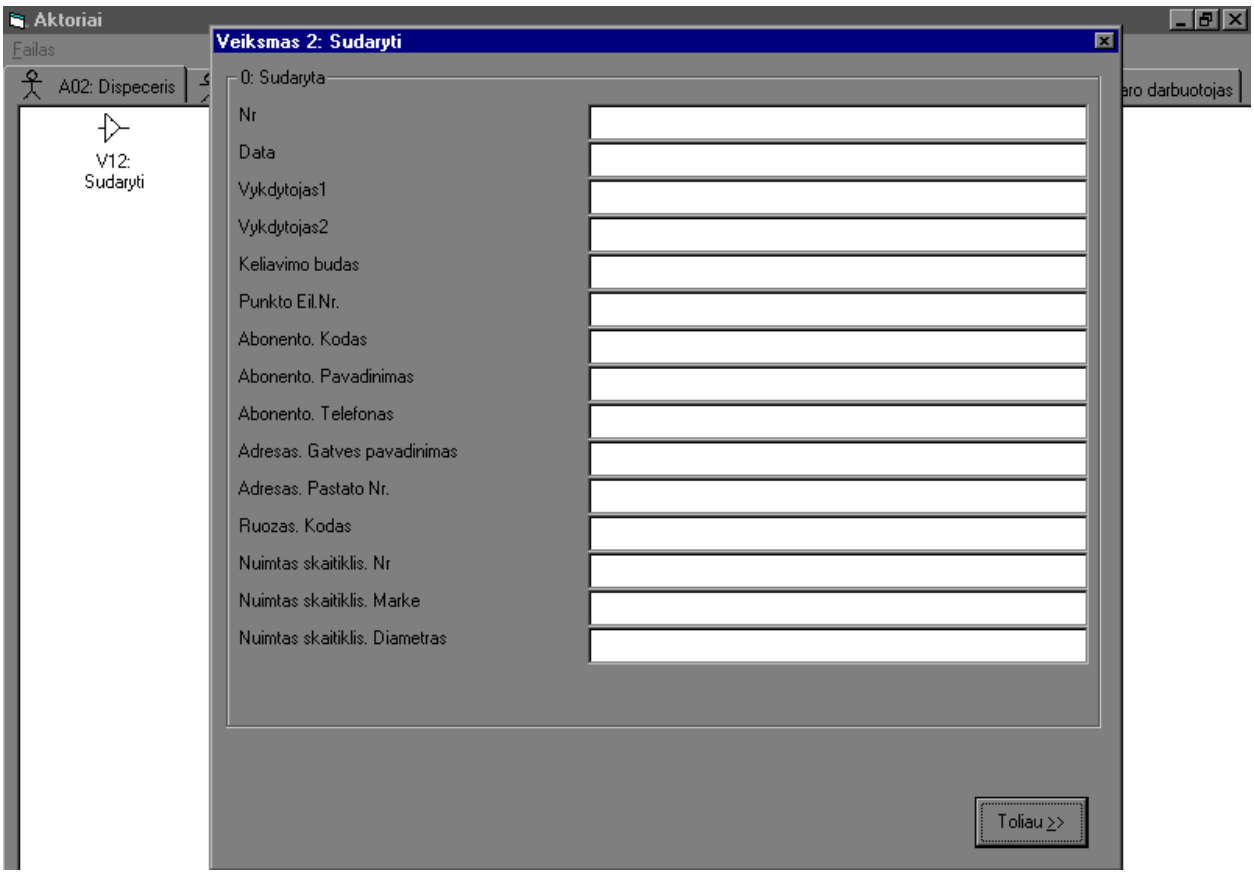


31 pav. Atsiimančiojo iš remonto veiksmai.

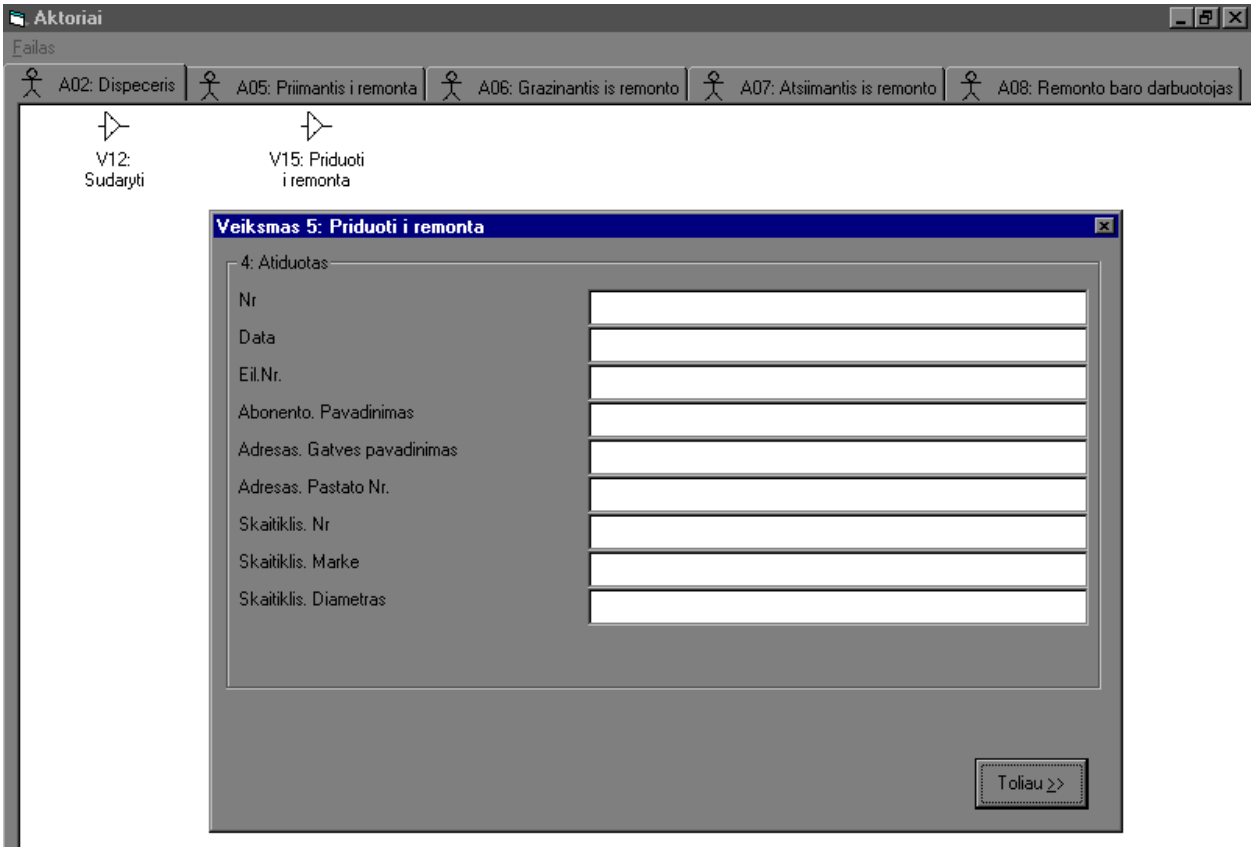


32 pav. Remonto baro darbuotojo veiksmai.

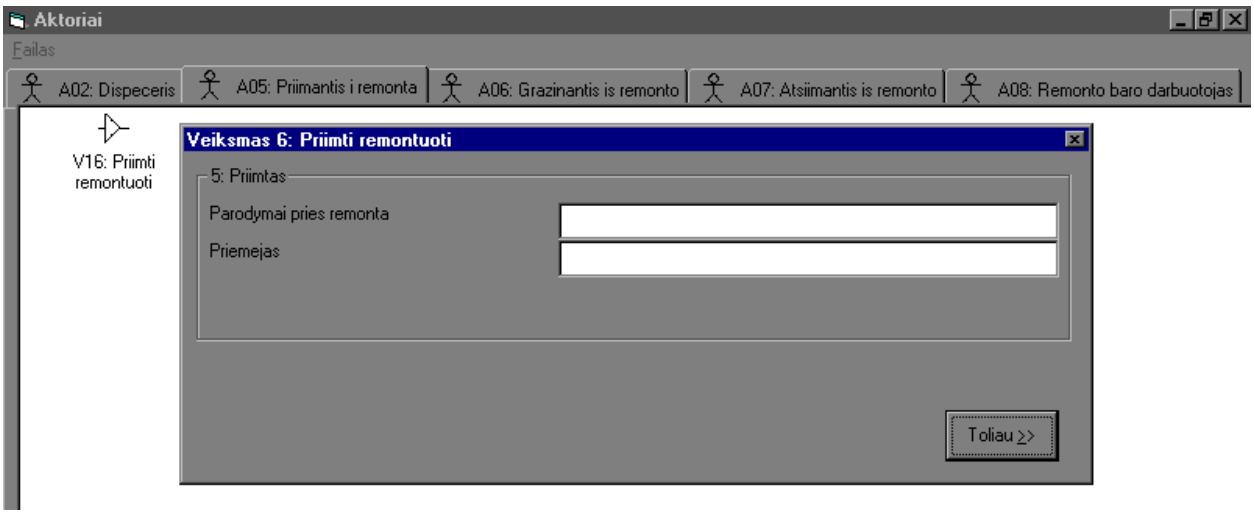
Kiekvienam veiksmui sugeneruojama ekraninė forma:



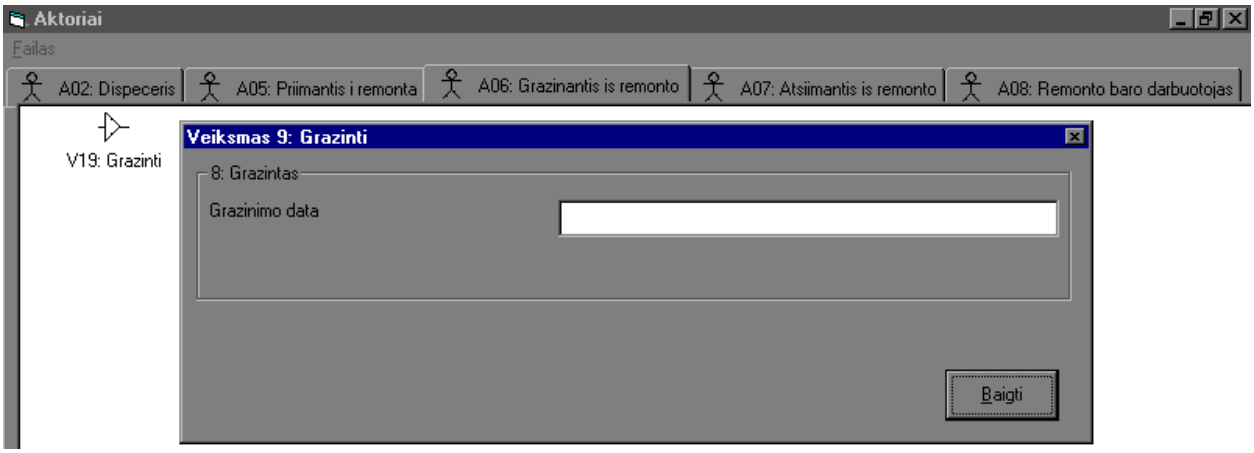
33 pav. Ekraninė forma veiksmui “Sudaryti”.



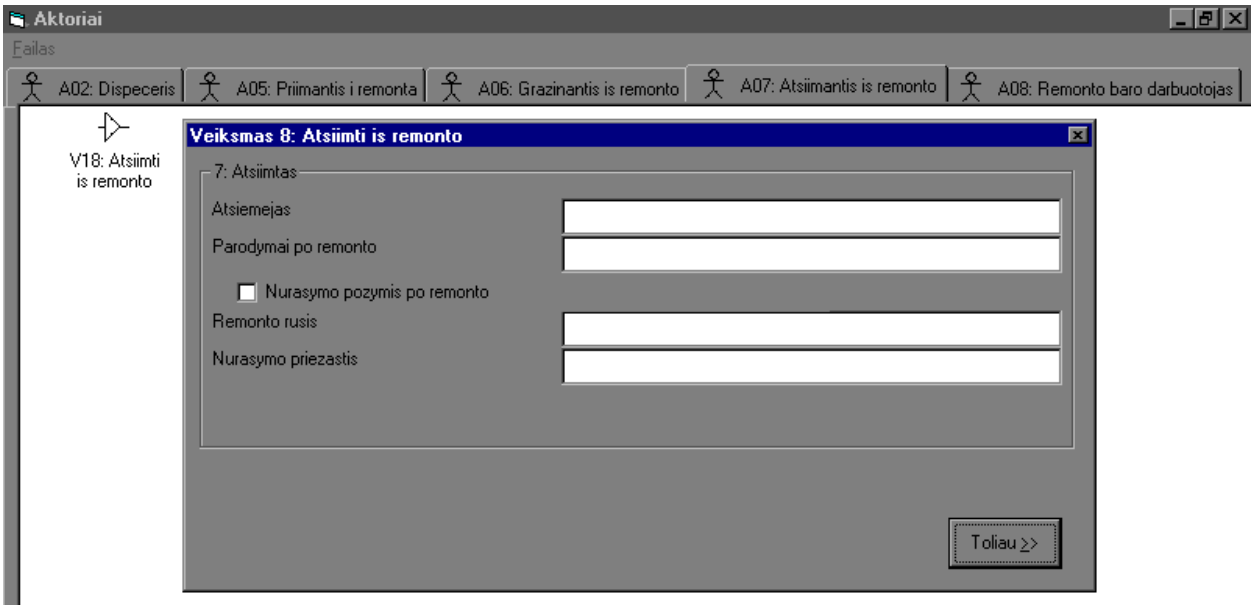
34 pav. Ekraninė forma veiksmui “Priduoti į remontą”.



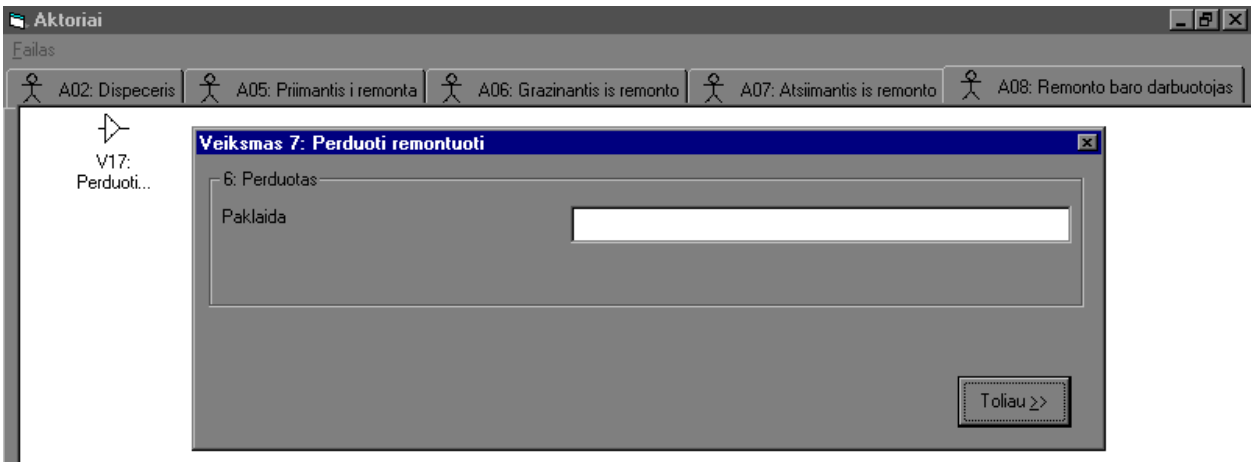
35 pav. Ekraninė forma veiksmui “Priimti remontuoti”.



36 pav. Ekraninė forma veiksmui “Grąžinti”.



37 pav. Ekraninė forma veiksmui “Atsiimti iš remonto”.



38 pav. Ekraninė forma veiksmui “Perduoti remontuoti”.

3.2.5. Reikalavimai sistemos funkcionavimo palaikymui

Reikalavimai techninei įrangai: procesorius AMD Duron, 750 MHz, operatyvinė atmintis 128 MB RAM.

Reikalavimai programinei įrangai: Microsoft Windows 98 operacinė sistema, Microsoft.NET Framework v.1.1.4322.

3.2.6. Sistemos naudojimo instrukcija

Paleidę vykdyti PĮ IS prototipui generuoti vykdomąjį failą ,,Aktoriai.exe", ekrane matome pagrindinę programos ekraninę formą su 3 mygtukais: ,,Generuoti sistemos prototipą", ,,Generuoti prototipą pasirinkto DŠ pagrindu" ir ,,Baigti darbą". Pasirinkę ,,Baigti darbą" išeiname iš programos.

Pasirinkę pagrindinės formos punktą ,,Generuoti prototipą pasirinkto DŠ pagrindu", matome duomenų šaltinio pasirinkimo formą su DŠ sąrašu ir požymiu, ar šiam šaltiniui jau buvo sugeneruoti ekr. formų prototipai. Ten matome 3 mygtukus: ,,Generuoti", ,,Paleisti vykdyti" ir ,,Išeiti". Pasirinkę punktą ,,Išeiti", grįžtame į pagrindinę programos formą. ,,Paleisti vykdyti" galime tik tada, jei pasirinktam šaltiniui jau sugeneruotos ekr. formos. Pasirinkę ,,Generuoti", matome pranešimą ,,Palaukite, vykdomas generavimas". Generavimui pasibaigus, matome pranešimą ,,Ar norite vykyti dabar?" ir mygtukus ,,Taip" ir ,,Ne". Pasirinkę ,,Ne", grįžtame į DŠ pasirinkimo formą. Jei pasirenkame ,,Taip", ekrane matome sugeneruotą pasirinkto DŠ pirmo etapo formą su įvedimo laukais ir mygtuku ,,Toliau". Paspaudę ,,Toliau", matome sekančio DŠ etapo formą. Jei etapas paskutinis, vietoje mygtuko ,,Toliau" matome mygtuką ,,Išeiti". Jį paspaudę grįžtame į pagrindinę programos formą.

Pasirinkę pagrindinės formos punktą ,,Generuoti sistemos prototipą", matome ekr. formą su mygtukais ,,Generuoti naują" ir ,,Paleisti vykdyti". Paleisti vykdyti galime tik jau turėdami sugeneruotą sistemos prototipą. Pasirinkę ,,Generuoti naują", matome pranešimą ,,Palaukite, vykdomas generavimas". Generavimui pasibaigus, matome pranešimą ,,Ar norite vykyti dabar?" ir mygtukus ,,Taip" ir ,,Ne". Pasirinkę ,,Ne", grįžtame į pagrindinę programos formą. Jei pasirenkame ,,Taip", ekrane matome sugeneruotą sistemos prototipą – aktorių meniu, su kiekvieno iš jų atliekamais veiksmais. Pasirinkę konkretų veiksmą atitinkantį meniu punktą, matome šį etapą atitinkančią ekr. formą su įvedimo laukais ir mygtuku ,,Išeiti". Jį paspaudę, grįžtame į aktorių meniu. Ir galime rinktis kitą aktorių ir vieną iš jo veiksmų. Aktorių meniu turi punktą ,,Išeiti". Jį pasirinkę, grįžtame į pagrindinės programos formos langą.

3.2.7. IS diegimo priemonių planas

Kad programa veiktų, pirmiausia reikia vartotojo kompiuteryje suinstaliuoti Microsoft .NET Framework v1.1.4322 į šį katalogą:

C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322. To reikia, nes prototipų generavimui naudojamas C# kompiliatorius.

Norint generuoti kompiuterizuotos IS prototipą, specifikacijos saugyklos DB (failas „Duombaze.mdb") ir vykdomasis IS prototipo generavimui skirtos PĮ failas („Aktoriai.exe") turi būti nukopijuoti į tą patį katalogą. IS prototipai generuojami į einamojo katalogo pakatalogį „Prototipai".

3.3. Projekto išvados

1. Specifikuoti funkciniai ir nefunkciniai reikalavimai kuriamai programinei įrangai, skirtai IS prototipo kūrimui. Sukurti vartotojų panaudojimo atvejų modelis, sistemos kontekstinė diagrama, DS klasių diagrama, vartotojų interfeiso modelis.
2. Suprojektuota sistemos architektūra, sudarytas realizacijos modelis, pateiktas kontrolinis duomenų pavyzdys.
3. Sistema realizuota naudojant Visual Basic 6 ir Visual C# programavimo kalbas.

4. Eksperimentinis tyrimas

4.1. Sukurtos sistemos kokybės tyrimas

Sukurtos sistemos kokybę įvertiname pagal 2.5 skyriuje apibrėžtus kokybės kriterijus 5 balų sistema (3 lentelė). 1 – sistema visiškai neatitinka iškelto kriterijaus, 2 – sistema blogai atitinka kriterijų, 3 – sistema patenkinamai atitinka kriterijų, 4 – sistema gerai atitinka kokybės kriterijų, 5 – sistema puikiai atitinka kriterijų.

3 lentelė

PĮ IS prototipui kurti atitikimas kokybės kriterijams

	1	2	3	4	5
PĮ funkcionalumas				+	
Sistemos tinkamumas specifikacijai validuoti				+	
Integracija į kuriamą CASE paketą			+		
Išbaigtumas			+		
Tolerancija klaidoms				+	
Perkeliamumas				+	

4.2. Tolimesnio sistemos tobulinimo, plėtojimo galimybės

1. Ateityje turi būti realizuota galimybė apjungti kelių duomenų šaltinių ekranines formas. Jei šie duomenų šaltiniai susiję, reikia įvertinti sąsają tarp jų.

2. Jei ateityje bus sukurtas CASE įrankio reikalavimų specifikacijai sudaryti modulis, generuojantis fizinę kliento DB, kuriamame IS prototipe turės būti realizuotas duomenų skaitymas iš kliento DB ir įrašymas į ją.

5. Išvados

1. Išanalizavus kompiuterizuotos IS prototipo kūrimo galimybes su automatizuoto projektavimo įrankiu *Oracle Designer* ir DBVS *Visual FoxPro* bei *MS Access* prieita prie išvados, kad reikia kurti naują sistemą, kuri leistų ankstyvoje kūrimo stadijoje sukurti sistemos prototipą, nežiūrint į tai, kad neturime sistemos projekto.

2. Specifikuoti funkciniai ir nefunkciniai reikalavimai kuriamai programinei įrangai, skirtai IS prototipo kūrimui.

3. Suprojektuota sistemos architektūra, sudarytas realizacijos modelis, pateiktas kontrolinis duomenų pavyzdys.

4. Kompiuterizuotos IS prototipo generavimas realizuotas naudojant Visual Basic 6 ir Visual C# programavimo kalbas.

5. Atliktas sukurtos sistemos kokybės tyrimas ir aptartos tolimesnio jos tobulinimo galimybės.

6. Reikalavimų inžinieriaus gaunama nauda iš sukurtos PĮ IS prototipo kūrimui:

    a) Norint sukurti IS prototipą, nereikia turėti suprojektuotos DB.

    b) Gaunamos eskizinės ekraninės formos palengvina reikalavimų inžinieriaus darbą tikslinant užsakovo poreikius ir reikalavimus.

6. Literatūra

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

2. Butkienė, R. Informacijos sistemų projektavimas Oracle Designer/2000 priemonėmis. Kaunas, ,,Technologija" 1998.

3. Green, S. The Use of Application Domain Knowledge (and Other Techniques) for Facilitating Requirements Capture [interaktyvus]. [žiūrėta 2003.01.15]. Prieiga per internetą: http://citeseer.nj.nec.com.

4. Mascardi. An Agent-Based Approach to Distributed Simulation (1999) [interaktyvus]. [žiūrėta 2003.01.15]. Prieiga per internetą: http://citeseer.nj.nec.com/mascardi99agentbased.html.

5. IEEE Recommended Practice for Software Requirements - Sponsor Software(1993) [interaktyvus]. [žiūrėta 2003.01.15]. Prieiga per internetą: http://citeseer.nj.nec.com/429289.html.

6. Zini. Caselp, A Rapid Prototyping Environment For Agent Based Software (2001) [interaktyvus]. [žiūrėta 2003.01.15]. Prieiga per internetą: http://citeseer.nj.nec.com/zini01caselp.html.

7. Pomberger. Prototyping-Oriented Software Development - Concepts [interaktyvus]. [žiūrėta 2003.01.15]. Prieiga per internetą: http://citeseer.nj.nec.com/396992.html.

8. Shefrin, Purtilo. Tool Support For Collaborative Software Prototyping (1994) [interaktyvus]. [žiūrėta 2003.01.15]. Prieiga per internetą: http://citeseer.nj.nec.com/shefrin94tool.html.

9. Ozcan, P. The Application of Visualisation to Requirements Engineering [interaktyvus]. [žiūrėta 2003.01.15]. Prieiga per internetą: http://citeseer.nj.nec.com/45481.html.

10. Kordon. Proposal For A Generic Prototyping Approach (1994) [interaktyvus]. [žiūrėta 2003.01.15]. Prieiga per internetą: http://citeseer.nj.nec.com/kordon94proposal.html.

11. Callahan, Schneider, Easterbrook. Automated Software Testing Using Model-Checking (1996) [interaktyvus]. [žiūrėta 2003.01.15]. Prieiga per internetą: http://citeseer.nj.nec.com/callahan96automated.html.

7. Terminų ir santrumpų žodynas

Informacinė sistema (IS) – tai sistema, kurios tarpusavyje susiję komponentai dirbdami kartu surenka, apdoroja, saugo ir platina informaciją, kuri organizacijoje padeda priimti sprendimus, koordinuoti ir kontroliuoti veiklą, analizuoti problemas, vizualizuoti sudėtingus objektus, kurti naujus produktus.

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

Reikalavimų inžinerija – tai reikalavimų analizės, reikalavimų specifikacijos sudarymo ir jos plėtros sisteminis procesas, kuriuo norima suprasti kuriamai programinei įrangai (PĮ) vartotojo reikalavimus.

DBVS – duomenų bazių valdymo sistema.

8. Priedai

8.1. Programos tekstas

Forma „MainForm”:

```
Private ActionSelected
Private Sub Data1_Validate(Action As Integer, Save As Integer)

End Sub

Private Sub ActionList_ItemClick(ByVal Item As MSComctlLib.ListItem)
   ActionSelected = Item.tag
End Sub


Private Sub ActionList_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)
   If ActionSelected <> -1 Then
      DataModule.PirmasEtapas (ActionSelected)
      ActionSelected = -1
   End If
End Sub

Private Sub ActorTabs_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)
   SelectActor (ActorTabs.SelectedItem.tag)
End Sub

Private Sub DB_MenuItem_Click()
On Error GoTo catch
   prevfn = DatabaseDialog.FileName
   DatabaseDialog.ShowOpen
   If prevfn = DatabaseDialog.FileName Then GoTo finally
   LoadDatabase (DatabaseDialog.FileName)
   GoTo finally
catch:
   MsgBox ("Įvyko klaida: " + Error)
finally:
End Sub

Private Sub DS_Menu_Click()
   DataModule.SelectDataSource
End Sub

Private Sub Exit_MenuItem_Click()
   Unload Me
End Sub

Private Sub Form_Load()
   ActionSelected = -1
   LoadDatabase (DataModule.DefaultDatabase)
End Sub

Public Sub addControl(collection As VB.Control, tag)
   prev = collection(collection.Count - 1)
   Load collection(collection.Count)
   collection(collection(Count - 1)).Top = prev.Top + prev.Height
End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
   Toolbar1.Buttons.Add
   Button = Toolbar.Buttons(Toolbar.Buttons.Count - 1)
   Button.Image = ImageList1(0)
```

```
End Sub

Private Sub Toolbar1_Click()
   Toolbar1.Buttons.Add
   Toolbar1.Buttons(1).Image = ImageList1.ListImages("actor")

End Sub

Private Sub Form_Resize()
   ActorTabs.Width = Width + 100
   ActorTabs.Height = Height - 500
   ActionList.Width = ActorTabs.ClientWidth - 100
   ActionList.Height = ActorTabs.ClientHeight - 100
End Sub
```

Forma „DatasourceDialog”:

```
Option Explicit

Private Sub DatasourceList_ItemClick(ByVal Item As MSComctlLib.ListItem)
   DatasourceDialog.Hide
   DataModule.LoadDataSource (Item.tag)
End Sub

Private Sub Form_Activate()
   Dim rs As DAO.Recordset
   Dim Item As MSComctlLib.ListItem

   DatasourceList.ListItems.Clear

   Set rs = DataModule.Datasource.OpenRecordset("duom_salt")

   If rs.EOF Then MsgBox ("Pasirinktoje duomenų bazėje nėra šaltinių")
   While Not rs.EOF
      DatasourceList.ListItems.Add , , rs.Fields("kodas") + ": " + rs.Fields("pavadinimas"), "datasource"
      Set Item = DatasourceList.ListItems(DatasourceList.ListItems.Count)

      Item.tag = rs.Fields("id")

      rs.MoveNext
   Wend

End Sub

Private Sub Form_Load()
   DatasourceList.Icons = MainForm.ImageList1
End Sub
```

Forma „EtapasForm”:

```
Sub ClearControls()
   For i = 1 To EtapasForm.Labels.Count - 1
      Unload EtapasForm.Labels(i)
   Next i

   For i = 1 To EtapasForm.Textboxes.Count - 1
      Unload EtapasForm.Textboxes(i)
   Next i

   For i = 1 To EtapasForm.Checkboxes.Count - 1
      Unload EtapasForm.Checkboxes(i)
   Next i
```

```
    Height = 2400
    Frame_Attrib.Height = 615
    Frame2.Top = 1395

End Sub

Sub CreateControl(Caption As String, ctrType As String, Value, MaxLength As Integer)
    ctrType = Trim(UCase(ctrType))

    Dim lbl As label
    Load Labels(Labels.Count)
    Set lbl = Labels(Labels.Count - 1)
    lbl.Caption = Caption
    lbl.Top = Labels(Labels.Count - 2).Top + 360

    Height = 2400 + Labels.Count * 360
    Frame_Attrib.Height = 615 + Labels.Count * 360
    Frame2.Top = 1395 + Labels.Count * 360

    Select Case ctrType
        Case "BOOLEAN":
            Dim chk As checkbox

            Load Checkboxes(Checkboxes.Count)

            Set chk = Checkboxes(Checkboxes.Count - 1)

            chk.Caption = Caption
            chk.Value = (Value = True)
            chk.Visible = True
            chk.tag = lbl.Index
            lbl.tag = chk.Index
            chk.Top = lbl.Top

        Case Else:
            Dim txt As textbox

            Load Textboxes(Textboxes.Count)

            Set txt = Textboxes(Textboxes.Count - 1)

            txt.Text = Value
            txt.MaxLength = MaxLength
            txt.Visible = True
            txt.Top = lbl.Top
            txt.tag = lbl.Index
            lbl.tag = txt.Index
            lbl.Visible = True

    End Select

End Sub


Private Sub NextButton_Click()
    failas = FreeFile()
    Open Trim(Str(DataModule.EtapoId)) + ".frm" For Output As failas
    FormosSaugojimas.SaveEtapasForm (failas)
    Close #failas

    ClearControls
    DataModule.Etapas (DataModule.KitoEtapoId)
```

```
End Sub

Private Sub OKButton_Click()
   failas = FreeFile()
   Open Trim(Str(DataModule.EtapoId)) + ".frm" For Output As failas
   FormosSaugojimas.SaveEtapasForm (failas)
   Close #failas

   ClearControls
   Unload Me
   MainForm.Show
End Sub
```

Modulis „DataModule”:

```
Public Const DefaultDatabase = "duombaze_msa2.mdb"
Public Datasource As DAO.database
Public DataSourceId As Integer
Public DataSourceName As String
Public ActorId As Integer
Public EtapoId As Integer
Public KitoEtapoId As Integer


Public Sub LoadDatabase(database As String)
   Dim rs As DAO.Recordset

   Set DataModule.Datasource = DAO.OpenDatabase(database)

   MainForm.ActorTabs.Tabs.Clear
   MainForm.ActionList.ListItems.Clear

   DataSourceId = 1
   SelectDataSource

   Set rs = Datasource.OpenRecordset("duom_salt")

End Sub

Public Sub SelectDataSource()
   DatasourceDialog.Show Modal = True, MainForm
End Sub

Public Sub LoadDataSource(Datasource As Integer)

   Dim rs As DAO.Recordset
   Dim sqlstmt As String

   sqlstmt = _
      "SELECT pavadinimas " + _
      "FROM Duom_Salt " + _
      "WHERE (id=" + Trim(Str(Datasource)) + ") "


   Set rs = DataModule.Datasource.OpenRecordset(sqlstmt)

   If Not rs.EOF Then DataSourceName = rs.Fields("pavadinimas") Else DataSourceName = "<noname>"
   'MsgBox (DataSourceName)

   DataModule.DataSourceId = Datasource
   LoadActors
End Sub
```

```
Public Sub LoadActors()
   Dim rs As DAO.Recordset
   Dim atab As MSComctlLib.Tab
   Dim sqlstmt As String

   MainForm.ActorTabs.Tabs.Clear

   sqlstmt = _
      "SELECT Aktorius.id, Aktorius.kodas, Aktorius.pavadinimas " + _
      "FROM Aktorius, Veiksmas_DS, Perejimas_Et, Etapas " + _
      "WHERE (Etapas.ds_id=" + Trim(Str(DataModule.DataSourceId)) + ") " + _
      "AND (Perejimas_Et.et_i_id=Etapas.id) " + _
      "AND (Veiksmas_DS.id=Perejimas_Et.v_id) " + _
      "AND (Aktorius.id=Veiksmas_DS.ak_id1) " + _
      "GROUP BY Aktorius.id, Aktorius.kodas, Aktorius.pavadinimas"

   'MsgBox (sqlstmt)

   Set rs = DataModule.Datasource.OpenRecordset(sqlstmt)

   MainForm.ActionList.Visible = False
   If rs.EOF Then
      MainForm.ActorTabs.Visible = False
      MsgBox ("Nėra aktorių, susietų su pasirinktu duomenų šaltiniu")
   Else
      MainForm.ActorTabs.Visible = True

    While Not rs.EOF
      MainForm.ActorTabs.Tabs.Add , , rs.Fields("kodas") + ": " + rs.Fields("pavadinimas"), "actor"
      Set atab = MainForm.ActorTabs.Tabs(MainForm.ActorTabs.Tabs.Count)

      atab.tag = rs.Fields("id")

      rs.MoveNext
    Wend

    SelectActor (MainForm.ActorTabs.Tabs(1).tag)

   End If

End Sub
Public Sub SelectActor(id As Integer)
   DataModule.ActorId = id

   Dim rs As DAO.Recordset
   Dim Item As MSComctlLib.ListItem
   Dim sqlstmt As String


   sqlstmt = _
      "SELECT * " + _
      "FROM Veiksmas_DS " + _
      "WHERE (Veiksmas_DS.ak_id1=" + Trim(Str(id)) + ")"

   'MsgBox (sqlstmt)
   Set rs = DataModule.Datasource.OpenRecordset(sqlstmt)

   MainForm.ActionList.ListItems.Clear
   If rs.EOF Then
      MainForm.ActionList.Visible = False
      MsgBox ("Nėra veiksmų, atliekamų šio aktoriaus")
   Else
      MainForm.ActionList.Visible = True
```

```
      While Not rs.EOF
        MainForm.ActionList.ListItems.Add , , rs.Fields("kodas") + ": " + rs.Fields("pavadinimas"), "action"
        Set Item = MainForm.ActionList.ListItems(MainForm.ActionList.ListItems.Count)

        Item.tag = rs.Fields("id")

        rs.MoveNext
      Wend
    End If
End Sub

Sub PirmasEtapas(veiksmoId As Integer)
    Dim rs As DAO.Recordset
    Dim Item As MSComctlLib.ListItem
    Dim sqlstmt As String

    sqlstmt = _
        "SELECT Etapas.*, Veiksmas_DS.pavadinimas AS vpav " + _
        "FROM Etapas,Perejimas_Et, Veiksmas_DS " + _
        "WHERE (Perejimas_Et.v_id=" + Trim(Str(veiksmoId)) + ") " + _
        "AND (Etapas.id=Perejimas_Et.et_i_id) " + _
        "AND (Veiksmas_DS.id=Perejimas_Et.v_id)"

    'MsgBox (sqlstmt)

    Set rs = DataModule.Datasource.OpenRecordset(sqlstmt)
    If rs.EOF Then
        MsgBox ("Šis veiksmas neturi etapų")
    Else
        EtapasForm.Caption = "Veiksmas " + Trim(Str(veiksmoId)) + ": " + rs.Fields("vpav")
        Etapas (rs.Fields("id"))
    End If

End Sub

Sub Etapas(id As Integer)
    EtapoId = id

    Dim rs As DAO.Recordset
    Dim Item As MSComctlLib.ListItem
    Dim sqlstmt As String

    sqlstmt = _
        "SELECT * " + _
        "FROM Etapas " + _
        "WHERE (Etapas.id=" + Trim(Str(id)) + ")"

    'MsgBox (sqlstmt)

    Set rs = DataModule.Datasource.OpenRecordset(sqlstmt)
    If rs.EOF Then
        MsgBox ("Nėra tolesnio etapo")
        GoTo exception
    Else
        EtapasForm.Frame_Attrib.Caption = Trim(Str(id)) + ": " + rs.Fields("pavadinimas")
    End If

    KitoEtapoId = KitasEtapas(id)

    EtapasForm.NextButton.Visible = (KitoEtapoId <> -1)
    EtapasForm.OKButton.Visible = (KitoEtapoId = -1)
```

```
    sqlstmt = "select * from etapo_atributai where et_id=" + Trim(Str(DataModule.EtapoId)) + " order by id"
    Set rs = DataModule.Datasource.OpenRecordset(sqlstmt)


    If rs.EOF Then
       MsgBox ("Nėra atributų!")
    Else
      While Not rs.EOF
       Call EtapasForm.CreateControl(rs.Fields("vard_org"), rs.Fields("tipas"), "", rs.Fields("ilgis"))
       rs.MoveNext
      Wend
    End If


    MainForm.Hide
    EtapasForm.Show Modal = True, MainForm
exception:
    MainForm.Show
End Sub

Function KitasEtapas(EtapoId As Integer) As Integer
    Dim rs As DAO.Recordset
    Dim sqlstmt As String

    sqlstmt = _
       "SELECT Perejimas_Et.et_i_id " + _
       "FROM Perejimas_Et " + _
       "WHERE (Perejimas_Et.et_is_id=" + Trim(Str(EtapoId)) + ")"

    Set rs = DataModule.Datasource.OpenRecordset(sqlstmt)

    If rs.EOF Then KitasEtapas = -1 Else KitasEtapas = rs.Fields(0)
End Function
```

Modulis „FormosSaugojimas”:

```
Sub SaveEtapasForm(failas)
Print #failas, "Version 5.00"
Print #failas, "Begin VB.Form Forma" + Trim(Str(EtapoId))
Print #failas, "   Caption = ";
Write #failas, EtapasForm.Caption + "; Etapas: " + Trim(Str(EtapoId)) + "; Saltinis: " +
DataModule.DataSourceName

Print #failas, "   ClientHeight = "; EtapasForm.Frame_Attrib.Height
Print #failas, "   Left = "; EtapasForm.Left
Print #failas, "   Top = "; EtapasForm.Top
Print #failas, "   ClientWidth = "; EtapasForm.Frame_Attrib.Width
Print #failas, "   ScaleHeight = 7620"
Print #failas, "   ScaleWidth = 9645"
Print #failas, "   StartUpPosition = 3    'Windows Default"


For i = 1 To EtapasForm.Labels.Count - 1
    Dim label As VB.label
    Set label = EtapasForm.Labels(i)
    Print #failas, "   Begin VB.Label Label" & Trim(Str(i))
    Print #failas, "      Height = "; label.Height
    Print #failas, "      Left = "; label.Left
    Print #failas, "      Top = "; label.Top
    Print #failas, "      Width = "; label.Width
    Print #failas, "      Caption = ";
    Write #failas, label.Caption
    Print #failas, "   End"
```

```
Next i

For i = 1 To EtapasForm.Textboxes.Count - 1
   Dim textbox As VB.textbox
   Set textbox = EtapasForm.Textboxes(i)
   Print #failas, "   Begin VB.Textbox Textbox" & Trim(Str(i))
   Print #failas, "      Height = "; textbox.Height
   Print #failas, "      Left = "; textbox.Left
   Print #failas, "      Top = "; textbox.Top
   Print #failas, "      Width = "; textbox.Width
   Print #failas, "      Text = ";
   Write #failas, textbox.Text
   Print #failas, "   End"
Next i

For i = 1 To EtapasForm.Checkboxes.Count - 1
   Dim checkbox As VB.checkbox
   Set checkbox = EtapasForm.Checkboxes(i)
   Print #failas, "   Begin VB.Checkbox Checkbox" & Trim(Str(i))
   Print #failas, "      Height = "; checkbox.Height
   Print #failas, "      Left = "; checkbox.Left
   Print #failas, "      Top = "; checkbox.Top
   Print #failas, "      Width = "; checkbox.Width
   Print #failas, "      Caption = ";
   Write #failas, checkbox.Caption
   Print #failas, "      Value = ";
   Write #failas, checkbox.Value
   Print #failas, "   End"
Next i



Print #failas, "End"

Print #failas, "Attribute VB_GlobalNameSpace = False"
Print #failas, "Attribute VB_Creatable = False"
Print #failas, "Attribute VB_PredeclaredId = True"
Print #failas, "Attribute VB_Exposed = False"
Print #failas, "Private Sub Form_GotFocus()"
Print #failas, "End Sub"


End Sub
```