KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Donata Montrimaitė

Reikalavimų specifikacijos pilnumo įvertinimo galimybių tyrimas

Magistro darbas

Darbo vadovė:
dr. doc. R. Butkienė

Kaunas, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PRAKTINĖS INFORMATIKOS KATEDRA

Donata Montrimaitė

Reikalavimų specifikacijos pilnumo įvertinimo galimybių tyrimas

Magistro darbas

Recenzentas
doc. dr. S. Maciulevičius
2007-01-09

Vadovė
doc. dr. R. Butkienė
2007-01

Atliko
IFM-1/4 gr. stud.
Donata Montrimaitė
2007-07-09

Kaunas, 2007

Turinys

1. Įvadas

Vartotojo reikalavimų kompiuterizuotai informacinei sistemai specifikavimas yra vienas iš svarbiausių žingsnių ne tik pradiniame, bet ir vėlesniuose programų sistemos kūrimo etapuose. Šio žingsnio rezultato – reikalavimų specifikacijos – kokybė lemia visos sistemos kūrimo sėkmę (arba nesėkmę). Todėl, sudaryti pakankamai kokybišką reikalavimų specifikaciją yra vienas iš pagrindinių reikalavimų inžinerijos tikslų.

Reikalavimų specifikacijos kokybę nusako įvairios charakteristikos, kurias analitikas turi patikrinti, kad nustatyti specifikacijos kokybės lygį. Specifikacijos kokybės tikrinimas yra sudėtingas uždavinys. Paprastai specifikacija tikrinama rankiniu būdu, bet, pavyzdžiui, įvertinti specifikacijos pilnumą, minimalumą, neperteklišKumą ir pan. būtų lengviau automatizuotomis priemonėmis. Tačiau automatizuotas priemones galima sukurti, jei reikalavimams specifikuoti naudojamas formalus arba pusiau formalus specifikavimo metodas. Vienas iš tokių – metodas ODRES (Output Driven REquirements Specification) - plėtojamas KTU Informacijos sistemų katedroje.

Šio darbo tikslas – sukurti įrankį, kurio pagalba nustatomas ODRES metodu sudarytos reikalavimų specifikacijos pakankamumas bei minimalumas. Automatiškai patikrinti, ar sudaryta reikalavimų specifikacija yra pilna – neįmanoma, nes įrankis negali patikrinti, ar žmogus ko nors nepamiršo. Tačiau ODRES metodas sudaro prielaidas sukurti įrankį, kuris automatiškai patikrintų, ar sudarytos specifikacijos pakanka, kad būtų galima sukurti veikiančią programų sistemą, bei identifikuotų vietas, kurios gal būt yra perteklinės.

Įrankis, ODRES metodu sudarytos reikalavimų specifikacijos pakankamumui bei minimalumui nustatyti, realizuotas Visio2003 priemonės pagalba, kuri palaiko VBA programavimo kalbą. Šia kalba parasytas programos kodas, leidžia prisijungti prie duomenų bazės ir leidžia įvertinti jos pakankamumą.

Darbo aprašo antrame skyriuje pateikta tyrimo srities, objekto ir problemos apibrėžtys, aprašomas reikalavimų specifikavimas, pateikta reikalavimų specifikacijos metodų ir priemonių parinkimo analizė. Taip pat šiame skyriuje yra pateiktas projekto tikslas, reikalavimų specifikacijos kokybės vertinimas, reikalavimų specifikacijų pilnumo ir pakankamumo analizė. Trečiame skyriuje pateikti reikalavimai RS kokybės vertinimui, ketvirtame skyriuje- reikalavimų specifikacijos pilnumo vertinimo įrankio projekto aprašymas, penktame skyriuje – eksperimento aprašymas.

2. Analitinė dalis

2.1 Tyrimo sritis, objektas ir problema

Projekto tyrimo sritis – IS reikalavimų specifikacijų pilnumo patikrinimas.

Projekto tyrimo srities objektas - tai informacijos sistemų katedroje plėtojamas ODRES metodas. Šis metodas skirtas specifikacijų pilnumui, minimalumui patikrinti.

IS problema - kaip nustatyti IS reikalavimų specifikacijų kokybę? Kaip ją pamatuoti? Kokia IS turi būti, kad būtų kokybiška?

2.2 Reikalavimų specifikavimas

Reikalavimų specifikavimas yra vienas svarbiausių IS kūrimo etapų. Nuo reikalavimų specifikavimo priklauso kaip bus realizuota kuriama informacinė sistema. Reikalavimai specifikuojami, norint sukonkretinti ir tiksliai apibrėžti, kokie yra vartotojo pateikti reikalavimai kuriamai informacinei sistemai.

Reikalavimų specifikacija - tai dokumentas, skirtas kokio nors objekto (proceso) reikalavimams aprašyti. Tikslus specifikacijos aprašymas yra vienas pagrindinių tikslų ruošiant reikalavimų specifikaciją. Tinkamai atlikus reikalavimų specifikavimą yra pasiekiama svarbi atrama kuriant programinę įrangą, kuri atitinka aukšto lygio funkcionalumą, paprastą palaikymą ir daugeliui vartotojų prieinamas išlaidas.[4]

Reikalavimai specifikuojami prieš programinės įrangos darbus. Norint gauti kokybišką reikalavimų specifikaciją, reikia išsiaiškinti visus veiksmus, kurie apima darbus susijusius su reikalavimų išsiaiškinimu, analizavimu, dokumentavimu ir palaikymu. Specifikuojant reikalavimus yra dedamos pastangos darbui su tarpininkais - surinkti visą informaciją, reikalingą suprasti problemą. Surinkus visus reikalavimus, jie yra analizuojami, po to siejami su reikalavimų tobulinimu ir modeliavimu. Analizės tikslas yra išsiaiškinti problemas, neužbaigtumus ir nesuderinamumus gautuose reikalavimuose.

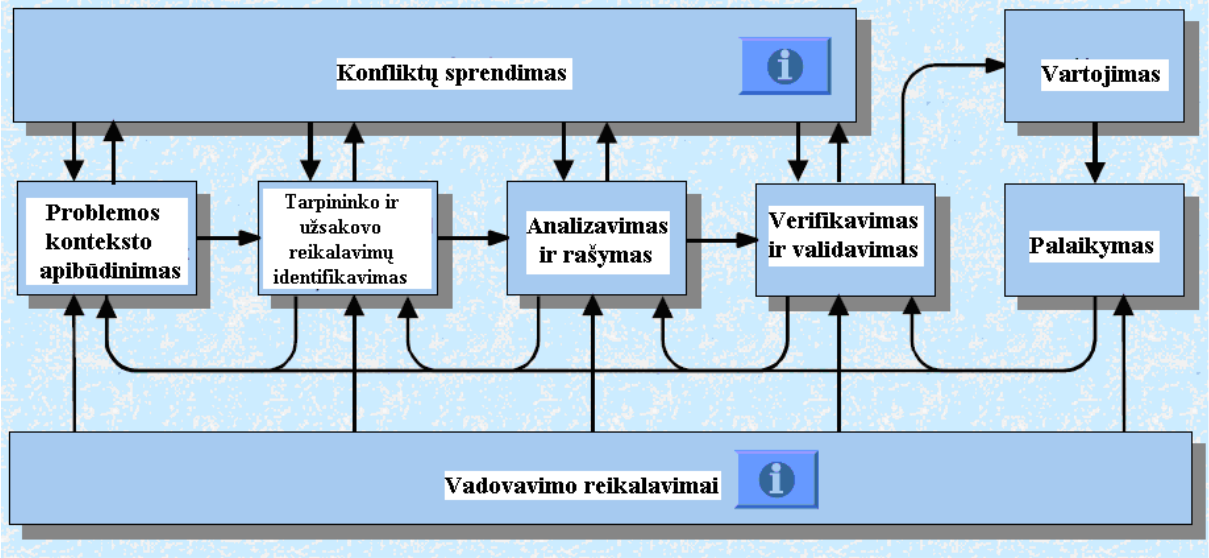
Pirminis programinės įrangos sėkmės matas yra laipsnis, kuris parodo kaip sistema atitinka paskirtį, kuriai ji buvo sukurta. Kalbant plačiąja prasme, programinės įrangos reikalavimų inžinerija – yra procesas šiai paskirčiai atrasti. Ir tai pasiekiama identifikuojant proceso dalyvius ir jų poreikius, dokumentuojant juos formoje, kuri būtų tinkama analizuoti, komunikuoti ir, galiausiai, realizuoti sistemai.

Šiame procese iškyla keletas būdingų sunkumų. Procese dalyvauja ne vienas dalyvis (įskaitant užsakovus, vartotojus ir kūrėjus), jų tikslai gali nesutapti žiūrint iš jų darbo aplinkos perspektyvos ir užduočių kurias tenka jiems atlikti. Jų išreikšti tikslai gali būti neaiškūs arba sunkiai įgyvendinami, dėl to tų tikslų įgyvendinimas gali būti apribotas daugybe faktorių nepriklausančių nuo dalyvių kontrolės.

Reikalavimų inžinerija - programinės inžinerijos šaka susieta su realaus pasaulio keliamais tikslais, funkcijomis, ribojamomis programinių įrangų sistemose. Ji taip pat susieta su šių faktorių tarpusavio ryšiu tiksliai programinės įrangos specifikacijų elgsenai.[4]

Apibūdinime pavartotas žodis realus pasaulis, kuris motyvuoja programinės įrangos kūrimą. Tai apibūdina sistemos kūrimo „kodėl" taip pat ir „kas" klausimus. Apibrėžimas pabrėžia ir „tikslias specifikacijas". Šie apibūdinimai suteikia pagrindą analizės reikalavimams, tikrinant ar jie iš tikrųjų yra tai ko nori vartotojas, apibrėžiant ką turi suprojektuoti projektuotojai

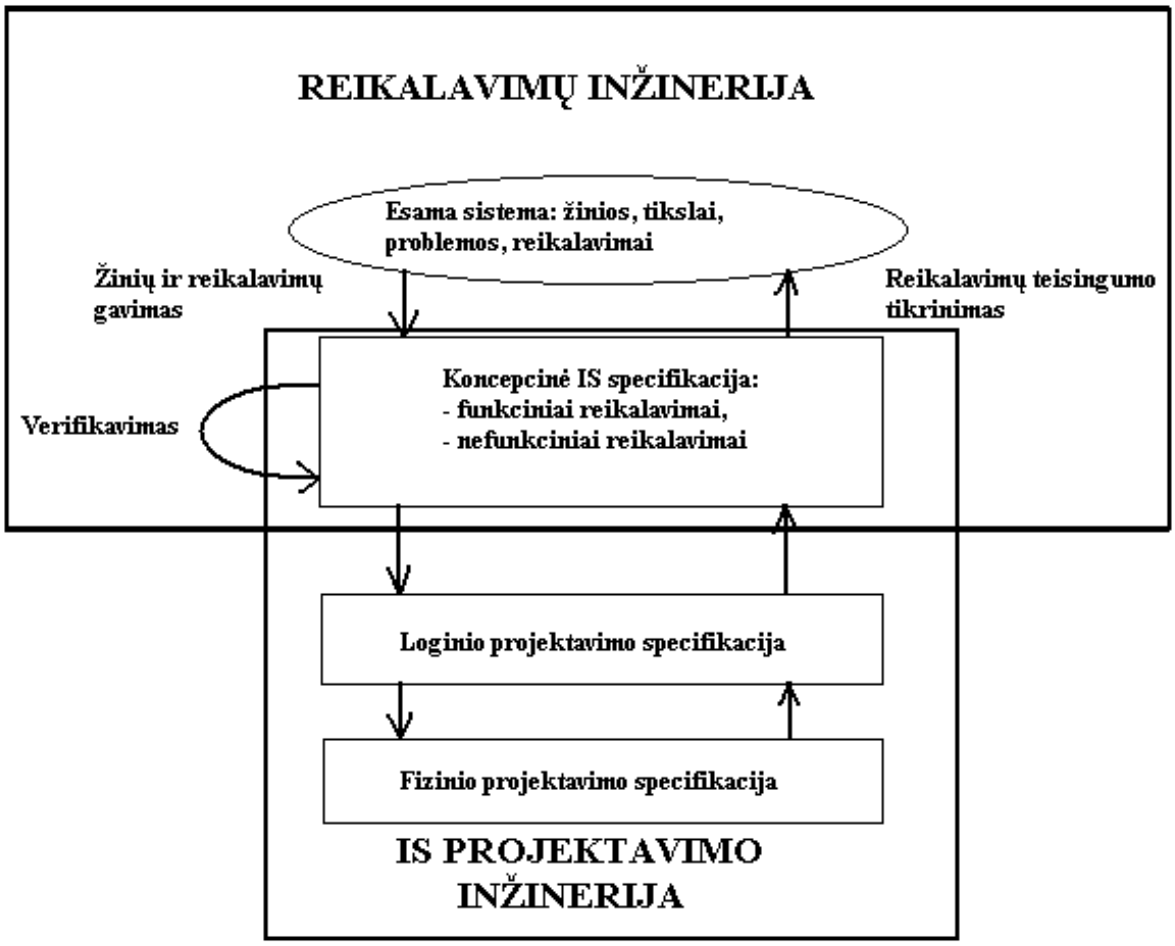
ir tikrinant ar jie tai atliko teisingai. Reikalavimų inžinerijos procesai pavaizduoti 1 paveikslėlyje.[4]



1 Pav. Reikalavimų inžinerijos proceso schema

IS projektavimo inžinerija - tolesnės kūrimo fazės, kuriose naudojama specifikacija, skirta funkcionuojančiai sistemai projektuoti ir diegti, bei sistemos patikrinimui. Ryšiai tarp reikalavimų inžinerijos ir IS projektavimo inžinerijos pateikti 2 paveiksle. Reikalavimų specifikacija suvokiama kaip jungtinis modelis iš submodelių:

- *esamos sistemos submodelis* – parodo organizacijos tikslų struktūrą, problemas susijusias su tikslais, problemų priežastis, egzistuojančias veiklas, procesus, rezultatus, išorinius ryšius ir t.t.
- *funkcinių reikalavimų submodelis* - charakterizuoja informacinių sistemų, palaikančių taikomųjų sričių veiklą, struktūrą ir turinį. Šis modelis apima sąveikų tarp informacinės sistemos ir jos aplinkos svarbiausias savybes, ir tiksliai nusako sistemos logiką, struktūrą ir elgseną.
- *nefunkcinių reikalavimų submodelis* - apibrėžia tikslus, reikalavimus bei funkcinių reikalavimų tipų įgyvendinimų apribojimus. [1]



2 pav. Reikalavimų inžinerijos ir IS projektavimo inžinerijos tarpusavio ryšių schema

IS kūrimo proceso reikalavimų inžinerijos subciklas susideda iš *žinių ir reikalavimų gavimo*, *reikalavimų teisingumo tikrinimo(validavimo) ir verifikavimo*.

Tiek verifikavimo, tiek reikalavimų teisingumo tikrinimo tikslai yra tie patys - patikrinti ar formali ir konceptuali specifikacija yra tinkama, ar ji tinkamai išreiškia funkcinius reikalavimus, kuriuos pateikė vartotojo. Atliekant verifikavimą, yra sprendžiamos sintaksiniame ir semantiniame lygyje esančios problemos:

- Sintaksinio lygmens problema - įrodyti, kad specifikacija yra pilna.
- Semantinio lygmens problema - įvertinti reikšmes, pateiktas specifikacijoje ir ieškoti nesuderinamumų.

Reikalavimų teisingumo tikrinimas yra trijų rūšių:

1. Taisyklėmis grindžiamų sistemų kūrimas, kurių vaidmuo yra įvertinti specifikaciją iš vartotojo pusės.
2. Perfrazuotos ar perrašytos informacijos apie specifikaciją vartotojui gražinimas, kuri būtų pateikta natūralia kalba.
3. Automatinis prototipo sukūrimas iš vykdomųjų specifikacijų.

2.3 RS metodų ir priemonių parinkimo analizė

2.3.1 Formalizmai reikalavimų specifikacijoms atvaizduoti

Reikalavimų specifikavimas – tai reikalavimų užrašymas. Šiam tikslui yra skirtingos notacijos (nuo nestruktūrizuotų ir neformalių tekstų iki aukšto lygio formalių matematinių notacijų).

Žinių išgavimas reikalavimų inžinerijoje yra surinkti ir atvaizduoti srities, kurią palaikys kompiuterizuota informacinė sistema, žinios. Žinių išgavimas yra sprendžiamas pokalbyje su vartotojais.

Sumažinti atotrūkį tarp vartotojo, ir projektuotojo yra gana sudėtinga. Vartotojo reikalavimų geros specifikacijos savybė yra šių reikalavimų išreiškimas jam suprantama notacija be papildomų apmokymų. Todėl kyla klausimas: kokia notacija geriausiai atitinka vartotojo supratimą? Vartotojas gali išreikšti reikalavimus natūralia kalba, tačiau išraiška tokia forma gali neišvengti įvairių prieštaravimų, dubliavimosi ir dviprasmiškumų.

Reikalavimų specifikacijos sudarymui didelę įtaka turi formalizmai, t.y. kokiu būdų buvo bendraujama su klientu, kokia forma užrašyti reikalavimai.

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.

Neformalios – neturi išbaigtų taisyklių rinkinių, kuriais kuriami modeliai būtų suvaržyti(natūrali kalba, nestruktūrizuoti vaizdai).

Pusiau formalios – turi apibrėžtą sintaksę (grafiniai būdai, turintys tikslias taisykles, kurios apibūdina sąlygas, pagal kurias yra leidžiamas konstravimas, tekstiniai, grafiniai aprašymai.

Formalios – turi griežtą apibrėžtą sintaksę ir semantiką (esminis teorinis modelis, pagal kurį gali būti patikrintas aprašymas išreikštas matematine notacija).

Formali specifikacija yra tam tikro savybių rinkinio, kurį turi atitikti sistema, išraiška, pateikta tam tikroje formalioje kalboje ir tam tikrame abstrakcijos lygyje.

Formalūs metodai – tai vienas iš būdų charakterizuoti reikalavimus esant matematiniam tikslumui ir griežtumui. Jie suteikia būdus kurti sistemos modeliams tiksliai nusakant jų struktūrą ir elgseną.

Formalių kalbos yra nedviprasmiškos ir tikslios, ir jas galima taikyti kompiuteriams bei matematinėms analizėms. Tačiau su ja dirbantis personalas turi išmokti atitinkamą kalbą. Formalios specifikacijos sudarymui reikia daugiau laiko.

Algebrinės specifikacijos yra formalus programinės įrangos sistemų specifikavimo būdas. Jo įvairiarūšiuose rinkiniuose pavaizduotos tam tikros operacijos.

Neformalias specifikacijas yra nesudėtingos, jas lengva sudaryti. Tačiau naudojant neformalias specifikacijas yra sudaroma menka sistemų analizė. Ji neužtikrina specifikacijų išbaigtumą, nedviprasmiškumą, bet jos yra plačiai naudojamos, nes yra nesudėtingos ir pigios.

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.

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 yra informatyvus, nes vartotojas ir užsakovas gali pamatyti ir patikrinti, kaip jo reikalavimus suprato inžinierius.

Dauguma specifikacijų yra sudaromos tekstiniu pavidalu. Šitokios specifikacijos leidžia nusakyti daugybę smulkių detalių, tačiau sudėtinga perprasti visos sistemos reikšmę. Aiškesnės specifikacijos yra grafinės, bet jas sudėtingiau praplėsti.

Formalios tekstinės specifikacijos(Z, B, VDM) ir formalios grafinės specifikacijos(Petri tinklai) yra nedviprasmiškesnės negu neformalios tekstinės(tekstas) ir neformalios grafinės(DFD, UML, ER diagramos) specifikacijos.

Praktiškumas didesnis yra grafinės specifikacijos tiek formalių tiek neformalių specifikacijų.

Sudaryti taikomosios srities konceptualią schemą taip pat galima naudojant natūralią kalbą. Teiginiai, apibūdinantys taikomąją sritį, naudojami konceptualiosios schemos kūrimui pažingsniui, o vartotojas apklausiamas dėl papildomai trūkstamos informacijos. Natūrali kalba

leidžia vartotojams lengviau suprasti produkto reikalavimus, tačiau galimas semantikos trūkumas padidina klaidų tikimybę, kurios atsiranda dėl neteisingo suvokimo ir būdingų dviprasmybių.

Natūralios kalbos, grafinės notacijos ir formalių metodų deriniui būdinga:

- Reikalavimų specifikacija – pateikta vartotojams suprantamu formatu;
- Formalūs metodai – naudojami reikalavimų modeliavimui ir valdymui.
- Vartotojui palankios ir formalios reikalavimų versijos - sinchronizuojamos automatiškai;
- Automatizavimas - formalių modelių generavimo iš natūralios kalbos reikalavimų palengvinimas.

2.3.2 Reikalavimų specifikacijos kokybės nustatymas

Reikalavimų specifikacijos kokybės nustatymas priklauso galimybių kokybiškai įvertinti IS pagal charakteristikas, kuriomis pasižymi pati reikalavimų specifikacija. Nustatyti IS pilnumą, išbaigtumą įvertinant įvairiausias „JEIGU..“ sąlygas (pvz.: jeigu analitikas teisingai suprato užsakovo reikalavimus; jeigu užsakovas tikrai nepamiršo pateikti analitikui visus savo reikalavimus; ir pan.) galima, tačiau tas kokybės įvertinimas bus apribotas tam tikromis sąlygomis ir nežinia kaip sistema elgsis, jeigu atsiras nenumatytų išorinių poveikių.

Reikalavimų specifikacijos kokybės įvertinimui ir nustatymui daug įtakos turi tai, kokios bus pasirinktos priemonės, metodai, metrikos IS reikalavimų specifikacijos kokybei pamatuoti. Taip pat daug įtakos turi formalizmai, t.y. priklausomai nuo to, kokia pasirenkama analitiko ir reikalavimų užsakovo bendravimo forma ir kaip aiškiai jie vienas kita supranta. Labai svarbu yra pasirinkti tinkamą metodą reikalavimų specifikacijoms sudaryti.

2.3.3 Specifikavimo kalbų kokybės vertinimo metodų analizė

Šiame skyriuje analizuojami egzistuojantys specifikavimo kalbų kokybės vertinimo metodai.

Bunge ontologija grindžiamas metodas

Filosofas Mario Bunge pasiulė [Bun77] ontologiją, kuri gali būti panaudota kaip “standartinė” (normatyvinė) ontologija. Darbuose [WW89a], [WW95] buvo pasiulyta kaip pritaikyti Bunge ontologiją specifikavimo kalboms bei jomis parašytoms specifikacijoms vertinti. Šie pasiūlymai vadinami Bunge, Wand ir Weber (BWW) modelių sistema. BWW modelis paskelbiamas norma, visos kitos specifikavimo kalbos vertinamos, lyginant, kiek jos tenkina tą normą. [6]

Bunge ontologija grindžiamo metodo plėtinys

Vienas iš pasiūlymų, kaip išplėsti BWW vaizdavimo modelio kategorijų sistemą, pateiktas darbe [Opd97], kuriame, remiantis radikaliojo konstruktyvizmo filosofijos prielaidomis, yra teigiama, kad neįmanoma atskirti, kokiu mastu analitiko turima socialinės realybės samprata yra objektyvi. Todėl darbe pasiulyta papildyti Bunge ontologiją dviem naujomis kategorijomis – perspektyva ir koncepcija. Pagrindinis Bunge ontologija grindžiamų metodų trūkumas yra tas, kad palyginimas su norma negali būti laikomas objektyviu. Eksperimentiniai specifikavimo kalbų tyrimai parodė, kad sukonstruoti tokią normą, kurioje būtų numatytos absoliučiai visos galimos specifikavimo kalbų konstrukcijos yra neimanoma. Be to, palyginant su norma visiškai neatsižvelgiama į specifinius konkrečcios sistemos poreikius. [6]

Chisholmo ontologija grindžiamas metodas

Darbe [MKK98] pasiulyta specifikavimo kalbas vertinti ne Bunge, bet Chisholmo ontologijos [Ch96] pagrindu. Du analitikai tos pačios specifikavimo kalbos priemonėmis gali parašyti skirtingas specifikacijas. Vienam iš jų prireiks vienų priemonių, kitam – kitų. Todėl, autorių nuomone, kalbas reikia vertinti ne per jų siūlomų konstrukcijų prizmę, bet nagrinėjant kokiu mastu kalbos konstrukcijomis galima specifikuoti socialinės realybės situacijas. Tam siūloma naudoti suderinamumų (angl. agreement) ir skirtumų (angl. difference) nustatymo metodus. Taikant šiuos metodus, analizuojama kokie specifikavimo kalbos suderinamumai ir kokie jos skirtumai yra stebimi, lyginant ją su Chisholmo ontologija. Ar koks nors kokybinis bruožas yra būdingas vertinamai specifikavimo kalbai, nustatoma naudojant kokybinę darnos/skirtumų penkių lygmenų skalę. Lyginant su BWW modeliu, šis metodas geriau atsižvelgia į specifinius konkrečios sistemos specifikavimo poreikius, bet vis vien yra siekiama sukonstruoti "patį geriausią" (etaloninį) tikrovės konceptualizavimo būdą. [6]

Ontologinių aspektų vertinimo metodas

Normatyvinių ontologijų trūkumus bandoma pašalinti darbe [Myl98], kuriame atsisakyta požiurio, kad būtina vadovautis kokia nors normatyvine ontologija. Siūloma specifikavimo kalbas vertinti atsižvelgiant į tai, kokiomis ontologijomis jos yra grindžiamos, kokie abstrakcijos mechanizmai jose yra palaikomi ir kokias instrumentines priemones joms galima sukurti. Naudojant autoriaus siūlomas klasifikavimo schemas, visus tris aspektus galima įvertinti pagal pasirinktą skalę. Bendras specifikavimo kalbos įvertis gaunamas atitinkamai kombinuojant jos aspektų įverčius. Kiekvienos dimensijos įvertinimas gaunamas sudėjus jos komponentų įverčius.

Ontologinių aspektų vertinimo metodas taip pat turi trūkumų. Visų pirma neaišku, ar siūlomas ontologijų rinkinys iš tiesų yra pakankamas visiems socialinės realybės aspektams

konceptualizuoti. Antra, visos vienos ontologijos kategorijų sistemos laikomos lygiavertėmis, o taip nėra. Trečia, įverčiai gaunami subjektyviai, neatliekant kokių nors tikslesnių matavimų, o pati skalė yra negriežta, kokybinė. [6]

Kokybės modeliu grindžiamas metodas

Mokslinėje literaturoje pasiūlyta ir kitokių, ne tik ontologinės analizės principais grindžiamų, specifikavimo kalbų vertinimo metodų. Tipiškas tokio metodo pavyzdys yra semiotinis karkasas [LSS94], [Kr03]. Metodo esmė – sukonstruoti analizuojamų specifikavimo kalbų kokybės charakteristikų hierarchiją, vadinamąjį kokybės modelį, ir vertinti tuo modeliu numatytas charakteristikas. Charakteristikoms ivertinti konstruojama kiekybinių arba kokybinių matų sistema. [6]

Semiotinis karkasas labiausiai priartėja prie tikslo sukurti objektyvų specifikavimo kalbų vertinimo metodą. Tačiau pasiūlytasis kokybės modelis nėra iki galo išbaigtas, neaišku kaip matuoti ir vertinti kokybės charakteristikas ir kaip vertinant kalbos kokybę atsižvelgti į konkretaus projekto poreikius. Darbe yra pasiūlyta egzistuojančių specifikavimo kalbų kokybės vertinimo metodų lyginamosios analizės metodika, kurioje koncepcinės analizės metodas suderintas su suderinamumų ir skirtumų nustatymo metodais. Koncepcinės analizės metodo paskirtis – išskirti pagrindinius egzistuojančių specifikavimo kalbų kokybės vertinimo metodų komponentus. Suderinamumų ir skirtumų nustatymo metodas naudojamas palyginimo lentelei užpildyti ir palyginti komponentus nustatant jų panašumus ir skirtumus.

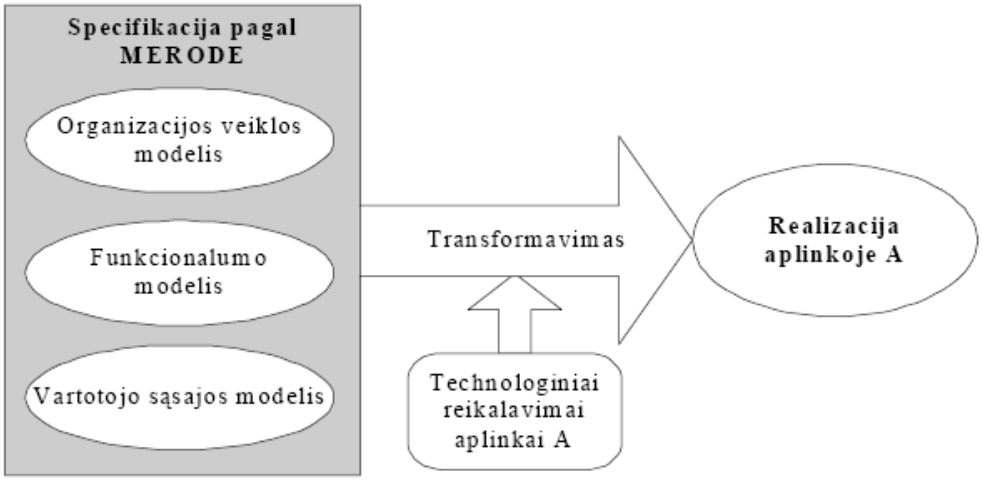
Atsižvelgus į lyginamosios analizės rezultatus galima teigti, kad specifikavimo kalbos kokybės vertinimo būdas turėtų būti konstruojamas derinant ontologinių aspektų vertinimo metodo ir kokybės modeliu grindžiamo metodo privalumus.

2.3.4 RS metodų analizė

MERODE metodas

MERODE (*Model–based Existence–dependency Relationship Object–oriented Development*) – tai objektiškai orientuotas metodas, pagrįstas egzistavimo priklausomybės sąryšiu. Šio metodo pagrindas yra egzistavimo priklausomybės koncepcija. Metodo kūrėjai siekė, kad jį naudojant būtų užtikrintas modeliavimo išbaigtumas, lankstumas bei paprastumas. Be to, buvo siekiama, kad modeliavimo metu būtų galima naudoti ir formalų modelių aprašymą, toliau leidžiantį atlikti gaunamų rezultatų kontrolę. Kita šiam metodui būdinga savybė – aiškiai matomas skirtumas tarp reikalavimų specifikavimo ir specifikacijos realizavimo. Sistema, specifikuota taikant šį metodą, gali būti realizuota įvairiais būdais, taikant bet kurią galimą

sistemos kūrimo aplinką. Tai gali būti objektiškai orientuota arba tradicinė kūrimo aplinka (žr. 3 pav.). Pabrėžtina, kad realizavimas atliekamas transformuojant, o ne nuosekliai plėtojant specifikaciją. MERODE metodas skiriasi nuo kitų objektiškai orientuotų metodų tuo, kad garantuoja pakankamai didelį nepriklausomumo laipsnį tarp organizacijos veiklos modelio ir funkcionalumo modelio, o taip pat tarp komponentų, esančių šių modelių viduje. Šios savybės pagerina kuriamos sistemos lankstumą ir išplečiamumą. [3]



3 pav. Bendra MERODE metodo sandaros ir panaudojimo schema

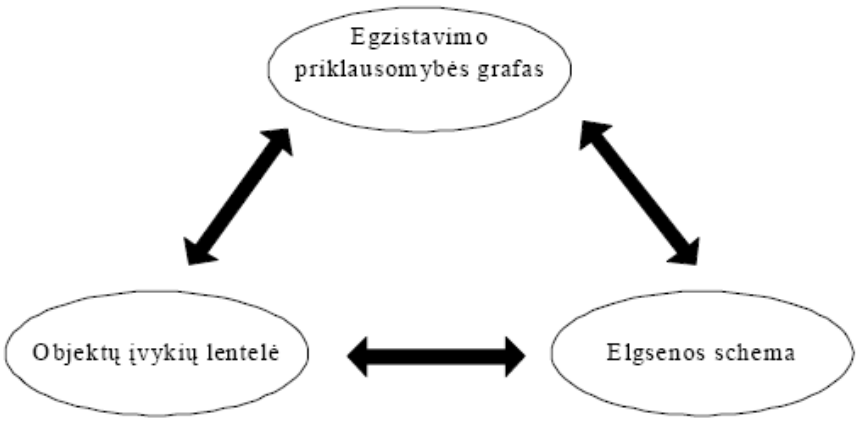
MERODE metodo pagalba modeliuojant organizacijos veiklą yra sudaromos trys schemos:

- Egzistavimo priklausomybės grafas;
- Objektų–įvykių lentelė;
- Elgsenos schema.

Kiekviena iš schemų aprašoma formaliai. Turint kiekvienos iš jų formalų aprašymą, galima atlikti organizacijos modelio neprieštaringumo patikrinimą, kitaip tariant, kokybės kontrolę (žr. 4 pav.). Organizacijos veiklos modeliavimo kokybė turi didžiausią įtaką realizuojamos sistemos kokybei.

MERODE metode specifikacijos kokybė apibrėžiama, kaip vidinis modelio neprieštaringumas ir korektiškumas.

Egzistavimo priklausomybės grafas, kuris yra metodo pagrindinė konstrukcija, naudojamas kaip pirminis objektas atliekant organizacijos modelio kokybės kontrolę. Metode nurodoma, kaip užtikrinti objektų įvykių lentelės ir elgsenos schemos neprieštaringumą egzistavimo priklausomybės grafui. Papildomai atliekamas ir elgsenos schemos bei objektų įvykių lentelės tarpusavio neprieštaringumo patikrinimas. Tai leidžia gerokai padidinti organizacijos modelio kokybę, nei tuo atveju, kai apsiribojama schemų sintaksės korektiškumo tikrinimu.[3]



4 pav. Trijų organizacijos modelio schemų tarpusavio neprieštaringumo patikrinimas

MERODE metodo specifikacijoje apibrėžiami reikalavimai modeliuose naudojamų sąvokų sintaksei bei semantikai, kad būtų galima verifikuoti modelių neprieštaringumą ir korektiškumą:

1. **Sintaksė:** metodologijos sintaksė turi būti tiksliai apibrėžta.

2. **Semantika:** kiekvieną metodologijos sąvoką turi lydėti formalus aprašas; sąvokos reikšmė negali būti aprašyta natūralia kalba.

3. **Neprieštaringumas tarp schemų:** turi būti apibrėžti formalūs procesai schemų tarpusavio neprieštaringumui patikrinti.

4. **Visos sistemos elgsena:** turi būti priemonės numatyti sistemos elgseną, integruojant atskirų objektų apibrėžtą sąveiką ir atliekamus veiksmus.

5. **Anomali sistemos elgsena:** jei visos sistemos veikla yra specifikuota, turi būti galimybė patikrinti sistemos elgseną, esant tam tikroms kritinėms situacijoms.

Pagrindinis MERODE pranašumas yra organizacijos veiklos modeliavimo metu atliekama kokybės kontrolė.[3]

ODRES metodas

Informacijos sistemų katedroje plėtojamas informacinės sistemos funkcinių reikalavimų specifikavimo metodas (ODRES – Output Driven Requirements Specification). Išskiriami septyni nuoseklūs etapai:

- konteksto išskyrimas;
- funkcionavimo rezultatų specifikavimas;
- duomenų šaltinių specifikavimas;
- duomenų šaltinių apdorojimo proceso specifikavimas;
- modeliavimas;
- automatizuotas informacines sistemos vartotojo sąsajos kūrimas;
- prototipo kūrimas.

Tai yra CASE metodas ir jam palaikyti yra kuriamas CASE įrankis.

Specifikuojant reikalavimus analitikas pildo reikalavimų ataskaitas, tačiau nežino kokią dalį jo įvesta informacija sudaro visos KIS reikalavimų specifikacijos, ar ji yra pilna ir kiek liko įvesti informacijos į sistemą, kad ji būtų pilna ar bent jau pakankama.

Analitikas kiekvienu etapu užpildęs tam tikrą dalį KIS reikalavimų specifikacijos, nėra užtikrintas, kad rengdamas reikalavimų specifikaciją, jis suvedė visą reikalingą informaciją. Norint patikrinti ar nėra nepalikta spragų specifikacijose, analitikas turi atlikti specifikacijų pilnumo patikrinimą pats, todėl norint palengvinti vartotojo darbą turi būti sukurtas CASE įrankis, kuris patikrintų KIS reikalavimų specifikacijos pilnumą ir pateiktų vartotojui informaciją, apie trūkstamą informaciją.

Gauti pilną kuriamos IS reikalavimų specifikacijos modelį yra labai sunku, nes informacija apie dalykinę sritį paprastai yra pateikiama neformalioje notacijoje (žodiniai KIS vartotojų komentarai, speciali literatūra, dokumentai ir pan.), o formalizuotomis priemonėmis patikrinti, ar ją atitinka jos pagrindu sudaryta formalizuota specifikacija, galima tik iš dalies. Pavyzdžiui, nėra formalių priemonių patikrinti, ar būsimasis KIS vartotojas nepamiršo pasakyti, kokią dar svarbią informaciją be jau paminėtos. Sudarytų modelių kokybė išsamumo atžvilgiu didžia dalimi priklauso nuo analitiko sugebėjimo išgauti reikiamą informaciją apie dalykinę sritį. Kadangi modelio išsamumo neįmanoma patikrinti vien tik formalizuotomis priemonėmis, tai kiekvienas modelio patikrinimas šiuo atžvilgiu negarantuoja, kad modelis yra tikrai išsamus, o tik padidinamas tikrumas, jog sudarytas modelis yra išsamus. Tačiau ir dalinis sudarytų modelių išsamumo patikrinimas formalizuotomis priemonėmis yra naudingas, nes tokias priemones galima automatizuoti ir tokiu būdu palengvinti analitiko darbą.[5]

Siekiant suteikti vartotojui informaciją apie šiuo metu kompiuterizuotos IS reikalavimų specifikacijos būklę, turi būti sukurtas CASE įrankio komponentas, kuris informuotų vartotoją apie dabartinę kompiuterizuotos IS reikalavimų specifikacijos kokybę.

ODRES metodas yra skirtas surinkti vartotojo pateiktą informacija ir reikalavimus, skirtus veikiančiai sistemai sudaryti. ODRES metodo pagrindu surinkta informacija iš vartotojo turi būti minimaliai pakankama, kad būtų galima sudaryti pakankamas specifikacijas veikiančiam prototipui gauti.

Pateiksime taisykles, kurios turi būti patenkintos, norint gauti pakankamai išsamų ir minimalų KIS modelį realizavimo atžvilgiu, pritaikius ODRES metodą.

Matematinis modelis

Kompiuterizuojamos IS funkcionalumo rezultatams identifikuoti naudojamas modelis yra trejetas

$M_2=<M_1, O, FO>$,

čia M_1 – modelis kontekstui išskirti;

O – kompiuterizuotinų IS funkcionalumo rezultatų aibė,

FO $\subseteq$ F_N x O – sąryšis, apibrėžiantis, kokie yra kompiuterizuotini tam tikrų funkcijų funkcionalumo rezultatai.

KIS funkcionalumo rezultatams ir funkcijoms susieti naudojamų aibių ir predikatų atitikmenys

Aibė, sąryšis, atvaizdas	Predikatas
FO	Functionality_result

Apribojimai

Kiekvienai kompiuterizuojamai nedetalizuotai funkcijai ***f*** turi būti specifikuotas nors vienas jos funkcionavimo rezultatas ***o***:

$\forall f \in F/(F_{TD} \cup F_{ND}) \; \exists \; o \in O$ [functionality_rezult(*o*,*f*)]

Kiekvienas rezultatas ***o*** turi būti bent vienos nedetalizuotos funkcijos ***f*** funkcionavimo rezultatu:

$\forall o \in O \; \exists \; f \in F/(F_{TD} \cup F_{ND})$ [functionality_rezult(*o*,*f*)]

Specifikavimo procesas

KIS funkcionalumo rezultatų identifikavimo ir jų susiejimo su tam tikromis nedetalizuotomis funkcijomis procesas susideda iš dviejų veiksmų: rezultato specifikavimo ir rezultato pašalinimo. Procesas visada prasideda nuo rezultato specifikavimo. Jei yra specifikuotas bent vienas rezultatas, tai galima jį pašalinti arba specifikuoti kitą rezultatą. Jei pašalinami visi specifikuoti rezultatai, tai galima atlikti tik rezultato specifikavimą.

KIS funkcionalumo rezultatus kuriantiems veiksmas specifikuoti naudojamų aibių ir predikatų atitikmenys

Aibė, sąryšis, atvaizdas	Predikatas
OP	from_output
AP_O	agent
PA_O	recepient
AA	a_compose
Z_A	a_is_a

Apribojimai

Kiekvienam specifikuotam rezultatui o turi būti specifikuotas bent vienas jį formuojantis veiksmas p:

$\forall o \in O \; \exists p \in P_O$ [from_output(p,o)]

Kiekvienas specifikuotas rezultatą formuojantis veiksmas p turi formuoti bent vieną rezultatą o:

$\forall p \in P_O \; \exists o \in O$ [from_output(p,o)]

Kiekvienam specifikuotam rezultatą formuojančiam veiksmui p specifikuojamas tik vienas šio veiksmo vykdytojas - agentas a:

$\forall p \in P_O \; \exists! a \in A$ [agent(p,a)]

Kiekvienam specifikuotam rezultatą formuojančiam veiksmui p specifikuojamas tik vienas šio veiksmo rezultato gavėjas a:

$\forall p \in P_O \; \exists! a \in A$ [recipient(p,a)]

Kiekvienai specifikuotai aktorių grupei a_g reikia specifikuoti bent du ją sudarančius aktorius ar jų grupes a' ir a“:

$\forall a_g \in A_G \; \exists a', a'' \in A$ [a_compose(a_g,a‘) $\wedge$ a_compose(a_g,a“)]

Kiekvienas specifikuotas aktorius a turi būti bent vieno veiksmo p agentu arba gavėju:

$\forall a \in A \; \exists p \in P$ [agent(p,a) $\vee$ recipient(p,a)]

Kiekvienam vienišam aktoriui a_a turi neegzistuoti aktorius-grupė a_g, kuris būtų aktoriaus a_a specializacija:

$\forall a_a \in A_A \neg \exists a_a \in A_A$ [a_is_a(a_a,a_g)]

Kiekvienam aktoriui-grupei a_g turi neegzistuoti vienišas aktorius a_a, kuris būtų aktoriaus a_g specializacija:

$\forall a_g \in A_G \neg \exists a_a \in A_A$ [a_is_a(a_g,a_a)]

Duomenų šaltinių struktūrai specifikuoti naudojamų aibių ir predikatų atitikmenys

Aibė, sąryšis, atvaizdas	Predikatas
Δ_I	describe_i
IC	condition_i
C_{LI}	limit_i

Apribojimai

Kiekvienam specifikuotam duomenų šaltiniui *i* reikia specifikuoti nors vieną jį aprašantį atributą t:

$\forall i \in I \exists t \in (T_T \cup TT_T)[describe\_i(i,t)]$

Kiekvienam duomenų šaltiniui *i* gali būti specifikuotas tik vienas (arba nei vieno) duomenų šaltinio tipą nusakantis elementas l_y:

$$\forall i \in I \exists! l_y \in L_{Y_I}[describe\_i(i,l_y)] \vee \neg \exists l_y \in L_{Y_I}[describe\_i(i,l_y)]$$

Kiekvienam duomenų šaltinio *i* tipą apibudinančiam elementui l_v turi būti specifikuotas tik vienas duomenų šaltinis *i*:

$$\forall l_y \in L_{Y_I} \exists! i \in I[describe\_i(i,l_y)]$$

Kiekvienam duomenų šaltiniui *i*, turinčiam jo tipą apibudinantį elementą l_y, gali būti specifikuotas bent vienas (arba nei vienas) duomenų šaltinio tipo potipį nusakantis elementas l_s:

$$\forall i \in I \forall l_y \in L_{Y_I}[describe\_i(i,l_y)] \Rightarrow \exists l_s \in L_{S_I}[describe\_i(i,l_s)] \vee \neg \exists l_s \in L_{S_I}[describe\_i(i,l_s)]$$

Kiekvienam duomenų šaltinio elementui l_b, nusakančiam duomenų šaltinio tipo potipį, turi būti specifikuotas vienintelis duomenų šaltinis *i* ir duomenų šaltinio *i* tipą apibūdinantis elementas l_y;

$$\forall l_b \in L_{B_I} \exists! i \in I \exists l_y \in L_{Y_I}[describe\_i(i,l_b) \wedge describe\_i(i,l_y)]$$

Kiekvienam duomenų šaltiniui *i* gali būti specifikuota tik viena (arba nei vienos) duomenų šaltinio *i* suformavimą sąlygojanti sąlyga c_i:

$$\forall i \in I \exists! c \in C_I[cindition\_i(i,c)] \vee \neg \exists c \in C_I[cindition\_i(i,c)]$$

Kiekvienai sąlygai c turi būti specifikuotas tik vienas duomenų šaltinis *i*, kurio suformavimą ir sąlygoja sąlyga c:

$$\forall c \in C_I \exists i \in I[condition\_i(i,c)]$$

Kiekvienam duomenų šaltiniui *i* gali būti specifikuotas bent vienas (arba nei vieno) duomenų šaltinio *i* atributų reikšmes ribojantis elementas l_v:

$$\forall i \in I \exists l_v \in L_{V_I}[desribe\_i(i,l_v)] \vee \neg \exists l_v \in L_{V_I}[describe\_i(i,l_v)]$$

Perduodamų duomenų srautų struktūrai specifikuoti naudojamų aibių ir predikatų atitikmenys

Aibė, sąryšis, atvaizdas	Predikatas
D	use_attribute

Apribojimai

Kiekvienam rezultatui o ir duomenų šaltiniui *i*, kuris yra naudojamas rezultatui o suformuoti, turi būti specifikuotas bent vienas neišvestinis atributas *t'*, kuris aprašo rezultatą o, ir atributas *t''*, kuris aprašo duomenų šaltinį *i* ir yra naudojamas atributui *t'* suformuoti:

$$\forall o \in O \forall i \in I[use\_data\_resourse\_o(o,i) \Rightarrow \exists t' \in L''_O \exists t'' \in L_{T_I}[describe\_o(o,t') \wedge describe\_i(i,t'') \wedge use\_attribute(o,t',i,t'')]]$$

Kiekvienam duomenų šaltiniui *i'* ir duomenų šaltiniui *i''*, kuris yra naudojamas duomenų šaltiniui *i'* suformuoti, turi būti specifikuotas bent vienas neišvestinis atributas t', kuris aprašo

duomenų šaltinį *i"*, ir atributas t", kuris aprašo duomenų šaltinį *i*" ir yra naudojamas atributui t' suformuoti:

$\forall i', i'' \in I[use\_data\_resourse\_i(i', i'') \Rightarrow \exists t' \in L''_I \exists t'' \in L_{T_I}[describe\_i(i,t') \wedge describe\_i(i,t'') \wedge use\_atribute(i,t',i,t'')]]$

Kiekvienam neišvestiniam rezultato o atributui t‘ turi būti specifikuotas bent vienas duomenų šaltinio *i* atributas t“ kuris yra naudojamas rezultato o atributui t‘ suformuoti:

$\forall o \in O \forall t' \in L''_O[describe\_o(o,t') \Rightarrow \exists i \in I \exists t'' \in L_{T_I}[describe\_i(i,t'') \wedge use\_atribute(o,t',i,t'')]]$

Pageidautina, kad kiekvienas duomenų šaltinį *i*' aprašantis atributas t' turi būti naudojamas arba rezultato o neišvestiniam atributui t" suformuoti arba kito duomenų šaltinio *i*" neišvestiniam atributui t'" suformuoti:

$\forall i \in I \forall t' \in L_{T_I}[describe\_i(i,t') \Rightarrow \exists o \in O \exists t'' \in L''_O[describe\_o(o,t'')] \wedge use\_atribute(o,t'',i,t')] \vee \exists i \in I \exists t''' \in L''_I[desribe\_i(i,t''')] \wedge use\_atribute(i,t''',i,t')]]$

2.3.4 ODRES metode taikomos išvedimo taisyklės

Jei kiekvienos funkcijos ***f‘***, apibendrinančios tam tikrą funkciją ***f“***, funkcionalumo rezultatas yra ***o***, tai ir funkcijos ***f“*** funkcionalumo rezultatas yra ***o***:

$\forall$ f‘, f“$\in F \forall o \in O$ [[f_compose(f‘,f“) $\wedge$ functionality_result(o,f‘)] $\Rightarrow$ functionality_result(o,f“)]

Jei kiekvienas aktorius a‘, apibendrinantis tam tikrą aktorių a“ yra veiksmo p agentu, tai ir aktorius a" yra veiksmo p agentas:

$\forall a', a'' \in A \ \forall p \in P$ [[a_is_a(a‘,a“) $\wedge$ agent(p,a“)] $\Rightarrow$ agent(p,a“)]

Jei kiekvienas aktorius a‘ apibendrinantis tam tikrą aktorių a", yra veiksmo p gavėju, tai ir aktorius a“ yra veiksmo p gavėjas:

$\forall a', a'' \in A \ \forall p \in P$ [[a_is_a(a‘,a“) $\wedge$ recipient(p,a“)] $\Rightarrow$ recipient(p,a“)]

Kiekvienas specifikuotas atributas t turi aprašyti bent vieną funkcionalumo rezultatą o arba duomenų šaltinį i:

$\forall t \in T \exists o \in O$ [describe_o(o,t) V describe_i(i,t)]

Kiekvienam specifikuotam rezultatui o reikia specifikuoti nors vieną jį aprašantį atributą t:

$\forall o \in O \exists t \in (T_T \cup TT_T)$ [describe_o(o,t)]

Kiekvienam rezultatui o gali būti specifikuotas tik vienas (arba nei vieno) rezultato tipą nusakantis elementas l_y.:

$\forall o \in O \exists! l_y \in L_{Y_O}$ [describe_o(o,l_y)] V $\neg \exists l_y \in L_{Y_O}$ [describe_o(o,l_y)]

Kiekvienam rezultato tipą apibūdinančiam elementui l_y turi būti specifikuotas tik vienas rezultatas o:

$\forall\, l_y \in L_{Y_O}\ \exists! o \in O$ [describe_o(o,l_y)]

Kiekvienam rezultatui o, turinčiam jo tipą apibūdinantį elementą l_y, gali būti specifikuotas bent vienas (arba nei vieno) rezultato tipo potipį nusakantis elementas l_s:

$\forall\, o \in O\ \forall\, l_y \in L_{Y_O}$ [describe_o(o,l_s)$\Rightarrow\ \exists\, l_s \in L_{S_O}$ [describe_o(o,l_s)] $\vee$

$\neg\, \exists\, l_s \in L_{S_O}$ [describe_o(o,l_s)]

Kiekvienam rezultato elementui l_s, nusakančiam rezultato tipo potipį, turi būti specifikuotas vienintelis rezultatas o ir rezultato o tipą apibūdinantis elementas l_y:

$\forall\, l_s \in L_{S_O} \exists! o \in O \exists\, l_y \in L_{Y_O} \left[describe\_o(o,l_s) \wedge desribe\_o(o,l_y)\right]$

Kiekvienam rezultatui o gali būti specifikuota tik viena (arba nei vienos) rezultato suformavimą sąlygojanti sąlyga c:

$\forall\, o \in O \exists! c \in C_O [condition\_o(o,c)] \vee \neg\, \exists\, c \in C_O [condition\_o(o,c)]$

Kiekvienai sąlygai c turi būti specifikuotas tik vienas rezultatas o, kurio suformavimą ir sąlygoja sąlyga c:

$\forall\, c \in C_O \exists\, o \in O [condition\_o(o,c)]$

Kiekvienam rezultatui o gali būti specifikuotas bent vienas (arba nei vieno) rezultato atributų reikšmes ribojantis elementas l_v:

$\forall\, o \in O \exists\, l_v \in L_{V_O} [describe\_o(o,l_v)] \vee \neg\, \exists\, l_v \in L_{V_O} [desribe\_o(o,l_v)]$

Kiekvienai rezultato o sąlygai c turi būti nurodytas bent vienas rezultatą aprašantis atributas t', kurio reikšmės bus šios sąlygos ribojamos, ir bent vienas rezultatą o aprašantis atributas t'', kurio reikšmės ribos atributo t' reikšmes, arba bent vienas rezultatui o specifikuotas atributo reikšmes ribojantis elementas l_v, kuris ribos atributo t' reikšmes:

$\forall\, o \in O \forall\, c \in C_O [condition(o,c) \Rightarrow\ \exists\, t', t'' \in (T_T \cup TT_T) [describe\_o(o,t') \wedge describe\_o(o,t'') \wedge$
$\lim it\_o(c,t',t'')] \vee \exists\, t' \in (T_T \cup TT_T) \exists\, l_v \in L_{V_O} [describe\_o(o,t') \wedge$
$describe\_o(o,l_v) \wedge \lim it\_o(c,t',l_v)]]$

Kiekvienam rezultato atributų reikšmes ribojančiam elementui l_v, turi būti specifikuotas vienintelis rezultatas o:

$\forall\, l_v \in L_{V_O} \exists! o \in O [describe\_o(o,l_v)]$

Kiekvienam rezultato o atributų reikšmes ribojančiam elementui l_v turi būti specifikuotas bent vienas rezultatą aprašantis atributas t, kurio reikšmes riboja elementas l_v rezultato o sąlygoje c:

$$\forall l_v \in L_{V_O} \forall o \in O[describe\_o(o,l_v)] \Rightarrow \exists t \in (T_T \cup TT_T) \exists c \in C_O[describe\_o(o,t) \wedge condition\_o(o,c) \wedge \lim it\_o(c,t,l_v)]]$$

Jei kiekvienam neišvestiniam atributui t', kuris aprašo rezultatam, suformuoti yra naudojamas atributas t", kuris aprašo duomenų šaltinį *i*, tai seka, kad rezultatui o suformuoti yra naudojamas duomenų šaltinis *i:*

$$\forall t' \in L"_O \forall o \in O \forall t" \in L_{T_I} \forall i \in I[[describe\_o(o,t') \wedge describe\_i(i,t") \wedge use\_atribute(o,t',i,t")] \Rightarrow use\_data\_resource\_o(o,i)]$$

Jei kiekvienam neišvestiniam atributui t', kuris aprašo duomenų šaltinį *i*', suformuoti yra naudojamas atributas t", kuris aprašo duomenų šaltinį *i*", tai seka, kad duomenų šaltiniui *i*' suformuoti yra naudojamas duomenų šaltinis *i*":

$$\forall t' \in L"_{Ot} \forall t" \in L_{T_I} \forall i', i" \in I[[describe\_i(i',t') \wedge describe\_i(i",t") \wedge use\_attribute(i',t',i",t")] \Rightarrow use\_data\_resourse\_i(i',i")]$$

2.3.5 Metrikos

Defektų duomenų analizė yra atliekama naudojant metrikas – kiekybinius matus, parodančius, kiek sistema ar jos komponentas atitinka nustatytus atributus. Projektuose gali būti naudojamos standartinės programinės įrangos defektų metrikos (pavyzdžiui, defektų tankis – defektų skaičiaus ir kodo eilučių kiekio santykis) arba sukurti individualūs, pritaikyti projektui arba įmonės procesui matavimai.

Metrikos parodo sistemos pilnumą, išsamumą.

2.3.6 CASE įrankiai ir jų galimybės įvertinti specifikacijos kokybę

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.

Specifikacijos kokybę taip pat galima įvertinti CASE sistemos ProVisionWorkbench (automatinis ataskaitų generavimas) arba „MagicDraw UML“ įrankiais (metrikos), kurie įvertina sistemos modelio pilnumą.

ProVisionWorkbench – tai organizacijos modeliavimo priemonė, kuri apjungia BPR ir OO A&D į vieną, integruotą modeliavimo priemonę. Šis įrankis gali automatiškai sugeneruoti IS testavimą ir tai didžiulis privalumas palyginus su rankinių klaidų ir neatitikimų paieškos projekto modeliuose būdu. ProVisionWorkbench IS testavimo režimai:

- Rašybos testavimas (Spell checker);
- Modelių pilnumo tikrinimas (Completeness checker);
- Modelių sudėties palyginimas.

MagicDraw yra daugiavartotojiškas CASE įrankis, kuris realizuoja UML. Sukurtas verslo analitikams, programinės įrangos analitikams, programuotojams, kokybės užtikrinimo inžinieriams ir dokumentuotojams.

Tai dinamiškas ir įvairiapusis IS kūrimo įrankis. Jis palengvina objektiškai orientuotų sistemų ir duomenų bazių analizę ir modeliavimą, suteikia priemones programos kodo inžinerijai (su visapusišku Java, C++, C# and CORBA IDL programavimo kalbų palaikymu). Taip pat realizuoja duomenų bazių specifikacijų DDL generavimą ir IS reinžineriją. [7]

MagicDraw yra CASE įrankis, kuris turi metrikas. Panaudojant metrikas galima įvertinti IS kobybę, t.y. pilnumą-pakankamumą, patikrinti ar specifikacijose neliko neaprašytų atributų ir t.t.

1 lentelė. CASE įrankių galimybių palyginimas

CASE įrankių galimybės	**Rational Rose**	**ProVisionWorkbench**	**MagicDraw**
Automatinis ataskaitų generavimas		+	+
Defektų duomenų analizė (metrikos)			+
Rašybos testavimas (Spell checker)		+	
Modelių pilnumo tikrinimas (Completeness checker)	+	+	+
Modelių sudėties palyginimas		+	
Priemonių programos kodo inžinerijai suteikimo galimybė			+
Duomenų bazių specifikacijų DDL generavimas	+		+

2.4 Projekto tikslas

Informacinės sistemos kokybės įvertinimas yra procesas, kurio metu reikia įvertinti daug faktorių apie vartotojo pateiktus reikalavimus, kad tiksliai suprasti ir išsiaiškinti ką vartotojas nori gauti iš sistemos. Todėl šio projekto tikslas yra sukurti CASE įrankį, kuris automatiškai įvertintų IS reikalavimų specifikacijos tam tikras kokybines charakteristikas, patikrintų jos pakankamumą ir pateiktų analitikui informaciją apie IS reikalavimų specifikacijos pilnumą grafine ir tekstine forma. Šis įrankis turi informuoti analitiką kiek reikalavimų specifikacijos yra pilnos, ar jų pakanka, kad sistema galėtų pradėti dirbti, o gal dar trūksta informacijos, kuria turi pateikti vartotojas.

Norint sukurti reikalavimų specifikacijos kokybiškumo charakteristikas vertinantį įrankį, reikia atlikti metodų ir priemonių, kuriais remiantis bus kuriamas šis įrankis, analizę.

2.5 Reikalavimų specifikacijos kokybės vertinimas

Specifikacijos yra didelės, todėl reikia automatizuoti reikalavimų specifikacijų kokybinių charakteristikų patikrinimą. Tačiau ne visas charakteristikas galima automatiškai patikrinti, nes kai kurios charakteristikos tik sąlyginai gali būti tiksliai įvertintos. Šiame projekte bus automatizuotas reikalavimų specifikacijų pilnumo įvertinimas.

Reikalavimų specifikacijoje turėtų būti nurodyta kokius funkcionalumus ir savybes sistema turi atitikti, kaip šiuos reikalavimus vykdyti. Tačiau, reikalavimai neturi nulemti projektavimo sprendimų, o turi suteikti projektuotojui laisvę pasirinkti bet kokį algoritmą, kuris įvykdys reikalavimą. [2]

Kokybės atžvilgiu, reikalavimų specifikacijos turi atitikti charakteristikas:

2 lentelė. RS charakteristikos

Ch-kos pavadinimas	Reikšmė
Pilnumas	Reikalavimų specifikacija turi tiksliai apibrėžti visas galimas sąlygas, su kuriomis bus susidurta, ir galimas reakcijas į jas. Joje turėtų būti įtraukti visi reikalavimai ir aspektai, susiję su funkcionalumu, charakteristikomis, projekto apribojimais, sukaupti reikalingi atsakymai į nenumatytus įvykius.
Neprieštaringumas	Specifikacija, kurioje nėra prieštaravimų tarp atskirų reikalavimų formuluočių, o specifikuotos elgsenos savybės ir apribojimai neturi neigiamos įtakos šiai elgsenai. Pagal profesorių Pericles Loucopoulos yra v*idinis neprieštaringumas* – jokių prieštaravimų modelyje ir i*šorinis neprieštaringumas* – tai, kas galioja probleminei sričiai, turi būti nustatyta ir reikalavimų modelyje.
Tikslumas	Reikalavimų specifikacija turi tiksliai apibrėžti sistemos pajėgumus ir kaip jie tarpusavyje siejasi.
Keičiamumas	Modifikuojama specifikacija turi būti tokia, kad būtų galima suderinti atitinkamas struktūras ir charakteristikas. Elementų dubliavimas padidina aiškumą, tačiau atlikus pakeitimus atsiranda keblumų. Loginė bei hierarchinė reikalavimų specifikacijos struktūra neturi apsunkinti reikalingą modifikaciją ir norint modifikuoti reikalavimų specifikacijas, susiję elementai turi būti sugrupuoti kartu, o nesusiję – atskirti.
Klasifikavimas	Specifikacijos reikalavimai turi būti hierarchiškai klasifikuojami remiantis pastovumu, saugumu, suvoktu įgyvendinimo sudėtingumu arba pagal kitokį kriterijų, kuris pagelbsti sudarant specifikaciją.
Testuojamumas	Reikalavimų specifikacija turi būti sudaryta tokiu būdu, kad naudojantis ja būtų galima patikrinti ar sistema tinkamai atitinka reikalavimus.
Atsekamumas	Reikalavimas specifikacijoje turi būti unikaliu būdu sutapatinamas su pradine informacija, taip pat turi būti ryšys, kuris leistų pereiti atgal į prieš tai buvusį dokumentą ar pirmyn į sekantį dokumentą.
Nedviprasmiškumas	Tik vienareikšmiškai suprantami reikalavimai, kurie negali turėti kelių prasmių(nebūtų vienodi vardai), kurių daugiausia yra naudojant natūralią kalbą.

Pagrįstumas	Reikalavimai turi atitikti pagrindinius dalyvių poreikius ir norus, kurie galėtų atsakyti į tokius klausimus: • Ar įmanoma užtikrinti, kad kiekvienas reikalavimas yra tai, ko užsakovas ištikrųjų nori ir jam reikia? • Ar įmanoma užtikrinti, kad kiekvienas reikalavimas yra tai, ko vartotojo atstovas nori ir jam reikia? • Ar įmanoma užtikrinti, kad kiekvienas reikalavimas yra tai, ko rinkos atstovas nori ir jam reikia? Pagrįsta specifikacija tai tokia, kurioje visi projekto dalyviai gali ją suprasti, analizuoti ir pritarti.
Patikrinamumas	Reikalavimai turi turėti pradinius informacijos šaltinius ir turi juos atitikti. Norint reikalavimų specifikaciją patikrinti viename abstrakcijos lygyje, ji turi būti suderinta su abstrakcija kitame lygyje.
Tinkamumas naudoti	Reikalavimai turi būti praktiški daugybei tarpininkų (ar reikalavimai suprantami ir patogūs užsakovų ir vartotojų tarpininkams, kurie turi jais naudotis srities valdymui ir įvertinimui; ar jie suprantami ir patogūs vadovams, kurie turi naudotis jais tiek srities, tiek išlaidų, planavimų valdymui, eigos metrikoms; ar reikalavimai suprantami ir patogūs projektuotojams ir programuotojams, kurie juos turi įgyvendinti).

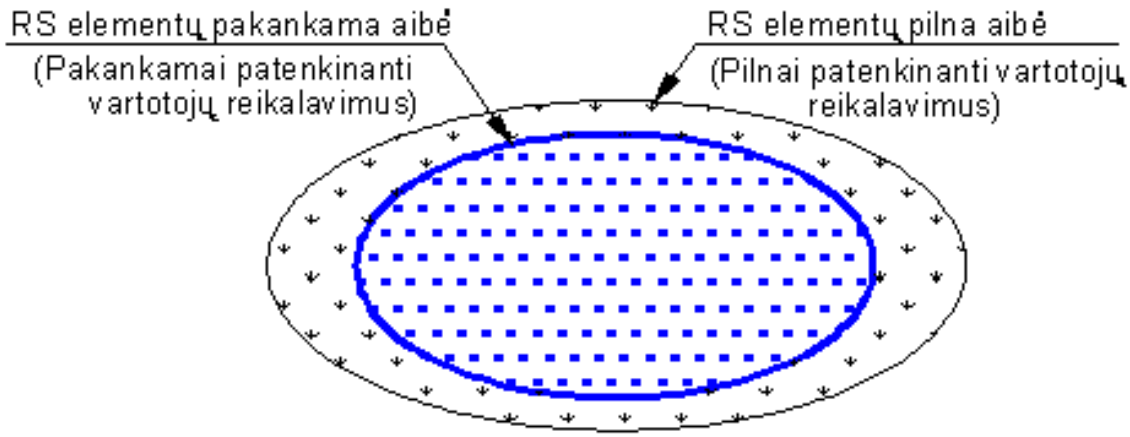
Norint užtikrinti programinės įrangos reikalavimų kokybę, turi būti atlikti validavimo ir verifikavimo procesai, kurie patikrintų surinktų reikalavimų atitikimus. Validavimo metu turi būti įsitikinama, kad užsakovo lūkesčiai ir norai tinkamai išsiaiškinti ir suprasti. Verifikavimas atlieka reikalavimų specifikacijų charakteristikų savybių tikrinimą.

Kaip žinoti, jog programinės įrangos reikalavimų specifikacija yra kokybiška? Kaip tai nustatyti? Atsakymas į tuos klausimus būtų pilna specifikacija, kuri atitiktų užsakovo reikalavimus konkrečiam produktui ar sistemai. Tačiau tai būna itin retai, nes sudėtinga visiškai įsitikinti, kad užsakovo pateikti reikalavimai yra galutiniai ir nekeičiami. Programinės įrangos specifikacijos kokybės požymiu yra subjektyvūs, todėl nereikalingi jokie rodikliai ar matavimo priemonės, parodančios kiek sudarytosios reikalavimų specifikacijos kalba yra tinkama. Reikalavimų specifikacijų kokybė yra labai svarbi. Ji garantuoja, kad nebus nustatytos sistemos klaidos prieš pradedant ja naudotis. Suradus defektus tokiame vėlyvame etape nebus išvengta žymių projekto nuostolių ir papildomų laiko sąnaudų klaidų ištaisymui. Reikalavimų specifikacija turėtų būti laikoma subjektu, naudojamu nuodugniai patikrinti programinę įrangą. Programinės įrangos reikalavimų specifikacijų kokybei vertinti informacijos sistemų katedroje plėtojamas ODRES metodas yra tinkamas metodas.

2.6 Reikalavimų specifikacijų pilnumo ir pakankamumo analizė

Reikalvimų specifikacijos pilnumas – plačiai suvokiama sąvoka, kadangi realiai mąstant, sąvoka pilnumas yra nepamatuojama ir neapskaičiuojama reikšmė, ypač jei vertinama vartotojo pateikta informacija. Tačiau sudarant 29iagrama29mu specifikacijas atsiranda noras išmatuoti RS pilnumą, įvertinti kiekybiškai kiek 29iagrama29mu specifikacija jau yra užpildyta, kiek dar liko įvesti informacijos, kad ji būtų pakankama, jog būtų galima sudaryti pradinį veikiantį sistemos prototipą.

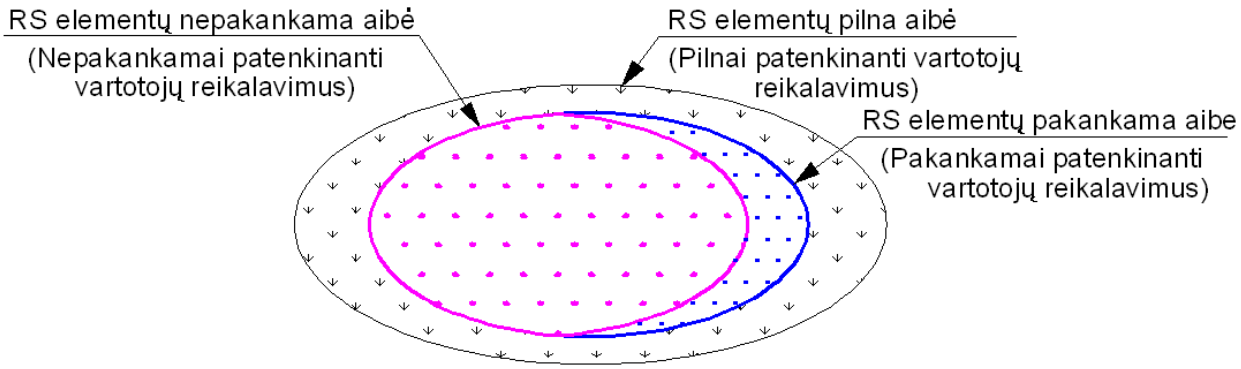
Sąvoka pilnumas galima suprasti kaip visiškai pilnas, jei tai yra apčiuopiamo daikto matavimas, pvz: pilna stiklinė vandens, jei matuotume stiklinės tūrio talpą, tai galėtume teigti, kad stiklinė yra pilna. Tačiau kalbant apie informaciją, žodis pilnumas įgyja sąlyginę reikšmę. Vartotojo pateiktą informaciją reikalavimų specifikacijos sudarymui, galima apibrėžti kaip pakankama reikalavimų specifikacija arba perteklinė reikalavimų specifikacija. Sąvokų pilnumas ir pakankamumas grafinė schema pateikta 5 paveikslėlyje.



5 pav. Reikalavimų specifikacijos elementų pilnumo ir pakankamumo aibės

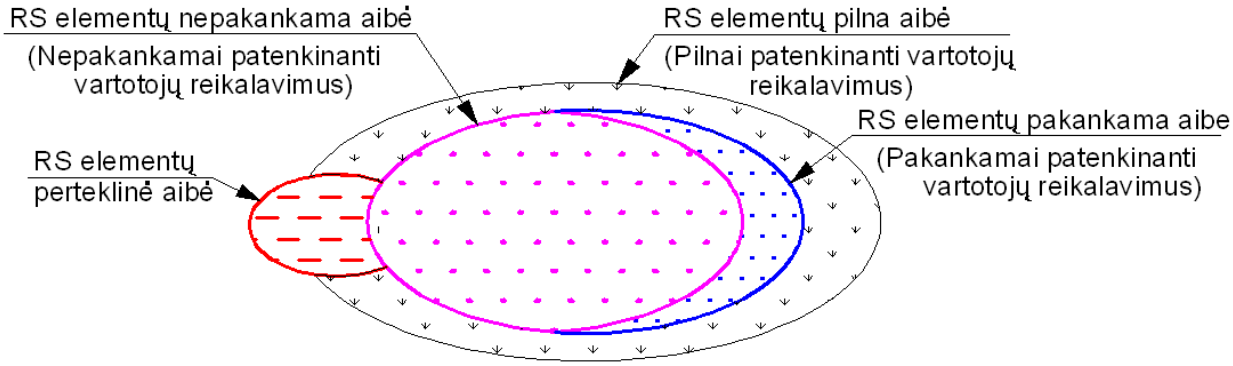
Reikalavimų specifikacijos pakankamumo aibė apibrėžiama, kaip RS elementų aibė, kuri yra pakankama, kad būtų galima sudaryti veikiantį sistemos prototipą, t.y. yra žinomas bent viena esybė, bent vienas jos atributas ir bent viena funkcija. Turėdami šiuos tris duomenis, mes galime sudaryti minimalų, veikiantį sistemos prototipą.

Jei sudarant reikalavimų specifikacijas trūksta informacijos ar duomenų, kad būtų galima sudaryt veikiantį prtototipą, tuomet reikalavimų specifikacijos elementų aibė yra nepakankama. RS nepakankamumas gali būti dėl to, kad vartotojas pateikdamas 29iagrama29mus, pamiršo pasakyti reikalingos informacijos sistemai sudaryti. RS elementų nepakankamumo aibė pavaizduota 6 paveikslėlyje.



6 pav. Reikalavimų specifikacijos elementų nepakankama aibė

Jei sudarant reikalavimų specifikacijas lieka nepanaudotos informacijos ar duomenų, o sudarytas veikiantis sistemos prototipas yra korektiškas, tai tuomet reikalavimų specifikacijos elementų aibė yra perteklinė. RS elementų perteklinė aibė pavaizduota 7 paveikslėlyje.



7 pav. Reikalavimų specifikacijos elementų perteklinė aibė

Norint įvertinti reikalavimų specifikacijos kokybę reikia:

- įvertinti reikalavimų specifikacijos pilnumą, neprieštaringumą, tikslumą ir kitas charakteristikas, kurios buvo aprašytos 2.5 skyriuje;
- nustatyti ar reikalavimų specifikacija yra minimali, pakankama;

RS kokybė vertinama tikrinant aukščiau išvardintas charakteristikas, tačiau šiame darbe reikalavimų specifikacija bus vertinama tikrinant ar RS yra pilna – pakankama, kad butų galima sudaryti veikianti sistemos prototipą.

Darbe yra pasiūlyta reikalavimų specifikacijos kokybės vertinimo metodų lyginamosios analizės metodika, kurioje koncepcinės analizės metodas suderintas su suderinamumų ir skirtumų nustatymo metodais. Koncepcinės analizės metodo paskirtis – išskirti pagrindinius reikalavimų specifikacijų kokybės vertinimo metodų komponentus. Suderinamumų ir skirtumų

nustatymo metodas naudojamas palyginimo lentelei užpildyti ir palyginti komponentus nustatant jų panašumus ir skirtumus.

Reikalavimų specifikacijos pilnumas suprantamas kaip specifikacijos išbaigtumas, kurioje yra pateikta visa reikalinga informacija. Šiame darbe RS pilnumas vertinamas bus įvertinant DB lentelių pilnumą, t.y. pagal sudarytą etaloninę DB, kurios sudėtis bus vertinama 100 % pilnumu. Vertinant reikalavimų specifikacijos pilnumą jos sudarymo metų, bus lyginamos RS duomenų bazė su etalonine duomenų baze ir į atitinkamą lentelę surašomi sutapimai ir nesutapimai tarp šių DB. Kiekviena DB lentelė bus lyginama su etaloninės DB atitinkama lentele, kur rasti sutapimai ir nesutapimai įtakos svorį galutiniam DB pilnumo įvertinimui.

2.7 Analitinės dalies išvados

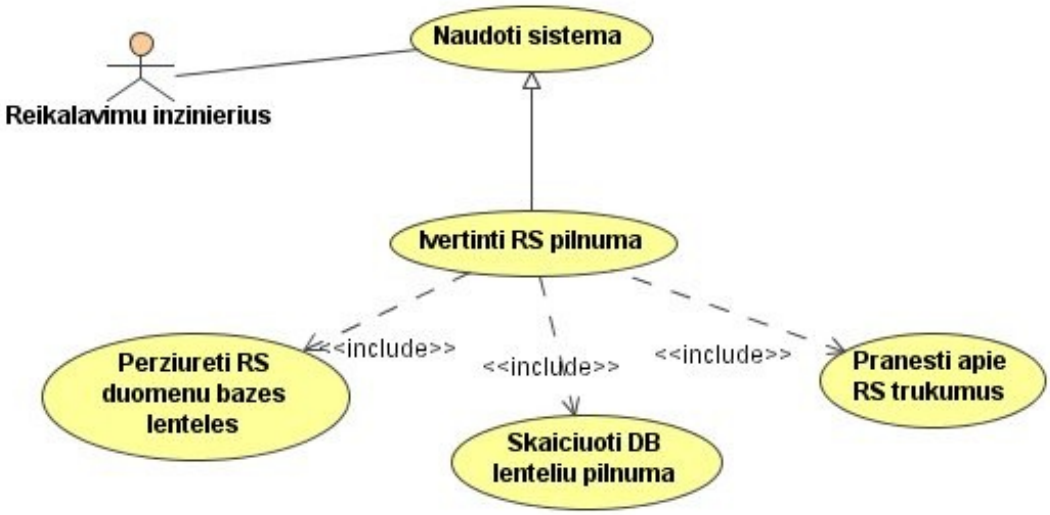
Atlikus reikalavimų specifikacijos pilnumo įvertinimo galimybių tyrimo analizę, prieita tokių išvadų:

- Reikalavimų specifikacijų kokybės įvertinimas – pakankamumas, užbaigtumas, nedviprasmiškumas, suderinamumas, pagrįstumas. Taip pat turi atitikti kitas reikalavimų specifikacijos kokybės charakteristikas.
- Automatiškai patikrinti, ar sudaryta reikalavimų specifikacija yra pilna – neįmanoma, nes įrankis negali patikrinti, ar žmogus ko nors nepamiršo.
- Reikalavimų specifikacijos kokybės įvertinimo analizės metu nustatyta, kad egzistuojantys CASE įrankiai, kurie turi galimybę vertinti RS kokybę, negali būti panaudoti kartu su ODRES metodą automatizuojančiu įrankiu, nes esantys CASE įrankiai negali naudoti ODRES metabazės.
- Tikslinga sukurti įrankį, kurio pagalba nustatomas ODRES metodu sudarytos reikalavimų specifikacijos pakankamumas bei minimalumas.

3. Reikalavimai RS kokybės vertinimui

3.1 Reikalavimų specifikacijos panaudojimo atvejai

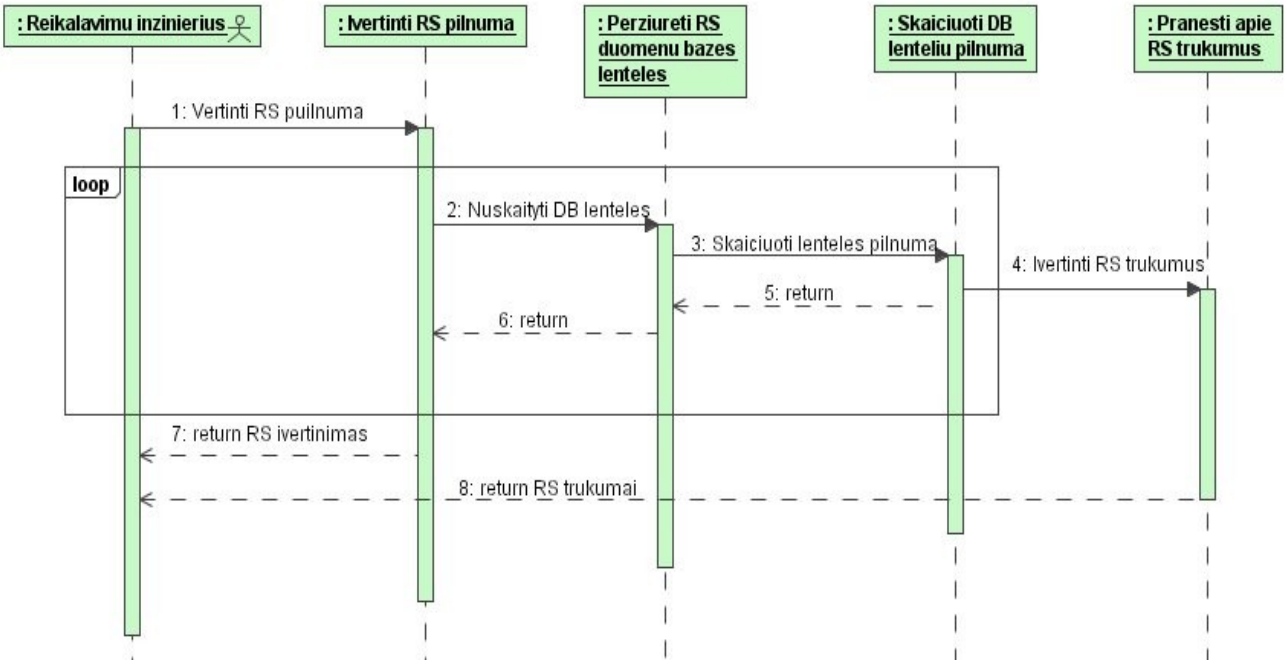
Pasirinktas kompiuterizuojamas etapas – reikalavimų specifikacijos pilnumo vertinimas. 8-ame paveiksle pavaizduoti pasirinkto kompiuterizuoti etapo panaudojimo atvejai:



8 pav. Pasirinkto etapo panaudojimo atvejai

Diagramoje pavaizduotas pagrindinis panaudojimo atvejis „Įvertinti RS pilnumą“. Šį panaudojimo atvejį detalizuoja dar trys panaudojimo atvejai „Peržiūrėti RS duomenų bazės lenteles“, Skaičiuoti DB lentelių pilnumą“ „Pranešti apie RS trūkumus“. Panaudojimo atvejuose dalyvauja veiklos aktorius „Reikalavimų inžinierius“ ir sistema.

Reikalvimų specifikacijos kokybės vertinimo veiksmų seka pavaizduota 9ame paveikslėlyje.



9 pav. Sekų diagrama panaudojimo atvejui „Įvertinti RS pilnumą“

3 lentelė: Panaudojimo atvejo „Įvertinti RS pilnumą“ specifikacija

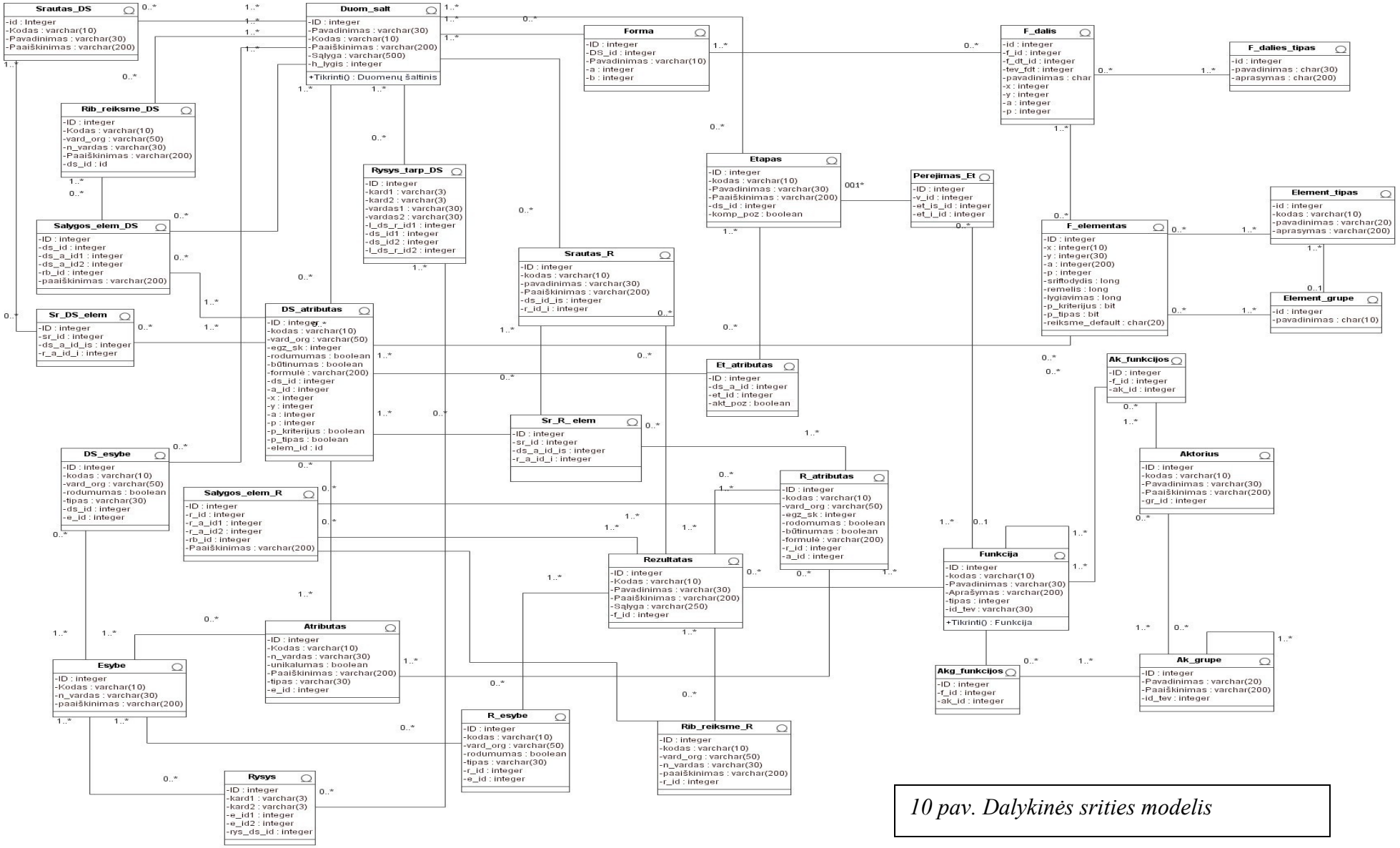
PA „**Įvertinti RS pilnumą**“		
Prieš sąlyga		Vartotojas inžinierius Užpildyta DB
Sužadinimo sąlyga		Inžinierius nori įvertinti RS pilnumą
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	Peržiūrėti RS duomenų bazės lenteles Skaičiuoti DB lentelių pilnumą Pranešti apie RS trūkumus
	Specializuoja PA	
Pagrindinis įvykių srautas		**Sistemos reakcija ir sprendimai**
1. Parenkamas duomenų šaltinis		**1.1** Sistema pateikia RS pilnumo vertinimo langą
2. Parenkami raktiniai elementai		**2.1** Tikrinamas DB lentelių pilnumas
3. Koreguojamos padėtys		
4. Išsaugoma		
Po sąlyga:		Pateikiama RS pilnumo įvertinimas tekstine ir grafine forma

Nefunkciniai reikalavimai

Sistemai keliami papildomi reikalavimai, kurie būtini užtikrinant efektyvų darbą:

- Sistema turi veikti patikimai, t.y. jos darbas neturi būti nutraukiamas dėl nenumatytų klaidų.
- Sistema privalo būti greitai įsisavinama naujų vartotojų, t.y. darbuotojų apmokymas turi trukti kiek įmanoma trumpiau.
- Sistema turi būti palaikoma, t.y. naujos sistemos galimybės gali būti nesudėtingai įtraukiamos.
- Sistemos struktūra turi būti intuityviai nuspėjama, tai palengvina darbą su dideliu informacijos kiekiu.

3.2 Dalykinės srities modelis



10 pav. Dalykinės srities modelis

10-ame paveiklėlyje pavaizduotas dalykinės srities modelis. Šioje diagramoje vaizduojamos esybės būtinos RS pilnumo įvertinimui, kurios turi kiekviena sau priklausančius atributus. Patogumo dėlei dalykinės srities modelis atvaizduojamas esybe „RS duomenys“.

11 pav. RS pilnumo vertinimo analizės modelis

Analizės modelyje (11 pav.) dalyvauja vienas langas, kuriame bus pateikiama reikalavimų specifikacijos pilnumo įvertinimas grafine forma ir pateikiami tekstiniai pranešimai apie RS trūkstamas dalis. RS pilnumo vertinimas atliekamas per valdiklį „Pilnumo vertinimo valdiklis“, kuris kreipiasi į esybę „RS duomenys“ .

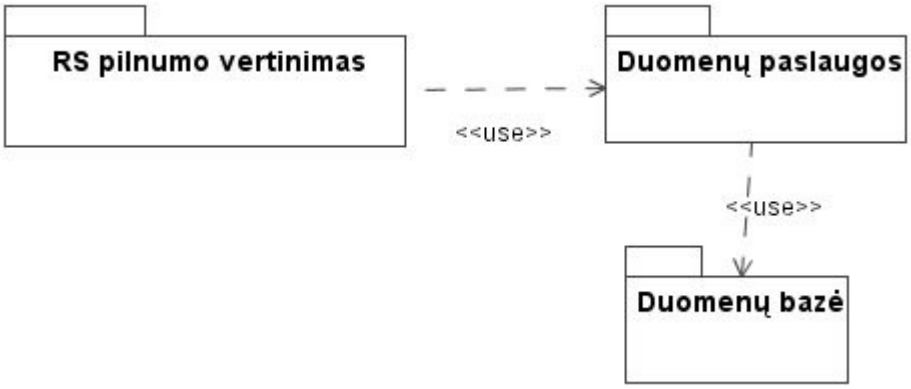
4. Reikalavimų specifikacijos pilnumo vertinimo įrankio projektas

4.1 Sprendimo pagrindimas

Reikalavimų specifikacijos pilnumo vertinimo įrankis yra susietas su ODRES metabaze. Šioje bazėje lentelės saugo informaciją, kuri reikalinga specifikacijai sudaryti. Informacijos metabazėje turi būti tiek, kad specifikacija būtų pakankama, jog būtų galima sudaryti veikiantį sistemos prototipą. Šio įrankio įnašas į šį metodą yra įvertintas reikalavimų specifikacijos pilnumas, pateiktas RS pilnumo įvertinimo lange grafine ir tekstine išraiška. Įrankio realizavimui pasirinkta MS Visio 2003, kadangi joje jau kuriami kiti ODRES metodo etapai.

4.2 Sistemos architektūra

4.2.1 Loginė sistemos architektūra

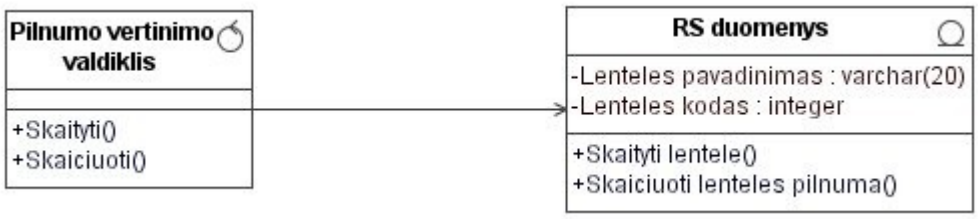


12 pav. Paketų diagrama

RS pilnumo vertinimo įrankis valdomas dialogo RS pilnumo vertinimas pagalba.

Projekte naudojama išorinė duomenų prieiga. Per duomenų paslaugų paketa susisiekiama su sistemos duomenų baze.

4.2.2 Vartotojo, veiklos, duomenų paslaugų klasių diagramos



13 pav. RS pilnumo vertinimo veiklos logikos klasių diagrama

Šioje diagramoje yra pavaizduoti ryšiai tarp duomenų klasių ir valdiklio klasės.

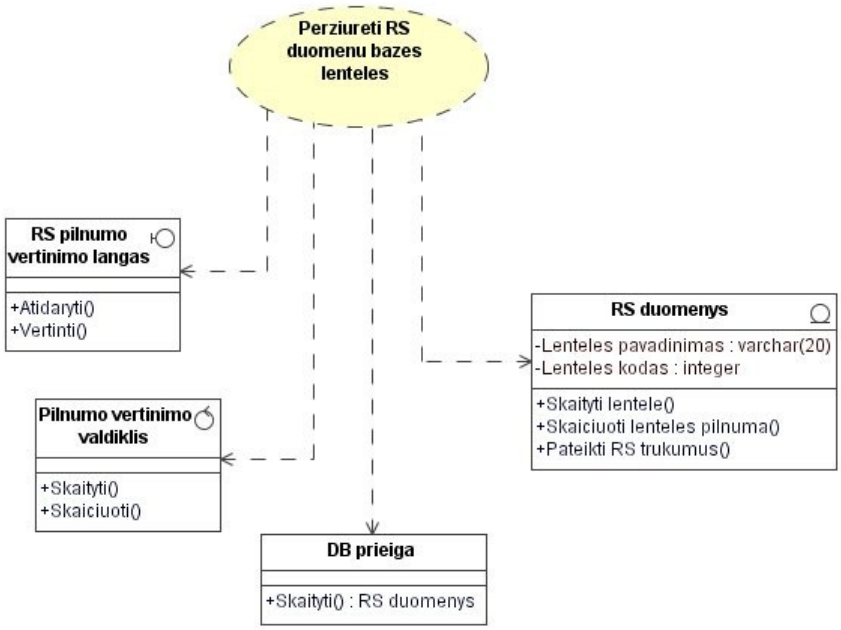
14 pav. RS pilnumo vertinimo vartotojo logikos klasių diagrama

Šioje diagramoje pateikiami valdikliai ir formos dalyvaujantys sąsajos generavimo procese. RS pilnumo vertinimo vartotojo logikos klasių diagramą sudaro vartotojo interfeiso ir veiklos logikos klasės. Nurodomi atributai ir operacijos.

4.3 Detalus projektas

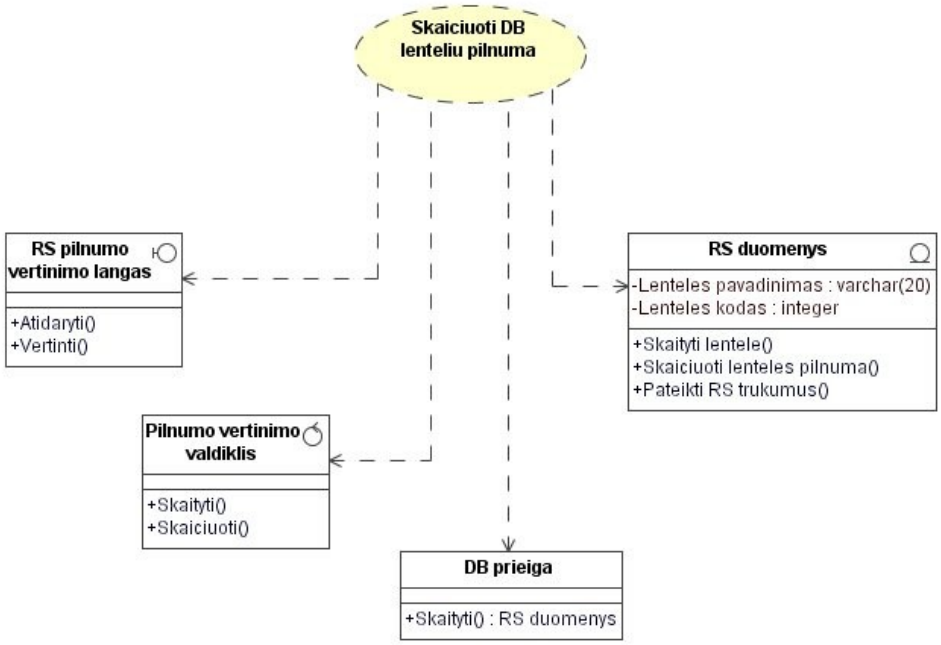
15 pav. Praplėsti panaudojimo atvejai

Panaudojimo atvejų realizacijos diagramos rodo, kokios klasės reikalingos realizuojant kiekvieną panaudojimo atvejį. Šios diagramos padeda išanalizuoti, ar yra visos reikiamos klasės, ar nėra nereikalingų ir pan.



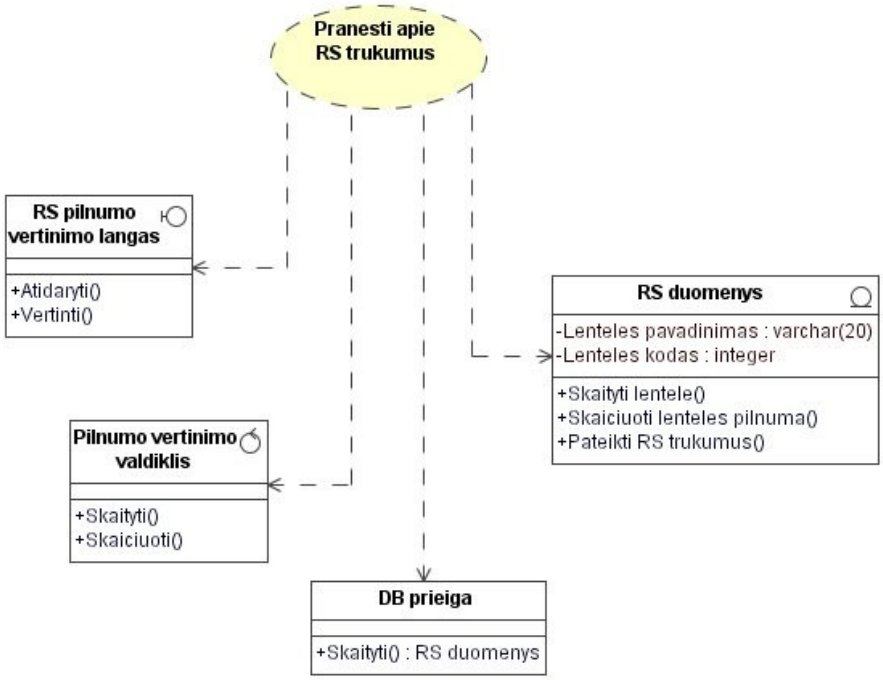
16 pav. Peržiūrėti duomenų bazės lenteles panaudojimo atvejo realizacijos klasių diagrama

Panaudojimo atvejis „*Peržiūrėti duomenų bazės lenteles*“ naudoja „Pilnumo vertinimo valdiklį“, „RS pilnumo vertinimo langą“, esybę „RS duomenys“ ir „DB prieigą“.



17 pav. Skaičiuoti DB lentelių pilnumą panaudojimo atvejo realizacijos klasių diagrama

Panaudojimo atvejis „*Skaičiuoti DB lentelių pilnumą*“ naudoja „Pilnumo vertinimo valdiklį“, „RS pilnumo vertinimo langą“, esybę „RS duomenys“ ir „DB prieigą“.

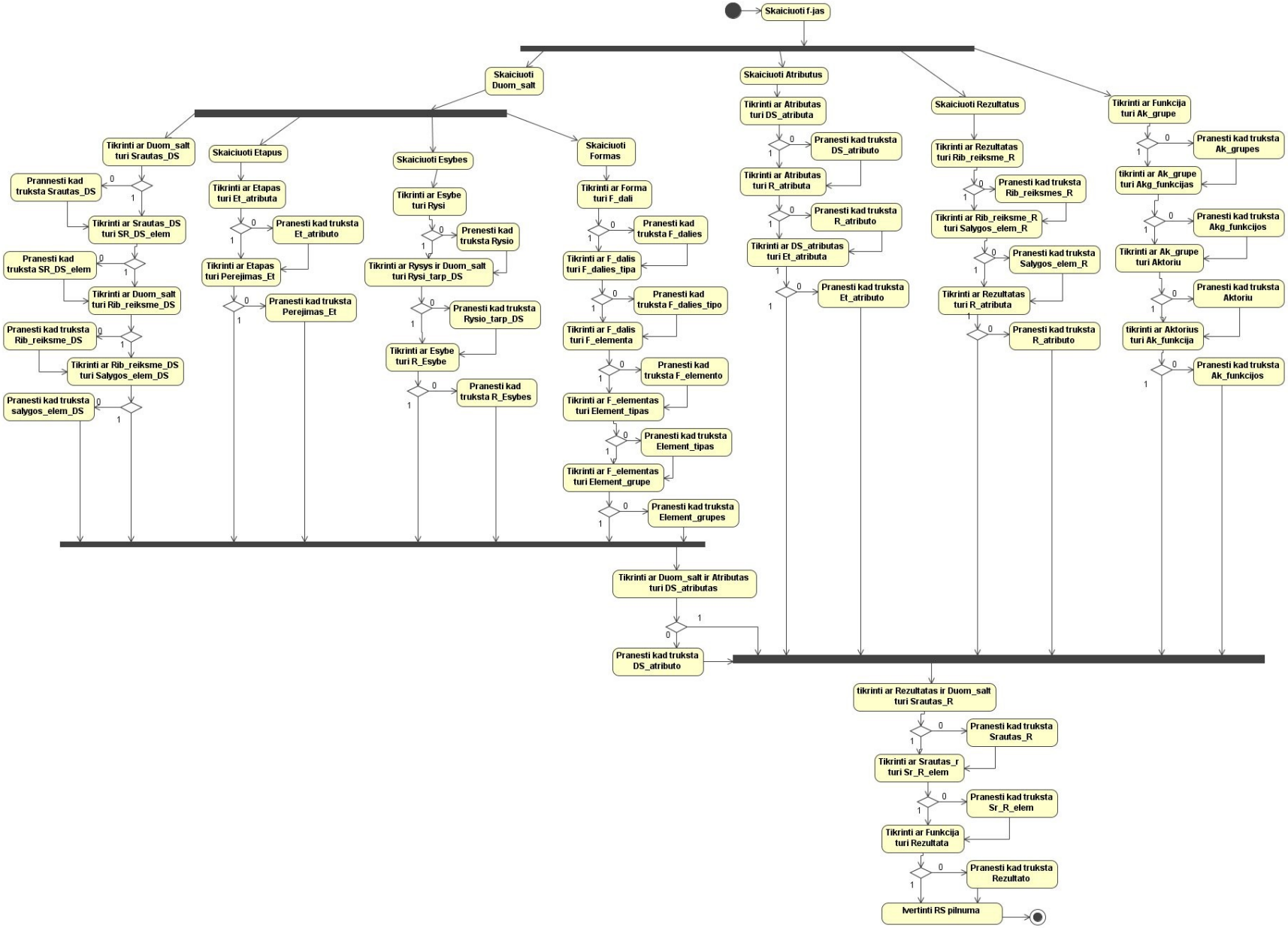


18 pav. Pranešti apie RS trūkumus panaudojimo atvejo realizacijos klasių diagrama

Panaudojimo atvejis „*Pranešti apie RS trūkumus*“ naudoja „Pilnumo vertinimo valdiklį“, „RS pilnumo vertinimo langą“, esybę „RS duomenys“ ir „DB prieigą“.

4.4 Sistemos elgsenos modelis

4.4.1 RS pilnumo vertinimo veiklos modelis



19 pav. RS pilnumo vertinimo veiklos diagrama

Veiklos diagramos pirmas žingsnis – tikrinti metabazės lentelės „Funkcija“ pilnumą. Visų pirma yra nuskaitoma ši lentelė, peržiūrima ar toje lentelėje yra nors vienas įrašas. RS pilnumo įvertinimas yra atliekamas patikrinant visas lenteles ir įvertinant jų pilnumą, jei kurioje nors lentelėje trūksta duomenų, iškart yra fiksuojamas pranešimas apie trūkstamus duomenis. Metabazės pilnumas tikrinamas keliais lygiagrečiais veiklos sekų blokais, kur atskirai yra įvertinamas duomenų šaltinių, etapų, esybių, formų, rezultatų, atributų pilnumas. Atlikus atskirų blokų pilnumo patikrinimą yra tikrinimas, ar tie blokai tarpusavyje siejasi, ar turi bendrų duomenų. Jei blokai ar juose esančios lenteles tarpusavyje nesisieja, tai metabazė yra nepakankamai užpildyta ir RS nėra pakankamai pilna.

Formulė pagal kurią bus skaičiuojamas bendras RS pilnumas:

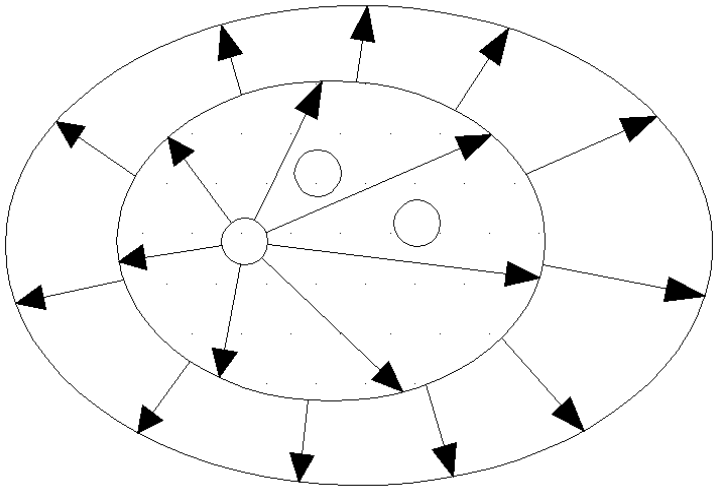
$k_1 = kiek_{Funkcija}$
$k_2 = kiek_{Rezultatas}$
$k_3 = kiek_{Perejimas\_Et}$
$k_4 = kiek_{Duom\_salt}$
$k_5 = kiek_{Esybe}$
$k_6 = kiek_{Atributas}$
$k_7 = kiek_{Srautas\_DS}$
$k_8 = kiek_{DS\_esybe}$
$k_9 = kiek_{Sr\_DS\_elem}$
$k_{10} = kiek_{Salygos\_elem\_DS}$
$k_{11} = kiek_{Rib\_reiksme\_DS}$
$k_{12} = kiek_{Rysys}$
$k_{13} = kiek_{Rysys\_tarp\_DS}$
$k_{14} = kiek_{DS\_atributas}$
$k_{15} = kiek_{Srautas\_R}$
$k_{16} = kiek_{R\_atributas}$
$k_{17} = kiek_{Sr\_R\_elem}$
$k_{18} = kiek_{Salygos\_elem\_R}$
$k_{19} = kiek_{Rib\_reiksme\_R}$
$k_{20} = kiek_{R\_esybe}$
$k_{21} = kiek_{Etapas}$
$k_{22} = kiek_{Et\_atributas}$
$k_{23} = kiek_{Forma}$
$k_{24} = kiek_{F\_dalis}$
$k_{25} = kiek_{F\_dalies\_tipas}$
$k_{26} = kiek_{F\_elementas}$
$k_{27} = kiek_{Element\_tipas}$
$k_{28} = kiek_{Element\_grupe}$
$k_{29} = kiek_{Aktorius}$
$k_{30} = kiek_{Ak\_funkcijos}$
$k_{31} = kiek_{Ak\_grupe}$
$k_{32} = kiek_{Akg\_funkcijos}$

$$P_{RS} = \frac{k_1 + k_2 + k_3 + k_4 + k_5 + k_6 + k_7 + k_8 + k_9 + k_{10} + k_{11} + k_{12} + k_{13} + k_{14} + k_{15} + k_{16} + k_{17} + k_{18} + k_{19} + k_{20} + k_{21} + k_{22} + k_{23} + k_{24} + k_{25} + k_{26} + k_{27} + k_{28} + k_{29} + k_{30} + k_{31} + k_{32}}{9 \cdot k_1 + 20 \cdot k_2} \cdot 100\%$$

Čia P_{RS} – reikalavimų specifikacijos bendras pilnumas;

$k_{1, 2, \dots n}$ – atitinkamos duomenų bazės lentelės įrašų kiekis.

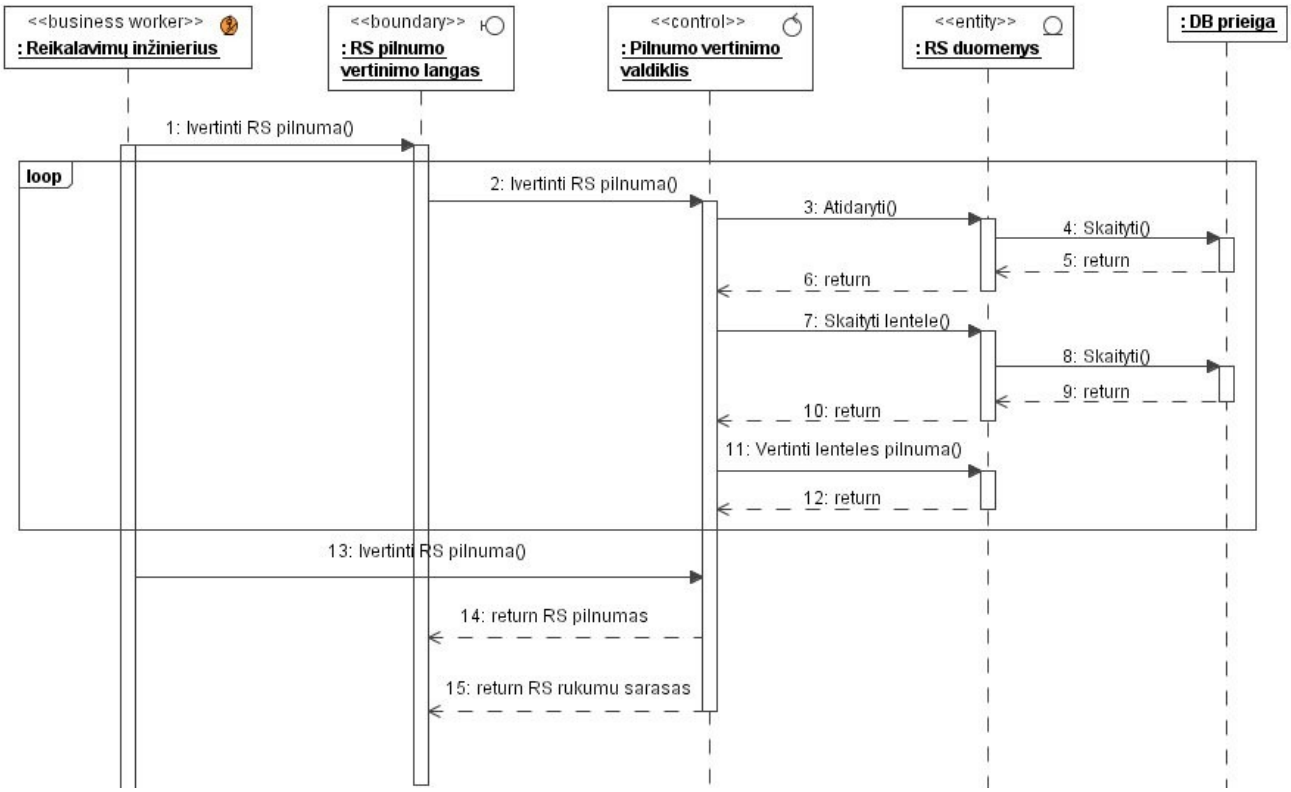
Atskiros lentelės pilnumas priklauso nuo prieš tai tikrintos lentelės gautų rezultatų. Pavyzdžiui, jei lentelėje „funkcija“ yra trys funkcijos, tai tikrinant lentelės „Rezultatas“ pilnumą, nepakanka, kad joje būtų vienas rezultatas, joje turi būti kiekvienai funkcijai priskirtas atitinkamas rezultatas. Tuomet lentelė „Rezultatas“ bus pakankamai pilna. Kitas atvejis – jei lentelėje „Rezultatas“ yra penki rezultatas, tai lentelėje „R_atributas“ turi būti bent penki atributai atitinkamai priskirti kiekvienam rezultatui, kad lentelė „R_atributas“ būtų pakankamai pilna, nors joje atributų gali būti ir daugiau nei penki.



20 pav. RS apimties dinamika

20-ame paveikslėlyje pavaizduota RS apimties dinamika, t.y. čia parodyta, kaip gali kisti reikalavimų specifikacijos apimtis, priklausomai nuo to, kiek joje yra elementų. Įrašius į specifikaciją nauja elementą, reikalvimų specifikacijos apimtis gali nepakisti arba gali išaugti į didesnę, tai priklauso koks elementas bus ivedamas į RS.

4.4.2 Sekų diagramos



21 pav. RS pilnumo įvertinimo sekos diagrama

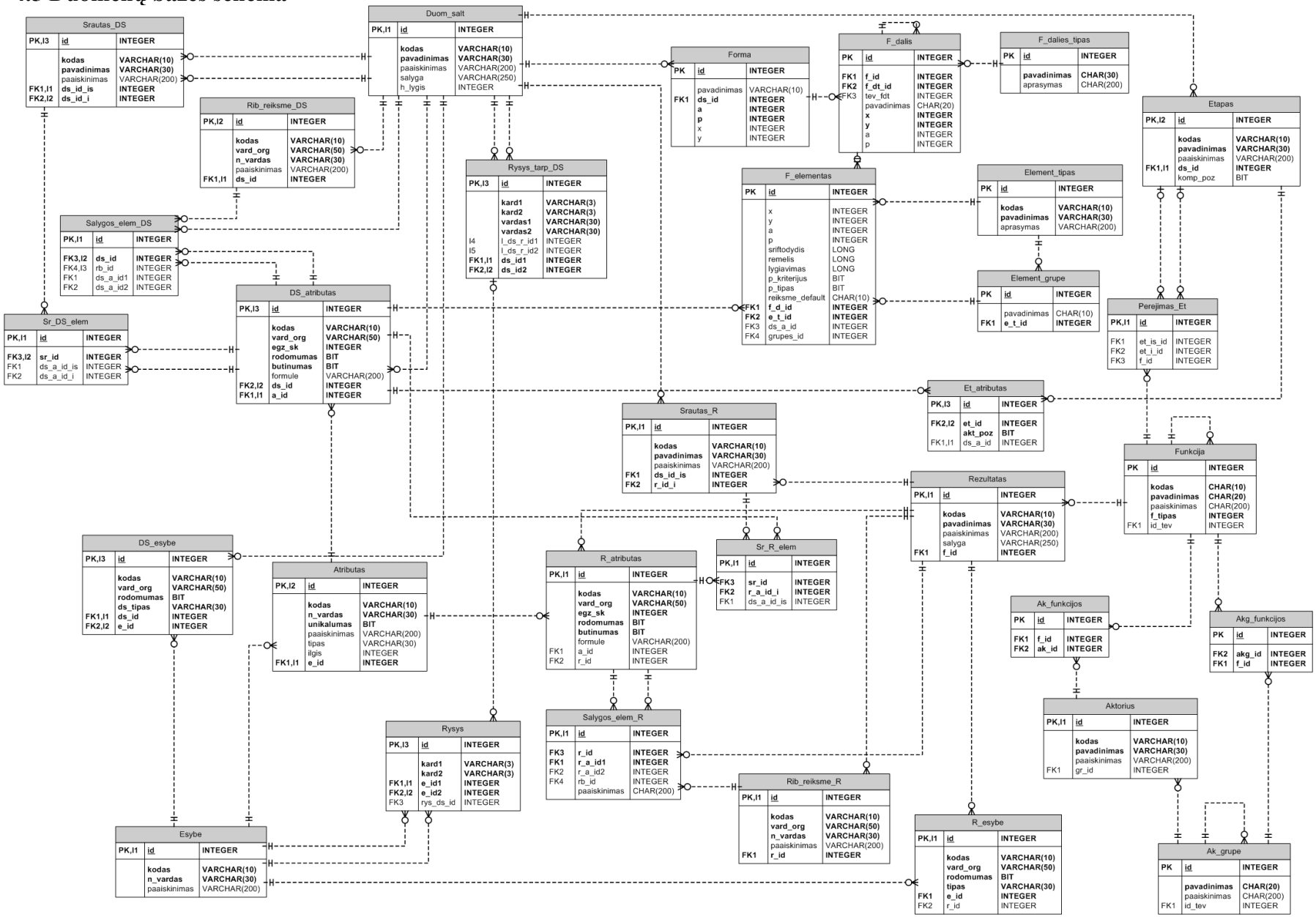
Ši sekos diagrama vaizduoja veiksmų seką, kuri yra atliekama norint įvertinti RS pilnumą. Sekoje yra atvaizduoti kokių nuoseklumu, per kokias klases yra atliekamas RS pilnumo įvertinimas.

4.4.3 Navigacijos planas

22 pav. RS pilnumo įvertinimo navigacijos planas

Navigacijos planą sudaro vienas langas, nes visa informacija apie RS pilnumo įvertinimą yra pateikiama tame pačiame lange.

4.5 Duomenų bazės schema



23 pav. ODRES metodui taikoma metabazės schema

23 paveiksle pateikta doc. dr. R. Butkienės metabazės schema, kuri taikoma ODRES metodui. Čia pateiktos lentelės ir jų tarpusavio ryšiai, kuriose bus saugoma visa reikalavimų specifikacijų informacija.

4.6 Testavimo modelis, pavyzdiniai duomenys bei kontrolinis pavyzdys

Keletos duomenų bazėje esančių lentelių duomenys:

Funkcija : Table

id	kodas	pavadinimas	paaiskinimas	f_tipas	id_tev
1	f01	prasymas	prasymo sudarymas	1	1
2	f02	ataskaita	ataskaitos sudarymas	2	1
3	f03	anketa	anketos sudarymas	3	2
4	f04	užsakymas	užsakymo sudarymas	4	2
5	f05	saskaita	saskaitos israsyma	5	1
6	f06	orderis	orderio israsymas	6	1

Record: 6 of 6

24 pav. Lentelės „Funkcija“ duomenys

Rezultatas : Table

id	kodas	pavadinimas	paaiskinimas	salyga	f_id
1	kod1	prasymas	suformuota sasaja	salyga1	1
2	kod2	anketa	sudaryta anketa	salyga2	3
3	kod3	užsakymas	sudarytas užsakymas	salyga3	4
4	kod4	saskaita	israsyta saskaita	salyga4	5
5	kod5	ataskaita	suformuota ataskaita	salyga5	2
6	kod6	orderis	israsytas orderis	salyga6	6

Record: 1 of 6

25 pav. Lentelės „Rezultatas“ duomenys

Srautas_R : Table

id	kodas	pavadinimas	paaiskinimas	ds_id_is	r_id_i
1	kodas1	srautas1	srautas tarp DS ir rezultato	1	1
2	kodas2	srautas2	srautas tarp DS ir rezultato	2	2
3	kodas3	srautas3	srautas tarp DS ir rezultato	3	3
4	kodas4	srautas4	srautas tarp DS ir rezultato	4	3
5	kodas5	srautas5	srautas tarp DS ir rezultato	4	4
6	kodas6	srautas6	srautas tarp DS ir rezultato	4	5

Record: 1 of 6

26 pav. Lentelės „Srautas_R“ duomenys

Duom_salt : Table

id	kodas	pavadinimas	paaiskinimas	salyga	h_lygis
1	DS1	Sutartis	sutarties duomenys	salyga1	1
2	DS2	Techninis pasa:	draudziamas objektas	salyga2	1
3	DS3	Darbo sutartis	draudimo agentai	salyga3	2
4	DS4	Asmens dokum	draudiku duomenys	salyga4	2

Record: 1 of 4

27 pav. Lentelės „Duom_salt“ duomenys

DS_esybe : Table

id	kodas	vard_org	rodomumas	ds_tipas	ds_id	e_id
2	drd	draudejas	-1	1	1	2
3	drobj	objektai	-1	1	1	3
4	es4	identifikavimo n	-1	1	2	4
			0			

Record: 5 of 5

28 pav. Lentelės „DS_esybe“ duomenys

Esybe : Table

kodas	n_vardas	paaiskinimas
nr	sutartis	Poliso duomenys
drd	draudejas	draudėjo duomenys
obj	objektai	apdrausti objektai
msn	masina	masinos duomenys

Record: 4 of 4

29 pav. Lentelės „Esybė“ duomenys

DS_atributas : Table

id	kodas	vard_org	egz_sk	rodomumas	butinumas	formule	ds_id	a_id
1	id1	Nuo	1	-1	0	f1	1	1
2	id2	Iki	1	-1	0	f2	1	2
3	id3	Trukme	1	-1	0	f3	1	4
4	id4	Komisas	1	0	0	f4	1	3
5	id5	Kodas	1	-1	0	f5	1	5
6	id6	Vardas	1	-1	0	f6	1	6
7	id7	Pavarde	1	-1	0	f7	1	7

Record: 1 of 7

30 pav. Lentelės „DS_atributas“ duomenys

Atributas : Table

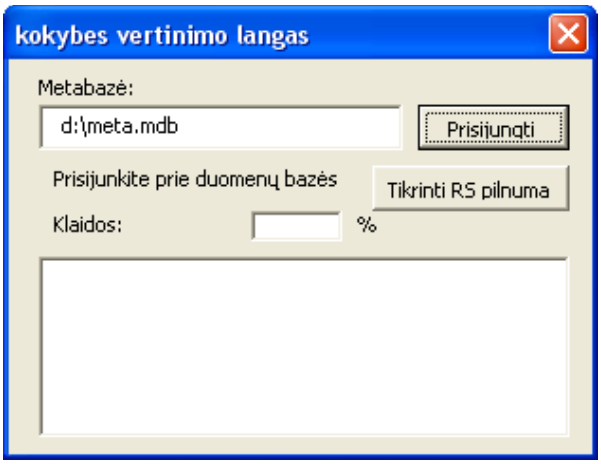
id	kodas	n_vardas	unikalumas	paaiskinimas	tipas	ilgis	e_id
2	atr02	Iki	0	Galioja iki	Date	10	1
3	atr03	Komisas	0	Komisiniu dydis	Integer	2	1
4	atr04	Trukme	0	Sutarties trukm	Date	10	1
5	atr05	Kodas	-1	Asmens kodas	String	15	2
6	atr06	Vardas	0	Draudejo vardas	String	15	2
7	atr07	Pavarde	0	Draudejo pavard	String	15	2
			0				

Record: 7 of 7

31 pav. Lentelės „Atributas“ duomenys

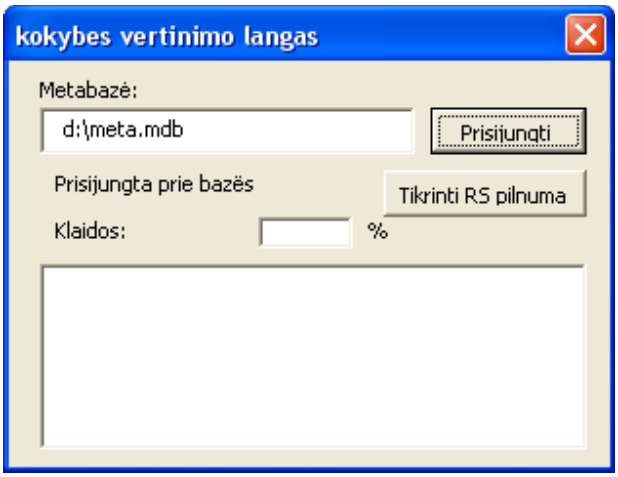
Kontrolinis pavyzdys:

Sugeneruotas RS kokybės vertinimo langas (32 pav.). Programa laukia kol vartotojas prisijungs prie duomenų bazės.



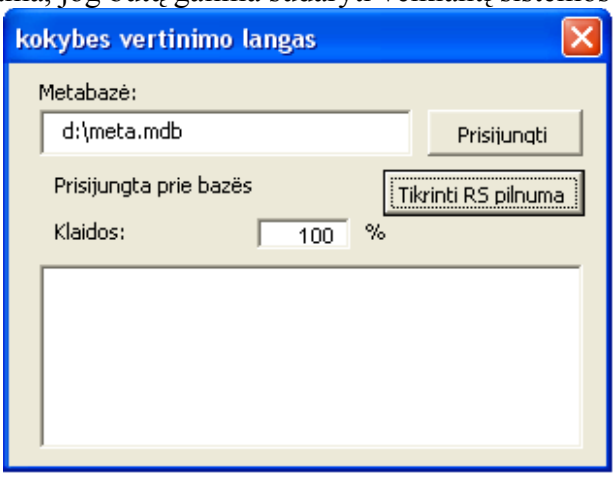
32 pav. Sugeneruotas kokybės vertinimo langas

Vartotojui prisijungus prie duomenų bazės išmetamas pranešimas, kad sėkmingai buvo prisijungta pri DB (33 pav).



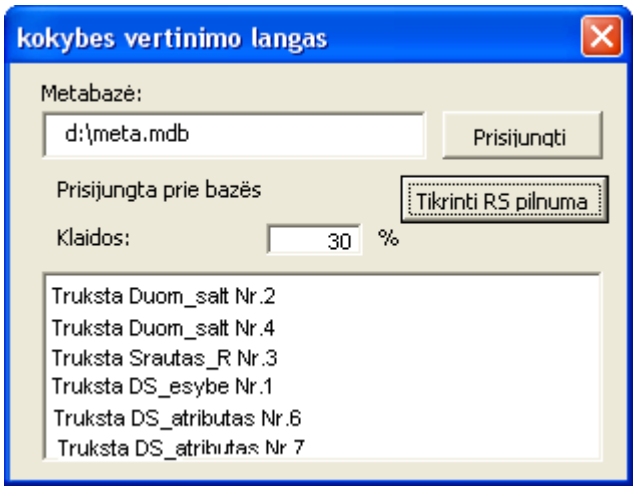
33 pav. Prisijungimas prie DB

Vartotojui paspaudus mygtuką „Tikrinti RS pilnuma“ ir gavus rezultate 100 %, galime teikgti, kad DB yra pakankama, jog būtų galima sudaryti veikiantį sistemos prototipą (34 pav).



34 pav. RS pakankamumo įvertinimas sakmės atveju

Jei gautas rezultatas yra mažiau nei 100 %, tai kokybės vertinimo lange yra išvedamas DB pilnumo lygis procentais, o klaidų laukelyje yra pateikiami RS trūkstami elementai (35 pav).



35 pav. RS pakankamumo įvertinimas nesėkmės atveju

5. Eksperimentinis tyrimas

5.1 Sukurtos sistemos kokybės tyrimas

Sukurtos sistemos kokybė įvertinama pagal 2.3 skyriuje išanalizuotus CASE įrankių galimybes įvertinti RS pakankamumą (4 lentelė).

4 lentelė. CASE įrankių ir įrankio, kuriamo ODRES metodui, galimybės įvertinti RS pakankamumą palyginimas

CASE įrankių galimybės	**Rational Rose**	**ProVisionWorkbench**	**MagicDraw**	**Įrankis ODRES metodui**
Įvertinti RS pilnumą	+	+	+	+
Pilnumo įvertinimas bet kuriuo metu, pildant RS				+
RS pilnumo įvertinimo grafinė-tekstinė išraiška				+
RS klaidų sąrašo pateikimas				+

Atlikus eksperimentą, galima nustatyti sukurtos sistemos veikimo ir funkcionavimo kokybę, kuri pateikta penkiabalėje įvertinimo sistemoje (5 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.

5 lentelė. Sistemos veikimo ir funkcionavimo kokybės įvertinimas

	Įvertinimas balais				
Vertinimo kriterijai	**1**	**2**	**3**	**4**	**5**
Sistemos išbaigtumas			+		
Automatinis RS pakankamumo įvertinimas				+	
Rezultatų pateikimas grafine ir tekstine forma			+		
Klaidų sąrašo pateikimas vartotojui				+	
Įrankio integracija į ODRES metodą			+		

6. Darbo rezultatai

1. Išanalizuoti reikalavimų specifikacijos kokybės vertinimo kriterijai, metodai taikomi RS kokybės vertinimui bei CASE įrankiai, kuriais galima įvertinti RS kokybę.

2. Apibrėžtos RS pilnumo, pakankamumo, nepakankamumo ir pertekliškumo elementų aibės.

3. Sudaryti funkciniai ir nefunkciniai reikalvimai RS kokybės įvertinimo įrankiui.

4. Suprojektuota RS kokybės vertinimo sistemos architektūra.

5. Realizuotas įrankis Microsoft VISIO 2003 paketu.

6. Atliktas eksperimentinis tyrimas – palygints ODRES metodui sukurtas įrankis su egzistuojančiais CASE įrankiais, galinčiais įvertinti RS kokybę.

7. Išvados

1. Atlikus pasirinktų metodų, priemonių, RS kokybės įvertinimui gauti, analizę, padaryta išvada, kad reikalingas įrankis galintis automatiniu būdu nustatyti sudaromos RS kokybės lygį. Taip palengvinamas vartotojo mechanišku būdu atliekamas RS kokybės patikrinimas.

2. Atlikus eksperimentą nustatyta, kad sukurtu ODRES metodui skirtu įrankiu, galima automatiniu būdu įvertinti RS pakankamumą.

3. Įrankis naudojantis ODRES metodą, gali pateikti RS kokybės įvertinimą grafine ir tekstine forma, bei identifikuoti trūkstamas ar perteklinies vietas metabazėje.

8. Literatūra:

[1] Enterprise Modeling in Practice: Situational Factors and their Influence on Adopting a Participative Approach, A. Persson

[2] Extending the Scope of Informatikon Modelling, A. Janis Bubenko jr

[3] Veiklos modeliavimo metodų analizė reikalavimų specifikavimo aspektu, Tomas Danikauskas, Rimantas Butleris

[4] Requirements Engineering: A Roadmap, Bashar Nuseibeh, Steve Easterbrook

[5] Butkienė R., Butleris R., „Taisyklės kompiuterizuotos IS modelio išsamumui ir minimalumui patikrinti“

[6] Jelena Gasperovič, „Specifikavimo kalbų funkcionalumo vertinimas“

[7] http://www.magicdraw.com/

9. Summary

Research of the possibilities to evaluate the completeness of the requirement specification

ODRES (output driven requirements engineering)method, which is being developed in the department of information systems comprises the following stages: separating out the context, specification of function results, specification of data sources, specification of data source processing, modelling, automatic projectionof interface of information system and prototype.

While creating requirement specification, there arises a need to evaluate the quality of it. In every stage of creating requirement specification an analyst does not know if he or she has got enough information in order to do produce a working prototype of the system. Therefore, in this study the problem of information system is analysed, namely, the problem of how to hallmark the quality of ISRS? How to measure this quality? What must the IS be, so that it is of good quality?

While seeking to evaluate the quality of RS, it is necessary to understant and analyse what is quality on the whole, how it is defined, how it is measured and if it is possible to measure it at all.

In this study methods which are used to measure the quality and the possibilities of CASE tools to evaluate specification quality are analyzed. In the research requirements to evaluate the RS quality have also been established and the experiment that realizes the RS quality evaluation has been done.

10. Priedai

Programos „Įvertinti RS kokybę“ išeities tekstas

```
Private Sub CommandButton1_Click()
prisijungti_db
End Sub

Private Sub CommandButton2_Click()
tikrinti_funkcijas
End Sub

Private Sub CommandButton3_Click()
End
End Sub

Private Sub TextBox1_Change()

End Sub
```

```
Dim klaidos() As String
Public conn As ADODB.Connection
Public kiek_fju As Integer, kiek_rez As Integer, kiek_per As Integer, kiek_srt As Integer
Public iek_ds As Integer, kiek_ds_es As Integer, kiek_es As Integer, kiek_atr As Integer,
kiek_ds_atr As Integer, kiek_r_atr As Integer

Sub TIKRINTI()
UserForm1.Show
'prisijungti_db

End Sub

Public Sub prisijungti_db()
Set conn = New ADODB.Connection
With conn
.Provider = "Microsoft.Jet.OLEDB.4.0"
.ConnectionString = "data source=" & UserForm1.TextBox1.Text
.Open
End With

UserForm1.Label2.Caption = "Prisijungta prie bazės"

End Sub

Public Sub tikrinti_funkcijas()

Dim funkcija As ADODB.Recordset
Dim perejimas As ADODB.Recordset
Dim rezultatas As ADODB.Recordset
```

```
        Dim srautas As ADODB.Recordset
        Dim ds_esybe As ADODB.Recordset
        Dim duom_salt As ADODB.Recordset
        Dim esybe As ADODB.Recordset
        Dim atributas As ADODB.Recordset
        Dim ds_atributas As ADODB.Recordset
        Dim R_atributas As ADODB.Recordset

        Set funkcija = New ADODB.Recordset
        Dim kiek_fju As Integer, kiek_rez As Integer, kiek_per As Integer, kiek_srt As Integer
        Dim kiek_ds As Integer, kiek_ds_es As Integer, kiek_es As Integer, kiek_atr As Integer,
kiek_ds_atr As Integer, kiek_r_atr As Integer

        Dim kiekis(300) As Variant
        Dim i As Boolean

        funkcija.Open "SELECT Funkcija.id FROM Funkcija;", conn
        kiek_fju = skaiciuoti_eilutes(funkcija)
        kiekis(1) = kiek_fju
        funkcija.Close

        kiek_rez = 0
        kiek_per = 0
        kiek_srt = 0
        kiek_ds = 0
        kiek_ds_es = 0
        kiek_es = 0
        kiek_atr = 0
        kiek_ds_atr = 0
        kiek_r_atr = 0
        i = False

        funkcija.Open "SELECT Funkcija.id FROM Funkcija;", conn
        Do Until funkcija.EOF

           Set rezultatas = New ADODB.Recordset
           rezultatas.Open "SELECT Funkcija.id FROM Funkcija INNER JOIN Rezultatas ON
Funkcija.id = Rezultatas.f_id WHERE (((Funkcija.id)=" & funkcija("id") & "));", conn

           Do Until rezultatas.EOF
           i = True
                 Set srautas = New ADODB.Recordset
                 srautas.Open "SELECT Srautas_R.id FROM Rezultatas INNER JOIN Srautas_R
ON Rezultatas.id = Srautas_R.r_id_i WHERE (((Srautas_R.id)=" & rezultatas("id") & "));",
conn

                 If srautas.EOF Then
                    UserForm1.ListBox1.AddItem ("Truksta srauto Nr." & rezultatas("id"))
                 Else
                    Do Until srautas.EOF
                       Set duom_salt = New ADODB.Recordset 'jei rastas srautas tai priskiriamas
naujas sarasas duom_salt
```

```
                duom_salt.Open "SELECT Srautas_R.id FROM Duom_salt INNER JOIN
Srautas_R ON Duom_salt.id = Srautas_R.ds_id_is WHERE (((Srautas_R.id)=" & srautas("id")
& "));", conn
                If duom_salt.EOF Then 'JEI NERA DUOMENU SALTINIU SUSIETO SU
SRAUTU
                   UserForm1.ListBox1.AddItem ("Truksta duom_salt Nr." &
duom_salt("id")) 'pranesimas
                Else  'jei randa susieta saltini su srautu
                   kiek_ds = kiek_ds + 1 ' sumuojami duomenu saltiniai

                   'ieskoti ds_esybes
                    Set ds_esybe = New ADODB.Recordset
                ds_esybe.Open "SELECT DS_esybe.id FROM Duom_salt INNER JOIN
DS_esybe ON Duom_salt.id = DS_esybe.ds_id WHERE (((DS_esybe.id)=" & duom_salt("id")
& "));", conn
                If ds_esybe.EOF Then
                   UserForm1.ListBox1.AddItem ("Truksta ds_esybes Nr." &
duom_salt("id"))
                Else
                   kiek_ds_es = kiek_ds_es + 1

                   'ieskoti esybes

                    Set esybe = New ADODB.Recordset
                esybe.Open "SELECT DS_esybe.id FROM Esybe INNER JOIN DS_esybe
ON Esybe.id = DS_esybe.e_id WHERE (((DS_esybe.id)=" & ds_esybe("id") & "));", conn
                If esybe.EOF Then
                   UserForm1.ListBox1.AddItem ("Truksta esybes Nr." & ds_esybe("id"))
                Else
                   kiek_es = kiek_es + 1

                   'ieskoti atributus

                    Set atributas = New ADODB.Recordset
                atributas.Open "SELECT Atributas.id FROM Esybe INNER JOIN Atributas
ON Esybe.id = Atributas.e_id WHERE (((Atributas.id)=" & esybe("id") & "));", conn
                If atributas.EOF Then
                   UserForm1.ListBox1.AddItem ("Truksta atributo Nr." & esybe("id"))
                Else
                   kiek_atr = kiek_atr + 1

                    'ieskoti ds_atributus

                    Set ds_atributas = New ADODB.Recordset
                ds_atributas.Open "SELECT DS_atributas.id FROM Atributas INNER JOIN
DS_atributas ON Atributas.id = DS_atributas.a_id WHERE (((DS_atributas.id)=" &
atributas("id") & "));", conn
                If ds_atributas.EOF Then
                   UserForm1.ListBox1.AddItem ("Truksta ds_atributo Nr." &
atributas("id"))
                Else
                   kiek_ds_atr = kiek_ds_atr + 1
```

```
                    'ieskoti R_atributus

                     Set R_atributas = New ADODB.Recordset
                    R_atributas.Open "SELECT R_atributas.id FROM Atributas INNER JOIN
R_atributas ON Atributas.id = R_atributas.a_id WHERE (((R_atributas.id)=" & atributas("id") &
"));", conn
                    If R_atributas.EOF Then
                       UserForm1.ListBox1.AddItem ("Truksta R_atributo Nr." &
atributas("id"))
                    Else
                       kiek_r_atr = kiek_r_atr + 1

                    R_atributas.MoveNext
                    End If

                    ds_atributas.MoveNext
                    End If

                    atributas.MoveNext
                    End If

                    esybe.MoveNext
                    End If

                    ds_esybe.MoveNext
                    End If

                    duom_salt.MoveNext 'pereinama i kita irasa
                    End If 'baigiama tikrinti ar srautas turi duom_salt

                  kiek_srt = kiek_srt + 1
                  srautas.MoveNext

                  Loop
               End If

        kiek_rez = kiek_rez + 1
        rezultatas.MoveNext

        Loop

        If i = False Then
        Set perejimas = New ADODB.Recordset
              perejimas.Open "SELECT Funkcija.id FROM Funkcija INNER JOIN
Perejimas_Et ON Funkcija.id = Perejimas_Et.f_id WHERE (((Funkcija.id)=" & funkcija("id") &
"));", conn
               If perejimas.EOF Then
                  UserForm1.ListBox1.AddItem ("Truksta rezultato ar perejimo funkcijos Nr." &
funkcija("id"))
                Else
               Do Until perejimas.EOF
```

```
                kiek_per = kiek_per + 1
                perejimas.MoveNext
            Loop
            End If
        End If
    i = False
    funkcija.MoveNext
    Loop

    'Loop

    Dim sss As Integer
    Dim sumuoti As Double
    sss = kiek_fju + kiek_rez + kiek_per + kiek_ds + kiek_srt + kiek_ds_es + kiek_es +
kiek_atr + kiek_ds_atr + kiek_r_atr
    If sss > 0 Then
    sumuoti = sss / (kiek_fju * 2 + 7 * kiek_rez) * 100
    UserForm1.TextBox2.Text = Str(sumuoti) & " %"
    Else
    UserForm1.TextBox2.Text = "0,00 %"
    End If

    End Sub

    Private Function skaiciuoti_eilutes(ByVal rsADO As ADODB.Recordset) As Integer
    Dim iCount As Integer

      iCount = 0

      If rsADO.EOF Then
        GetADORecordCount = 0
        Exit Function
      End If

      Do While Not rsADO.EOF
        iCount = iCount + 1
        rsADO.MoveNext
      Loop

      skaiciuoti_eilutes = iCount
    End Function
```