KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

INFORMACIJOS SISTEMŲ KATEDRA

Vytautas Paulauskas

Paulius Vaidotas

Duomenų modeliavimo ir schemos tikrinimo metodika

Magistro darbas

Darbo vadovas

Doc. dr. Lina Nemuraitė

Kaunas, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

INFORMACIJOS SISTEMŲ KATEDRA

Vytautas Paulauskas

Paulius Vaidotas

Duomenų modeliavimo ir schemos tikrinimo metodika

Magistro darbas

Recenzentas
Doc. dr. V. Pilkauskas
2007-01

Vadovas
Doc. dr. Lina Nemuraitė
2007-01

Atliko
IFM-4 gr. stud.
Vytautas Paulauskas
Paulius Vaidotas
2007-01-09

Kaunas, 2007

Turinys

1. Įvadas

Informacinės sistemos projektuojamos, kuriamos ir sėkmingai diegiamos jau daugelį metų, pradedant 1970-ųjų pradžia. Per šį laikotarpį IS kūrimo procesas evoliucionavo: laikantis kad ir 10 metų senumo metodikų ir naudojantis to laikmečio įrankiais, būtų sunku sėkmingai sukurti informacinę sistemą, galinčią konkuruoti su šiuolaikiškais kitų gamintojų produktais.

Vienas iš pagrindinių veiksnių, dėl kurių informacinių sistemų kūrimo procesas greitai tobulėja, yra pastovus specifikacijos, projektavimo, realizavimo ir apskritai darbo komandoje proceso tobulinimas. Minėtoms naujovėms galima priskirti tokias, kaip objektiškai orientuotas projektavimas, AGILE projekto valdymo metodai, CASE priemonių naudojimas, ketvirtos kartos programavimo kalbos.

CASE (angl. *Computer Aided Software Engineering*) – tai kompiuterizuota informacinių sistemų inžinerija, kuomet žinios apie norimą sukurti produktą kaupiamos grafiniais vaizdais, įvairiomis diagramomis, o specializuoti CASE įrankiai šias schemas gali paversti pradiniu projekto išeities kodu ar duomenų bazės lentelių struktūros skriptu. Tokie įrankiai palengvina informacinių sistemų kūrimą, jei yra tinkamai naudojami reikalavimų analizės, projektavimo ar realizavimo etapų metu. Automatizuodami dalį operacijų, kaip kad vartotojo poreikių specifikavimą, DB modeliavimą, CASE įrankiai prisideda prie sėkmingo ir spartaus informacinės sistemos sukūrimo.

Šis darbas skirtas pristatyti metodikai, leisiančiai automatizuoti duomenų bazės schemos tikrinimą, bei pristatyti sukurtą įrankį šiai metodikai įgyvendinti. Įrankis leidžia praplėsti esamą CASE įrankį – DB schemos testus galima būtų integruoti į fizinio DB modelio generavimo procesą, taip palengvinant schemos validavimą ir sumažinant klaidų tikimybę dar ankstyvo projektavimo etapo metu.

Darbo analizėje atskleidžiama, kad minėta metodika yra aktuali ir reikalinga – esami CASE įrankiai neįgyvendina automatinio DB struktūros tikrinimo. Be to, duomenų schemos tikrinimo metodiką tikslinga integruoti jau į egzistuojantį CASE įrankį, o ne kurti naują – informacinės sistemos kūrimo metu naudojant kelis panašų funkcionalumą įgyvendinančius CASE įrankius, žmonės priversti dubliuoti informaciją abiejose sistemose, taip pat rankiniu būdu tikrinti gautus rezultatus su siekiamais. Įvertinus šiuos aspektus, buvo pasirinktas atitinkamas sistemos architektūrinis sprendimas, o analizės etapo išvados buvo panaudotos sistemos projektavimo etape.

Norint sėkmingai sukurti kad ir nedidelės apimties ar sunkumo informacinę sistemą, reikia nepagailėti įdirbio ir sukurti tos sistemos projektą. Sistemos projektas tarnauja kaip dokumentacija ir klientui (produkto užsakovui), ir vykdytojo programuotojų komandai. Klientas sistemos projekto dokumentu remiasi sistemos validavimo metu, tikrindamas, ar visos aptartos

funkcijos buvo įgyvendintos, ar vartotojo sąsaja atitinka suplanuotąją. Tuo tarpu sistemos kūrėjų tarpe projektavimo dokumentas turi tapti pagrindas, kurio laikomasi realizuojant sistemą, kad atitinkamų funkcijų realizavimas vyktų kiek galima sklandžiau ir tiksliau.

Duomenų modeliavimo ir schemos tikrinimo metodikos įskiepiui taip pat buvo kuriamas sistemos projektas. Nors šio produkto konkretaus užsakovo nėra, tačiau projektas naudingas, kaip jau buvo minėta, tuo aspektu, jog padės suprogramuoti sistemą daug tiksliau nei programavimas neplanuotai.

Kaip parodė mūsų kuriamos sistemos analizės modelyje sukaupta informacija, duomenų schemos tikrinimo metodikos sukūrimas yra aktualus, kadangi, bent jau kol kas, dar nėra sukurta įrankio, galinčio CASE sistema kuriamą duomenų bazės schemą patikrinti su iš tos schemos gauta fizine DB.

Darbų pasiskirstymas

***Paulius Vaidotas*:**

Analizės metodų alternatyvų įvertinimas, literatūros šaltinių paieška ir apžvalga, projekto tikslo ir kokybės kriterijų įvertis.

Reikalavimų modelio sudarymas.

Dokumento skyriai:

- 2.2 Analizės metodų, priemonių parinkimas;
- 2.3 Pasaulio bei Lietuvos literatūros šaltiniuose pateiktų duomenų bazės schemos tikrinimo metodikų ir įrankių lyginamoji analizė;
- 2.4 Projekto tikslas ir jo pagrindimas, kokybės kriterijų apibrėžimas;
- 3.1.1 Vartotojo panaudojimo atvejų diagramos;
- 3.1.2 Sistemos kontekstinė diagrama;
- 3.1.3 Panaudojimo atvejų specifikacijos;
- 3.1.5 Vartotojo sąsajos modelis;
- 3.1.6 Nefunkciniai reikalavimai;
- 4.1 Sistemos veikimo aprašymas;
- 4.4 Sukurtos sistemos kokybės tyrimas;
- 4.5 Tolimesnio sistemos tobulinimo, plėtojimo galimybės;
- 2 priedas. Vartotojo vadovas;
- 6. Literatūra.

***Vytautas Paulauskas*:**

Tyrimo srities identifikavimas bei duomenų bazių schemų tikrinimo metodikos analizė, projektavimo metodų ir priemonių parinkimas, sistemos architektūros planavimas, testinio pavyzdžio prototipo sudarymas.

Reikalavimų modelio tobulinimas, sistemos realizacijos alternatyvų analizė. Sistemos projekto dalis.

Suprojektuotos sistemos realizavimas.

Dokumento skyriai:

- 2.1 Tyrimo sritis, objektas ir problema;
- 2.5 Projektavimo metodų, priemonių parinkimas;
- 2.6 Kompiuterizuojamos sistemos varianto parinkimas;
- 3.1.4 Dalykinės srities klasių diagrama;
- 3.2.1 Projekto tikslas;
- 3.2.2 Sistemos panaudojimo atvejų realizacijos;
- 3.2.3 Sistemos architektūra;
- 3.2.4 Sistemos elgsenos modelis;
- 3.2.5 Realizacijos modelis;
- 3.2.7 Reikalavimai sistemos funkcionalumo palaikymui;
- 4. Eksperimentinis tyrimas;
- 6. Literatūra;
- 8. Santrauka anglų kalba.

2. Duomenų bazių schemų tikrinimo metodų ir įrankių analizė

2.1. Tyrimo sritis, objektas ir problema

Tyrimo sritis – duomenų bazės schemos tikrinimo algoritmai ir įrankiai, panagrinėjant ir platesnę sritį – programinės įrangos testavimo metodus ir rekomendacijas.

Objektas – duomenų modeliavimo ir schemos tikrinimo metodika.

Problemos ir temos aktualumo pagrindimas:

- Dabartiniuose CASE įrankiuose galima generuoti duomenų bazių schemas, bet nėra esamos ar naujai kuriamos duomenų bazės schemos tikrinimo funkcijų. CASE įrankių apžvalga ir jų galimybių analizė pateikta žemiau šiame skyriuje;
- Norint įsitikinti, kad sukurta duomenų bazės schema leidžia saugoti korektiškus duomenis apie dalykinę sritį, ją reikia ištestuoti su tikrais arba sugeneruotais dalykinės srities duomenų rinkiniais, apimančiais galimas duomenų variantų aibes;
- Projektuotojai ir testuotojai, norėdami įsitikrinti duomenų bazės validumu, turi naudotis keliais CASE įrankiais vienu metu, arba šalia CASE įrankio naudoti savo parašytą programinį sprendimą.

Praktikoje ne visada lengva lygiagrečiai naudoti kelis projektavimo automatizavimo įrankius, nors šių įrankių taikymo sritys būna glaudžiai susijusios (konkrečiai – darbas su reliacinėmis DB). Dauguma rinkoje esančių CASE įrankių apriboti vien SQL kalbos skriptų generavimu ir atvirkštine inžinerija, DB schemų perteikimu vaizdinėmis diagramomis, arba duomenų generavimu apkrovos testams (angl. *Stress tests*). CASE įrankių įvairovė ir jų realizuojamos funkcijos apibendrinamos [8]. Tik dalis šių įrankių turi įgyvendintas kelias galimybes vienu metu, pvz. SQL skripto generavimo, DB schemos vizualizavimo ir dokumentacijos sudarymo galimybes.

Remiantis minėtu pavyzdžiu, vartotojas, norėdamas sukurti duomenų bazės schemą (DDL), ją įdiegti realioje duomenų bazių valdymo sistemoje (DBVS), ir ištestuoti, ar tikrai sukurta schema atitinka visus apribojimus, kurie buvo numatyti schemos projektavimo metu, yra priverstas naudotis keliais CASE įrankiais, arba CASE įrankiu ir savo sukurtu programiniu sprendimu. Tai sukelia tam tikras problemas: vartotojas turi arba rankiniu būdu tikrinti informaciją, gautą testavimo metu, su DB schemos standartu, arba pakartotinai į testavimo įrankį suvesti DB schemą, kuri buvo naudojama CASE priemonėje, sugeneravusioje fizinę DB struktūrą.

Šiame darbe pateikta metodika, leisianti fizinę duomenų bazės schemą tikrinti naudojant tuos pačius metaduomenis, kurie naudojami DB modeliavimo metu. Toks sprendimas padės sujungti duomenų bazių modeliavimo uždavinius, kurie dabartiniais CASE įrankiais turi būti atliekami atskirais etapais.

2.2. Analizės metodų, priemonių parinkimas

Prieš nusprendžiant, kokiais metodais bus specifikuojami darbe keliami uždaviniai, reikia rasti pagrindinius kriterijus, kurie leistų palyginti kelias reikalavimų analizės ir projektavimo strategijas. Remiantis šiais kriterijais, bus parinktas tinkamiausias metodas darbo analizei ir projektavimui. Svarbiausi kriterijai šiam projektui:

- vartotojo poreikių pateikimo galimybės;
- metodą palaikančių įrankių įvairovė;
- galimybė dirbti su aukšto lygio programavimo kalba, palaikančia automatinį atminties valdymą (pvz. .NET platformos kalbos, Java. Automatinio atminties valdymo privalumai ir išsamesnis programavimo kalbų sąrašas pateiktas [12]);
- teorinė ir praktinė patirtis naudojant šį metodą.

Vadovaujantis šiais kriterijais, bus analizuojami UML (angl. *Unified Modeling Language*) ir struktūrinio projektavimo metodai. Metodų palyginimai pateikti 2.1 lentelėje.

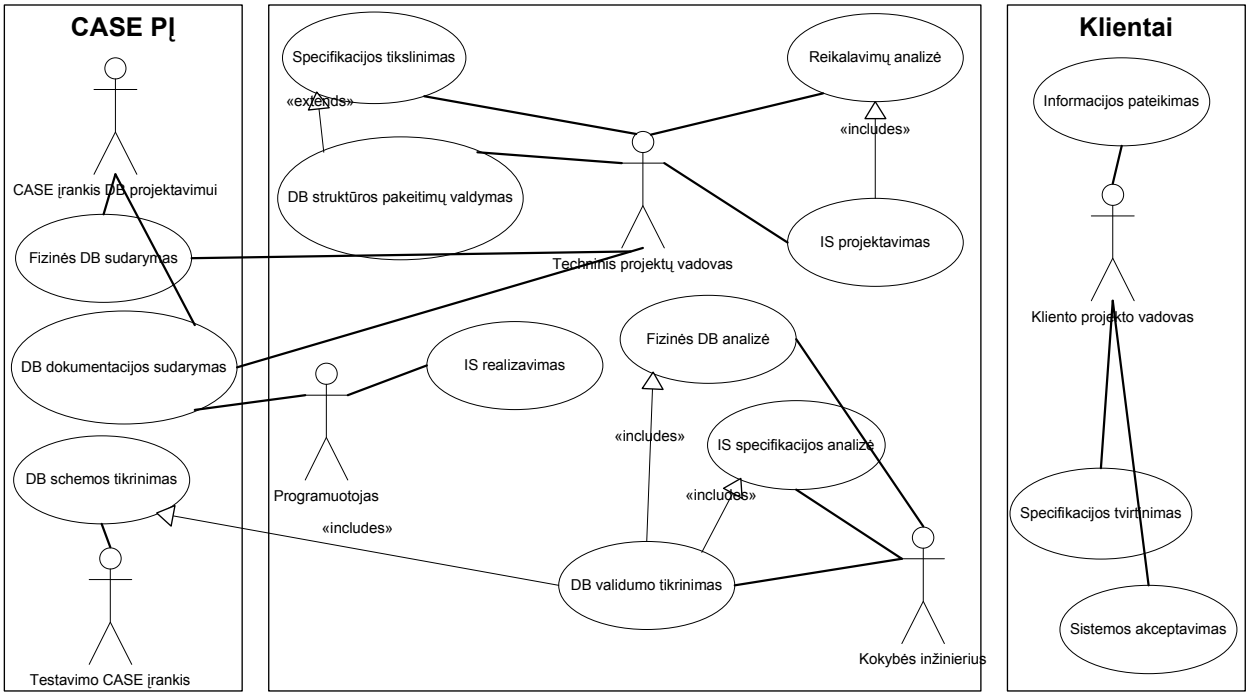
2.1 lentelė. UML ir struktūrinio projektavimo palyginimas

	Unified Modeling Language	**Struktūrinis projektavimas**
Vartotojo poreikių pateikimo galimybė	Pritaikyta specifikuoti vartotojo poreikius	Apibrėžia sistemą labiau iš techninės srities, diagramos nelabai tinkamos pateikti galutiniam vartotojui, t.y. klientui
Įrankių pasiūla	Platus pasirinkimas, populiariausi: MagicDraw UML, Rational Rose, Microsoft Office Visio	Pakankamas pasirinkimas, pvz.: Oracle Workflow, Provision Workbench, Software through Pictures
Aukšto lygio programavimo kalbų palaikymas	Kai kurie įrankiai palaiko JAVA, C#, kitas .NET kalbas	Kai kurie įrankiai palaiko JAVA
Patirtis naudojant metodą	Didelė	Maža

Apibendrinus reikalavimų analizės ir projektavimo specifikavimo metodų privalumus ir trūkumus, šiame darbe nuspręsta naudotis UML standartu. Analizės metu diagramos bus kuriamos vienu iš populiariausių bei lengvai prienamu UML paketu – Microsoft Office Visio 2003 [11].

2.2.1. Duomenų bazės schemos tikrinimo sąveikų modelis

Veiklos sąveikų modelis patogus tuomet, kai norima suprasti kompiuterizuojamos dalykinės srities sąveiką su išoriniais veikėjais ir kitais panaudojimo atvejais. Kaip jau minėta skyriaus įvade (2.1 skyrius), šiame darbe nagrinėjami duomenų schemos sukūrimo, duomenų modeliavimo ir schemos tikrinimo uždaviniai. Dalykinę sritį, kurioje bus naudojami įrankiai, sukurti vadovaujantis šia metodika, patogiausia nagrinėti, vaizduojant procesus tam tikros įmonės mastu. Įsivaizduojamosios įmonės veiklos sąveikų diagrama pateikta 2.1 paveiksle.



2.1 pav. Duomenų bazių schemų tikrinimo veiklos sąveikų modelio diagrama

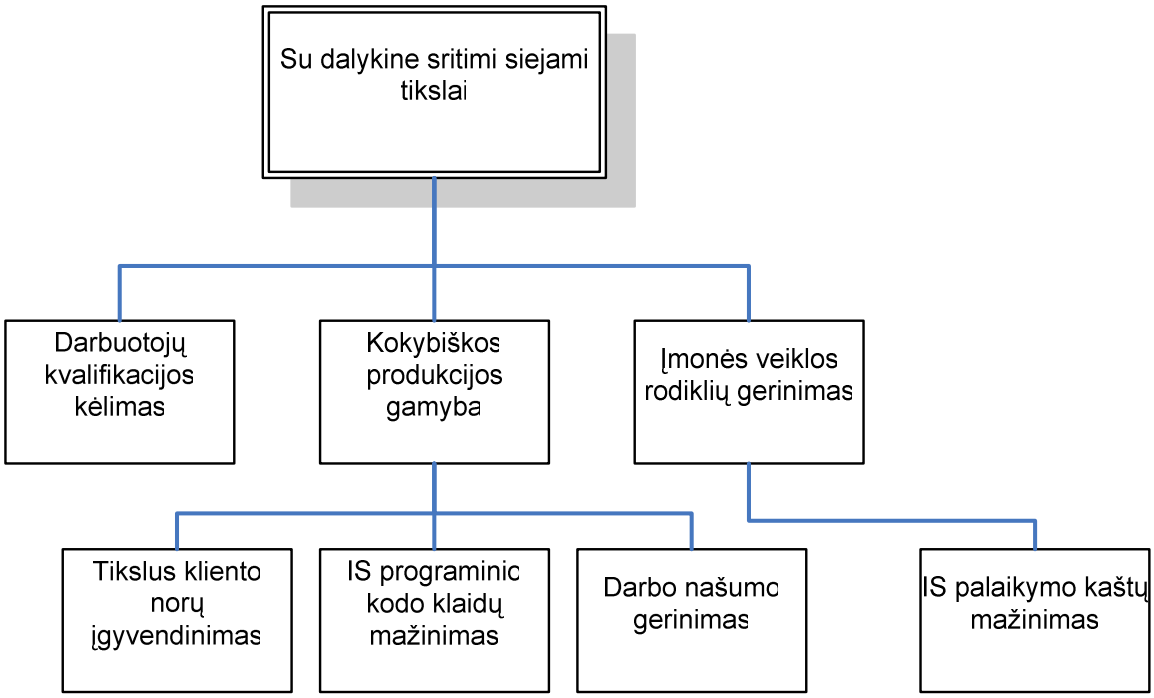
Vidiniai sistemos aktoriai yra „techninis projektų vadovas“, „kokybės inžinierius“ ir „programuotojas“. Būtent šie veikėjai kuria informacines sistemas pagal kliento poreikius, ir savo darbe naudojasi CASE įrankiais.

Kokybės inžinierius, norėdamas patikrinti fizinės duomenų bazės validumą ir atitikimą specifikacijai, turi pasinaudoti dviem CASE įrankiais (pateiktais gamintojo ar sukurtais specifiniam uždaviniui), vienas kurių generuoja fizinę DB struktūrą ir gali jau esamą schemą dokumentuoti, kitas – tikrina fizinės DB schemos validumą. Tokiame darbe iškyla žmogiškų klaidų tikimybė: šalia dviejų CASE įrankių, darbuotojas turi vadovautis dar ir IS specifikacijos projektu, tuo pačiu pastebėtas klaidas ir neatitikimus žymėtis sistemos testavimo dokumente.

Siekiant informacines sistemas kurti kokybiškai ir neviršijant numatytų kaštų, ši organizacijos veiklos dalis turi būti pakeista. Likusius darbo uždavinius atspindi tikslų modelis.

2.2.2. Duomenų bazės schemos tikrinimo tikslų modelis

Veiklos tikslų modelis, pateiktas diagramoje žemiau, sudarytas taip pat jau minėtai įsivaizduojamai įmonei. Pateikti tik įmonės tikslai, kurie susiję konkrečiai su informacinių sistemų kūrimu, manant, kad tokie tikslai turėtų būti siektini kompanijai, norinčiai kurti patikimus ir konkurencingus programinius sprendimus.



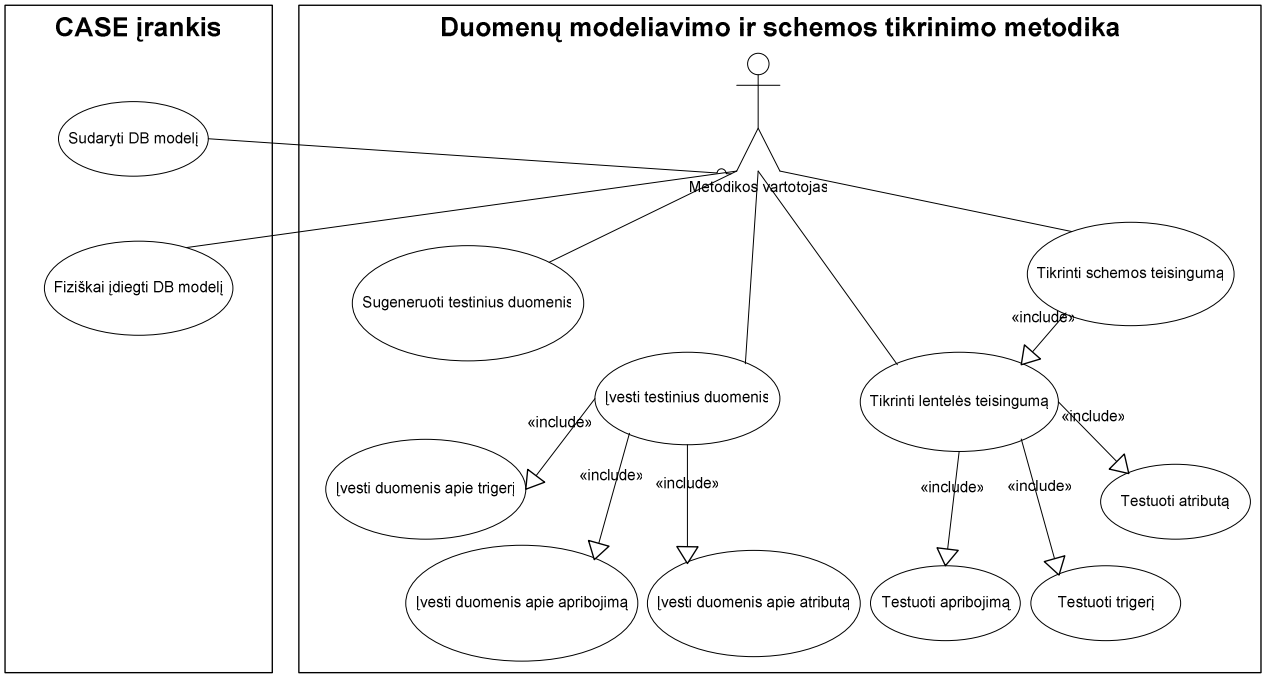
2.2 pav. Duomenų bazių schemų tikrinimo tikslų modelio diagrama

Tinkamos metodikos, ir ja remiantis patobulinto CASE įrankio dėka, įmonė galėtų sumažinti programinio kodo klaidas bei palengvinti IS palaikymo kaštus – laiku diagnozavus duomenų schemos neatitikimus standartams (pvz. kliento klaida, ar techninės įrangos problemų sukeltas neplanuotas sistemos persikrovimas), tokie nesklandumai būtų pašalinami greičiau ir efektyviau.

2.2.3. Duomenų bazės schemos tikrinimo panaudojimo atvejų modelis

Šiame analizės etape jau galima lengvai įvardinti funkcijas, kurias reikia numatyti kuriamoje metodikoje, bei kurias turėtų įgyvendinti pasirinkto CASE įrankio įskiepis. Panaudojimo atvejų diagrama pateikta 2.3 pav. Posistemyje „CASE įrankis“ pavaizduoti panaudojimo atvejai, kurie metodikoje nebus realizuoti, kadangi jie yra įgyvendinti pačiame CASE pakete. Duomenų modeliavimo ir schemos tikrinimo metodika apima testinių duomenų generavimą (kurie reikalingi norimam testui atlikti), taip pat – rankinį duomenų generavimą, bei schemos arba atskirų schemos elementų tikrinimą.

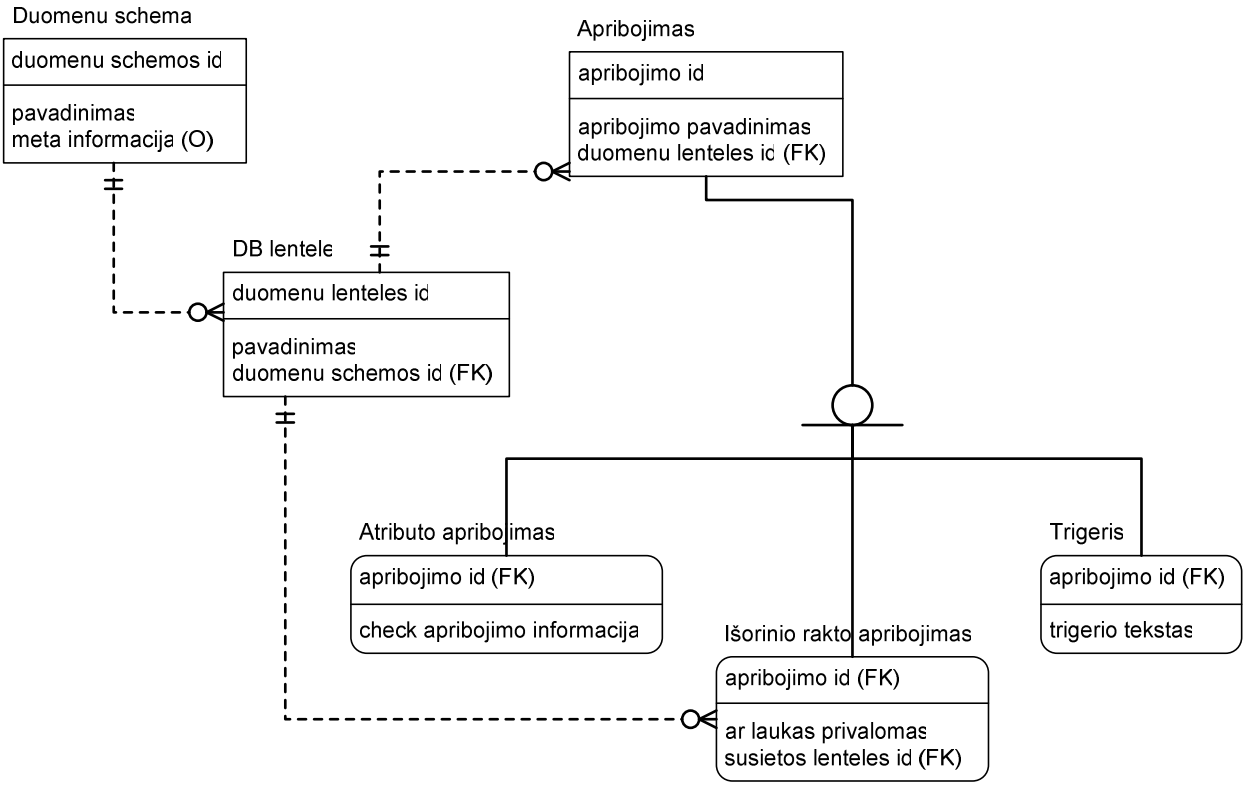
Pradiniame analizės etape numatoma, jog bus tikrinami FOREIGN KEY, NOT NULL, UNIQUE, reikšmių apribojimai (angl. *CHECK constraints*), bei apribojimai, realizuojami DB trigeriais. Panaudojimo atvejai „Įvesti testinius duomenis“ bei „Tikrinti lentelės teisingumą“ patikslinti konkrečiais veiksmais, tuo pabrėžiant, jog sistema atliks testus, norėdama patikrinti CHECK, NOT NULL apribojimus, UNIQUE, FOREIGN KEY integralumo užtikrinimo taisykles ir trigerių veikimą.



2.3 pav. Duomenų bazių schemų tikrinimo panaudojimo atvejai

2.2.4. Duomenų bazių schemų tikrinimo objektų modelis

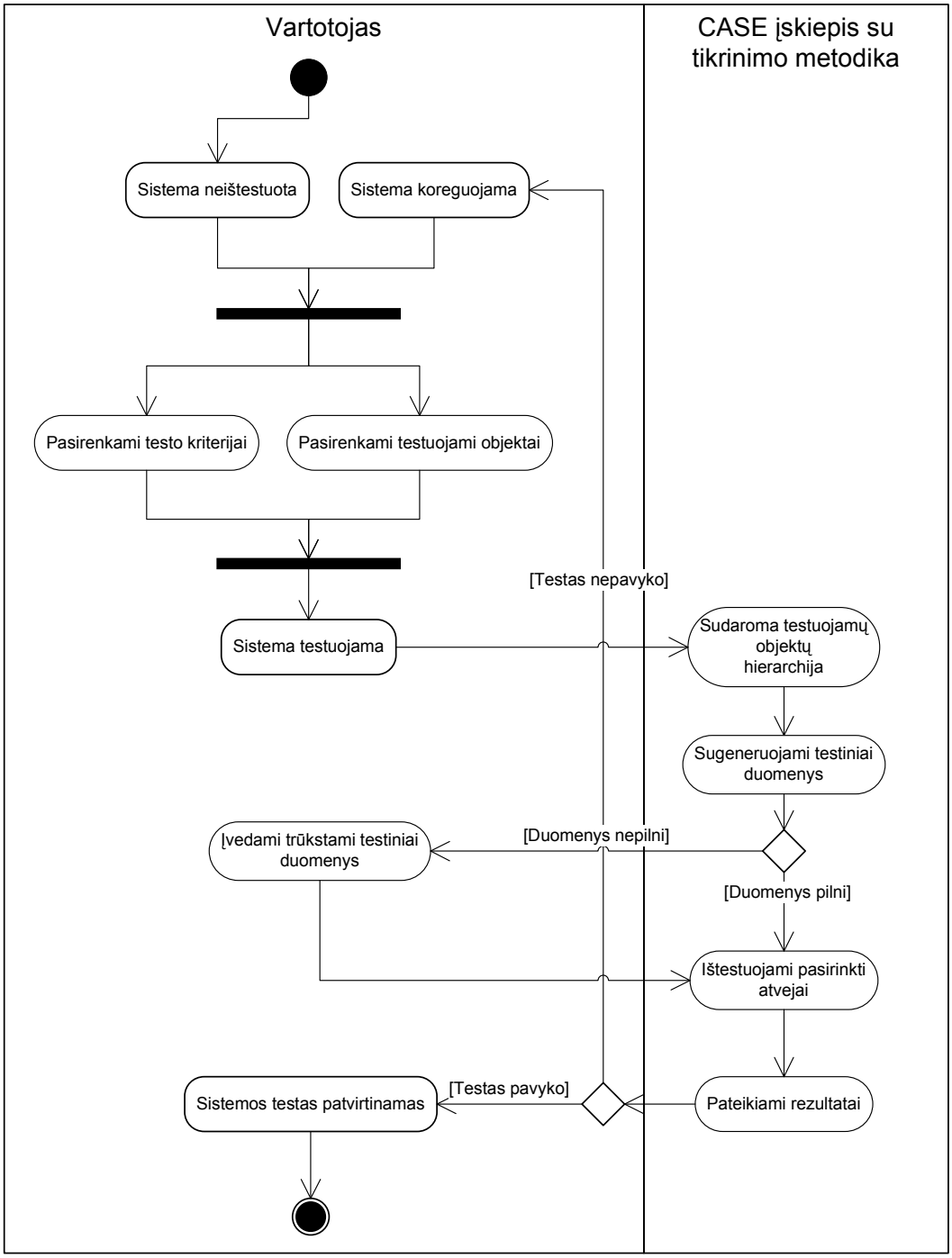
Dalykinės srities objektai sudaryti remiantis aukščiau pateiktu panaudojimo atvejų modeliu. Dalykinės srities objektai pateikti ER (angl. *Entity Relational*) modeliu 2.4 paveiksle. Reikia pabrėžti, jog šis modelis nėra išbaigtas ar galutinis. Modelis tikslinamas metodikos projektavimo etape. Šiame etape dalis modelio esybių yra tik konceptai, pvz. esybės „Duomenų schema“ atributas „meta informacija“ nėra konkretus, kadangi, dar neišnagrinėjus architektūros alternatyvų, nėra galutinai aišku, kokiame formate (XML, vidinė ar tam tikra išorinė DB, specifinės duomenų struktūros) bus saugoma DB lentelės meta informacija.



2.4 pav. Duomenų bazių schemų tikrinimo objektų ER modelis

2.2.5. Duomenų bazių schemų tikrinimo procesų modelis

Duomenų bazių schemų tikrinimo metodikos procesai yra lengvai suprantami ir aiškiai apsibrėžti ankstesniuose analizės etapuose. Aiškiausiai veiklos procesus galime įsivaizduoti kaip veiklos diagramą, kurioje dalyvautų vartotojas ir CASE sistemos įskiepis su įdiegta duomenų schemos tikrinimo metodika. Vartotojas šiame įskiepyje pasirinktų, kokias duomenų schemos lenteles nori tikrinti, ir pradėtų testą. Tuomet CASE įrankis pagal metodiką atrinktų, nuo kurios iš pasirinktų lentelių pradėti testavimą, kokius duomenis bandyti rašyti, ir kaip pateikti rezultatus vartotojui (2.5 pav). Visus duomenis, reikalingus testui, CASE įrankis turi bandyti sugeneruoti automatiškai. Tik trūkstant tam tikrų duomenų, vartotojo paprašoma įvesti reikiamus duomenis, po įvestų duomenų patvirtinimo sistema tęsia įprastinę testavimo procedūrą.

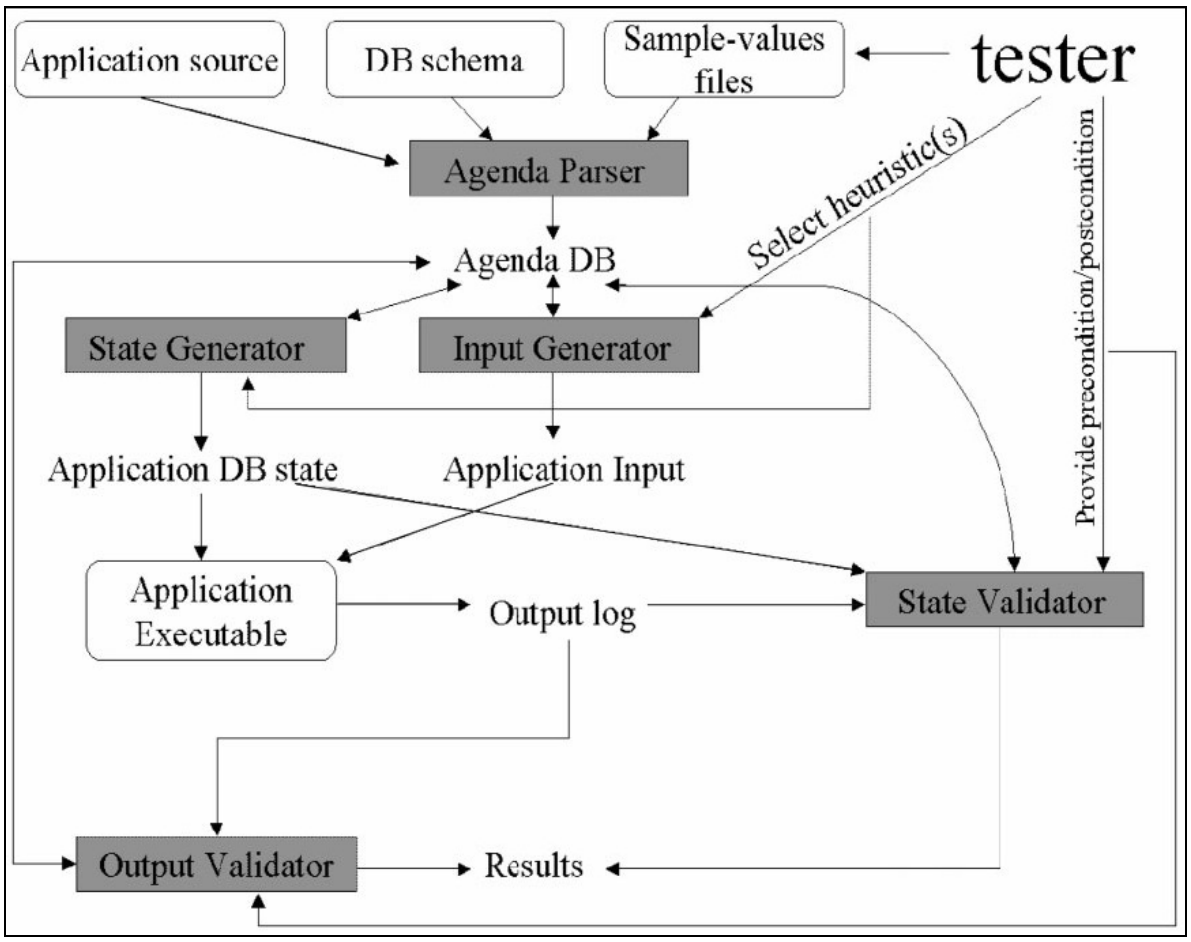


2.5 pav. Procesų modelis, pateiktas veiklos diagrama

2.3. Pasaulio bei Lietuvos literatūros šaltiniuose pateiktų duomenų bazės schemos tikrinimo metodikų ir įrankių lyginamoji analizė

Straipsnyje [3] aprašomas AGENDA – testavimo įrankių rinkinys. Šio straipsnio medžiaga naudinga, kadangi pateikia vieną iš duomenų bazės testavimo metodikos įgyvendinimo alternatyvų. Šiame straipsnyje siūloma duomenų bazės schemą testuoti remiantis jos būsenomis.

Testuojant DB būsena prieš ir po vartotojo operaciją sudaro svarbią reikšmę, kaip ir vartotojo dalyvavimas įvedime bei sistemos dalyvavimas išvedime. Yra pristatoma DB testavimo įrankių rinkinio struktūra.



2.6 pav. AGENDA įrankio architektūra

AGENDA kaip įėjimą paima duomenų bazės, kurioje taikomoji programa veikia, schemą; programos išeities kodą; pavyzdinius failus, kuriuose yra rekomenduojamos atributų reikšmės. Vartotojas pasirenka testavimo euristiką bei suteikia informaciją apie testavimo atvejų elgseną. Naudodama šią informaciją, AGENDA suformuoja DB, sugeneruoja taikomosios programos įėjimus, įvykdo taikomąją programą, naudodamasi šiais įėjimais, bei patikrina DB būsenų bei programos išėjimo teisingumą.

AGENDA veikimas pagrįstas penkiais pagrindiniais komponentais, kurių pagalba ir vykdomas testavimas.

Pagrindiniai AGENDA DB testavimo įrankių rinkinio komponentai:

- *Agenda Parser* – šis komponentas atrenka tinkamą informaciją iš DB schemos, taikomosios programos užklausas, pateiktus pavyzdinius failus, bei padaro visa tai prieinamais kitiems keturiems komponentams. Tai atliekama sukuriant AGENDA DB, šiose DB saugoma atrinkta informacija. AGENDA DB yra naudojama ir/ar modifikuojama likusių keturių komponentų;
- *State Generator* - antrasis komponentas naudoja DB schemą kartu su pavyzdinių failų informacija apie atributų reikšmes, bei sukuria DB lenteles su duomenimis,

tenkinančiais reikiamus apribojimus. Komponentas išskiria informaciją apie lenteles, atributus, apribojimus, naudojamus programiniame sprendime, bei pavyzdinius duomenis iš AGENDA DB bei sukuria pirminę taikomosios programos DB būseną. Euristika, aprašomą detaliau, naudojama kontroliuoti programos DB būsenai bei įėjimams;

- *Input Generator* – trečiasis komponentas generuoja įėjimo duomenis, kurie pateikiami taikomajai programai. Duomenys yra sukuriami naudojantis sugeneruota Agenda Parser bei State Generator komponentų informacija, kartu su informacija, paimta išanalizavus taikomosios programos SQL užklausą. Pvz.: jei dvi eilutės su identiškais atributais yra generuojamos kažkuriai lentelei siekiant ištestuoti ar programa teisingai reaguoja į dubliavimą, informacija yra registruojama norint nustatyti atributo reikšmę, ir naudojama sudarant rekomenduojamus įėjimus testuotojui. Informacija, išgauta išeities kodo analizės metu, taip pat gali būti naudinga rekomenduojant įėjimus, kuriuos testuotojas turėtų naudoti taikomojoje programoje. Naudojantis AGENDA DB kartu su testuotojo euristikos pasirinkimais, Input Generator sugeneruoja programos įėjimo parametrus su realiomis reikšmėmis, taip sugeneruodama testinius įėjimus;
- *State Validator* – ketvirtasis komponentas tiria, kaip DB, naudojamos programoje, būsena keičiasi vykdant testavimą. Jis automatiškai registruoja pasikeitimus duomenų bazės lentelėse ir pažymi būsenos pasikeitimus;
- *Output Validator* – penktasis komponentas nuskaito PĮ išėjimus nei tikrina užklausos pradines sąlygas su galutinėmis sąlygomis kurios buvo sugeneruotos įrankio ar pateiktos testuotojo.

UML 2.0 standarto testavimo modelis apibrėžtas specifikacijoje [2]. Toliau naudojama šioje UML testavimo specifikacijoje aprašyta testavimo klasių dalis, atitinkanti šio darbo tikslus. Žemiau pateikiamos svarbiausios idėjos iš minėto dokumento, aktualios kuriamajai metodikai.

Apžvalga

UML testavimo modelis apibrėžia kalbą, skirtą testavimo sistemų projektavimui, analizei, vizualizacijai, kosntravimui bei dokumentavimui. Ši kalba gali būti naudojama su daugeliu technologijų, skirtų sistemų testavimui. UML testavimo forma taip pat gali būti naudojama viena ar integruota į kitą sistemą.

UML testavimo modelio pagrindas yra UML 2 Superstructure Adopted Specification. Formoje naudojamas metamodelinis UML variantas. Jo architektūros principai:

- UML integracija

- Pakarotinio panaudojimo galimybė bei minimalizmas

Kadangi UML testavimo profilis yra ir UML profilis, jis paveldi UML charakteristikas:

- Sluoksniavimas
- Paketinė struktūra
- Plėtojimo bei adaptyvumo galimybė

UML testavimo modelio struktūra

UML testavimo modelis yra padalintas į keturias logines konceptų grupes:

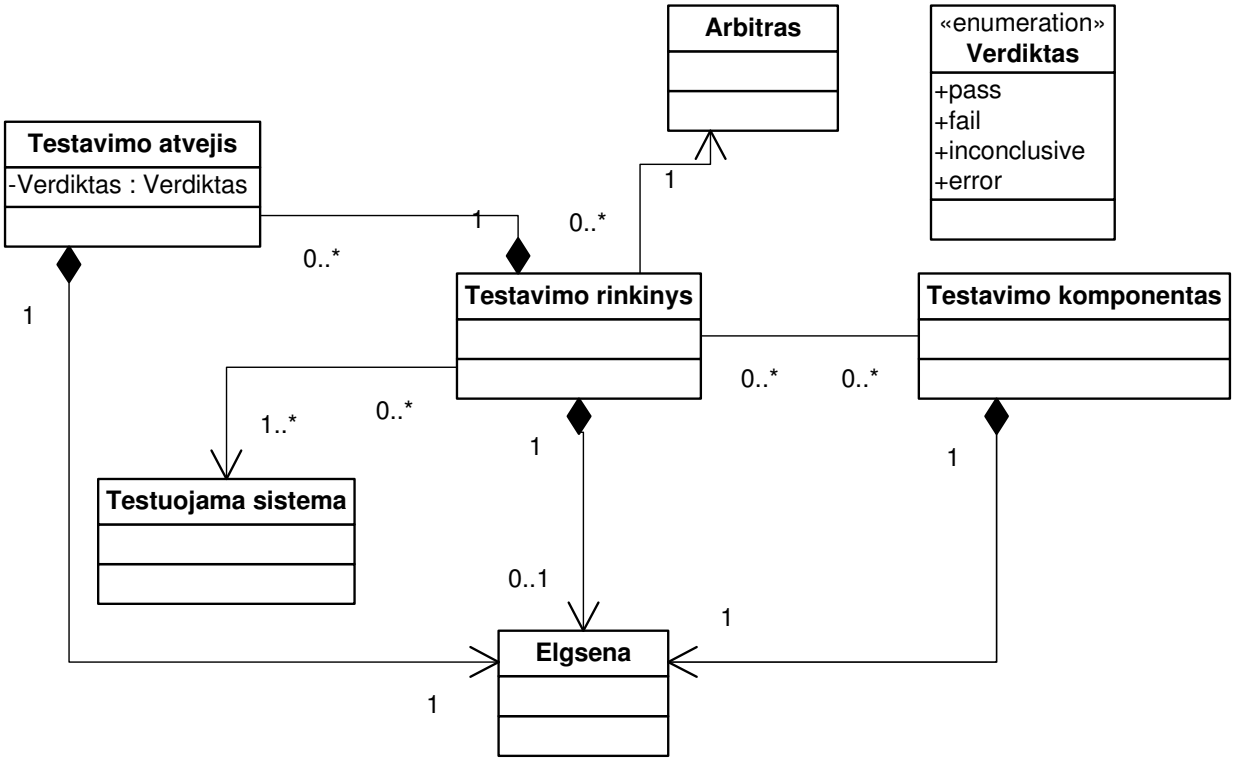
- Testavimo architektūra, apibrėžianti konceptus, susijusius su testavimo struktūra ir testavimo nustatymais;
- Testavimo duomenys, apibrėžiantys konceptus, skirtus testavimo duomenims, naudojamiems testavimo procedūruose;
- Testavimo elgsena, apibrėžianti konceptus, susijusius su dinaminiais testavimo procedūrų aspektais;
- Testavimo laikas, apibrėžiantis konceptus testavimo procedūrų laiko kiekiui.

UML testavimo modelis yra specifikuotas:

- Terminologija UML testavimo formos konceptų supratimui;
- UML 2.0 metamodelio apibrėžimas UML testavimo formoje;
- MOF (angl. *Meta Object Facility*) modelio, skirto leisti UML testavimo formai naudotis testavimo forma nepriklausomai nuo UML, apibrėžimas;
- Pavyzdžių, skirtų UML testavimo formos sisteminių bei komponentinių lygių naudojimui suteikimas.

Visos dalys formuoja UML testavimo modelio apibrėžimą.

Remiantis šiuo dokumentu, supaprastinome ir savo uždaviniui pritaikėme UML testavimo modelį. Supaprastinto testavimo modelio diagrama pateikta 2.7 paveiksle.



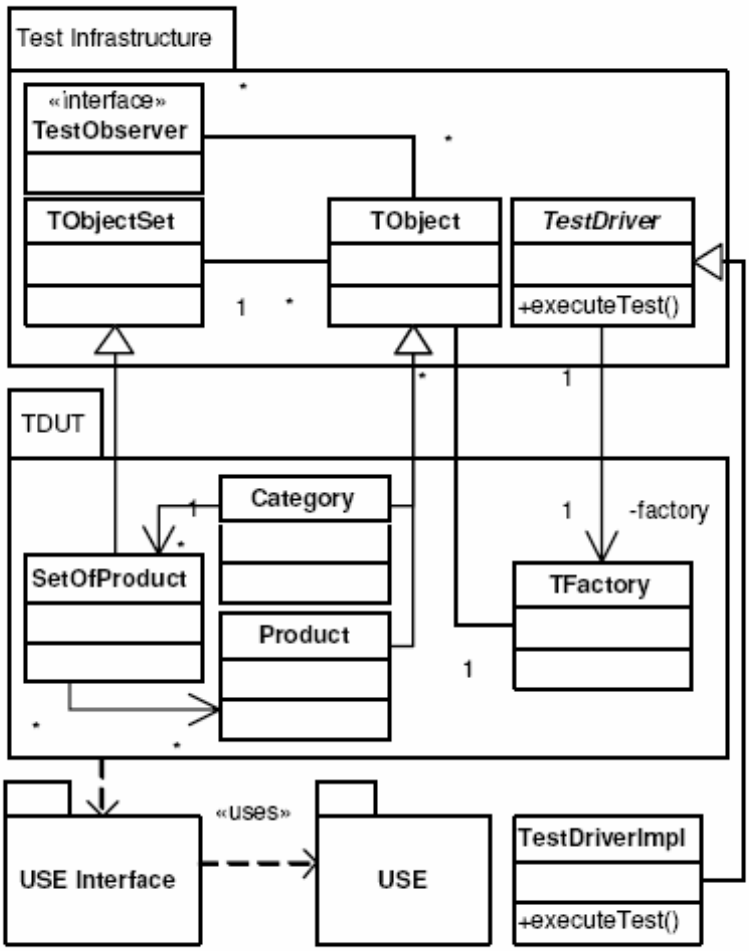
2.7 pav. Supaprastinta UML testavimo modelio klasių diagrama

Žemiau pateikti elementų apibrėžimai.

2.2 lentelė. UML Testing Profile objektų apibūdinimas

Objektas	Apibūdinimas
Elgsena	Ši klasė atitinka testavimo atvejo, atvejų rinkinio ar sistemos elgesį. Tai yra aukšto lygio konceptas, leidžiantis kreiptis į žemiau paminėtus elementus programiškai.
Testavimo rinkinys	Į testavimo rinkinį paprastai įeina 0 arba daugiau testavimo atvejų.
Testuojama sistema	Tai juodosios dėžės abstrakcija sistemos, kurią bandome testuoti.
Testavimo komponentas	Komponentas (komponentai), per kuriuos realizuota testuojamos sistemos elgsena.
Arbitras	Esybė, kuri, pasibaigus testų rinkiniui, galutinai nusprendžia, ar testai pavyko.
Testavimo atvejis	Testavimo atvejis – tai aibė elgsenų, vykstančių testuojant sistemą. Kiekvienas testavimo atvejis priklauso kažkokiam testavimo rinkiniui.
Verdiktas	Pagalbinis tipas, naudojamas Arbitro klasės. Turi sekančias reikšmes: testas pavyko (*pass*); testas nepavyko (*fail*); nustatyti neįmanoma (*inconclusive*); įvyko klaida (*error*).

[7] straipsnis aprašo įrankio, automatiškai generuojančio testus UML klasių diagramoms, projektavimo ir realizacijos ypatybes. Šio straipsnio idėjos, o konkrečiai – testavimo komponentų objektinis modelis – tinka nagrinėjamai dalykinei sričiai. Minėtame straipsnyje pateiktas komponentų objektinis modelis patvaizduotas 2.8 pav.



2.8 pav. Testavimo komponentų klasių diagramos alternatyva

Svarbiausi paketai šiame modelyje yra „Test Infrastructure“ ir „TDUT“, vaizduojantys atitinkamai bazinius testavimo modulius ir konkrečią sistemą, kuri bus testuojama. „USE Interface“ ir „USE“ paketai vaizduoja nemokamą JAVA kalbai pritaikytą įrankį, naudojamą straipsnio metodologijoje, todėl šie du paketai nėra aktualūs. Likusios programinės klasės apibrėžiamos sekančiai:

- *TestObserver* – klasė, sąveikaujanti su paketo „Test Infrastructure“ vartotojo sąsaja, sugebanti koordinuoti testo vykdymą, o apie klaidas pranešti vartotojams;
- *TObject* – bazinė klasė, iš kurios turi būti kildinamos visos klasės, su kuriomis norima atlikti testavimus;
- *TObjectSet* – jau minėtų *TObject* klasių masyvas. Šis masyvas turi įgyvendinti metodus, skirtus konkretaus elemento įterpimui ir pašalinimui iš masyvo;

- *TestDriver* – abstrakti klasė, turinti abstraktų metodą executeTest(). Norint sukurti konkretų testavimo atvejį, reikia paveldėti *TestDriver* klasę, ir įgyvendinti jos metodą executeTest();
- *TFactory* – taipogi konkrečioje testuojamoje sistemoje įgyvendinama klasė, galinti sukurti testuojamų klasių rinkinius. Paveikslėlyje parodyta *TFactory* implementacija, galinti dirbti su *Category* ir *Product* tipų objektais.

Apibendrinus šiuos literatūros šaltinius, galima sudaryti lyginamąją šaltiniuose minimų metodikų lentelę (2.3).

2.3 lentelė. Lyginamoji metodikų pasaulio literatūros šaltiniuose analizė

Metodas	**Apibendrinimas**	**Naudojamos technologijos**	**Tinkamumas kuriamai metodikai**
[3]: AGENDA duomenų bazės taikomųjų programų testavimui pagal DB būsenas	Apibrėžiama, kaip vykdyti duomenų bazę naudojančių taikomųjų programų testavimą, remiantis DB būsenomis. Testavimo įrankis realizuojamas kaip DBVS modulis	Oracle	Tiesiogiai netinkamas, kadangi siekiama sukurti metodiką testuoti DB schemą su vientisumo apribojimais projektavimo etape, sukuriant testavimo įrankį kaip CASE įrankio papildymą, nepriklausomą nuo DBVS ir nuo taikomųjų programų. Mūsų tikslams galima panaudoti „AGENDA“ testinių duomenų generavimo būdus
[2]: UML testavimo modelis	Pristatoma UML 2.0 standarto posistemė, skirta testavimo praktikos formavimui ir testų klasėms apsibrėžti	UML, pavyzdžiai realizuoti JAVA, JUnit	Tinkamas – testavimo klasių modelis buvo supaprastintas ir pritaikytas kuriamai metodikai
[7]: UML modelių automatinis testavimas	Pateikta testavimo idėja, pristatyti automatizavimo įrankiai, leidžiantys automatizuoti UML klasių testavimą	JAVA	Naudingas – objektiškai orientuotas objektų modelis gali būti naudingas sistemos projektavimo etape

Tačiau nei viename literatūros šaltinyje nebuvo pateikta nuosekli metodika, kaip nuosekliai, sistemiškai organizuoti duomenų bazės schemos su vientisumo apribojimais

patikrinimą. „AGENDA“ literatūros šaltiniuose daroma prielaida, kad projektuotojas žino, kokiose būsenose turi būti kuriama duomenų bazė ir tikrina, ar ji iš tiesų pereina į norimas būsenas. Mūsų darbe siekiama sukurti duomenų bazę, nepriklausomą nuo taikomųjų programų. Šioje duomenų bazėje turi būti tenkinami apribojimai, kurie galioja visoms leistinoms dalykinės srities būsenoms, nepriklausomai nuo taikomųjų uždavinių. Vadinasi, kokia bebūtų DB įrašų sukūrimo, atnaujinimo ar naikinimo operacija, ji turi tenkinti DBVS sukurtus apribojimus. Specifiniai uždaviniai gali reikalauti specifinių apribojimų, tačiau tokie apribojimai neturėtų būti realizuojami DBVS priemonėmis. DBVS priemonėmis tikslinga realizuoti tik tuos apribojimus, kurie galioja visoms leistinoms dalykinės srities būsenoms. Tokie apribojimai vadinami vientisumo apribojimais ir jų užtikrinimui skiriama mūsų sukurta metodika.

2.4. Projekto tikslas ir jo pagrindimas, kokybės kriterijų apibrėžimas

Pagrindinį projekto tikslą, kuris pristatytas jau anksčiau, galime suskirstyti į potikslius, kurie lengviau atspindės siekiamus su sistema atlikti veiksmus:

- Įsisavinti testavimo metodologijos pateikimo ir realizavimo sprendimus, paminėtus mokslinėje literatūroje;
- Pasirinkti tinkamiausią architektūrinį sprendimą, leisiantį kuriama metodologija išplėsti jau egzistuojantį CASE įrankį, ir tuo pagerinti vartotojų darbo našumą bei sumažinti klaidų tikimybę;
- Sukurti universalų algoritmą, leisiantį duomenų bazės teisingumą tikrinti universaliai, t.y. nekonkrečiai, duomenų struktūrai;
- Praktiškai realizuoti duomenų schemos tikrinimo metodiką, sukuriant pasirinkto supaprastinto duomenų modelio pavyzdinį testą.

Projekto kokybės kriterijai, t.y. nefunkciniai reikalavimai, bus detalizuoti sistemos projektavimo etape. Tačiau jau dabar galima paminėti, kad svarbiausi darbo kokybės kriterijai yra metodikos išbaigtumas ir universalumas, išsami ir korektiška dokumentacija, aprašanti metodiką, bei testinio pavyzdžio išsamumas (t.y. testinis pavyzdys turi apimti pakankamą situacijų aibę).

2.5. Projektavimo metodų, priemonių parinkimas

Siekiame, kad pasirinktas projektavimo metodas padėtų valdyti projektą neapkraunant dirbančių prie jo asmenų, bet tuo pačiu būtų matomi projekto valdymo rezultatai. Kadangi šis darbas nėra skirtas konkrečiai įmonei-užsakovei, todėl manome, jog netikslinga projekto valdymui pasirinkti „sunkias“ projektavimo metodologijas kaip kad RUP (angl. *Rational Unified Process*), MSF (angl. *Microsoft Solutions Framework*) ar kt.

Daug tinkamesnis šiam darbui yra „lengvas“ metodas. Keletas iš žinomiausių tokio tipo metodų, dažnai naudojamų modernių informacinių sistemų kūrime, yra tokie [14]:

- XP (*eXtreme Programming*);
- MSF Agile;
- Adaptive Software Development.

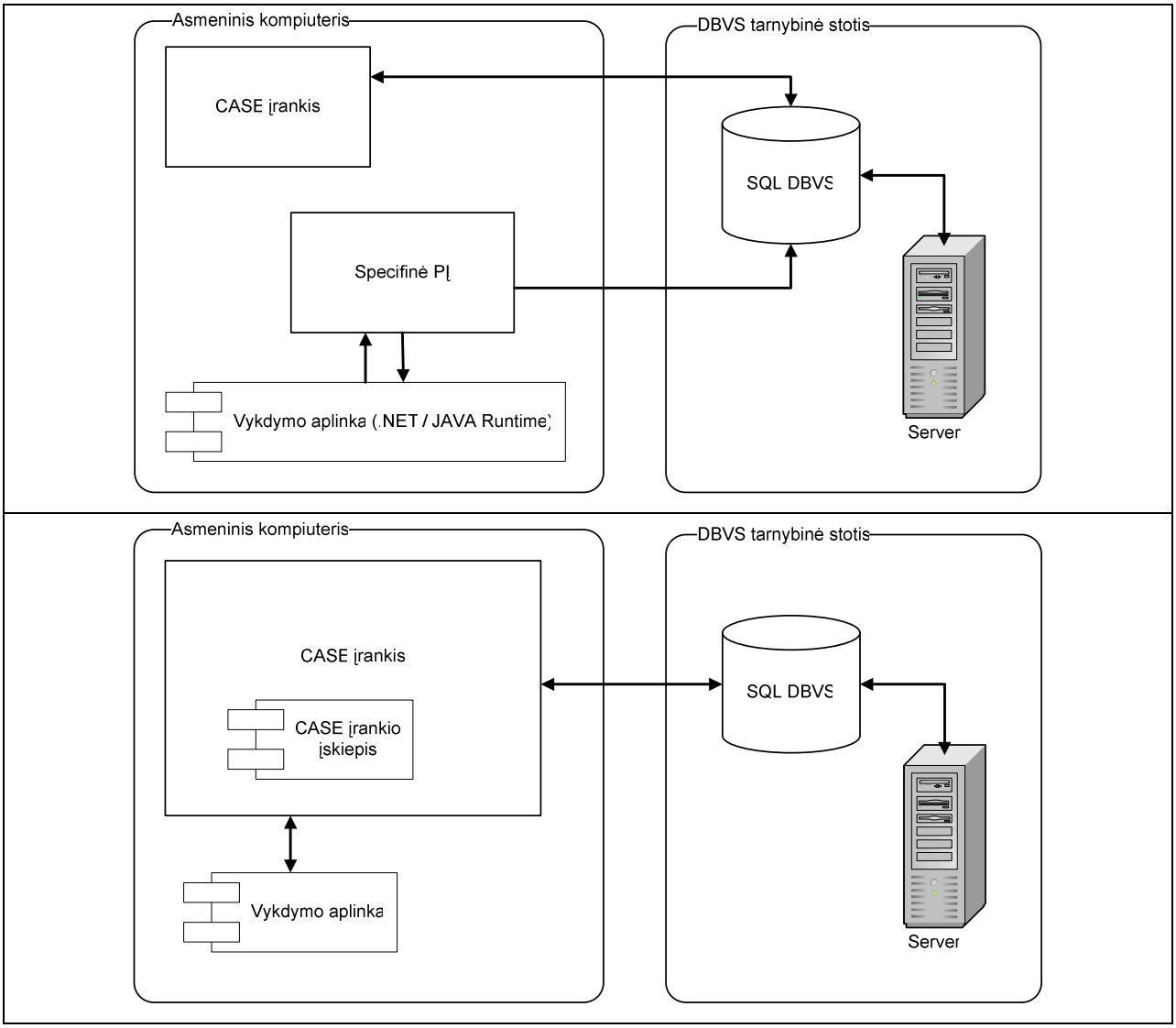
Šiame darbe vadovausimės XP projektavimo metodologija. Taip buvo nuspręsta dėl keleto priežasčių:

- Kaip jau minėta, nėra kuriamas komercinis produktas, turintis įgyvendinti formalius kliento reikalavimus ir kruopščiai dokumentuoti reikalavimų analizės ir projekto valdymo etapus;
- Kadangi kuriamai metodologijai nėra analogų, tikėtina, jog darbo eigoje keisis reikalavimai;
- Prie projekto dirba dviejų žmonių komanda. Abu komandos nariai galės pasinaudoti XP rekomenduojamais sprendimais, kaip kad greitas projekto aptarimas, porinis programavimas ir kt.

2.6. Kompiuterizuojamos sistemos varianto parinkimas

2.6.1. Architektūros alternatyvų analizė

Prieš pasirenkant konkrečią sprendimo architektūrą, reikia išanalizuoti sistemos realizacijos galimybes kuriant specifinį programavimo produktą, ir realizacijos galimybę praplečiant jau egzistuojantį CASE įrankį.



2.9 pav. Architektūros alternatyva nediegiant CASE įrankio įskiepio (viršuje) ir praplečiant CASE įrankį (apačioje)

Pirmasis nepatogumas sistemos testavimo atvejį realizuoti atskirai nuo CASE įrankio yra diegimo sudėtingumas. Vartotojas, norėdamas įsidiegti duomenų schemos tikrinimo įrankį, turės šalia savo naudojamo projektavimo automatizavimo įrankio diegti papildomą programinę įrangą, kuriai taip pat reikės vykdymo aplinkos (.NET Runtime arba JAVA Runtime), kurią, tikėtina, diegti reikėtų taip pat papildomai. Tuo tarpu duomenų schemos tikrinimo metodikos pavyzdį realizavus kaip CASE sistemos įskiepį, šis komponentas betarpiškai sąveikautų su CASE sistemos programiniais interfeisais, ir jam papildomos vykdymo aplinkos nereiktų, nebent jos reikalautų pats CASE įrankis.

Taip pat, naudojantis CASE sistemos įskiepio sprendimu, palengvėja priėjimas prie CASE įrankyje saugomų duomenų schemos meta duomenų, taip eliminuojant duomenų dubliavimą

skirtingose sistemose. Įvertinus šiuos du aspektus, darosi pakankamai aišku, jog ir vartojimo, ir realizavimo prasme patogesnė realizacija kaip CASE sistemos įskiepis.

Siekiant surasti, kokiam CASE įrankiui geriausia realizuoti praplėtimą, buvo palyginti du įrankiai, šiuo metu esantys vieni iš populiariausių – tai MagicDraw UML ir Microsoft Visio. Palyginimo rezultatai pateikti sekančioje lentelėje.

2.4 lentelė. MagicDraw UML ir Microsoft Visio programinio praplėtimo galimybių analizė

	MagicDraw UML	**Microsoft Visio**
DB schemos generavimo funkcija	Realizuota	Realizuota
Generavimo metu palaikomos DBVS	ANSI-SQL, Oracle, Microsoft SQL, Microsoft Access, MySQL, PostgreSQL, DB2 ir kt.	OLE DB, ODBC, Microsoft SQL, Microsoft Access, Oracle, Informix, DB2, Sybase
DB reinžinerijos funkcija	Realizuota	Realizuota
Sistemos plėtimo galimybė	Yra [11]	Yra [10]
Palaikomos programavimo kalbos	JAVA	C++, C#, Visual Basic, Visual Basic .NET
Reikalaujama vykdymo aplinka	JAVA Runtime, JDBC valdikliai prisijungimui prie MS SQL	Microsoft Office

Palyginus šiuos du įrankius, esminių funkcionalumo skirtumų nepastebėta. Nežymus MagicDraw trūkumas yra tai, jog kai kuriuos prisijungimo komponentus reikia parsisiųsti iš Windows Update tarnybos ir įdiegti, prieš pradedant naudotis, pvz., Microsoft SQL DB reinžinerijos priemonėmis. Todėl galutinį pasirinkimą nulėmė daugiau asmeninė patirtis naudojant vieno iš gamintojų produktus.

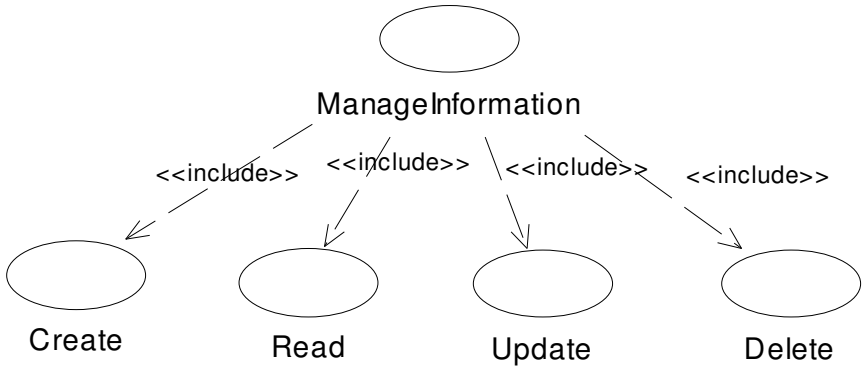
Realizavimo priemonės:

- UML įrankis – Microsoft Office Visio 2003;
- UML įrankio API – Microsoft Office Visio 2003 SDK;
- RDBVS – Microsoft SQL Server.

2.6.2. Testinio pavyzdžio parinkimas

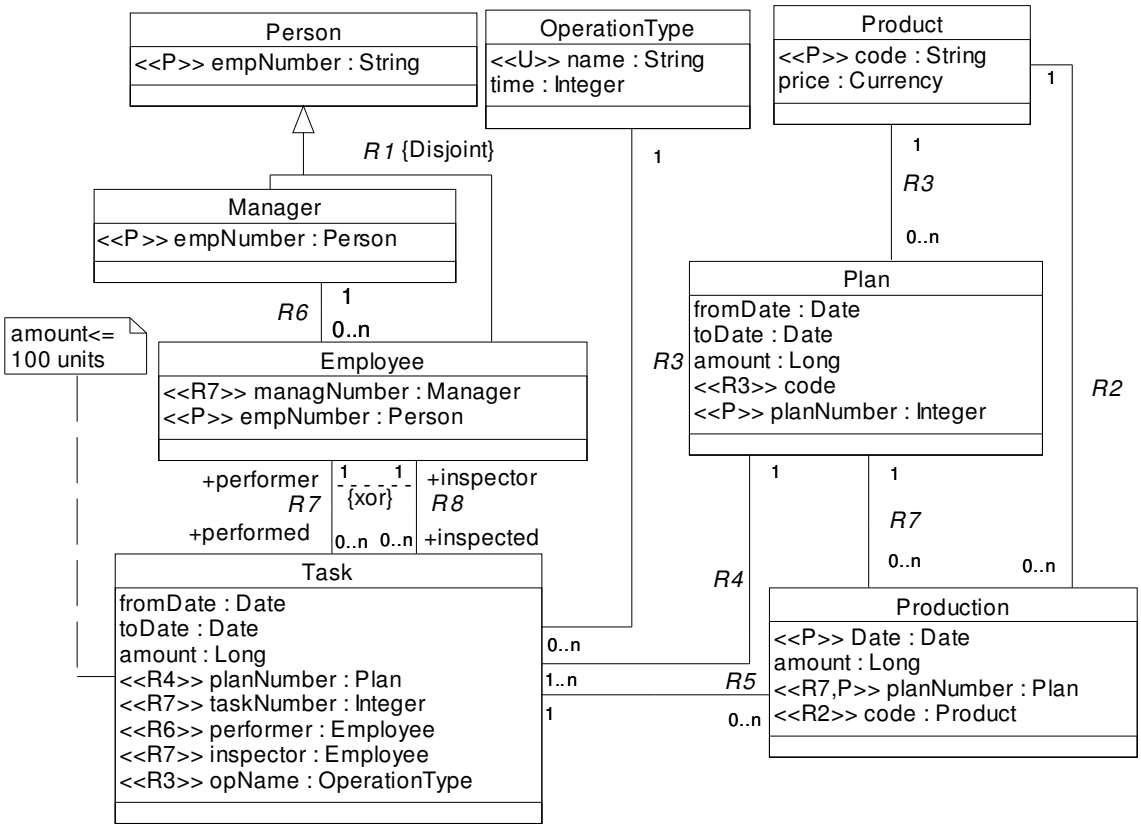
Kadangi duomenų bazės schema testuojama nepriklausomai nuo taikomųjų uždavinių, duomenų bazės panaudojimo atvejus galima apibendrinti panaudojimo atveju, kuris turės

užtikrinti CRUD (angl. *Create*, *Read*, *Update*, *Delete*) funkcijas, dar vadinamas DML (angl. *Data Manipulation Language*). *Read* operacijos nebus tikrinamos (2.10 pav.).



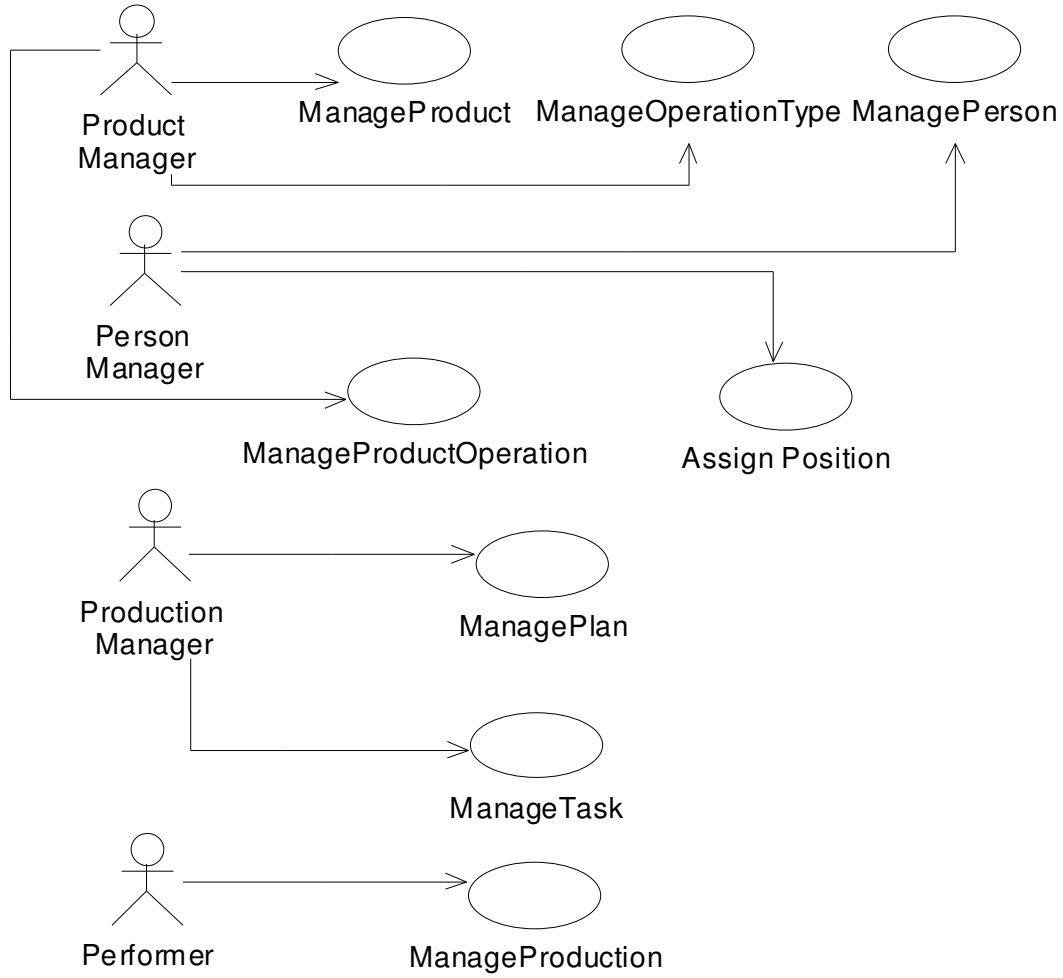
2.10 pav. DML operacijos, nagrinėjamos testiniame pavyzdyje

Pasirinkta duomenų bazės schema pavaizduota paveikslėlyje žemiau. Buvo siekiama parinkti tokį testavimo pavyzdį, kuris apimtų pakankamai daug testavimo atvejų, bet sistema būtų neperpildyta. Testiniame pavyzdyje, kaip matyti iš ryšių tarp lentelių, daug išorinio rakto apribojimų, taip pat – CHECK suvaržymų (TASK: amount <= 100; TASK: fromDate <= toDate; PLAN: fromDate <= toDate).



2.11 pav. Testinė duomenų bazė, kuriai bus bandoma sukurti praktinę metodo realizaciją

Nagrinėjamas schemos kūrimo ir testavimo metodas remiasi laipsniška analize. Panaudojimo atvejai išdėstomi jų rezultate gaunamos informacijos gyvavimo trukmės mažėjimo kryptimi, t.y., einama nuo pastovesnės, ilgiau gyvuojančios informacijos apdorojimo panaudojimo atvejų link trumpiau gyvuojančios ir dažniau apdorojamos informacijos. Kontroliniame pavyzdyje realizuojamos DML operacijos pavaizduotos 2.12 paveiksle.



2.12 pav. Panaudojimo atvejai, realizuotini testuojamoje sistemoje

Schemos testavimo metu nuosekliai nagrinėjami panaudojimo atvejai einant nuo aukštesnio lygmens prie žemesnio. Vieno lygmens panaudojimo atvejus galima nagrinėti lygiagrečiai, kadangi jie vienas nuo kito nepriklauso. Esybių klasės sukuriamos tokiu būdu: nagrinėjami esybių tipai, kurių reikia naujai esybei sukurti. Sistemos projektavimo etapo metu algoritmas, pagal kurį bus laipsniškai rūšiuojamos ir vykdomos DML operacijos, bus sukonkretintas, ir testinis pavyzdys bus paminėtas dar kartą, kartu su algoritmo iliustracija.

2.7. Analizės išvados:

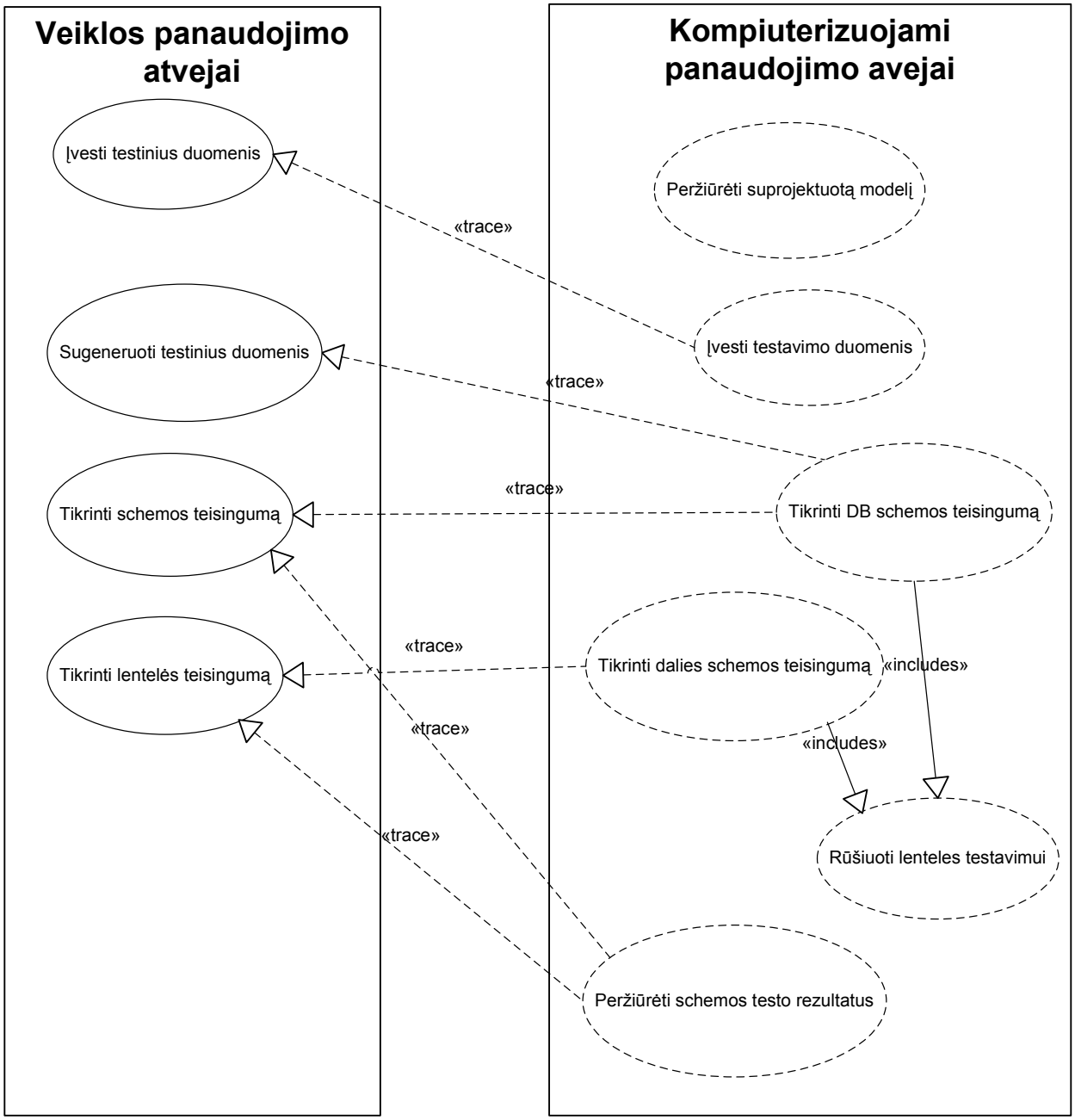
- Duomenų schemos tikrinimo metodikos analizės ir projektavimo etapams atlikti pasirinkti modernūs ir produktyvūs metodai ir priemonės: UML modeliavimo kalba bei *Extreme Programming* projekto valdymo idėjos;
- Darbo sėkmė priklausys nuo sėkmingo metodikos algoritmų aprašymo, dokumentacijos paruošimo, ir testinio pavyzdžio realizavimo. Sekančiuose projekto etapuose šiems veiksniams turi būti skiriamas ypatingas dėmesys;
- Atlikta mokslinės literatūros analizė leido sukaupti naudingos informacijos: testavimo metodikos perteikimo standartus, objektiškai orientuotą testavimo modelį. Ši informacija bus naudojama sistemos projektavimo ir realizacijos etapuose;
- Iš CASE paketų įvairovės išsirinkti konkretūs produktai, kuriems bus realizuojamas duomenų modeliavimo ir schemos tikrinimo metodikos praktinis pavyzdys: Microsoft Office Visio CASE įrankis, Microsoft SQL Server RDBVS.

3. Duomenų bazių testavimo įrankio modelis ir projektas

3.1. Reikalavimų modelis

3.1.1. Vartotojo panaudojimo atvejų diagramos

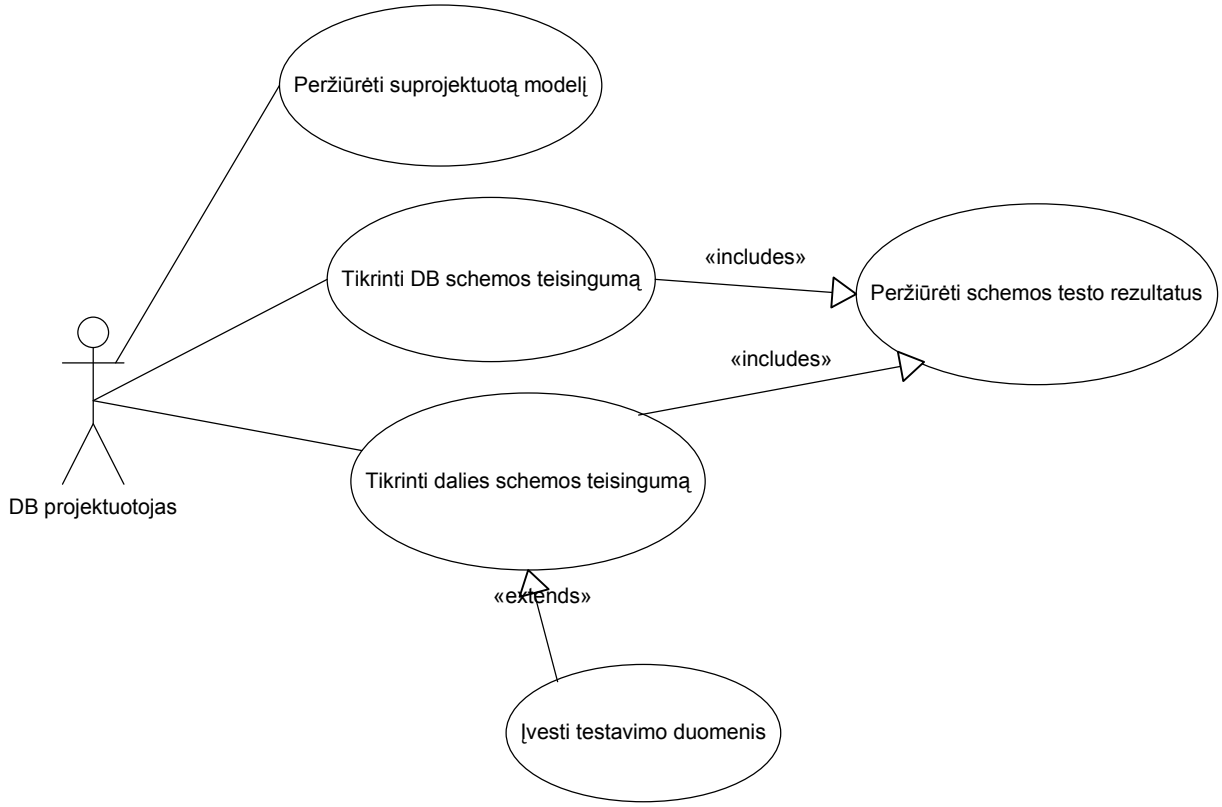
Vartotojo panaudojimo atvejų modelis gaunamas iš analizės dalyje sukauptos informacijos apie dalykinę sritį. Kompiuterizuojami panaudojimo atvejai pavaizduoti 3.1 pav.



3.1 pav. Kompiuterizuojami panaudojimo atvejai

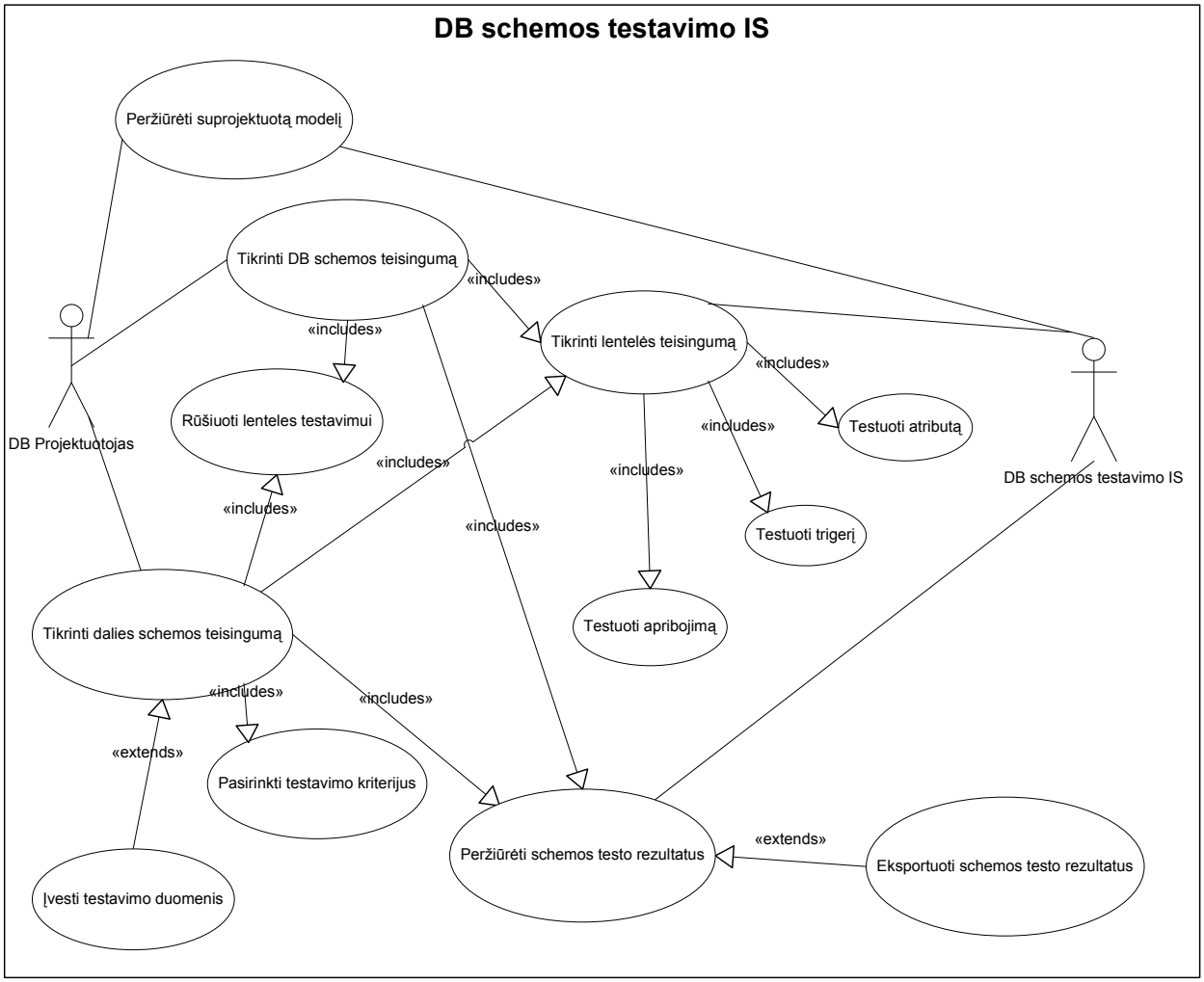
Kadangi vartotojo panaudojimo atvejų diagramoje norėta atvaizduoti tik kompiuterizuojamas funkcijas, iš analizės modelio neperkeltos funkcijos, jau realizuotos

Microsoft Visio 2003 CASE įrankyje, t.y. schemos SQL skripto generavimas ir fizinės DB diegimas. Kompiuterizuojamų panaudojimo atvejų sąsaja su vartotoju pateikiama 3.2 pav.



3.2 pav. Vartotojo panaudojimo atvejai

3.1.2. Sistemos kontekstinė diagrama



3.3 pav. Sistemos kontekstinė diagrama

Kontekstinėje diagramoje (3.3 pav.) vaizduojamos sistemos ribos, t.y. visos funkcijos, kurias buvo nuspręsta kompiuterizuoti. Šis modelis taip pat nėra labai nutolęs nuo dalykinės srities analogiško modelio, tačiau, kaip ir ankstesniame vartotojo panaudojimo atvejų modelyje, pašalintos funkcijos, kurios jau yra realizuotos CASE įrankyje.

3.1.3. Panaudojimo atvejų specifikacijos

3.1.3.1. Panaudojimo atvejų specifikacijos lentelės

1. Panaudojimo atvejis – **Peržiūrėti suprojektuotą modelį**		
1.1.	Tikslas	Peržiūrėti į sistemą įvestą DB schemos modelį
1.2.	Aktoriai	DB projektuotojas DB schemos testavimo IS
1.3.	Nefunkciniai reikalavimai	Vartotojas turi pateikti, jo nuomone, pilnai sukurtą / suprojektuotą DB schemos modelį.
1.4.	Prieš-sąlygos	-

1.5.	Sužadinimo sąlyga	Vartotojas įveda schemos modelį į sistemą.
1.6.	Po-sąlygos	1. DB schemos modelis peržiūrimas
1.7.	Pagrindinis scenarijus	1. Vartotojas pasileidžia sistemos langą. 2. Įveda savo norimą testuoti DB schemos modelį. 3. Aktyvuojama schemos modelio peržiūra
1.8.	Alternatyvus scenarijus	1. Klaida įvedant DB schemos modelį 2. Vartotojas nutraukia darbą su programa

2. Panaudojimo atvejis – **Tikrinti DB schemos teisingumą**		
2.1.	Tikslas	Patikrinti DB schemos teisingumą
2.2.	Aktoriai	DB projektuotojas
2.3.	Nefunkciniai reikalavimai	DB schema turi būti įvesta be sutrikimų
2.4.	Prieš-sąlygos	Įvesta DB schema
2.5.	Sužadinimo sąlyga	Vartotojas nori ištestuoti DB schemą
2.6.	Po-sąlygos	Pateikiami tikrinimo rezultatai
2.7.	Pagrindinis scenarijus	1. Vartotojas inicijuoja lentelių tikrinimą (panaudojimo atvejis Nr. 7) 2. Nurodo testavimo savybes (panaudojimo atvejai Nr. 8-10)
2.8.	Alternatyvus scenarijus	1. Įvesti nurodymai yra klaidingi, vartotojas grąžinamas į pirmą žingsnį ir gali pakartoti įvedimą 2. Vartotojas nutraukia darbą su programa

3. Panaudojimo atvejis – **Tikrinti dalies schemos teisingumą**		
3.1.	Tikslas	Patikrinti dalies schemos teisingumą
3.2.	Aktoriai	DB projektuotojas
3.3.	Nefunkciniai reikalavimai	Įvedami duomenys turi būti teisingi
3.4.	Prieš-sąlygos	Turi būti jau inicijuotas DB schemos testavimas
3.5.	Sužadinimo sąlyga	Vartotojas nori patikrinti dalies schemos teisingumą
3.6.	Po-sąlygos	Pateikiami tikrinimo rezultatai
3.7.	Pagrindinis scenarijus	1. Vartotojas pasirenka testavimo kriterijus (panaudojimo atvejis Nr. 6) 2. Vartotojas įveda testavimo duomenis (panaudojimo atvejis Nr. 5) 3. Vartotojas inicijuoja lentelių tikrinimą (panaudojimo atvejis Nr. 7) 4. Nurodo testavimo savybes (panaudojimo atvejai Nr. 8-10)
3.8.	Alternatyvus scenarijus	1. Vartotojas įveda neteisingus duomenis, vartotojas turi pakartoti duomenų įvedimą.

		2. Vartotojas nutraukia darbą su programa

4. Panaudojimo atvejis – **Peržiūrėti schemos testo rezultatus**		
4.1.	Tikslas	Sistemos pateikta forma peržiūrėti testavimo rezultatus
4.2.	Aktoriai	DB projektuotojas DB schemos testavimo IS
4.3.	Nefunkciniai reikalavimai	Teikiama informacija turi išsamiai apibūdinti testavimo rezultatus
4.4.	Prieš-sąlygos	Turi būti pateikti visi testavimui reikalingi duomenys
4.5.	Sužadinimo sąlyga	Baigiamas testavimas ir vartotojas nori pamatyt i jo rezultatus
4.6.	Po-sąlygos	Suformuojami testavimo rezultatai
4.7.	Pagrindinis scenarijus	1. Baigiamas testavimas 2. Tvirtinami testavimo rezultatai (testavimas pavyko) 3. Atliekamas testavimo rezultatų suformavimas 4. Testavimo rezultatai pateikiami vartotojui 5. Eksportuoti testavimo rezultatus (panaudojimo atvejis Nr.11)
4.8.	Alternatyvus scenarijus	1. Rezultatų išvedimo procese įsivelia klaida 2. Vartotojas nutraukia darbą su programa

5. Panaudojimo atvejis – **Įvesti testavimo duomenis**		
5.1.	Tikslas	Įvesti duomenis reikalingus testavimui
5.2.	Aktoriai	DB projektuotojas
5.3.	Nefunkciniai reikalavimai	Duomenys turi būti įvesti teisingai
5.4.	Prieš-sąlygos	Turi būti inicijuotas dalies schemos teisingumo tikrinimas (panaudojimo atvejis Nr. 3)
5.5.	Sužadinimo sąlyga	Atsiranda poreikis duomenims siekiant patikrinti schemos teisingumą
5.6.	Po-sąlygos	Tęsiamas schemos testavimas
5.7.	Pagrindinis scenarijus	1. Įvedami reikalaujami duomenys
5.8.	Alternatyvus scenarijus	1. Duomenų įvedimo procese įsivelia klaida 2. Vartotojas nutraukia darbą su programa

6. Panaudojimo atvejis – **Pasirinkti testavimo kriterijus**

6.1.	Tikslas	Pasirinkti kriterijus reikalingus testavimui
6.2.	Aktoriai	DB projektuotojas
6.3.	Nefunkciniai reikalavimai	Pasirinkti teisingi kriterijai
6.4.	Prieš-sąlygos	Turi būti inicijuotas dalies schemos teisingumo tikrinimas (panaudojimo atvejis Nr. 3)
6.5.	Sužadinimo sąlyga	Vykdomas dalies schemos teisingumo tikrinimas (panaudojimo atvejis Nr. 3)
6.6.	Po-sąlygos	Tęsimas schemos testavimas
6.7.	Pagrindinis scenarijus	1. Pasirenkami kriterijai
6.8.	Alternatyvus scenarijus	1. Kriterijų pasirinkimo procese įsivelia klaida 2. Vartotojas nutraukia darbą su programa

7. Panaudojimo atvejis – **Tikrinti lentelės teisingumą**

7.1.	Tikslas	Patikrinti DB lentelės teisingumą
7.2.	Aktoriai	DB schemos testavimo IS
7.3.	Nefunkciniai reikalavimai	-
7.4.	Prieš-sąlygos	Inicijuotas tikrinimo procesas (panaudojimo atvejai Nr. 2 ar Nr. 3)
7.5.	Sužadinimo sąlyga	Vykdomas testavimas
7.6.	Po-sąlygos	Suformuojami testavimo rezultatai
7.7.	Pagrindinis scenarijus	1. Inicijuojamas tikrinimas 2. Testuojami apribojimai (panaudojimo atvejis Nr. 8) 3. Testuojami trigeriai (panaudojimo atvejis Nr. 9) 4. Testuojami atributai (panaudojimo atvejis Nr. 10)
7.8.	Alternatyvus scenarijus	1. Testavimo procese įsivelia klaida 2. Vartotojas nutraukia darbą su programa

8. Panaudojimo atvejis – **Testuoti apribojimą**

8.1.	Tikslas	Ištestuoti apribojimus
8.2.	Aktoriai	DB schemos testavimo IS
8.3.	Nefunkciniai reikalavimai	-
8.4.	Prieš-sąlygos	Inicijuotas lentelės tikrinimo procesas (panaudojimo atvejis Nr. 7)
8.5.	Sužadinimo sąlyga	Baigiamas testavimas ir vartotojas nori pamatyt i jo rezultatus
8.6.	Po-sąlygos	Suformuojami testavimo rezultatai

8.7.	Pagrindinis scenarijus	1. Testuojami apribojimai
8.8.	Alternatyvus scenarijus	1. Testavimo procese įsivelia klaida 2. Vartotojas nutraukia darbą su programa

9. Panaudojimo atvejis – **Testuoti trigerį**		
9.1.	Tikslas	Ištestuoti trigerius
9.2.	Aktoriai	DB schemos testavimo IS
9.3.	Nefunkciniai reikalavimai	-
9.4.	Prieš-sąlygos	Inicijuotas lentelės tikrinimo procesas (panaudojimo atvejis Nr. 7)
9.5.	Sužadinimo sąlyga	Baigiamas testavimas ir vartotojas nori pamatyt i jo rezultatus
9.6.	Po-sąlygos	Suformuojami testavimo rezultatai
9.7.	Pagrindinis scenarijus	1. Testuojami trigeriai
9.8.	Alternatyvus scenarijus	1. Testavimo procese įsivelia klaida 2. Vartotojas nutraukia darbą su programa

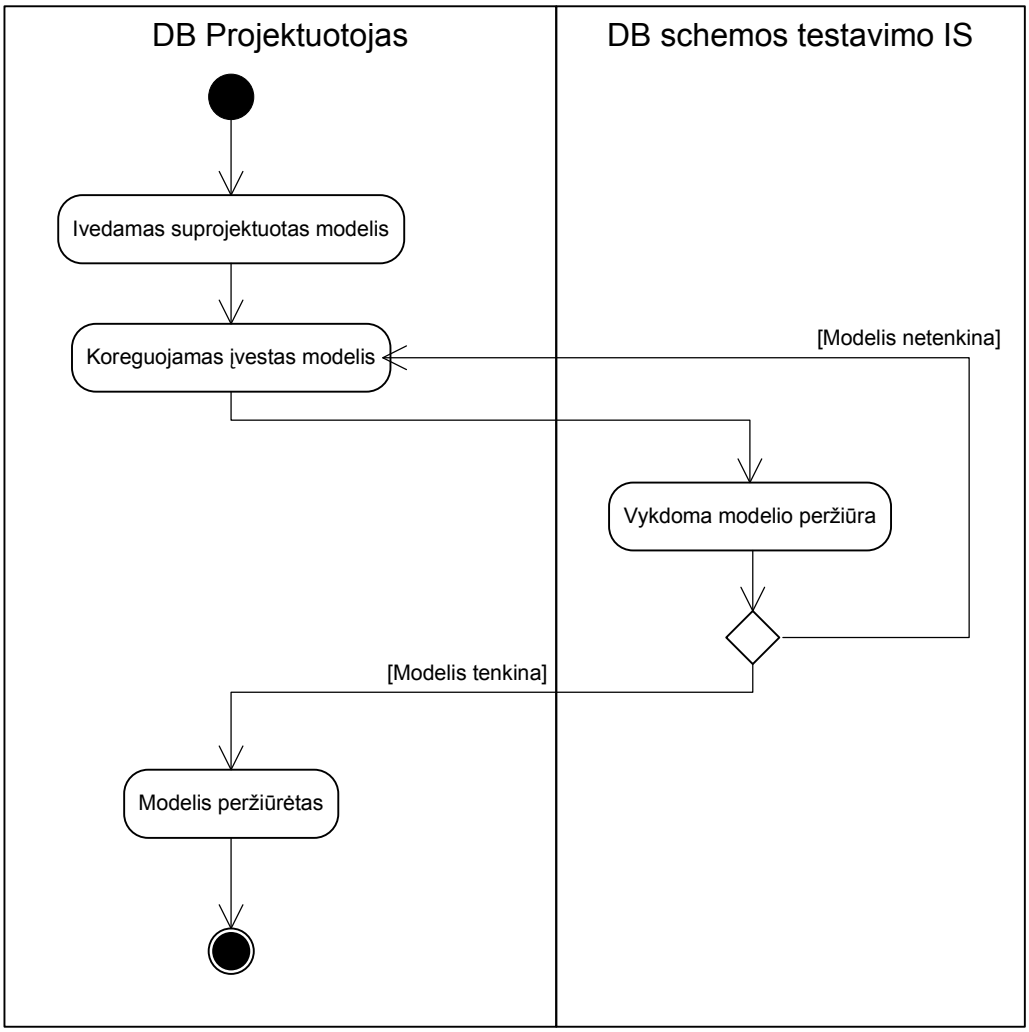
10. Panaudojimo atvejis – **Testuoti atributą**		
10.1.	Tikslas	Ištestuoti atributus
10.2.	Aktoriai	DB schemos testavimo IS
10.3.	Nefunkciniai reikalavimai	-
10.4.	Prieš-sąlygos	Inicijuotas lentelės tikrinimo procesas (panaudojimo atvejis Nr. 7)
10.5.	Sužadinimo sąlyga	Baigiamas testavimas ir vartotojas nori pamatyt i jo rezultatus
10.6.	Po-sąlygos	Suformuojami testavimo rezultatai
10.7.	Pagrindinis scenarijus	1. Testuojami atributai
10.8.	Alternatyvus scenarijus	1. Testavimo procese įsivelia klaida 2. Vartotojas nutraukia darbą su programa

11. Panaudojimo atvejis – **Eksportuoti schemos testo rezultatus**		
11.1.	Tikslas	Ekportuoti testavimo rezultatus
11.2.	Aktoriai	DB projektuotojas

		DB schemos testavimo IS
11.3.	Nefunkciniai reikalavimai	Rezultatai turi būti pateikti aiškiai ir suprantamai
11.4.	Prieš-sąlygos	Baigtas testavimas
11.5.	Sužadinimo sąlyga	Vartotojas nori eksportuoti rezultatus
11.6.	Po-sąlygos	-
11.7.	Pagrindinis scenarijus	1. Rezultatai ekportuojami
11.8.	Alternatyvus scenarijus	1. Eksportavimo procese įsivelia klaida 2. Vartotojas nutraukia darbą su programa

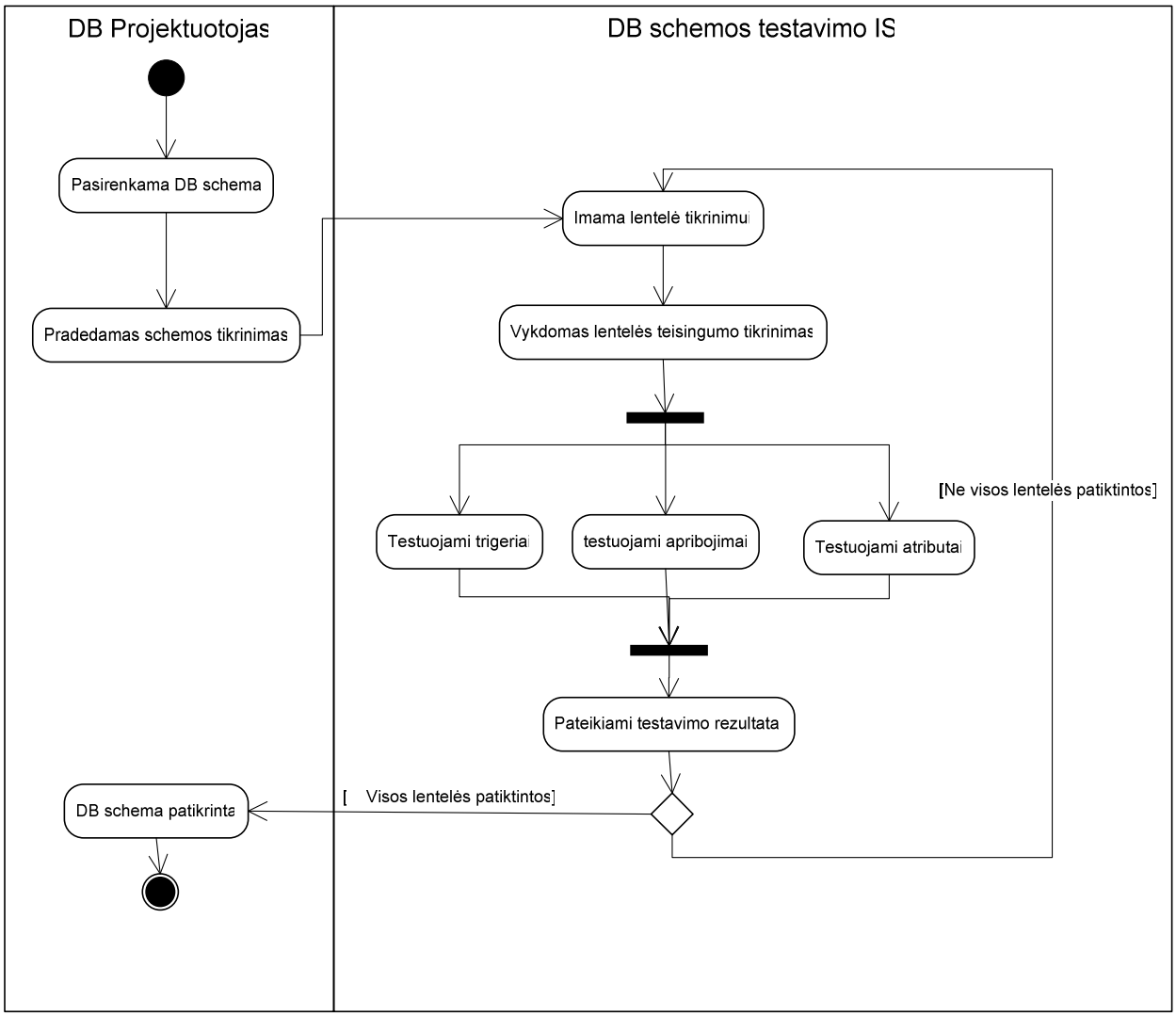
3.1.3.2. Panaudojimo atvejų veiklos diagramos

Bendra veiklos schema, kuria remiantis dirbs DB projektuotojas, yra identiška analizės etape gautai procesų modelio diagramai. Šiame poskyryje bus išsamiau išnagrinėtos stambiausių panaudojimo atvejų veiksmų sekos. Šioms sekoms vizualizuoti taip pat naudojamos veiklos diagramos.

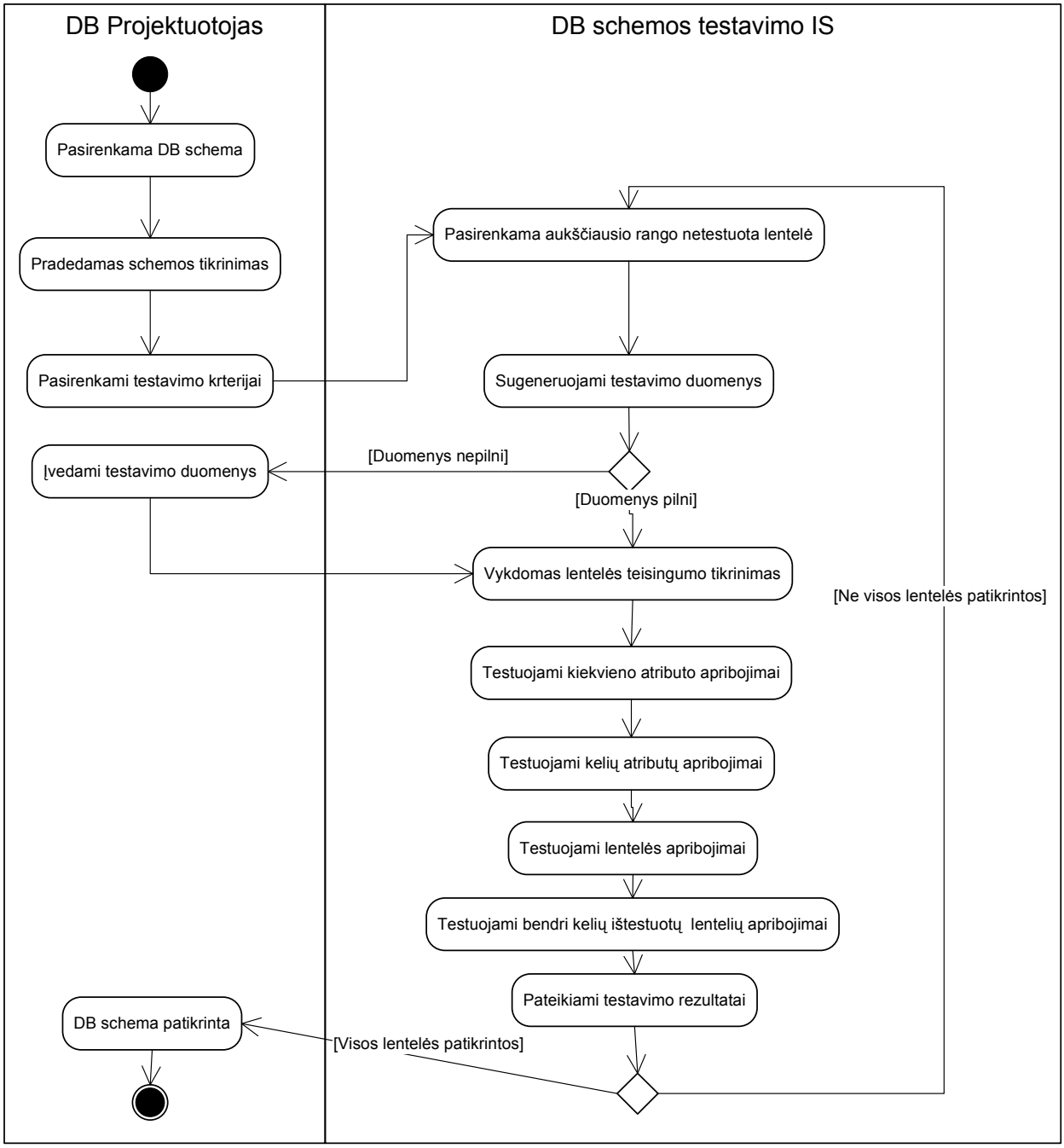


3.4 pav. Panaudojimo atvejo „Peržiūrėti suprojektuotą modelį“ veiklos diagrama

Paveiksluose 3.4 ir 3.5 atitinkamai pavaizduotos panaudojimo atvejų „Peržiūrėti suprojektuotą modelį“ bei „Tikrinti DB schemos teisingumą“ veiklos diagramos.

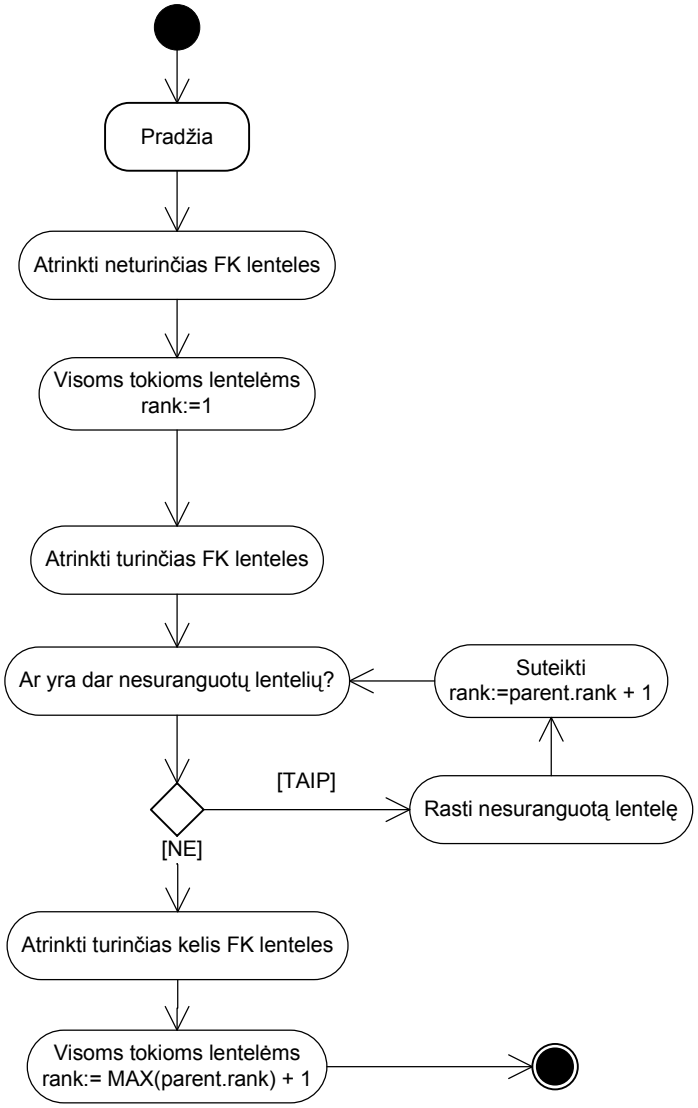


3.5 pav. Panaudojimo atvejo „Tikrinti DB schemos teisingumą“ veiklos diagrama



3.6 pav. Panaudojimo atvejo „Tikrinti dalies schemos teisingumą“ veiklos diagrama

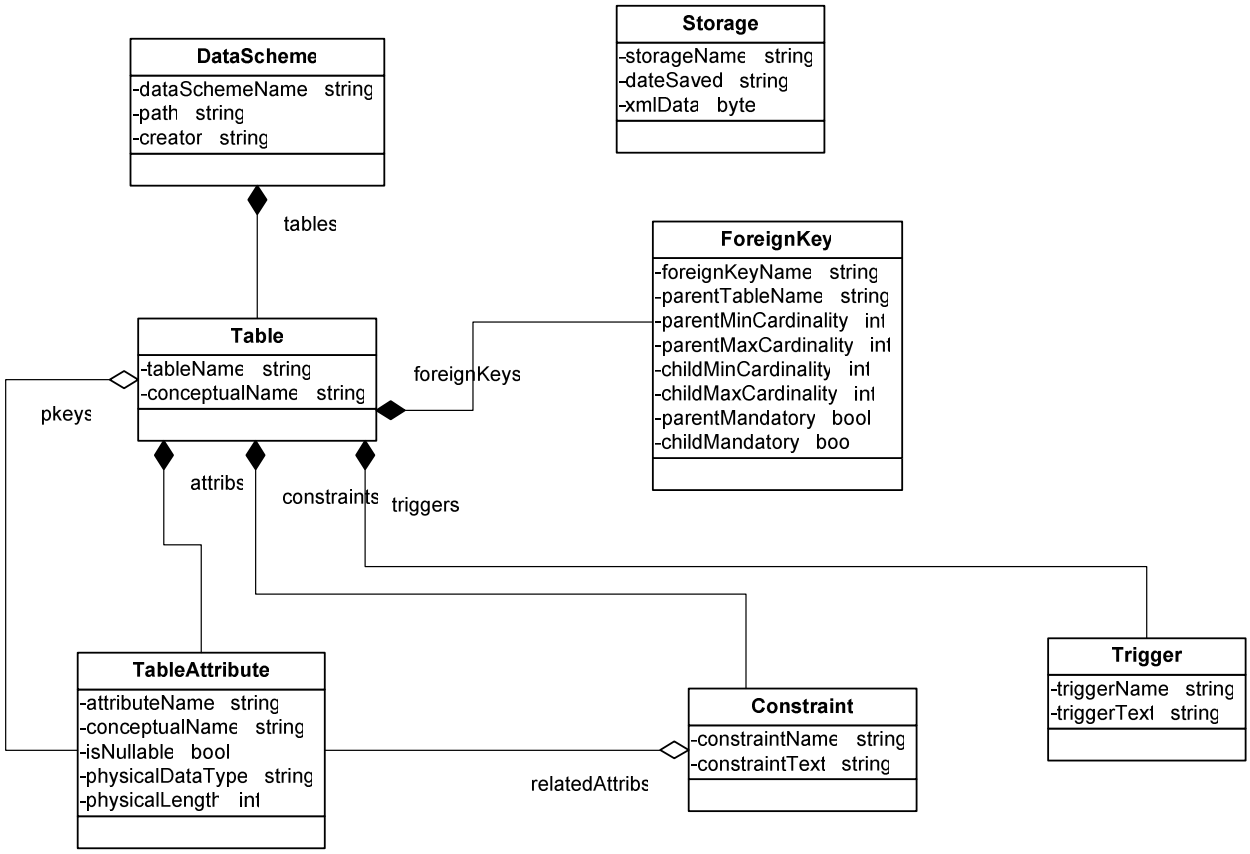
Kaip matoma paveiksle aukščiau, vaizduojančiame panaudojimo atvejį „Tikrinti dalies schemos teisingumą“, DB projektuotojas galės testuoti ne visą duomenų bazės schemą, o tik dalį (vieną ar daugiau) jo pasirinktų lentelių. Ir testuojant lenteles pasirinktinai, ir pasirinkus pilnos schemos testavimą, naudotojui suteikiama galimybė įvesti savo norimus duomenis. Testo rezultatai pateikiami tokiu pat būdu, kaip ir po pilno schemos tikrinimo. Plačiau apie vartotojo sąsajos organizavimą – 3.1.5 poskyryje, „Vartotojo sąsajos modelis“.



3.7 pav. Panaudojimo atvejo „Rūšiuoti lenteles testavimui“ veiklos diagrama

Panaudojimo atvejis „Rūšiuoti lenteles testavimui“ (3.7 pav.) skirtas tam, kad sistema lenteles išrikiuotų ta tvarka, kuria testo metu bus generuojami testiniai duomenys. T.y. pirmiausia turi būti užpildomos lentelės, neturinčios priklausomybės nuo tėvinių lentelių, vėliau – priklausomos nuo jų lentelės ir t.t.

3.1.4. Dalykinės srities klasių diagrama



3.8 pav. Dalykinės srities klasių modelis

Dalykinės srities klasių diagrama, pavaizduota 3.8 paveiksle, sudaryta remiantis veiklos objektų modeliu. Šį modelį sudaro tokie klasių tipai:

- DataScheme: duomenų schemos aprašas, bus kuriama viena tokia klasė vienam Visio dokumentui;
- Storage: duomenų struktūra, skirta schemoje sugeneruotų testinių duomenų išsaugojimui duomenų bazėje arba XML dokumente;
- Table: klasė, nešanti savyje informaciją apie vieną Visio duomenų lentelę. Klasėje „DataScheme“ visos priklausančios schemai lentelės kaupiamos kompozicijos ryšiu per atributą „tables“;
- TableAttribute: klasė, apibrėžianti vieną duomenų bazės atributą, t.y. stulpelį. Su lentele „Table“ siejasi dvi atributų kolekcijos: „attribs“ nurodo visus tos klasės atributus, o „pkeys“ pakartotinai nurodo atributus, kurie Visio schemoje pažymėti kaip pirminiai raktai;
- ForeignKey: klasė, aprašanti vieną išorinį raktą. Išorinių raktų priklausomybė atitinkamai lentelei apibrėžiami per kolekciją „foreignKeys“, esančią lentelėje „Table“;

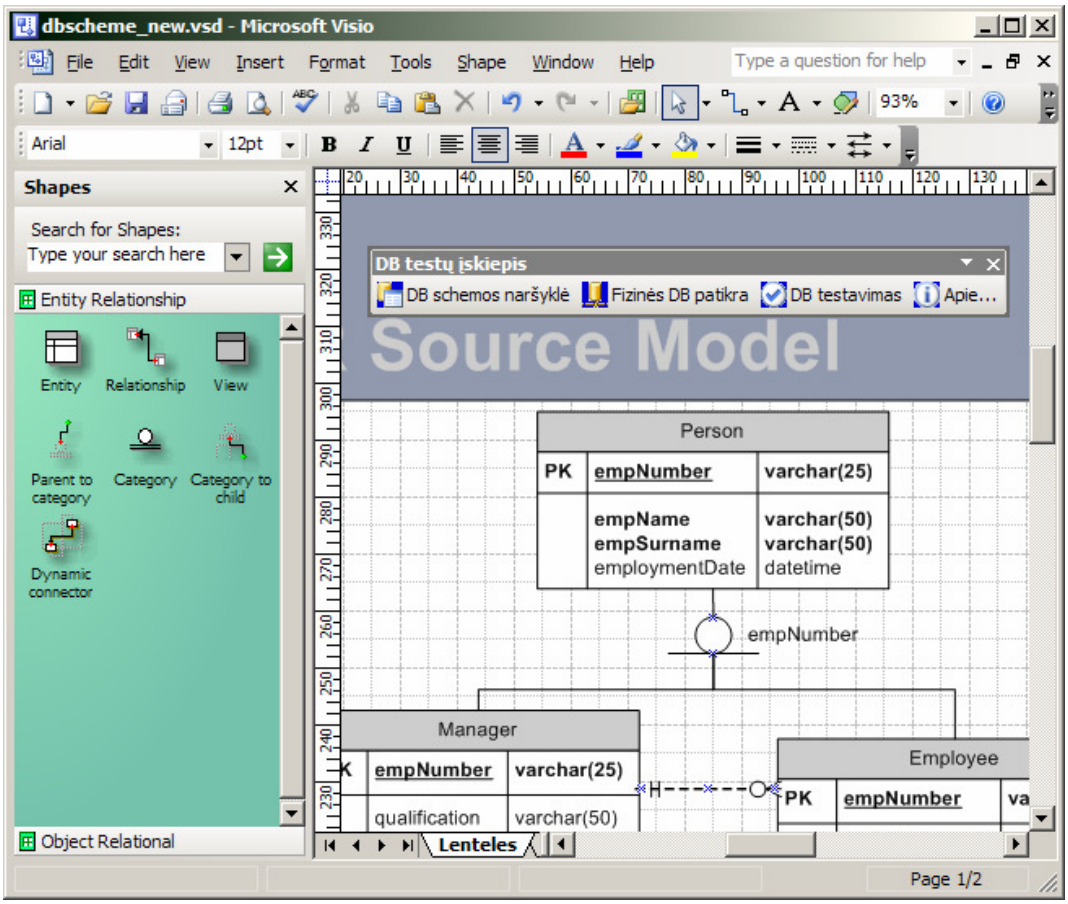
- Constraint: klasė, aprašanti vieną CHECK apribojimą. Apribojimo išgavimo metu bandoma nustatyti, su kuriuo lentelės atributu (kuriais atributais) susijęs tas apribojimas, ir ši informacija išsaugoma kolekcijoje „relatedAttribs“;
- Trigger: klasė, aprašanti vieną trigerį. Su „Table“ klase taip pat susieta per kompozicijos ryšį.

3.1.5. Vartotojo sąsajos modelis

3.1.5.1. Bendri vartotojo sąsajos principai

Kuriant sistemos prototipą, buvo bandoma remtis panašių projektų, t.y. Microsoft Office produktų įskiepių, kūrimo patirtimi, naudotasi įvairiais forumais, programavimo grupių, oficialios MSDN dokumentacijos pagalba. Buvo pastebėta, kad dauguma įskiepių tam tikro Microsoft Office produkto funkcionalumą išplečia sukurdami naują įrankių juostą (angl. *Toolbar*), šioje įrankių juostoje pateikdami mygtukus, skirtus atlikti norimas funkcijas.

Šios sistemos kūrimui taip pat buvo nuspręsta laikytis tokios pat vartotojo sąsajos idėjos. Visio 2003 naudotojui pasirinkus naują arba atsidarius esamą „Database Model Diagram“ tipo schemą, bus sukuriama ir parodoma įrankių juosta „DB testų įskiepis“, kuri leis iškviesti norimas funkcijas (3.9 pav.).

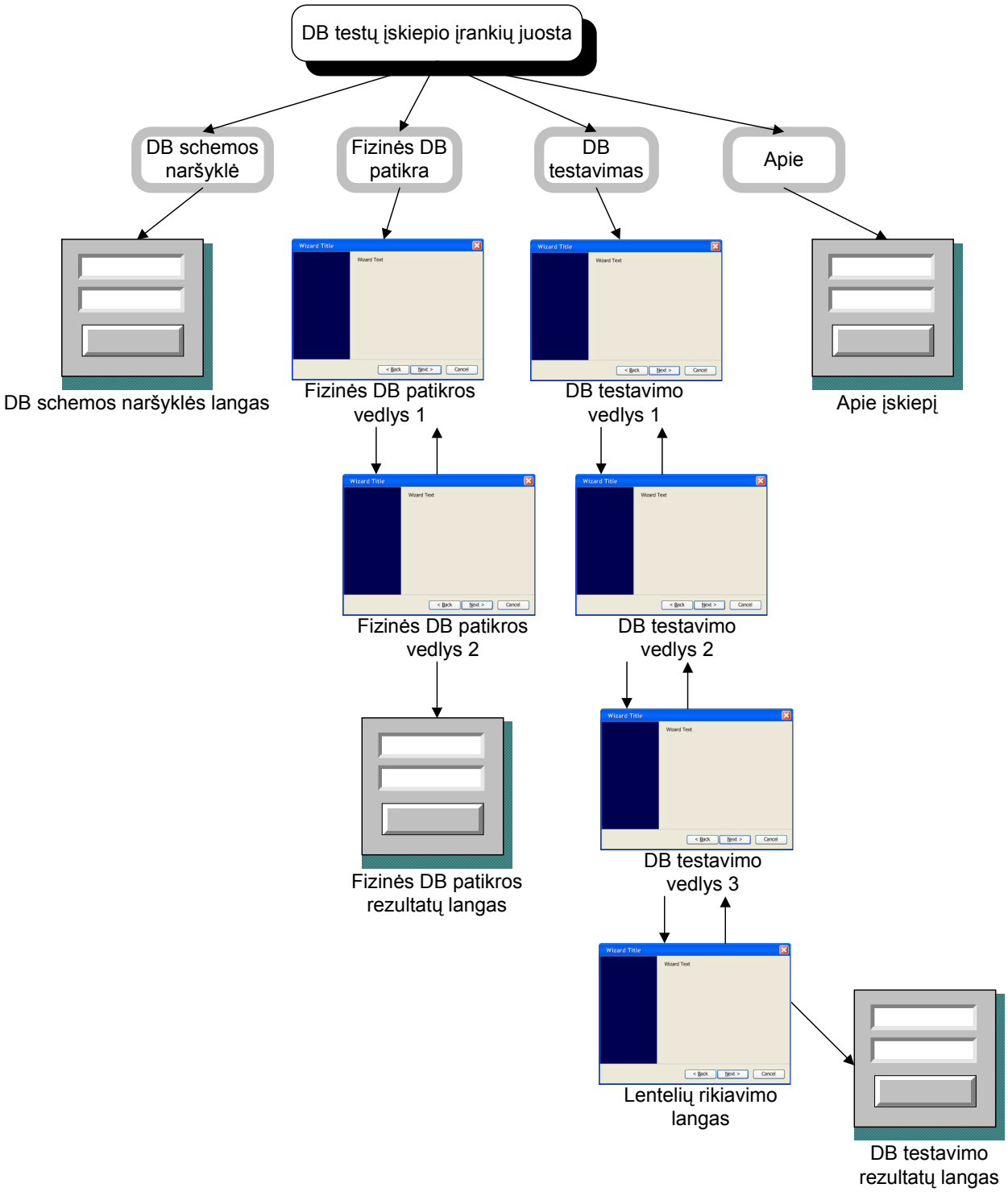


3.9 pav. Įskiepio vartotojo sąsaja – papildoma įrankių juosta Microsoft Visio 2003 programoje

Įskiepio sukurta įrankių juosta turi 4 mygtukus, kurių atliekamos funkcijos bus tokios:

- DB schemos naršyklė: paspaudus šį mygtuką, bus parodoma ekraninė forma, bandanti vizualizuoti schemos elementus ir juos pateikti vartotojui;
- Fizinės DB patikra: šis mygtukas iškvies konfigūracijos vedlį (angl. *Wizard*), kuris padės patogiai vartotojui surinkti informaciją apie tai, prie kokios fizinės DB bus vykdomas prisijungimas, ir kokių DB objektų tipų atitikimo bus ieškoma tarp Visio schemos bei realios duomenų bazės;
- DB testavimas: šio mygtuko paspaudimas iškvies beveik analogišką konfigūracijos vedlį informacijos, skirtos prisijugimui prie DB, testuojamų lentelių, testavimo duomenų įvedimui. Iš vedlio surinkta informacija bus naudojama pagrindinei įskiepio funkcijai – DB testavimui – atlikti. Pabaigus testavimą, vartotojui bus parodomas rezultatų langas;
- Apie: mygtukas iškviečia informacinį langą, kuriame pristatoma sistemos paskirtis ir informacija apie jos kūrėjus.

Norint tinkamai suvokti sistemos teikiamas galimybes ir vartotojo sąsajos projektą, pravartu sudaryti sistemos navigacijos planą. Šis modelis pateiktas 3.10 paveiksle.

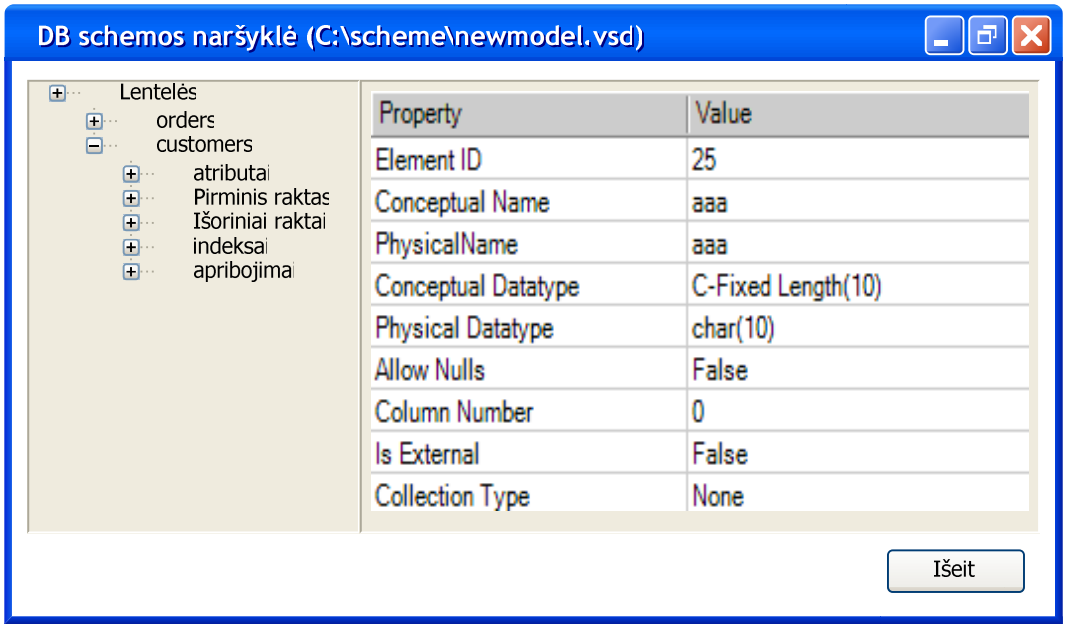


3.10 pav. Sistemos navigacijos planas

3.1.5.2. Ekraninių formų specifikacija

DB schemos naršyklė

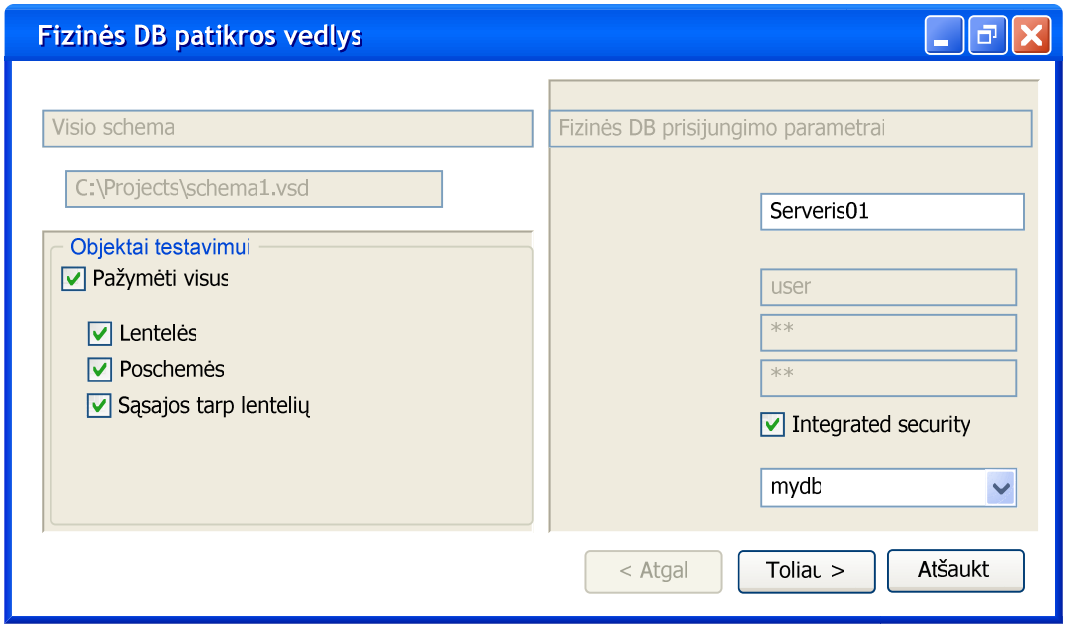
DB schemos naršyklė skirta suprojektuotos schemos peržiūrai. Pagrindinė šios naršyklės nauda būtų informacijos, kurią vartotojas įvedė į VSD modelį, susisteminimas. DB schemos naršyklės lango protipas pateiktas paveiksle žemiau.



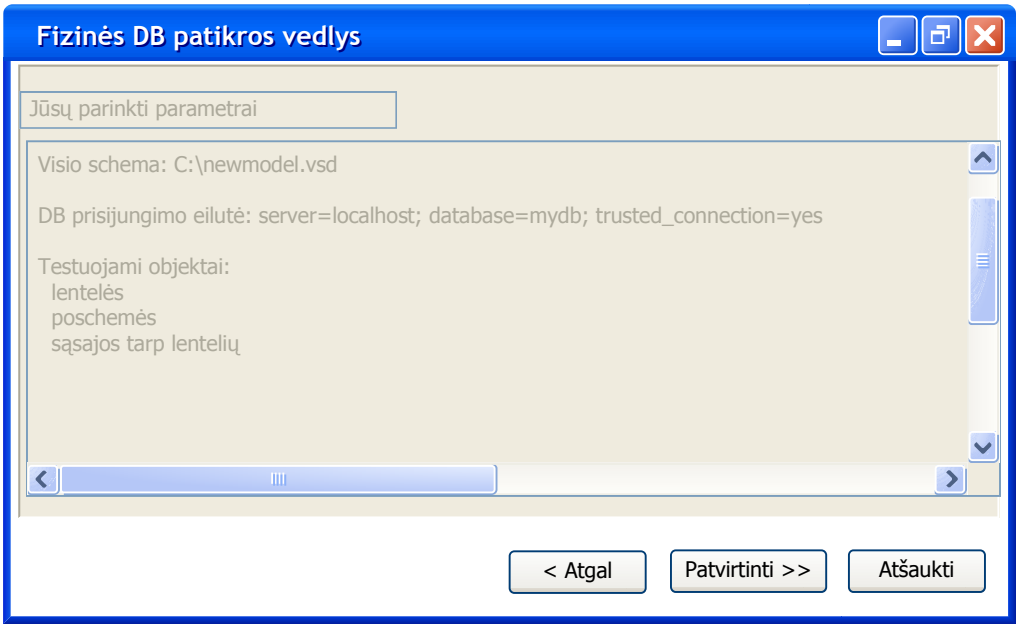
3.11 pav. DB schemos naršyklės prototipas

Fizinės DB patikra

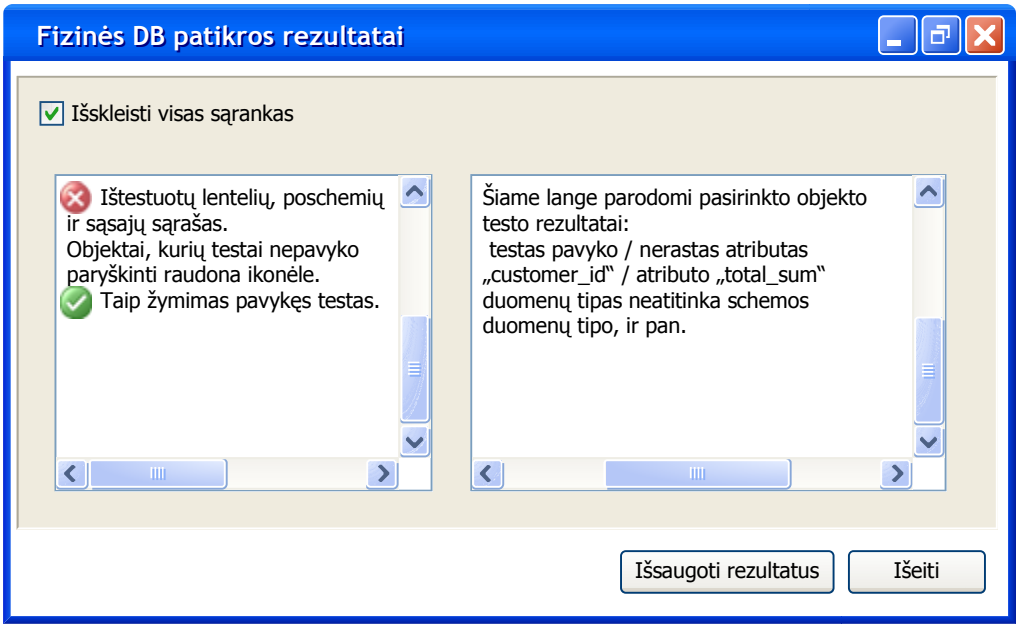
Fizinės DB patikros vedliai, kaip ir pavaizduota sistemos navigacijos plane, yra 2. Vienas iš jų skirtas informacijos apie prisijungimą prie DB ir tikrinamų schemos objektų pasirinkimui, kitas – šios informacijos galutiniam patikrinimui. Abiejų šių ekraninių formų prototipai pateikti atitinkamai 3.12 ir 3.13 paveiksluose.



3.12 pav. Fizinės DB patikros vedlys 1 – prisijungimo parametrai ir tikrinamų objektų tipai



3.13 pav. Fizinės DB patikros vedlys 2 – veiksmo patvirtinimas

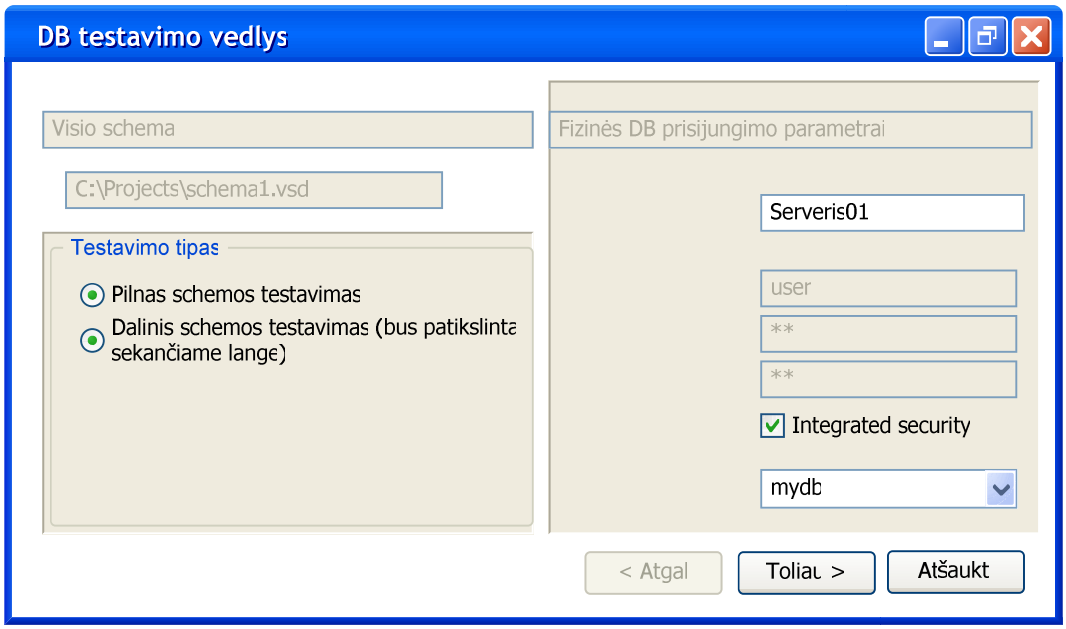


3.14 pav. Fizinės DB patikros rezultatų lango prototipas

Fizinės DB patikros rezultatų langas bus organizuotas kaip „tėvo“ – „vaiko“ forma – kairiojoje ekrano pusėje bus rodomi DB objektai, kurie buvo ištestuoti, šalia jų grafine forma (žalia / raudona ikonėle) bus pažymėta, ar testas pavyko. Pele aktyvavus kažkurį objektą iš kairiojo sąrašo, duomenimis apie šį objektą bus užpildomas sąrašas dešiniojoje ekrano dalyje (3.14 paveikslas).

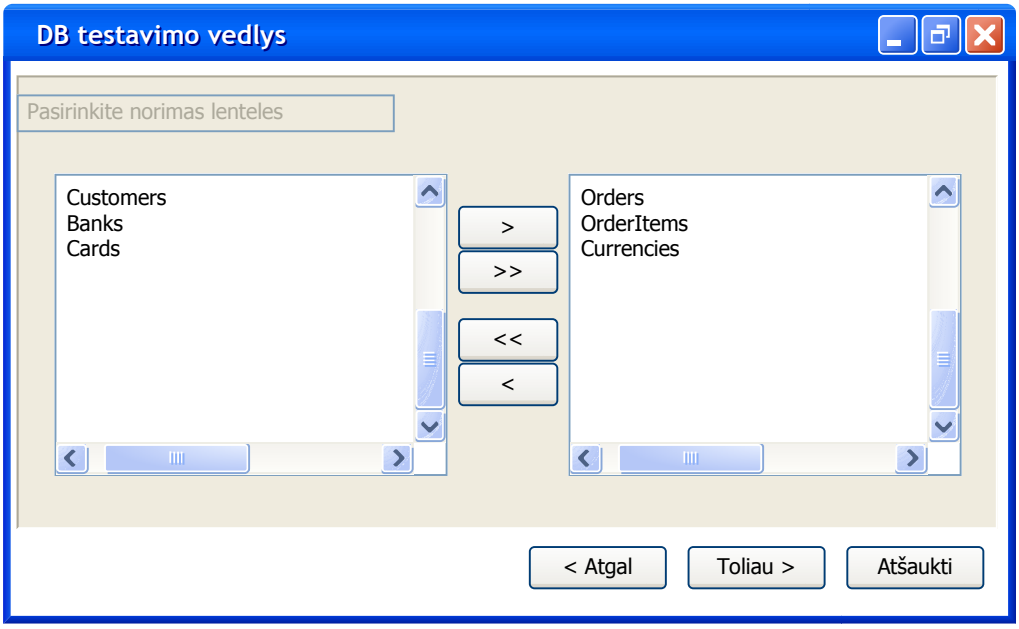
DB testavimas

DB testavimo vedliai, priklausomai nuo pasirinkto testavimo tipo, gali būti 3 arba 4. Pirmasis vedlio langas vartotojo paklausia prisijungimo informacijos bei testavimo tipo – pilnas DB objektų testavimas ar dalies schemos testavimas. Šio lango prototipas pavaizduotas 3.15 pav.

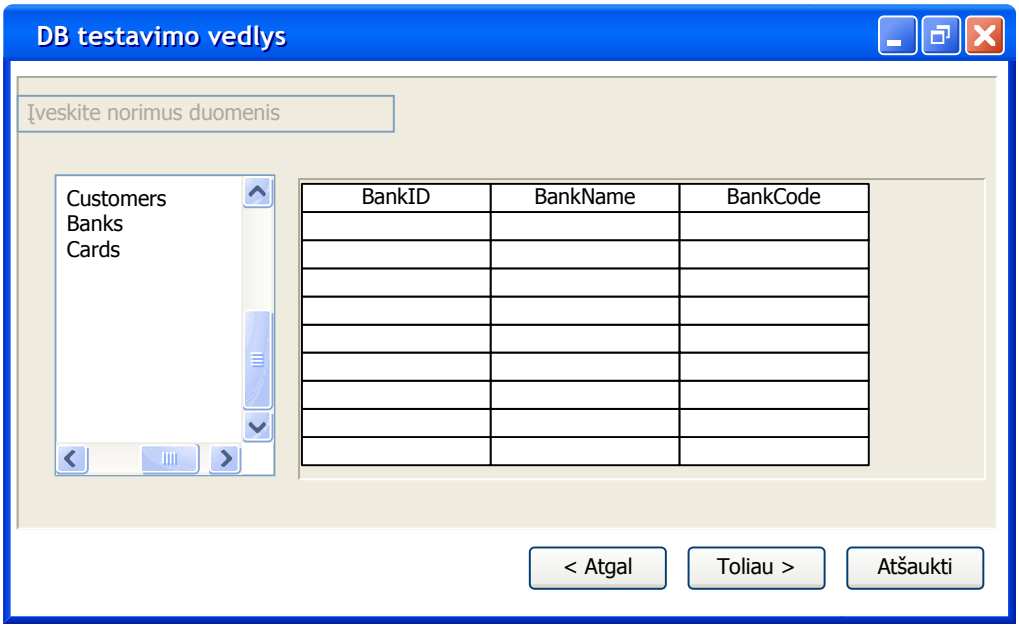


3.15 pav. DB testavimo vedlys 1 – prisijungimo parametrai bei testavimo tipas

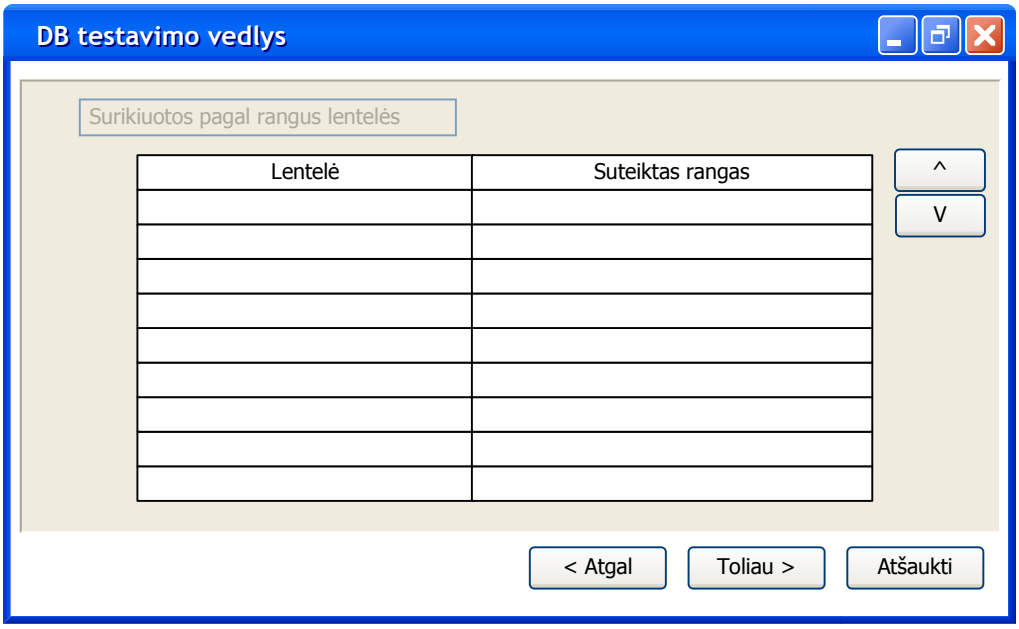
Kitame žingsnyje vartotojui parodomi DB testavimo vedlio langai 2 ir 3 – norimų testavimui lentelių pasirinkimui bei norimų testo duomenų įvedimui. Šių langų prototipai pateikti atitinkamai 3.16 ir 3.17 paveiksluose. Jei buvo pasirinktas pilnas schemos testavimas, norimų testavimui lentelių lange parodoma, kad bus testuojamos visos lentelės, ir vartotojui lentelių iš pasirinktų sąrašo šalinti negalima.



3.16 pav. DB testavimo vedlys 2 – lentelių testavimui pasirinkimas



3.17 pav. DB testavimo vedlys 3 – trūkstamų duomenų įvedimas



3.18 pav. :Lentelių rūšiavimo rikiavimui langas

Lentelių rūšiavimo langas bus parodomas vartotojui kaip priešpaskutinis langas testuojant pilną DB modelį ar dalį modelio. Šiame lange bus pateiktos DB lentelės, kurias DB schemos įskiepis surikiavo automatiškai. Pamatęs, jo nuomone, neteisingai surikiuotą lentelę, vartotojas galės keisti jos rangą mygtukais „^“ (aukštyn) ir „v“ (žemyn).

DB testavimo patvirtinimo langas ir DB testo rezultatų langas bus beveik identiški atitinkamiems fizinės DB patikros langams (3.13 ir 3.14 paveikslai).

3.1.6. Nefunkciniai reikalavimai

Nefunkcinių reikalavimų šaltinis yra ankstesniame poskyryje pateiktos panaudojimo atvejų specifikacijos, taip pat – bendri principai, kurių turi būti laikomasi kuriant informacinę sistemą. Pagrindiniai DB modeliavimo ir schemos tikrinimo metodikai keliami nefunkciniai reikalavimai pateikti 3.1 lentelėje.

3.1 lentelė. Sistemai keliami nefunkciniai reikalavimai

Kam keliamas reikalavimas	Reikalavimo tipas	Reikalavimo apibrėžimas
Kuriama IS	Patikimumas	Sistema turi veikti patikimai, t.y. jos darbas neturi būti nutraukiamas dėl nenumatytų klaidų. Kilus bet kokiai neapdorotai klaidai, klaida turi būti aptinkama – vartotojui parodomas klaidos pranešimas ir detalus aprašas: kreipinių stekas, programinio kodo eilutė, kurioje įvyko klaida ir t.t. Turi būti sudaryta galimybė šio klaidų peržiūros lango duomenis nusikopijuoti į laikinąjį teksto buferį (angl. *Clipboard*), kad šiuos duomenis vėliau būtų galima panaudoti klaidos identifikavimui ir taisymui.
Vartotojo sąsaja	Lengvas vartotojo sąsajos įsisavinamumas	Sistema privalo būti greitai įsisavinama naujų vartotojų, t.y. vartotojų apmokymas turi trukti kiek įmanoma trumpiau.
Vartotojo sąsaja	Ergonomika	Vienas pagrindinių kuriamos DB testavimo sistemos tikslų – sistemos patogumas bei suprantamumas. Siekiama, kad vartotojas, pirmąkart naudodamasis sistema, nesusidurtų su sunkumais. Informacija sistemoje turi būti pateikta aiškiai suprantama lietuvių kalba. Sistemos vykdymo savybių visumos tikslas – atliekant kuo mažiau veiksmų iš vartotojo pusės gauti kuo efektyvesnį rezultatą.
Techninė realizacija	Sistemos plečiamumas	Naujos sistemos galimybės gali būti nesudėtingai įtraukiamos į sistemą, įskiepio atnaujinimą turi būti galima sukurti naudojant bet kokią iš

		Microsoft .NET platformos palaikomų programavimo kalbų.
Techninė realizacija	Sistemos atsako laikas	Sistema privalo veikti stabiliai, be ilgesnių laiko pauzių. Jei gresia ilgesnė laiko pauzė (testuojama sudėtinga DB schema), sistema privalo įspėti apie tai vartotoją. Ilgai trunkančios operacijos turi būti vykdomos atskiroje gijoje (angl. *Thread*), vartotojo sąsajoje animuojant veiksmo progresą, ir suteikiant galimybę nedelsiant nutraukti per ilgai trunkantį veiksmą.
Projektinė realizacija	DB variklio suderinamumas	Sistema turi veikti su bent viena iš šiuo metu naudojamų Microsoft RDBVS produktų: Microsoft SQL Server 2000 (MSDE 2000) arba Microsoft SQL Server 2005 (Microsoft SQL Server 2005 Express). Klasės, atsakingos už prisijungimą prie DB, turi būti suprojektuotos taip, kad būtų galima apjungti ir kitus reliacinių DB variklius.
Diegimas	Suderinamumas	Sistema privalo būti suderinama su Microsoft Visio 2003 programinių paketu. Nei įskiepio įdiegimo, nei šalinimo metu neturi būti paveiktas kokybiškas Microsoft Visio paketo darbas.
Diegimas	Perkeliamumas	Sistema privalo būti įdiegiama lengvai ir suprantamai net programavimo patirties neturinčiam vartotojui. Kadangi sprendimas realizuojamas Microsoft Visio 20003 aplinkoje, sistema turi pati prisitaikyti prie esamos versijos ir įsidiegti priėjimą prie savęs pagrindiniame MS Visio lange. Lygiai taip pat privalomas sistemos pašalinimo galimybės paprastumas bei aiškumas.
Kuriama IS	Bendri reikalavimai	Pagrindinis reikalavimas, keliamas veikimui, yra sistemos stabilumas, greitaveika bei efektyvus rezultatų pateikimas. Nuo šių veiksnių priklauso

		produkto naudingumas. Jei rezultatai bus neaiškūs arba labai lėtai išgaunami, vartotojai bus nepatenkinti, ir CASE sistemos įskiepiu naudosis nenoriai.

3.2. Sistemos projektas

3.2.1. Projekto tikslas

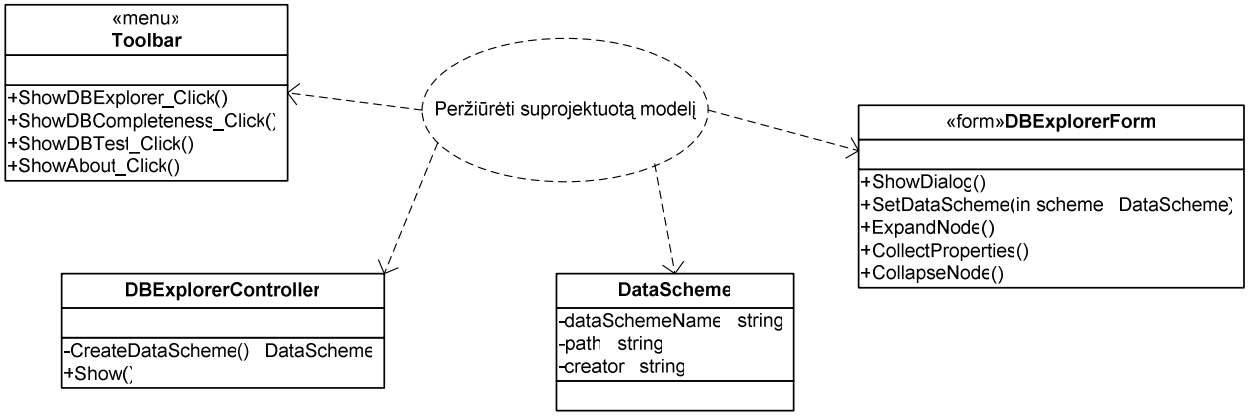
Šio projekto tikslas yra, remiantis analizės dalyje surinktais duomenimis, suprojektuoti ir sukurti duomenų modeliavimo ir schemos tikrinimo metodikos prototipą, kuris įgalintų DB projektuotoją vykdyti tokias funkcijas:

- Suprojektuoti duomenų bazės schemą (bus naudojamas CASE įrankio funkcionalumas);
- Peržiūrėti sukurtą duomenų bazės schemą;
- Fiziškai įdiegti DB schemą (bus realizuota CASE įrankyje) ir ją ištestuoti pagal modelyje turimus metaduomenis.

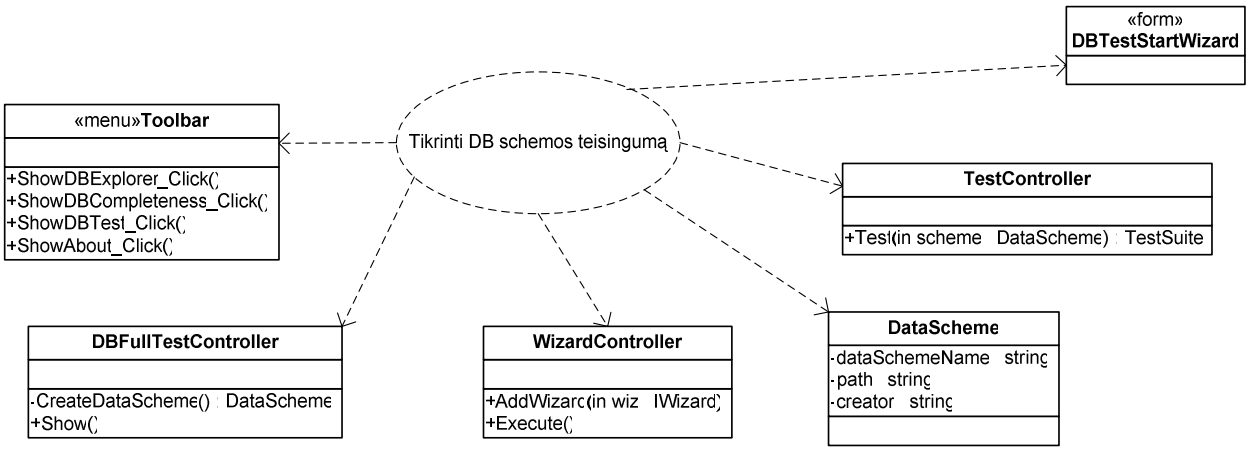
3.2.2. Sistemos panaudojimo atvejų realizacijos

Šiame poskyryje pateikiamos diagramos klasių, realizuojančių atitinkamus panaudojimo atvejus. Paveikslėliai pateikti tokia tvarka:

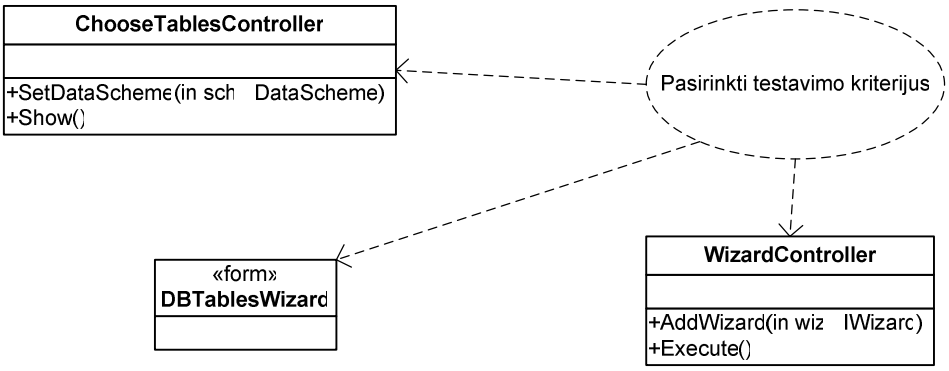
- 3.19 pav. – PA „Peržiūrėti suprojektuotą modelį“ realizacija;
- 3.20 pav. – duomenų schemos tikrinimo PA realizacija;
- 3.21 pav. – PA „Pasirinkti testavimo kriterijus“ realizacija;
- 3.22 pav. – PA „Įvesti testavimo duomenis“ realizacija;
- 3.23 pav. – PA „Peržiūrėti schemos testo rezultatus“ realizacija.



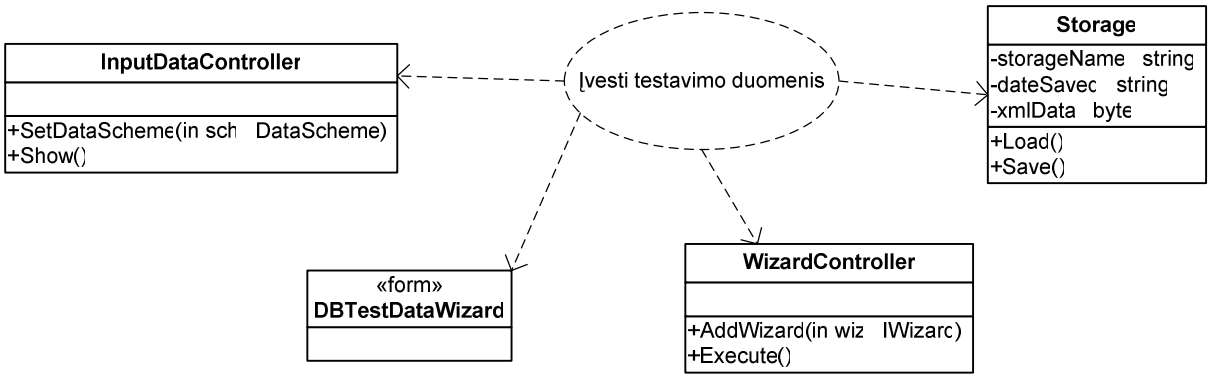
3.19 pav. Panaudojimo atvejį „Peržiūrėti suprojektuotą modelį“ realizuojančios klasės



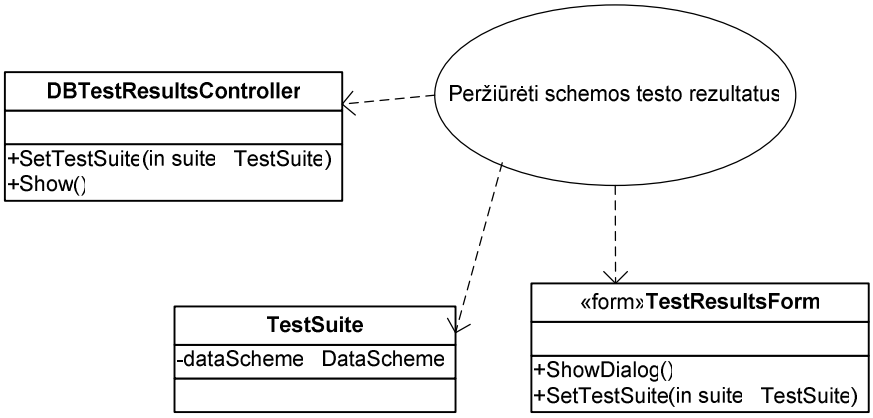
3.20 pav. Panaudojimo atvejus „Tikrinti DB schemos teisingumą“ ir „Tikrinti dalies schemos teisingumą“ realizuojančios klasės



3.21 pav. Panaudojimo atvejį „Pasirinkti testavimo kriterijus“ realizuojančios klasės



3.22 pav. Panaudojimo atvejį „Įvesti testavimo duomenis“ realizuojančios klasės

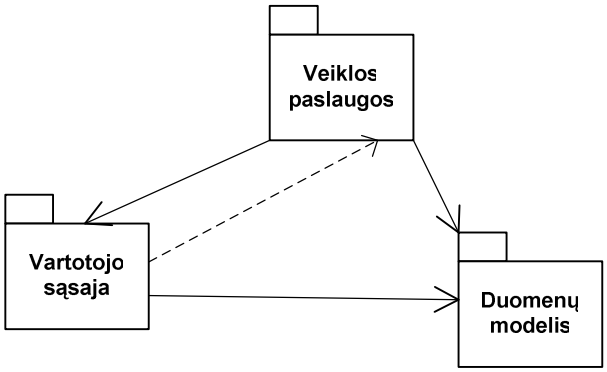


3.23 pav. Panaudojimo atvejį „Peržiūrėti schemos testo rezultatus“ realizuojančios klasės

3.2.3. Sistemos architektūra

Sistemos architektūros pagrindu bus laikomas klasikinis MVC (angl. *Model – View – Controller*) projektavimo šablonas. Programinės klasės sistemoje bus suskaidytos į tris paketus:

- Vartotojo sąsaja – šiame pakete laikomos klasės, atsakingos už bendravimą su vartotoju – meniu, formos, vedlio langai;
- Veiklos paslaugos – klasės, kurios, kaip matyti ši diagramos (3.24 pav.), turi sąryšį ir su vartotojo sąsajos modeliu, ir su duomenų modeliu. Šio paketo paskirtis yra valdyti programos eigą nepriklausomai nuo to, kaip bus realizuotos vartotojo sąsajos ir duomenų modelio klasės;
- Duomenų modelis – žemiausio lygio klasės, skirtos betarpiškam bendravimui su duomenų modeliu, t.y. su Visio duomenų bazės modelio schema bei testuojama MS SQL Server duomenų baze.



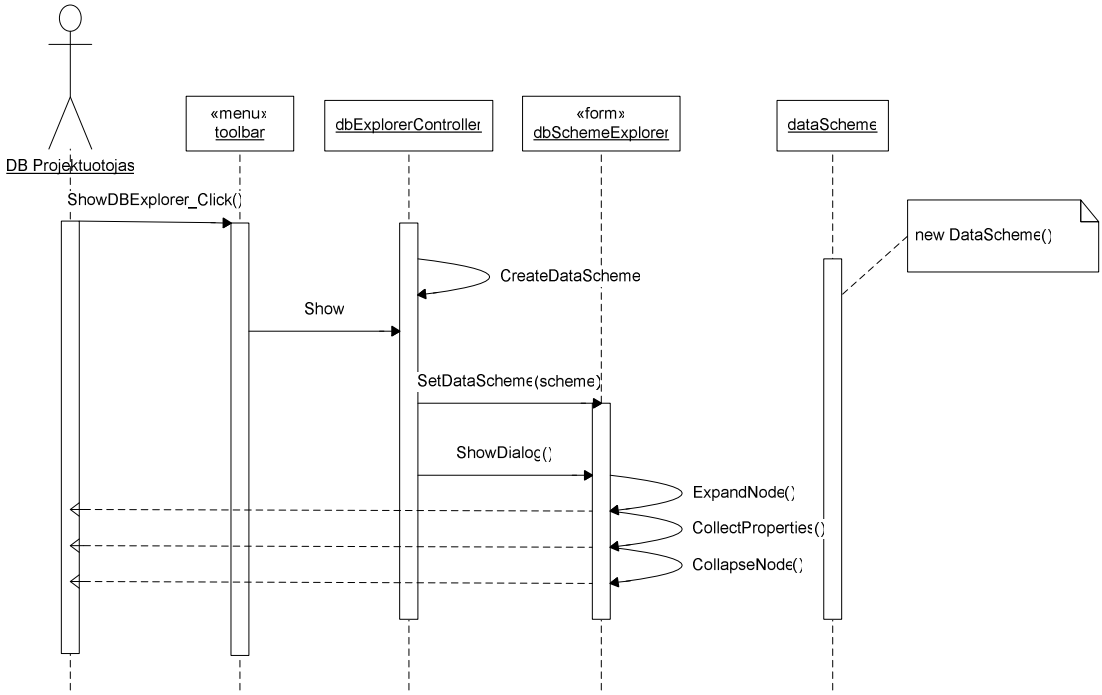
3.24 pav. Loginė sistemos architektūra

Privalumai sistemą realizuoti MVC šablono pagrindu yra tai, kad sistemos veiklos modelis bus nepriklausomas nuo vartotojo sąsajos. Taip suprojektuota sistema tenkins nefunkcinių reikalavimų dalį, susijusią su sistemos plėtimo lengvumu.

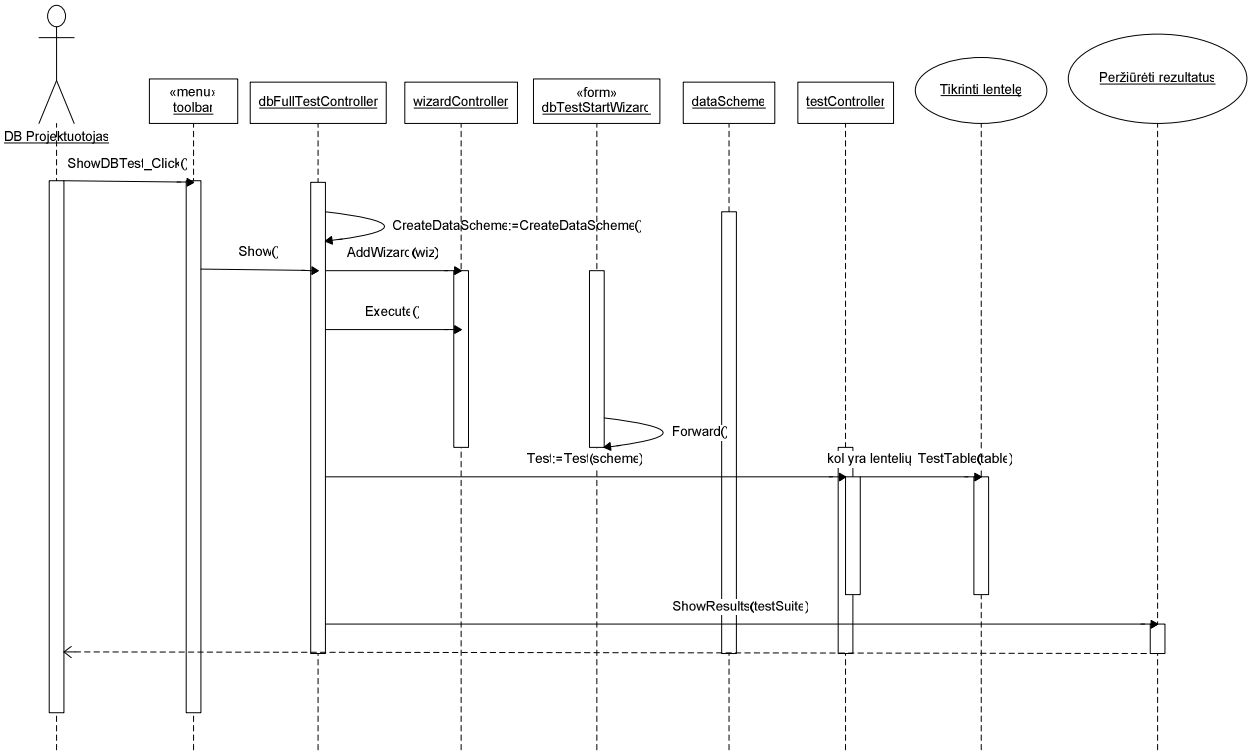
3.2.4. Sistemos elgsenos modelis

Šiame poskyryje pateikiamas svarbiausių panaudojimo atvejų elgsenos modelis, t.y. kiekvieno panaudojimo atvejo sekų diagrama. Diagramos pateikiamos tokia tvarka:

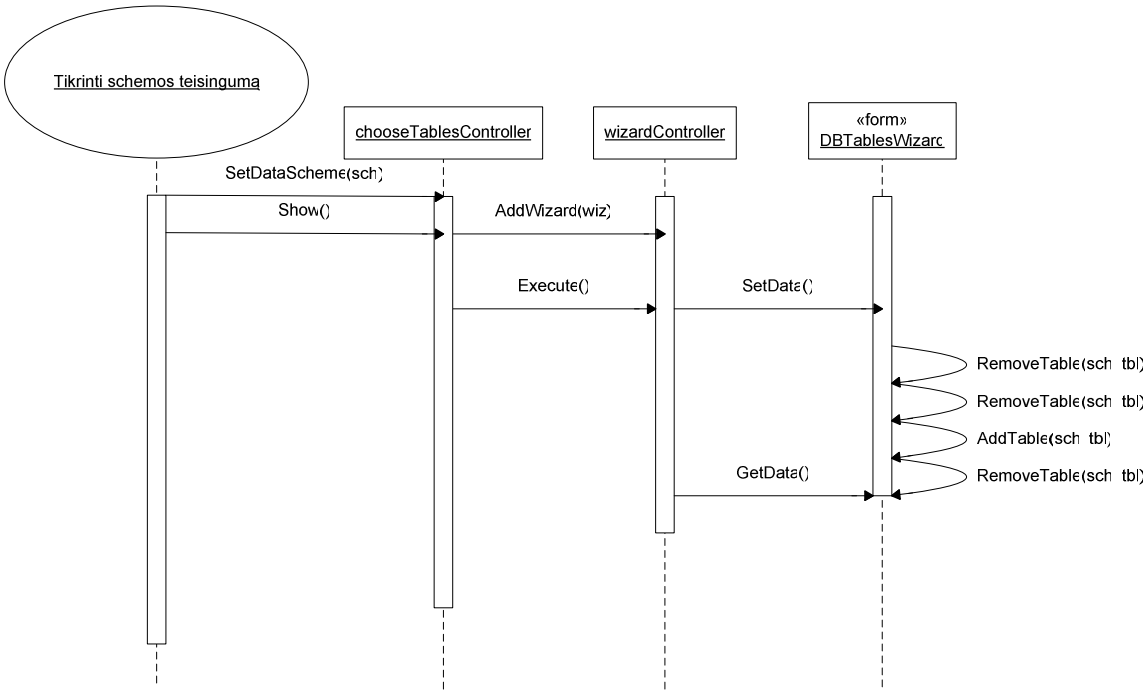
- 3.25 pav. – PA „Peržiūrėti suprojektuotą modelį“ sekų diagrama;
- 3.26 pav. – PA „Tikrinti DB schemos teisingumą“ sekų diagrama;
- 3.27 pav. – PA „Pasirinkti testavimo kriterijus“ sekų diagrama;
- 3.28 pav. – PA „Įvesti testavimo duomenis“ sekų diagrama;
- 3.29 pav. – PA „Peržiūrėti schemos testo rezultatus“ sekų diagrama.



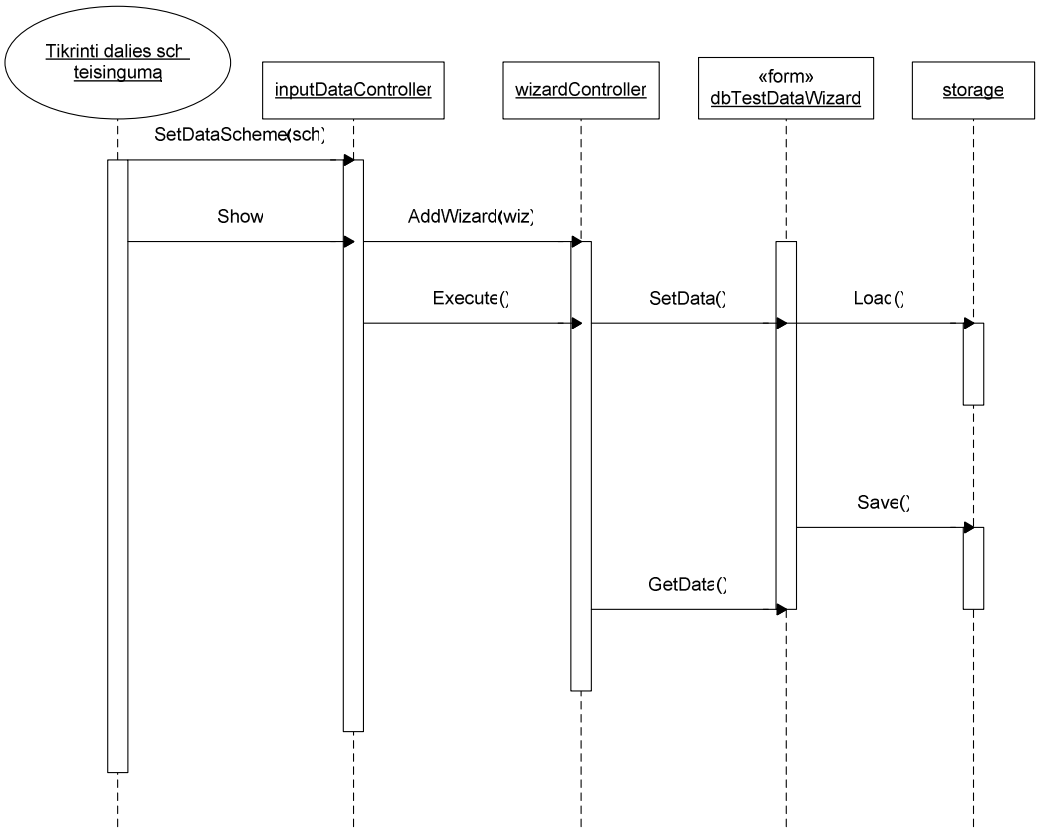
3.25 pav. Panaudojimo atvejį „Peržiūrėti suprojektuotą modelį“ paaiškinanti sekų diagrama



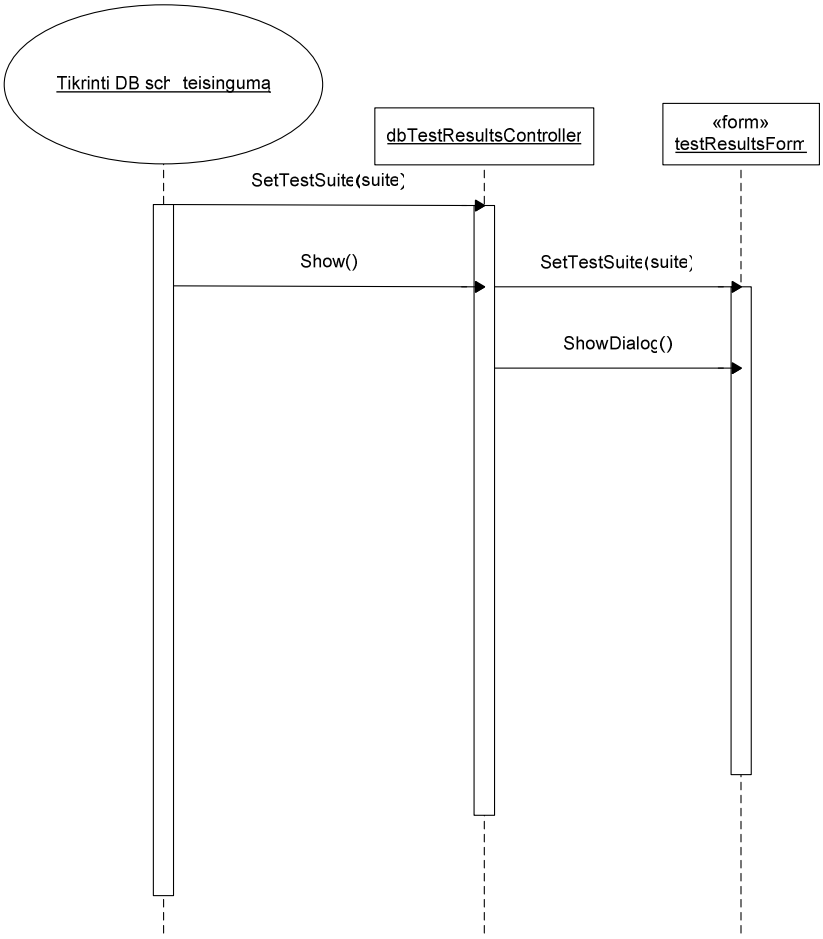
3.26 pav. Panaudojimo atvejį „Tikrinti DB schemos teisingumą“ paaiškinanti sekų diagrama



3.27 pav. Panaudojimo atvejį „Pasirinkti testavimo kriterijus“ paaiškinanti sekų diagrama

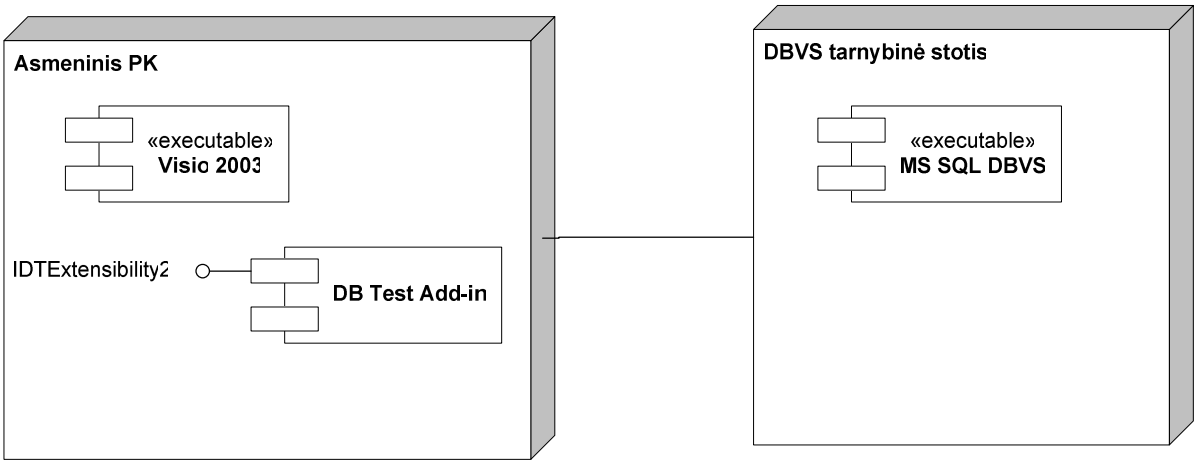


3.28 pav. Panaudojimo atvejį „Įvesti testavimo duomenis“ paaiškinanti sekų diagrama



3.29 pav. Panaudojimo atvejį „Peržiūrėti schemos testo rezultatus“ paaiškinanti sekų diagrama

3.2.5. Realizacijos modelis



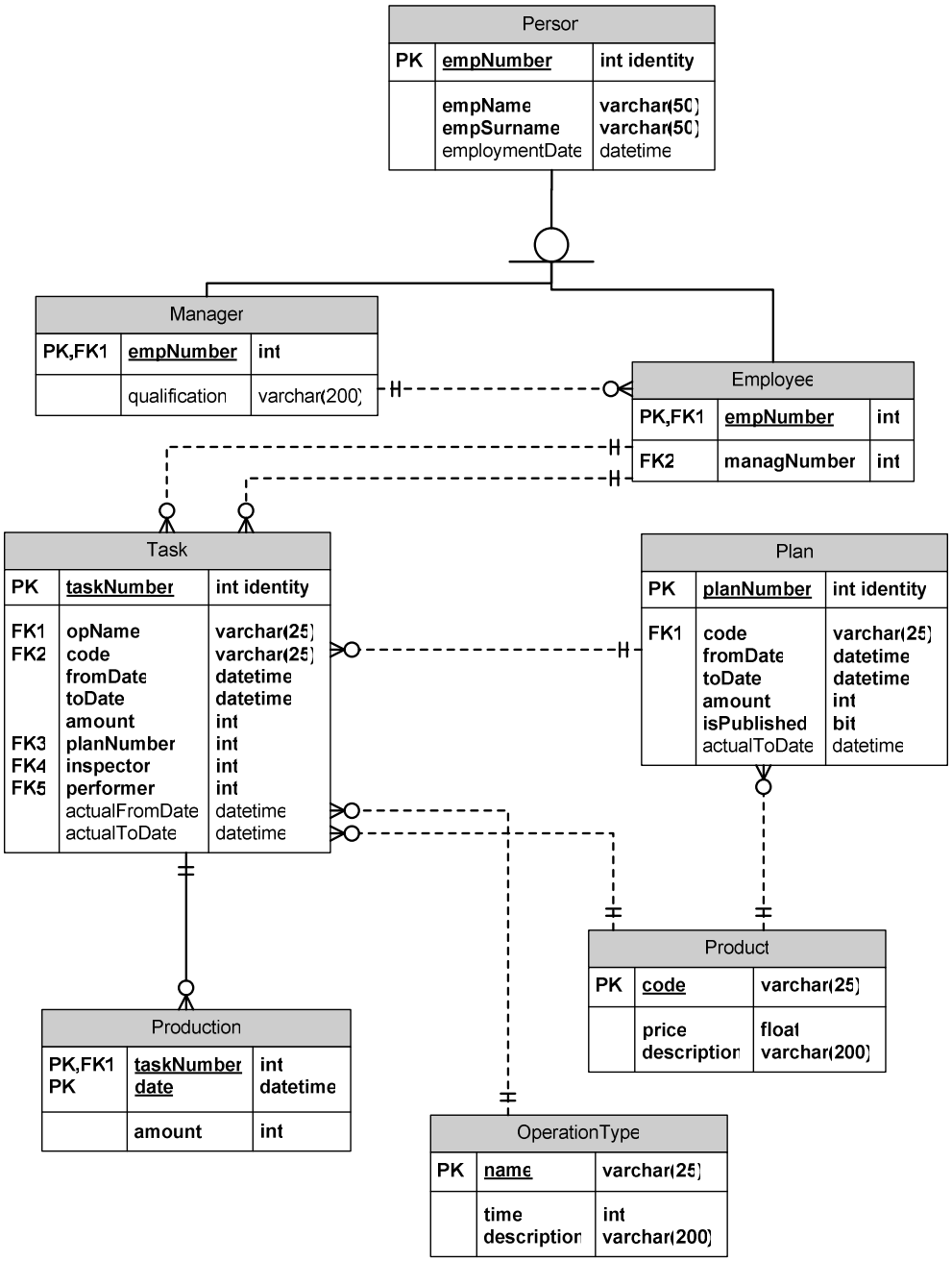
3.30 pav. Sistemos diegimo diagrama

Sistemos realizacijos modelį geriausiai paaiškina diegimo diagrama, kuri pavaizduota 3.30 paveiksle. DB schemos tikrinimo įskiepis bus diegiamas kliento kompiuteryje, o DBVS tarnybinėje stotyje turės būti užtikrintas prieinamumas duomenų bazės, kurią norėsime testuoti.

Kliento kompiuteryje, norint sėkmingai naudotis DB schemos tikrinimo įskiepiu, reikės turėti įdiegtą Microsoft Visio 2003 programinį paketą, ir taip pat įdiegtą DB testavimo įskiepį, kurį pateiks šio projekto kūrėjai. Norint pabrėžti, jog šis įskiepis bus kuriamas remiantis atitinkama Visio biblioteka, ir diegimo modelyje buvo parodyta, jog DB įskiepis kildinamas iš IDTExtensibility2 interfeiso, kuris leidžia sąveikauti su Microsoft Office dokumentais.

3.2.6. Eksperimentinis pavyzdys

Sistemos testavimo modelį sudarys bandomoji Visio schema, kurios pagrindai jau buvo paminėti sistemos analizės modelyje. Schema patobulinta ir transformuota į Microsoft Office Visio „Database Model Diagram“ tipo diagramą (3.31 pav.).



3.31 pav. Visio Database Model Diagram diagrama, naudojama sistemos testavimui

Employee

Lauko pavadinimas	Duomenų tipas	Apribojimas	Privalomas laukas	Komentaras
empNumber	int		✔	Pirminis identifikatorius (PI), nuoroda į Person
managNumber	int		✔	Vadovo ID

Manager

Lauko pavadinimas	Duomenų tipas	Apribojimas	Privalomas laukas	Komentaras

empNumber	int		✔	PI, nuoroda į Person
qualification	varchar	Iki 200 simb.		Kvalifikacinis laipsnis

OperationType

Lauko pavadinimas	Duomenų tipas	Apribojimas	Privalomas laukas	Komentaras
name	varchar	Iki 25 simb.	✔	PI, operacijos pavadinimas
time	int		✔	Laikas, per kurį turi būti atliekama operacija
description	varchar	Iki 200 simb.	✔	Apibūdinimas

Person

Lauko pavadinimas	Duomenų tipas	Apribojimas	Privalomas laukas	Komentaras
empNumber	int		✔	PI
empName	varchar	Iki 50 simb.	✔	Vardas
empSurname	varchar	Iki 50 simb.	✔	Pavardė
employmentDate	datetime			Įdarbinimo data

Plan

Lauko pavadinimas	Duomenų tipas	Apribojimas	Privalomas laukas	Komentaras
planNumber	int		✔	PI
code	varchar	Iki 25 simb.	✔	Produkto kodas
fromDate	datetime	fromDate <= toDate	✔	Data NUO (plano)
toDate	datetime	fromDate <= toDate	✔	Data IKI (plano)
amount	int		✔	Gamintinas kiekis
isPublished	bit		✔	Ar planas paskelbtas gamybai
actualToDate	datetime			Data IKI (fakto)

Product

Lauko pavadinimas	Duomenų tipas	Apribojimas	Privalomas laukas	Komentaras
code	varchar	Iki 25 simb.	✔	PI, produkto kodas
price	float		✔	Savikaina

description	varchar	Iki 200 simb.	✔	Apibūdinimas

Production

Lauko pavadinimas	Duomenų tipas	Apribojimas	Privalomas laukas	Komentaras
taskNumber	int		✔	Užduoties ID, dalis pirminio rakto
date	datetime		✔	Gamybos data, dalis pirminio rakto
amount	int		✔	Pagamintas kiekis

Task

Lauko pavadinimas	Duomenų tipas	Apribojimas	Privalomas laukas	Komentaras
taskNumber	int		✔	PI
opName	varchar	Iki 25 simb.	✔	Operacijos pavadinimas
code	varchar	Iki 25 simb.	✔	Produkto kodas
fromDate	datetime	fromDate <= toDate	✔	Data NUO (plano)
toDate	datetime	fromDate <= toDate	✔	Data IKI (plano)
amount	int	<= 100 vnt.	✔	Gamintinas kiekis
planNumber	int		✔	Plano nr.
inspector	int	inspector <> performer	✔	Prižiūrintis darbuotojas
performer	int	inspector <> performer	✔	Gaminantis darbuotojas
actualFromDate	datetime			Data NUO (fakto)
actualToDate	datetime			Data IKI (fakto)

3.2.7. Reikalavimai sistemos funkcionalumo palaikymui

DB schemos modeliavimo ir tikrinimo metodikos įskiepis suprojektuotas ir sukurtas taip, kad būtų prieinamas kuo didesnei daliai potencialių naudotojų. Kompiuteriui, kuriame norima naudotis DB testų įskiepiu, nebus keliami dideli reikalavimai, priešingai – tvarkingame kompiuteryje, pastoviai atnaujinamame per „Microsoft Update“ tarnybą, DB schemos įskiepis bus paleidžiamas be jokių nesklandumų.

Pagrindiniai reikalavimai programinei įrangai DB serveryje:

- Microsoft SQL Server 2000 arba SQL Server 2005 pagrindu veikianti RDBVS.

Su šia duomenų baze bus sąveikaujama testuojant DB schemą.

Pagrindiniai reikalavimai programinei įrangai kliento kompiuteryje yra tokie:

- Operacinė sistema, palaikanti .NET platformą (Windows 98 SE arba aukštesnė);
- Microsoft .NET 1.1 vykdymo aplinka;
- Microsoft Office Visio 2003 arba Microsoft Visio 2002 programinis paketas.

Klientinės dalies komponentai nereikalauja didesnių sistemos resursų nei rekomenduojami dirbti su minėtais Microsoft korporacijos paketais: .NET 1.1 bei Visio. Klientinės dalies komponentas nedirbs su sistemos registru, visus reikalingus duomenis saugodamas duomenų bazėje arba lokalioje XML rinkmenoje, todėl šiuo įrankiu be problemų galės naudotis administratoriaus teisių kompiuteryje neturintys vartotojai.

Klientinės dalies kompiuterio ir tarnybinės stoties aptarnavimas niekuo nesiskiria nuo standartinių kompiuterių priežiūros reikalavimų:

- į kompiuterį turi būti diegiami gamintojo rekomenduojami atnaujinimai;
- kompiuteris turi būti apsaugotas nuo virusų ar įsilaužėlių atakų;
- SQL serverio duomenų bazės turi būti reguliariai, pagal įmonėje nusistovėjusią tvarką, archyvuojamos.

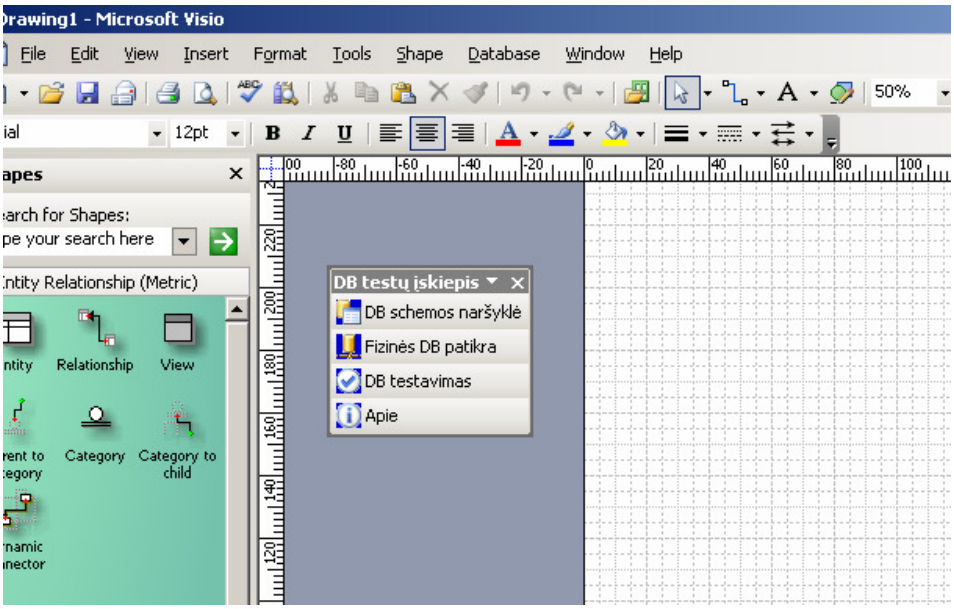
4. Eksperimentinis tyrimas

4.1. Sistemos veikimo aprašymas

Darbo metu suprojektuota ir sukurta sistema „DbTestAddin“, kuri atlieka duomenų bazių schemų teisingumo tikrinimą. Sistema veikia kaip Microsoft Office Visio 2002 / 2003 įskiepis.

Sistemai įdiegti sukurtas diegimo vedlys, kurio veikimo principas sutampa su standartinių Windows diegimo vedlių logika.

Paleidę Microsoft Office Visio programą bei atsidarę duomenų bazių schemų kūrimo formą, iškart pamatysime „DbTestAddin“ įskiepio įrankinę (4.1 pav.).



4.1 pav. Microsoft Office Visio langas su „DBTestAddin“ įskiepiu

Įskiepis valdomas keturiais pagrindiniais mygtukais:

- **DB schemos naršyklė** – mygtukas, paleidžiantis duomenų bazės schemos, nupaišytos Visio lange, naršyklę;
- **Fizinės DB patikra** – mygtukas, paleidžiantis fizinės duomenų bazės (nutolusios arba lokalios) tikrinimą, lyginant ją su Visio suformuota schema;
- **DB testavimas** – mygtukas, paleidžantis DB testavimą;
- **Apie** – mygtukas, parodantis informaciją apie sistemą, jos autorius bei versiją.

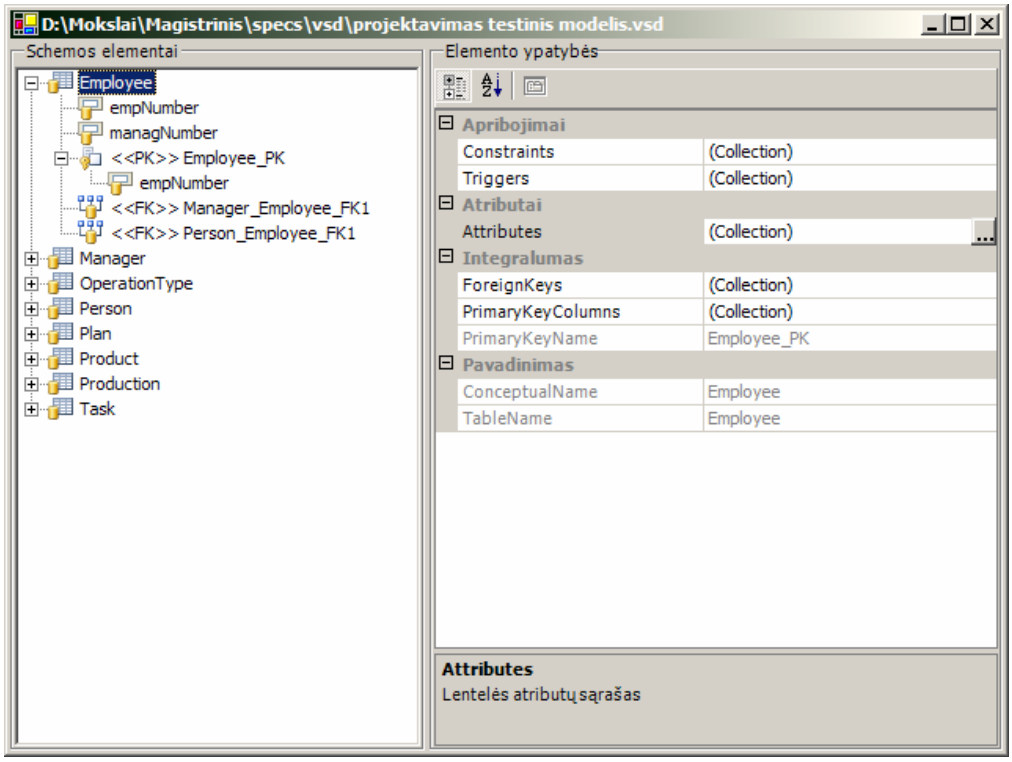
4.1.1. DB schemos naršyklė

Paspaudus mygtuką „DB schemos naršyklė“ atsidaro langas (4.2 pav.), kuriame iškart matoma visa Microsoft Office Visio formoje nupaišytos duomenų bazės schemos informacija. Informacija pateikiama medžio principu (angl. *treeview*).

Pirminiame medžio lygyje matome lenteles, gilesniame – atributus bei kitus lentelės elementus – pirminius, išorinius raktus.

Pažymėję kurį nors elementą ar atributą, dešinėje lango dalyje galime pamatyti visą informaciąj apie pažymėtą objektą. Yra galimybė atitinkamus to objekto laukus rūšiuoti pagal abėcelę arba temas.

Apatinėje dešinėje lango dalyje matome lietuvišką pažymėto objekto paaiškinimą.



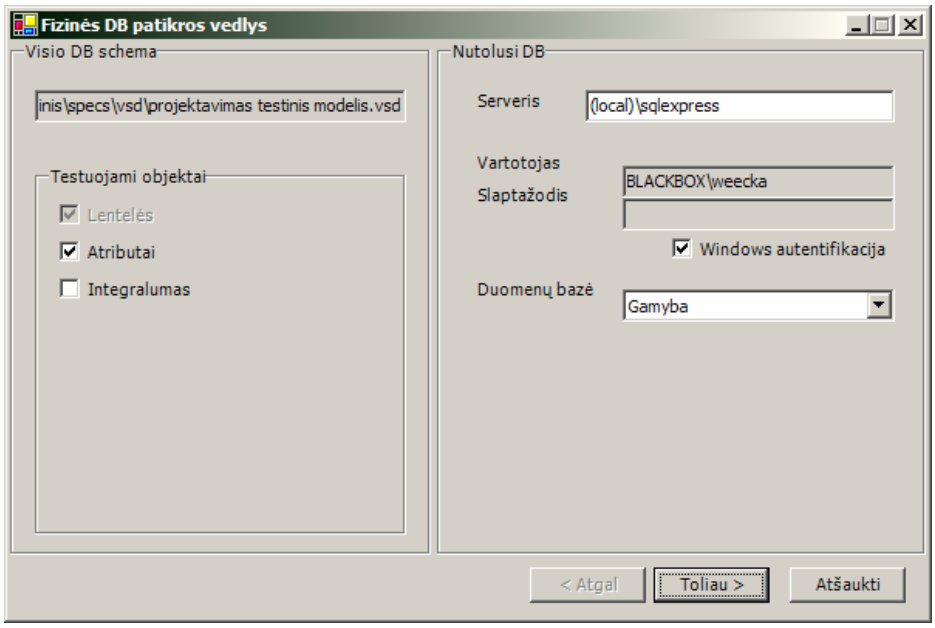
4.2 pav. „DB schemos naršyklės“ langas

4.1.2. Fizinės DB patikra

Mygtukas „Fizinės DB patikra“ skirtas paleisti DB tikrinimo/lyginimo funkciją. Sistema patikrina, ar DB schema, suformuota Visio ir esanti realioje duomenų bazėje, yra identiškos.

Paspaudus mygtuką „Fizinė DB patikra“, atsidaro fizinės DB patikros vedlys. Pirmajame jo lange (4.3 pav.), palime pasirinkti kokius objektus tikrinsime (dalis „Testuojami objektai“), nustatyti nutolusios duomenų bazės prisijungimo nustatymus (dalis „Nutolusi DB“).

Testui pavykus, visi nustatymai yra įsimenami vartotojo „ApplicationData“ kataloge, ir pakraunami kito testo metu.

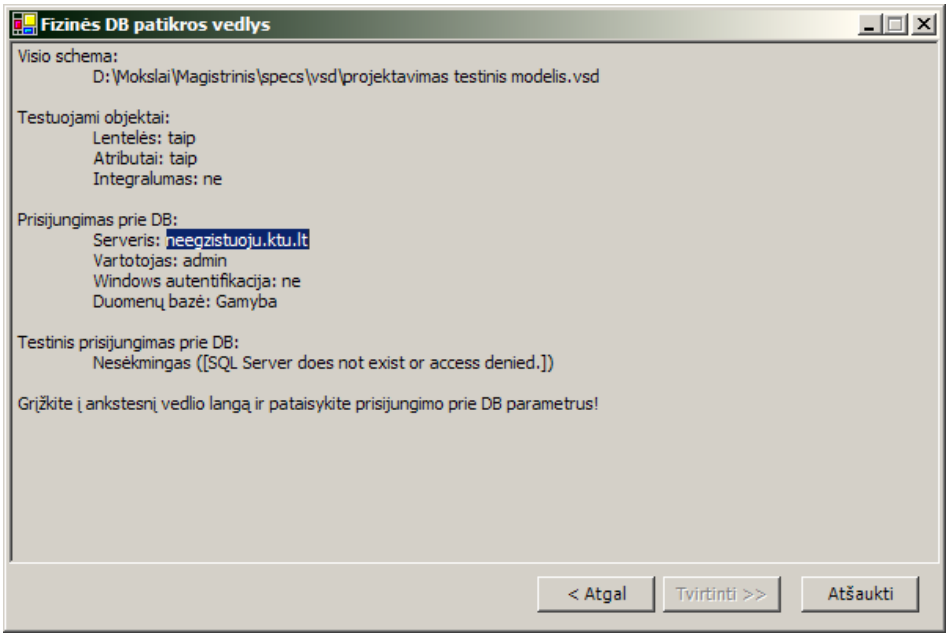


4.3 pav. „Fizinės DB patikros“ vedlio pirmasis langas

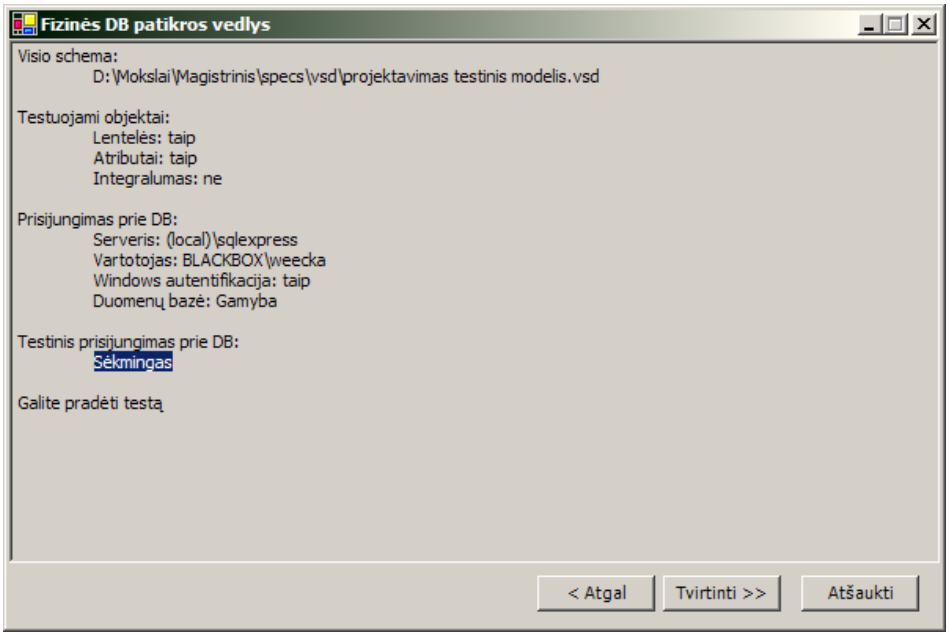
Paspaudus mygtuką „Toliau >“ atsidaro antras fizinės DB patikros vedlio langas (4.4 pav.). Jame galime pamatyti/patikrinti nustatymus, kuriuos pasirinkome prieš tai buvusiame žingsnyje (jei nustatymai netenkina, galima spausti mygtuką „< Atgal“, kuris sugražins į prieš tai buvusį langą).

Taip pat sistema patikrina prisijungimą prie nutolusio DB serverio. Jei prisijungimas sėkmingas (4.5 pav.), vartotojas apie tai informuojamas ir aktyvuojamas mygtukas „Tvirtinti >>“, kurį paspaudus paleidžiamas DB tikrinimas.

Jei prisijungimo tikrinimas nesėkmingas, atsiranda pranešimas prašantis patikrinti prisijungimo nustatymus (4.4 pav.). Nesėkmingo prisijungimo atveju sistema taip pat parodo informaciją apie nesėkmės priežastį.



4.4 pav. „Fizinės DB patikros“ vedlio antrasis langas (nesėkmingas prisijungimas)

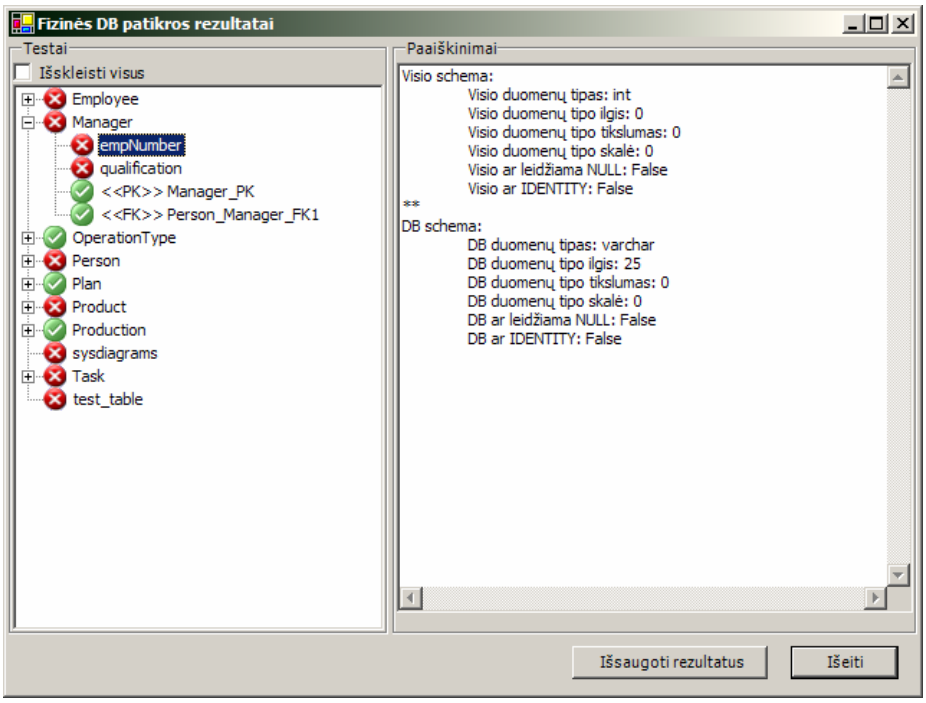


4.5 pav. „Fizinės DB patikros“ vedlio antrasis langas (sėkmingas prisijungimas)

Paspaudus mygtuką „Tvirtinti >>“, paleidžiamas fizinės DB tikrinimas. Jam pasibaigus, sistema parod fizinės DB patikros rezultatų langą (4.6 pav.), kuriame matomi tikrinimo rezultatus.

Tikrinimo rezultatai pateikti taip pat medžio principu. Aukščiausiame lygyje matomos lentelės, žemesniame – atributai, išoriniai bei pirminiai raktai (analogiškai kaip schemos naršyklėje). Paspaudus ant pasirinkto objekto, dešinėje lango dalyje pamatysime išsamią tikrinimo rezultatų informaciją.

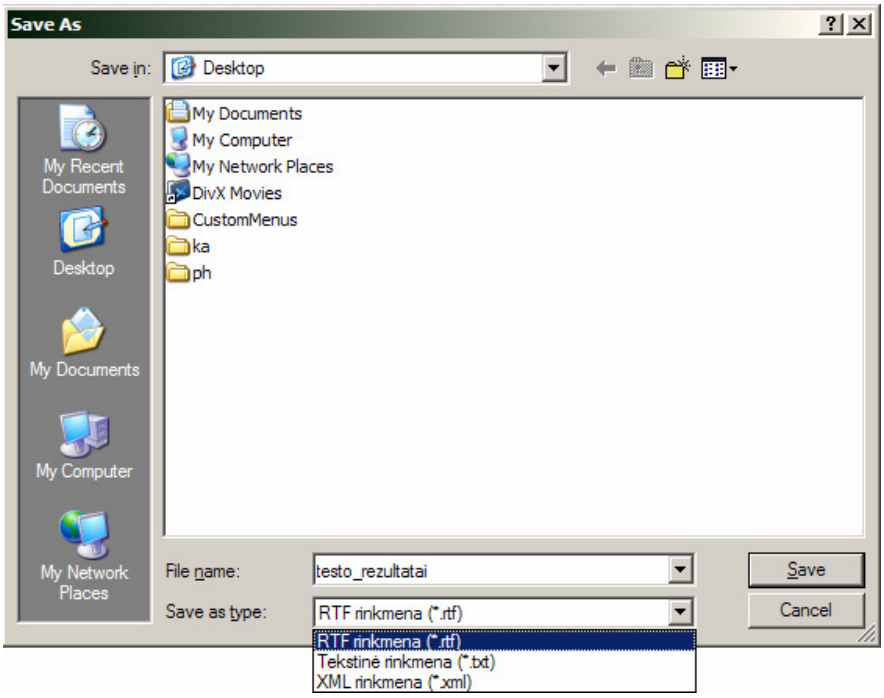
Prie kiekvieno objekto yra skirtingos piktogramos. „Žalia varnelė“ reiškia, kad objekto tikrinimas buvo sėkmingas, „Raudonas kryžiukas“ – tikrinimas nesėkmingas.



4.6 pav. „Fizinės DB patikros“ rezultatų langas

Tikrinimo rezultatus galima išsaugoti išoriniuose failuose. Paspaudus mygtuką „Išsaugoti rezultatus“, atsidaro dialogo langas (4.7. pav.), kuriame galima pasirinkti, kokiu formatu norime išsaugoti rezultatus. Įmanomi trys formatai:

- RTF: eksportuojama į tekstinį formatą su formatavimu;
- TXT: tokia pati informacija be formatavimo;
- XML: hierarchinė rezultatų schema, kurią, esant reikalui, galima apdoroti su specifine XML apdorojimo programa.



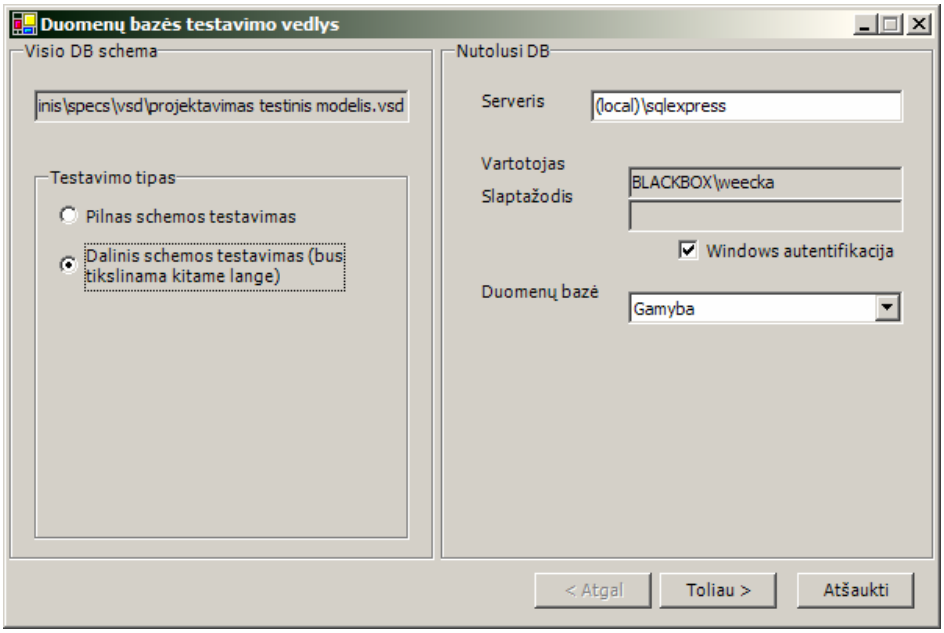
4.7 pav. „Fizinės DB patikros“ rezultatų išsaugojimo dialogo langas

4.1.3. DB testavimas

Mygtukas „DB testavimas“ skirtas paleisti DB testavimo funkcijai. Sistema patikrina DB schemos teisingumą, naudodama realius duomenis.

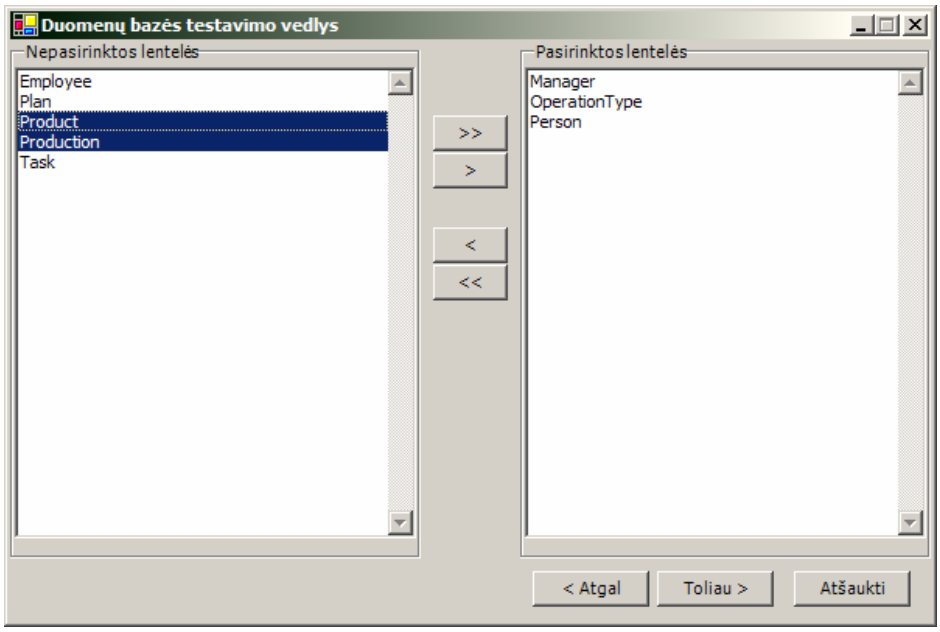
Paspaudus mygtuką „DB testavimas“, atsidaro duomenų bazės testavimo vedlys. Pirmajame jo lange (4.8 pav.) galime pasirinkti, kokį testavimo tipą naudosime (dalis „Testavimo tipas“), nustatyti nutolusios duomenų bazės prisijungimo parametrus (dalis „Nutolusi DB“).

Kaip ir fizinėje DB patikroje, testui pavykus, visi nustatymai yra įsimenami vartotojo „ApplicationData“ kataloge ir pakraunami kito seanso metu.



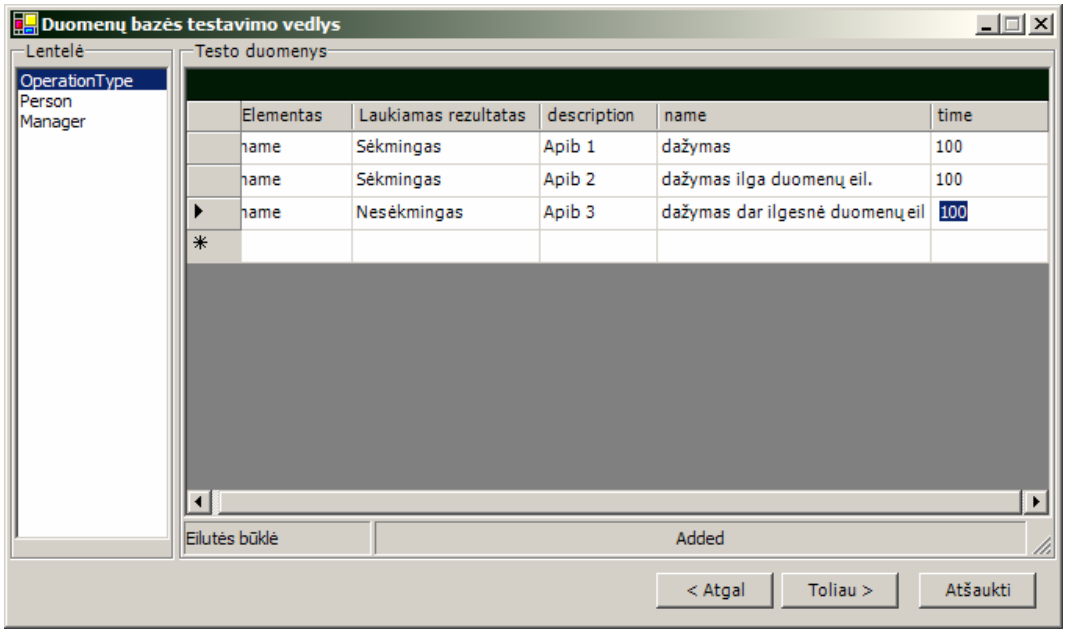
4.8 pav. „DB testavimo“ vedlio pirmasis langas

Jei pasirinkome testavimo tipą „Dalinis schemos testavimas“, paspaudus mygtuką „Toliau >“ atsidaro DB testavimo vedlio antrasis langas (4.9. pav.), kuriame galima pasirinkti, kokias lenteles testuosime. Lentelių sąrašas patekiama kairėje lango dalyje. Mygtukais „>“, „>>“ ir „<“, „<<“ galime norimas lentelės pridėti ar atimti iš testuojamųjų sąrašo.



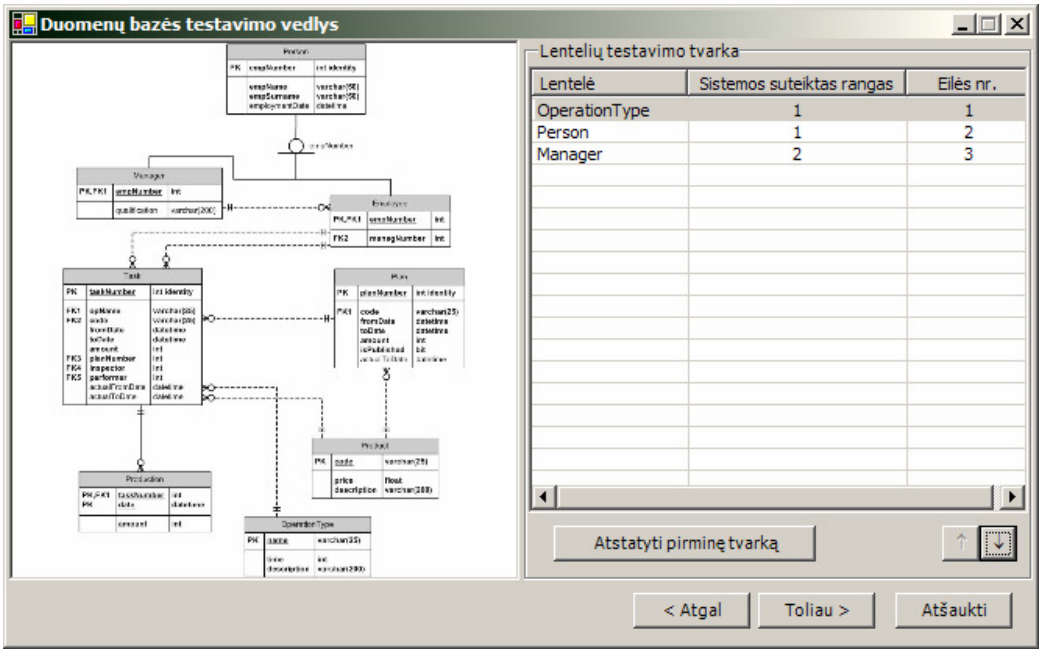
4.9 pav. „DB testavimo“ vedlio antrasis langas

Paspaudus mygtuką „Toliau >“ atsidaro trečiasis duomenų bazės testaivo vedlio langas, kuriame galima įvesti norimus testavimo duomenis ranka. Dėšinėje lango pusėjė matomas lentelių sąrašas, kairėje – testavimo duomenys.



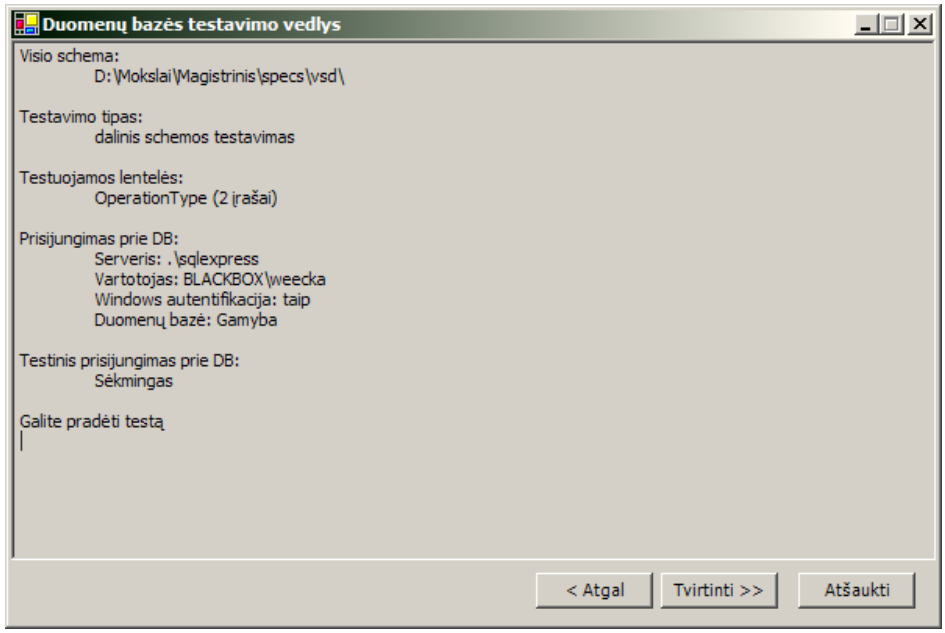
4.10 pav. „DB testavimo“ vedlio trečiasis langas

Paspaudus mygtuką „Toliau >“ atsidaro ketvirtasis duomenų bazės testavimo vedlio langas (4.11 pav.), kuriame galima rikiuoti lenteles norima tvarka (pirminį rikiavimą atliks DB testavimo įskiepis). Kairėje pateikiamas einamosios Visio schemos vaizdas, dešinėje – rikiavimo laukas (rikiavimas vykdomas rodyklėlių arba drag‘n‘drop pagalba). Mygukas „Atstatyti pirminę tvarką“ skirtas pirminiam lentelių rūšiavimui (koks buvo pateiktas sistemos) atstatyti.



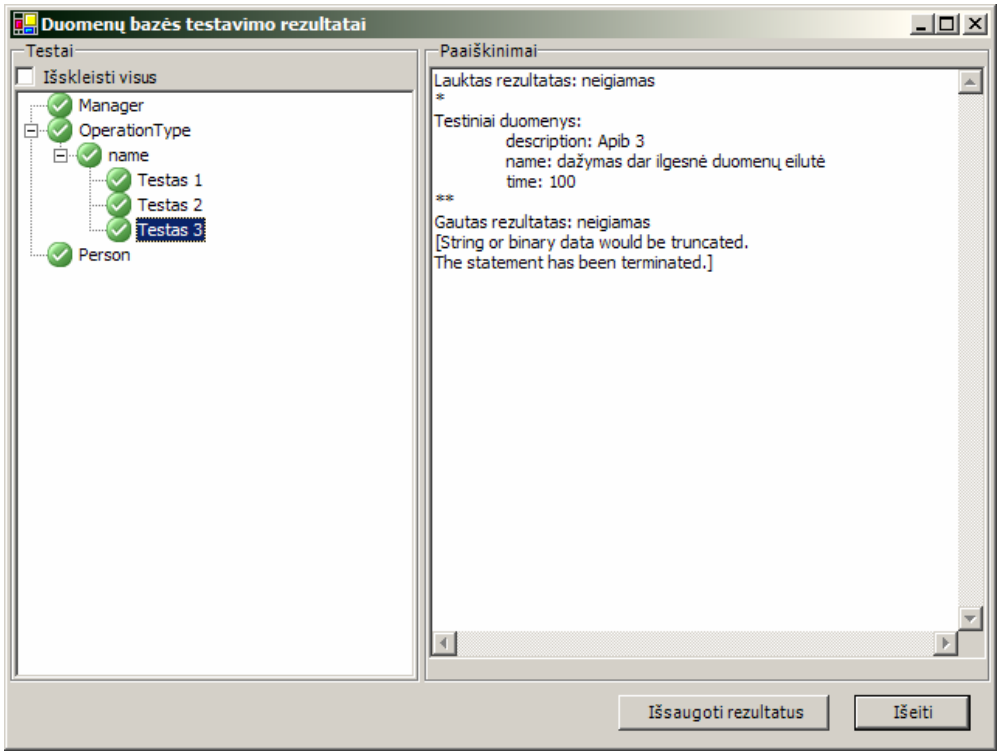
4.11 pav. „DB testavimo“ vedlio ketvirtasis langas

Paspaudus mygtuką „Toliau >“ atsidaro testavimo rezultatų patvirtinimo bei serverio prisijungimo patikrinimo langas (4.12 pav.).



4.12 pav. „DB testavimo“ vedlio patvirtinimo langas

Testavimo rezultatai pateikiami beveik analogiškai kaip „Fizinėje DB patikroje“ (4.13 pav.), tik prie kiekvieno testuojamo elemento parodoma visų su juo atliktų testų informacija. Testiniame pavyzdyje, kuris buvo rodomas visuose ankstesniuose languose, lentelės OperationType atributui name buvo įvestos trys testinės eilutės, jos ir yra parodomos po atributu – šios testinės eilutės rezultatų lange pavadintos „Testas 1“, „Testas 2“ bei „Testas 3“.



4.13 pav. „DB testavimo“ rezultatų langas

4.2. Sistemos veikimo eksperimentinis pavyzdys

Sukurtos sistemos tikslas – duomenų bazės testavimo vykdymas. Sėkmingas testavimas betarpiškai priklauso nuo testinių duomenų, kuriuos į sistemą suves vartotojas. Sistemos testų veikimo principas yra labai panašus į vienetų testavimo (angl. *unit testing*) principą: vartotojas įveda vieną lentelės duomenų porciją, ir nurodo, ar ši eilutė bus perkelta į DB sėkmingai (laukiamas rezultatas – sėkmingas), ar nesėkmingai (laukiamas rezultatas – nesėkmingas, t.y. SQL *INSERT* operacijos metu bus gauta klaida iš DB). Vartotojas taip pat gali testuoti ir duomenų atnaujinimo (angl. *UPDATE*) arba trynimo (angl. *DELETE*) operacijas, tačiau jos šiame pavyzdyje nenagrinėjamos.

Pateikiame vienos lentelės testavimo scenarijų, testavimo atvejus bei rezultatus, kurie leis aiškiau suprasti sistemos veikimo principus, o tuo pačiu įrodys, kad sistema veikia be klaidų. Testavimui pasirinkta lentelė *Task*, nes remiantis ja, galima, sumodeliuoti daugiausiai testavimo atvejų. Išorinio rakto testavimui taip pat bus naudojama susieta lentelė *Product*.

Task

Lauko pavadinimas	Duomenų tipas	Apribojimas	Privalomas laukas	Komentaras
taskNumber	int		✔	PI
opName	varchar	Iki 25 simb.	✔	Operacijos pavadinimas
code	varchar	Iki 25 simb.	✔	Produkto kodas
fromDate	datetime	fromDate <= toDate	✔	Data NUO (plano)
toDate	datetime	fromDate <= toDate	✔	Data IKI (plano)
amount	int	<= 100 vnt.	✔	Gamintinas kiekis
planNumber	int		✔	Plano nr.
inspector	int	inspector <> performer	✔	Prižiūrintis darbuotojas
performer	int	inspector <> performer	✔	Gaminantis darbuotojas
actualFromDate	datetime			Data NUO (fakto)
actualToDate	datetime			Data IKI (fakto)

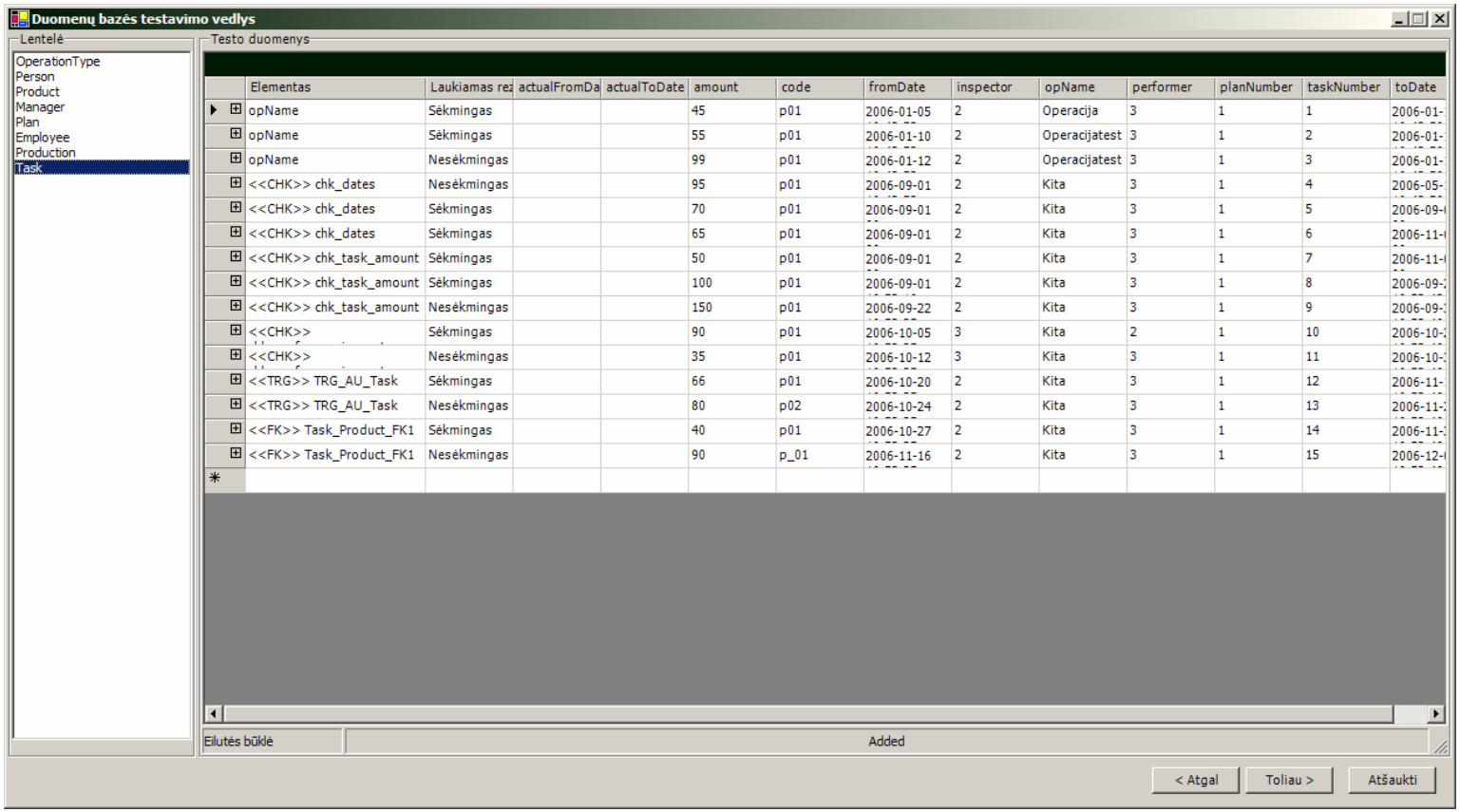
4.1 lentelė. Testavimo atvejai lentelei „Task“

Testavimo atvejis	Testuojamas elementas	Testuojamas objekto apribojimas	Įvedami testavimo duomenys	Testavimo kometarai	Laukiami testo rezultatai	Sistemos pateikti testo rezultatai
Duomenų tipo apribojimo testavimas	opName	Iki 25. simbolių	Įvedamos opName reikšmės: 1. Operacija 2. Operacijatestavimasvienas 3. Operacijastestavimasvienaspradeti	Įvedamos trys testinės eilutės, kurių lauko *opName* reikšmės yra kaip nurodytos. Testui pasirenkamos ribinės reikšmės, t.y. pirmuoju atveju įvedama tekstinė eilutė, kurios ilgis mažesnis nei 25 simboliai, kitu atveju – lygiai 25 simboliai, ir galiausiai – ilgesnė nei 25 simboliai eilutė. Kitų laukų duomenys įvedami tokie, kokie vartotojui atrodo teisingi.	1. Sėkmingas 2. Nesėkmingas 3. Nesėkmingas	1. Sėkmingas 2. Nesėkmingas 3. Nesėkmingas
Pradžios data turi būti ankstesnė nei pabaigos data	<<CHK>> chk_dates `ALTER TABLE [dbo].[Task] WITH CHECK ADD CONSTRAINT [chk_dates] CHECK ([fromDate]<=[toDate])`	fromDate <= toDate	Įvedamos fromDate ir toDate reikšmės: 1. fromDate: 2006-09-01; toDate: 2006-05-01; 2. fromDate: 2006-09-01; toDate: 2006-09-01; 3. fromDate: 2006-09-01; toDate: 2006-11-01.	Įvedami trys testavimo atvejai su taip pat ribinėmis reikšmėmis: pirmu atveju fromDate yra didesnė nei toDate, todėl turi suveikti „CHECK“ apribojimas, ir duomenų įterpimas turi nepavykti. Kitais dviem atvejais įvedama užduoties pabaigos data tokia pat kaip ir pradžios bei vėlesnė (šios dvi duomenų eilutės turi būti įterptos sėkmingai).	1. Nesėkmingas 2. Sėkmingas 3. Sėkmingas	1. Nesėkmingas 2. Sėkmingas 3. Sėkmingas

Kiekio reikšmės apribojimas	<<CHK>> chk_task_amount `ALTER TABLE [dbo].[Task] WITH CHECK ADD CONSTRAINT [chk_task_amount] CHECK ([amount] <= 100)`	<= 100 vnt.	Įvedamos amount reikšmės: 1. 50 2. 100 3. 150	Šiam „CHECK“ apribojimui įvedamos trys testinės eilutės: 1 eilutė amount < 100 2 eilutė amount = 100 3 eilutė amount > 100. Trečiuoju atveju duomenų įterpimas turi būti nesėkmingas, pirmais dviem – pavykti.	1. Sėkmingas 2. Sėkmingas 3. Nesėkmingas	1. Sėkmingas 2. Sėkmingas 3. Nesėkmingas
Prižiūrintis darbuotojas negali būti tas pats kaip ir gaminantis darbuotojas	<<CHK>> chk_performer_inspector `ALTER TABLE [dbo].[Task] WITH CHECK ADD CONSTRAINT [chk_performer_inspector] CHECK ([performer] <> [inspector])`	performer <> inspector	Įvedamos performer ir inspector reikšmės: 1. performer: 002; inspector: 002; 2. performer: 002; inspector: 003.	Įvedami du testai: viename performer ir inspector laukuose rašomi vienodi darbuotojų kodai (jie turi egzistuoti lentelėje Employee!), kitame teste – skirtingi. Pirmu atveju duomenų įterpti neleis CHECK apribojimas, antru atveju – turi pavykti.	1. Nesėkmingas 2. Sėkmingas	1. Nesėkmingas 2. Sėkmingas
Produkto kodas lentelėje Task turi	<<TRG>> TRG_AU_Task	Trigeris, parodantis klaidą	Įvedamas planas Plan, kurio produktu pasirenkamas pvz. „p01“, o to plano numeris yra 1.	Pirmoji eilutė turi būti įterpiama sėkmingai. Antruoju atveju trigeris turi išmesti	1. Sėkmingas 2. Nesėkmingas	1. Sėkmingas 2. Nesėkmingas

atitikti produkto kodą lentelėje Plan		vedant planui nepriskirto produkto kodą	Į užduotį Task vedamos dvi eilutės: 1. planNumber: 1; code: „p01“; 2. planNumber: 1; code: „p02“.	klaidos pranešimą. Trigerio tekstas: `CREATE TRIGGER [dbo].[TRG_AU_Task] ON [dbo].[Task] AFTER INSERT AS DECLARE @cnt INT BEGIN SET NOCOUNT ON; SELECT @cnt = COUNT(*) FROM INSERTED ii JOIN [Plan] p ON p.planNumber = ii.planNumber JOIN Product pr ON pr.code = p.code WHERE ii.code != p.code IF @cnt > 0 BEGIN RAISERROR('Į užduotį bandoma įvesti produktą, nenumatytą plane', 16, 1) RETURN END END`		
Išorinio rakto testavimas	<<FK>> Task_Product_FK1	Išorinis raktas	Ivedamos code reikšmės: 1. ab123 2. ab124	Ivedami du testai: 1. reikšmė, kuri yra lentelėje Product laukelyje code.	1. Sėkmingas (yra susijusių duomenų	1. Sėkmingas 2. Nesėkmingas

				2. reikšmė, kurios nėra lentelėje Product laukelyje code.	Product lentoje) 2. Nesėkmingas (kodas neįvestas į Product lentelę)	

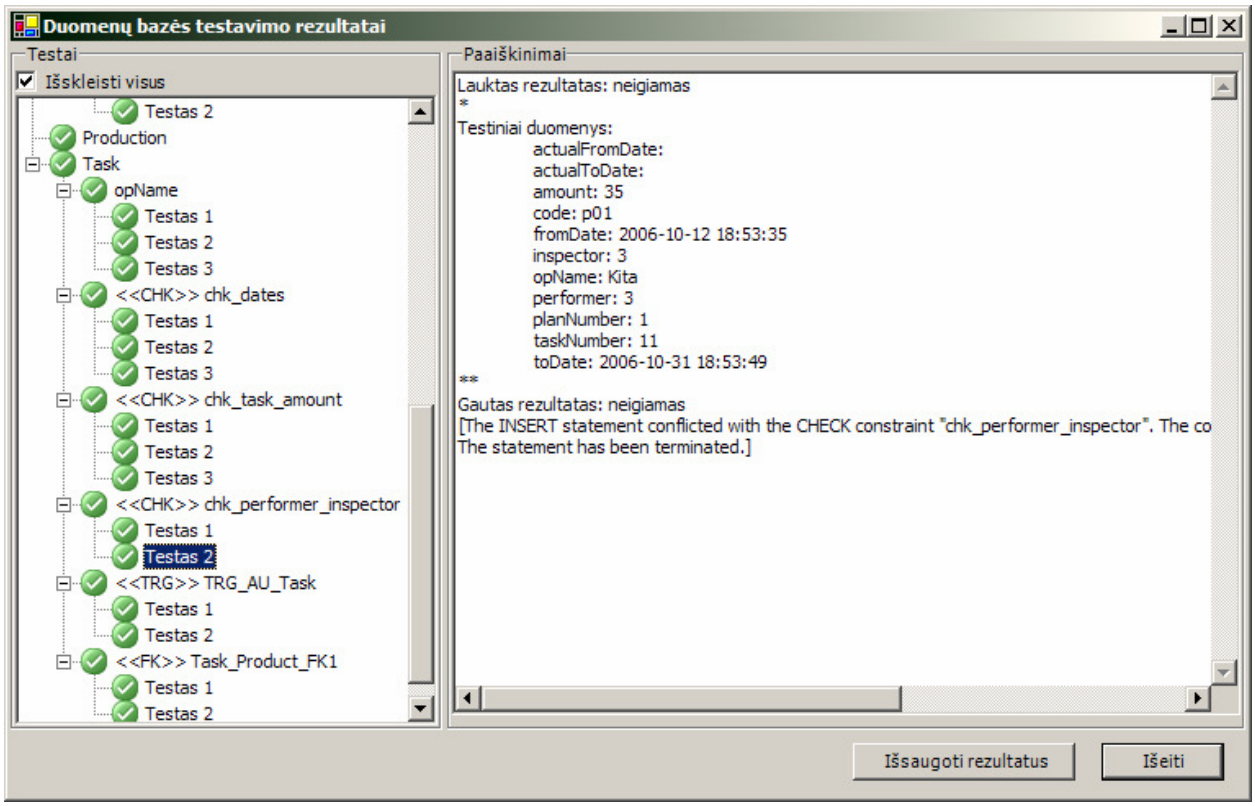


4.14 pav. Testiniai duomenys Task lentelei

Kaip matome iš testinio pavyzdžio, lentelė „Task“ yra glaudžiai susijusi su beveik visomis kitomis DB lentelėmis (jai suteikiamas sistemos rangas yra 5), todėl per DB testavimo įskiepį galima įvesti tokius testinius duomenis:

- OperationType lentelei – trys testiniai įrašai;
- Person – trys testiniai įrašai;
- Product – du testiniai įrašai;
- Manager – vienas įrašas;
- Plan – vienas įrašas;
- Employee – vienas įrašas;
- Production – 0 įrašų (ji yra priklausoma nuo Task, ir šiame pavyzdyje nėra testuojama);
- Task – 15 įrašų (kiekvienam iš testavimo atvejų padengti, jie yra pateikti 4.14 paveiksle).

Įvykdžius testą su šiais testiniais duomenimis, galime įsitikinti, kad jis buvo sėkmingas – visos testinės eilutės buvo bandomos įterpti į DB, ir, atitinkamai nuo laukto rezultato visi testai pavyko. Testo rezultatų lange (4.15 pav.) aiškiai matomas vienas iš <<CHK>> chk_performer_inspector testų, kuomet buvo laukiama klaida, ir duomenų įterpimo metu ji buvo gauta.



4.15 pav. Task lentelės testo rezultatų langas

4.3. Sukurtos sistemos testavimas

Sistema yra pilnai ištestuota ir veikianti. Bendras testavimas laikomas sėkmingu. Pateikiame trijų pagrindinių panaudojimo atvejų testavimo rezultatus:

- Panaudojimo atvejo „Peržiūrėti suprojektuotą modelį“ testavimas;

Testavimas	**Testas sėkmingas**	**Testas nesėkmingas**
Ar nuskaitomas Visio formoje suformuotas modelis?	X	
Ar pakoregavos modelį pakeitimai matomi sistemos lange?	X	
Ar be sutrikimų vysksta peržiūros reultatų naršymas?	X	

- Panaudojimo atvejo „Tikrinti DB schemos teisingumą“ testavimas;

Testavimas	**Testas sėkmingas**	**Testas nesėkmingas**
Ar sėkmingai pasirenkama DB schema?	X	
Ar sistema nuskaito DB schemos duomenis (objektus, atsributs, apribojimus)?	X	
Ar sėkmingai vykdomas lentelės teisingumo tikrinimas?	X	
Ar sėkmingai testuojami trigeriai, aprobojimai, atributai?	X	
Ar sistema sėkmingai pateikia testavimo rezultatus?	X	
Ar sistema reaguoja į teatavimo baigtumą (ar visos lentelės patikrintos)?	X	

- Panaudojimo atvejo „Peržiūrėti schemos testo rezultatus“ testavimas.

Testavimas	**Testas sėkmingas**	**Testas nesėkmingas**
Ar sistema užfiksuoja testavimo pabaigą?	X	
Ar sistema sėkmingai suformuoja testavimo rezultatus?	X	
Ar sistema sėkmingai įšveda rezultatus į langą?	X	
Ar sistema sėkmingai ledžia vartotojui pasirinkti rezultatų ekportavimo variantus?	X	

Ar sistema sėkmingai ekportuoja rezultatus visais formatais (TXT, RTF, XML)?	X	

4.4. Sukurtos sistemos kokybės tyrimas

Analizės metu ištyrėme kelias panašias metodikas bei įrankius duomenų bazėms analizuoti bei testuoti. Suformuodami pagrindinius tokių sistemų kokybės rodiklius, kurie reikalingi DB testavimo sistemai, galime įvertinti mūsų suskurtos sistemos kokybę.

4.2 lentelė. DB testavimo sistemų palyginimas

Rodiklis	**Ideali DB testavimo sistema**	**Agenda**	**„DBTestAddin“**
Patogi vartotojo sąsaja	x		x
Papildomos kompetencijos nereikalaujantis naudojimas	x		x
Sąveika su CASE įrankiu (Microsoft Office Visio 2003/2002)	x		x
„Savos“ DB turėjimas	x	x	
Patogus ir išsamus rezultatų atvaizdavimas	x	x	x
Lengvas sistemos įdiegimas	x		x
Testavimo atvejai visiems galimiems apribojimams	x		
Saveika su įvairiais CASE įrankiais	x		
Didesnė darbo sus sistema dalis automatizuota	x		x
Nuoseklus vykdymo algoritmas	x		x
DB testavimas vykdomas ankstyvoje fazėje	x		x

Iš lentelės matome, kad mūsų sistema užildo didesnę dalį kokybės rodiklių.

Kadangi jau projekto pradžioje buvo minėta, kad sukurta DB schemos testavimo metodika leis patobulinti ir paspartinti duomenų bazės projektavimo procesus, norėdami tuo įsitikinti, eksperimentinį tyrimą vertinsime pagal laiką, kuris reikalingas testams atlikti be duomenų schemos tikrinimo įskiepio ir su juo.

Buvo atlikti tokie testai:

- Fizinės DB patikra 1: vartotojui buvo pateikta Visio DB schema bei identiška duomenų bazė. Vartotojui nežinant, iš kažkurios DB lentelės buvo atsitiktinai pašalintas vienas atributas, ir paprašyta aptikti, kuri lentelė nebeatitinka Visio schemos;
- Fizinės DB patikra 2: vartotojui pateikta Visio DB schema bei identiška duomenų bazė. Vartotojui nežinant, iš DB buvo pašalintas vienas išorinis raktas, ir paprašyta surasti, kuri lentelė nebeatitinka Visio schemos;
- DB testavimas 1: testuojami vienos lentelės atributai bei išoriniai raktai (lentelė Manager);
- DB testavimas 2: atliekamas pilnas schemos testavimas.

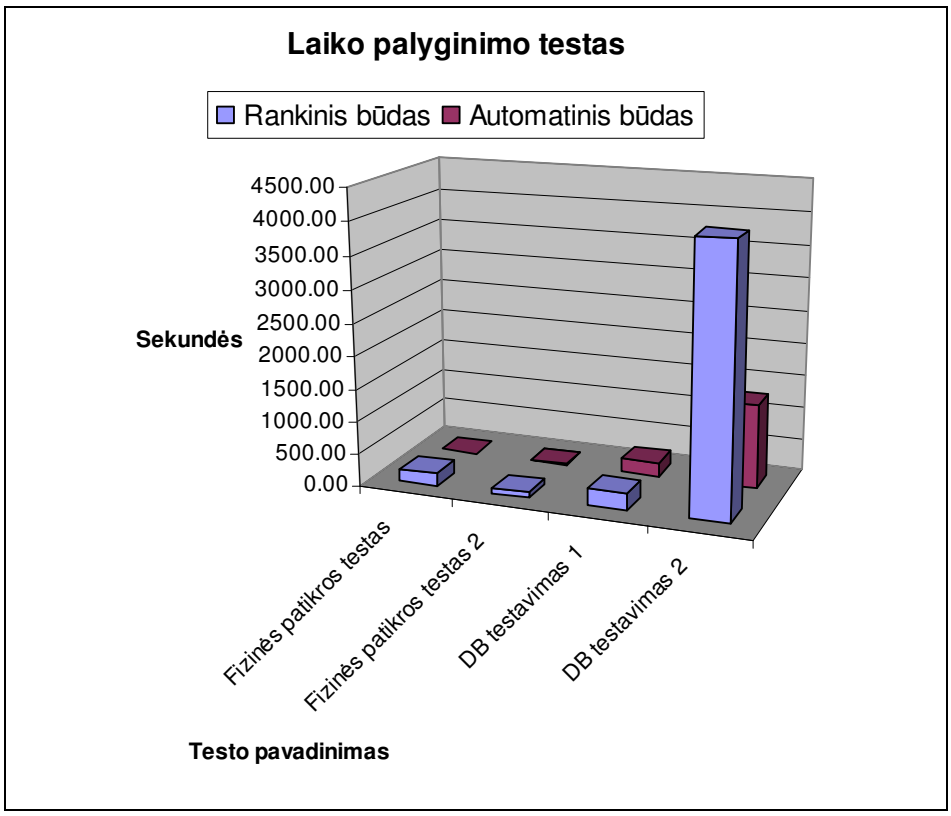
Kiekvieną šių testų pirmu bandymu atlieka vartotojas be DB schemos testavimo įskiepio, antru bandymu – naudodamas DB schemos testavimo įskiepį. Buvo matuojami testų atlikimo laikai, kurie apibendrintai pateikiami lentelėje žemiau.

4.3 lentelė. DB testų rezultatai

	Rankinis būdas, hh:mm:ss	**Automatinis būdas, hh:mm:ss**
Fizinės patikros testas	00:03:08	00:00:10
Fizinės patikros testas 2	00:01:23	00:00:08
DB testavimas 1	00:04:17	00:03:33
DB testavimas 2	01:07:35	00:21:30

Gauti testų rezultatai grafiniame pavidale pateikiami 4.16 paveiksle. Iš šių rezultatų matome, kad fizinės DB patikros atlikimas automatiniu būdu yra ženkliai greitesnis nei rankinis tokio testo imitavimas (atitinkamai 1780% ir 938% greičiau).

DB testavimo atlikimas per Visio įskiepį taip pat yra greitesnis (21% ir 214%). Čia reikėtų atkreipti dėmesį į tai, kad kuo sudėtingesnė DB schema, tuo vartotojui sunkiau tokius testus atlikti rankiniu būdu, kadangi vis didesnė žmogiškųjų klaidų tikimybė, reikia analizuoti didelės apimties duomenų schemą ir pan.



4.16 pav. DB testavimo laikų grafikas

4.5. Tolimesnio sistemos tobulinimo, plėtojimo galimybės

Sistema „DBTestAddin“ yra tik pradinėje kūrimo stadijoje. Yra sukurta tik pirmoji versija. Ateityje iškilus poreikiui plėtoti sprendimą galimi šie tobulinimai:

- Automatinis testavimo duomenų generavimas;
- Sistemos integravimas į kitus CASE įrankius;
- Centralizuotos saugyklos testų duomenims, vartotojų profiliams, schemų aprašams sukūrimas;
- Tikrinimo ir testavimo variantų išplėtimas (testavimo atvejų didinimas);
- Vartotojo sąsajos tobulinimas;
- Galimybė testuoti kitas DBVS (Oracle, MySQL);
- Nuotolinio prisijungimo prie sistemos galimybė (sistemos naudojimas „on-line“).

5. Išvados

Bendros išvados:

- Atlikta rinkoje naudojamų CASE priemonių analizė parodė, jog darbo tikslas yra aktualus: šiuo metu nėra sukurta įrankių, testuojančių universalią duomenų bazės schemą;

- Naudojantis analizės modelyje sukauptomis žiniomis, sukurtas ir patobulintas informacinės sistemos reikalavimų modelis;

- Sistemos projektas įtraukia svarbiausias sudedamąsias IS projekto dalis: dalykinės srities klasių diagramą, vartotojo sąsajos modelį, fizinę duomenų struktūrą;

- Microsoft Office Visio 2003 įrankiu pavyko realizuoti duomenų modeliavimo ir schemos tikrinimo eksperimentinį pavyzdį, todėl Visio kaip realizacijos platforma buvo pasirinkta sėkmingai;

- Išbandytas DB schemos testavimo įskiepio prototipas. Testo metu nustatyta, jog kuriamas produktas nereikalaus specifinių programinės ar techninės įrangos reikalavimų, įskiepio diegimas taip pat gana paprastas;

- Sukurta testavimo metodika: laipsniškas lentelių tikrinimo algoritmas;

- Šios metodikos pagrindu sukurtas duomenų bazių testavimo įrankis „DbTestAddin“ naudojamas kaip Micrososoft Office Visio 2003 įskiepis, kuris leidžia praktiškai įgyvendinti suformuotą uždavinį;

- Sistemos kokybės vertinimo etape buvo nustatyta, kad sistema ženkliai (net keliolika kartų) paspartina fizinės DB patikros funkciją. DB testavimo funkcija, atliekama per Visio įskiepį, taip pat būna spartesnė. Kadangi DB testo atlikimo spartumas priklauso ir nuo vartotojo įgūdžių, ir nuo schemos sudėtingumo, rezultatų pagerėjimas gali svyruoti nuo 20 iki 200%.

- Sistemos tolimesnio plėtojimo etape didžiausią dėmesį reikėtų skirti automatiniam testinių duomenų generavimui. Taip pat išanalizuotos kitos plėtojimo galimybės.

6. Literatūra

[1] Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. Addison Wesley 1999, 391 p.

[2] Unified Modeling Language: Testing Profile, v2.0. 2003, 89 p.

[3] Chays D., Deng Y., Frankl P. An agenda for testing relational database applications. 2004, 28 p.

[4] Myers G. The Art of Software Testing, Second Edition. Wiley 2004, 255 p.

[5] Tian J. Software Quality Engineering. Testing, Quality Assurance, and Quantifiable Improvement. Wiley 2005, 441 p.

[6] Eidukevičius E. Vientisumo apribojimų įgyvendinimo duomenų bazėse metodika. Magistro darbas. Kaunas, 2006.

[7] Dinh-Trong T., Kawane N., Ghosh S., France R. A Tool-Supported Approach to Testing UML Design Models. IEEE 2005, 10 p.

[8] AgileData.org: Agile Database Tools, Sandboxes and Scripts [žiūrėta 2006-01-22]. Prieiga per internetą: <http://www.agiledata.org/essays/tools.html>

[9] Microsoft Office Online: Visio Home Page [žiūrėta 2005-11-27]. Prieiga per internetą: <http://office.microsoft.com/en-us/FX010857981033.aspx>

[10] Microsoft Office Visio 2003 SDK Documentation. Microsoft Corporation, 2003.

[11] MagicDraw Inc. Open API User‘s guide. 2005, 78 p.

[12] The Memory Management Reference: Beginner’s Guide [žiūrėta 2006-01-22]. Prieiga per internetą: <http://www.memorymanagement.org/articles/begin.html>

[13] Extreme Programming: A gentle introduction [žiūrėta 2006-01-15]. Prieiga per internetą: <http://www.extremeprogramming.org>

[14] Agile Software Development [žiūrėta 2006-01-22]. Prieiga per internetą: <http://en.wikipedia.org/wiki/Agile_software_development>

7. Terminų ir santrumpų žodynas

7.1 lentelė. Terminai ir santrumpos

Terminas	Apibūdinimas
CASE	Angl. *Computer Aided Software Engineering* – kompiuterizuota informacinių sistemų inžinerija
DB	Duomenų bazė
DBVS	Duomenų bazių valdymo sistema
DDL	Angl. *Data Definition Language* – *SQL* kalbos aibė, skirta duomenų bazių objektų sukūrimo ir modifikavimo operacijoms atlikti, pvz. CREATE TABLE, DROP VIEW, ALTER COLUMN ir t.t.
DML	Angl. *Data Manipulation Language* – *SQL* kalbos aibė, skirta informacijos, esančios duomenų bazėje, skaitymo ir manipuliavimo operacijoms atlikti, pvz. SELECT, INSERT, UPDATE, DELETE.
IS	Informacinė sistema
RDBVS	Reliacinė *DBVS*
SQL	Angl. *Structured Query Language* – standartizuota kalba, skirta manipuliavimui duomenimis ir duomenų bazės objektais
XML	Angl. *eXtended Markup Language* – atviro standarto apsikeitimo duomenimis formatas. Plačiai naudojamas, kadangi yra lengvai suprantamas ir žmogaus, ir kompiuterinių sistemų, veikiančių nepriklausomose viena nuo kitos platformos.

8. Santrauka anglų kalba

Data Modeling and Schema Testing Methodology

We present methodology, which allows automatically test database schemes. Such methodology allows you to create new one or expand existing CASE tool. You can integrate database tests in generating physical database model process. In this way you can make scheme validation much easier, as a result it lowers the possibility of errors in early steps of modelling.

Acording to our analysis, there are very few tools that covers such methodotlogy. So during our work we have designed and created a new tool – „DbTestAddin“, tool that tests database schemes. It works as Microsoft Office Visio 2003 Add-in. With this tool you can easily test any database scheme you want, compare it with a real database etc.

In future, there are several updates and functional enhacements which can make this tool more attractive to use.

9. Priedai

1. Straipsnio kopija
2. Vartotojo vadovas

1 priedas. Straipsnio kopija

Duomenų modeliavimo ir schemos tikrinimo metodika

Vytautas Paulauskas, Paulius Vaidotas

Kauno Technologijos Universitetas, Informatikos fakultetas, Informacinių sistemų katedra

1. Įžanga

Informacinės sistemos projektuojamos, kuriamos ir sėkmingai diegiamos jau daugelį metų, pradedant 1970-ųjų pradžia. Per šį laikotarpį IS kūrimo procesas evoliucionavo: laikantis kad ir 10 metų senumo metodikų ir naudojantis to laikmečio įrankiais, būtų sunku sėkmingai sukurti informacinę sistemą, galinčią konkuruoti su šiuolaikiškais kitų gamintojų produktais.

Vienas iš pagrindinių veiksnių, dėl kurių informacinių sistemų kūrimo procesas greitai tobulėja, yra pastovus specifikacijos, projektavimo, realizavimo ir apskritai darbo komandoje proceso tobulinimas. Tarp šių naujovių galima paminėti tokias sąvokas, kaip objektiškai orientuotas projektavimas, AGILE projekto valdymo metodai, CASE priemonių naudojimas, ketvirtos kartos programavimo kalbos.

CASE (angl. *Computer Aided Software Engineering*) – tai kompiuterizuota informacinių sistemų inžinerija, kuomet žinios apie norimą sukurti produktą kaupiamos grafiniais vaizdais, įvairiomis diagramomis, o specializuoti CASE įrankiai šias schemas gali paversti pradiniu projekto išeities kodu ar duomenų bazės lentelių struktūros skriptu. Tokie įrankiai palengvina informacinių sistemų kūrimą, jei yra tinkamai naudojami sistemos reikalavimų analizės, projektavimo ar realizavimo etapų metu. Automatizuodami dalį operacijų, kaip kad vartotojo poreikių specifikavimą, DB modeliavimą, CASE įrankiai prisideda prie sėkmingo ir spartaus informacinės sistemos sukūrimo.

Norime pristatyti metodiką, leisiančią automatizuoti duomenų bazės schemos tikrinimą. Tokia metodika leistų sukurti arba praplėsti esamą CASE įrankį – DB schemos testus galima būtų integruoti į fizinio DB modelio generavimo procesą, taip palengvinant schemos validavimą ir sumažinant klaidų tikimybę dar ankstyvo projektavimo etapo metu.

Duomenų schemos tikrinimo metodiką tikslinga integruoti jau į egzistuojantį CASE įrankį, o ne kurti naują – informacinės sistemos kūrimo metu naudojant kelis panašų funkcionalumą įgyvendinančius CASE įrankius, žmonės priversti dubliuoti informaciją abiejose sistemose, taip pat rankiniu būdu tikrinti gautus rezultatus su siekiamais.

2. Problema

Esmė:

- Dabartiniuose CASE įrankiuose galima generuoti duomenų bazių schemas, bet nėra esamos ar naujai kuriamos duomenų bazės schemos tikrinimo funkcijų. CASE įrankių apžvalga ir jų galimybių analizė pateikta žemiau šiame skyriuje;
- Norint įsitikinti, kad sukurta duomenų bazės schema leidžia saugoti korektiškus duomenis apie dalykinę sritį, ją reikia ištestuoti su tikrais arba sugeneruotais dalykinės srities duomenų rinkiniais, apimančiais galimas duomenų variantų aibes;
- Projektuotojai ir testuotojai, norėdami įsitikrinti duomenų bazės validumu, turi naudotis keliais CASE įrankiais vienu metu, arba šalia CASE įrankio naudoti savo parašytą programinį sprendimą.

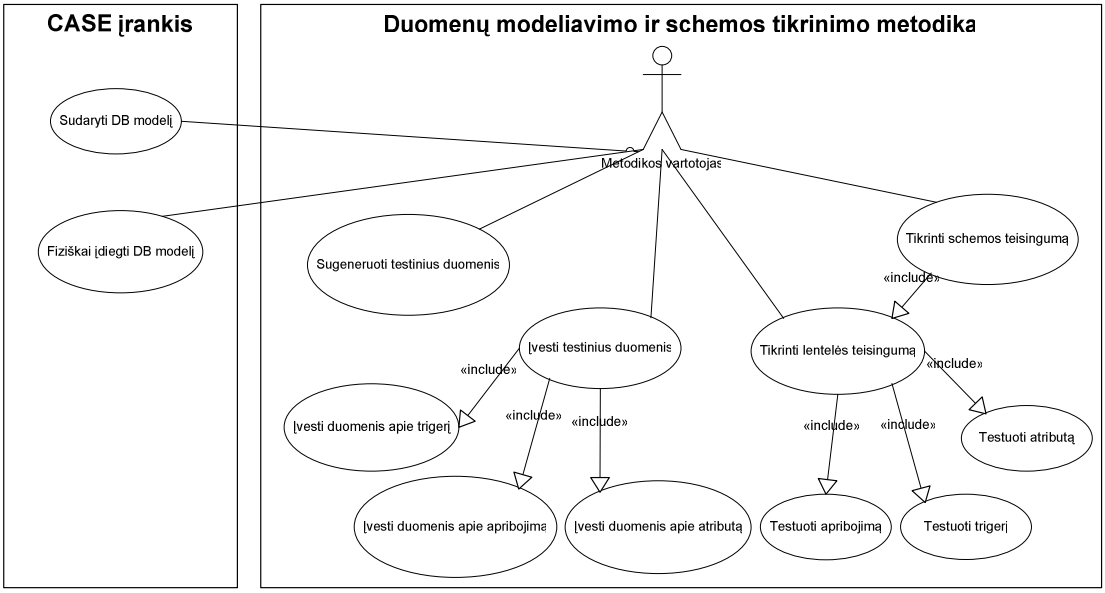
Praktikoje ne visada lengva lygiagrečiai naudoti kelis projektavimo automatizavimo įrankius, nors šių įrankių taikymo sritys būna glaudžiai susijusios, konkrečiai – darbas su reliacinėmis DB. Dauguma rinkoje esančių CASE įrankių apriboti vien SQL kalbos skriptų generavimu ir atvirkštine inžinerija, DB schemų perteikimu vaizdinėmis diagramomis, arba duomenų generavimu apkrovos testams (angl. *Stress tests*). CASE įrankių įvairovė ir jų realizuojamos funkcijos apibendrinamos. Tik dalis šių įrankių turi įgyvendintas kelias galimybes vienu metu, pvz. SQL skripto generavimo, DB schemos vizualizavimo ir dokumentacijos sudarymo galimybes.

Remiantis minėtu pavyzdžiu, vartotojas, norėdamas sukurti duomenų bazės schemą (DDL), ją įdiegti realioje duomenų bazių valdymo sistemoje (DBVS), ir ištestuoti, ar tikrai sukurta schema atitinka visus apribojimus, kurie buvo numatyti schemos projektavimo metu, yra priverstas naudotis keliais CASE įrankiais, arba CASE įrankiu ir savo sukurtu programiniu sprendimu. Tai sukelia tam tikras problemas: vartotojas turi arba rankiniu būdu tikrinti informaciją, gautą testavimo metu, su DB schemos standartu, arba pakartotinai į testavimo įrankį suvesti DB schemą, kuri buvo naudojama CASE priemonėje, sugeneravusioje fizinę DB struktūrą.

3. Panaudojimo atvejų modelis

Norime įvardinti funkcijas, kurias reikia numatyti metodikoje, bei kurias turėtų įgyvendinti pasirinkto CASE įrankio įskiepis. Panaudojimo atvejų diagrama pateikta 2.3 pav. Posistemyje „CASE įrankis“ pavaizduoti panaudojimo atvejai, kurie metodikoje nebus realizuoti, kadangi jie dažniausiai yra įgyvendinti pačiame CASE pakete. Duomenų modeliavimo ir schemos tikrinimo metodika apima testinių duomenų generavimą (kurie reikalingi norimam testui atlikti), taip pat – rankinį duomenų generavimą, bei schemos arba atskirų schemos elementų tikrinimą.

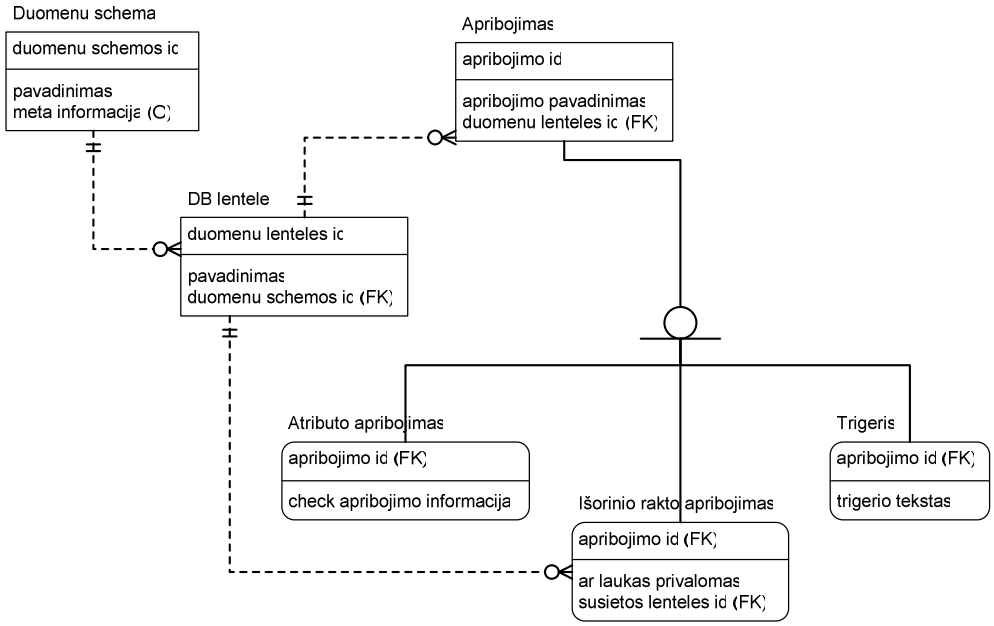
Pradiniame etape numatoma, jog bus tikrinami CHECK, FOREIGN KEY, NOT NULL apribojimai, bei DB trigerių veikimas. Panaudojimo atvejai „Įvesti testinius duomenis“ bei „Tikrinti lentelės teisingumą“ patikslinti konkrečiais veiksmais, tuo pabrėžiant, jog sistema atliks testus, norėdama patikrinti CHECK, NOT NULL apribojimus, FOREIGN KEY integralumo užtikrinimo taisykles ir trigerių veikimą.



1 pav. Dalykinės srities panaudojimo atvejai

4. Dalykinės srities objektų modelis

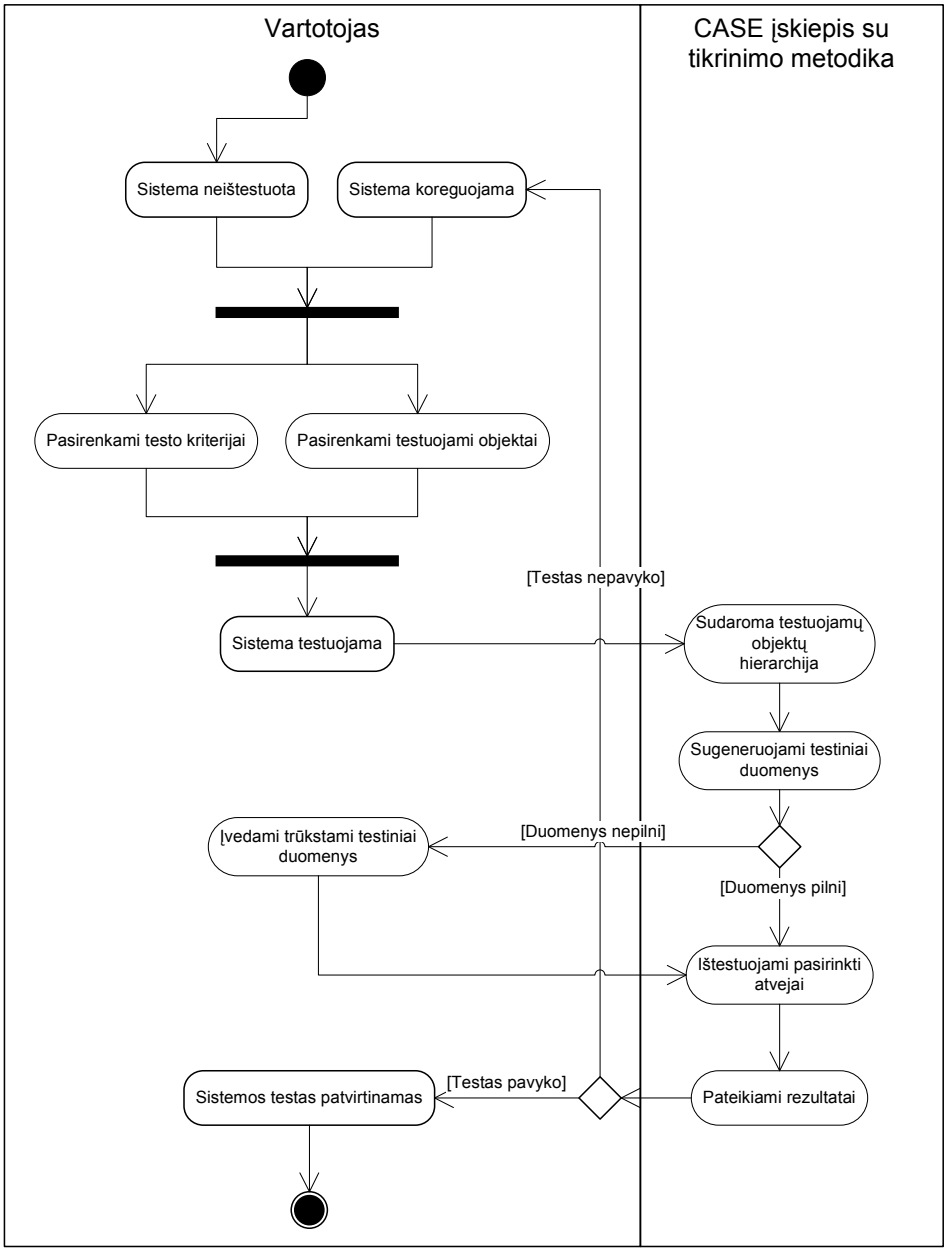
Dalykinės srities objektai sudaryti remiantis aukščiau pateiktu panaudojimo atvejų modeliu. Dalykinės srities objektai pateikti ER (angl. *Entity Relational*) modeliu 2.4 paveiksle.



2 pav. Dalykinės srities objektų ER modelis

5. Dalykinės srities procesų modelis

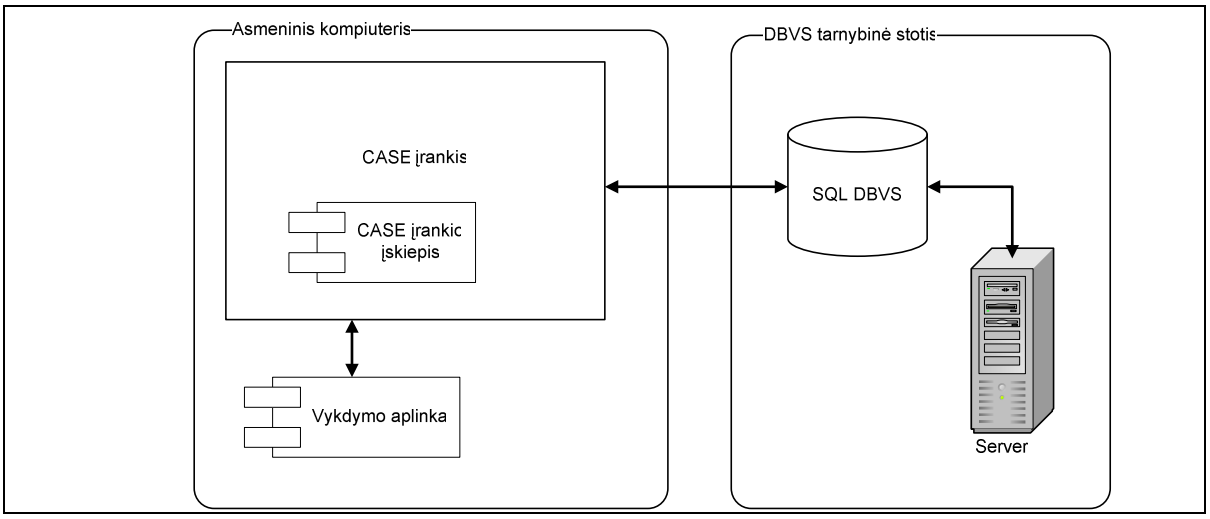
Veiklos procesus galime įsivaizduoti kaip veiklos diagramą, kurioje dalyvautų vartotojas ir CASE sistemos įskiepis su įdiegta duomenų schemos tikrinimo metodika. Vartotojas šiame įskiepyje pasirinktų, kokias duomenų schemos lenteles nori tikrinti, ir pradėtų testą. Tuomet CASE įrankis pagal metodiką atrinktų, nuo kurios iš pasirinktų lentelių pradėti testavimą, kokius duomenis bandyti rašyti, ir kaip pateikti rezultatus vartotojui (3 pav). Visus duomenis, reikalingus testui, CASE įrankis turi bandyti sugeneruoti automatiškai. Tik trūkstant tam tikrų duomenų, vartotojo paprašoma įvesti reikiamus duomenis, po įvestų duomenų patvirtinimo sistema tęsia įprastinę testavimo procedūrą.



3 pav. Procesų modelis, pateiktas veiklos diagrama

6. Architektūra

Duomenų schemos tikrinimo metodikos pavyzdį realizavus kaip CASE sistemos įskiepį, šis komponentas betarpiškai sąveikautų su CASE sistemos programiniais interfeisais, ir jam papildomos vykdymo aplinkos nereiktų, nebent jos reikalautų pats CASE įrankis.

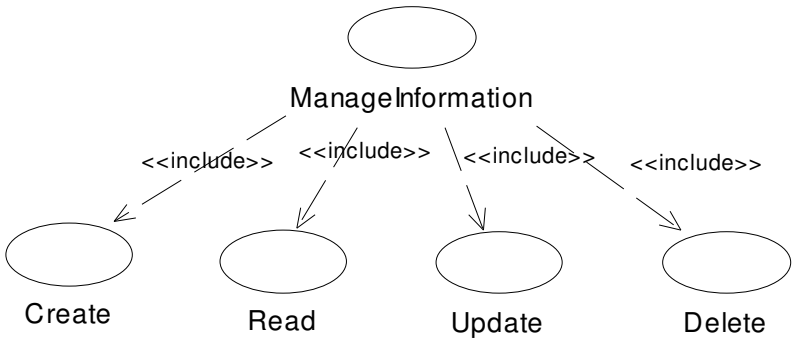


4 pav. Architektūra praplečiant CASE įrankį

Taip pat, naudojantis CASE sistemos įskiepio sprendimu, palengvėja priėjimas prie CASE įrankyje saugomų duomenų schemos meta duomenų, taip eliminuojant duomenų dubliavimą skirtingose sistemose. Įvertinus šiuos du aspektus, darosi pakankamai aišku, jog ir vartojimo, ir realizavimo prasme patogesnė realizacija kaip CASE sistemos įskiepis.

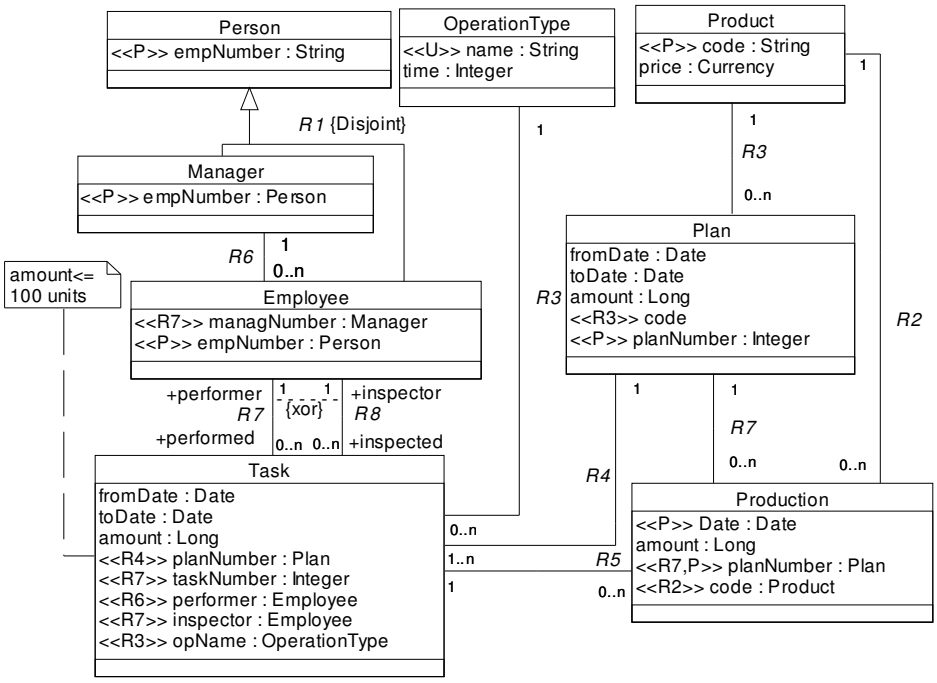
7. Testinis duomenų bazės testavimo pavyzdys

Kadangi duomenų bazės schema testuojama nepriklausomai nuo taikomųjų uždavinių, duomenų bazės panaudojimo atvejus galima apibendrinti panaudojimo atveju, kuris turės užtikrinti CRUD (angl. *Create*, *Read*, *Update*, *Delete*) funkcijas, dar vadinamas DML (angl. *Data Manipulation Language*). *Read* operacijos nebus tikrinamos (5 pav.).



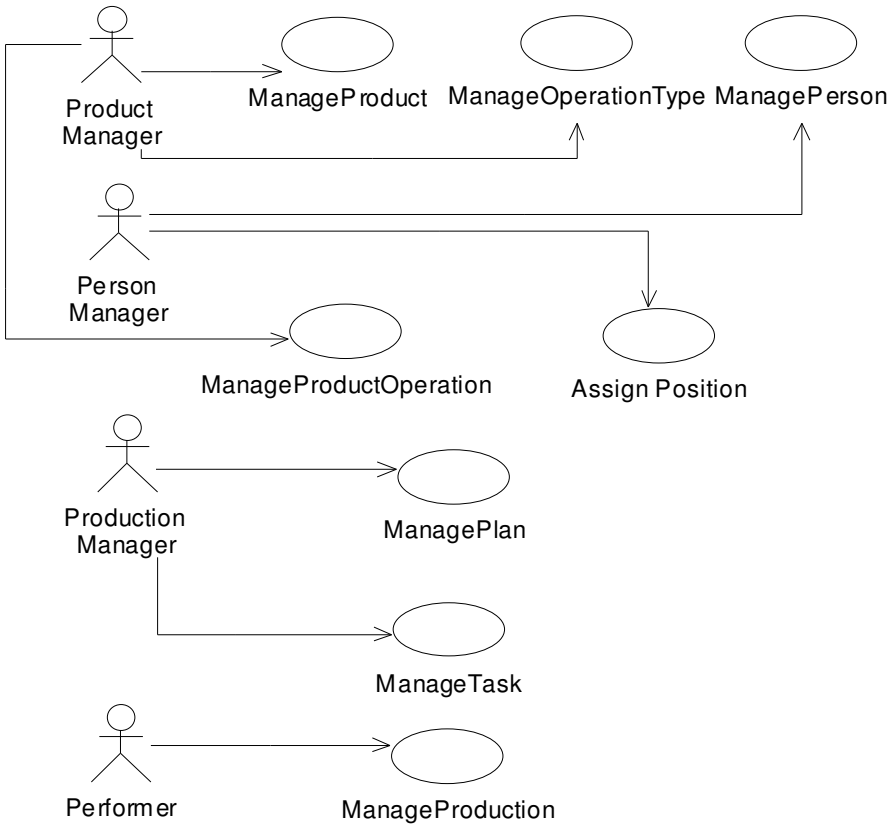
5 pav. DML operacijos, nagrinėjamos testiniame pavyzdyje

Pasirinkta duomenų bazės schema pavaizduota paveikslėlyje žemiau. Siekiama parinkti tokį testavimo pavyzdį, kuris apimtų pakankamai daug testavimo atvejų, bet sistema būtų neperpildyta. Testiniame pavyzdyje, kaip matyti iš ryšių tarp lentelių, daug išorinio rakto apribojimų, taip pat – CHECK suvaržymų (TASK: amount <= 100; TASK: fromDate <= toDate; PLAN: fromDate <= toDate).



6 pav. Testinė duomenų bazė, kuriai bus bandoma sukurti praktinę metodo realizaciją

Schemos kūrimo ir testavimo metodas remiasi laipsniška analize. Panaudojimo atvejai išdėstomi jų rezultate gaunamos informacijos gyvavimo trukmės mažėjimo kryptimi, t.y., einama nuo pastovesnės, ilgiau gyvuojančios informacijos apdorojimo panaudojimo atvejų link trumpiau gyvuojančios ir dažniau apdorojamos informacijos. Kontroliniame pavyzdyje realizuojamos DML operacijos pavaizduotos 7 paveiksle.



7 pav. Panaudojimo atvejai, realizuotini testuojamoje sistemoje

Schemos testavimo metu nuosekliai nagrinėjami panaudojimo atvejai einant nuo aukštesnio lygmens prie žemesnio. Vieno lygmens panaudojimo atvejus galima nagrinėti lygiagrečiai, kadangi jie vienas nuo kito nepriklauso. Esybių klasės sukuriamos tokiu būdu: nagrinėjami esybių tipai, kurių reikia naujai esybei sukurti.

Literatūros sąrašas:

[15]Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. Addison Wesley 1999, 391 p.

[16]Unified Modeling Language: Testing Profile, v2.0. 2003, 89 p.

[17]Chays D., Deng Y., Frankl P. An agenda for testing relational database applications. 2004, 28 p.

[18]Myers G. The Art of Software Testing, Second Edition. Wiley 2004, 255 p.

[19]Tian J. Software Quality Engineering. Testing, Quality Assurance, and Quantifiable Improvement. Wiley 2005, 441 p.

[20]Eidukevičius E. Vientisumo apribojimų įgyvendinimo duomenų bazėse metodika. Magistro darbas. Kaunas, 2006.

[21]Dinh-Trong T., Kawane N., Ghosh S., France R. A Tool-Supported Approach to Testing UML Design Models. IEEE 2005, 10 p.

[22]AgileData.org: Agile Database Tools, Sandboxes and Scripts [žiūrėta 2006-01-22]. Prieiga per internetą: <http://www.agiledata.org/essays/tools.html>

[23]Microsoft Office Online: Visio Home Page [žiūrėta 2005-11-27]. Prieiga per internetą: <http://office.microsoft.com/en-us/FX010857981033.aspx>

[24]Microsoft Office Visio 2003 SDK Documentation. Microsoft Corporation, 2003.

[25]MagicDraw Inc. Open API User‘s guide. 2005, 78 p.

[26]The Memory Management Reference: Beginner’s Guide [žiūrėta 2006-01-22]. Prieiga per internetą: <http://www.memorymanagement.org/articles/begin.html>

[27]Extreme Programming: A gentle introduction [žiūrėta 2006-01-15]. Prieiga per internetą: <http://www.extremeprogramming.org>

[28]Agile Software Development [žiūrėta 2006-01-22]. Prieiga per internetą: <http://en.wikipedia.org/wiki/Agile_software_development>

Trumpas aprašymas anglų kalba:

Methodology of testing database schemes

We present methodology, which allows automatically test database schemes. Such methodology allows you to create new one or expand existing CASE tool. You can integrate database tests in generating physical database model process. In this way you can make scheme validation much easier, as a result it lowers the possibility of errors in early steps of modelling.

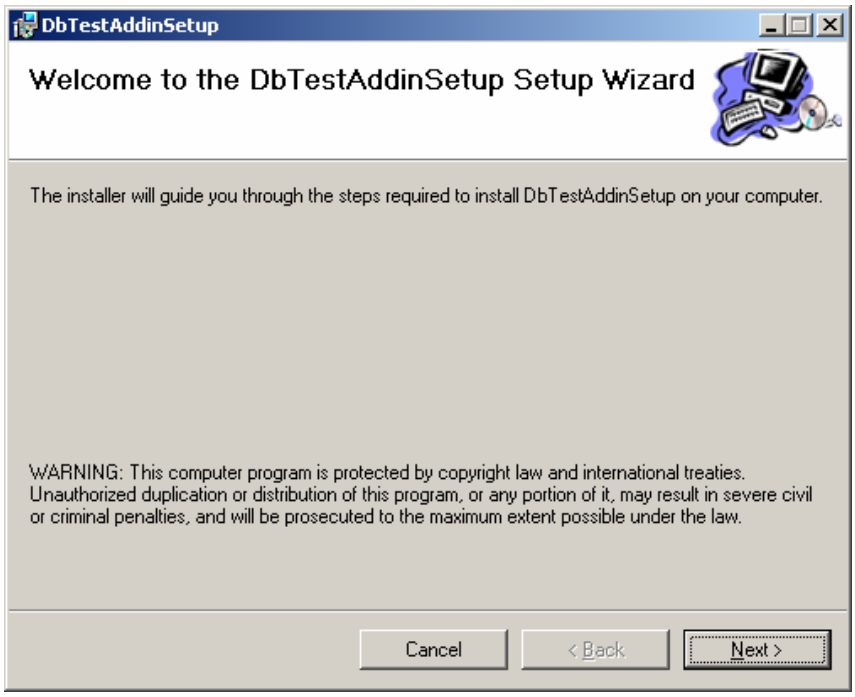
2 priedas. Vartotojo vadovas

Microsoft Office Visio 2003 įskiepio „DBTestAddin“ vartotojo vadovas

1. Įskiepio įdiegimas

Norėdami įdiegti „DbTestAddin“ įskiepį, turite alikti šiuos veiksmus:

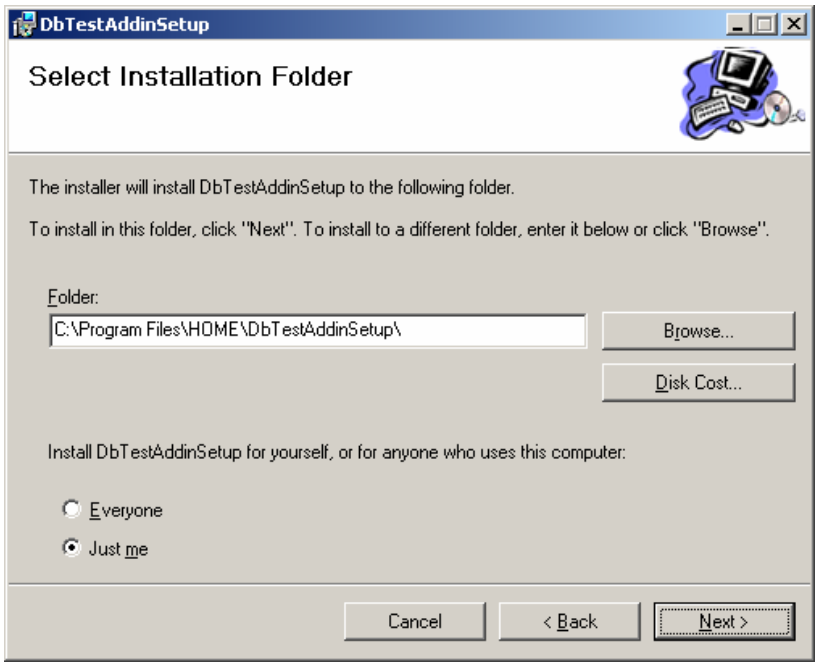
1. išarchyvuokite bylą DbTestAddinSetup.zip;
2. vietoje, kurioje išarchyvavote bylą, atsidarykite direktoriją Debug;
3. paleiskite bylą Setup.exe
4. atsidarius langui (1 pav.) spauskite „Next“;



1 pav. Įskiepio „DBTestAddin“ įdiegimo įrankio pradinis langas

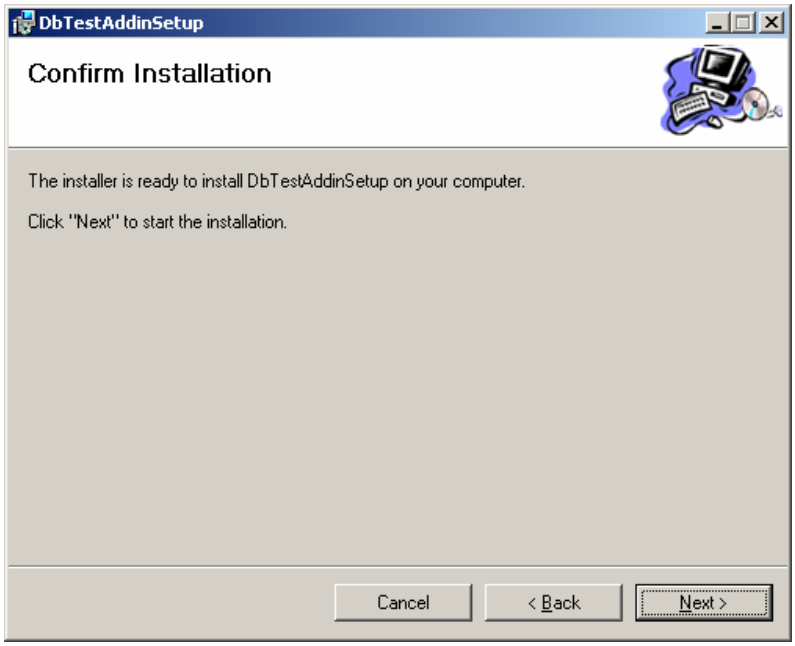
5. atsidarius langui (2 pav.) pasirinkite:

- Folder – vietą, kurioje diegsite įskiepį (jei nenorite rašyti ranka, galite ją pasirinkti paspaudę mygtuką „Browse“);
- Pasirinkite, ar įskiepiu galės naudotis visi kompiuterio vartotojai („Everyone“) ar tik jūsų vartotojas („Just me“);



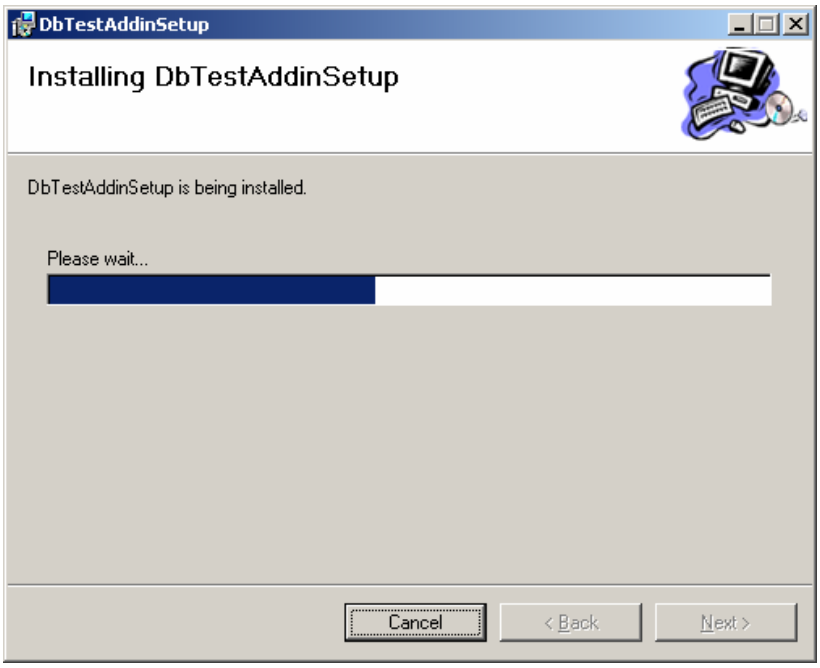
2 pav. Įskiepio „DBTestAddin“ įdiegimo įrankio antrasis nustatymų langas

6. spauskite „Next“;
7. atsidarius baigiamajam diegimo langui (3 pav.) spauskite „Next“;



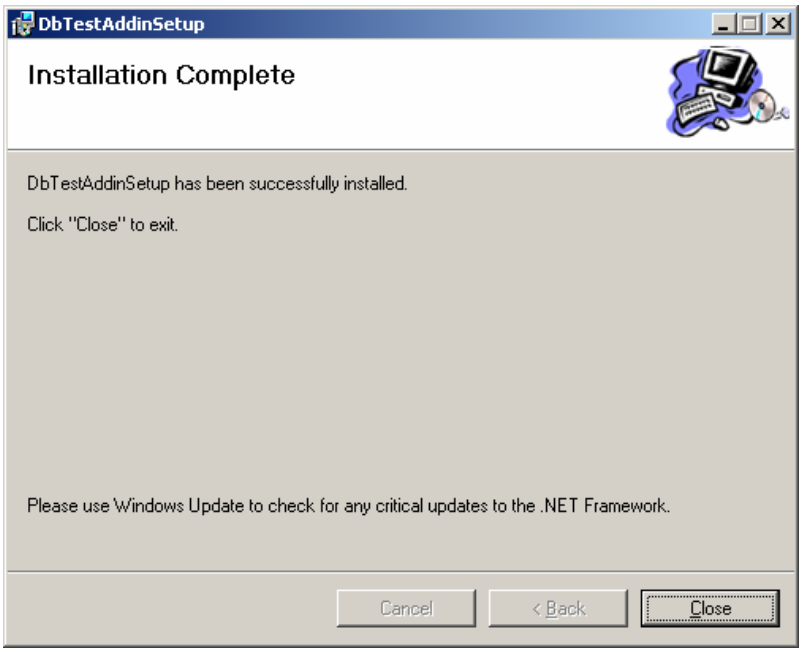
3 pav. Įskiepio „DBTestAddin“ įdiegimo bagiamasis langas

8. diegimo lange (4 pav.) slenkanti juosta reiškia, kad įskiepis diegiamas. Palaukite, kol diegimas baigsis (tai gali užtrukti);



4 pav. Įskiepio „DbTestAddin“ vykdomo diegimo langas

9. diegimui sėkmingai pasibaigus atsiranda finalinis langas (5 pav.), spauskite „Close“

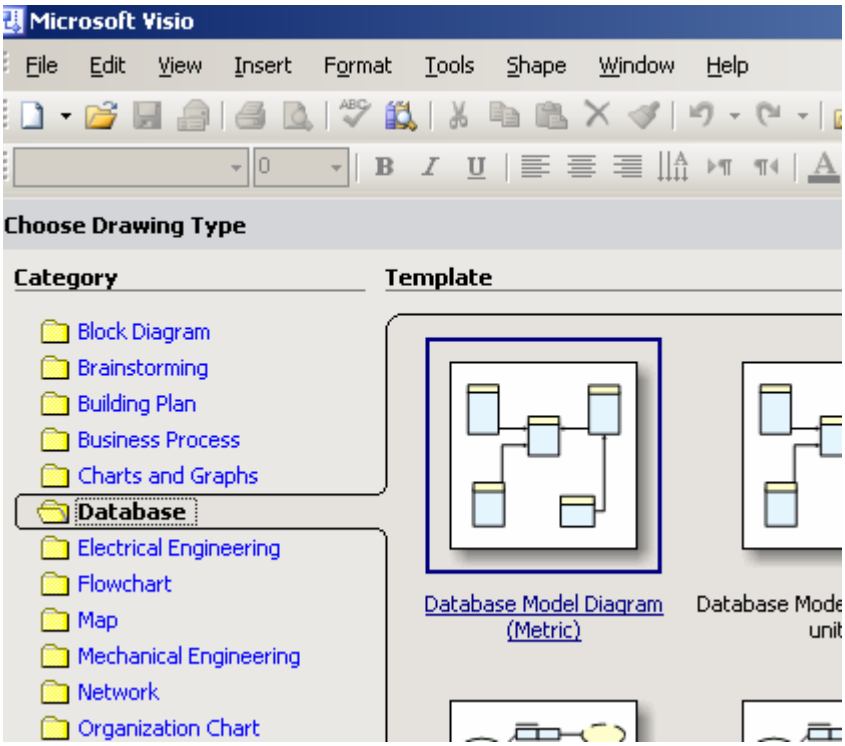


5 pav. Įskiepio „DBTestAddin“ vykdomo diegimo langas

10. Įskiepis įdiegtas

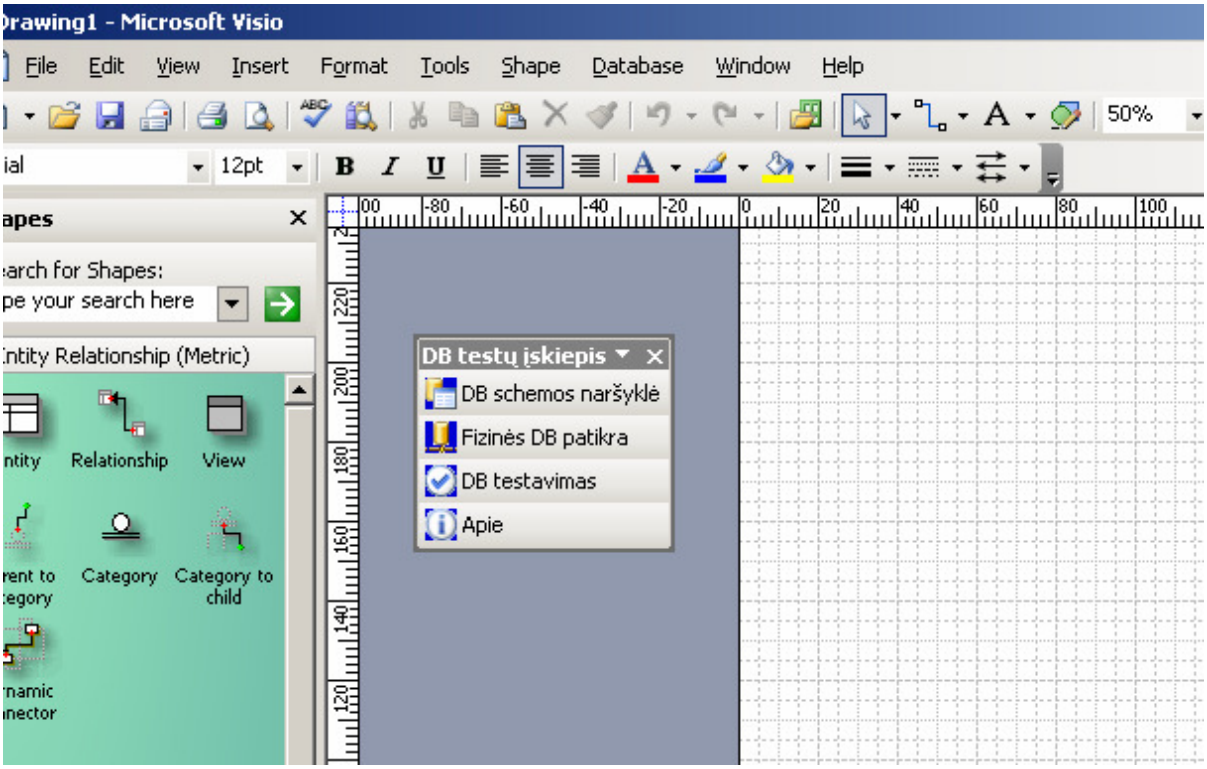
2. Įskiepio paleidimas

1. Paleiskite „Microsoft Office Visio 2003“;
2. Pasirinkite formos tipą „Database Model Diagram (Metric)“ (6 pav.);



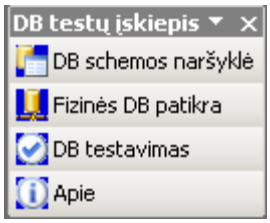
6 pav. Microsoft Office Visio 2003 formos tipo pasirinkimas

3. Atsidarius formą iškart matysite langelį „DB testų įskiepis“ (7 pav.)



7 pav. Microsoft Office Visio 2003 formoje esantis langelis „DB testų įskiepis“

3. Įskiepio naudojimas



Įskiepį sudaro keturi mygtukai:

- DB schemos naršyklė
- Fizinės DB patikra
- DB testavimas
- Apie

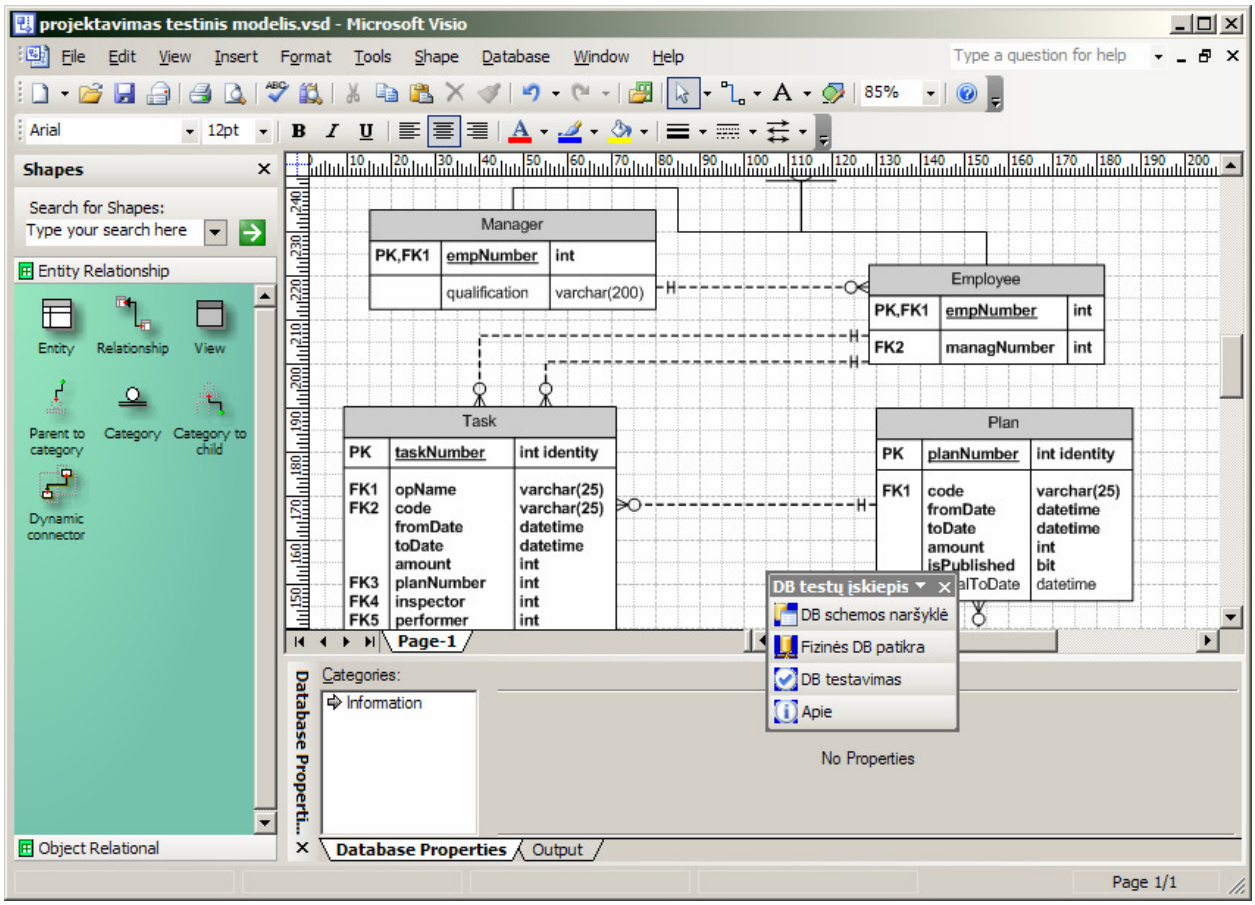
Jei norite paleisti reikalingą funkcionalumą, paspauskite atitinkamą mygtuką.

DB schemos naršyklė

DB schemos naršyklė skirta peržiūrėti suformuotai DB schemai Visio formoje.

Naudojimas:

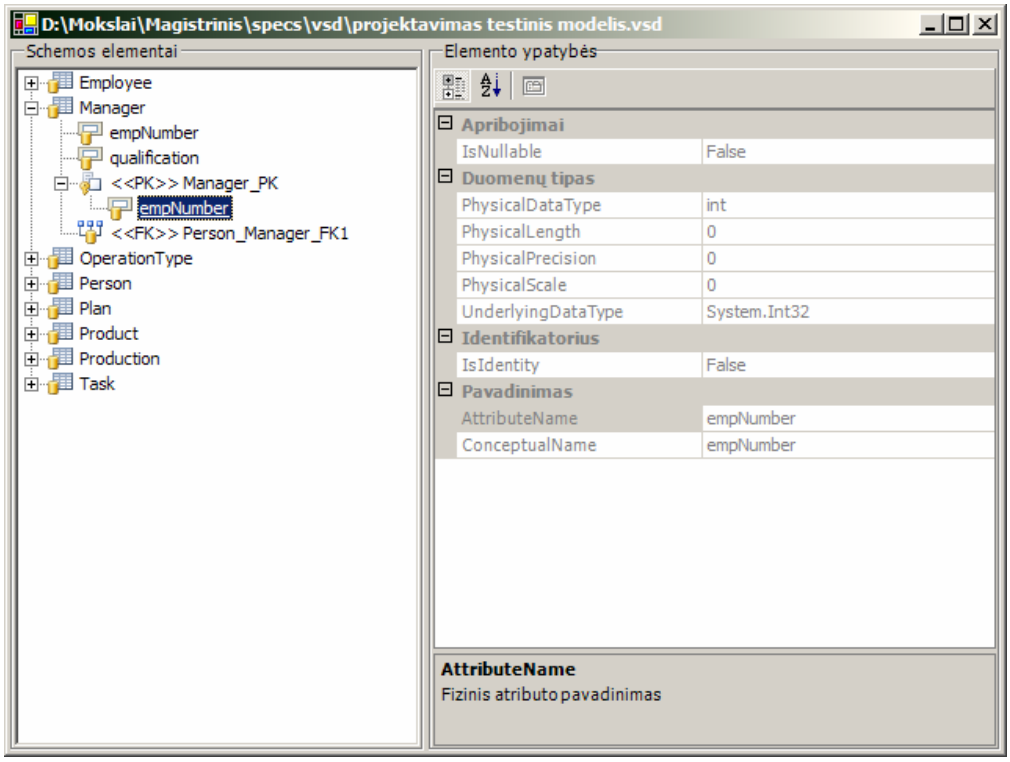
1. Suformuojate DB schemą Visio formoje (8 pav.);



8 pav. Microsoft Office Visio 2003 suformuota schema

Pastaba: Turint schemą su vienodais lentelių pavadinimais ar atributų pavadinimais vienoje lentelėje, DB schemos naršyklėje prie pasikartojančių atributų pridedami atsitiktiniai simboliai.

2. Spaudžiate mygtuką „DB schemos naršyklė“
3. Atsidariusio lango (9 pav.) kairėje pusėje matote elementus esančius schemoje pavaizduotus „medžio“ principu (angl. *treeview*);



9 pav. Suformutos schemos elementų informacija

4. Pažymėkite norimą elementą;
5. Dešinėje pusėje pasirodys informacija apie tą elementą;
6. Spustelėje ent elemento du kartus arba paspaudę „pliusiuką“ šalia elemento išskleisite jį (pamatysite elemento atributus) ;
7. Pažymėję atributą dešinėje pusėje pamatysite informaciją apie jį ;
8. Apatinėje dešinėje pusėje esančiame langelyje visada matysite lietuvišką pažymėto elemento ar atsributo ar apribojimo paaiškinimą.
9. Schemos peržiūra baigta.

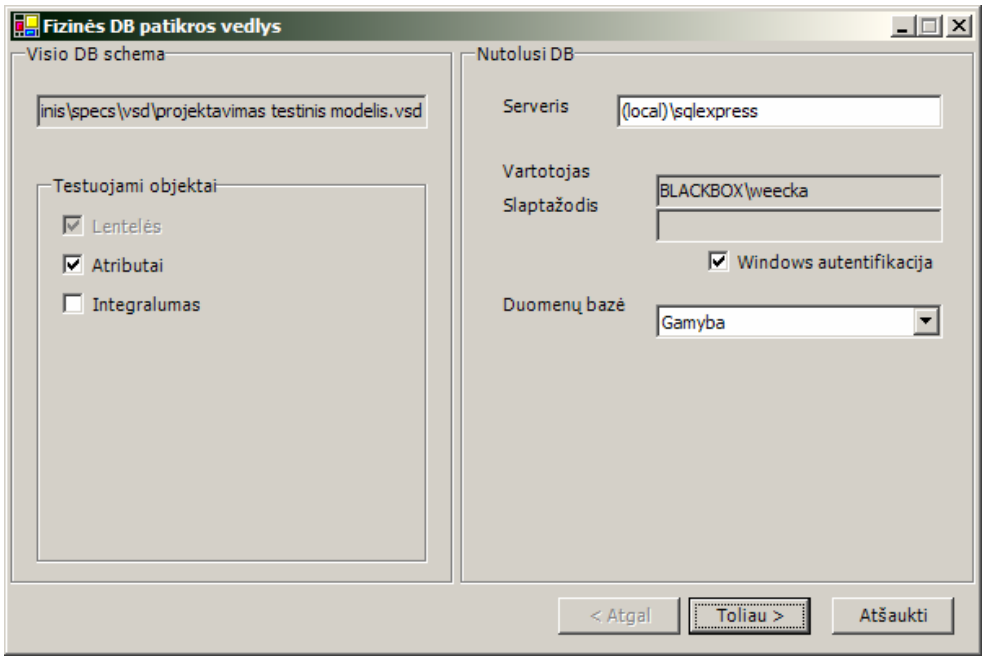
Fizinės DB patikra

Fizinės DB patikra suteikią galimybę palyginti Visio schemoje nupaišytą DB schemą su realia fiziniame serveryje esančia DB.

Naudojimas:

1. Suformuojate DB schemą Visio formoje, pagal ją suformuojate DB serveryje;
2. Spauskite mygtuką „DB fizinė patikra“;

3. Atsidariusiame lange (10 pav.) pasirinkite DB tikrinimo kriterijus, serverį ir prisijungimo prie jo nustatymus;



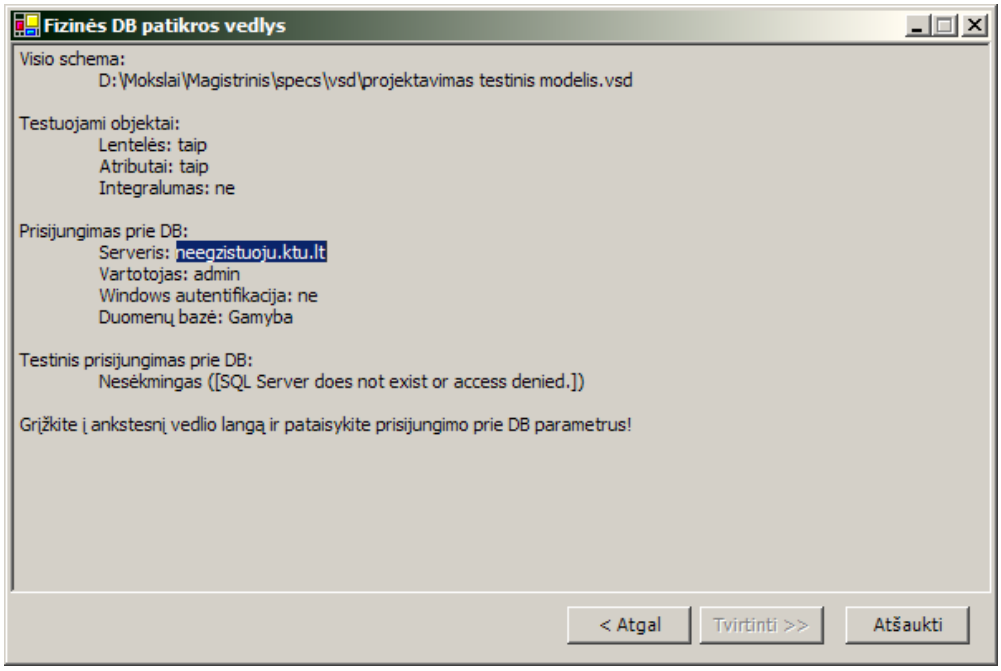
10 pav. Fizinės DB patikros nustatymai

Pastaba: testui pavykus, visi nustatymai yra įsimenami vartotojo „ApplicationData“ kataloge, ir pakraunami kito testo metu.

4. Galimi testavimo nustatymai:

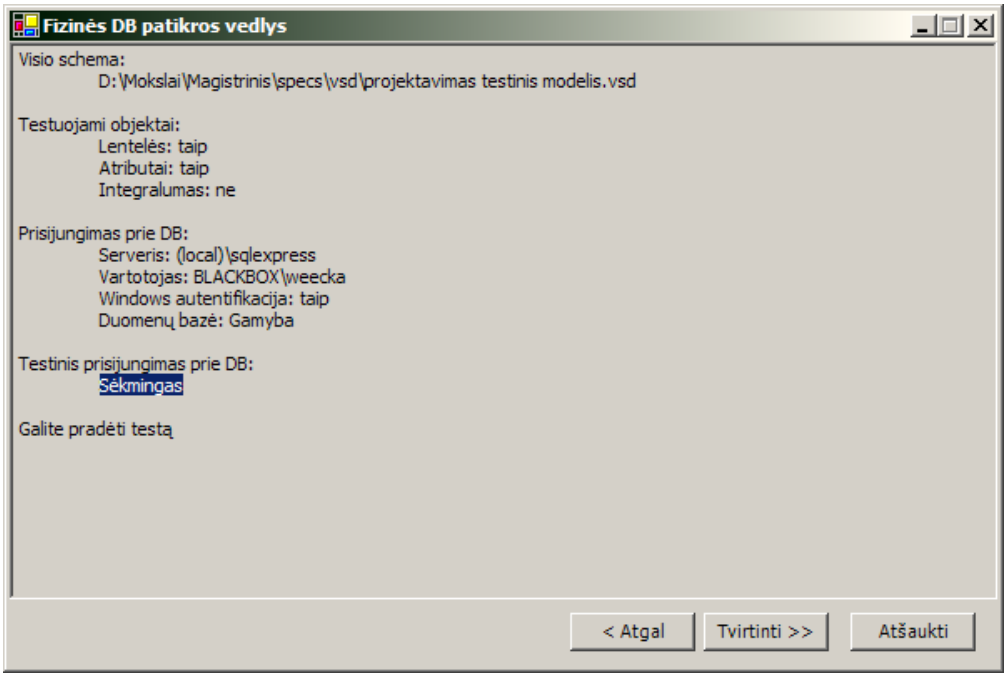
- Jeigu pirmame vedlio pasirinksite testuoti tik lenteles (nuimsite varneles „Atributai“ ir „Integralumas“, ištestuos tik patį faktą – lentelė yra ar nėra fizinėje DB).
- Uždėjus „Atributai“, testuojami visi lentelių atributai.
- Uždėjus „Integralumas“, tuo pačiu bus patikrinti ir pirminiai bei išoriniai raktai.

5. Spauskite „Toliau >“;
6. Atsidariusiame lange (11, 12 pav.) galite peržiūrėti pasirinktus kriterijus bei informaciją apie jungimąsi prie serverio;



11 pav. Informacija apie DB patikros nustatymus bei apie nesėkmingą prisijungimą prie serverio

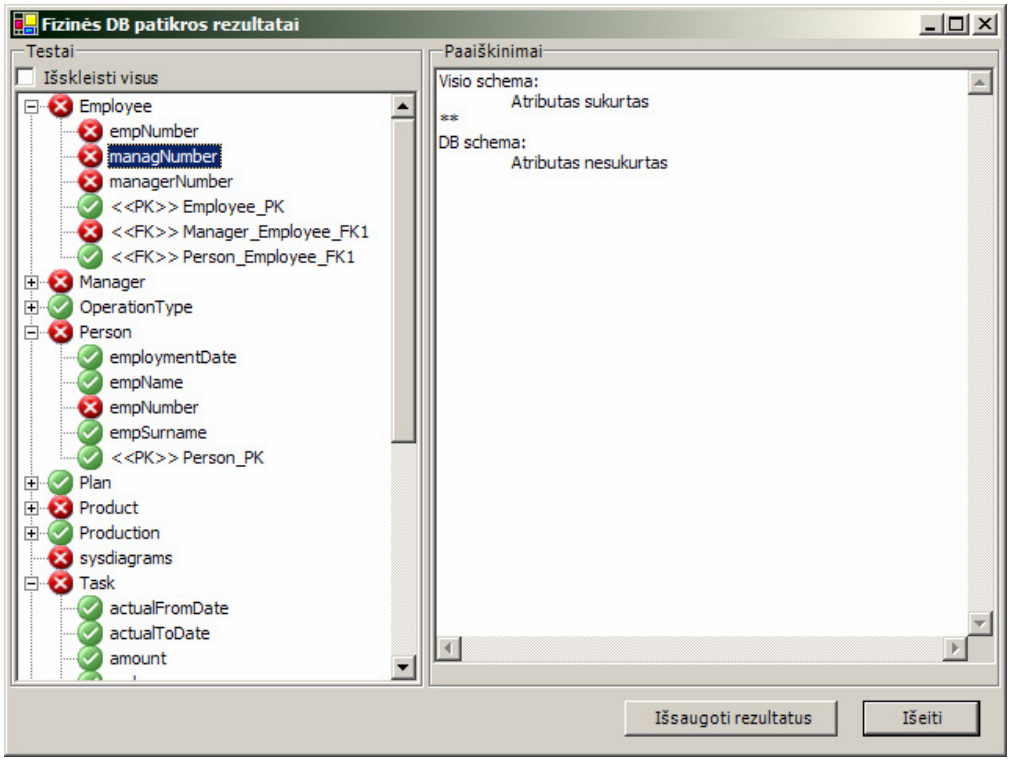
Pastaba: jei prisijungimas prie serverio bus nesėkmingas, negalėsite tęsti fizinės DB patikros (mygtukas „Toliau >“ bus neaktyvus).



12 pav. Informacija apie DB patikros nustatymus bei apie sėkmingą prisijungimą prie serverio

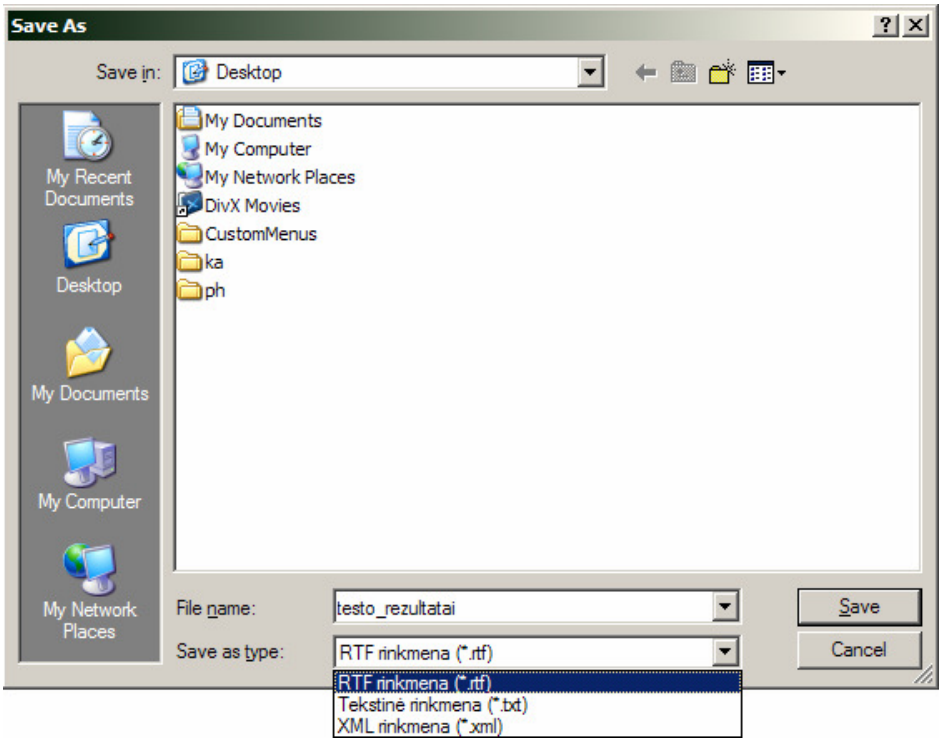
7. Spauskite „Tvirtinti >>“;
8. Atsidariusiame lange (13 pav.) galite peržiūrėti DB patikrinimo rezultatus pagal jūsų pasirinktus kriterijus. Žalia varnelė reiškia, kad tikrinimas buvo sėkmingas; raudonas kryžiukas reiškia, kad tikrinimas nesėkmingas (yra nesutapimų ar

klaidų). Pasirinkę norimą elementą dešinėje lango pusėje galite peržiūrėti informaciją apie tikrinimo rezultatus;



13 pav. Fizinės DB patikros rezultatai

9. Jei paspausite mygtiką „Išsaugoti rezultatus“, jums atsidarys dialogo langas (14 pav.), kuriame galėsite išsaugoti rezultatus į atskirą bylą trimis skirtingais fotmatais:
    - RTF - eksportuojama į tekstinį formatą su formatavimu.
    - TXT - tokia pati informacija be formatavimo.
    - XML - hierarchinė rezultatų schema, kurią, esant reikalui, galima būtų apdoroti su specifine programa.



14 pav. Rezultatų saugojimo dialogas

10. Fizinės DB patikra baigta.

11. Rezultatų, išsaugotų RTF faile, fragmentas:

OperationType
Testo rezultatas: teigiamas

description
Testo rezultatas: teigiamas

name
Testo rezultatas: teigiamas

time
Testo rezultatas: teigiamas

<<PK>> OperationType_PK
Testo rezultatas: teigiamas

Person
Testo rezultatas: neigiamas

employmentDate
Testo rezultatas: teigiamas

empName
Testo rezultatas: teigiamas

empNumber
Testo rezultatas: neigiamas

- Visio schema:
- Duomenų tipas: int

- Visio duomenų tipo ilgis: 0
- Visio duomenų tipo tikslumas: 0
- Visio duomenų tipo skalė: 0
- Visio ar leidžiama NULL: False
- Visio ar IDENTITY: True
- **
- DB schema:
- DB duomenų tipas: varchar
- DB duomenų tipo ilgis: 25
- DB duomenų tipo tikslumas: 0
- DB duomenų tipo skalė: 0
- DB ar leidžiama NULL: False
- DB ar IDENTITY: False

empSurname

Testo rezultatas: teigiamas

<<PK>> Person_PK

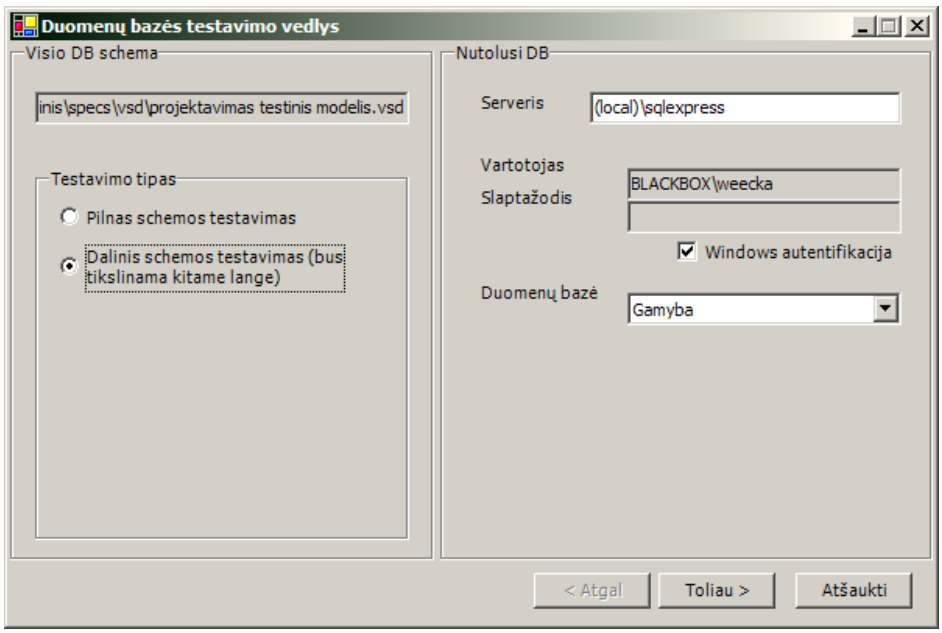
Testo rezultatas: teigiamas

DB testavimas

DB testavimas suteikia galimybę ištestuoti DB schemos teisingumą naudojant realius duomenis.

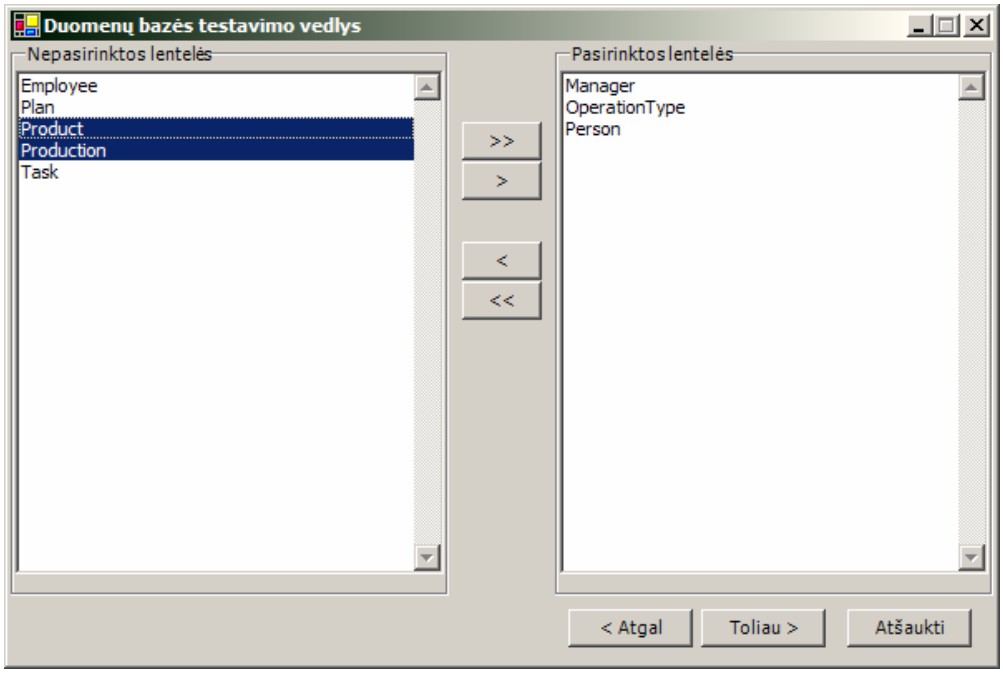
Naudojimas:

1. Suformuojate DB schemą Visio formoje;
2. Spauskite mygtuką „DB testavimas“;
3. Atsidaro DB testavimo nustatymų langas (15 pav.), kuriame galite nustatyti prisijungimą prie serverio bei testavimo kriterijus:
    - „Pilnas schemos testavimas“ – testuojamos visos lentelės schemoje,
    - „Dalinis schemos testavimas“ – galima bus pasirinkti norimas lenteles.



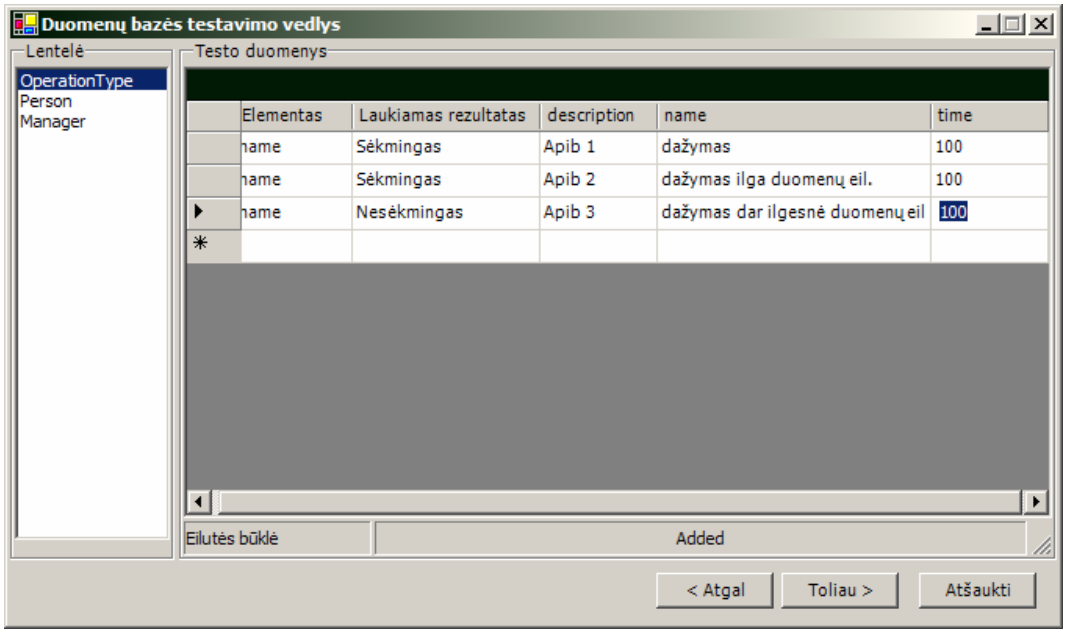
15 pav. Informacija apie DB patikros nustatymus bei apie sėkmingą prisijungimą prie serverio

4. Spauskite „Toliau“;
5. Jei pasirinksite dalinį testavimą, atsidarys langas (16 pav.) kuriame galima pasirinkti norimas testuoti lenteles (pasirinktos lentelės perkeliamos mygtukų „>“ ir išimamos „<“ pagalba, jei paspausite mygtukus „>>“ arba „<<“, perkels arba išims visas lenteles);



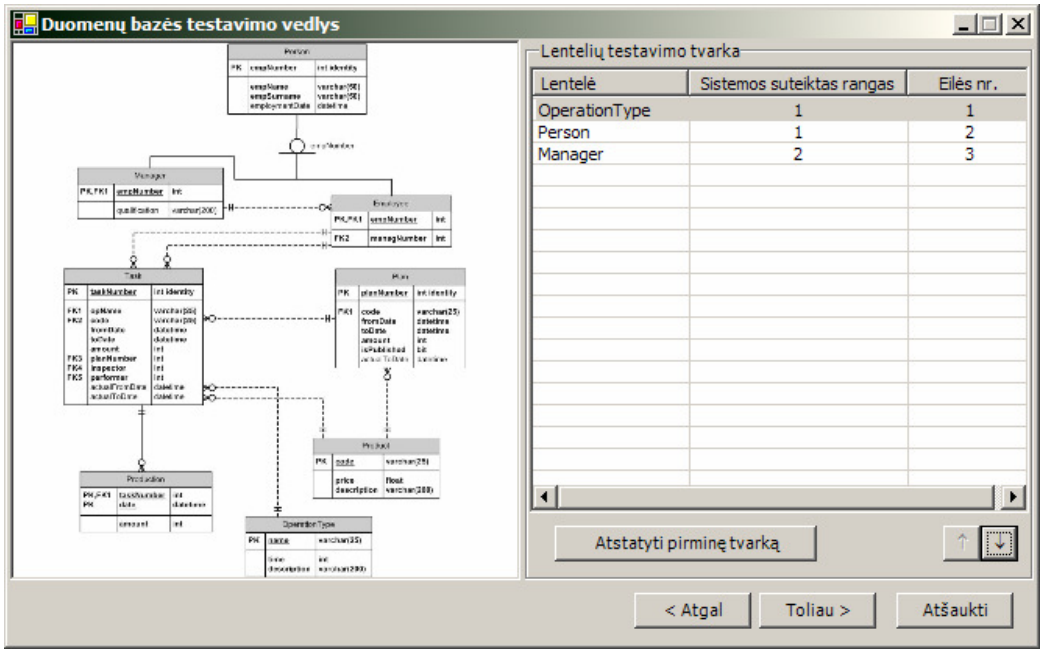
16 pav. Dalinio testavimo lentelių pasirinkimo langas

6. Spauskite „Toliau“;
7. Atsidaro langas (17 pav.), kuriame galite įvesti norimus testavimo duomenis ranka;



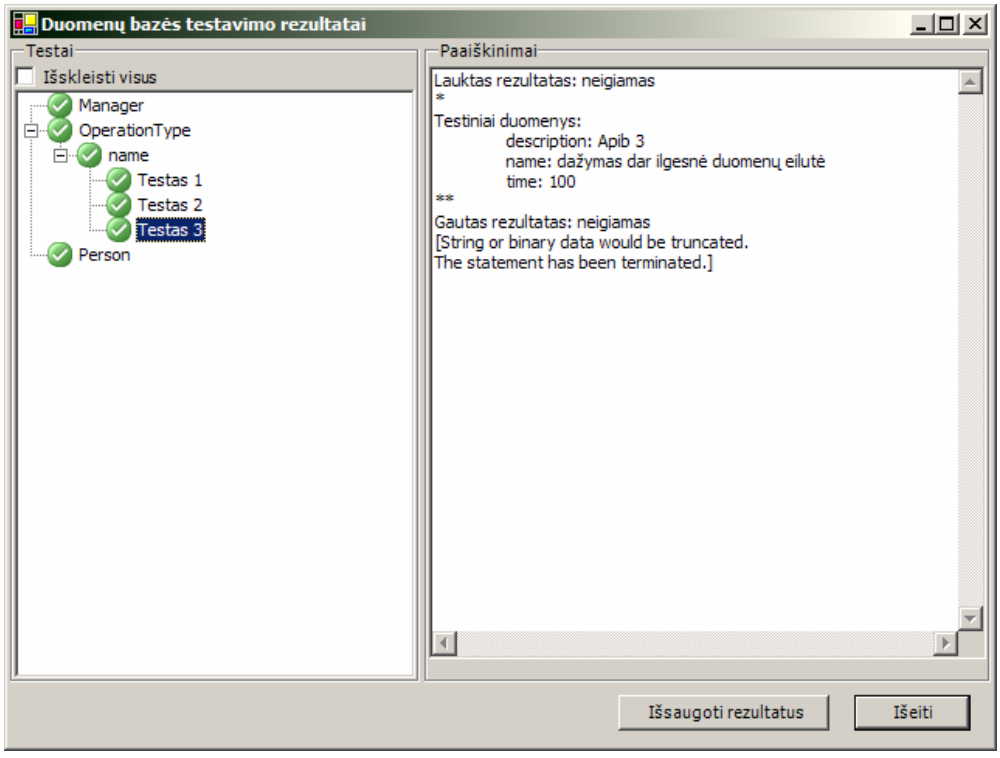
17 pav. Duomenų įvedimo langas

8. Spauskite „Toliau“;
9. Atsidaro langas (18 pav.), kuriame galima rikiuoti lenteles norima tvarka (pirminį rikiavimą atliks DB testavimo įskiepis). Kairėje lango dalyje pateikiamas einamosios Visio schemos screenshotas, dešinėje – lentelių rikiavimo laukas.



17 pav. Lentelių rikiavimo langas

10. Spauskite „Toliau“;
11. Atsidaro testavimo rezultatų langas, kuris yra panašus į fizinės DB patikros tikrinimo langą, bet pateikia šiame teste aktualesnę informaciją (18 pav.);

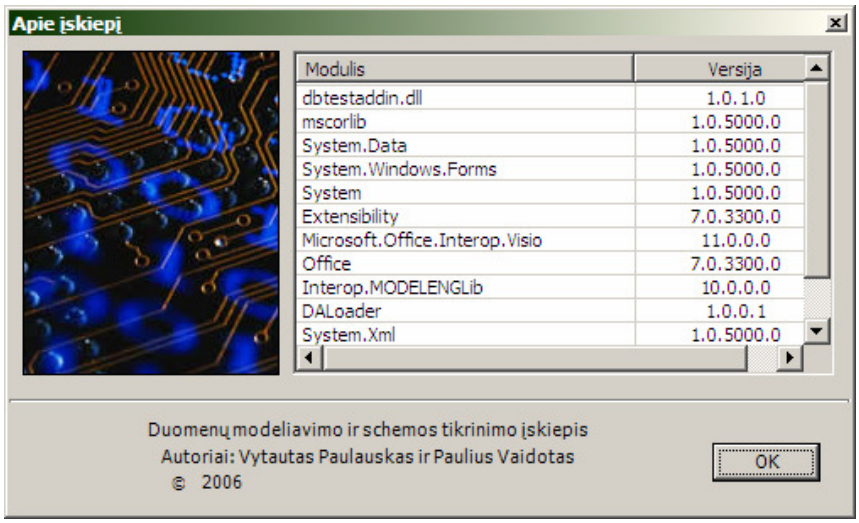


18 pav. DB testavimo rezultatų langas

12. DB testavimas baigtas.

Apie

Mygtukas Apie suteikia informaciją apie įskiepį, jo autorius bei versijos numerį.



19 pav. „Apie“ langas