KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Virgilijus Gurgždys

KOMPONENTINIS IS MODELIO TRANSFORMAVIMAS

Magistro darbas

Vadovas
prof. S. Gudas

KAUNAS, 2006

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

KOMPONENTINIS IS MODELIO TRANSFORMAVIMAS

Magistro baigiamasis darbas

Vadovas
prof. S. Gudas
2006-01-09

Recenzentas
doc. dr. E. Karčiauskas

IFM 0/4 gr. stud.
V. Gurgždys
2006-12-20

KAUNAS, 2006

Component-based modeling

The traditional approaches to engineering of information systems focus on identifying business requirements and delivering the specific functionality required to automate some activities. Not enough attention is being attached to how the created system will interact with the rest of the business. As a result, there is often a gap between the business requirements and the systems implemented to support them. To bridge this gap, many organizations are developing enterprise architecture to provide a holistic vision of how systems will support their business.

Model-driven architecture (MDA) focuses on modelling activities in software development process and shifts the software development process from the writing code to the modelling activities. MDA separates the business level from the technological platform which implements information system. The key feature of the model-driven architecture is its ability to transform automatically the platform-independent model (PIM) into the platform-specific model (PSM). MDA uses modelling languages as programming languages

Using interface-based programming is the evolution of the object-oriented programming and design. The interfaces have made the software design more adaptable to the rapid changes of the business environment. While using interfaces, software systems achieve reusability, extensibility and maintainability.

The CBD approaches use the interface-based design idea and therefore have advantages such as more effective management of complexity, a greater degree of reuse and a wider range of usability.

The CBD requires a systematic approach to focus on the component aspects of software development. The traditional software engineering approaches must be adjusted to the component-based approach. The software reuse and rapid IS development are obtained building systems from components.

The building of the information systems from components requires methodologies and processes not only in relation to the aspect of development, but also to the entire life cycle of system.

This paper deals with the strengths and the limitations of temporary IS development approaches and depicts the IS development approach based on the component-based system model (CBSM) and the MDA principles.

Turinys

1 Įvadas

Viena iš veiklos procesų ir taikomųjų programų integravimo metodologijų vadinama “architektūriniu modeliavimu” ar “architektūra grindžiamas IS projektavimas” (architecture-driven).

Veiklos informacinė architektūra apima bendros sistemos struktūros, sistemos komponenčių, loginių jų ryšių ir išoriškai matomų savybių modeliavimą (projektavimą). Organizacijų informacinės architektūros modeliavimas tiesiogiai skirtas informacijos sistemų, atitinkančių realius veiklos poreikius, projektavimo ir realizavimo metodams vystyti.

Aptariamas metodas aprašo architektūrinio IS projektavimo etapą, kuriame identifikuojami IS projekto komponentai ir jų sąsajos (interfeisai).

Organizacijos informacijos sistemos komponentams ir sąsajoms tarp jų identifikuoti siūloma nauja grafinė notacija – komponentinis sistemos modelis. Šis modelis apjungia veiklos informacinės architektūros (VIA) modelio ir darbų sekos modelio savybes.

Veiklos informacinės architektūros modelis apibrėžia IS komponentų tipus, atitinkančius organizacijos veiklos domenus.

.

2 Sistemos analizė

2.1 Analizės tikslas

Šios analizės tikslas:

- išanalizuoti pasirinktą architektūrą bei ja paremtas sistemas, kompiuterinius metodus, projektuotojo poreikius, panašias sistemas bei standartus;
- išsiaiškinti technologijos bei ja paremtų sistemų trūkumus bei esamas savybes;
- apibrėžti savo kuriamos sistemos tikslus ir siekiamus kriterijus.

2.2 Tyrimo sritis, objektas ir problema

Komponentinis IS projektavimas yra orientuotas į didelių sistemų kūrimą su galimybe integruoti anksčiau egzistavusios programinės įrangos komponentus. Norint padidinti sistemų pritaikomumą ir palaikymą, komponentinis projektavimas gali būti potencialiai nudojamas sumažinti programinės įangos kūrimo kaštus, parengti sistemas greitai, ir supaprasti didelių sistemų atnaujinimo procesą. Šio metodo pagrindas yra prielaida, kad tam tikros didelių programinės įrangos sistemų dalys vėl pasirodo reguliariai sėkmingai, kai dažniausiai naudojami sistemos komponentai galetų būti sukuriami vieną kartą. Tokie komponentai galėtų buti ir nusiperkami bei sėkmingai pritaikomi turimoje sistemoje.

Kompontais paremtos sistemos apima komercinius produktus ir komponentus pasiekiamus kitais būdais (pvz. nekomerciniai produktai). Komponentinės programinės įrangos kūrimas yra tinkamas jeigu:

- Pagerinama produktų kokybė ir įvairovė;
- Ekonominis spaudimas sumažina sistemos kūrimo ir palaikymo kaštus;
- Aiški komponentų integravimo technologija;

Padidinamas egzistuojančios programinės įrangos skaičius organizacijose kurios gali būti pertvarkomos naujomis sistemomis. Šioje sistemoje projektuotojas turės sukurti darbų sekos modelį ir pateikti jį sistemai, sistema sugeneruos komponentinių sistemos modelių hierarchiją.

2.3 Aplinkos (organizacijos veiklos, jos dalies, funkcijos, proceso ir pan.) analizė

Darbas bus realizuojamas įmonėje, kurios pagrindinė veiklos funkcija – programinės įrangos projektavimas ir kūrimas. Šią funkciją galima išskaidyti į tokius procesus:

- Analizę;
- Projektavimą;
- Realizavimą;
- Testavimą;
- Palaikymą.

Analizės etape pirmiausiai išnagrinėjama rinka ir joje randama niša tam tikros srities sistemai. Jei analizės metu išaiškėja, kad kurti sistemą būtų naudinga ir organizacijai, ir klientams, pradedamas projektavimo etapas. Šiame etape nagrinėjamos panašios jau veikiančios sistemos, tikrinama ką būtų galima pritaikyti naujoje sistemoje. Projektuojas sudaro būsimos sistemos modelį (pvz. darbų sekos diagramą). Realizuojant sistemą suprogramuojamas algoritmas leidžiantis nuo sistemos modelio (darbų sekos diagramos) pereiti prie komponentinio sistemos modelių hierarchijos.

2.4 Vartotojų analizė

2.4.1 Vartotojų aibė, tipai ir savybės

- Komponentų kūrėjai. Komponentai yra kuriami specializuotų komponentų kūrėjų arba projektuotojų dirbančių namuose ir priklausančių didelėms įmonėms. Komponeno kūrėjo užduotis yra įvykdyti dažniausių klientų arba vartotojų reikalavimus.
- Komponentų vartotojai. Dažniausiai komponentai yra adapuoti, kad tiktų konkrečiam vartotojui. Komponentų vartotojas integruoja nustato ar komponentas atitinka sistemos specifikacijas ir integruoja jį į sistemą.
- Sistemos analitikas. Jis išsiaiškina kliento reikalvimus. Analizuoja klieto turimas sistemas.
- Sistemos architektas. Sudaro sistemos kūrimo planą. Atrenka reikaimus komponentus. Seka sistemos kūrim procesą techniniu aspektu, tikrina rezultatų kokybę.
- Sistemos koordinatorius. Dažniausiai yra viena indviduli labai didelio projekto dalis. Jis kontroluoja visą kūrimo procesą, ypač dėl darbo terminų ir kaštų. Jis yra atsakingas tiesiogiai klientui dėl projekto baigimo laiku ir sąnaudų limito.

2.5 Problemos sprendimo metodų literatūros šaltiniuose analizė

Programavimo inžinerija anksčiau buvo labiau orientuota į originalų programinės įrangos kūrimą, tačiau dabar kompanijos nepasiruošusios investuoti laiką ir pinigus į standartizuotos programinės įrangos kūrimą savo pačių žmogiškaisiais ištekliais. Jos siekia, kad naujai diegiamos ir jau įdiegtos programinės įrangos funkcijos galėtų kuo lanksčiau bendradarbiauti tarpusavyje.

Geriausias būdas tai įgyvendinti - komponentų panaudojimas. Komponentai – tai programinės įrangos struktūriniai vienetai, kurie remiasi tam tikrais specifiniais principais:

1. Funkcijos ir duomenų bendras standartas. Tai programinės įrangos objektai susidedantys iš duomenų reikšmių (arba būsenų) ir funkcijų, kurios apdoroja tuos duomenis.

2. Enkapsuliavimas. Programinės įrangos klientas yra nepriklausomas nuo to, kaip programinėje įrangoje apdorojami duomenys ir vykdomos funkcijos. Kitaip tariant, klientas priklauso nuo objekto specifikacijos, o ne nuo jo integracijos.

3. Identifiškumas. Kiekvienas programinės įrangos objektas turi jam unikalią identifikaciją, nepriklausomai nuo jo būsenos.

Šiuos principus detalizuoja komponento sąsaja.

Iš šių principų išsiskiria tokios komponentų savybės:

1. komponentai teikia paslaugą nepriklausomai nuo to, kur vykdomas komponentas ir nepriklausomai nuo jo programavimo kalbos;
2. Komponentas yra nepriklausoma vykdomoji programos dalis, kuri gali susidėti iš vienos ar daugiau vykdomųjų objektų.
3. Komponento sąsaja yra paskelbta ir visi veiksmai vykdomi per paskelbtą sąsają.
4. Komponentų dydžiai gali skirtis nuo paprastų funkcijų iki taikomųjų sistemų.

Komponentinių sistemų taikymas sudėtingų sistemų valdymui – problema suskaidoma į mažesnes problemas ir kiekviena iš jų sprendžiama atskirai, o vėliau – visi sprendimai apjungiami į visumą. Didžiausias uždavinys yra sujungti funkcijas ir su jomis susijusius duomenis į vieną veikiančią sistemą.

Viena iš pagrindinių komponentų savybių yra ta, kad jis privalo būti lengvai pakeičiamas kitu – arba esamo komponento atnaujinta versija, arba visiškai nauju komponentu turinčiu sąsają, kuri palaiko tas pačias funkcijas kaip ir buvęs komponentas.

Pritaikant komponentų pakartotinį panaudojimą naujos funkcijos gali būti lengvai integruojamos į jau sukurtą sistemos karkasą. Atskiros sistemos dalys gali būti pakartotinai panaudojamos arba pakeičiamos naujomis nesutrikdant bendro sistemos veikimo.

Komponentiškai pagrįstas kūrimas nuo kitų metodų skiriasi tuo, kad specifikavimas yra atskiriamas nuo integravimo ir padalinama į sąsajas (interface). Komponentų specifikacijos padalinimas į vieną ar daugiau sąsajų reiškia, kad tarpkomponentinės priklausomybės gali būti apribojamos individualių ryšių tarp susijusių komponentų, o ne vieno komponento ryšiu su visa sistema.

Tai sumažina komponentų atnaujinimo įtaką sistemai. Galima vieną komponentą pakeisti kitu su skirtinga specifikacija bet turinčiu tą pačią sąsają.

Pakartotinio panaudojimo pranašumai:

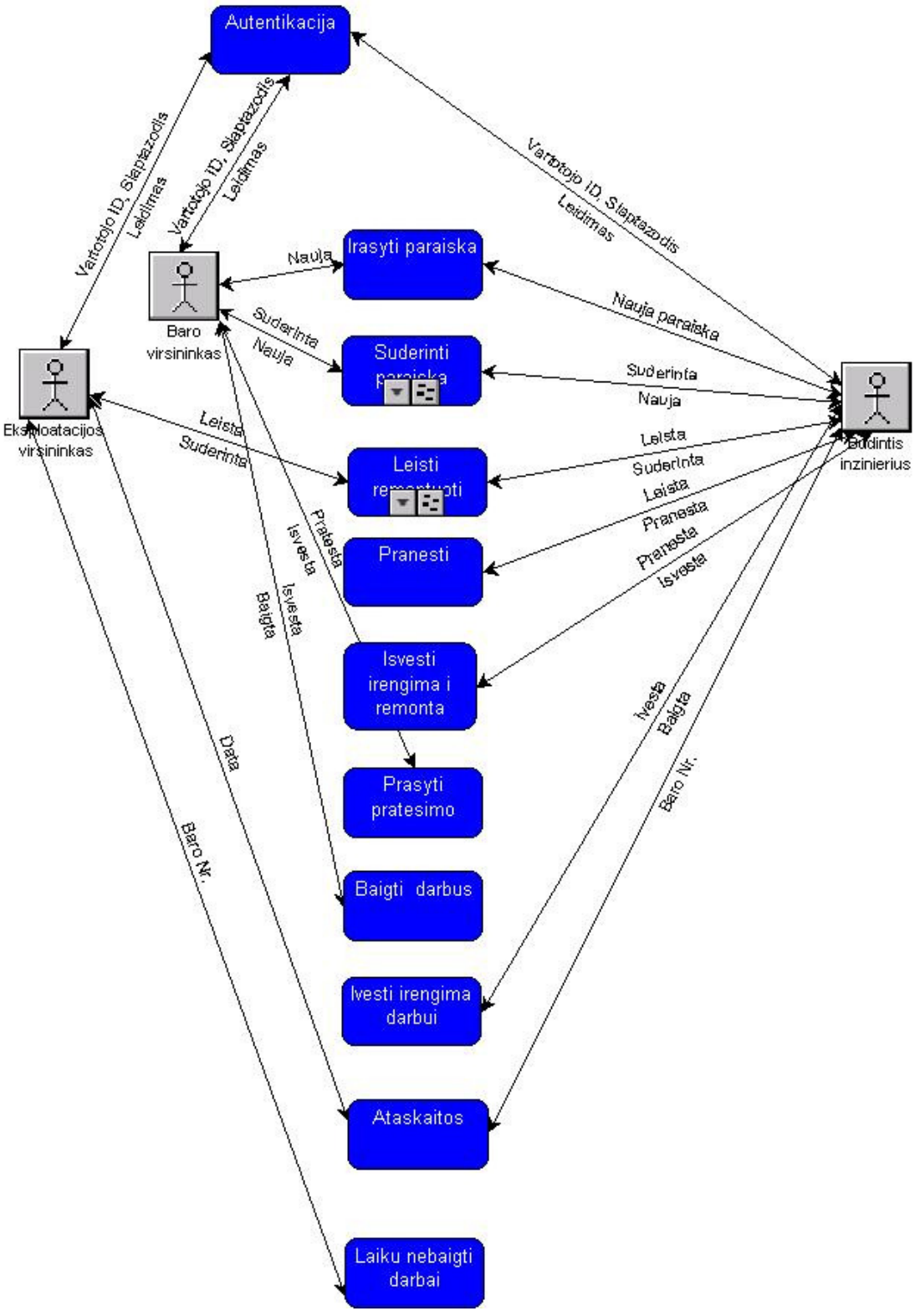
1. padidintas patikimumas – komponentai jau patikrinti veikiančiose sistemose;
2. sumažinta proceso rizika – mažesnis kūrimo kaštų neapibrėžtumas;
3. efektyvus specialistų panaudojimas – žmogiškieji ištekliai gali būti skiriami naujų komponentų kūrimui, o ne senųjų perrašymui;
4. atitikimas standartams – komponentuose įdiegiami standartai;
5. pagreitintas sistemos kūrimas – išvengiama visos sistemos kūrimo nuo pradinio etapo.

2.6 Panašios realizuotos sistemos analizė

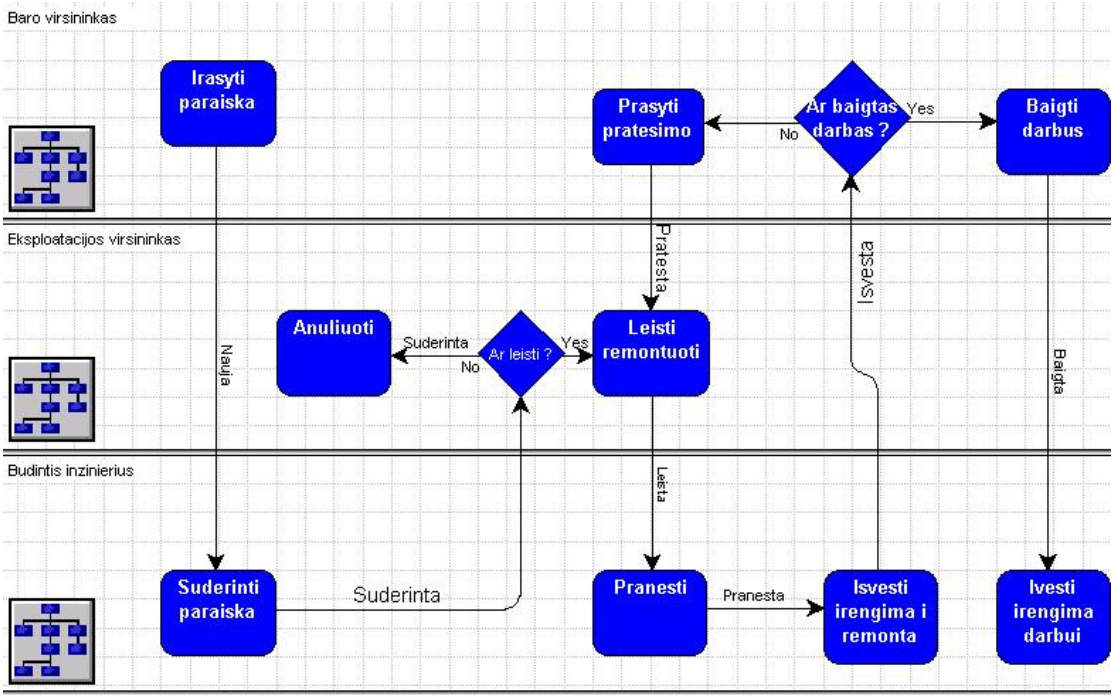
2.6.1. Veiklos proceso ir vartotojo informacinių poreikių analizė.

2.1. Panaudojimo atvejų ir darbų sekų diagramos

Pro Vision aplinkoje suprojektuojame panaudojimo atvejų ir darbų sekų modelius.



1 pav. Paraiškos kortelių panaudojimo atvejų modelis



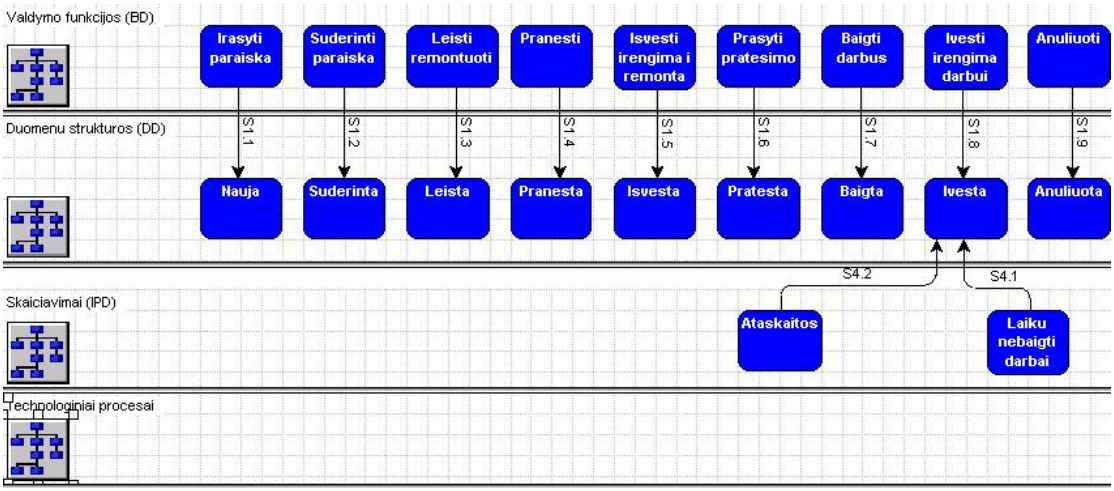
2 pav. Paraiškos kortelių darbų sekų modelis

2.6.2. Komponentinio sistemos sudarymas

Komponentinis veiklos modelis suformuojamas transformuojant darbų sekų modelį pagal šias taisykles:

1. Darbų sekos modelyje skaičiavimą atliekantys procesai transformuojami į informacinių procesų domeno (IPD) komponentus;
2. Darbų sekos modelyje valdymą atliekantys procesai transformuojami į verslo domeno (BD) komponentus;
3. Informacijos srautai, jungiantys procesus darbų sekos modelyje, transformuojami į duomenų domeno (DD) komponentus.
4. Materialūs srautai darbų sekos modelyje transformuojami į technologinių procesų domeno (TPD) komponentus.

Transformavus paraiškos kortelių darbų sekų modelį, pagal aukščiau pateiktas taisykles, mes gavome komponentinį sistemos modelį (KSM).



4 pav. Paraiškos kortelių KSM

2.6.3. KSM specifikavimas

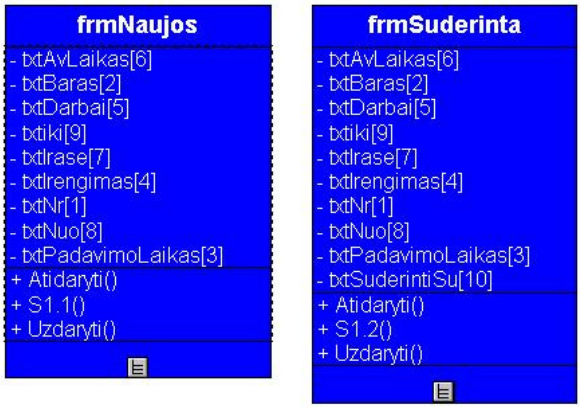
Po KSM sudarymo, kiekvienas KSM domeno komponentas aprašomas kaip išplėsta klasė)

Pastaba: *išplėsta klasė yra UML klasė, papildyta sąsajų sluoksniu šalia tradicinių atributų ir metodų sluoksnių)*

Turi būti sudarytos šios klasių diagramos:

- biznio (valdymo) lygmens vartotojų sąsajos komponentų KD;
- duomenų komponentų KD
- uždavinių komponentų KD;
- gamybos (apskaitos) lygmens vartotojų sąsajos komponentų KD;

Straipsnyje bus pavaizduota BD ir DD domenų sąveikos realizacija.



4 pav. Paraiškos kortelių BD domeno komponentų klasių diagramos fragmentas

5 pav. Paraiškos kortelių DD domeno komponentų klasių diagramos fragmentas

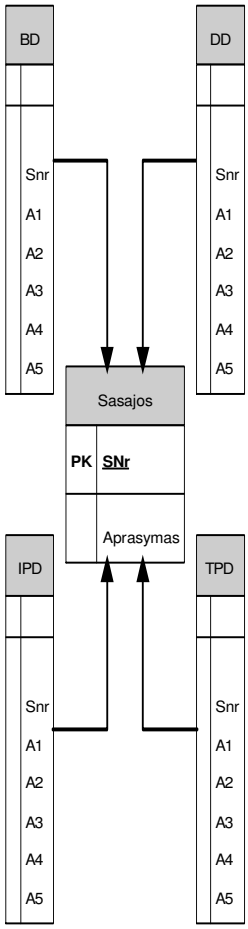
Kiekvienoje KSM domeno komponento klasėje įvedame papildomus atributus, kurie parodo sąsajos numerį, per kurią komponentas bendrauja su kitais komponentais. Be to prie kiekvieno atributo yra įvedamas indeksas, kuris parodo skirtingų domenų komponentų atributų ryšį.

2.6.4. KSM specifikacijų generavimas

Visą projektą sukurtą Pro Vision aplikoje eksportuojame į MS Access DBVS. Šioje DB yra visa informaciją apie projektą.

Tam kad specifikuoti KSM mes turime išplėsti CASE sistemos saugyklą t.y. papildyti žinių struktūra, reikalinga komponentiniam IS realizavimui.

Todėl sukuriame DB "Domenai".



6 pav. KSM atributų saugyklos ER diagrama

7 pav. matome ER diagramą, skirtą KSM atributų informacijai saugoti. Lentelė "Sąsajos" siejasi su KSM domenų lentelėmis:

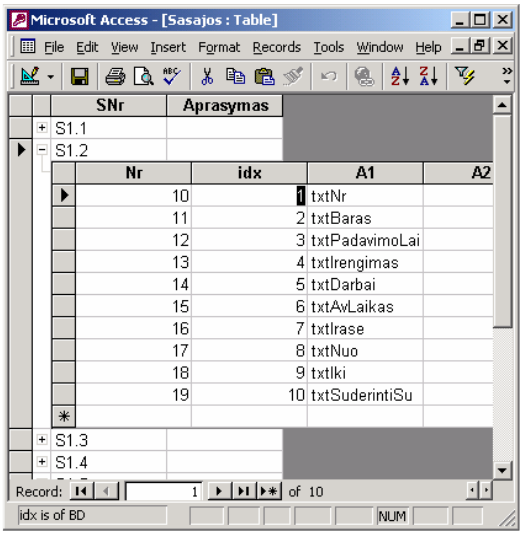
- "BD" – BD domeno komponentų atributų lentelė;
- "DD" - DD domeno komponentų atributų lentelė;
- "IPD" - IPD domeno komponentų atributų lentelė;
- "TPD" - TPD domeno komponentų atributų lentelė;

- Snr – sąsajos numeris;
- A1..An – atributai.

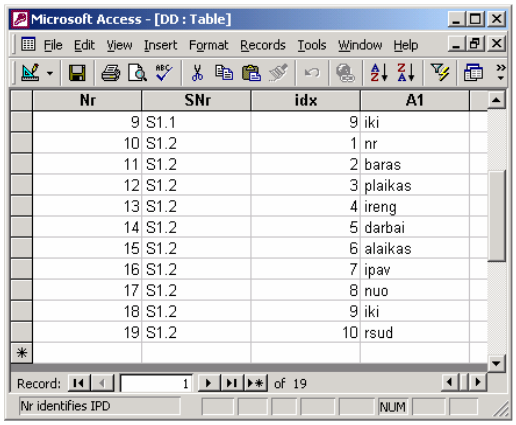
Ši struktūra pakankamai lanksti, mes galime ją lengvai keisti, jei įvestume naujus KSM domenus.

Sukūrus tokią KSM saugyklą, mes galime perkelti duomenis iš CASE sistemos saugyklos į KSM saugyklą.

Po perkėlimo operacijos mes gausime tokį rezultatą (sąsajos S1.2 aprašymas) :



7 pav. KSM S1.2 sąsajos BD komponento atributai



8 pav. KSM S1.2 sąsajos DD komponento atributai

2.6.5. Realizavimas

a. Vartotojo sąsajos generavimas.

Pasinaudodami Provision vartotojo sąsajos generavimo galimybėmis, sugeneruojame ekrano formas. Provision generatorius gali generuoti VB formas.

Formos yra generuojamos iš KSM BD domeno klasių diagramos, todėl visi formų laukai atitiks atributus iš klasių diagramos.

Sugeneravus formas iškyla jų užpildymo duomenimis problema, t.y. formos nėra "pririštos" prie duomenų, kurie yra saugomi DBVS.

VB aplinkoje yra keli duomenų atvaizdavimo, vartotojo sąsajoje, būdai:

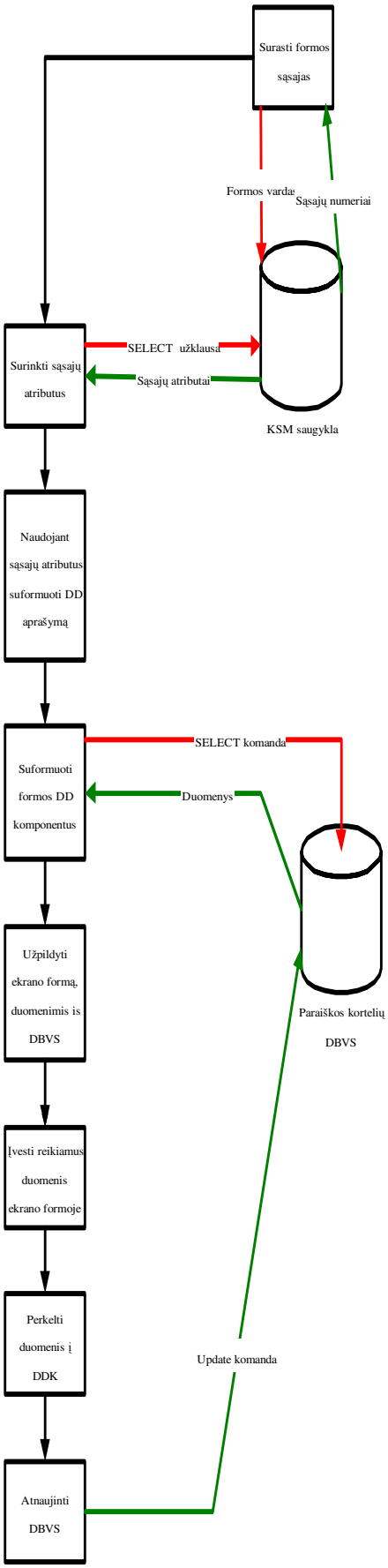
1. VB aplinkoje, nurodome kiekvienam formos laukui atitinkamą duomenų struktūros komoponento lauką. Šis būdas gana lengvai įgyvendinamas ir gali tikti tokiose sistemose, kuriose duomenys į duomenų struktūrų komponentus, programos darbo metu, yra nuskaitomi iš DBVS tik vieną kartą. Toks būdas galėtų būti realizuotas sistemose skirtose vienai darbo vietai.

2. Jei informacinė sistema veikia kompiuterių tinkle ir duomenys gali būti įvedinėjami iš skirtingų darbo vietų 1 būdas – netinka, kadangi atnaujinus duomenis iš DBVS VB aplinkoje visi formų ir kitų objektų laukai, naudojančių informaciją iš DBVS, "atsikabina" nuo duomenų struktūrų komponentų, juose yra rodoma pasenusi informacija.

Todėl užkraunat formą, jos laukams reikia priskirti reikšmes iš duomenų struktūrų komponentų.

Realizuojant paraiškos kortelių IS pagal komponentinį sistemos modelį, buvo sukurtas algoritmas, kuris leidžia dinamiškai kurti duomenų struktūrų komponentus ir susieti juos su verslo funkcijų komponentais.

sc



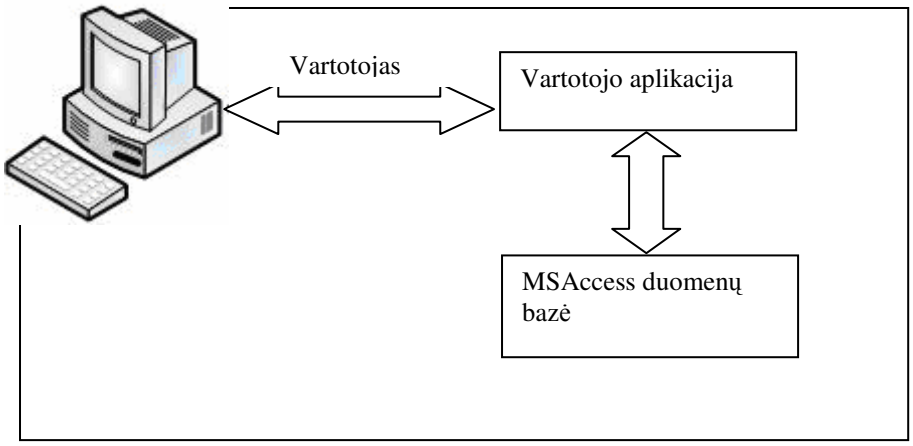
9 pav. Atributų iš komponentų DB panaudojimo algoritmas

2.7 Problemos sprendimo metodų literatūros šaltiniuose analizė

Komponentas apibrėžiamas per atributus. Komponentas privalo turėti:

- Standartą - tai kad yra galimybė surinkti aplikaciją iš atskirų komponentų, reiškia kad visi tie komponentai turi laikytis vieningo standarto;
- Specifikaciją;
- Sąsają;
- Įdiegimą – pats komponentas turi priklausyti tik nuo specifikacijos – jei jis priklausys ir nuo įdiegimo, tai šio komponento pakeitimo ar pakartotinio panaudojimo galimybės sumažės.

.



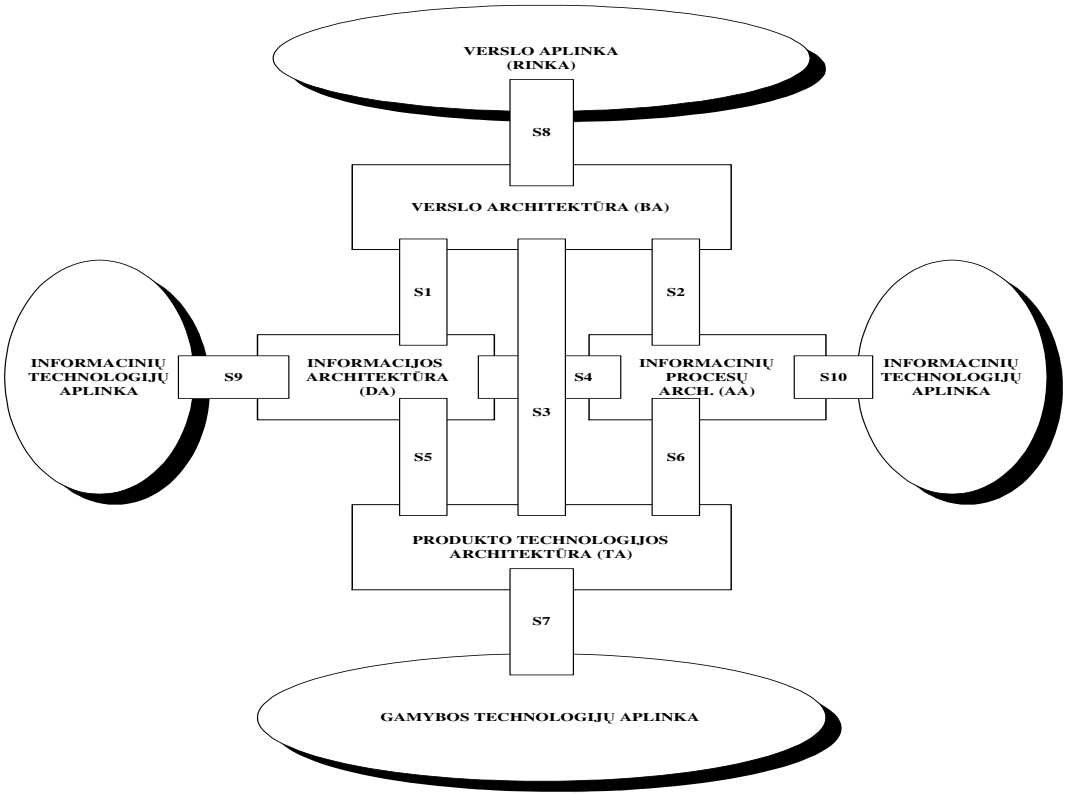
10 pav. Įgyvendinimo priemonių analizė

2.8 Darbo tikslas ir siekiami privalumai

Šio darbo tikslas – išanalizuoti pakartotinio panaudojimo komponentų architektūros principus bei juos pritaikyti informacinių sistemų kūrime. Sukurta sistema turėtų būti lanksti naudojimo atžvilgiu (turi tenkinti tiek patyrusio, tiek naujo vartotojo poreikius), lengvai suprantama, intuityvi, lengvai modifikuojama, palaikyti papildomų komponentų įdiegimą ar esamų pašalinimą nesutrikdant bendro internetinės sistemos funkcionalumo.

Darbo privalumai – įdiegus sistemą tikimės, jog ženkliai sumažės laiko sąnaudos skirtos vienos IS sistemos kūrimui. Sukūrus naują komponentą naujai sistemai, jį bus galima panaudoti ir sekančioms sistemoms, ar juo pakeisti senesnius esamų sistemų komponentus.

Organizacijos veiklos informacinės architektūros (VIA) modelis, sudarytas iš keturių domenų:



11. pav.Veiklos domenai

Galimos sasajos tarp domenų. Domenų sąsajų paskirtis yra integruoti domenų sąveikas, siekiant organizacijos tikslų. Domenų sąsajų architektūros modelis yra gaunamas iš informacijos, surinktos apie domenus, t.y. yra išvedamas iš domenų informacinės architektūros modelio. Domenų sąsajos yra šių domenų informacinės architektūros (IA) komponentų sąsajos.

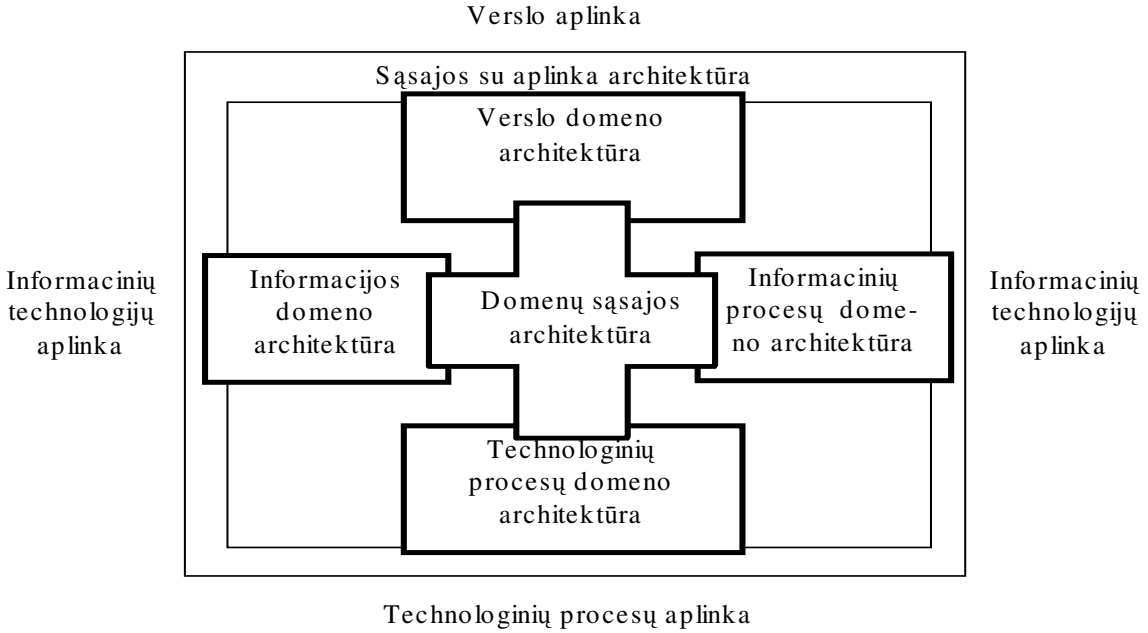
Duomenų sąsajų tipai:

1 lemtelė, sąsajų tipai

Sąsajos tipas	Nurodyto tipo sąsajos siejami komponentai
S1	Duomenų domeno (duomenų komponentų) ir verslo domeno (verslo komponentų) sąsaja
S2	Informacinių procesų domeno (funkcinių komponentų) ir verslo domeno (verslo IA komponentų) sąsaja
S3	Technologinių procesų domeno (TP IA komponentų) ir verslo domeno (komponentų) sąsaja
S4	Informacinių procesų domeno (funkcinių komponentų) ir informacijos domeno (komponentų) sąsaja
S5	Technologinių procesų domeno (TP IA komponentų) ir informacijos domeno (duomenų komponentų) sąsaja

S6	Informacinių procesų domeno ir technologinių procesų domeno (TP IA komponentų) sąsaja
S7	Technologinių procesų domeno sąsaja su produkto gamybos technologijų aplinka (aplinkos IA komponentais).
S8	Verslo domeno sąsajos su verslo aplinka
S9, S10	Informacinių procesų domeno ir informacijos domeno sąsajos su informacinių technologijų aplinka (aplinkos IA komponentais)

Apibendrintas veiklos informacinės architektūros modelis, kuriame skirtingų tipų domenų sąsajos sujungtos į vieną bloką (interfeisų lauką), per kurį integruojama VIA pagrindinių domenų sąveika:

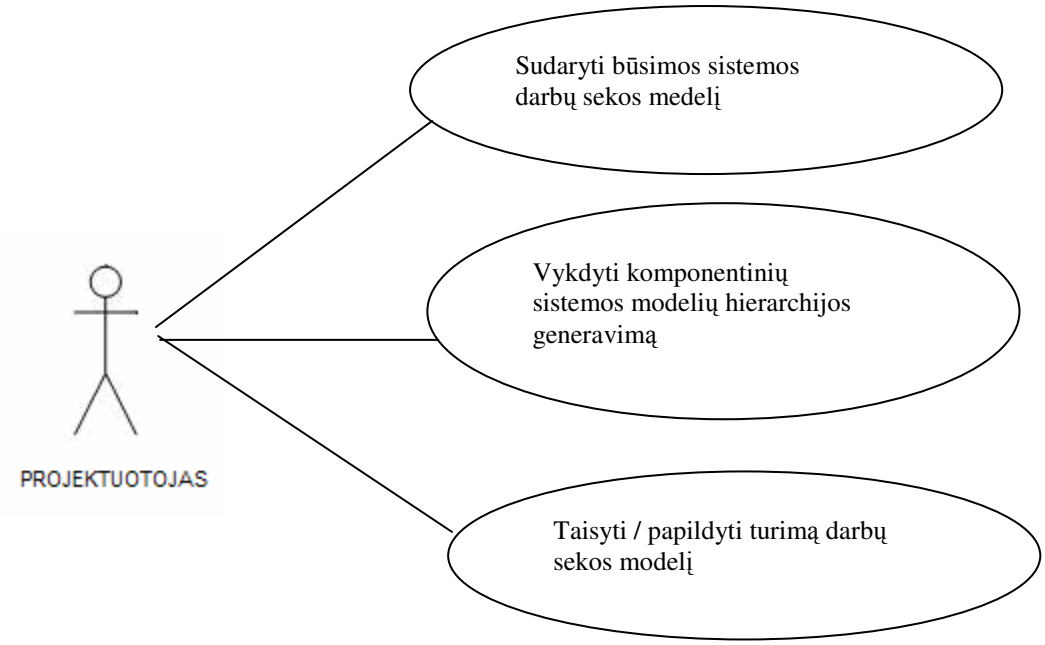


12. pav. Informacinės architektūros modelis

Potencialūs komponentinės IS projektavimo sistemos vartotojai – programinės įrangos kūrėjai, veiklos procesų modelių ir taikomųjų programų integravimui naudojantys metodologiją grindžiamą „architektūriniu modeliavimu (architekture-driven).

2.9 Kompiuterizuojamos sistemos funkcijos

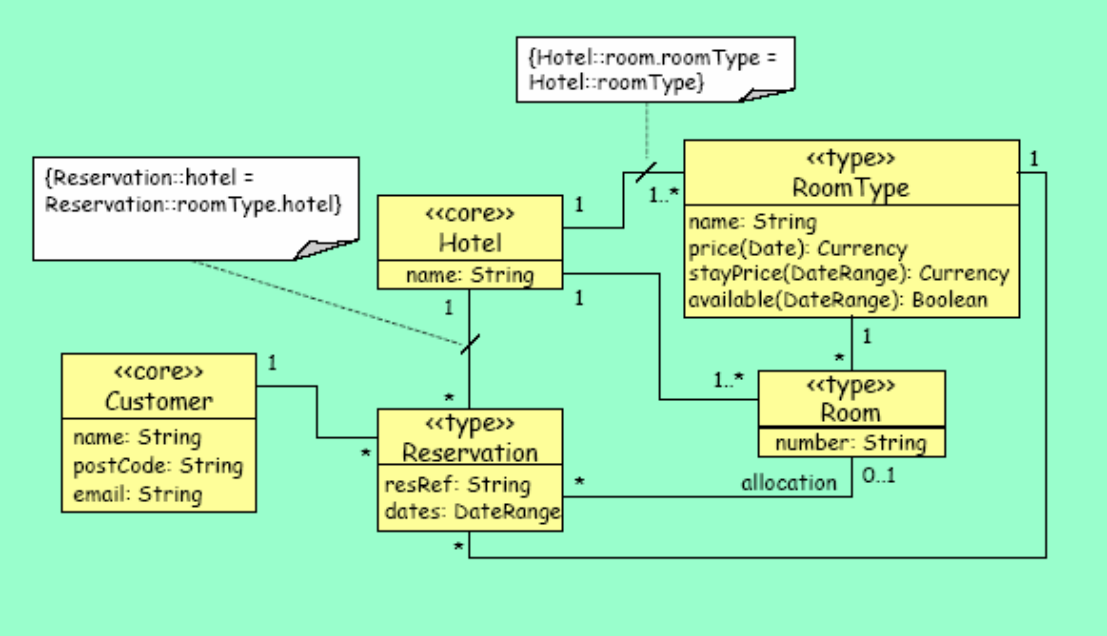
Sistemos pagrindinis vartotojas – projektuotojas. Jis pats projektuoja būsimą sistemą ir įvertina jos teisingumą. Registruoja suprojektuotoje sistemoje pastebėtas klaidas ir jas taiso.



13 pav. Panaudojimo atvejai

2.10 Reikalavimai duomenims

Vartotojas (projektuotojas) sudarydamas kuriamos sistemos darbų sekos modelį turi buti pilnai atsakingas už šio sudaryto modelio teisingumą. Konkrečios IS duomenų tipai duomenų bazėje:



14 pav. Duomenų tipų pavyzdys

2.11 Nefunkciniai reikalavimai ir apribojimai

2.11.1 Reikalavimai standartams

1. Komponentas turi turėti tokias savybes:

- Specifikacija;
- Sąsaja su išore;
- Įgyvendinimas;

Pats komponentas turi priklausyti tik nuo specifikacijos – jei jis priklausys ir nuo įgyvendinimo, tai šio komponento pakeitimo ar pakartotinio panaudojimo galimybės sumažės.

2. Turi būti kompromisas tarp naudojamumo ir pakartotinio panaudojamumo.

Kuo bendresnė sąsaja, tuo geresnis pakartotinis panaudojamumas, bet tada sąsaja sudėtingesnė, vadinasi gali būti mažiau naudojama.

2.11.2 Reikalavimai veikimui

- Sistema turi dirbti stabiliai;
- Tie patys pradiniai duomenys turi duoti tuos pačius rezultatus, nesvarbu kiek kartų būtų vykdoma programa**.**

2.11.3 Reikalavimai sąveikai/suderinamumui su kitomis sistemomis

Sistema veiks naudodama MsAccess duomenų bazę. Vartotojo sąsaja veiks „Windows“ operacinių sistemų terpėje.

2.11.4 Kiti reikalavimai

Sistema turi būti patikima ir saugi. Labai svarbu, kad sistemą būtų galima lengvai plėsti – tokiu būdų kiekvienas galės pritaikyti sistemą savo poreikiams naudodamas jau sukurtus komponentus bei kurdamas savus. Vartotojų duomenys turi būti apsaugoti, t.y. kiti jų negali matyti nebent pats vartotojas tai leidžia.

2.12 Rizikos faktorių analizė

Norint pilnai išnaudoti šios technologijos galimybes, ją reikia taikyti ypač dideliuose projektuose, tokio dydžio sistemą sukurti užimtų daug laiko. Gali būti, kad negalėsime pilnai išnaudoti šios sistemos galimybių.

Su sistema dirbantis projektuotojas, pradiniame etape gali klaidingai suprojektuoti būsimos IS darbų sekų modelį.

2.13 Rezultato kokybės kriterijai

Svarbiausi sistemos kokybės kriterijai:

1. Sistemos veiksmingumas:
    - Ar sukurta sistema veikia tinkamai?
    - Ar išsprendžia išsikeltas problemas?
    - Ar atlieka nustatytus funkcinius reikalavimus?
2. Universalumas:
    - Ar sistemą galima visiškai ar bent iš dalies pritaikyti įvairių portalų kūrimui?
3. Išplečiamumas:
    - Ar sistema lengvai plečiama modifikuojant jau sukurtus komponentus arba pridedant naujus komponentus?
    - Kokios galimybė kurti naujus komponentus?
4. Patikimumas:

- Ar panaikinti arba sumažinti rizikos faktoriai?
- Ar sistema veikia be klaidų?

5. Naudojimosi paprastumas ir lengvumas:
    - Ar visi komponentai būtinai reikalingi?
    - Ar sistemą sudėtinga pritaikyti vartotojo reikmėms?

- Sistemos efektyvumas:
    - Ar sumažėjo sistemos kūrimo laikas naudojant komponentus?

2.14 Analizės išvados

1. Išnagrinėjus jau sukurtas sistemas pastebėjome, kad visos turi savų trūkumų ir privalumų. Dažniausias trūkumas, ribotos pritaikymo ir išplėtimo galimybės.

2. Pasirinktas metodas – pakartotinis komponentų panaudojimas – plačiai taikomas visose jau gyvuojančiose panašaus tipo sistemose, todėl jį savo darbui taikysime ir mes.

3. Apsibrėžėme kompiuterizuojamos sistemos funkcijas ir nustatėme reikalavimus sistemai bei veikimui.

4. Rezultatų kokybės faktoriai:

4.1.Sistemos veiksmingumas

4.2. Sistemos našumas

4.3.Universalumas:

4.4.Išplečiamumas:

4.5.Patikimumas:

4.6.Saugumas:

4.7.Naudojimosi paprastumas ir lengvumas

4.8.Sistemos efektyvumas

5. Darbo metu bus suprojektuota ir realizuota:

5.1.Vidiniai standartai kuriamiems komponentams;

5.2. Komponentinės sistemos pagrindas;

3 Projektavimo dalis

3.1 Įžanga

Tradicinis požiūris į informacinių sistemų inžineriją, paremtas verslo poreikių identifikavimu ir specifinio funkcionalumo suteikimu, reikalingu automatizuoti daliai veiklų. Yra skiriamas nepakankamas dėmesys išaiškinti kaip sukurta sistema sąveikaus su likusia verslo dalimi. To pasiekoje dažnai gaunama spraga tarp verslo poreikių, ir sistemos sukurtos juos išspręsti. Daug organizacijų šią problemą sprendžia kurdamos įmonės architektūrą suteikiančią viziją, kuri nusako kaip sistemos įtakos visą verslo procesą.

Modeliu paremta architektūra (angl. Model-driven architecture (MDA)) remiasi veiklos modeliavimu programinės įrangos kūrimo procese ir padaliną programinės įrangos kūrimo procesą, kodo rašymą atskiria nuo veiklų modeliavimo. MDA atskira verslo lygį, nuo technologinės platformos kurią įgyvendina informacinė sistema. Svarbiausia modeliu paremtos architektūros savybė yra galimybė automatiškai transformuoti nepriklausomą nuo platformos modelį (angl. platform-independent model (PIM)) į specifinį platformos modelį (platform-specific model (PSM)). MDA naudoja modeliavimo kalbas lygiai taip pat, kaip ir programuojant naudojamos programavimo kalbos.

Naudojant interfeisu paremtą programavimą vyksta objektinio programavimo ir projektavimo evoliucija. Interfeisai daro programonės įrangos kūrimo procesą labiau adaptuojamą ir lankstesnį biznio aplinkos pasikeitimams. Kol naudojami interfeisiai, programinės įrangos sistemose galimas pakartotinis panaudojimas, išplečiamumas ir eksploatavimo patikimumą užtikrinančios priemonės.

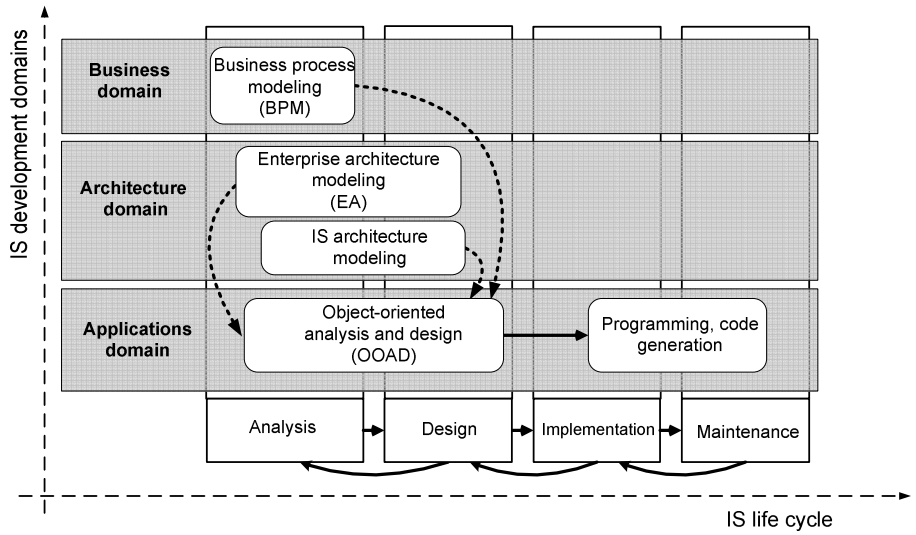
Interfeisu paremto projektavimo idėja ir programavimas yra daugiausiai naudojamas orientuotoje į aptarnavimą architektūroje (angl. service-oriented architecture (SOA)) ir komponentiniame projektavime (angl. component-based development (CBD), kuris yra realizuojamas technoligijomis, tokiomis, kaip Microsoft .NET platforma ir Java 2 Enterprise Edition.

Komponentinio projektavimo technologija, naudoja iterfeisu paremtą projektavimo idėja, ir dėl šios priežasties turi privalumus, tokius kaip efektyvesnis sudėtingų situacijų valdymas, didesnė pakartotinio panaudojimo galimybė ir platesnis pritaikymo spektras.

Komponentiniam projektavimui reikalingas sisteminis metodas, kuris fokusuojasi į komponento aspektus, reikalingus programinės įrangos kūrime. Tradiciniai programinės įrangos inžinerijos metodai turi būti komponentiniam sistemos modeliui. Pakartotinis panaudojimas ir lankstus IS kūrimas pasiekiamas kuriant sistemas iš atskirų komponentų.

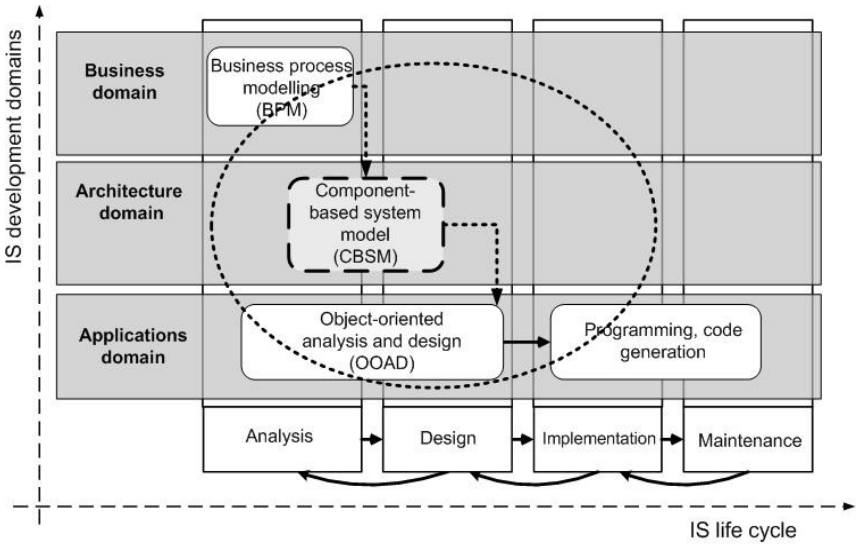
Informacinių sistemų kūrimas iš komponentų reikalauja metodologijų ir procesų ne tik sistemos kūrime, bet ir vientisame programos gyvavimo cikle.

3.2 Semantinė spraga IS kūrimo procese



15 pav., IS projektavimas

15 pav. vaizduojamas IS inžinerijos metodą IS projektavimo cikle. Horozontalioje ašyje yra informacinės sistemos projektavimo ciklas, vertikalioje ašyje yra IS projektavimo lygiai. IS projektavimo metodai, tokie kaip organizacijos architektūra (enterprise architecture (EA)) , verslo procesų modeliai, objektiškai orientuota analizė ir projektavimas yra naudojami skirtinguose lygiuose IS kūrimo procese. Labai dažnas maršrutas IS kūrimo procesuose yra iš BPM į OOAD arba iš EA į OOAD. Dėl šios priežasties iškyla keletas problemų, nes praktikoje BPM, EA ir OOAD kūrimas dažnai yra išskaidytas ir dėl to semantinė sparga atsiranda tarp verslo vizijos ir aukšto lygmens struktūros, taip pat sistemų sukurtų jas aptarnauti.



16 pav., Komponentinio sistemos modelio vieta IS inžinerijos procese

Siekiant sumažinti spragą tarp IS kūrimo veiklų, yra priežastis sukurti kompontinį metodą (paveikslėlyje jis pažymėtas taškuotu ovalu), kuris yra naudjamas integruoti elementus iš skirtingų modeliavimpo technikų ir komponentinį sistemos modelį, kuris integruoja įmonės architektūrą su informacinės sistemos architektūra (kvadratas apvaliais kampais).

3.3 Pagrindiniai šiuolsikinio IS projektavimo metodo požymiai

BPM aprašo biznio domeną be IT vaizdo. Biznio procesai padeda aprašyti svarbiausius procesus įmonės viduje ir tarp organizacijų. Biznio procesai taip pat aprašo sąveikas, tarp bet kurių dalyvių.

Procesai keičiasi informacija su kitais procesais. Taigi mes turime sąveikaujančių procesų sistemą ir sąveika yra atliekama per sąsają. Sąsaja naudojama informacijos apsikeitimui tarp įrašų. Sąsaja gali būti realizuota naudojant interfeisus. Šie interfeisai bus aprašomi komponentiniamia sistemos modelyje.

Darbų sekos modelis atspindi tradicinį procesų įgyvendinimo įmonėje vaiszdą. Darbų sekos modelyje, biznio procesų modeliavimas seka iš eilės veiklų kurios gali būti padalintos lygiagrečias sekas ir padalintos į subsekas ir elementarias veiklas.

BPM metodas leidžia galutiniame vaizde matyti funkcines sistemos dalis, bet paprastai jos nėra pasiekiamos architektūroje ir įgyvendinamos domene.

Įmonės architektūros sistemos, tokios kaip DoDAF, TOGAF, Zachman yra naudojamos architektūros modeliavimui IS kūrime. Sistemos suteikia struktūrizuotus ir semantinius duomenis, žinias informacinės sistemos projektavimui.

„Architektūros“ apibrėžimas naudojamas ANSI/IEEE Std 1471-2000 yra: „pagrindinė sistemos organizacija, įgyvendinama jos komponentuose, jų tarpusavio ir aplinkos santykiuose, pagrindiniai principai yra konstrukcija ir evoliucija“

Architektūros sistema padeda pagerinti domeno problemos supratimą, bet kai kurie architektūros aspektai lieka neaiškūs. Yra keletas dviprasmiškumų:

- Ar turėtų architektūros sritis apimti tik programinės įrangos komponentus ar įtraukti kitus informacinės sistemos kūrimo aspektus?
- Architektūros veiklos apima konstrukciją ir modeliavimą, bet kuris detalių lygis priklauso architektūrai ir kada detalios konstrukcijos veiklos startuoja?
- Koks yra santykis tarp įmonės architektūros ir informacinės sistemos architektūros?

Siūlomas komponentinio modeliavimo metodas siekia pašalinti šiuos dviprasmiškumus.

Objektiškai orientuota analizė ir projektavimas aplikacijų lygyje leidžia efektyviai projektuoti ir kurti aplikacijas.

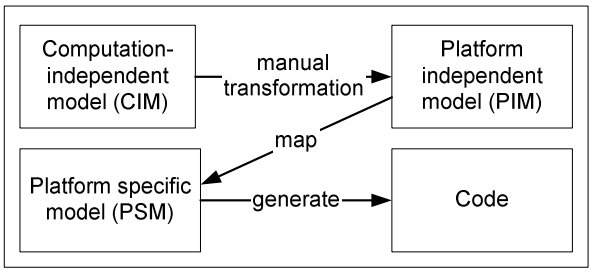
Pagrindinės problemos yra tos, kad OOAD metodas yra naudojamas pagrinde naudojamas aplikacijos lygyje, bet ne BPM lygyje, ir OOAD projektavimo lygyje skaidymo gylio, klasės lygyje, kuris yra per žemas abstrakčiame biznio proceso modeliavime. Stipri asociacija, tokia kaip paveldimumas, sukuria glaudų ryšį ir priklausomumą tarp OOAB įrašų, todėl yra sudėtinga eksploatuoti arba praplėsti sistemą, „nelaužant“ egzistuojančio IS sistemos kodo.

OOAD modeliai yra naudojami, programinės įrangos sistemų modeliavimui, techninei programinės įrangos specifikacijai. BPM ir OOAD yra susiję, biznio procesų modelis gali būti OOAD modelio specifikacija, kuri kartu yra ir specifikacija informacinės sistemos.

OOAD yra labai reikšmingas metodas, padedantis projektuoti pakeičiant klasių struktūrą IS viduje.

Modeliavimu paremta architektūra (angl. Model driven architecture (MDA))

17 pav. vaizduojamas MDA procesas, turis prasideda skaičiavimų nepriklausmo modelio (CIM) sukūrimu. Šis modelis vaizduoja sistemą biznio domeno projekto nariams. Reikalavimai yra užfiksuoti CIM. CIM palengvina komunikaciją tarp domeno ekspertų ir sistemos projektuotojų.



17 pav., Modelių sąveika

Po to kai CIM modelis yra sukurtas, sistemos kūrėjai kuriai objektiškai orientuotus modelius. Šie modeliai yra vaizduojami PIM kur sistemos reikalavimai vaizduojami UML diagramomis. PIM turi užfiksuoti domeno semsntiką. Neprikalausomi nuo platformos modeliai yra žymimi į PSM modelius. Žymėjimas tarp modelių yra fiktyvus, siekiant gauti vieną arba duagiau modelių kaip įėjimo ir gaminti vieną išėjimo modelį. Taisyklė žymejimui yra aprašomos žymėjimo funkcijos viduje. Žymėjimo f-jos leidžia pasirinktų modelių statybą, kurie yra sinchronizuoti su jų šaltinio modeliais. Žymėjimo f-jos savarankiškai ne visada yra pakankamos transformuoti šaltinio modelį pilnai ir papildomi įėjimo duomenys gali būti reikalingi atlikti žymėjimą, dėl to yra naudjamos žymės. Žymės yra modelių plėtinys, kurie fiksuoja informaciją reikalingą modelio transformacijai.

Naudinga modeliu paremtos architektūros savybė yra automatikšai transformuoti nepriklausomą nuo platformos modelį į specifinį paltformos modelį. Projektuotojai kuriai transformavimo taisykles , kurios automatikškai konvertuoja modelį į kodą. UML modeliavimas suteikia aukšto lygio programinės įrangos sistemos atvaizdavimą. Sugeneruotas kodas sukuria pagrindą IS pagrindą kuri bus realizuota.

3.4 Komponentinis sistemos modelis

Remiantis šiuolaikinio IS projektavimo rezultatu, buvo sukurtas komponentais paremtas sistemos kūrimo metodas, kuris leidžia taikyti skirtingus elementus iš laikinų IS kurimo metodų, siekiant paspartinti ir supaprastinti IS

inžinerijos procesą. Tai yra privalumas, kurį IT architektūra glaudžiai susijęs su IT architektūros procesais. Šis metodas leidžia sulyginti IT architektūrą su biznio procesu.

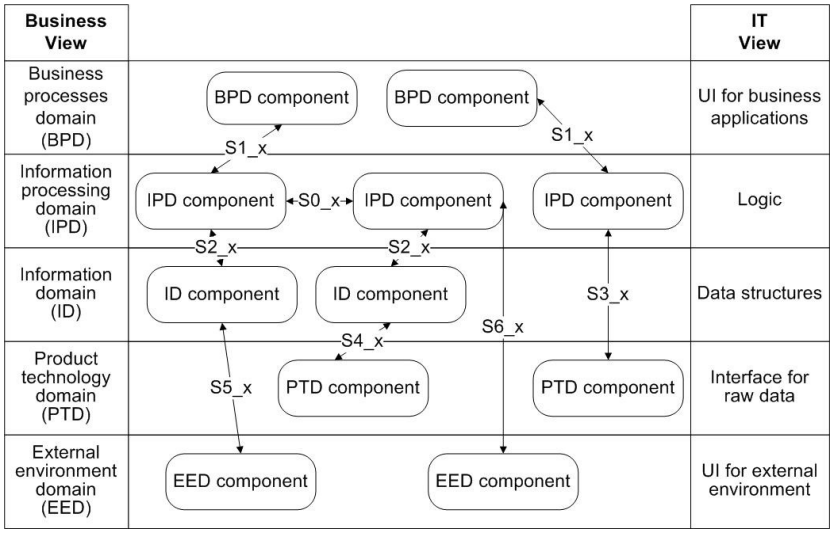
Komponentais paremtas vaizdas yra naudojamas visuose IS kūrimo etapuose. Tai leidžia surinkti informacinę sistemą iš paskirstytų komponentų ir paslaugų ir naudoti ir naudoti orientuotą į paslaugas vaizdą IS kūrimo procese.

Sukurtas metodas naudoja idėjas iš kompoentų ir modelių paremtos architektūros ir integruoja biznio reikalavimus ir realizuotą informacinę sistemą biznio procesų aptarnavimui.

… pav vaizduoja pateiktą komponentinį sistemos modelį (CBSM) grafinėje notacijoje, informacinės sistemos komponentų ir jų santykių identifikavimui.

Komponentinis sistemos modelis padalinti IS į domenus ir nuststyti ryšį organizacijos architektūros su IS architektūra. Modelis aprašo IS kaip rinkinį komponentų arba servisų kurie sąveikauja tarpusavyje, įveda aiškų apibrėžimą apie šių komponentų ar servisų bendravimą tarpusavyje.

Komponentinis sistemos modelis leidžia nuspręsti kaip komponentai gali būti grupuojami, akip jie įtakoja sistemą ir vienas kitą, kaip komponentais dalinamasi informacinėje sistemoje.



17 pav., Komponentinis sitemos modelis

Komponentai komponentiniame sietemos modelyje yra rodomi, kvadratai apvaliais kampais, interfeiai vaizduojami linijomis nurodant interfeiso tipą. Yra taisyklės pagal kurias komponntinis sistemos modelis gaunamas iš verslo proceso aprašymo:

Biznio procesų domenas (BPD) – apima biznio procesus, kritinius organizacijos funkcionalumui ir plėtrai, marketingui, operacijų strategijai, gsmybos planavimui ir žmogiškųjų išteklių valdymui.

Informacinių procesų domenas (IPD) – identifikuoja pagrindinę veiklos informaciją, kuri reikalinga įmonei vykdyti verslą paremtą sprendimais ir produktais;

Informacijos domenas (ID) – apima veiklas skirtas apdoroti duomenis ir žinias reikalingas įmonės valdymui ir informacijos sistemos komponetų ir jų sąveikų producktų kūrimui; pvz.: kokybės standartų valdyme, produktų ir procesų apibrėžimuose, inventorizuojant failus ir t.t.

Technologinių procesų domenas (PTD) – apima technologinį procesą ir įrankius įmonės produktų vystymui, pvz.: produktų projektavime, meterialių išteklių valdyme.

Išorinių aplinkos veiksnių domenas **(EED) –** apima veiklas skirtas organizuoti procesus su įmonės tiekėjais ir klientais.

Komponentinio projektavimo reikšmė informacinių technologijų požiūriu:

Vartotojo aplikacijos ir verslo aplikacijos – apima komponentus kuriuos įmonės darbuotojai naudos sąveikai su sistema. Pvz.: langai, ekranai, formos.

Logika – apima aplikacijos logikos komponentus. Komponentai vykdo verslo taisykles;

Duomenų struktūros – apima duomenų struktūras;

Interfeisas neapdorotiems duomenis – apima duomenis iš technologinio proceso ar valdymo sistemų;

Interfeisas išoriniams vartotojams – apima interfeisą išoriniams vartotojams arba klientų/tiekėjų informacijos sistemoms.

Biznio domenai sąveikauja vienas su kitu. Interfeisų tikslas integruoti domenų sąveikas. Interfeisai yra grupuojami į šiuos tipus:

S0_x – interfeisas tarp komponentų esančių tame pačiame domene;

S1_x – interfeisas tarp biznio procesų domeno ir informacijos procesų domeno;

S2_x – interfeisas tarp informacijos procesų domeno ir informacijos domeno;

S3_x – interfeisas tarp informacinių procesų domeno ir produktų technologijų domeno;

S4_x – interfeisas tarp produktų technologijų domeno ir informacijos domeno;

S5_x – interfeisas tarp išorinių aplinkos veiksnių domeno ir informacijos domeno**;**

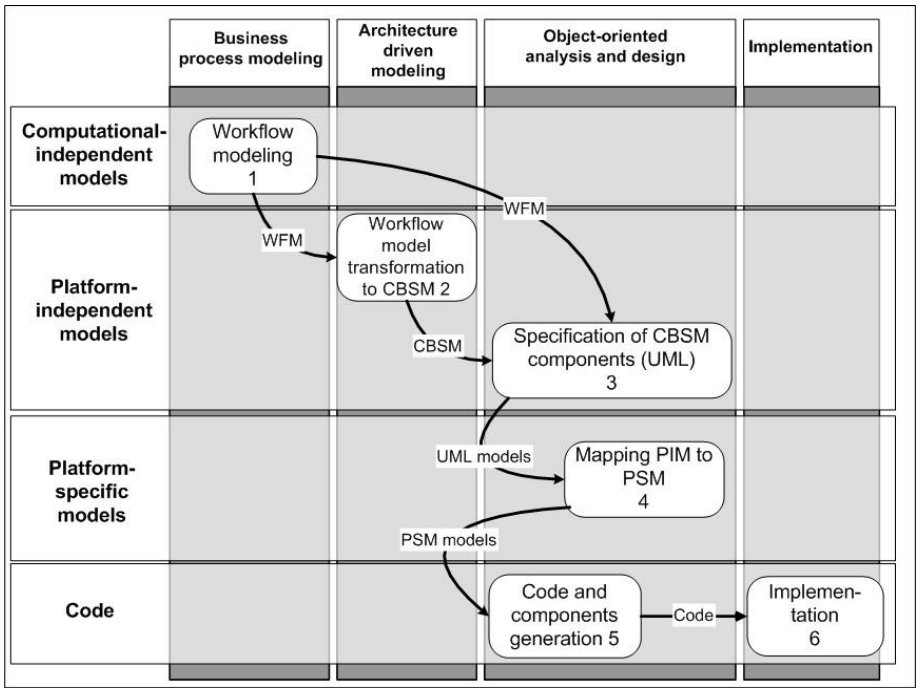
S6_x – interfeisas tarp išorinių aplinkos veiksnių domeno ir informacijos procesų domeno;

Indeksas “x” parodo interfeiso numerį.

Siekiant geriau išnaudoti žinias komponentinininiame sistemos modelyje buvo sukurtas CBSM UML profailas. Šis profailas turi penkis stereotipus - stereotypes - <<BPD>>, <<IPD>>, <<ID>>, <<PTD>>, ir <<EED>>. Stereotipai naudojami komponentų žymėjimui. Kai komponnetas yra komponentinio sistemos modelio dalis, jis automatiškai yra pažymimas takelio sterotipu.

3.5 Informacinė sistemos projektavimas – atvejų studija

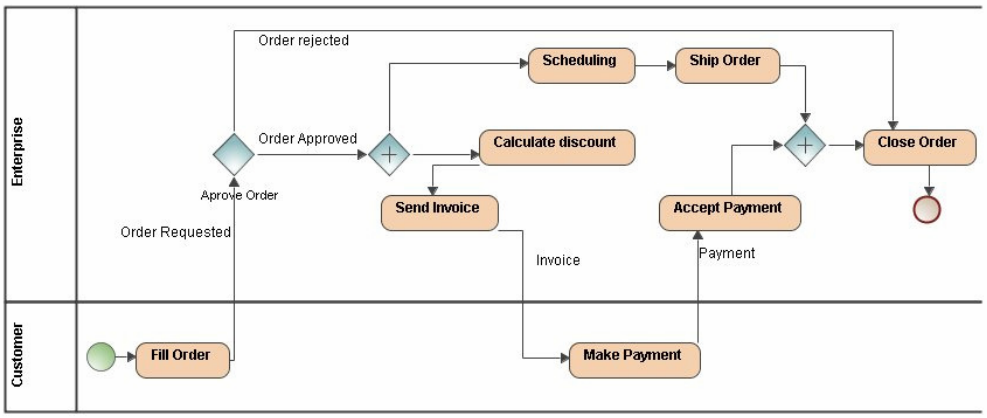
Pav. nr ... vaizduoja pagrindinius vaidmenis ir veiklas komponentiniame sistemos modelyje.



18 pav., Komponentinis sitemos modelis

Informacinės sistemos projektavimas naudojant CBSM.

Pirmame žingsnyje atliekamas biznio procesų modeliavimas siekian išsiaiųkinti biznio reikalavimus. Biznio modelaivimui naudojamas MagicDraw įrankis ir Business Process Modelling Notation (BPMN). BPMN yra biznio procesų standartas biznio procesams. BPMN.

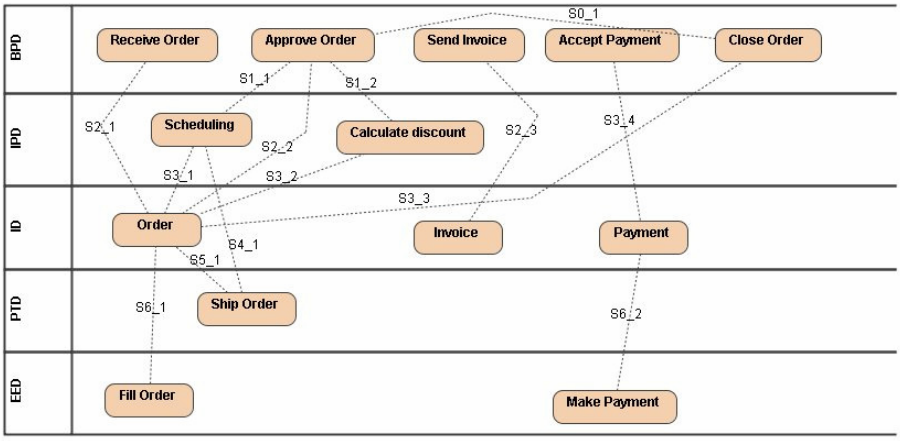


17 pav., Biznio modelis

Antrame žingsnyje atliekamas žymėjimas iš nepriklausomo skaičiavimų modelio į platformai priklausantį modelį. Žymėjimo taisyklės yra nustatomos kai projektuotojas atlieka komponentinę analizę – analizuoja BPD siekiant sukurti komponentinį sistemos modelį. MDA požiūriu. Iš MDA požiūrio, tai yra platformos nepriklausomas modelis. Projektuotojas vadovaujasi šiomis žymėjimo taisyklėmis:

- Skaičiavimo procesiai yra transformuojami į Informacinių procesų domeno (IPD) komponentus;
- Valdymo procesai ir išėjimai yra transformuojami į biznio procesų domeno komponentus;
- Informacijos srautai jungiantys procesus yra transformuojami informacijos domeno(ID) komponentus;
- Materialūs procesai yra transformuojami technologijų procesų domeno komponentus;
- Procesai iš išorinės biznio aplinkos transformuojami į išorinių aplinkos veiksnių komponentus;

Išaiškinti komponentai talpinami komponentiniame sistemos modelyje. Sąveikos santykiai tarp komponentų yra laikomi komponentų interfeisais.

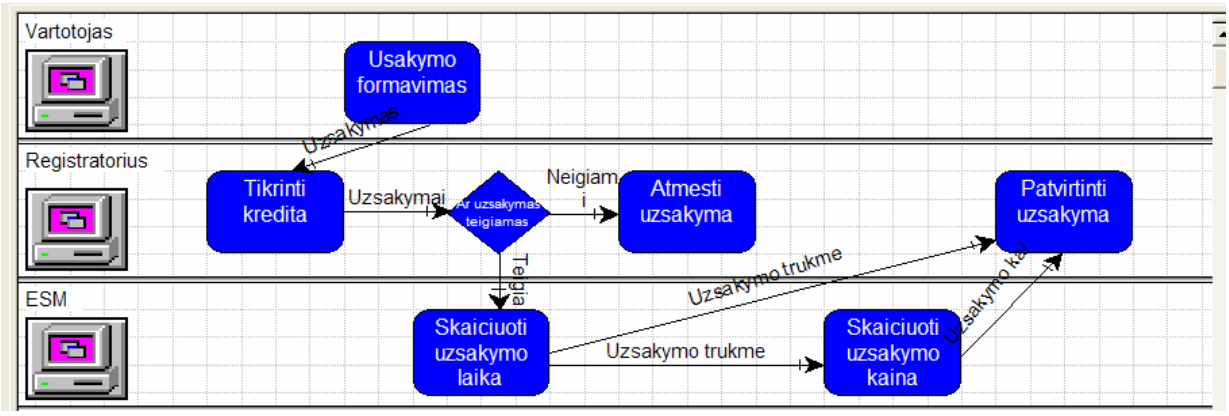


18 pav., Komponentinis modelis

4 Realizacija

4.1 Darbų sekos diagrama

Įmonėje vykstantys paprastai ir suprantamai darbe darbų sekos diagrama. Savo darbe naudojausi „ProVision Workbench v4.0“ įrankiu šiai diagramai sudaryti. Diagramos pavyzdys:



19 pav. Darbų sekos modelis

Kuriant realiai veikiančią sistemą pagal darbų sekos diagramą, programuotojui dažnai trūksta informacijos sudaryti pilną informacinės sistemos modelį, pagal kurį jau būtų galima rašyti programos kodą. Mano nagrinėjamas metodas skirtas sistemų projektuotojui, transformuojant darbų sekos modelį į komponentinį sistemos modelis. Atlikus šį procesą užpildoma trūkstamos informacijos spraga tarp įmonės procesų darbų sekos modelio ir užduoties skirtos

programuotojui. Komponentinis sistemos modelis atvaizduoja informacinę sistemą atskirais sistemos komponentais, priskiriant juos atskiriems organizacijos veiklos domenams, o ryšį tarp komponentų identifikuojant sąsajomis.

4.2 Organizacijos veiklos domenai

2 lentelė, veiklos domenai

Domennas	Žymėjimas	Paskirtis
Verslo procesų domenas	BD	Tai ekonominę ir gamybinę veiklą vykdančios organizacijos dalies (valdymo funkcijos, ekonominė veikla) informaciniai poreikiai ir reikalavimai IS
Informacijos domenas	DD	Tai duomenys, žinios ir tikslai, jų saugojimo ir perdavimo organizacijos padaliniams procesai;
Informacinių procesų doemenas	IPD	Organizacijoje atliekami skaičiavimai, sprendimo priėmimo procesai, galima vadinti taikomųjų uždavinių domenu
Technologinių procesų doemnas	TPD	Tai organizacijos dalies, atliekančios produkto gamybą (ar formavimą) - produkto gamybos procesų informaciniai poreikiai ir reikalavimai IS
Darbo vietų domenas	DVD	Darbo vietų visuma, informaciniai reikalavimai darbo vietose atliekamoms funkcijoms. Darbo vietų domenas yra pasiskirstęs, t.y. darbo vietos išsidėstę kituose domenuose ir sąsajose

4.3 Objektų transformavimo taisyklės iš darbų sekos modelio į komponentinį sistemos modelį.

Darbų sekos modelio transformavimas į komponentinį sistemos modelį vyksta naudojantis šiomis taisyklėmis:

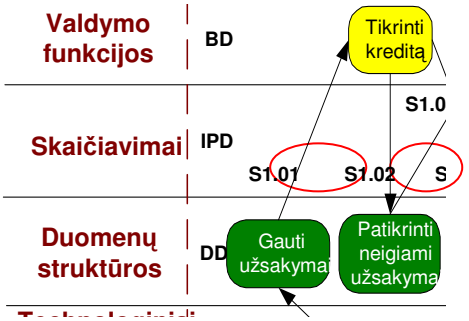
1. Skaičiavimo procesai transformuojami į Informacinių Procesų Domeno komponentus;
2. Valdymo procesai transformuojami į Verslo Procesų Domeno komponentus;
3. Informacijos srautai sujungiantys procesus darbų sekos modelyje transformuojami į Informacijos Domeno komponentus;
4. Materialūs procesai transformuojami į Technologinio Proceso komponentus;

5. Procesai iš arba į išorinę verslo aplinką yra transformuojami į išorinių aplinkos veiksnių domeną;

4.4 Sąsajos tarp komponentų ir jų žymėjimas.

1. Verslo proceso domeno komponentai gali sąveikauti su Informacinių procesų ir Informacijos domeno komponentais;
2. Informacinių procesų domeno komponentai gali sąveikauti su verslo procesų domeno, informacijos domeno ir technologinių procesų domeno komponentais;
3. Technologinių procesų domeno komponentai gali bendrauti su informacijos domeno komponentais ir su informacinių procesų domeno komponentais;
4. Išorinių aplinkos veiksnių komponentai gali bendrauti su informacinių procesų domeno komponentais ir su informacijos domeno komponentais;

Sąsajos numeruojamos pagal tai kokių organizacijos veiklos domenų komponentai sąveikauja. (galimi sąsajų tipai aprašyti lentelėje nr 1) Be sąsajos tipo, kiekviena sąsaja dar numeruojama iš eilės einančiu sąsajos numeriu. Sąsajų ir sąsajų numeravimo pavyzdys:



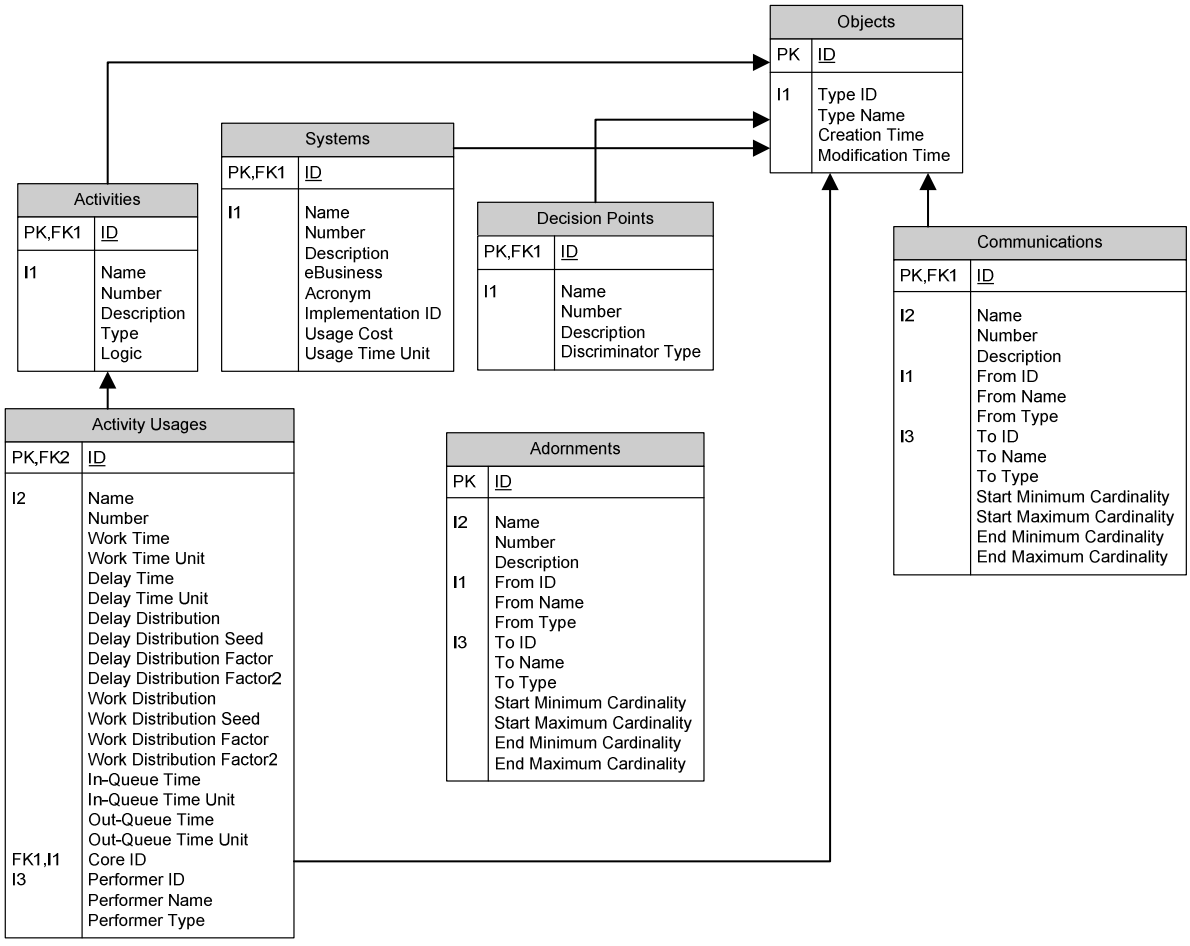
20 pav., sąsajų žymėjimas

Sąsajų S1.01 ir S1.02 žymėjimai reiškia, kad sąveikauja DB ir DD domenų komponentai, 01 ir 02 reiškia sąsajos eilės numerį.

4.5 Transformavimo algoritmas.

4.5.1. **Pradinių duomenų paruošimas**.

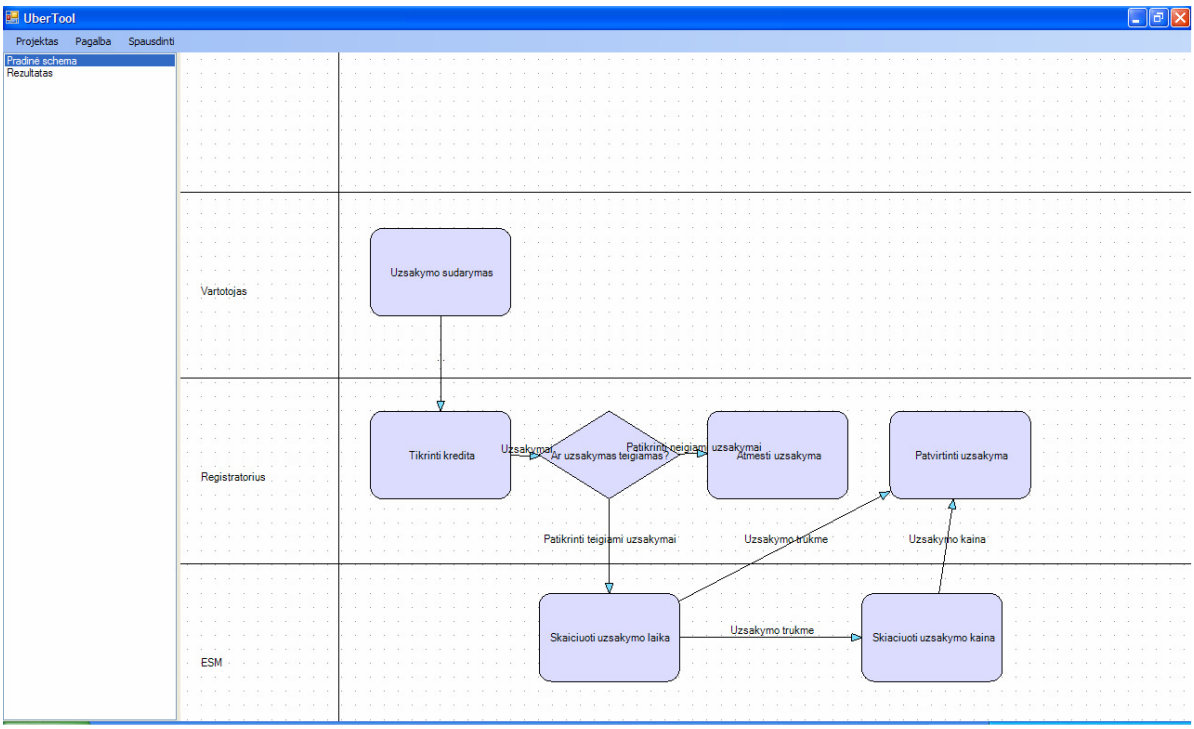
Pradiniams duomenis paruošti naudojama „ProVision Work Bech“ programa. (Pradinių duomenų pavyzdys: 8. pav. Darbų sekos modelis). Naudojantis šia programa aprašomi įmonėje vykstantys procesai sudarant darbų sekos diagramą. Gauta diagrama išeksportuojama į MS Access duomenų bazės failą. Tai yra pradiniai duomenys komponentinio transformavimo sistemai.



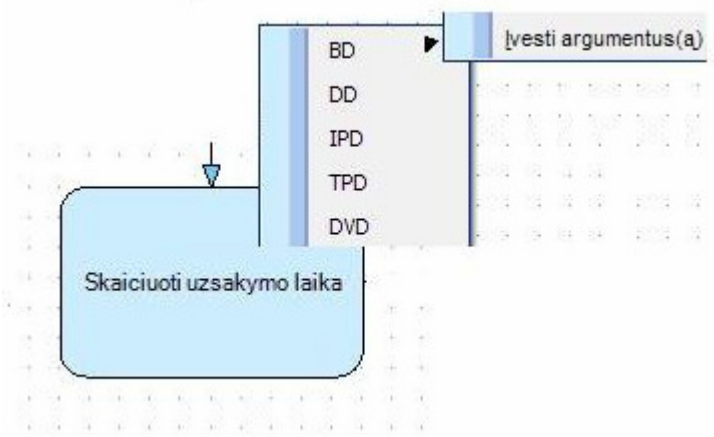
21 pav. Sugeneruota MS Access duomenų bazės schema

4.5.2. **Pradinių duomenų nuskaitymas**

Startavus programai atidarome prieš tai susikurtą duomenų bazės failą. Sukurta darbų sekos diagrama vaizduojama ekrane.



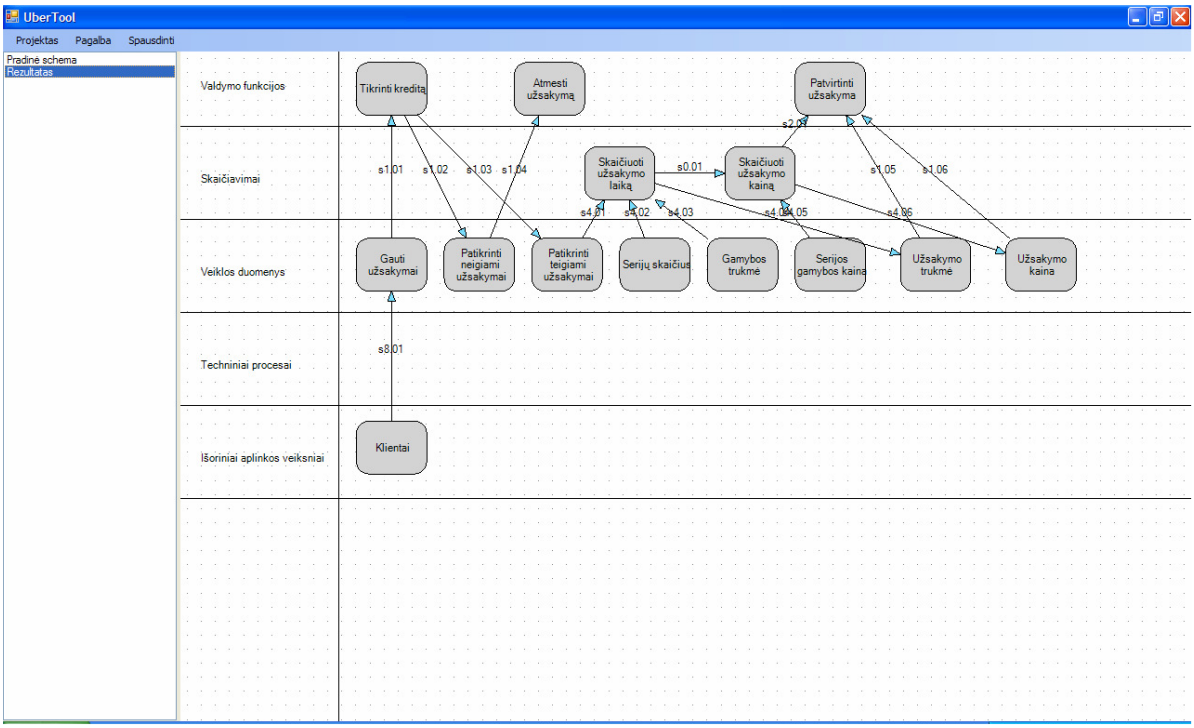
22 pav., programos veikimas, veiklos domenų priskyrimas



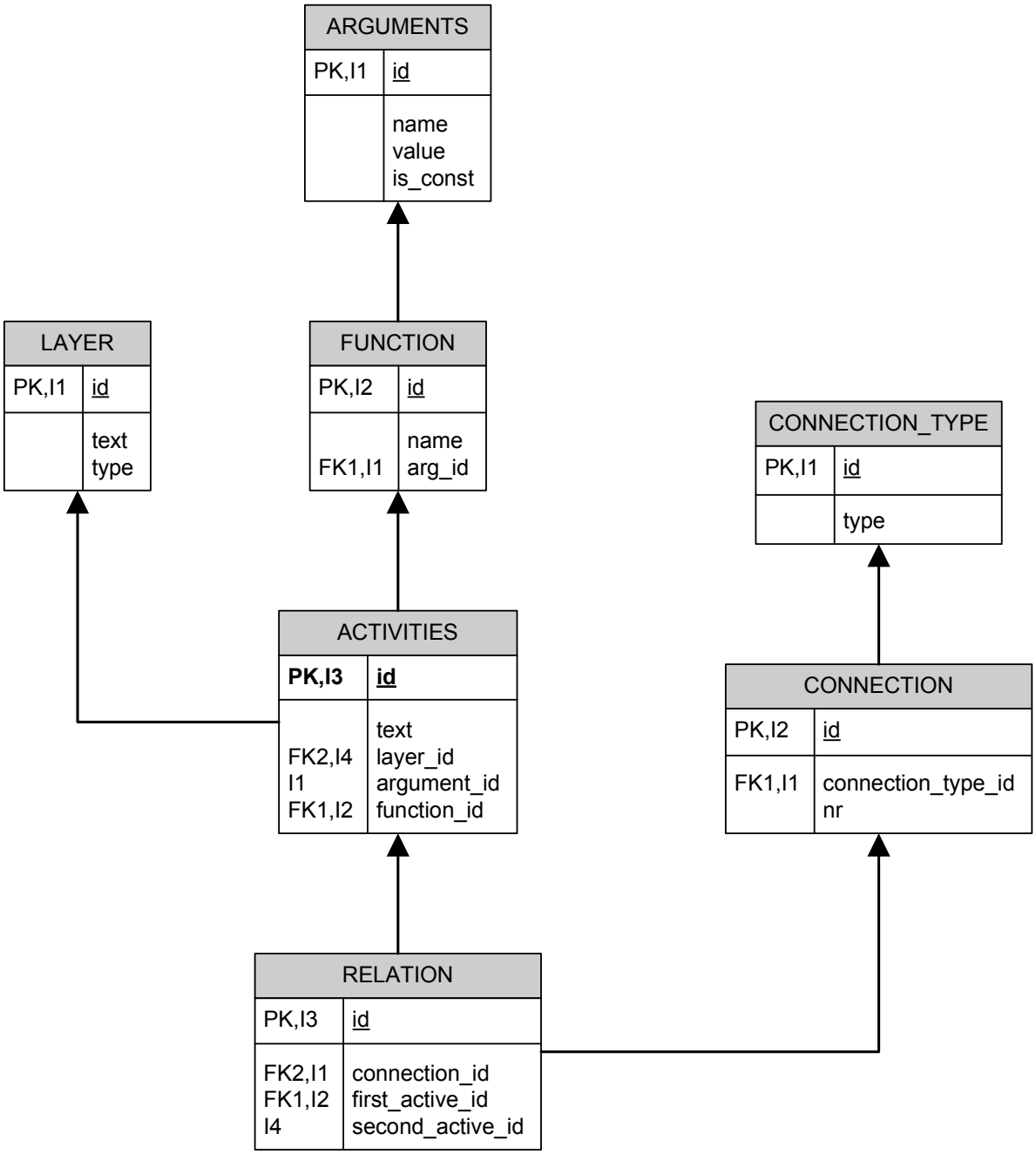
23 pav. Programos veikimas, argumentų įvedimas

4.5.3. Komponentinio sistemos modelio generavimas.

- Ekrane vaizduojamoje diagramoje prie objektų priskiriame vieną iš organizacijos veiklos domenų.
- Naudojame programos f-ją tikrinti sudarytą modelį. Generavimas ir modelio esančias taisykles. Šios f-jos pagalba sužinome sistemos objektus kurie nagali būti transformuojami į komponentinį sistemos modelį, projektuotojui neįvedus papildomos (trūkstamos) informacijos. Tranasformacija gali neveikti neįvedus argumentų
- Teisingai suvedus trūkstamą informaciją atliekame komponentinio sistemos modelio generavimą
- Rezultato išvedimas. Sugeneravus komponentinį sistemos modelį jis išsaugomas duomenų bazėje, gali būti atspausdintas ir eksportuotas į poveikslėlio tipo formatus.



23 pav. Programos veikimas, rezultato išvedimas



24 pav., rezultato duomenų bazė

ACTIVITIES:

- **id** - pirminis raktas, kodinis pavadinimas kreipiantis į lentelę;
- **text** – objekto pavadinimas;
- **layer_id** – argumento sluoksnio identifikatorius;
- **function_id** – transformavimo f-jos identifikatorius.

Lentelė skirta apibūdinti objektams komponentiniame sistemos modelyje

REALATION:

- **id** - pirminis raktas, kodinis pavadinimas kreipiantis į lentelę;
- **connection_id** – sąsajos identifikatius;
- **first_active_id** – „pirmo“ sąveikaujančio objekto identifikatius;
- **second_active_id** – „antro“ sąveikaujančio objekto identifikatius;

Lentelė skirta apibūdinti atskirų objektų sąveikai komponentiniame sistemos modelyje.

CONNECTION:

- **id** - pirminis raktas, kodinis pavadinimas kreipiantis į lentelę;
- **connection_type_id** – sąsajos identifikatius;
- **nr** – sąsajos numeris;

Lentelė skirta apibūdinti sąsajoms tarp obketų.

CONNECTION_TYPE:

- **id** - pirminis raktas, kodinis pavadinimas kreipiantis į lentelę;
- **type** – sąsajos tipas;

Lentelė skirta apibūdinti sąsajų tipams.

LAYER:

- **id** - pirminis raktas, kodinis pavadinimas kreipiantis į lentelę;
- **text** – sluoksnio pavadinimas;
- **type** – sluoksnio tipas;

Lentelė skirta pažymėti įmonės veiklos domenui.

FUNCTION:

- **id** - pirminis raktas, kodinis pavadinimas kreipiantis į lentelę;
- **name** – transformavimo f-jos pavadinimas;
- **arg_id** – argumentų identifokatorius;

ARGUMENTS:

- **id** - pirminis raktas, kodinis pavadinimas kreipiantis į lentelę;
- **name** – transformavimo f-jos pavadinimas;
- **value** – argumento reikšmė;
- **is_const** – konstantos reikšmė;

4.6 Trumpinimų aprašymai

3.lentelė, terminų žodynas

Trumpinimas	Angliškas pavadinimas	Lietuviškas paaiškinimas
MDA	Model-driven architecture	Modeliu grindžiama architektūra
DB		Verslo procesų domenas
DD		Informacijos domenas
IPD	Information processing domain	Technologinių procesų domenas
DVD	Online Analytical Processing	Darbo vietų domenas
CBC	Component based development	Komponentinis projektavimas
PIM	Platform independent model	Nepriklausomas nuo organizacijos veiklos sričių modelis
BPM	Business process model	Veiklos procesų modelis
EA	Enterprise architecture	Užklausų vykdymo kalba duomenų kubuose
PSM	Platform-specific model	
PIM	Platform-independent model	
Web		World Wide Web sutrumpinimas
Web		World Wide Web sutrumpinimas

5 Išvados

Integruotų kompiuterizuotų informacijos sistemų projektavime tikslinga apjungti IS kūrimą informacinės architektūros modelio pagrindu bei komponentinį IS projektavimą, siekiantį surinkti IS iš kompiuterizuotų veiklos komponentų.

Pasiūlytas metodas susieja IS architektūros modelį, darbų sekų modelį ir atvaizduoja juose esančią informaciją į naujo tipo modelį – komponentinį sistemos modelį, kuriame išskiriami tokio tipo komponentai: valdymo funkcijos (BD), skaičiavimai arba funkciniai komponentai (IPD), duomenų struktūros (DD), technologiniai procesai (TPD), išorinė aplinka (ENV).

Pagrindiniai komponentinio sistemos modelio sudarymo tikslai yra išsaugoti veiklos modelyje egzistuojančias sąsajas tarp IS informacinės architektūros komponentų bei tiksliau specifikuoti komponentus ir jų sąsajas. Toks modelis padėtų užtikrinti organizacijos veiklos ir visų projektuojamų sistemų integralumą, bei pakartotinį komponentų saugomų duomenų bazeje panaudojimą.

6 Naudota literatūra:

1. S.Ambler Architecture-driven modeling. – http://msdn/microsoft.com/developer/news/sdmag/arcmodel.htm 1999
2. S.Gudas The Information Architecture Framework for Enterprise Integration. – Databases&Information Systems. Proceedings of the 4th IEEE International Baltic Workshop. Vol.12. Vilnius, Technika, 2000, p.168-175.
3. S.Gudas IS kūrimas veiklos informacinės architektūros pagrindu. - Konferencijos pranešimų medžiaga "Integruotos projektavimo sistemos", Kaunas, Technologija, 2000, 4 -14p.
4. E. Turban, E.McLean, J.Wetherbe Information technology for management. – John Wiley&Son.Inc., 1999.
5. Framework for Managing Process Improvement, Department of Defense, 1994, http://www.dtic.mil/c3i/bprcd/3003sb.htm
6. The Requirements for a Component-based Architecture, Jochen Rütschlin, DaimlerChrysler AG, Research and Technology. http://www.informatik.uni-stuttgart.de/ipvr/as/personen/ruetschlin/publications/ProSTEPScienceDays2000.pdf
7. Component-Based Architecture Development, Paull C. Barr. http://www.stc-online.org/cd-rom/1999/slides/SofComp8.pdf
8. *UML Components* by John Cheesman and John Daniels, Addison-Wesley. 2000. http://www.umlcomponents.com
9. Desmond D'Souza and Allan Wills. *Objects, Components, and Frameworks with UML: The Catalysis Approach*. to appear, http://www.iconcomp.com/catalysis
10. Brown, Alan W. "Preface: Foundations for Component-Based Software Engineering," vii-x. *Component-Based Software Engineering: Selected Papers from the Software Engineering Institute.* Los Alamitos, CA: IEEE Computer Society Press, 1996.
11. Clements, Paul C. "From Subroutines to Subsystems: Component-Based Software Development," 3-6. *Component-Based Software Engineering:*

Selected Papers from the Software Engineering Institute. Los Alamitos, CA: IEEE Computer Society Press, 1996