VILNIAUS UNIVERSITETAS
KAUNO HUMANITARINIS FAKULTETAS

INFORMATIKOS KATEDRA

Verslo informatikos studijų programa
Kodas 62109P101

MANTAS BIKAUSKAS

MAGISTRO BAIGIAMASIS DARBAS

UML BŪSENŲ MODELIO GENERAVIMAS VEIKLOS ŽINIŲ BAZĖS PAGRINDU

Kaunas 2008

VILNIAUS UNIVERSITETAS
KAUNO HUMANITARINIS FAKULTETAS

INFORMATIKOS KATEDRA

MANTAS BIKAUSKAS

MAGISTRO BAIGIAMASIS DARBAS

UML BŪSENŲ MODELIO GENERAVIMAS VEIKLOS ŽINIŲ BAZĖS PAGRINDU

Leidžiama ginti _____

Magistrantas _____
(parašas)

Darbo vadovas_____
(parašas)

_____
(darbo vadovo mokslinis laipsnis, mokslo pedagoginis vardas, vardas ir pavardė)

Darbo įteikimo data_____

Registracijos Nr._____

Kaunas 2008

TURINYS

SANTRUMPŲ SĄRAŠAS

UML – unified modeling language

IT – informacinės technologijos

IS – informacinės sistemos

VM – Veiklos modelis

SD – būsenų diagrama

DB – duomenų bazė

PAVEIKSLŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

BIKAUSKAS, Mantas. (2008) *Generation UML state modelį based on enterprise metamodel*. MBA Graduation Paper. Kaunas: Vilnius University, Kaunas Faculty of Humanities, Department of Informatics. 58 p.

SUMMARY

Recently project models of IS engineering are created empirically, officially not exanimate according to real subject field. Target of this work is to intellectualize IS engineering designing stage by generating UML state model from subject field knowledge base namely called enterprise model.

The point of this work is to generate UML state model based on enterprise model, performing analysis of enterprise model and excluding enterprise model parameters matching UML state model elements and adding missing elements to enterprise metamodel knowledge base.

In this work UML state metamodel generation is performed. Generated state metamodel will help to intellectualize designing stage. It is important that UML state model generation will be performed according to enterprise model – database of real processes in the organization.

Tasks of this work: perform analysis of enterprise model composition UML state model aspect, to create UML state model generation based on enterprise model, suggest software solution for created method.

In this research it was discovered that state model generation intellectualize IS engineering designing stage. After analyzing enterprise metamodel and state metamodel was found that we need to add flow object in enterprise metamodel. Created UML state generation algorithm was tested with real data and was suggested realization prototype.

ĮVADAS

Tema: UML būsenų (state) modelio generavimas veiklos žinių bazės pagrindu

Darbo problema: Šiuo metu IS inžinerijoje projektiniai modeliai yra kuriami empiriškai, formaliai jų nepatikrinus pagal realią dalykinę sritį. Šiuo darbu siekiama intelektualizuoti IS inžinerijos projektavimo etapą generuojant UML state modelį iš dalykinės srities žinių saugyklos vadinamos veiklos modeliu.

Darbo esmė – sugeneruoti veiklos modeliu grindžiamo UML state modelį, atliekant veiklos modelio analizę ir išskiriant veiklos modelio parametrus atitinkančius UML state modelio elementus, papildyti veiklos metamodelio žinių bazę trūkstamais elementais.

Šiame darbe atliekamas UML (angl. unified modelling language) state metamodelio generavimas. Sugeneruoto state metamodelio pagalba bus intelekualizuojamas projektavimo etapas padedantis identifikuoti problemas kylančias IS kūrimo metu bei ir optimizuoti patį kūrimo procesą. Reikalinga akcentuoti tai, kad UML state modelio generavimas bus vykdomas, remiantis veiklos modeliu – realių įmonėje vykstančių procesų saugykla.

Darbo objektas: žiniomis grindžiamos IS inžinerijos projektavimo etapas.

Darbo tikslas: Intelektualizuoti IS inžinerijos projektavimo etapą, taikant veiklos žinių pagrindu generuojamą būsenų modelio būdą.

Darbo uždaviniai:

- Atlikti veiklos modelio sudėties analizę UML state modelio aspektu.
- Sukurti veiklos modeliu grindžiamą UML state modelio generavimo būdą (algoritmą)
- Pasiūlyti sukurto metodo programinės realizacijos prototipą

Rašant magistro darbą bus naudojami šie **tyrimo metodai:**

- Bendrieji mokslinio tyrimo metodai:
  - ✓ Palyginimas;
  - ✓ Indukcija (darant išvadas);
  - ✓ Dedukcija (pereinant nuo bendrų sprendimų prie atskirų dalių);
- Konkrečios mokslo šakos metodai:
  - ✓ Analizės metodas (analizės dalyje);
  - ✓ Dokumentų analizės metodas – (renkant pirminius duomenis);
  - ✓ Apibendrinimo metodas (bendrų savybių ir požymių nusakymui).
- Empiriniai metodai:
  - ✓ Stebėjimas (pirminės informacijos rinkimui);
  - ✓ Pokalbio metodas (gilinant ir patikslinant gautą informaciją).

Darbo struktūra: magistro darbas susideda iš analitinės dalies, siūlomo sprendimo metodikos ir eksperimentinės dalies.

Analitinėje dalyje aptariama pirmieji IS inžinerijos raidos etapai, žiniomis grindžiamos informacijos sistemų inžinerijos principai. Apžvelgiami jau egzistuojantys veiklos metamodeliai. Siūlomo sprendimo metodikos dalyje analizuojamas veiklos metamodelis ir būsenų (state) diagramos formalaus modelio generavimas. Eksperimentinėje dalyje analizuojama veiklos metamodelio ir būsenų (state) diagramos metamodelio sąsaja. Algoritmo būsenų diagramai generuoti sukūrimas. Eksperimento patvirtinimas realiais duomenimis, bei algoritmą realizuojančio įrankio prototipo pasiūlymas.

Darbą sudaro 58 puslapiai, 5 lentelės, 41 paveikslėliai, 2 priedai

1. ANALITINIS SKYRIUS

1.1. Pirmieji IS inžinerijos raidos etapai

Pirmuoju etapu IS inžinerijos, veiklos modeliavimo ir kitų sričių metodai ir priemonės vystosi praktiškai nepriklausomai. Šiuo laikotarpiu ir IS inžinerija dar nėra vientisa sritis, ją sudaro trys sąlyginai savarankiškai plėtojamos kryptys – duomenų bazių, programinės įrangos ir reikalavimų (informacijos sistemai) inžinerijos.

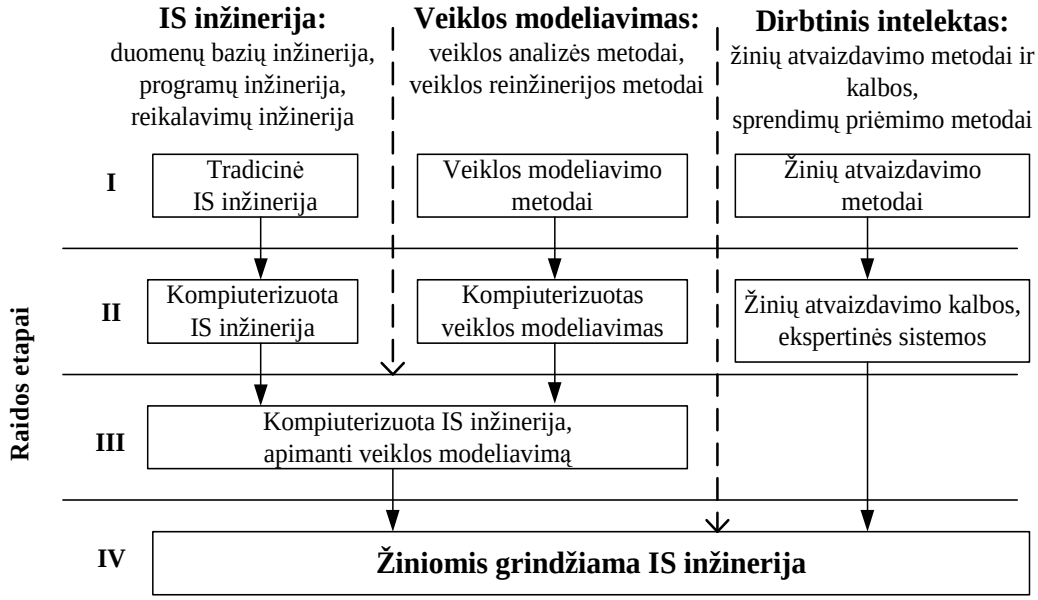
Duomenų bazių inžinerija plėtoja duomenų modeliavimo metodus ir priemones, duomenų bazių valdymo sistemų standartus. Programų inžinerija kuria programinės įrangos konstravimo aplinkas (programavimo kalbas ir priemones) ir taikomųjų programų kodo kūrimo metodus (struktūrinis programavimas, modulinis programavimas) (Laudon ir kt. 2000)**.** Reikalavimų inžinerija įgyvendina kertinį taikomųjų IS kūrimo principą — teikia metodus ir priemones IS vartotojo poreikiams analizuoti, reikalavimams aprašyti, būsimosios IS reikalavimams specifikuoti (Boman ir kt., 1997).

Veiklos modeliavimo srityje atsiranda įvairių organizacijų veiklos procesų ir funkcijų aprašymo metodų (GRAI, TOVE, PERA, CIM-OSA), sudaromas svarbus organizacijų projektavimo ir veiklos modeliavimo standartas GERA (Schekkerman, 2003)

IS kūrimas ir veiklos modeliavimas vykdomi nekompiuterizuotomis arba iš dalies kompiuterizuotomis priemonėmis, šiuo laikotarpiu dar nėra integruotų IS kūrimo aplinkų — CASE sistemų.

Dirbtinio intelekto metodai (žinių atvaizdavimo metodai, sprendimų logikos modeliavimas) ir priemonės atsiranda ir formuojasi kaip santykinai savarankiška sritis. Greta plėtojasi pragmatiškesnė sprendimų priėmimo procesų modeliavimo ir kompiuterizavimo sritis — ekspertinių sistemų teorija ir priemonės, sprendimų paramos sistemos (Laudon ir kt., 2000)**.**

Antruoju IS inžinerijos raidos etapu kompiuterizuojami IS inžinerijos metodai ir veiklos modeliavimo metodai. IS kuriamos pasitelkiant kompiuterizuotus įrankius — CASE sistemas, kurios apima dalį IS kūrimo gyvavimo ciklo – projektavimo, dokumentavimo ir kodavimo etapus. Duomenų bazių inžinerija visiškai integruojama CASE aplinkose. Veiklos modeliavimas, dirbtinio intelekto metodai ir priemonės išlieka atsieti nuo IS inžinerijos [1].



Šaltinis: GUDAS S., LOPATA A. (2005) Žiniomis grindžiamos informacijos sistemų inžinerijos bruožai.

1 pav. Pagrindiniai IS inžinerijos raidos etapai.

1.2. Žiniomis grindžiamos informacijos sistemų inžinerijos principai

Žiniomis grindžiamos IS inžinerijos principai 1–5, kurie apibrėžia kokybinius žiniomis grindžiamų IS inžinerijos metodų ir CASE sistemų ypatumus.

1. Informacijos sistemų teorinė paskirtis yra kompiuterizuoti veiklos srities dėsningumą. IS inžinerijos objektas yra organizacijų veiklos informaciniai poreikiai. Organizacijų veiklos informacinius poreikius formuoja vykdomos veiklos dėsningumai. Todėl IS inžinerijoje analizuojama ir kompiuterizuojama realybės sritis turi būti apibrėžta ir formalizuotai aprašyta kaip tam tikras dėsningumas, išreiškiantis organizacijos veiklos esmė. Organizacijos veiklos esmines savybes, būtinas IS inžinerijos procesui vykdyti, turi aprašyti adekvatus šiai paskirčiai veiklos modelis – formalizuotas veiklos srities (teorinis) modelis, kuris išreiškia veiklos srities dėsningumą.[1]

2. Veiklos modelio paskirtis IS inžinerijoje yra specifikuoti veiklos srities dėsningumą – veiklos valdymo sistemos komponentus ir jų informacines sąveikas (identifikuoti būtinus ir pakankamus IS inžinerijai veiklos komponentus). Organizacijos veiklos teorinio modelio paskirtis – specifikuoti veiklos srities dėsningumą. Valdymo teorijos požiūriu esminis organizacijos veiklos dėsningumas yra veiklos valdymo procesas. Veiklos teorinio modelio sudėtis ir elgsena apibrėžiami atliekant veiklos analizę valdymo aspektu (Gudas, Skersys, Lopata, 2004). Šiuo metu IS inžinerijoje naudojamas koncepcinis veiklos modeliavimas. Koncepcinis veiklos modelis (probleminės srities modelis, domeno modelis) yra reikalavimų inžinerijos pagrindas. Esminis skirtumas tarp formalizuoto veiklos modelio ir koncepcinio veiklos modelio yra tas, kad koncepcinis veiklos modelis yra empirinis savo sudėtimi ir sudarymo eiga, o formalizuotas veiklos modelis yra

objektyvizuotas savo sudėtimi ir sudarymo eiga, kadangi grindžiamas veiklos srities dėsningumu. Veiklos teorinis modelis (žiniomis grindžiamos IS inžinerijos poreikiams) yra informacinių valdymo procesų modelis, identifikuojantis veiklos komponentus (materialius veiklos elementus) ir jų sąveikas, veiklos valdymo komponentus (informacinius veiklos elementus: duomenų, žinių ir tikslų struktūras) ir jų sąveikas (statines ir dinamines organizacinės sistemos savybes). Veiklos teorinis modelis yra formalizuotas veiklos modelis [1].

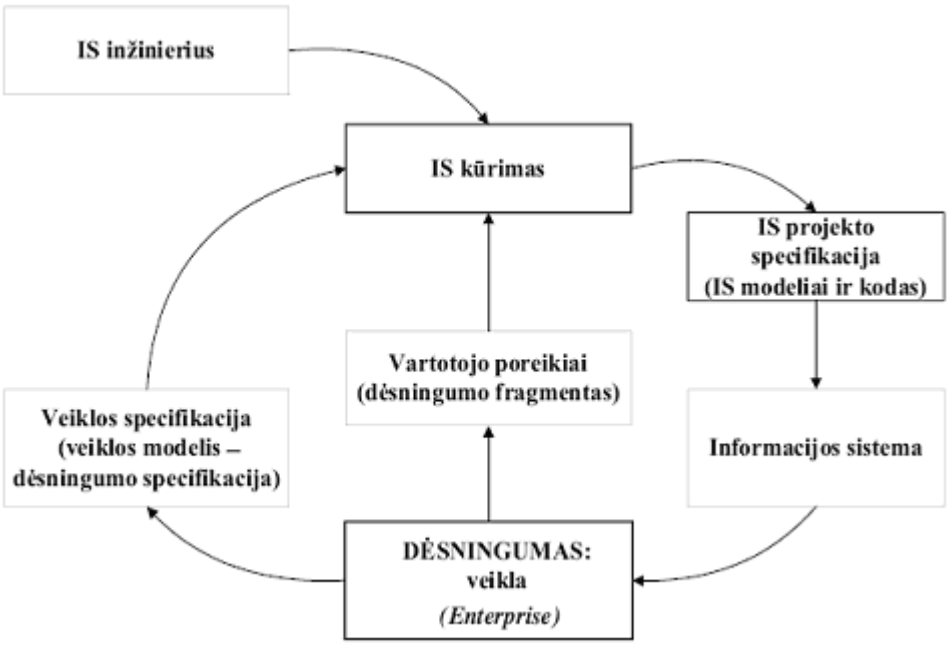
3. Žiniomis grindžiamų CASE metodų paskirtis yra apibrėžti IS inžinerijos procesą formalizuoto veiklos modelio specifikacijos – veiklos metamodelio pagrindu. Veiklos teorinis modelis apibrėžia veiklos metamodelio sudėtį – būtinus elementų tipus ir jų sąveikų tipus, kurie turi būti įvertinti praktiniams IS kūrimo žingsniams skirtuose (konkrečia notacija sudaromuose) veiklos modeliuose. Žiniomis grindžiamas IS inžinerijos procesas kiekviename IS gyvavimo ciklo etape naudoja veiklos metamodelį kaip apribojimų šaltinį, kuris teikia kriterijus verifikuojant gaunamos apie veiklos sritį informacijos turinį. Pirmuoju IS inžinerijos etapu sudaromas formalizuotas konkrečios veiklos modelis, atitinkantis veiklos metamodelio sudėtį, t. y. Veiklos metamodelio specifikacijos apribojimus elementų tipams ir jų tarpusavio ryšiams [1].

4. Žiniomis grindžiamos CASE sistemos būtini komponentai apibrėžiami įvertinant žiniomis grindžiamo CASE metodo būtinų komponentų sudėtį, t. y. veiklos metamodelį ir formalizuotą veiklos modelį. Žiniomis grindžiamą CASE sistemą turi sudaryti tokie veiklos žinias organizuojantys komponentai: 1. CASE sistemos žinių posistemis, kurio paskirtis – saugoti veiklos modelio specifikaciją ir ją naudoti kaip žinių apie konkrečią veiklos sritį šaltinį, teikti žinias apie veiklos sritį aprašantį formalizuotą veiklos modelį kiekvienam IS kūrimo gyvavimo ciklo etapui. Žinių posistemio branduolys yra veiklos žinių bazė. 2. CASE sistemos žinių bazė, kurios paskirtis yra saugoti teorinio veiklos modelio (veiklos metamodelio) specifikaciją ir formalizuotą konkrečios veiklos srities modelį [1].

5. Žiniomis grindžiamos CASE sistemos paskirtis – verifikuoti IS projektinius sprendimus saugykloje laikomų žinių atžvilgiu. CASE sistemos žinių bazėje saugomas formalizuotas veiklos modelis identifikuoja konkrečios veiklos (organizacijos) valdymo informacinių procesų sąveikas, jos yra specifikuotos, prieš tai patikrinus veiklos metamodelio (t. y. formalizuotų kriterijų) atžvilgiu. Taip objektyvizuojamas IS projekto verifikavimas remiantis CASE sistemoje saugomomis veiklos žiniomis, skirtingai nuo tradicinių CASE metodų, kurie grindžiami vien žmogaus (vartotojo, analitiko) teikiama informacija (išorinis modeliavimas). Informacijos sistemų inžinerija, tenkinanti [1].

1–5 principus, vadinama žiniomis grindžiama IS inžinerija (2 pav.). Vartotojas yra tarpininkas tarp veiklos proceso (dėsningumo) ir IS inžinieriaus, kuris dėl objektyvių priežasčių pajėgus išreikšti savo atsakomybės srities diktuojamą nuomonę – savo reikalavimus būsimajai IS,

kurie apibūdina šio konkretaus vartotojo poreikius – pareigų ir teisių sritį. Iš čia gaunama išvada, kad veiklos visumos neapima nė vienas atskiras vartotojas, nes kiekvienas turi ribotus savo atsakomybės srities diktuojamus poreikius. Veiklos visumą apima veiklos specifikacija (veiklos modelis), nes jis specifikuoja veiklos dėsningumą, kuris išreiškia esmines organizacijos veiklos savybes. Todėl IS inžinerija turėtų būti grindžiama specifiniu veiklos modeliu – veiklos žinių modeliu, kuris pajėgus specifikuoti veiklos dėsningumą. Toks veiklos žinių modelis turi būti pakankamai formalizuotas (susistemintas), grindžiamas organizacijų veiklos valdymo teoriniais principais, siejamais su valdymo procesų automatinio reguliavimo teorija [1].



Šaltinis: LOPATA A.; GUDAS S. (2006) Informacijos mokslai.

2 pav. Informacijos sistemų inžinerija veiklos žinių pagrindu

1.3. Egzistuojančių Veiklos metamodelių tyrimas

Šiuo metu yra keletas skirtingų organizacijos modeliavimo metodų ir kalbų IDEF, OMT, UML, CIMOSA, ARIS ir kt. Tai sąlygojo tarpusavyje nesuderinamų programinių paketų, skirtų organizacijos veiklos modeliavimui atsiradimą (ARIS ToolSet, System Architect, FirstSTEP, CimTool ir kt.). Pagrindinės organizacijos modeliavimo mokslinės grupės, tokios kaip ODP, OMG, PSL/NIST, didžiausi programinės įrangos gamintojai (Oracle, Microsoft) bei pagrindinės standartizacijos organizacijos (ISO, CEN) stengiasi sukurti bendrą standartą, kuriuo remiantis būtų kuriami nauji organizacijos veiklos modeliavimo metodai, kalbos bei su jais suderinta programinė įranga. Pagrindiniai organizacijos veiklos modeliavimo standartai pateikti 1 lentelėje [2].

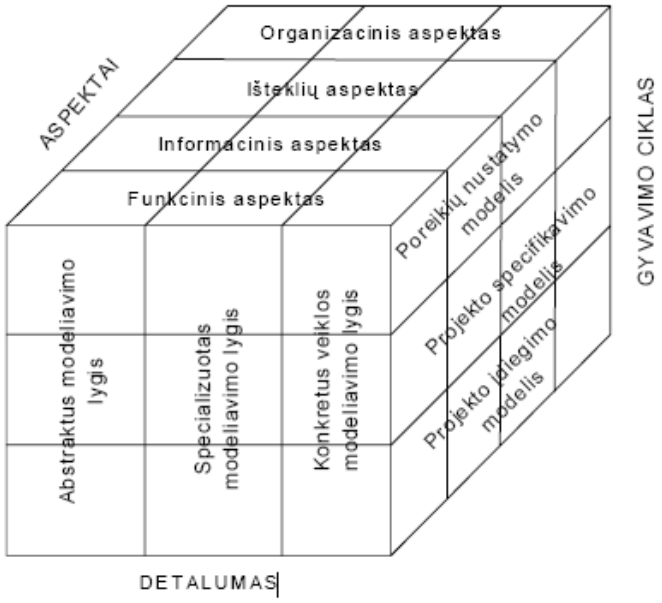
1 lentelė

Pagrindiniai organizacijos veiklos modeliavimo standartai

Standarto kodas	Standarto pavadinimas
ISO 14258	Concepts and Rules for Enterprise Models
ISO 15704	requirements for Enterprise reference architectures and methodologies
ISO 10314	Shop floor production model
ISO/IEC 15288	System life cycle processes

Šaltinis: sudaryta autoriaus.

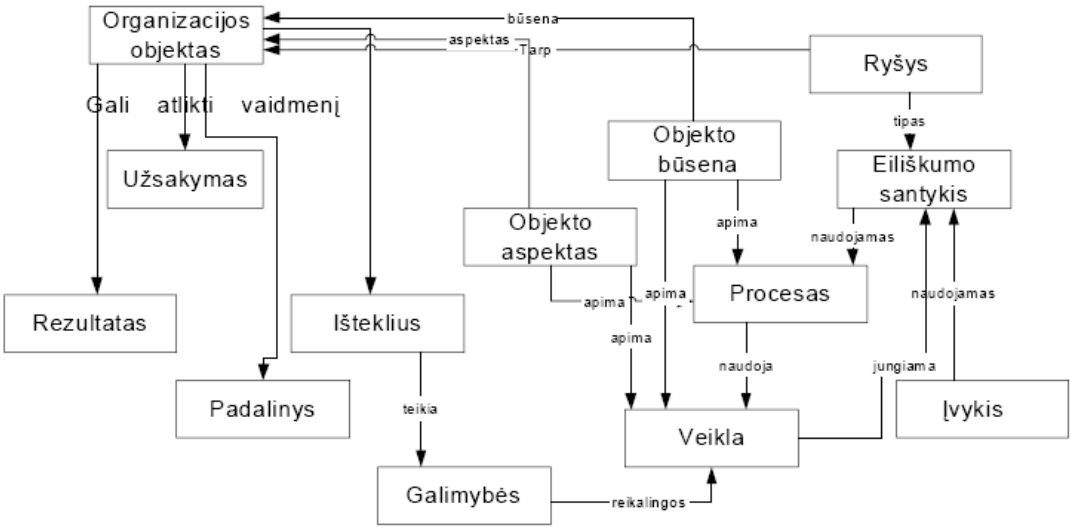
Europos standartizacijos komitetas (CEN), bendradarbiaudamas su tarptautine standartizacijos organizacija (ISO), remdamasis 1 lentelėje pateiktais standartais sukūrė CEN ENV 40003 ir CEN ENV 12 204 standartus, kuriuose apibrėžti pagrindiniai organizacijos veiklos modeliavimo principai. CEN ENV 40003 standartas yra sukurtas CIMOSA [3] modeliavimo metodo pagrindu. CEN ENV 40003 standarte veiklos modelio projektavimo procesas pateikiamas kaip kubas, kurio ašys aprašo modeliavimo aspektus, projektavimo gyvavimo ciklo etapus bei modelio detalumo lygius [2]. (3 pav.)



Šaltinis: LOPATA A. (2006) Veiklos modelių sudėties analizė

3 pav. CEN ENV 4003 standarte naudojama principinė veiklos modeliavimo schema

CEN ENV 40003 standarte veiklos modeliavimui būtinas sudėtines dalis (konstruktus) apibrėžia CEN ENV 12204 standartas (4 pav.):



Šaltinis: LOPATA A. (2006) Veiklos modelių sudėties analizė.

4 pav. Organizacijos modeliavimui būtini pagrindiniai konstruktai, apibrėžti CEN ENV 12204 standarte

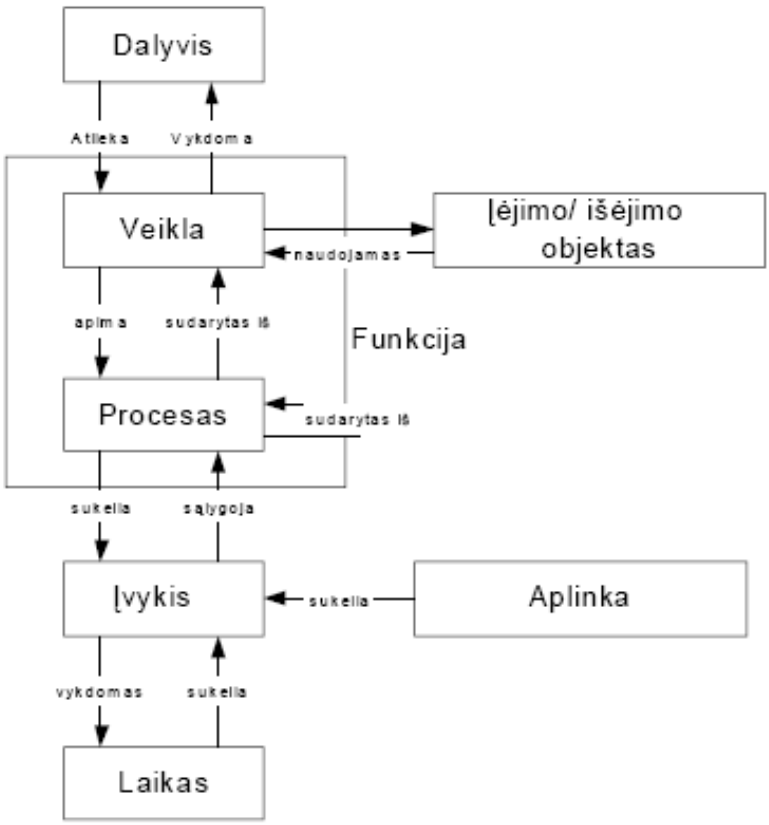
Tačiau, kaip parodė praktika CEN ENV 40003 ir CEN ENV 12204 standartai pilnai nepatenkino nei organizacijos veiklos projektuotojų nei programinės įrangos gamintojų poreikių, todėl šiuo metu yra kuriamos naujos šių standartų versijos, o taip pat kuriami nauji standartai (2 lentelė), bei kalbos, tokios kaip UEML [4] (Unified Enterprise Modeling Language) [2] (5 pav.).

2 lentelė

Šiuo metu kuriami su organizacijos modeliavimu susiję standartai

Standarto santrumpa	**Standarto pavadinimas**
UEML	Universal Enterprise Modeling Language
PSL	Process Specification Language
XBRL	Extensible Business Reporting Language

Šaltinis: sudaryta autoriaus.

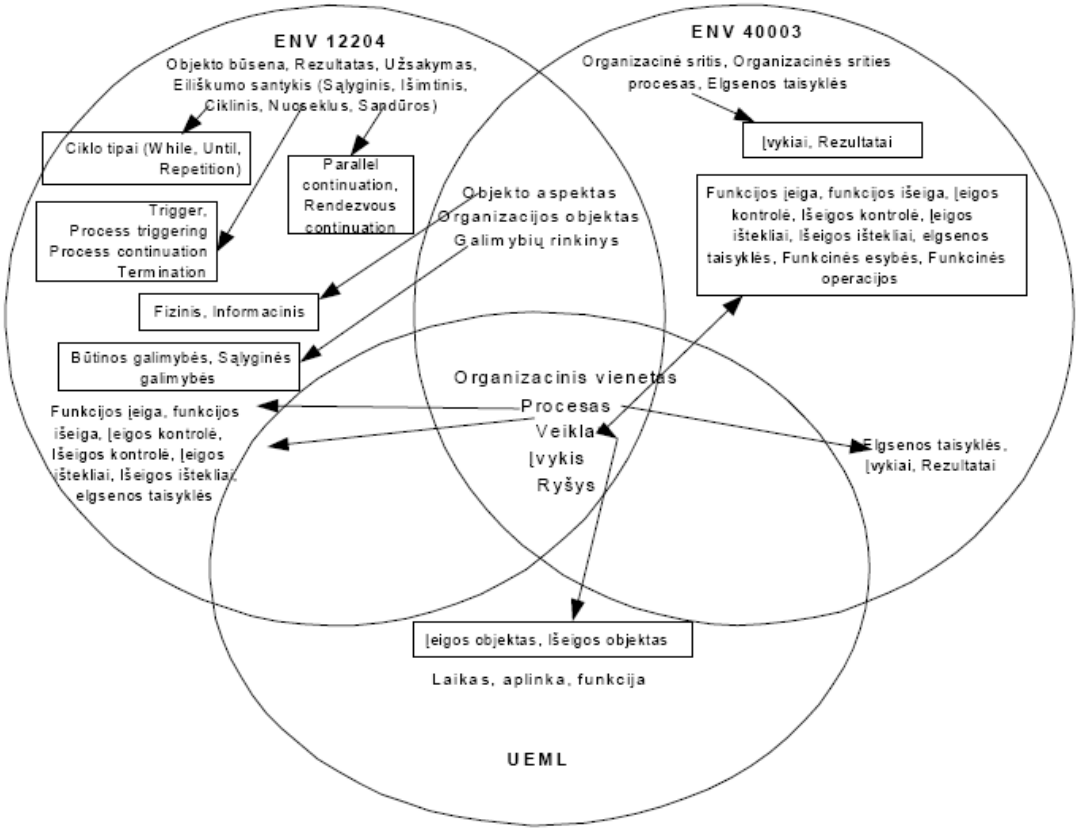


Šaltinis: LOPATA A. (2006) Veiklos modelių sudėties analizė

5 pav. UEML principinė schema

1.4. Veiklos modelių sudėties analizė

Atlikus palyginamąją CEN ENV 40003, CEN ENV 12204, ir UEML standartų analizę išskiriamos pagrindinės bei papildomos sudėtinės dalys (konstruktai) būtinos organizacijos veiklos modeliavimui (6 pav.)



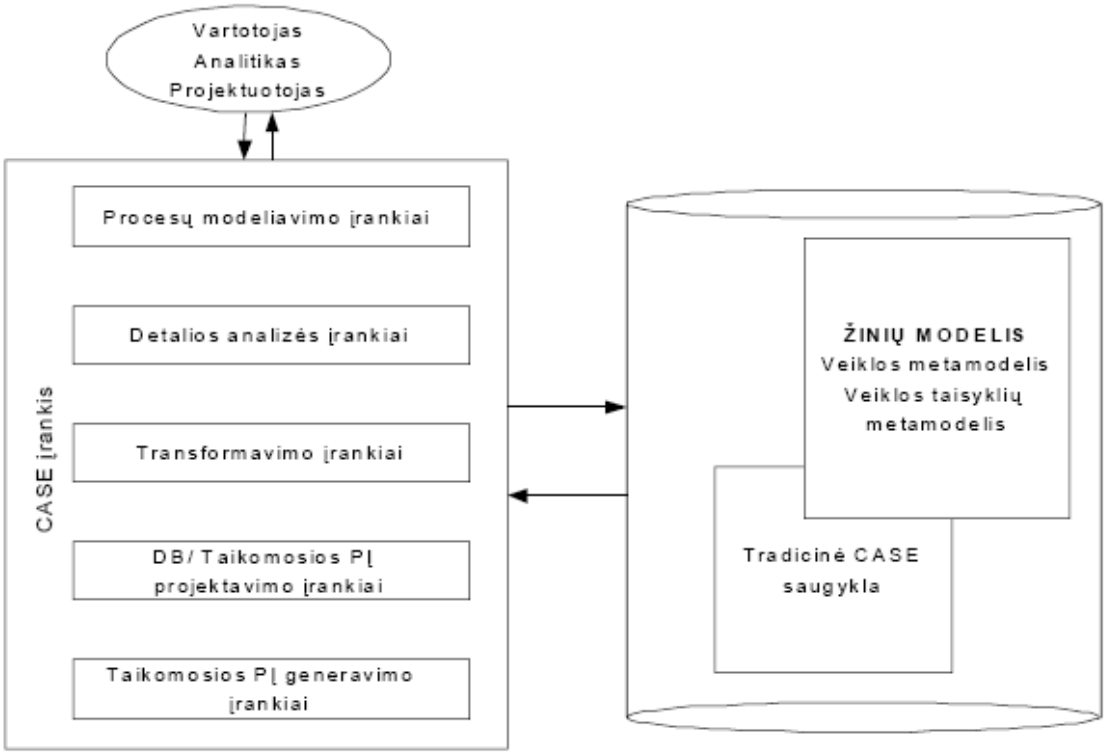
Šaltinis: LOPATA A. (2006) Veiklos modelių sudėties analizė.

6 pav. ENV 12204, ENV 40003 ir UEML standartuose išskirtų pagrindinių konstruktų palyginimas

Nei vienas iš nagrinėjamų standartų pilnai neatitinka organizacinės sistemos projektuotojo poreikių ne tik dėl riboto konstruktų kiekio ir funkcijų, bet ir todėl, kad nei viename standarte nėra apibrėžta veiklos modelio struktūra bei jos santykis su pagrindiniais konstruktais. Veiklos modelis sukurtas remiantis Porter vertės grandinės ir J.Henderson modelių pagrindu. Dalis veiklos modelio komponentų (Įvykis, Procesas, Ryšys, Organizacinis vienetas), atitinka bendrus bei dalinai bendrus (Funkcija) standartų konstruktus. Nagrinėjamuose standartuose nėra detalizuota kaip turėtų būti apibrėžiamos veiklos modelio veiklos taisyklės.

Projektuotojui, kuriančiam organizacijos informacinės sistemos projektą, būtų patogu naudotis ne tik projektavimui reikalinga informacija, esančia tradicinėje CASE

įrankio saugykloje, bet ir žinių modelio saugykla, kurioje saugomi organizacijos veikloje egzistuojančių veiklos taisyklių bei organizacijos veiklos metamodeliai.[2] (7 pav.).



Šaltinis: LOPATA A.(2006) Veiklos modelių sudėties analizė.

7 pav. Veiklos taisyklių saugyklos vieta CASE įrankyje

Organizacijos funkcinių IS projektavimo kompiuterizuotai sistemai (CASE sistemai) teikia papildomą informaciją tokios veiklos modelio klasių sąsajos:

- Pagal funkcijos pavadinimą gali būti generuojamas susietų su šia funkcija procesų sąrašas, pateikiama kiekvieno proceso sudėtis,
- Pagal funkcijos pavadinimą arba proceso pavadinimą gali būti generuojamas susietų padalinių (vartotojų) sąrašas;
- Pagal funkcijos pavadinimą gali būti generuojama darbų sekų modelio, taip pat vartotojo taikomųjų uždavinių modelio (*use case model*) pradinė sudėtis;

- Pagal funkcijos pavadinimą gali būti generuojamas esybių ryšių diagramos (arba klasių modelio) pradinė sudėtis.
- Aptarėme tik dalį galimybių, kurias teikia organizacijos veiklos taisyklių bei CASE įrankio duomenų saugyklos informacinės sistemos kūrimo procese [2].

1.5. UML apžvalga ir Būsenų modelio aprašymas

Pats savaime UML nėra metodas; tačiau jis sukurtas taip, kad derėtų su pagrindiniais tuo metu egzistavusiais į objektus orientuotais programinės įrangos kūrimo metodais (pavyzdžiui, OMT, Booch, Objectory). Nuo to laiko, kai išsivystė UML, kai kurie šių metodų buvo perdaryti, pasinaudojant naujos notacijos (pavyzdžiui, OMT) pranašumais, ir UML pagrindu buvo sukurti nauji metodai. Geriausiai žinomas Rational Unified Process (RUP). Yra daug kitų UML paremtų metodų, tokių kaip abstrakcijos metodas, Dynamic Systems Development Method (dinaminių sistemų kūrimo metodas) ir kt., skirti vystyti specialiems sprendimams arba kitokiems tikslams[11].

1.5.1. UML apžvalga

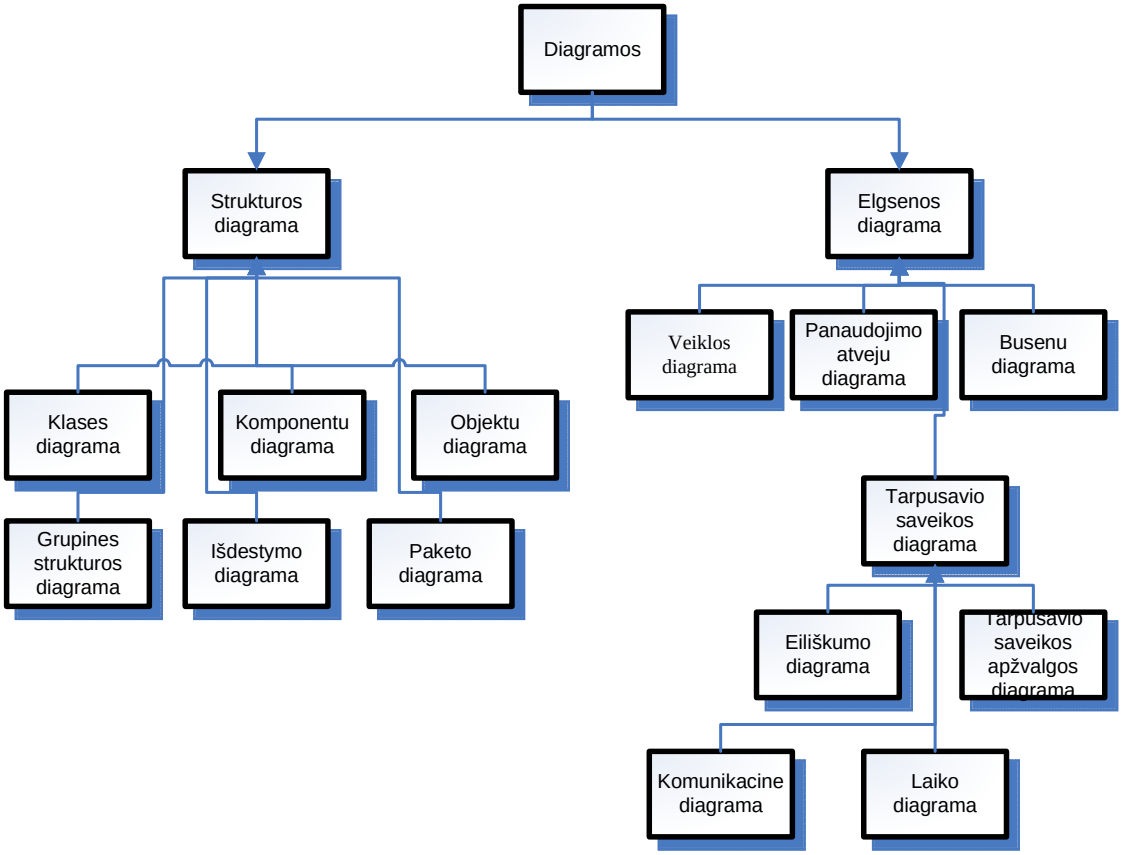
Labai svarbu nepainioti UML modelio ir sistemos diagramų komplekto. Diagrama dalinai grafiškai atvaizduoja sistemos modelį. Modelyje taip pat yra „semantinė jungiamoji plokštė“ — tokie dokumentai, kaip raštiški naudojimo scenarijai, valdantys modelio elementus ir diagramas.

UML diagramos vaizduoja tris skirtingus sistemos modelio vaizdus:

- Funkcinių reikalavimų vaizdas - pabrėžia funkcinius sistemos reikalavimus vartotojo požiūriu. Apima naudojimo scenarijų diagramas (use case diagrams).
- Statinės struktūros vaizdas - pabrėžia statinę sistemos struktūrą naudojant objektus,atributus, operacijas, ir tarpusavio santykius. Apima klasės diagramas (class diagrams) ir grupinės struktūros diagramas (composite structure diagrams).
- Dinaminio režimo vaizdas - pabrėžia sistemos dinaminę elgseną, parodydami tarpusavio sąveikas tarp objektų, ir vidinius objektų būsenų pokyčius. Apima eiliškumo diagramas (sequence diagrams), veiklos diagramas (activity diagrams) ir mašinos būsenos (state machine diagrams).

UML modeliais galima keistis tarp UML įrankių, naudojant XMI tarpusavio apsikeitimo formatą [5].

UML 2.0 yra 13 tipų diagramų. Norint jas perprasti, verta jas surūšiuoti hierarchiškai, kaip parodyta hierarchinėje schemoje dešinėje.



Šaltinis: sudaryta autoriaus.

8 pav. UML 2.0 diagramos

Struktūros diagramos parodo, kokie dalykai turi būti modeliuojamoje sistemoje:

- Klasės diagrama
- Komponentų diagrama
- Grupinės struktūros diagrama
- Išdėstymo diagrama
- Objekto diagrama
- Paketo diagrama

Elgsenos diagramos parodo, kas turi vykti modeliuojamoje sistemoje:

- Veiklos diagrama
- Mašinos būsenos diagrama
- Naudojimo scenarijų diagrama

Tarpusavio sąveikos diagramos, elgsenos diagramų posakis, parodo valdymo ir duomenų srautą tarp modeliuojamos sistemos elementų:

- Komunikacinė diagrama
- Tarpusavio sąveikos diagrama (UML 2.0)
- Tarpusavio sąveikos apžvalgos diagrama
- UML Laiko diagrama (UML 2.0)

Protokolo būsenos mašina yra papildomas būsenos mašinos variantas. Jis gali būti naudojamas modeliuoti tinklų komunikacijos protokolus.

UML neriboja UML elementų tipų iki tam tikro diagramos tipo. Aplamai, kiekvienas UML elementas gali pasitaikyti beveik kiekvieno tipo diagramoje. Toks lankstumas dalinai ribojamas UML 2.0. Laikantis inžinerinių brėžinių tradicijų, UML diagramoje visada leidžiama pateikti komentarą ar pastabą dėl naudojimo, apribojimų ar paskirties.

UML naudoja daugelį koncepcijų iš įvairių šaltinių. Baigtinį sąrašą galite rasti Unifikuotų modeliavimo kalbos terminų glosarijuje. Čia pateiktos žinomiausios koncepcijos.

- *Struktūrai* - Actor (aktorius, veikėjas, dalyvis), atributas, klasė, komponentas, sąsaja, objektas, paketas.
- *Elgsenai* - Veikla, įvykis, pranešimas, metodas, operacija, būsena, naudojimo scenarijus.
- *Sąveikoms* - Aggregation (konglomeratas, sankaupa), asociacija, composition (sudėtis, sandara, kompozicija), priklausomybė, generalizacija (arba paveldimumas, veldinys - inheritance).
- *Kitos koncepcijos* - Stereotipas. Jis apibūdina jam priskirtą simbolį. Daugybiškumo sąvoka, atitinkanti duomenų bazės modeliavimo kardinalumą (cardinality - daugelio elementų galią), pvz., 1, 0..1, 1..* [5]

1.5.2. Būsenų modelio aprašymas

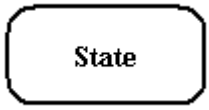
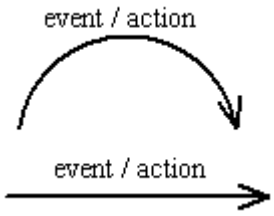
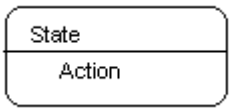
Būsenos diagramos naudojamos aprašyti sistemos elgsenai. Būsenos diagramos apibūdina visas galimas objekto būsenas, vykstant įvykiams. Paprastai kiekviena diagrama vaizduoja tos pačios klasės objektus, ir atseka įvairias savo objektų būsenas visoje sistemoje.

Būsenos diagramas naudokite pademonstruoti objekto elgsenai pagal įvairius sistemos naudojimo scenarijus. Būsenos diagramas naudokite tik klasėms, kai perprasti objekto elgseną visoje sistemoje. Būsenos diagramų reikia ne visoms klasėms, būsenos diagramos nenaudingos apibūdinant visų objektų tarpusavio sąveikas viename naudojimo scenarijuje. Būsenos diagramos derinamos su kitomis diagramomis, tokiomis kaip tarpusavio sąveikos ir veiklos diagramos.

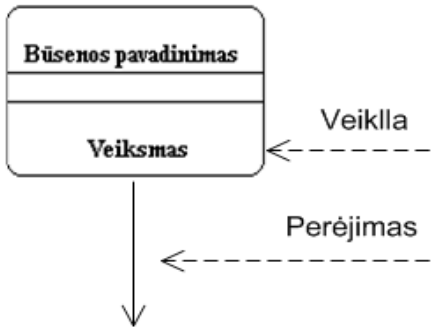
Būsenos diagramos turi labai nedaug elementų. Pagrindiniai elementai – užapvalinti langeliai, vaizduojantys objekto būseną, ir rodyklės, rodančios perėjimą į sekančią būseną. Būsenos simbolio veiklos dalis vaizduoja, ką darys objektas būdamas toje būsenoje [11].

3 lentelė

UML 2.0 būsenų diagramos elementai

Pavadinimas	Paskirtis	Grafinis žymuo
Pradinis taškas (angl. Control State: Initial State)	Naudojamas Būsenos pradžiai žymėti, kuomet pradedama veikla, aprašoma UML 2.0 State diagramoje.	
Būsenos pabaiga (angl. Control State: Final State: Final State)	Final State žymi vienos būsenos pabaigą UML 2.0 State.	
Būsena (angl. State)	Būsenos atvaizduoja situacijas objekto gyvavimo etape.	
Įvykis (angl. Event/action)	Šis žymėjimas nurodo ryšius tarp objekto būsenų	
Veiksmas (angl. Action)	Pažymi veiksmą, kurio metu keičiasi objekto būsena	

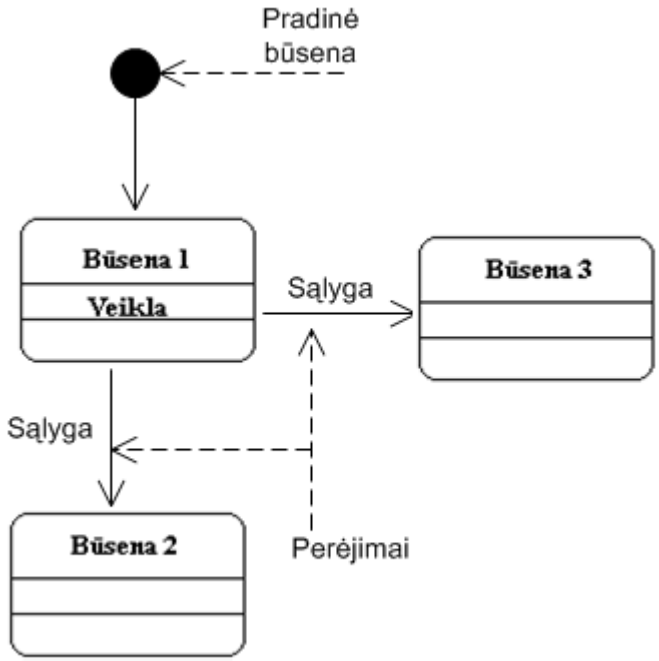
Šaltinis: sudaryta autoriaus.



Šaltinis: sudaryta autoriaus.

9 pav. Būsenų diagramos pavyzdys nr.1

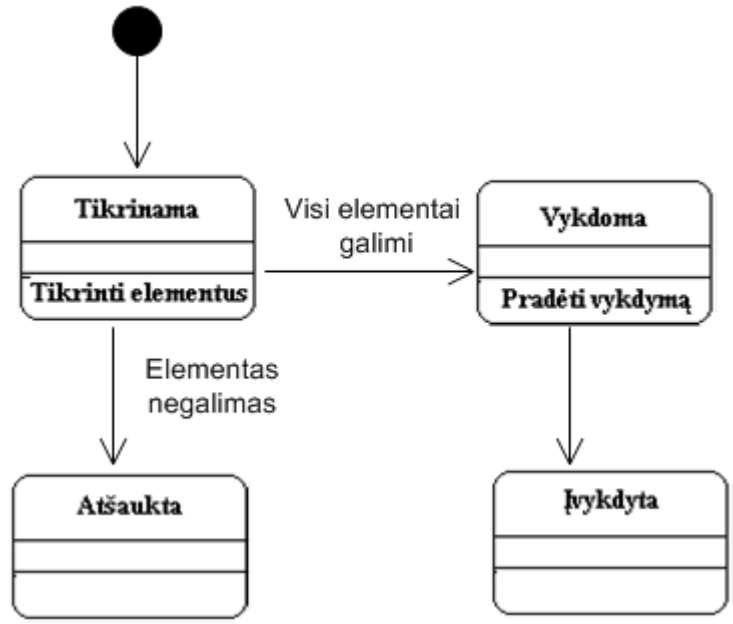
Visos būsenos diagramos rodo pradinę objekto būseną. Tai tokia objekto būsena, kai jis kuriamas. Po pradinės būsenos objektas pradeda būsenas keisti. Nuo veiklos priklausančios sąlygos gali nulemti, į kokią kitą būseną objektas pereis.



Šaltinis: sudaryta autoriaus.

10 pav. Būsenų diagramos pavyzdys nr.2

Žemiau pateikimas būsenos diagramos pavyzdys galėtų tikti Tvarkos objektui. Kai objektas įeina į Tikrinimo būseną, jis atlieka veiklą „tikrinti elementus“. Baigus veiklą, objektas pereina į sekančią būseną priklausomai nuo sąlygų [visi elementai prieinami/galimi] arba [elementas neprieinamas/ negalimas]. Jei elementas neprieinamas/ negalimas, tvarka atšaukiama. Jei visi elementai prieinami/ galimi, tuomet tvarka vykdoma. Objektui pereinant į Vykdymo būseną, atliekama veikla „pradėti pristatymą“. Baigus šią veiklą, objektas vėl pereina į būseną Įvykdyta.

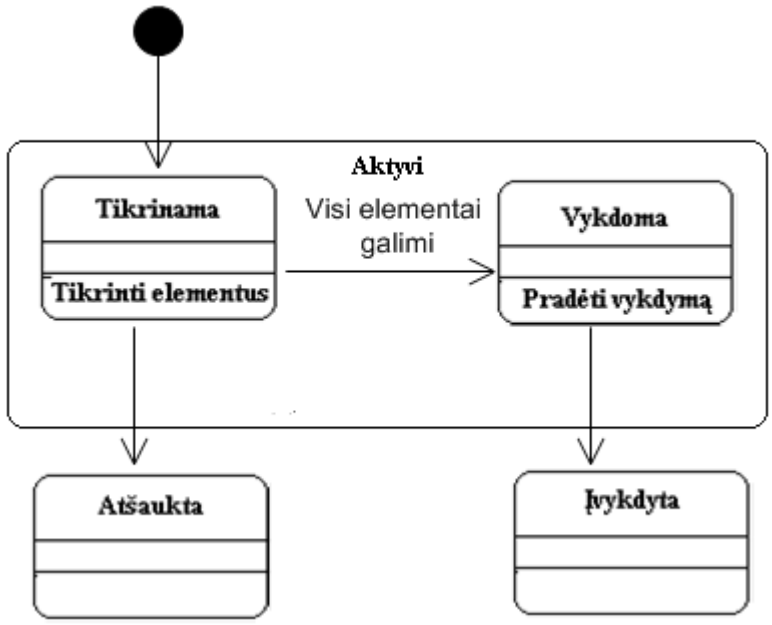


Šaltinis: sudaryta autoriaus.

11 pav. Būsenų diagramos pavyzdys nr.3

Būsenos diagramos taip pat gali parodyti objekto super-būseną. Super-būsena naudojama, kai daug perėjimų veda į tam tikrą būseną. Vietoj to, kad atvaizduoti visus perėjimus nuo kiekvienos būsenos į perteklinę (atsarginę) būseną, galima naudoti super-būseną atvaizduoti, kad visos būsenos super- būsenos viduje gali perteklinę (atsarginę) būseną. Taip būsenos diagramą lengviau peržiūrinėti.

Diagramoje žemiau vaizduojama super- būsena. Ir Tikrinimo, ir Vykdymo būsenos gali pereiti į būseną Atšaukta, todėl atvaizduotas perėjimas iš super- būsenos, pavadintos Aktyvia, būseną Atšaukta. Priešingai, būsena Vykdoma gali pereiti tik į būseną Įvykdyta, todėl brėžiame rodyklę tik iš būsenos Vykdoma į būseną Įvykdyta[6].



Šaltinis: sudaryta autoriaus.

12 pav. Būsenų diagramos pavyzdys nr.4

1.6. Egzistuojančiuose CASE įrankiuose realizuotų projektinių modelių generavimo galimybių tyrimas.

1.6.1. CASE įrankių palyginimas

Tarp populiariausių CASE priemonių, pirmauja daug galimybių turinčios, visus arba daugumą sistemos kūrimo gyvavimo ciklo etapų palaikančios programos. (4 lentelė) pateiktos laisvai prieinamų priemonių savybės ir galimybės

4 lentelė

CASE priemonių charakteristikos

CASE įrankis Savybės	ArgoUML	Visual Paradigm	Poseidon	UMLet	MagicDraw
UML versija	**UML 1.3**	**UML2.0**	**UML2.0**	**UML2.0**	**UML2.0**
UML diagramos**					
Class diagram	+	+	+	+	+
Object diagram	+	+	+	+	+
Use Case diagram	+	+	+	+	+
Sequence diagram	+	+	+	+	+
Collaboration diagram	+	-	+	+	-
State diagram	+	-	-	-	-
Activity diagram	**- ***	+	+	+	+
Component diagram	+	+	-	+	+
Deployment diagram	+	+	+	+	+
Timing Diagram	-	+	-	+	
State Machine Diagram	-	+	+	+	+
Package Diagram	-	+	-	+	+
Interaction Overview Diagram	-	+	-	+	-
Composite Structure Diagram	-	+	-	+	+
Communication Diagram	-	+	-	+	+
Kodo generavimas	JAVA, C++, C# PHP	+	-	-	-
XMI palaikymas	+	+	+	-	+

Šaltinis: sudaryta autoriaus.

ArgoUML ir UMLet yra Java įrankiai, abu atviro kodo. UMLet galima naudoti kaip įskiepį Eclipse-3 programinei įrangai. Kadangi šių abiejų priemonių pagrindas yra Java, tai reiškia, kad juos galima naudoti skirtingose platformose (Windows, Unix ir kt.), tačiau UMLet stipriai nusileidžia savo galimybėmis likusiems įrankiams. UMLet daugiau braižymo priemonė, kuri gali būti skirta greitam eskizų darymui, mokymo tikslams, todėl iš aprašytų įrankių UMLet lieka paskutinis.

Apžvelgus tokias galimybes kaip kodo generavimas ir inžinerija, tarp nemokamų CASE priemonių geriausia šiuo atveju ArgoUML. Kiti įrankiai yra komerciniai, todėl nemokamos jų versijos skirtos daugiau demonstraciniams tikslams, supažindinti potencialų klientą su produktu.

1.6.2.Visual Paradigm modelių generavimo galimybės

Kompanijos *Visual Paradigm International* kuriamas programinis paketas skirtas programinės įrangos inžinieriams, projektuotojams bei projektų vadovams. Jo galimybės apima didelę dalį veiklos modeliavimo, programinės įrangos projektavimo bei kodo generavimo procesų;

su jo pagalba šie uždaviniai gali būti realizuoti greičiau ir paprasčiau bei, kas labai svarbu bet kuriai IS įmonei ar projekto komandai, su mažesniais kaštais.

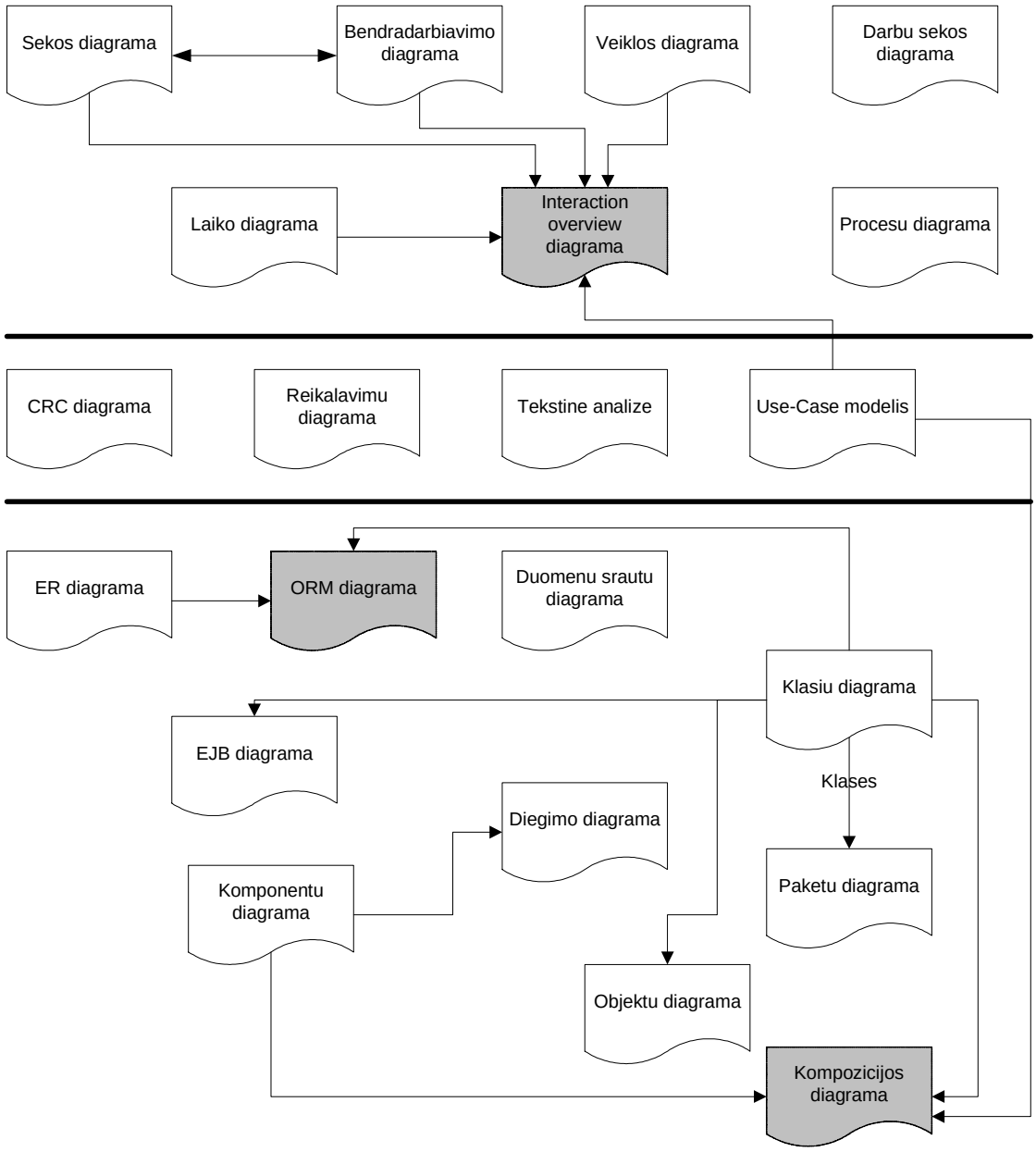
Visual Paradigm for UML pakete naudojami du skirtingi metodai – verslo procesų modeliavimui naudojama BPMN (*Business Process Modelling Notation*) notacija, o vartotojo poreikių specifikavimui bei programinės įrangos inžinerijai taikomas populiarus UML (*Unified Modelling Language*) metodas.

Naudojantis BPNM notacija, galima specifikuoti įvairius veiklos procesus bei jų seką. Šioje notacijoje pateikiami visi elementai, būtini tokio tipo analizei bei aprašymui.

Veiklos procesų modeliavimui taip pat gali būti naudojamos ir UML diagramos:

- sekos diagrama (*sequence diagram*);
- bendradarbiavimo diagrama (*communication diagram*) – ši diagrama panaši į sekos diagramą, o Visual Paradigm leidžia jas sinchronizuoti (t.y., generuoti vieną iš kitos);
- ***state machine diagrama*;**
- UML veiklos (*activity)* diagrama;
- UML laiko (*timing*) diagrama;
- UML Interaction Overview diagrama.

Žemiau pateiktame (13 pav.) yra pavaizduoti Visual Paradigm modelių tarpusavio ryšys, ir galimybė, juos generuoti. Tamsiame fone pateiktos diagramos, turi galimybę būti generuojamos.



Šaltinis: sudaryta autoriaus.

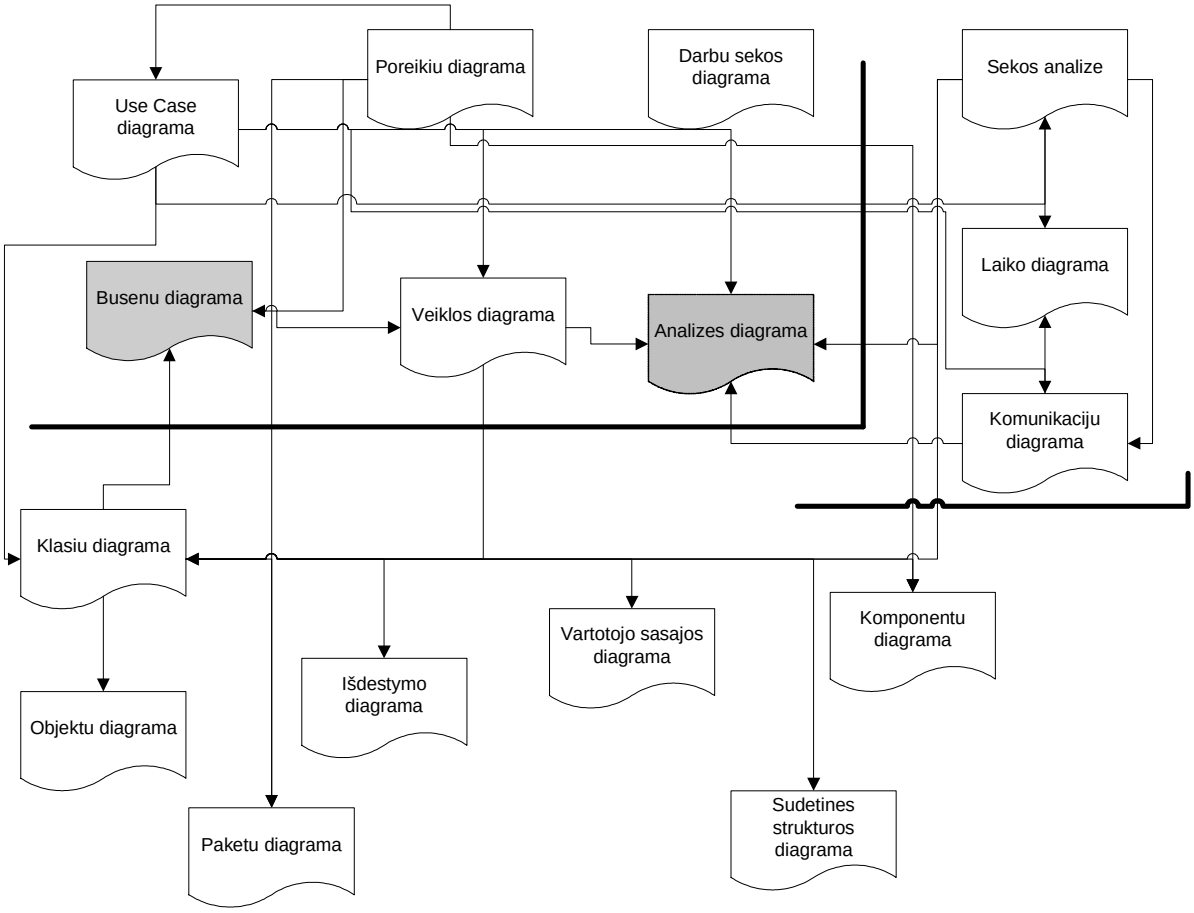
13 pav. Visual Paradigm modelių generavimo galimybės

1.6.3. Enterprise architect modelių generavimo galimybės

Enterprise Architect yra visapusiškas UML analizės ir dizaino įrankis, apimantis programinės įrangos plėtrą nuo poreikių surinkimo per analizės stadijas, modeliavimą, testavimą ir įgyvendinimą. Tai įrankis, kuris gali būti naudojamas ne vieno vartotojo. Enterprise Architect yra grafinis įrankis, kuris padeda sukonstruoti programinę įrangą.

EA padeda valdyti painią visumą su įrankiu, kuris yra labai lankstus, turi daug modelių. Taip pat yra galimybė kurti ataskaitas. Ši programa palaiko reinžinerijos galimybes iš kodo daugeliui populiarių kalbų, įskaitant ++, C#, Java, Delphi, VB.Net, Visual Basic, ActionScript ir PHP. Enterprise Architect paremtas UML modeliavimu.

Žemiau pateiktame (14 pav.) yra pavaizduoti modelių tarpusavio ryšys, ir galimybė, juos generuoti. Tamsiame fone pateiktos diagramos, turi galimybę būti generuojamos.

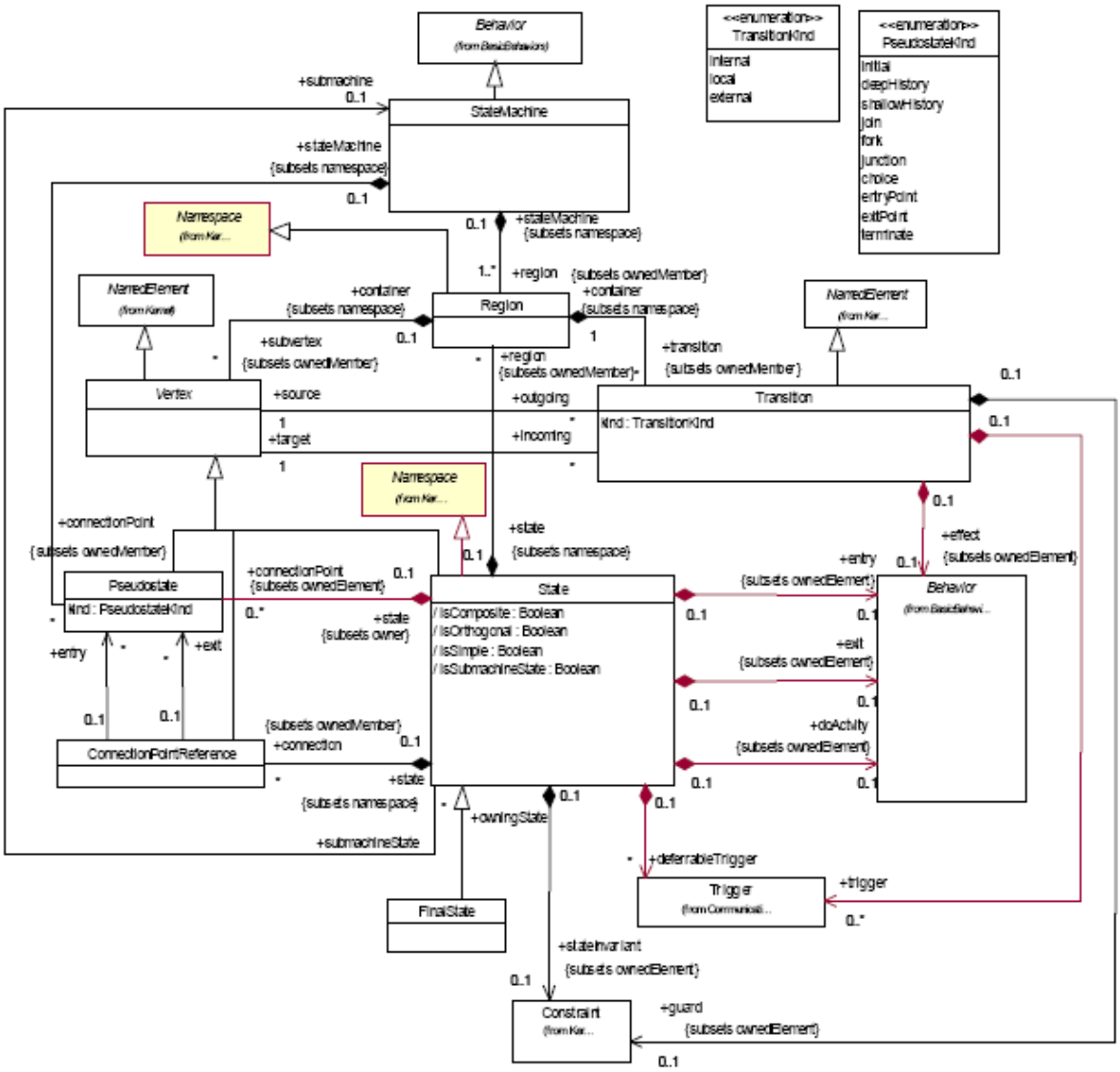


Šaltinis: sudaryta autoriaus.

14 pav. Enterprise architect modelių generavimo galimybės

2.SIŪLOMO SPRENDIMO METODIKA

2.1.Būsenų (state) metamodelis



Šaltinis: OMG, [2005] Unified modelig language: Superstructure.

15 pav. Būsenų diagramos metamodelis [9]

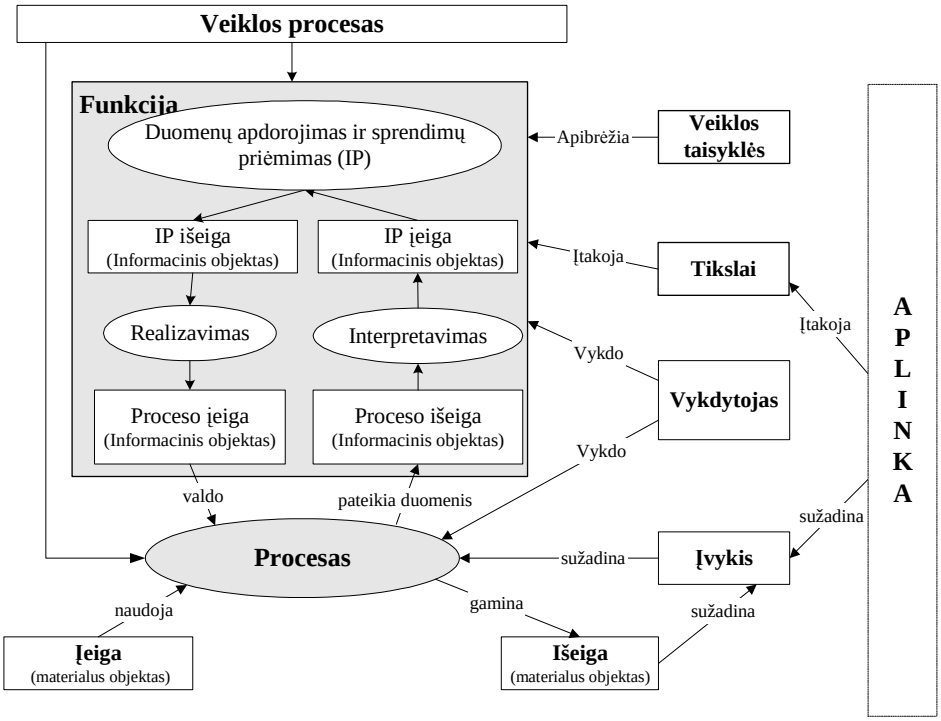
MDA (Model Driven Architecture) kiekvienas modelis yra paremtas specifiniu metamodeliu, apibūdinančiu modeliavimo kalbą. Visi metamodeliai paremti vienu metamodeliu vadinamu MOF (Meta Object Facility), todėl modelių transformacijos yra galimos tarp dviejų metamodelių apibrėžiant transformavimo taisykles. Transformavimo taisyklės aprašo apjungimą (mapping) tarp šaltinio ir rezultato metamodelių, kurie turi ekvivalenčią, ar panašią semantiką.

2.2.Veiklos metamodelio aprašymas

Veiklos metamodelis apibrėžia kompiuterizuojamos veiklos dalykinės srities žinių, būtinų projektiniams modeliams kurti ir programiniam kodui generuoti sudėtį. Veiklos žinių saugykla užpildyta konkrečios kompiuterizuojamos veiklos dalykinės srities žiniomis vadinamas veiklos modeliu.

2.2.1.Veiklos metamodelio konceptuali schema

Veiklos metamodelio sudėties esminis bruožas– proceso ir funkcijos sankirta. Procesas– darbų sekų elementas, materialų įeigos srautą transformuojantis į materialų išeigos srautą, tam kad įgyvendintų organizacijos keliamą tikslą. Funkcija– darbų sekų elementas, skirtas procesų valdymui ir kontrolei. Kiekvieną procesą, transformuojantį materialų įeigos srautą į materialų išeigos srautą valdo bent viena funkcija. Procesas pateikia valdymo funkcijai proceso būsenos atributus, kurie interpretavimo metu paverčiami duomenų apdorojimo ir sprendimo priėmimo funkcijos sudėtinės dalies įeigos atributais. Interpretavimas tai taisyklių rinkinys, skirtas transformuoti proceso būsenos atributus (Proceso išeiga) į duomenų apdorojimo ir sprendimų priėmimo funkcijos dalies (IP) apdorojimui paruoštą informacinį srautą, vadinamą duomenų apdorojimo ir sprendimų priėmimo įeiga (IP įeiga). Interpretavimas yra būtina funkcijos sudėtinė dalis, nes proceso išeigos atributų formatas, gali neatitikti duomenų IP įeigos srautui (IP Įeiga) patvirtinto duomenų formato. Duomenų apdorojimas ir sprendimų priėmimas (IP) tai funkcijos sudėtinė dalis, kurios pagrindinė paskirtis atlikti procesą valdančius informacijos apdorojimo ir sprendimo priėmimo veiksmus. IP duomenų apdorojimo ir sprendimo priėmimo įeigos informacinį srautą (IP įeiga) transformuoja į duomenų apdorojimo ir sprendimų priėmimo išeigos informacinį srautą (IP išeiga). Realizavimas– tai funkcijos sudėtinė dalis atliekanti veiksmą priešingą interpretavimui. Realizavimas duomenų apdorojimo ir sprendimų priėmimo išeigos duomenis (IP Išeiga) transformuoja į proceso valdymo atributų formatą (Proceso Įeiga). Interpretavimo, IP ir Realizavimo funkcijos sudėtinių dalių vidinę struktūrą apibrėžia veiklos taisyklės. Funkcija įgyvendina bent vieną organizacijos keliamą tikslą arba jo potikslį. Procesą ir funkciją vykdo organizacijos vykdytojas. Vykdytoju gali būti ne tik asmuo ar organizacinis padalinys, bet ir programinė įranga, įrengimas ir t.t. Materialaus proceso vykdymą sužadina aplinkos inicijuojamas įvykis. Aplinka sužadina įvykį bei įtakoja organizacijos keliamus tikslus. Veiklos metamodelio konceptuali schema pateikta 16 paveiksle.



Šaltinis: GUDAS S., LOPATA A. (2005) Žiniomis grindžiamos informacijos sistemų inžinerijos bruožai.

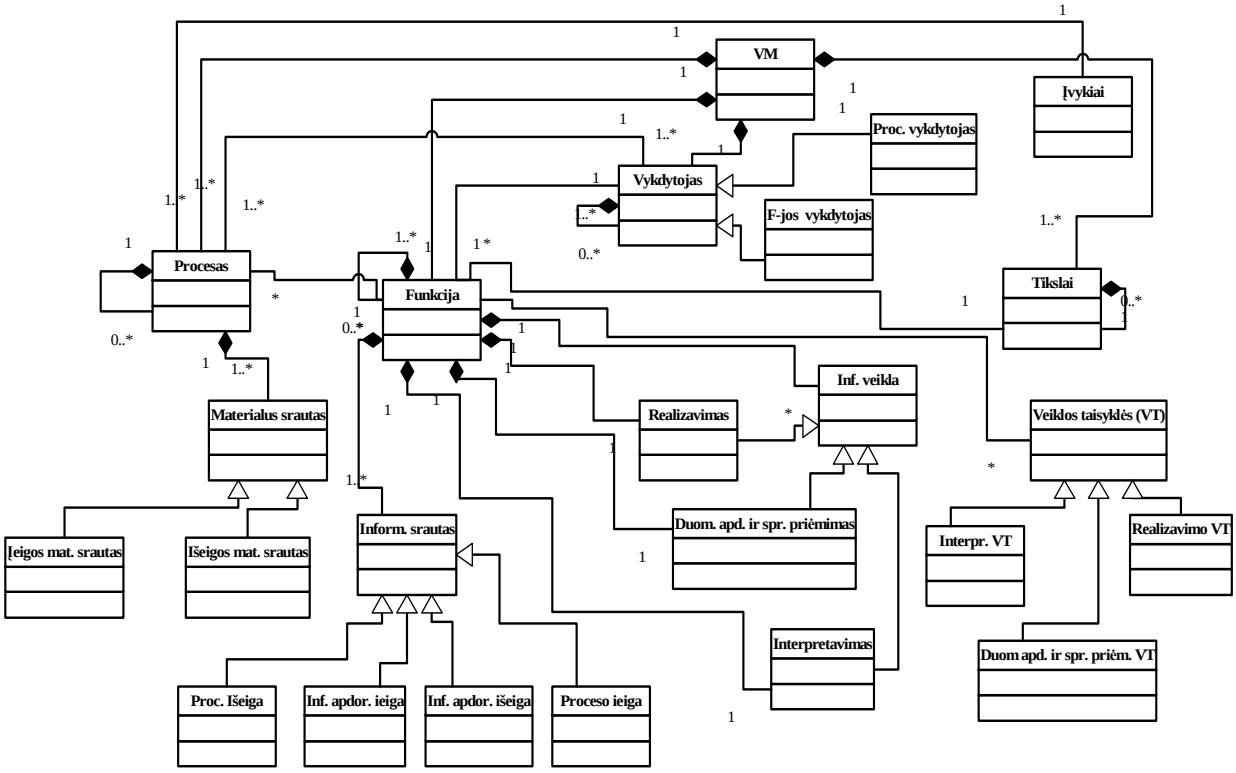
16 pav. Veiklos metamodelio konceptuali schema

Veiklos metamodelio pagrindu kuriamas veiklos modelis gali būti hierarchinės sudėties. Tai atspindi veiklos metamodelio klasių diagrama, pateikta 17 paveiksle.

2.2.2.Veiklos metamodelio klasių modelis

Veiklos metamodelio klasių modelio branduolį sudaro dvidešimt trys klasės. Esminės klasės yra *Procesas, Funkcija* ir *Vykdytojas*. Klasės *Procesas*, *Funkcija, Vykdytojas ir Tikslai* gali turėti vidinę hierarchinę struktūrą. Tai atspindi agregavimo santykis. Klasė *Procesas* agregavimo santykiu siejama su klase *Materialus srautas*. Klasė *Materialus srautas* apibendrinimo santykiu siejama su klasėmis *Įeigos materialus srautas* ir *Išeigos materialus srautas*. Klasė *Procesas* asociacijos santykiais susieta su klasėmis *Funkcija, Vykdytojas* ir *Įvykis*. Klasė *Funkcija* agregavimo santykiais susieta su klasėmis *Informacinis srautas, Informacinė veikla, Interpretavimas, Duomenų apdorojimas ir sprendimų priėmimas* ir *Realizavimas*. Šiais santykiais apibrėžiama funkcijos vidinė sudėtis. Klasė *Informacinis srautas* apibendrinimo santykiu susieta su klasėmis „*Proceso Išeiga*“, *Informacijos apdorojimo ir sprendimo priėmimo įeiga, Informacijos apdorojimo ir sprendimo priėmimo išeiga, Proceso įeiga*. Klasė *Informacinė veikla* apibendrinimo santykiais susieta su klasėmis *Interpretavimas, Duomenų apdorojimas ir sprendimų priėmimas* ir *Realizavimas*. Klasė *Funkcija* asociacijos santykiais susieta su klasėmis *Vykdytojas, Tikslai* ir *Veiklos taisyklės*. Klasė *Veiklos taisyklės* (VT) apibendrinimo santykiu siejama su klasėmis *Interpretavimo VT, Duomenų apdorojimo ir sprendimo priėmimo VT* ir *Realizavimo VT*. Klasė *Vykdytojas* apibendrinimo

santykiu siejama su klasėmis *Funkcijos vykdytojas* ir *Proceso vykdytojas*. Veiklos metamodelio klasių modelis pateiktas 17 paveiksle.



Šaltinis: S. Gudas, A. Lopata, (2007) Workflow models based acquisition of enterprise knowledge.

17 pav. Veiklos metamodelio klasių modelis

2.2.3.Formalizuotas veiklos metamodelio aprašymas

Pateiksime formalizuotą veiklos modelio (VM) metamodelio aprašą, kuris reikalingas būsenų diagramos modelio generavimo algoritmui aprašyti. VM metamodelį aprašysime abstrakčiosios algebros pagrindu kaip algebrinę sistemą M1 (Malcev, 1970):

M1=<K, R>;

čia: M1 – veiklos metamodelis kaip algebrinė sistema;

K – sistemos M1 elementų aibė.

K={K1, K2,…, K21}, kur K1,....K21 yra veiklos metamodelio metaklasės;

R – ryšių tarp elementų aibė, kur R={r1, r2, r3}.

Kiekvieno aibės K elemento Kn sudėtis apibrėžiama taip:

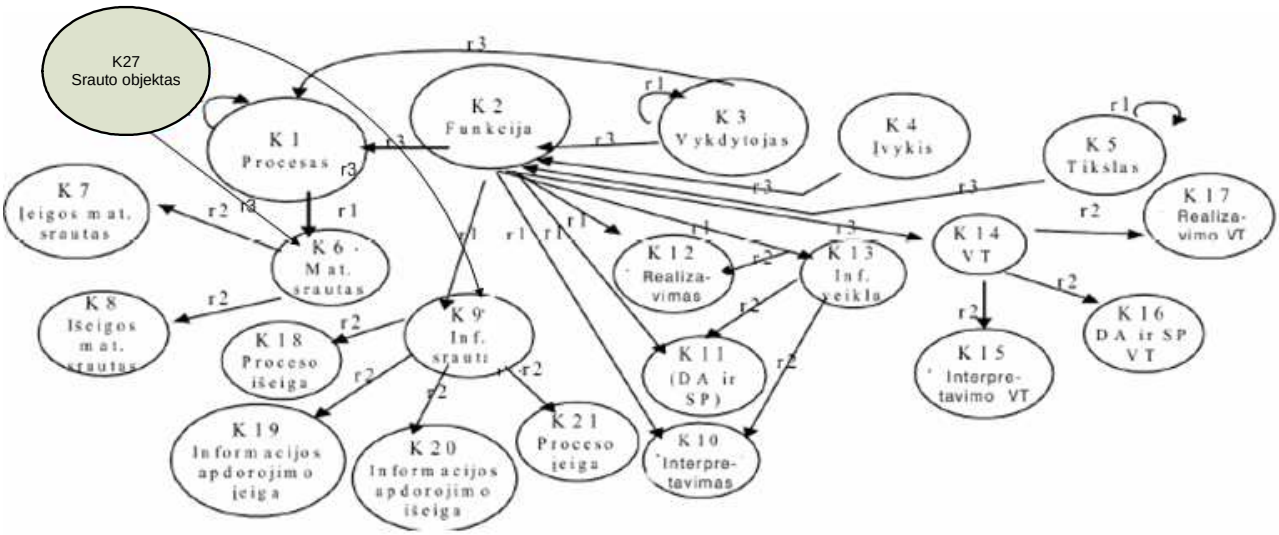
Kn=<{an1, an2,…,ank}, {mn1, mn2,…,mnl}>, kur {an1, an2,…,ank} – elemento Kn atributai,

{mn1, mn2,…,mnl}– elemento Kn metodai.

Veiklos metamodelio M1 sudėtis yra tokia:

M1=<{K1, K2,...,K21}, {r1, r2, r3}>;

čia: K1– metaklasė *Procesas*, K2 – metaklasė *Funkcija*, K3 – metaklasė *Vykdytojas*, K4 – metaklasė *Įvykis*, K5 – metaklasė *Tikslas*, K6 – metaklasė *Materialus srautas*, K7– metaklasė *Įeigos materialus srautas*, K8 – metaklasė *Išeigos materialus srautas*, K9 – metaklasė *Informacijos srautas*, K10 –metaklasė *Interpretavimas*, K11 – metaklasė *Duomenø apdorojimas ir sprendimų priėmimas* (DA ir SP), K12 – metaklasė *Realizavimas*, K13 – metaklasė *Informacinė veikla*, K14 – metaklasė *Veiklos taisyklės* (VT), K15 – metaklasė *Interpretavimo veiklos taisyklės*, K16 – metaklasė *Duomenų apdorojimo ir sprendimų priėmimo veiklos taisyklės (DA ir SP VT)*, K17 – metaklasė *Realizavimo veiklos taisyklės*, K18 – metaklasė *Proceso išeiga*, K19 – metaklasė *Informacijos apdorojimo įeiga*, K20 – metaklasė *Informacijos apdorojimo išeiga*, K21 – metaklasė *Proceso įeiga*, r1 – agregavimo santykis, r2 – apibendrinimo santykis, r3 – asociacija. Veiklos metamodelio grafinė schema pateikiama 18 paveiksle [12].



Šaltinis: sudaryta autoriaus.

18 pav. Veiklos modelio M1 grafinė schema

Toliau aprašomi veiklos metamodelio komponentų tarpusavio ryšiai:

• (K1)r1(K1), (K3)r1(K3), (K5)r1(K5) – klasės „Procesas“ (K1), „Vykdytojas“ (K3) ir „Tikslai“ (K5) turi vidinę hierarchinę sudėtį;

• (K1)r1(K6) – klasė „Procesas“ (K1) agregavimo ryšiu (r1) susieta su klase „Materialus srautas“ (K6);

• (K6)r2(K7), (K6)r2(K8) – klasė „Materialus srautas“ (K6) apibendrinimo ryšiu (r2) susieta su klasėmis „Įeigos materialus srautas“ (K7) ir „Išeigos materialus srautas“ (K8);

• (K2)r1(K9), (K2)r1(K10), (K2)r1(K11), (K2)r1(K12), (K2)r1(K13) – klasė „Funkcija“ (K2) agregavimo ryšiu (r1) susieta su klasėmis „Informacinis srautas“ (K9), „Interpretavimas“ (K10), „Duomenų apdorojimo ir sprendimo priėmimas“ (K11), „Realizavimas“ (K12) ir „Informacinė veikla“ (K13);

• (K13)r2(K10), (K13)r2(K11), (K13)r2(K12) – klasė „Informacinė veikla“ (K13) apibendrinimo ryšiu (r2) susieta su klasėmis „Interpretavimas“ (K10), „Duomenų apdorojimas ir sprendimų priėmimas“ (K11) ir „Realizavimas“ (K12);

• (K14)r2(K15), (K14)r2(K16), (K14)r2(K17) – klasė „Veiklos taisyklės“ (K14) apibendrinimo ryšiu (r2) susieta su klasėmis „Interpretavimo veiklos taisyklės“ (K15), „Duomenų apdorojimo ir sprendimo priėmimo veiklos taisyklės“ (K16) ir „Realizavimo veiklos taisyklės“ (K17);

• (K3)r3(K1) – klasė „Vykdytojas“ (K3) asociacija (r3) susieta su klase „Procesas“ (K1);

• (K3)r3(K2) – klasė „Vykdytojas“ (K3) asociacija (r3) susieta su klase „Funkcija“ (K2);

• (K5)r3(K2) – klasė „Tikslai“ (K5) asociacija (r3) susieta su klase „Funkcija“ (K2);

• (K2)r3(K14) – klasė „Funkcija“ (K2) asociacija (r3) susieta su klase „Veiklos taisyklės“ (K14).

2.2.4.Būsenų (State) diagramos formalaus modelio generavimas

Prieš pateikiant IS būsenų (state) modelio generavimo veiklos modelio pagrindu algoritmą, parodysime, kad veiklos metamodelio pagrindu sukurtoje žinių bazėje saugomų žinių pakanka būsenų (state) modeliui generuoti.

State modelio formalų aprašymą sudarome remdamiesi, state diagramos metamodeliu, kuris pateikiamas 20 pav.

Algebrinis state diagramos M2 aprašysime taip:

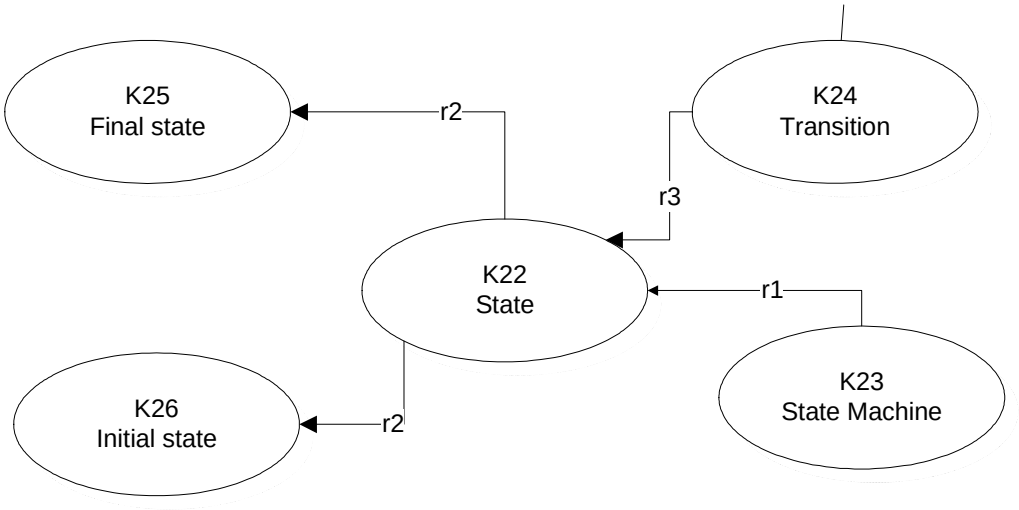
M2=<{K22, K23,...,K26}, {r1, r2, r3}>.

M2 – State diagramos modelis, K22 – klasė State, K23 – klasė State Machine, K24 – klasė Transition, K25 – klasė Final state, K26 – klasė Initial state, r1 – agregavimo santykis, r2 – apibendrinimo santykis, r3 – asociacija.

(K22)r2(K25), (K22)r2(K26) – klasė State apibendrinimo ryšiais yra susieta su Initial state ir Final state klasėmis.

(K23)r3(K22) – klasė State Machine asociacijos ryšiu yra susieta su State klase.

(K24)r1(K22) – Klasė Transition agregavimo ryšiu yra susieta su State klase.



Šaltinis: sudaryta autoriaus.

19 pav. Būsenų (State) modelio grafinė schema M2

3. EKSPERIMENTINIS SKYRIUS

Aprašysime veiklos modelio ir state diagramos modelio loginį ryšį nagrinėdami jų klasių modelius, sukurto algoritmo, kurio pagalba yra generuojama būsenų (state) diagrama etapų aprašymas. Sukurto algoritmo patvirtinimas naudojant realius duomenis ir programinės įrangos prototipo sukurtam algoritmui pasiūlymas.

3.1.Veiklos modelio ir būsenų (state) diagramos modelio sąsaja

State diagramos modelio generavimui būtinos veiklos modelio klasės *Procesas (K1) , Funkcija (K2), Informacinis srautas (K9), Materialus srautas (K6) ir srauto būsena(K27).*

Generuojant state diagramos modelį šios klasės atvaizduojamos į atitinkamas state diagramos modelio klases *State* (K22), *StateMachine* (K23), *Transition* (K24), *Final state*(K25) ir *Initial state* (K26). Tai formaliai galima aprašyti aibių atvaizdžiais: φ1 : K27->K23; φ2: K1->K24; φ3: K2->K24; φ4: K6->K22; φ5: K6->K26; φ6: K6->K25; φ7 K9->K22; φ8: K9->K26; φ9: K9->K25 (4 lentelė). Aibių pavadinimai atitinka klasių pavadinimus.

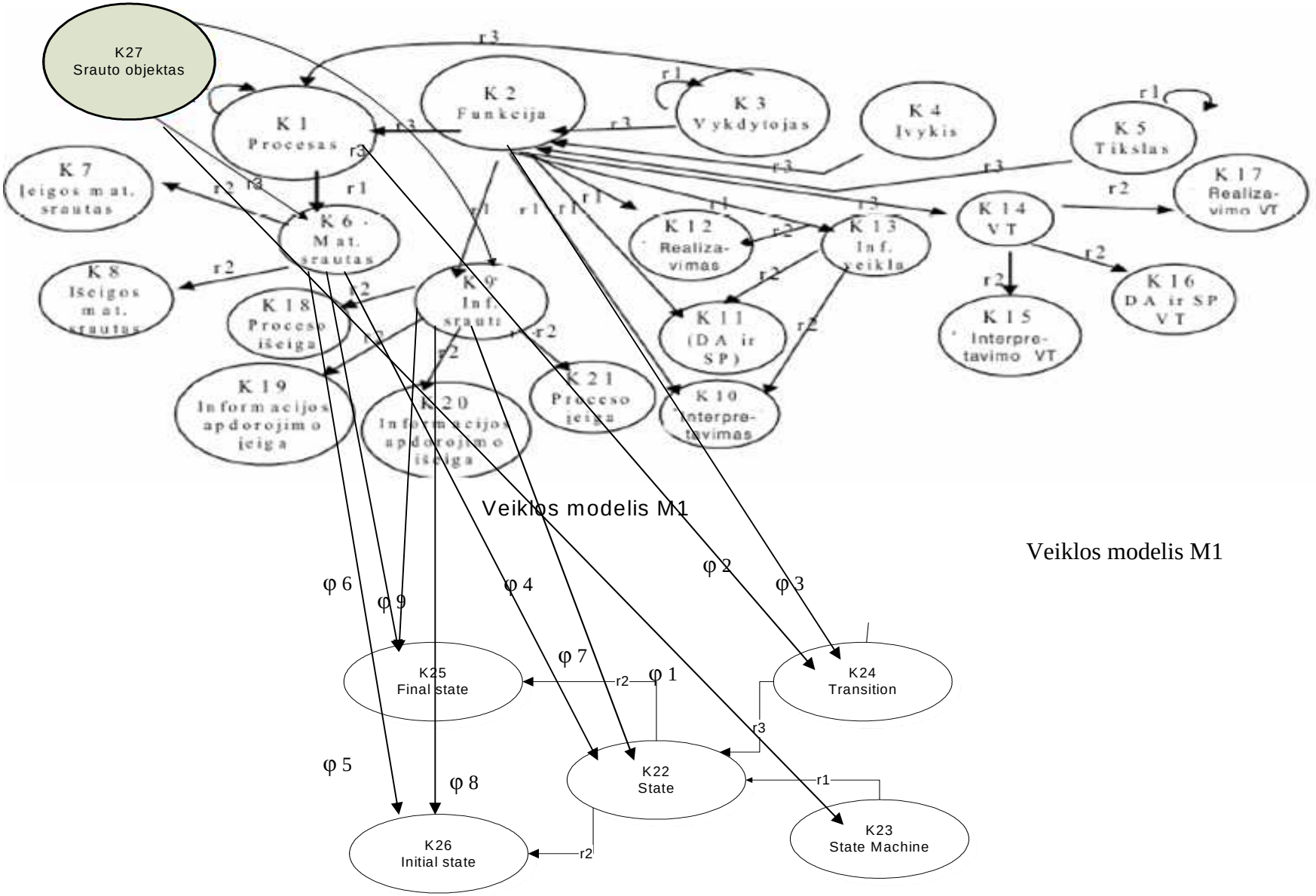
5 lentelė

Veiklos modelio klasių atvaizdžiai į atitinkamas state diagramos modelio klases

Veiklos modelio klasių aibės	**Atvaizdas**	**Atvaizdis Activity diagramos modelio klasių aibės**	**Formalus aprašymas**
srauto objektas(K27)	φ 1	*StateMachine* (K23)	K27->K23
Procesas (K1)	φ 2	*Transition* (K24)	K1->K24
Funkcija (K2)	φ 3	*Transition* (K24)	K2->K24
Materialus srautas (K6)	φ 4	*State* (K22)	K6->K22
Materialus srautas (K6)	φ 5	*Initial state* (K26)	K6->K26
Materialus srautas (K6)	φ 6	*Final state*(K25)	K6->K25
Informacinis srautas (K9)	φ 7	*State* (K22)	K9->K22
Informacinis srautas (K9)	φ 8	*Initial state* (K26)	K9->K26
Informacinis srautas (K9)	φ 9	*Final state*(K25)	K9->K25

Šaltinis: sudaryta autoriaus.

Analizuosime state diagramos elementų generavimo galimybes, pasinaudodami veiklos modelyje sukauptomis žiniomis. Grafinis lentelės atvaizdavimas pateikiamas 20 paveiksle.



Veiklos modelis M1

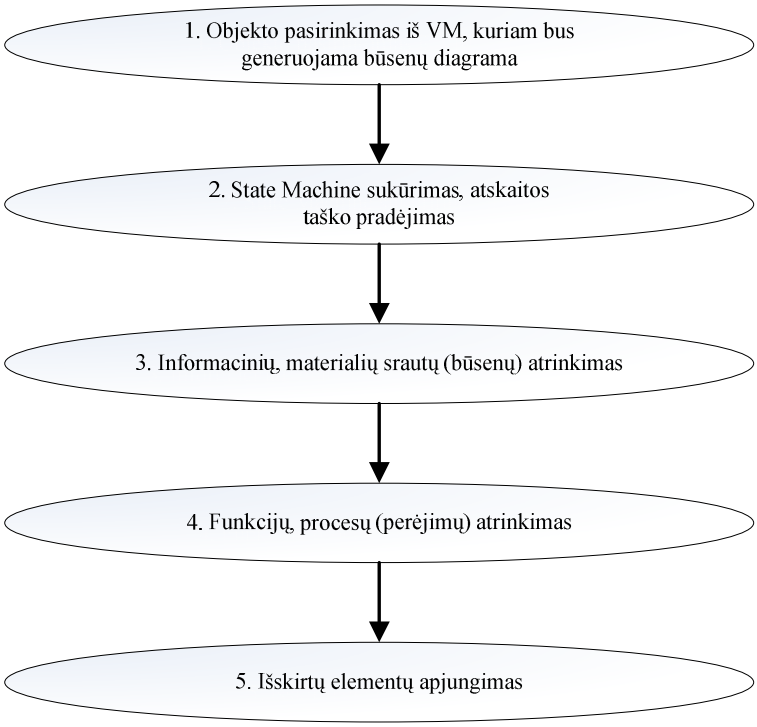
Šaltinis: sudaryta autoriaus.

Būsenų (state) diagramos modelis M2

20 pav. Veiklos modelio elementų atvaizdavimas į būsenų (state) diagramos modelį

3.2.Informacinės sistemos state modelio generavimo algoritmas

Pateikiamas bendro pavidalo algoritmas UML būsenų diagramos generavimui.



Šaltinis: sudaryta autoriaus.

21 pav. IS būsenų modelio generavimo algoritmas

1 žingsnis Pasirenkamas objektas iš veiklos modelio, kuriam pradedamas būsenų (state) diagramos projektas. Atlikus objekto pasirinkimą pradedamas naujas būsenų (state) projekto (diagramos) kūrimas, kurį įvardinkime SD (state diagram). Objektas pasirenkamas iš veiklos modelio duomenų bazės lentelės SD_SRAUTO_OBJEKTAS.

2 žingsnis State Machine sukūrimas, naudojame būsenų diagramos duomenų bazės lentelę SD_STATE_MACHINE. Atskaitos taško sukūrimas Initial State ir atvaizduojame SD projekte.

3 žingsnis Su pasirinktu vartotojo objektu būsenų atrinkimas ir atvaizdavimas SD projekte. Būsenoms atrinkti naudosime veiklos modelio duomenų bazės lenteles: matsrautas ir infsrautas.Atrinkę juos perrašome į būsenų diagramos duomenų bazės lentelę SD_STATE.

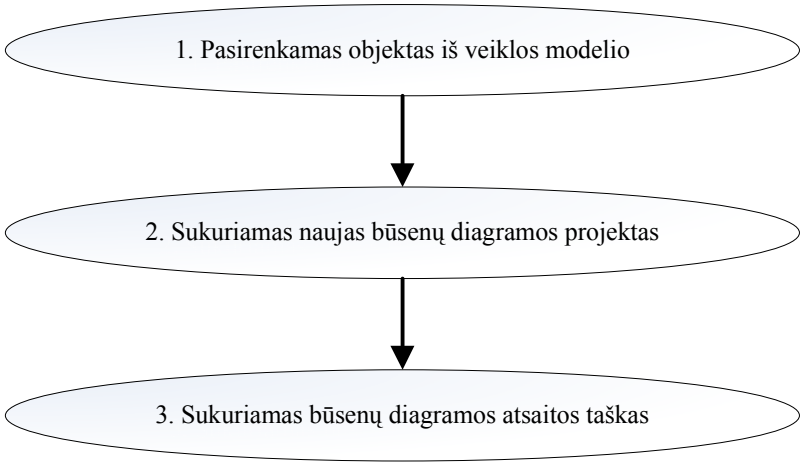
4. žingsnis Su atrinktomis būsenomis susijusių perėjimų atrinkimas ir atvaizdavimas SD projekte. Atrinkimui naudosime veiklos modelio duomenų bazės lenteles: procesas, funkcija. Atrinkę juos perrašome į būsenų diagramos duomenų bazės lentelę SD_TRANSITION

5. žingsnis Išskirtų elementų apjungimas.Atrinkus būsenas ir perėjimus visi elementai yra sujungiami tarpusavyje ir gaunama būsenų diagrama tam objektui, kurį pasirinko vartotojas. Visa tai atvaizduojama SD projekte.

3.3.Algoritmo žingsnių aprašymas

Šiame skyriuje pateikiamas formalizuotas IS būsenų modelio generavimo veiklos modelio pagrindu algoritmo žingsnių aprašymas.

1, 2 žingsniai



Šaltinis: sudaryta autoriaus.

22 pav. Algoritmo pirmas ir antras žingsniai

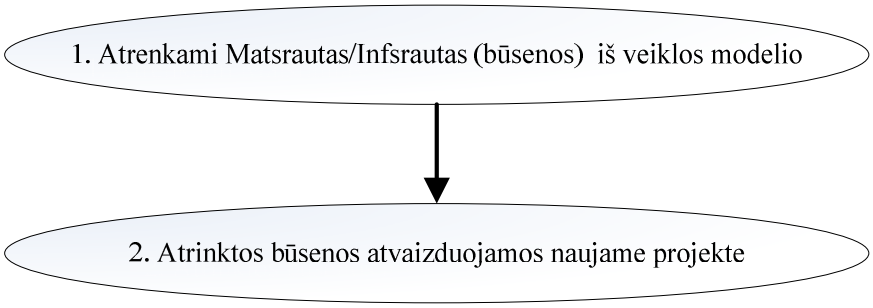
1. Būsenų diagramos kūrimas (generavimas) prasideda tuomet, kai vartotojas nurodo objektą, kuriam remiantis veiklos modelio saugyklos duomenimis norima atlikti generavimą. Šį objektą vartotojas pasirenka iš veiklos modelio duomenų bazės lentelės SD_SRAUTO_OBJEKTAS ir tuomet pasirinkimas yra perkeliamas į būsenų diagramos duomenų bazės lentelę SD_STATE_MACHINE. Tarkime objekto pav. B
2. Pasirinkus objektą, kuriam bus generuojama būsenų diagrama yra sukuriamas naujas projektas būsenų diagramai, kurį pavadinkime SD.
3. Naujai sukurtame projekte yra sukuriamas diagramos atskaitos taškas Initial pint

Žemiau pateiktame 23 pav. Pavaizduota būsenų diagrama po pirmo ir antro žingsnio.

Šaltinis: sudaryta autoriaus.

23 pav. Būsenų SD diagrama po 1 ir 2 žingsnių

3 žingsnis

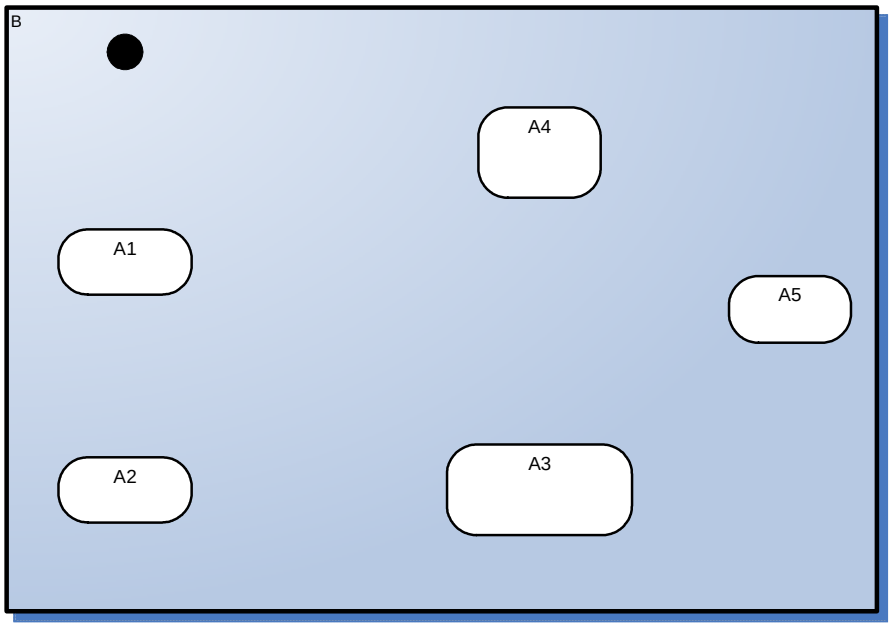


Šaltinis: sudaryta autoriaus.

24 pav. Algoritmo trečias žingsnis

4. Pagal vartotojo pasirinktą objektą iš veiklos modelio yra atrenkami materialūs ir informaciniai srautai, kurie yra susiję su pasirinktu Srauto objektu. Atrinkti duomenys surašomi būsenų diagramos duomenų bazės lentelę SD-STATE. Pavyzdžiui iliustruoti naudosime būsenas nuo A1 iki A5.
5. Atrinktos būsenos atvaizduojamos jau sukurtame projekte, kuris skirtas būsenų diagramai generuoti.

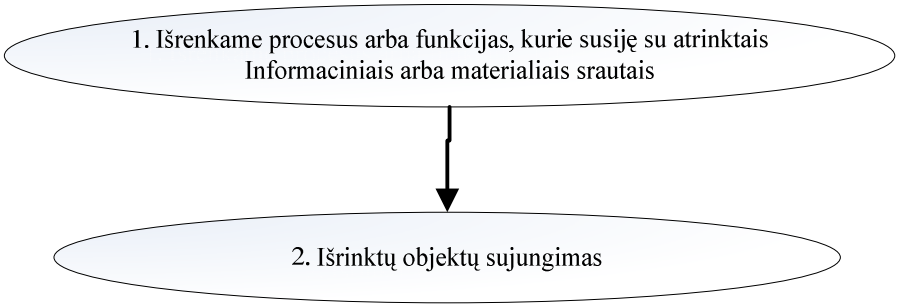
Žemiau pateiktame 25 pav. Pavaizduota būsenų diagrama po trečio žingsnio



Šaltinis: sudaryta autoriaus.

25 pav. Būsenų SD diagrama po 3 žingsnio

4, 5 žingsniai

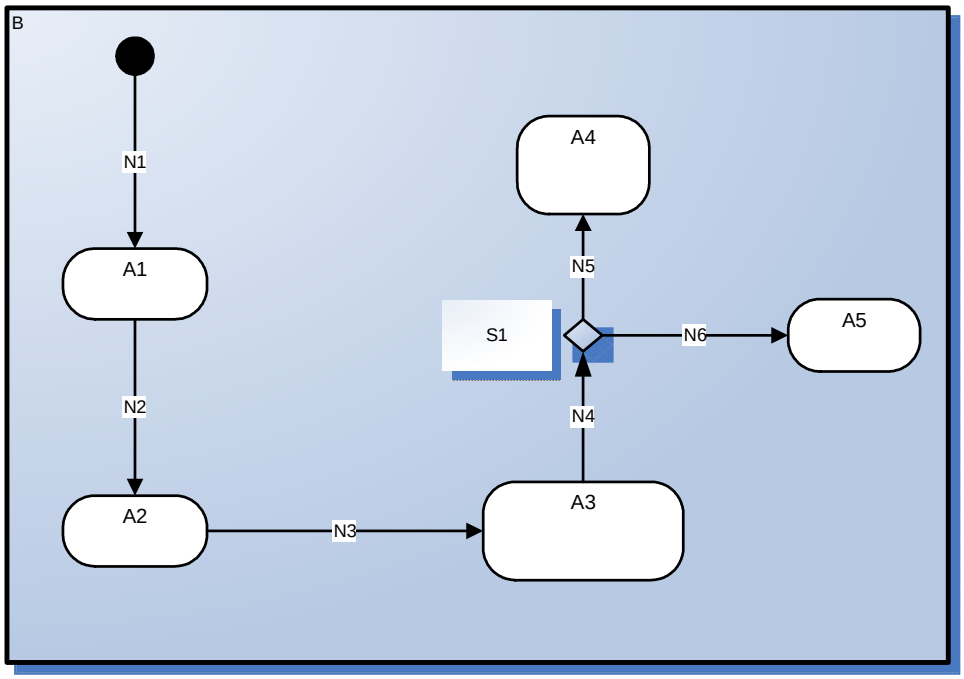


Šaltinis: sudaryta autoriaus.

26 pav. Algoritmo ketvirtas ir penktas žingsniai

1. Iš veiklos modelio atrenkami procesai arba funkcijos, kurie yra susiję su atrinktais informaciniais arba materialiais srautais. Atrinkimui naudosime veiklos modelio duomenų bazės lenteles: funkcija, procesas. Atrinktus duomenis surašome į būsenų diagramos lentelę SD_TRANSITION
2. Atrinktus duomenis, kurie yra surašyti į būsenų diagramos lenteles: SD_STATE_MACHINE, SD_TRANSITION ir SD_STATE algoritmas apjungia tarpusavyje pagal įrašų id.

Žemiau pateiktame 27 pav. Pavaizduota būsenų diagrama po paskutinio algoritmo žingsnio

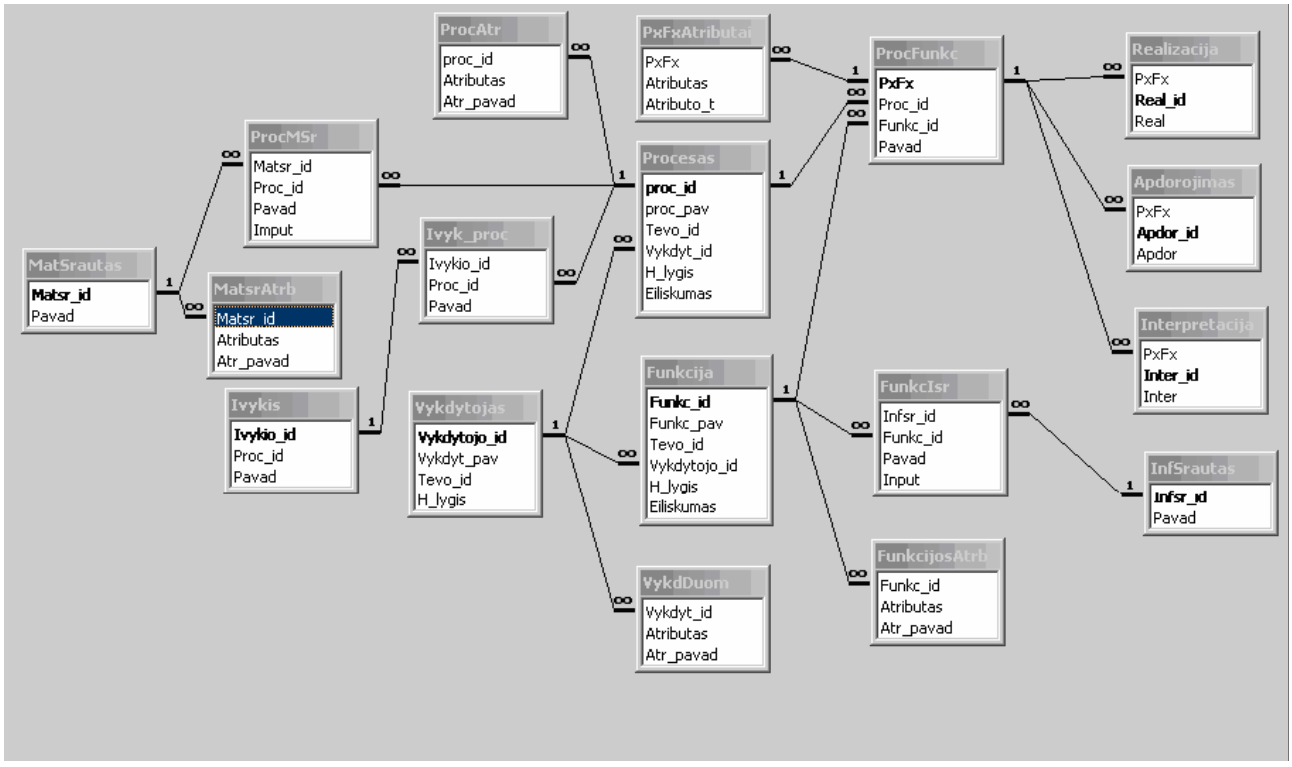


Šaltinis: sudaryta autoriaus.

27 pav. Būsenų SD diagrama po 4 ir 5 žingsnių

3.4. Būsenų (state) diagramai generuoti naudojama duomenų struktūra

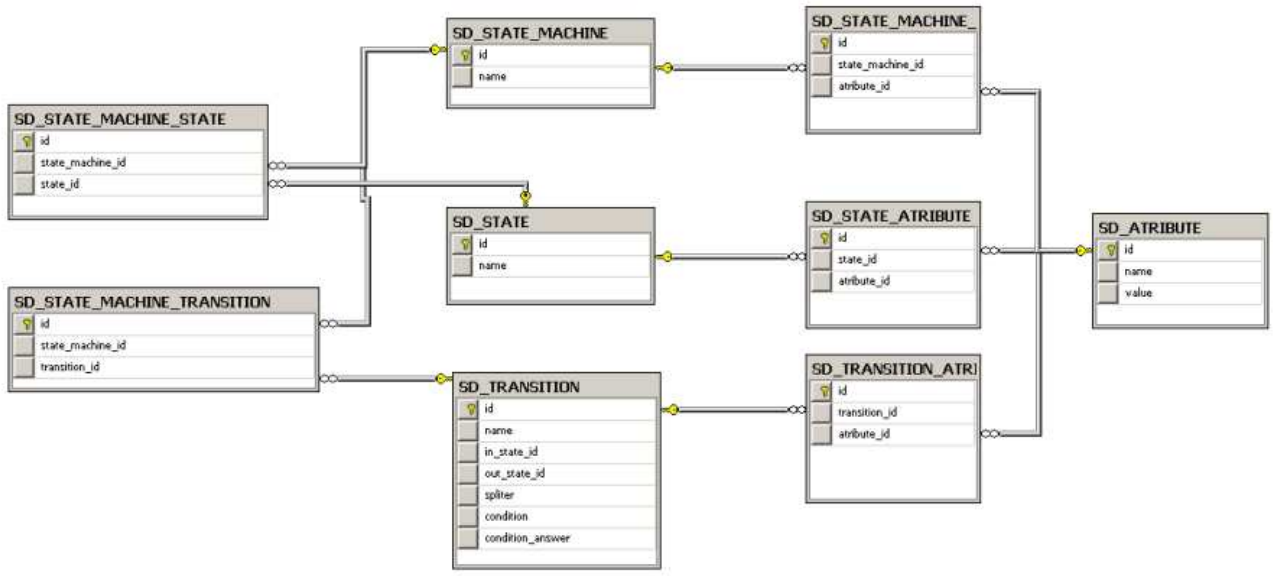
Eksperimentinės duomenų bazės struktūra, kuria remiantis galima sugeneruoti bendrus būsenų (state) diagramos elementus. Detalesni laukų aprašymai pateikiami 1 priede.



Šaltinis: sudaryta autoriaus.

28 pav. Eksperimentinės duomenų bazės struktūra

Duomenų bazės struktūra, kuri skirta būsenų (state) diagramai generuoti pateikta 29 pav. Ši duomenų bazė buvo kuriama atskirai, dėl to, kad būtų laisviau prieinami duomenys.

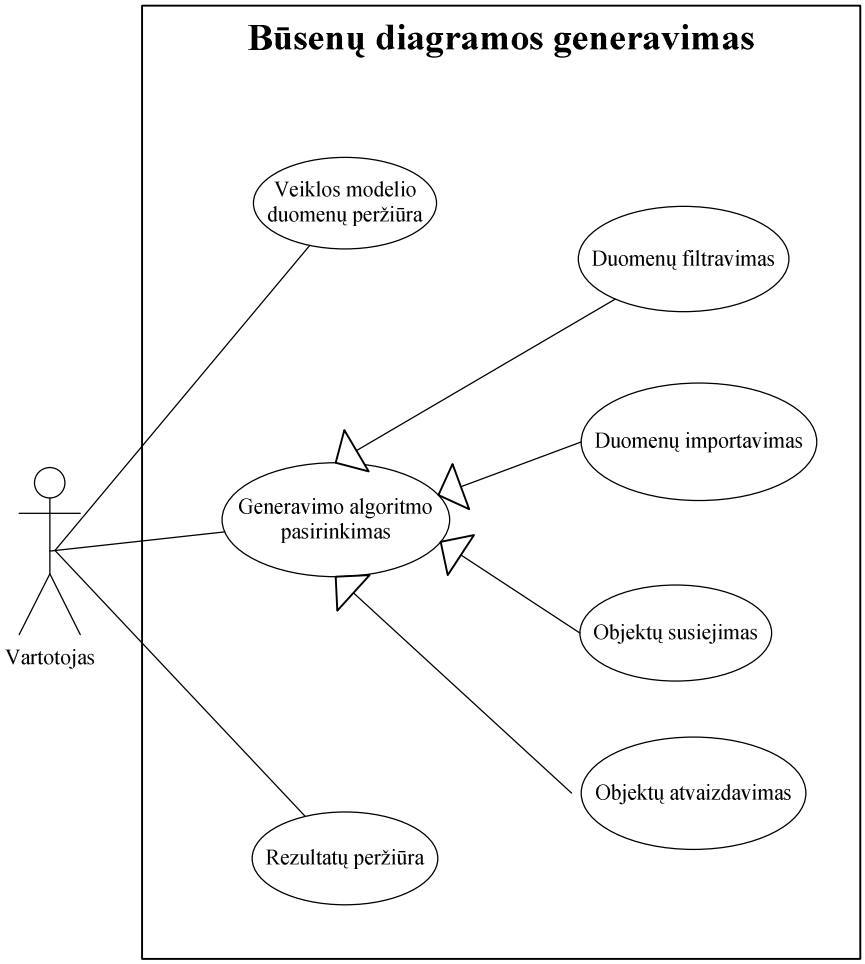


Šaltinis: sudaryta autoriaus.

29 pav. Eksperimentinė duomenų bazės struktūra skirta būsenų diagramai generuoti

3.5. Būsenų (state) diagramos generavimo elgsenos modelis

Žvelgdami į panaudojimo atvejų diagramą matome, kad vartotojas turi turėti teisę peržiūrėti ir pasirinkti norimą objektą iš veiklos modelio. Taipogi turi galimybę pasirinkti generavimo algoritmą ir peržiūrėti jo rezultatus. Visi likę procesai yra nepriklausomi nuo vartotojo, viską atliks generavimo algoritmas

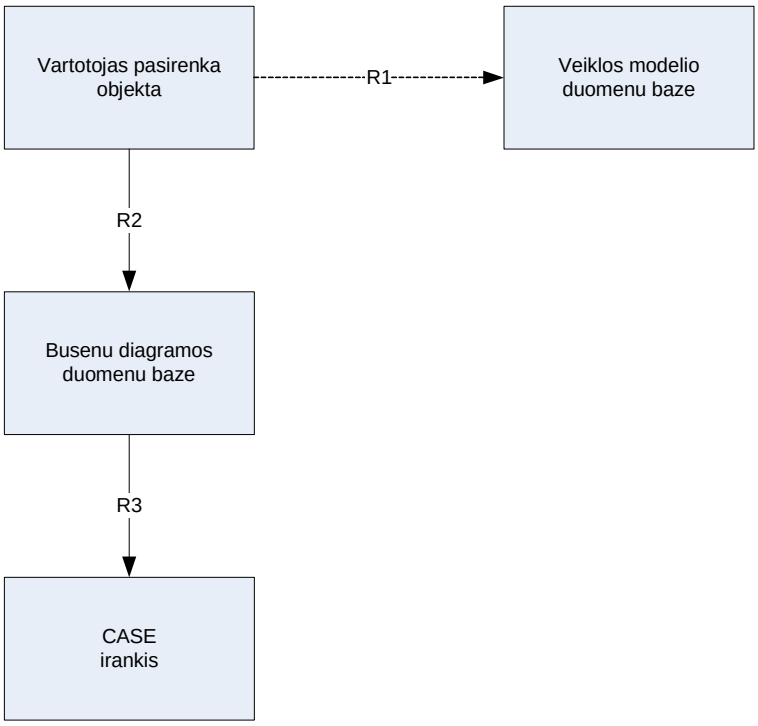


Šaltinis: sudaryta autoriaus.

30 pav. Būsenų (state) diagramos generavimo vartojimo atvejų diagrama

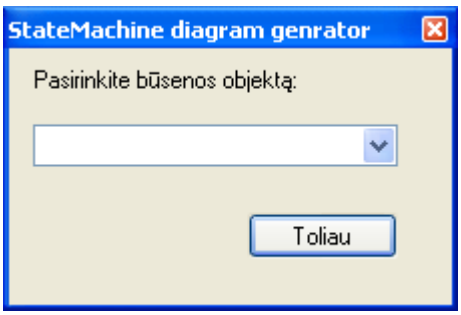
3.6. Algoritmą realizuojančio įrankio prototipas

Žemiau pateiktame 31 pav. pavaizduota algoritmą realizuojančio įrankio prototipo veikimo žingsniai. Pirmiausiai vartotojas, naudodamasis forma pavaizduota 32 pav. iš veiklos modelio duomenų bazės pasirenka objektą, kuriam bus generuojam būsenų (state) diagrama.



Šaltinis: sudaryta autoriaus.

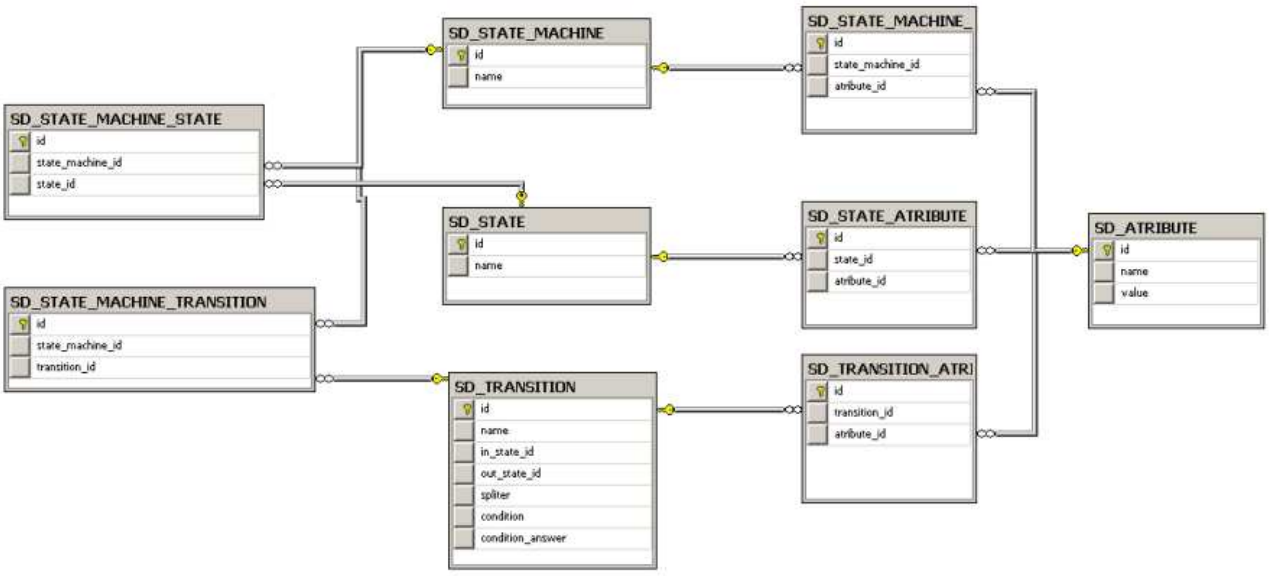
31 pav. Algoritmą realizuojančio prototipo žingsniai



Šaltinis: sudaryta autoriaus.

32 pav. Algoritmą realizuojančio prototipo pagrindinė forma

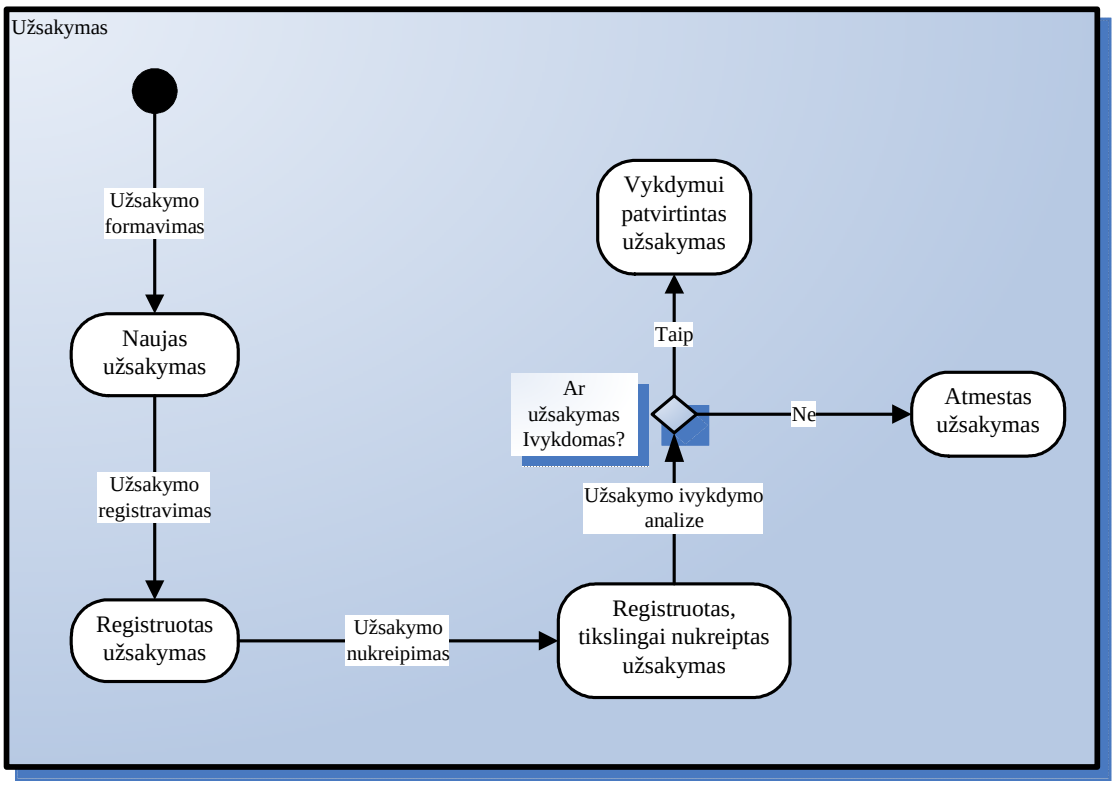
Vartotojo pasirinkimas vyksta naudojant R1 ryšį. Vartotojui pasirinkus objektą iš veiklos modelio duomenų bazės yra atrenkami visi elementai, kurie bus naudojami būsenų diagramai generuoti ir perrašomi į būsenų diagramos duomenų bazę, kuri pavaizduota 33 pav.



Šaltinis: sudaryta autoriaus.

33 pav. Užpildyta būsenų diagramos duomenų bazė

Algoritmo ir Case įrankio pagalba iš užpildytos būsenų diagramos duomenų bazės yra generuojama būsenų diagrama, kuri pavaizduota 34 pav.

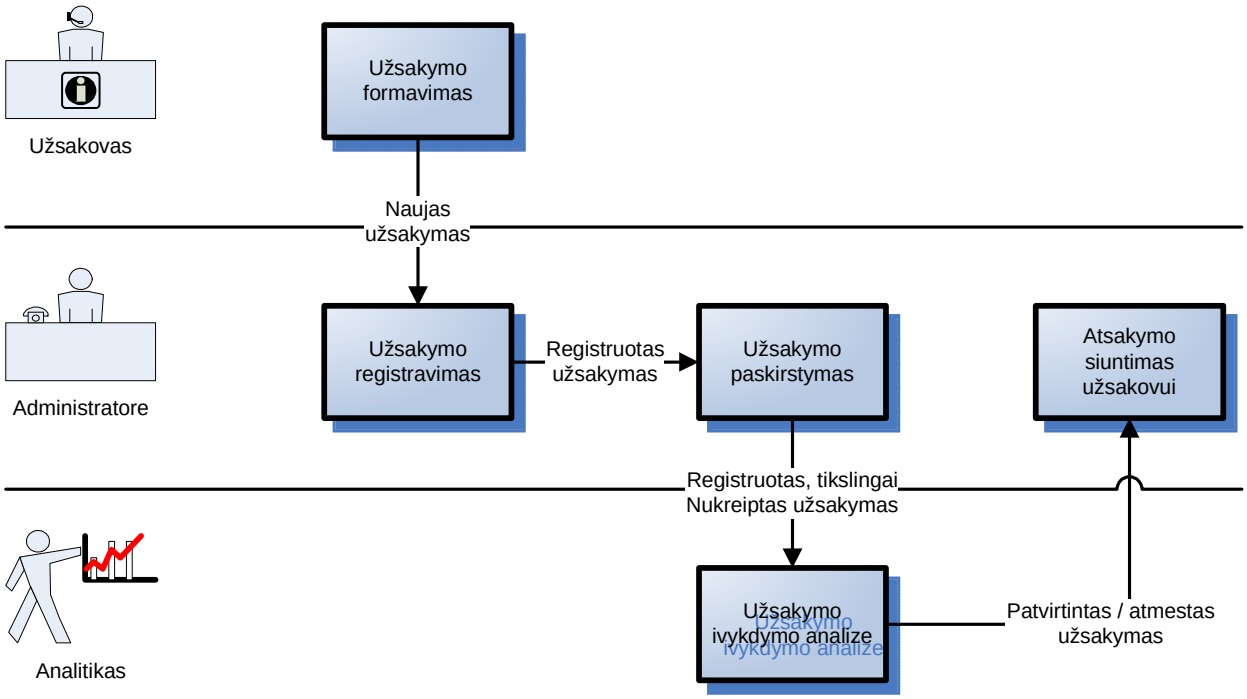


Šaltinis: sudaryta autoriaus.

34 pav. Sugeneruota būsenų diagrama

3.7. Eksperimento patvirtinimas

Norint patikimai patvirtinti modelį ar metodą, pagal statistinių tyrimų principus reikėtų didelės reliacinės aibės, tačiau informacinių sistemų ir programų inžinerijos srityje tą galima pasiekti, pateikiant pakankamai (vieną arba diaugiau) reprezentatyvių pavyzdžių. Remdamiesi šiuo teiginiu patikrinsime sudaryto būsenų (state) diagramos generavimo algoritmo teisingumą.



Šaltinis: sudaryta autoriaus.

35 pav. Eksperimentiniai duomenys

Patikrinimą atliksime su pavyzdinių duomenų aibe. Duomenis naudosime iš eksperimentui pasirinkto duomenų sekų diagramos 35 pav. Naudodami tuos duomenis užpildome duomenų bazės lenteles, kurios bus naudojamos testavimui atlikti.

1,2 žingsniai

	id	name
	1	Užsakymas
▶*	NULL	NULL

Šaltinis: sudaryta autoriaus.

36 pav. SD_STATE_MACHINE lentelė

Vartotojas iš veiklos modelio duomenų bazės lentelės SD_SRAUTO_OBJEKTAS pasirenka objektą (šiuo atveju Užsakymas), kuriam nori sugeneruoti būsenų diagramą. Duomenys algoritmo pagalba automatiškai perkeliami į būsenų diagramos duomenų bazės lentelę SD_STATE_MACHINE, kuri pavaizduota 36 pav., ir kuri bus naudojama būsenų diagramai

generuoti. Pasirinkus objektą sukuriamas naujas būsenų diagramos projektas ir atvaizduojamas atskaitos taškas (initial pint). Gautas rezultatas pavaizduotas 37 pav.

Šaltinis: sudaryta autoriaus.

37 pav. Būsenų diagrama po 1 ir 2 žingsnio

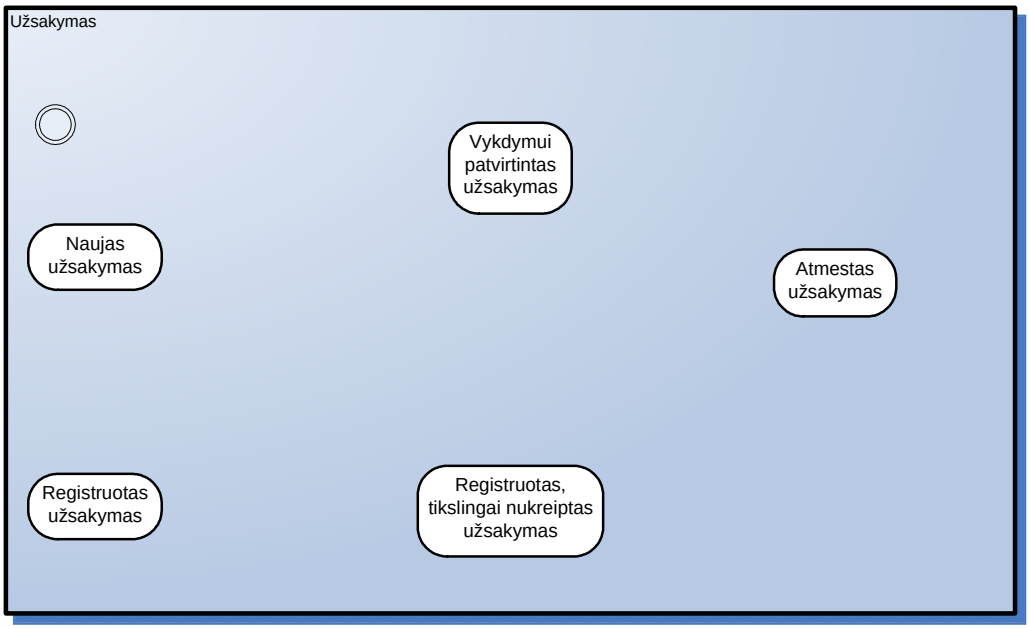
3 žingsnis

	id	name
	1	Registruotas užsakymas
▶	2	Registruotas, nukreiptas užsakymas
	3	Vykdymui patvirtintas užsakymas
	4	Atmestas užsakymas
	5	Naujas užsakymas
*	NULL	NULL

Šaltinis: sudaryta autoriaus.

38 pav. SD_STATE lentelė

Pagal vartotojo pasirinktą objektą (šiuo atveju Užsakymas) iš veiklos modelio duomenų bazės lentelių: Infsrautas ir Matsrautas yra atrenkami materialūs ir informaciniai srautai, kurie yra susiję su pasirinktu objektu. Atrinkti duomenys algoritmo pagalba yra surašomi į būsenų diagramos duomenų bazės lentelę SD_STATE 38pav. Iliustruojant pavyzdinius duomenis yra atrenkamos 5 būsenos: Naujas užsakymas – 5(id), registruotas užsakymas - 1(id), registruotas/tikslingai nukreiptas užsakymas - 2(id), vykdymui patvirtintas užsakymas - 3(id) ir atmestas užsakymas 4(id), kurios atvaizduojamos jau sukurtame būsenų diagramos projekte 39 pav.



Šaltinis: sudaryta autoriaus.

39 pav. Būsenų diagrama po 3 žingsnio

4,5 žingsniai

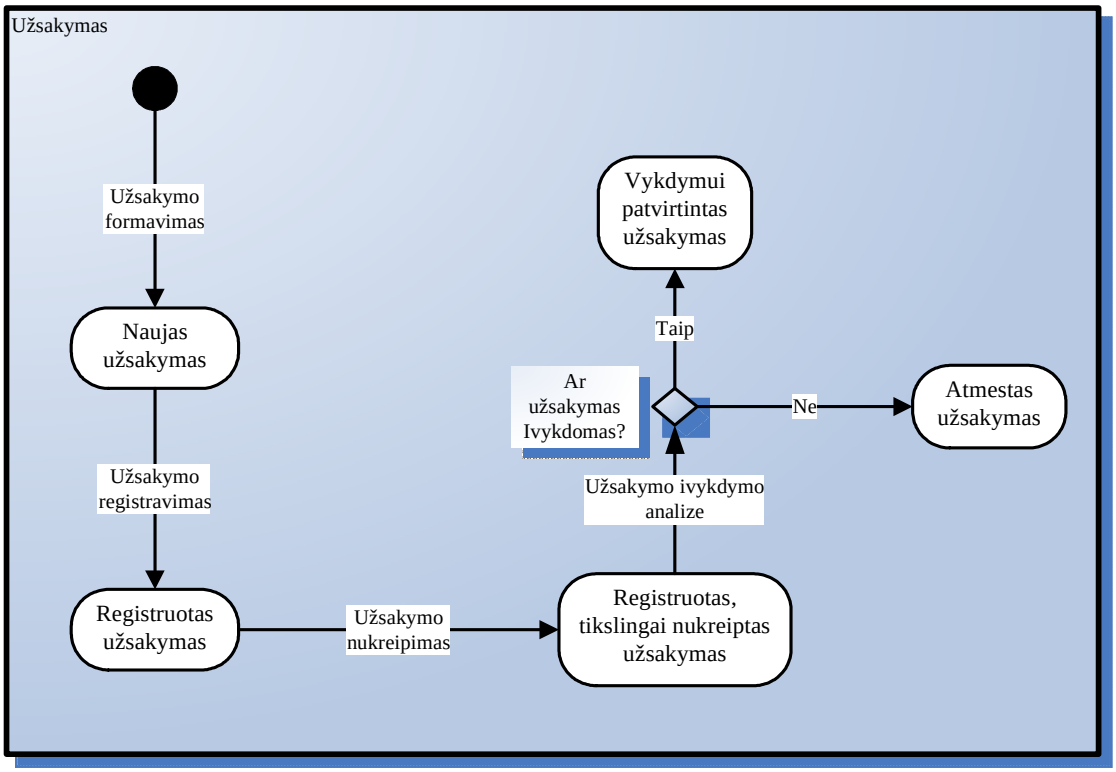
	id	name	in_state_id	out_state_id	spliter	condition	condition_answer
	1	Užsakymo forma...	5	-1	False	NULL	NULL
	2	Užsakymo regist...	5	1	False	NULL	NULL
▶	3	Užsakymo nukrei...	1	2	False	NULL	NULL
	4	Užsakymo ivykd...	2	3	True	Ar užsakymas iv...	Taip
	5	Užsakymo ivykd...	2	4	True	Ar užsakymas iv...	Ne
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Šaltinis: sudaryta autoriaus.

40 pav. SD_TRANSITION lentelė

Pagal atrinktus informacinius ir materialius srautus algoritmas iš veiklos modelio duomenų bazės lentelių: Procesas arba Funkcija atrenka procesus ir funkcijas, kurie yra susiję su atrinktais srautais. Algoritmas automatiškai surašo atrinktus duomenis į būsenų diagramos duomenų bazės lentelę SD_TRANSITION 41 pav. Iliustruojant pavyzdinius duomenis yra atrenkami 5 perėjimai.

Atrinkti perėjimai atvaizduojami jau sukurtame būsenų diagramos projekte ir pagal papildomas sąlygas (in_state_id, out_state_id, Splite, condition, condition_answer) išskirti elementai tarpusavyje sujungiami. Išskirtų elementų apjungimas pavaizduotas 41 pav.



Šaltinis: sudaryta autoriaus.

41 pav. Būsenų diagrama po 4 ir 5 žingsnių

IŠVADOS IR PASIŪLYMAI

1. Atlikus tyrimą, nustatyta kad būsenų modelio generavimas intelektualizuoja IS inžinerijos projektavimo etapą tuo, kad UML būsenų modelis sukuriamas interaktyviai, t.y. vykdant generavimo iš veiklos modelio (kuriame saugomos dalykinės srities žinios yra patikrintos valdymo požiūriu) algoritmą dalyvaujant analitikui ar projektuotojui.
2. Išanalizavus veiklos metamodelio ir būsenų metamodelio sudėtį nustatyta, kad Veiklos metamodelį reikia papildyti įterpiant srauto objektą UML būsenų modelio generavimui.
3. Viena iš priežasčių, kuri apriboja CASE sistemų funkcionalumą (pavyzdžiui yra ribotas projektinių modelių ir programinio kodo generavimo galimybės) yra nepakankamos empiriniu būdu sudarytų veiklos modelių savybės ir nepakankamos vartotojo informacinių reikalavimų specifikacijų savybės. Šią problemą tikslinga spręsti taikant veiklos metamodelį, kuriame saugomos valdymo požiūriu patikrintos dalykinės srities žinios UML modelių (šiuo atveju būsenų modelio) generavimui.
4. Sukurtas UML būsenų (state) generavimo algoritmas, buvo patikrintas realiu pavyzdžiu.
5. Sukurtam UML būsenų (state) generavimo algoritmui pasiūlytas realizavimo prototipas.

LITERATŪRA

1. Gudas S., Lopata A. Žiniomis grindžiamos informacijos sistemų inžinerijos bruožai, 2005
2. A. Lopata Veiklos modelių sudėties analizė [žiūrėta 2008 m. balandžio 19 d.].] prieiga per internetą http://www.ktu.lt/lt/apie_renginius/konferencijos/2006/k6_02/IT2002/XI_sekcija.pdf
3. K.D.Tham. CIM-OSA: Enterprise modelling. –Enterprise integration laboratory, University of Toronto [Interaktyvus]. [žiūrėta 2007 m. kovo 19 d.]. Prieiga per internetą: www.ie.utoronto.ca/EIL/entmethod/cimosa
4. Universal Enterprise modeling language, IFAC-IFIP Task Force, [Interaktyvus]. [žiūrėta 2007 m. Gegužės 19 d.]. Prieiga per internetą: http://www.cit.gu.edu.au/~bernus/taskforce/archive/UEML-TF-IG.ppt
5. Wikipedia UML [interaktyvus]. [žiūrėta 2007 m. birželio 19 d.]. Prieiga per Internetą: http://en.wikipedia.org/wiki/Unified_Modeling_Language
6. UML Tutorial [interaktyvus]. [žiūrėta 2007 m. birželio 11 d.]. Prieiga per internetą: http://atlas.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/state.htm
7. A. Lopata. VEIKLOS MODELIŲ SUDĖTIES ANALIZĖ. Informacinės technologijos ir valdymas, 2000.
8. Wikipedia (2006) database [interaktyvus]. [žiūrėta 2006 m. lapkričio 20 d.]. Prieiga per internetą http://en.wikipedia.org/wiki/Data
9. OMG (2005) Unified Modeling Language: Superstructure [interaktyvus]. [žiūrėta 2008 m. vasario 20 d.]. Prieiga per internetą http://www.omg.org/docs/formal/05-07-04.pdf
10. S. Gudas, A. Lopata, Workflow models based acquisition of enterprise knowledge, INFORMATION TECHNOLOGY AND CONTROL, 2007, Vol.36, No.1A, [žiūrėta 2008 m. Sausio 19 d.].] prieiga per internetą http://itc.ktu.lt/itc361/Gudas361.pdf
11. OMG (2006) UML [interaktyvus]. [žiūrėta 2006 m. lapkričio 19 d.]. Prieiga per internetą: http://www.ceeol.com/aspx/getdocument.aspx?logid=5&id=8af01929-ec6d-4156-b763-6738c7ced62a
12.
13. M. RAFFAI, Enterprise application integration, a metamodel approach, [žiūrėta 2008 m. Kovo 25 d.].] prieiga per internetą http://www.sea.uni-linz.ac.at/idimt2007/SessionF.pdf
14. M. S. Abdullah, A. Evans, I. Benest, Ch. Kimble, Department of Computer Science, University of York, United Kingdom, Developing a UML Profile for Modelling Knowledge-Based Systems [žiūrėta 2008 m. Balandžio 10 d.].] prieiga per internetą http://wwwusers.cs.york.ac.uk/~kimble/research/UML_Profile_for_Modelling_KBS.pdf

15. Business Process Modeling Notation (BPMN), [žiūrėta 2008 m. Balandžio 10 d.].] prieiga per internetą http://www.workflownp.org.uk/fileupload/upload/BPMN-V1.01882004261512.pdf
16. Straipsniai ISI duomenų bazėje, Gudas S., Lopata A., Skersys T. Approach to Enterprise Modelling for Information Systems Engineering. INFORMATICA, Vol. 16, No. 2, Institute of Mathematics and Informatics, Vilnius, 2005, pp. 175-192., 2005
17. Unified Modeling Language: Superstructure, version 2.0. [žiūrėta 2008 m. Gegužės 11 d.].] prieiga per internetą http://www.omg.org/docs/formal/05-07-04.pdf
18. S. A. White, IBM Corporation, Introduction to BPMN, [žiūrėta 2008 m. Gegužės 22 d.].] prieiga per internetą http://www.bpmn.org/Documents/Introduction%20to%20BPMN.pdf
19. Gudas S. Lopata A., Skersys T. Framework for knowledge based IS Enginering, ADVIS, 2004
20. European Committee for Standardization. ENV 40 003: Computer Integrated Manufacturing – Systems Architecture - Framework for Enterprise Modelling, CEN/CENELEC, 2006.03.28.

PRIEDAI

1 Priedas

Veiklos modelio klasių diagramos lentelių aprašymas

Procesas		
Atributas	**Tipas**	**Atributų aprašymas**
Proc_id	Integer	Proceso unikalus numeris (identifikatorius)
Proc_pav	Tekstas	Proceso pavadinimas
Tevo_id	Integer	Proceso aukštesnio proceso („tėvo“) ID
Vykdyt_id	Integer	Vykdytojo, kuris vykdo procesą, ID
H_lygis	Integer	Proceso hierarchijos lygis
Eiliškumas	Integer	Proceso vykdymo numeris

Lentelėje **Procesas** saugomi organizacijos procesai.

ProcAtr		
Atributas	**Tipas**	**Atributų aprašymas**
Proc_id	Integer	Proceso unikalus numeris (identifikatorius)
Atributas	Tekstas	Proceso atributo pavadinimas
Atr_pavad	Tekstas	Proceso atributo reikšmė

Lentelėje **ProcAtr** saugomi proceso atributai.

Funkcija		
Atributas	**Tipas**	**Atributų aprašymas**
Funkc_id	Integer	Funkcijos unikalus numeris (identifikatorius)
Funkc_pav	Tekstas	Funkcijos pavadinimas
Tevo_id	Integer	Funkcijos aukštesnės funkcijos („tėvo“) ID
Vykdyt_id	Integer	Vykdytojo, kuris vykdo funkcija, ID
H_lygis	Integer	Funkcijos hierarchijos lygis
Eiliškumas	Integer	Funkcijos vykdymo numeris

Lentelėje **Funkcija** saugomos organizacijos funkcijos.

FunkcijosAtrb		
Atributas	**Tipas**	**Atributų aprašymas**
Funkc_id	Integer	Funkcijos unikalus numeris (identifikatorius)
Atributas	Tekstas	Funkcijos atributo pavadinimas
Atr_pavad	Tekstas	Funkcijos atributo reikšmė

Lentelėje **FunkcijosAtrb** yra saugomi funkcijose atributai.

Vykdytojas		
Atributas	**Tipas**	**Atributų aprašymas**
Vykdyt_id	Integer	Vykdytojo unikalus numeris (identifikatorius)
Vykdyt_pav	Tekstas	Vykdytojo pavadinimas
Tevo_id	Integer	Aukštesnio lygio vykdytojo („tėvo“) ID
H_lygis	Integer	Vykdytojo hierarchijos lygis

Lentelėje **Vykdytoja**s saugomi organizacijos skyriai, padaliniai, darbuotojai.

VykdDuom		
Atributas	**Tipas**	**Atributų aprašymas**
Vykdyt_id	Integer	Vykdytojo unikalus numeris (identifikatorius)
Atributas	Tekstas	Vykdytojo atributo pavadinimas
Atr_pavad	Tekstas	Vykdytojo atributo reikšmė

Lentelėje **VykdDuom** yra saugomi darbuotojų papildoma informacija. (telefonai, adresai ir t.t)

MatSrautas		
Atributas	**Tipas**	**Atributų aprašymas**
Matsr_id	Integer	Materialaus srauto unikalus numeris (identifikatorius)
Pavad	Tekstas	Materialaus srauto pavadinimas

Lentelėje **MatSrautas** yra saugomas materialus srautas (žaliavos, pusgaminiai ir t.t).

MatsrAtrb		
Atributas	**Tipas**	**Atributų aprašymas**
Matsr_id	Integer	Materialaus srauto unikalus numeris (identifikatorius)
Atributas	Tekstas	Materialaus srauto atributo pavadinimas
Atr_pavad	Tekstas	Materialaus srauto atributo reikšmė

Lentelėje **MatsrAtrb** yra saugomi papildomi materialaus srauto atributai (pvz: ar žaliavos matuojamas kilogramais, ar metrais ir t.t).

InfSrautas		
Atributas	**Tipas**	**Atributų aprašymas**
Infsr_id	Integer	Informacinio srauto unikalus numeris (identifikatorius)
Pavad	Tekstas	Informacinio srauto pavadinimas

Lentelėje **InfSrautas** yra saugomas informacinis srautas (pvz: dokumentacija einanti su materialiu srautu ir t.t).

ProcMSr		
Atributas	**Tipas**	**Atributų aprašymas**
Matsr_id	Integer	Materialaus srauto ID
Proc_id	Integer	Proceso ID
Pavad	Tekstas	Srauto pavadinimas
Input	Tekstas	Požymis ar materialus srautas yra proceso įėjimas ar išėjimas

Lentelėje **ProcMSr** yra saugojamas procesų-materialaus srauto sąrašas. Jis parodo proceso naudojamą materialų srautą ir šio srauto tipą. (Ar tai proceso įėjimas ar išėjimas).

FunkcIsr		
Atributas	**Tipas**	**Atributų aprašymas**
Infsr_id	Integer	Informacinio srauto ID
Funkc_id	Integer	Funkcijos ID
Pavad	Tekstas	Srauto pavadinimas
Input	Tekstas	Požymis ar informacinis srautas yra funkcijos įėjimas ar išėjimas

Lentelėje **FunkcIsr** yra saugojamas funkcijų – informacinio srauto sąrašas. Jis parodo funkcijų naudojamą informacinį srautą ir šio srauto tipą. (Ar tai funkcijos įėjimas ar išėjimas).

Ivykis		
Atributas	**Tipas**	**Atributų aprašymas**
Ivykio_id	Integer	Įvykio unikalus numeris (identifikatorius)
Ivykio_pav	Integer	Įvykio pavadinimas
Proc_id	Tekstas	Proceso ID

Lentelėje **Ivykis** yra saugomi įvykiai: (Įvykis paleidžia konkrečius procesus).

Ivyk_Proc		
Atributas	**Tipas**	**Atributų aprašymas**
Ivykio_id	Integer	Įvykio ID
Proc_id	Integer	Proceso ID
Pavad	Tekstas	Pavadinimas

Lentelėje **Ivyk_Proc** yra saugomas įvykių – procesų sąrašas. Jis parodo kokie įvykiai sužadina procesus.

ProcFunkc		
Atributas	**Tipas**	**Atributų aprašymas**
PxFx	Integer	Proceso-Funkcijos sankirtos unikalus numeris (identifikatorius)
Proc_id	Integer	Proceso ID
Funkc_id	Integer	Funkcijos ID
Pavad	Tekstas	Sankirtos pavadinimas

Lentelėje **ProcFunkc** yra susiejami procesai su funkcijomis.

PxFx		
Atributas	**Tipas**	**Atributų aprašymas**
PxFx	Integer	Proceso-Funkcijos sankirtos ID (identifikatorius)
Atributas	Tekstas	Sankirtos atributas
Atributo_t	Tekstas	Sankirtos atributo tipas

Lentelėje **PxFx** saugomi sankirtos atributai, bei jų tipai.

Interpretacija		
Atributas	**Tipas**	**Atributų aprašymas**
PxFx	Integer	Proceso-Funkcijos sankirtos ID (identifikatorius)
Interpr_id	Integer	Interpretacijos unikalus numeris (identifikatorius)
Interpr	Tekstas	Interpretacija

Realizacija		
Atributas	**Tipas**	**Atributų aprašymas**
PxFx	Integer	Procesų ir funkcijų sankirtos ID
Real_id	Integer	Realizacijos unikalus numeris (identifikatorius)
Real	Tekstas	Realizacija

Apdorojimas		
Atributas	**Tipas**	**Atributų aprašymas**
PxFx	Integer	Procesų ir funkcijų sankirtos ID
Apdor_id	Integer	Apdorojimo unikalus numeris (identifikatorius)
Apdor	Tekstas	Apdorojimas

State Machine diagrama