KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

INFORMACINIŲ SISTEMŲ KATEDRA

Daiva Danielaitytė

UML veiklos modelio generavimas veiklos žinių saugyklos pagrindu

Magistro darbas

Darbo vadovas

dr. A. Lopata

Kaunas, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

INFORMACINIŲ SISTEMŲ KATEDRA

Daiva Danielaitytė

Enterprise kowledge model based generation of the UML activity model

Magistro darbas

Darbo vadovas

dr. A. Lopata

Kaunas, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

INFORMACINIŲ SISTEMŲ KATEDRA

Daiva Danielaitytė

UML veiklos modelio generavimas veiklos žinių saugyklos pagrindu

Magistro darbas

Recenzentas

2008-01-17

Vadovas

dr.A. Lopata

2008-01-17

Atliko

IFM-2/4 gr. stud.

Daiva Danielaitytė

2008-01-17

Kaunas, 2008

TURINYS

ABSTRACT 6
ĮVADAS 7
1. Organizacijos veiklos modeliu paremtų Informacinių Sistemų analizė 8
1.1. Veiklos žiniomis pagrįstos Informacinės Sistemos 8
1.2. Žinių modelis Informacinių Sistemų Inžinerijoje 11
2. Veiklos modelio sudarymas žinių saugyklos pagrindu 14
2.1. Formalizuotas UML Veiklos (activity) diagramos metamodelio aprašymas 14
2.2. Formalizuotas veiklos modelio aprašymas 17
2.3. Veiklos (Activity) diagramos formalaus modelio generavimas 20
2.4. Veiklos modelio ir veiklos (activity) diagramos modelio sąsaja 21
3. Business Process Modeling ir Unified Modeling Language grafinių notacijų palyginimas 23
3.1. Nesugeneruojamų veiklos (activity) diagramos elementų išskyrimas 25
3.1.1. Dėl notacijos nepilnumo 25
3.1.2. Dėl duomenų saugyklos nepilnumo 28
4. Informacinės Sistemos veiklos (activity) modelio generavimo algoritmas 30
4.1. Formalizuotas algoritmo aprašymas 31
5. Veiklos (activity) diagrmai generuoti naudojama duiomenų struktūra 40
6. Veiklos (activity) diagramos generavimo elgsenos modelis 41
6.1. Panaudojimo atvėjų modelis 41
6.2. Sekų diagrama 42
7. Algoritmą realizuojančio įrankio prototipas 43
8. Rezultatų patvirtinimo eksperimentas 44
9. Išvados 51
LITERATŪRA 52

PAVEIKSLĖLIŲ SĄRAŠAS

1 pav. Principinė organizacijos veiklos meta-modelio schema 10
2 pav. CASE sistemos su žinių baze architektūra 13
3 pav. Veikos (Activity) diagramos metamodelis 17
4 pav. Veiklos metamodelio klasių modelis 18
5 pav. Veiklos modelio M1 grafinė schema 19
6 pav. Veiklos (Activity) modelio grafinė schema 21
7 pav. Veiklos modelio elementų atvaizdavimas į veiklos (activity) diagramos modelį 22
8 pav. IS veiklos (activity) modelio generavimo algoritmas 30
9 pav. Algoritmo 1 žingsnis 31
10 pav. Veiklos (activity) diagrama AD1 po pirmo žingsnio 32
11 pav. Algoritmo 2 žingsnis 32
12 pav. Veiklos (activity) diagrama AD1 po antro žingsnio 33
13 pav. Algoritmo 3 žingsnis 34
14 pav. Veiklos (activity) diagrama AD1 po trečio žingsnio 35
15 pav. Algoritmo 4 žingsnis 36
16 pav. Algoritmo 4 žingsnis, tęsinys 37
17 pav. Veiklos (activity) diagrama AD1 po ketvirto žingsnio 39
18 pav. Eksperimentinės duomenų bazės struktūra 40
19 pav. Pakoreguota eksperimentinės duomenų bazės struktūra 41
20 pav. Veiklos (activity) diagramos generavimo vartojimo atvejų diagrama 41
21 pav. Veiklos (activity) diagramos generavimo sekų diagrama 42
22 pav. Prototipo vykdomieji failai 43
23 pav. Veiklos (activity) diagramos generavimo funkcija įrankiu juostoje 43
24 pav. Veiklos (activity) diagramos generavimo funkcijos realizacija 44
25 pav. Sugeneruota pavyzdine veiklos (activity) diagrama 50

ABSTRACT

The principles and major steps of enterprise Meta-Model (EMM) based development of Activity model in CASE system enviroment are presented in this paper.

The Enterprise Meta-Model represents the key concepts of domain knowledge. The enterprise processes, management functions, and their interactions are concidered as a components of the domain knowledge accumulated as Enterprise model in the knowledge base of CASE system. The formal and practical background for generation of Activity diagram model is mapping rules of EMM constructs to constructs of Activity diagram meta-model. In this paper experimental Enterprise knowledge model is eveluated in order to find out with activity diagram elements we can‘t generate without adding aditional classes to Enterprise model.

ĮVADAS

Informacijos sistemos kūrimas, pradedant nuo vartotojo reikalavimų surinkimo iki IS diegimo, palaikymo ir reinžinerijos vadinamas informacijos sistemų inžinerija. Pastaruoju metu šio mokslo vystymosi eigoje formuojasi naujas etapas – žiniomis grindžiama kompiuterizuota IS inžinerija. IS kūrimo aplinka CASE yra papildoma veiklos žinių kaupimo posistemiu, intelektualizuojančiu informacijos sistemos kūrimo procesą. Veiklos žinių posistemio paskirtis teoriškai yra užtikrinti galimybę saugomų dalykinės srities žinių pagrindu generuoti IS konceptualaus ir detalaus projektavimo etapų modelius ir programinį kodą. Veiklos modelio panaudojimas gerina sprendimų kokybę ir taupomas darbo laikas, kadangi veiklos modelyje jau yra sukauptos formalių kriterijų atžvilgiu patikrintos žinios. Priešingai negu tradicinėje kompiuterizuotoje IS inžinerijoje, kur informacijos sistema kuriama empiriškai, pradedant vartotojo poreikių išsiaiškinimu, analize ir specifikavimu. Tokiu IS kūrimo būdu dauguma CASE įrankiams taikomų projektinių modelių yra generuojami tik iš dalies ir juos iki galo realizuoti gali tik sistemos analitikas neautomatizuotu būdu.

Šiame darbe bus gilinamasi į galimybę panaudojant veiklos modelį, kurio pagrindu sudarytas dalykinės srities žinių posistemis, generuoti veiklos (activity) diagramos projektinį modelį.

Šio darbo tikslas – iš veiklos metamodelio, formaliai apibrėžtos veiklos modelio struktūros, sugeneruoti UML projektinį veiklos (activity) modelį. Projektinio medelio generavimui pasinaudojant Kauno Technologijos universiteto, informatikos fakulteto Informacijos sistemų katedroje sukurto meta-modelio pavyzdžiu.

1. Organizacijos veiklos modeliu paremtų Informacinių Sistemų analizė

1.1. Veiklos žiniomis pagrįstos Informacinės Sistemos

Žinių naudojimas ir valdymas daugeliui organizacijų tapo komercine būtinybe, su tikslu valdyti savo intelektualinį kapitalą ir įgyti konkurencinį pranašumą. Dauguma žinių yra saugoma žmonių atmintyje ir jų panaudojimas rodosi labiau į žmogų, o ne technologija orientuotas procesas, tačiau technologija įgalina organizacijos žinių valdymą pasitelkiant automatizuotus įrankius, kurių vienas žinomiausiu – žiniomis grindžiamos sistemos (angl. kowlege-based systems – KBS). Nors daugelis ankstesnių žiniomis grindžiamų sistemų, kiekvieną kartą prireikus naujos, sugebančios sąveikauti su kitomis organizacijoje veikiančiomis, turėjo būti pradedamos kurti nuo pradžios, pavyzdžiui, ekspertinės sistemos – sistemos sukurtos remiantis vieno ar kelių ekspertų žiniomis. Pirmos kartos ekspertinėse sistemose eksperto žinios į sistemos žinių bazę buvo perkeliamos taisyklių pavidalu, sistemoje atsispindimų sudėtingu kodu, pagal kurį sunku suprasti suformuotų taisyklių ryšius ir sąsajas. Taip sukuriant problemą, atnaujinti žinių bazę reikalingos didelės pastangos atpažįstant saugomas taisykles, jų atnaujinimui.

Dabar žinių inžinerija nebėra tik žinių išgavimas iš eksperto galvos, ji apima žinių surinkimo, modeliavimo, atvaizdavimo ir panaudojimo metodus ir technologijas. Šis pasikeitimas įgalino pakartotinai panaudoti žinias skirtingais tos pačios probleminės srities atvejais. Žiniomis grindžiamos sistemos gali būti panaudotos, kaip technologinė priemonė tiek tikslių tiek neapibrėžtų organizacinių žinių surinkimui ir valdymui. Žiniomis grįstos sistemos kuriamos panaudojant žinių inžinerijos metodus. Žinių inžinerijos metodai turi panašumų su programinės inžinerijos metodais, bet juose orientuojamasi į žinių, o ne į duomenų ar informacijos apdorojimą.

IT požiūriu žiniomis grindžiama sistema turi būti sujungta su Organizacijos saugykla – organizacijos žinių baze organizacijos IS inžinerinei veiklai. IT ir IS požiūriu turi būti apibrėžtas bendra struktūra organizacijos modeliavimui ir žiniomis grindžiamai IS inžinerijai. Organizacijos žinių bazei aprašyti skirta formali struktūra vadinama organizacijos meta-modeliu. Organizacijos veiklos meta-modelis yra naudojamas kaip architektūra kontroliuojanti organizacijos veiklos modelio projektavimą konkrečiai verslo sričiai. Organizacijos veiklos modelis (angl. Enterprise model) formalizuoja organizacijos struktūrą ir elgesį, tam kad būtų geriau suprasta verslo įmonė, apibrėžti reikalavimai ir pagerintas Informacinės Sistemos kūrimo ir įdiegimo procesas. Žiniomis paremtos CASE sistemose veiklos meta-modelis, panaudojamas viso IS vystymo proceso automatizavimui.

Didžiausia žinių modeliavimo problema, tai kad nėra standartinio metodo, žinių modeliavimui žiniomis grindžiamos sistemos vystymui. Daugelis metodų, kurie naudojami žinių inžinerijoje yra perimti iš programinės irangos inžinerijos. Žinių modeliavimui naudojami metodai yra projektiai

pagrįsti įvairiomis notacijomis, tokiomis kaip UML, IDEF, SADT, OMT, Multi – perspektyvos modeliavimu ir t.t.

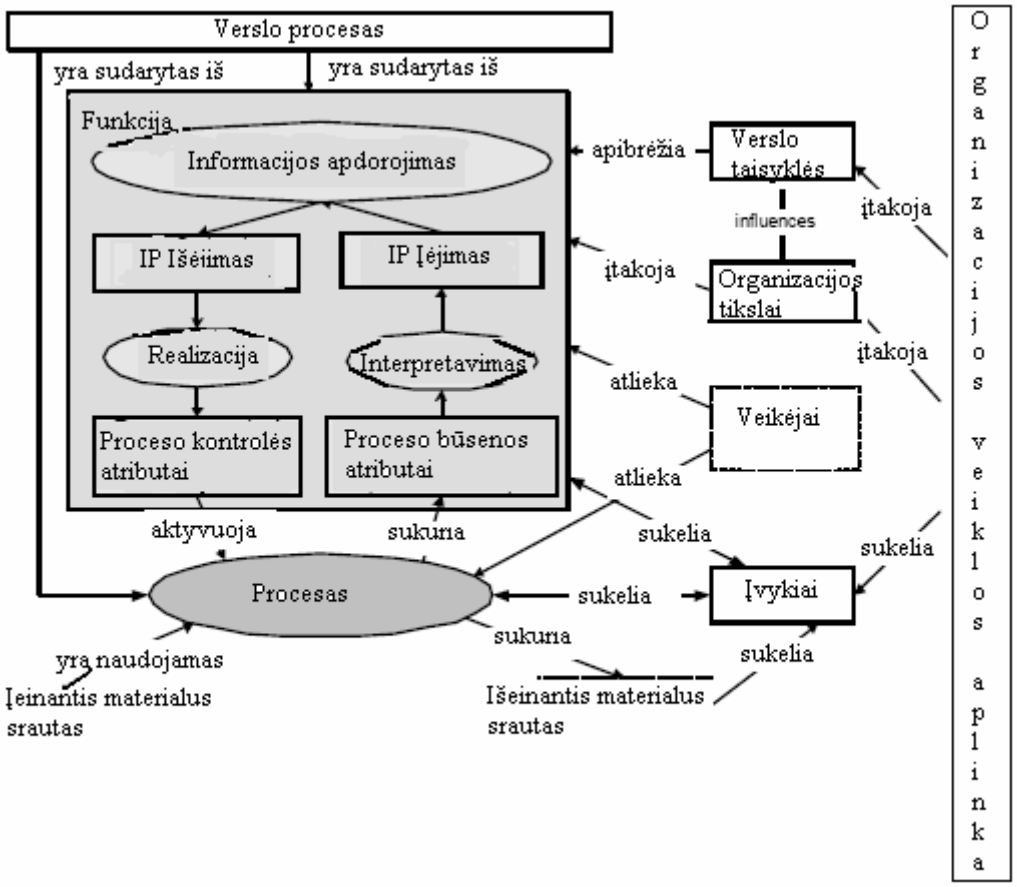
Prie normalizuoto organizacijos veiklos modelio pateikimo dirba IFAC-IFIP Task Force, kurios tikslas bendros organizacijos modeliavimo kalbos UEML (Unifief Enterprise Modeling Language) vystymas. OMG (Object Management Group) ir Task Force – atsakingos už IS inžinerijos proceso standartizavimą – MDA struktūra ištobulinta ir 2005 priimta kaip metodas į modelius orientuotai inžinerijai įgyvendinti. Model-Driven Architecture yra naujas požiūris į organizacijos architektūros konstravimą, abstraktizuojant ir vizualizuojant verslo reikalavimus nuo platformos ir nuo įgyvendinimo technologijos nepriklausomais modeliais, atskiriant įgyvendinimo detales nuo verslo funkcijų, ir suteikiant galimybę vykdyti greitą organizacijos integravimą (angl. Rapid Enterprise Integration).

MDA ir UEML tikslas organizacijos veiklos modeliavimo ir Informacinių Sistemų inžinerijos proceso integracija. UEML koncentruojamasi į organizacijos modeliavimo būdų apjungimą ir svarbi IS inžinerijos metodų ir įrankių sustiprinimui. Siekiama įkomponuoti MDA (modeliais paremtą architektūrą) technologijas į UML, kuris pajėgus modeliuoti organizacijos veiksmus. Kaip skelbia OMG (angl. Object Management Group)UML (angl. Unified Modeling Language) kartu su OCL (angl. Object Constraint Language) yra pagrindinis standartas objektų modeliavimui programinės įrangos inžinerijoje,. UML pagrinde yra modeliavimo kalba kuri apima platų skirtingų sričių spektrą. Kaip ir kiti objektiškai orientuoti metodai UML yra papuliari dėl savo ekspresyvumo, lankstumo ir naudojimo paprastumo. Vienas iš svarbių UML bruožų, šios kalbos praplečiamumas, ši savybė UML padaro viena iš fovoritinių metodų žinių modeliavime, tiek KBS vystymo metodiniu, tiek standartizacijos aspektu. UML plėtiniai gali būti formaliai paskelbti panaudojant UML profilį žinių modeliavimui. Tačiau UML vis dėlto nepatenkina poreikių ir reikalavimų verslo srities žinių modeliavimui IS inžinerijos srityje [3].

MDE (angl. Model Driven Engineering) – modeliais paremtos inžinerijos tikslas pakeisti dabartinę situaciją, kuomet modeliai (t.y., UML diagramos) yra naudojami kaip grafinis atvaizdavimas to, ką programuotojas turi pavesrti programine įranga, į situaciją, kuomet modeliai yra naudojami kaip pagrindinis vystymo įrankis. Modeliais paremtos inžinerijos modeliai yra automatiškai transformuojami į kitus modelius ir programinį kodą. MDE sistemas siūlo atvaizduoti modeliais ir ryšiais tarp jų. Modeliai yra valdomi automatiškai panaudojant žymėjimo (angl.mapping) technologijas. Siekiant tinkamai tvarkyti modelius ir automatiškai jais manipuliuoti MDE pasiūlė metamodelio notaciją.

Pateikiant normalizuotą organizacijos meta-modelio aprašymą pasiekiamas aukštesnis integravimo laipsnis tarp IS inžinerijos sistemų ir organizacijos veiklos modeliavimo. Meta-modelis apibrėžiamas kaip bendra loginė struktūra skirta grafinių modelių klasifikavimui ir sisteminimui,

organizacijos bei organizacijos sistemų plėtotei įskaitant informacines sistemas. Meta-modelio paskirtis – integruoto branduolio sudarymas, skirtingų organizacijos veiklos modeliavimo būdų ir metodologijų struktūros apibendrinimas [3].



1 pav. Principinė organizacijos veiklos meta-modelio schema.

Pirmame paveiksle pavaizduotoje schemoje informacijos apdorojimo blokas transformuoja sistemizuotus realius duomenis apie kontroliuojamus procesus į valdymo sprendimus. Realizacijos ir interpretacijos blokai talpina informacijos grįžtamųjų ryšių ciklų rinkinius. Verslo taisyklių blokes iliustruoja sąlygų, suvaržymų ir skaičiavimų rinkinį. Veikėjai, tai aktyvūs resursai: žmonės arba mašinos su kontroliniu įrenginiu [3].

Įvertinus visus veiksnius integruotą Informacinių sistemų inžinerija gali būti pasiekta tik gerai organizuojant susijusiu modelių aibės valdymą. Informacija apie sudėtinga organizacijos sistemą yra surenkama iš įvairių šaltinių (sistemos inžinierių, vadybinių, programinės įrangos inžinierių, kokybės inžinierių ir pan.) kurią kiekvienas informaciją gali pateikti panaudojant specifine semantika. Bet kadangi pagal šia informacija sukurti modeliai reikalingi bendrai sistemai kurti, jie turi būti apjungti. Tokiu pagrindu atsiranda poreikis įrankiui galinčiam atlikti formalų tokios skirtingos informacijos apjungimą ir išskyrimą. Meta-modeliavimo technologija gali efektyviai įgyvendinti šį poreikį.

Metamodeliavimas pagrinde yra analizė, projektavimas ir įgyvendinimas struktūrų, taisyklių, apribojimų, modelių ir teorijų pritaikomų ir naudingų atitinkamų problemų modeliavimui, tai konkrečios srities sąvokų rinkinio kūrimas. *Metamodelis – tai tikslus konstrukcijų ir taisyklių, reikalingų semantiniams modeliams sukurti apibrėžimas.* Modelis yra realaus pasaulio abstrakcija, metamodelis yra kita abstrakcija išryškinanti paties modelio savybes, tai abstrakti kalba skirtingų metaduomenų apibrėžimui. Metamodelis yra glaudžiai susijęs su ontologijomis, abi šios sąvokos dažnai naudojamos aprašyti ir analizuoti ryšius tarp sąvokų, taip pat reiktų pažymėti, kad metamodelis yra ontologija, bet ne visos ontologijos yra tiksliai sumodeliuotos kaip metamodeliai. Metamodelis taip pat gali būti laikomas detaliu aprašymu (sąvokomis ir taisyklėmis) kaip yra sukonstruotas atitinkamas sričiai būdingas modelis. Metamodelis taip pat apima formalizuota atitinkamos srities modelio specifikacijos notaciją. Daugeliu atveju metamodelis panaudojamas kaip schema semantiniams duomenims, kuriais reikia apsikeisti ir/arba saugoti saugykloje, ir aprašyti kalbą kuri palaiko reikalinga metodologiją arba procesus. Tuomet modelis įgyja betkokios metaduomenų kolekcijos prasmę, o organizacijos integracijos metamodelis tampa metaduomenų apibrėžimo struktūra, iliustruojantčia integracijos procesą objektų saugyklos ribose, objektų modeliavimo įrankius ir metaduomenų valdymą išskirtose verslo srityse. Meta-modelis taip pat apibrėžia kalbą, aprašančia probleminę sritį. Pavyzdžiui UML (Unified Modeling Language) aprašo objiektiškai- orientuotos programinės įrangos sitemos artefaktus. Kiti meta-modeliai gali aprašyti tokias sritis kaip procesas, organizacija, paslaugų kokybė ir pan. Nors meta-modeliai aprašomi kaip atskiros struktūros, tarp jų egzistuoja ryšiai. Kiekvienas rišys metamodelyje atspindimas asociacija su įvairiaus kardinalumais, kiekvienas objektas, modeliavimo kalbos sąvoka – metamodelio klase.

Pateikus tinkamą metamodelį, tampa įmanoma apibrėžti automatinį procesą, kuris pavyzdžiui gali būsenų diagrama transformuoti į klasių digramą. Pirmiausia tarp transformuojamo modelio metamodelio ir UML metamodelio turi būti nustatyti sąryšiai. Toks sąryšių sudarymas gali būti aprašomas kaip modeliostruktūrinis apjungimas (mapping), aprašanti ryšius tarp dviejų arba daugiau modelių, apibrėžiant ryšius tarp elementų metamodeliuose. Skirtinguose modeliuose kelatas arba visa grupė elementų gali atspindėti tą pati dalyką skirtinguose abstrakcijos lygiuose arba iš skirtingų požiūrio taškų.

1.2. Žinių modelis Informacinių Sistemų Inžinerijoje

Tipinis šiandieninės kompiuterizuotos IS inžinerijos bruožas – jos empirinė prigimtis, kadangi CASE sistemos modelių saugykla yra sudaryta remiantis tik pagrindiniais organizacijos veiklo bruožais, o surinkti organizacijos duomenys nėra patikrinti pagal formalizuotus kriterijus. Probleminės srities duomenų surinkimo procesas pagrinde atliekamas analitiko ir busimųjų sistemos vartotojo, todėl dažni apie probleminę sritį surinkta informacija yra nepakankamai tiksli.

Probleminės srities duomenų rinkimo procese, be pagrindinį vaidmenį atliekančio žmogaus yra naudojama keletas formalizuotų informacijos rinkimo metodų. Kitas šių dienų CASE metodo trūkumas tai, kad projektavimo stadijos modeliai sudaromi interaktyviai (dalyvaujant ne tik CASE įrankiui bet ir projektuotojui), ir tik keletas modelių IS projektavimo stadijoje dalinai sugeneruojami, ką įtakoja žinių apie organizaciją trūkumas. Pirmoje IS projektavimo ciklo stadijoje, CASE sistema sugeneruoja funkcinės hierarchijos diagramą, remdamasi probleminės srities informacija, tuo tarpu paskutinėje IS projektavimo ciklo stadijoje, remiantis klasių modeliu ir duomenų bazės specifikacija yra sugeneruojamas programinis kodas. Kiti projekto modeliai yra suformuojami interaktyviai, t.y., projektuotojas, analitikas ir programuotojas sukuria IS projekto modelius analizuodami ankstesniame etape sukurtus modelius. Taigi spragos IS kūrimo procese yra įtakojamos žmogiškojo faktoriaus.

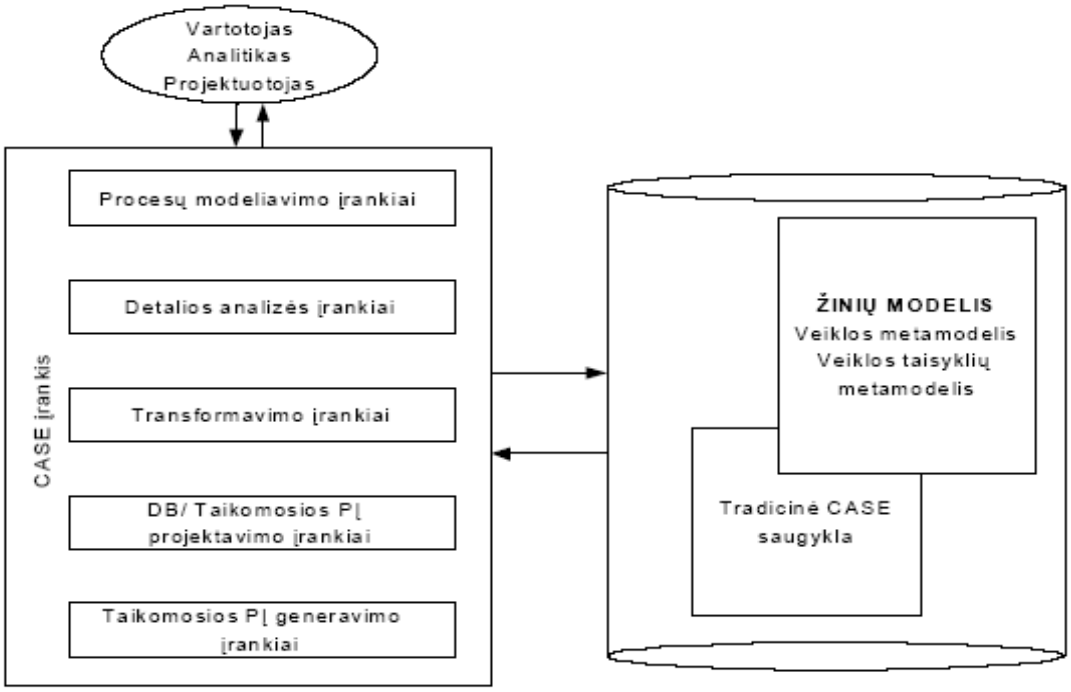
Šios spragos iliustruoja tai, kad IS projektas yra sukurtas interaktyviu būdu (procese dalyvaujant žmogui), o ne pasinaudojant algoritmu. Tai nulemia IS projekto modelių nesuderinamumą ir IS projektavimo proceso nenuosekluma, kadangi IS projektavimo procese dalyvauja per daug žmonių. Taikant formalizuotis (algoritmu pagrįstus) informacijos analizės, kontrolės ir generavimo metodus galima išvengti daugelio klaidų. Yra daug organizacijos modeliavimo metodų : CIMOSA, GERAM, IDEF suite, GRAI, DoD , MDA), standards (ISO 14258, ISO 15704, PSL, ISO TR 10314, CEN EN 12204, CEN 40003 [2].

Analizuojant žiniomis pagrystos IS inžinerijos spragas galima ižvelgti galimus pasikeitimus CASE įrankiu architekturoje. Žiniomis grindziamos IS inžinerijos principai buvo sudaryti remiantis šių spragų analize. Žiniomis grindžiamos IS inžinerijos principai pateikia Organizacijos modeli, organizacijos meta-modeli, ir formalų organizacijos modelį (formaliai aprašytas Organizacijos schema), kaip neatsiejamus bet kurios žiniomis paremtos CASE metodo ir CASE įrankio dalis.

Organizacijos veiklos modelis (Enterprise model) yra nepriklausomas struktūros, veiklos, procesų, informacijos, resursų, žmonių, elgesio, tikslų ir verslo, vyriausybinių ar pačios organizacijos apribojimų atspindėjimas. NAYLOR apibrėžia modelį kaip *„ ... bandymą aprašyti vidinius ryšius tarp organizacijos finansinių, vadybos ir gamybos veiklų pasinaudojant matematiniais ir loginiais ryšiais atvaizduotais kompiuteryje*.“ Šie vidiniai ryšiai turėtų atspindėti firma visais aspektais *„... fizines kompanijos operacijas, finansines veiklas, bei reakcija į investicijas*“. Organizacijos modeliavimas, tai organizacijos veiklos gerinimas sukuriant organizacijos modelį (enterprise model). Tai apima tiek verslo tiek IT procesų modeliavimą. Yra keletas metodų organizacijos modeliavimui, tokių kaip aktyvus žinių modeliavimas, procesų modeliavimas (CIMOSA, PERA, LOVEM ir DYA ir t.t), objiektiškai orientuotas modeliavimas ir/arba organizacijos modeliavimas pasinaudojant „multi-agent“ sistemomis. Probleminės srities žinios (kurios yra patikrintos remiantis formaliais kriterijais) turėtų būti saugomos organizacijos

žinių saugykloje, kuri CASE įrankio dalis ir turėtų būti naudojamos kontroliuoti vartotjo bei analitiko žinias, taip pat IS projekto sprendimo vertinimui. Ši CASE įrankio saugykla yra naudojama taip pat ir IS projektinių modelių generavimui. Organizacijos veiklos modelio struktūra yra kontroliuojama rementis formaliais metodais grindžiama specifikacija, vadinama organizacijos meta-modeliu. IS vystymo problema pasireiškia tuomet, kai empiriškai surinkta informacija (reikalavimai) turi būti patikrinta ir įvertinta.

Organizacijos meta modelis pateikiamas paveiksle. Organizacijos meta-modelis turi būti pagrindinė formali struktūra sukurta informacijos apie probleminę sritį surinkimui, rezultate pateikianti konkrečios verslo srities organizacijos veiklos modelį. Organizacijos meta–modeliu paremtas organizacijos modelis turi būti naudojamas kaip šaltinis žinių apie veiklos sritį kiekvienoje informacinių sistemų inžinerijos stadijoje ir IS vystymo žingsnyje [3]. Žiniomis grindžiamos CASE sistemos architektūra pavaizduota 2 pav. aprašo CASE sistemos sustiprintos žinių baze architektūrą.



2 pav. CASE sistemos su žinių baze architektūra.

CASE sistemos žinių bazė yra sudaryta iš dviejų dalių: organizacijos veiklos meta-modelio ir organizacijos veiklos modelio. Organizacijos modelis (Enterprise model) ir organizacijos meta-modelis (Enterprise meta-model) yra konceptualizuojami ir formaliai kaip specifikacija aprašomi pastovūs organizacijos veiklo duomenys. Organizacijos meta-modelis (EMM) šioje sustiprintoje informacinės sistemos vystymo aplinkoje yra iš anksto apibrėžtas informacijos šaltinis ir yra naudojamas kontroliuoti srities žinių rinkimo ir analizės procesą. Žiniomis grindžiamos IS

vystymas remiasi prielaida, kad visos IS vystymo gyvenimo ciklo stadijos yra palaikomos CASE sistemos žinių duomenų bazės.

CASE sistemos žinių bazė turi būti trečias aktyvus organizacijos veiklas žinių šaltinis Informacinių sistemų inžinerijoje greta vartotojo ir analitiko. Vartotojas laikomas tarpine grandimi tarp realaus pasaulio (organizacijos) ir IS sistemos projektuotojo, tačiau vartotojo žinios apie organizaciją yra ribotos (vartotojas gerai žino tik tą sritį, kurioje dirba), ko pasekoje vartotojo reikalavimai sistemai yra nepastovūs. IS plėtojimo problema iškyla tuomet, kai empyriškai surinkta informacija (vartotojo reikalavimai) turi būti patkrinta ir įvertinta.

CASE sistemos žinių bazė (apjungta su atitinkamu algoritmu) užtikrina neprieštaringuma tarp verslo analizės ir IS modelių, suteikia naujas galimybes IS vystymui gaunamas patikrinimas ir patvirtinimas visuose gyvavimo ciklo etapuose.

2. Veiklos modelio sudarymas žinių saugyklos pagrindu

2. 1. Formalizuotas UML Veiklos (activity) diagramos metamodelio aprašymas

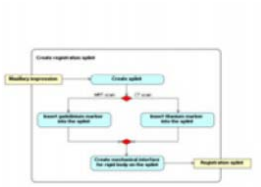
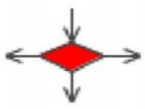
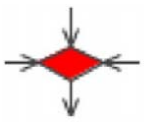
Unified Modeling Language (UML) yra grafinių notacijų rinkinys, palaikomas meta-modelio, padedančio aprašyti ir projektuoti objektiškai orientuotu metodu paremtos programinės įrngos sistemas. Viena iš sistemų analizei UML siūlomų diagramų yra veiklos (activity) diagrama.

UML veiklos (activity) diagrama dažnai naudojama programinės įrangos vystymo ir verslo procesų darbų sekoms (workflow) vaizduoti. Taip aprašomas dinaminis sistemos elgesys.

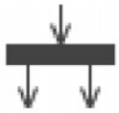
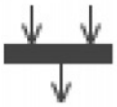
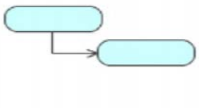
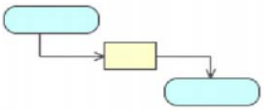
Pagrindinė veiklos (activity) diagramos paskirtis – aprašyti veiksmus bei procesus, į kuriuos įtraukiami vienas ar daugiau objektų. Taip pat apibrėžti objektų, dalyvaujančių atitinkamuose veiksmuose, pasikeitimus vaidmenyse, būdenose bei atributų vertėse. Veiklos diagramos akcentas yra veiksmų seka ir veiksmų vykdymo sąlygos, o ne objektai, kurie yra pagrindiniai būsenų diagramose. Diagrama nusako veiksmų seką, kuri gali turėti sprendimus, šakas (fork), jungtis (join), ciklus. Veiklos (activity) diagramoje yra identifikuojami veiksmai, kuriuos reikia realizuoti, bei veiksmų tarpusavio sąryšiai. Perėjimus iš vieno veiksmo į kitą šioje diagramoje inicijuojuoja įvykiai: ankstesnio įvykio užbaigimas, objekto tam tikroje būsenoje atsiradimas, signalo pasirodymas, tam tikrų sąlygų išpildymas. Seka baigiasi tada, kai visi veiksmai įvykdomi, įvyksta klaida arba veiksmai nutraukiami. Veiklos (activity) modelis gali turėti padalas (partitions), kurias galima priskirti klasifikatoriams - veiklas galima susieti su objektais.

Pagrindinis diagramos elementas pateikti žemiau esančioje 1 lentelėje.

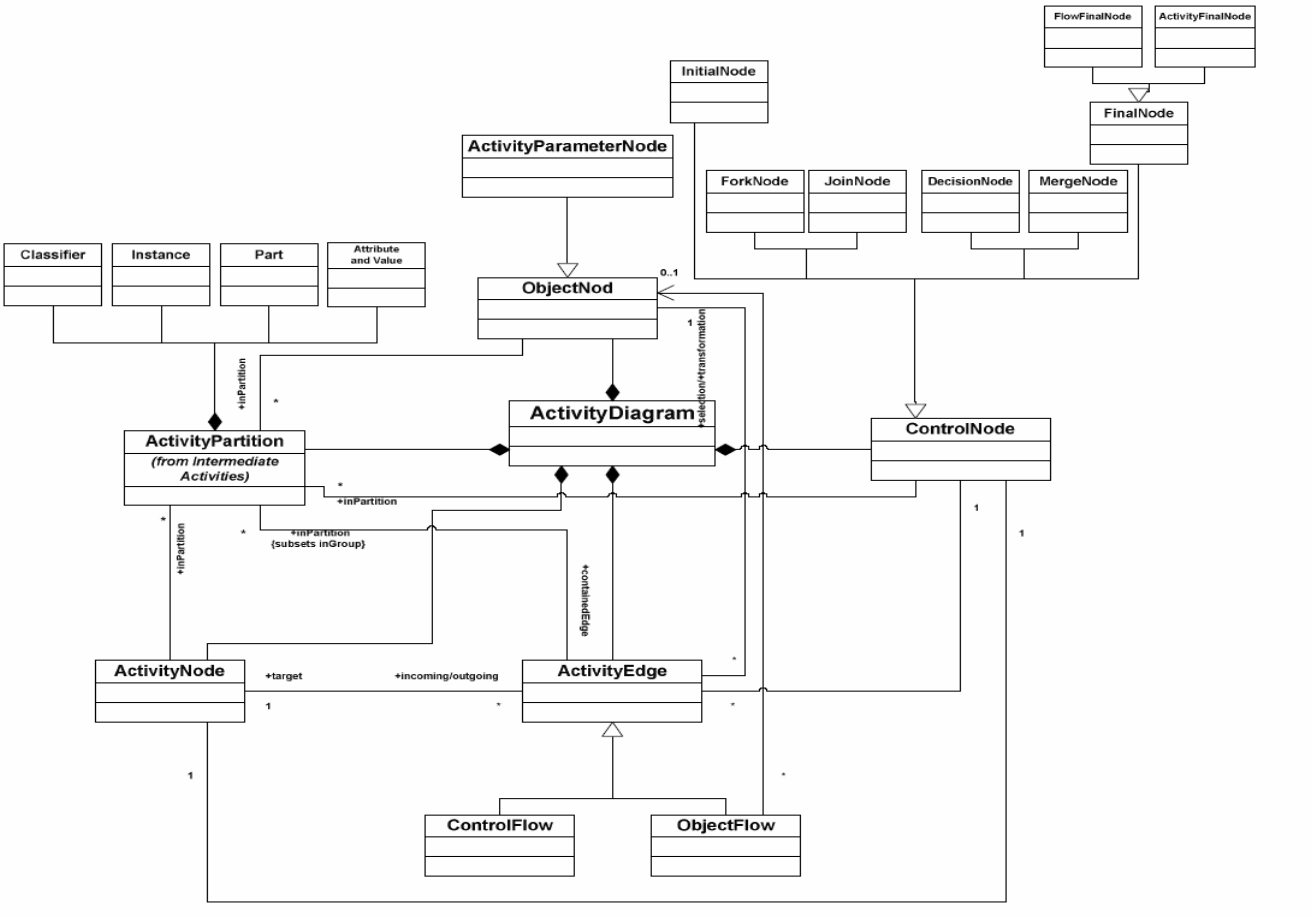
1 lentelė. Pagrindiniai veiklos (activity) diagramos elementai

		Pavadinimas	Grafinis žymėjimas	Aprašymas
		Veikla/Veiklos diagrama (angl. activity/activity diagram)		Veikla yra parametrizuota elgesio sekos specifikacija. Veiklos diagrama parodo darbų seką nuo pradžios iki pabaigos taško, detalizuojant sprendimo kelius, kurie egzistuoja įvykių, sudarančių veiklą progresijoje.
Veiklos (Activity Nodes)		Veiksmas (angl. Action)		Atvaizduoja vieną žingsnį veikloje. Veiksmas neprasideda tol, kol nėra patenkinamos visos sąlygos, reikalingos jam prasidėti. Veiksmo pabaiga inicijuoja kito veiksmo pradžią arba aprašomos veiklos pabaigą.
		Objekto būsena (angl. Object Node)		Objekto būsena apibūdina objektų srautą veikloje ir žymi objektų judejima veiklos (activity) diagramoje.
	Taisyklės (Control Nodes)	Pradinis taškas (angl.Initial node)		Veiklos srauto pradžios taškas, naudojamas srauto pradžiai žymėti, kuomet pradedama veikla, aprašoma UML 2.0 Activity diagramoje.
		Veiklos srauto pabaiga (angl. Activity Final Node)		Visus veiklos srautus užbaigiantis taškas žymi vieno srauto pabaigą UML 2.0 Activity diagramoje.
		Veiklos pabaiga (angl. Flow Final Node)		Veiklos pabaiga nutraukia veiklos srautą. Ji sunaikina visus veiksmus atkeliavusius iki veiklos pabaigos, tačiau neturi jokios įtakos kitiems veiklos srautams.
		Sprendimas (angl. Decision Node)		Sprendimas, remiantis nurodyta sąlyga, pasirenka tarp išeinančių srautų, kas leidžia kontroliuoti srautą, jei reikalingos apsauginės sąlygos.
		Apjungimas (angl. Merge Node)		Apjungimas sujungia keletą alternatyvių srautų.

1 lentelė. Pagrindiniai veiklos (activity) diagramos elementai tęsinys

		Pavadinimas	Grafinis žymėjimas	Aprašymas
Veiklos (Activity Nodes)	Taisyklės (Control Nodes)	Išsišakojimas (angl. Fork Node)		Išsišakojimas parodo konkurencinio srauto pradžią. Išskirsto srautą į kelis lygiagrečius srautus
		Sujungimas (angl.Join Node)		Sujungimas sinchronizuoja kelis įeinančius srautus į vieną išeinantį.
Activity Edges		Informacinis srautas (angl. Activity Edge)		Informacinis srautas yra du veiksmus apjungianti kryptinga jungtis, žyminti informacinius srautus bei apimanti objektų būsenų bei kontrolės mechanizmų pokyčius.
		Kontrolės srautas (angl. Control Flow)		Parodo kontrolinį srautą nuo vieno veiksmo prie kito, pažymint naujo veiksmo (Activity Node) pradžią, kuomet prieš tai buvęs yra baigtas.
		Objektų srautas (angl. Object Flow)		Ojektinis srautas gali tarp dviejų veiksmų apjungti duomenis arba objektą. Jis modeliuoja srauta iš arba į objektą.

Paveikslėlyje pateikiama veiklos (activity) diagramos metamodelio dalis, kuri apima metamodelio elementus atvaizduojančius darbo srautus (work flow), objektų srautus (object flow), veiksmus (action), operacinius įvykius (operational calls) ir kitus metamodelio elementus, išreiškiančius taisykles. Šio modelio sukūrimui įvairūs veiklų ir veismų modeliavimo elemetai, aprašomi UML 2.0 struktūrinėje specifikacijoje buvo abstraktizuoti ir apjungti.



3 pav. Veikos (Activity) diagramos metamodelis.

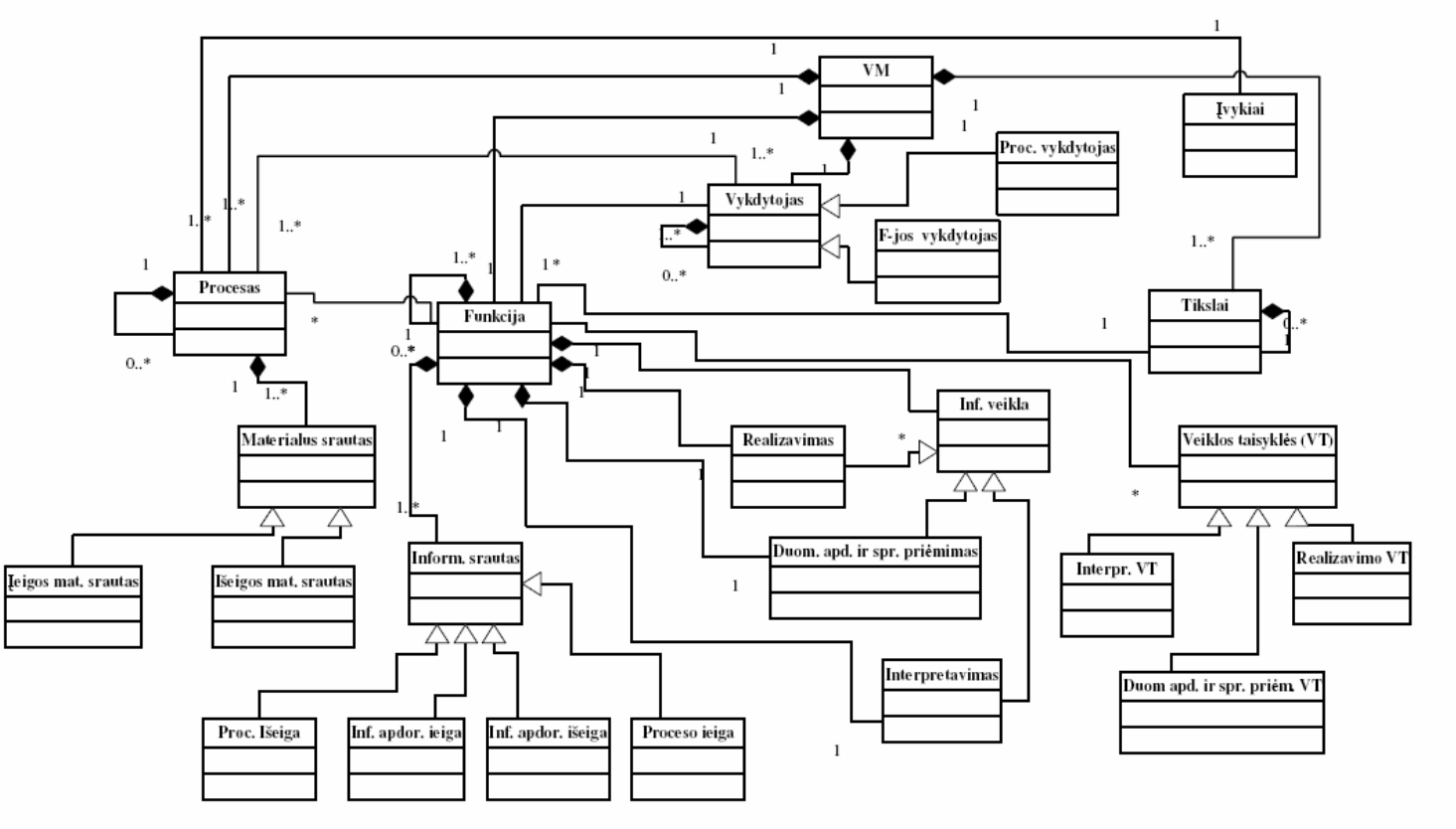
MDA (Model Driven Architecture) kiekvienas modelis yra paremtas specifiniu metamodeliu, apibūdinančiu modeliavimo kalbą. Visi metamodeliai paremti vienu metamodeliu vadinamu MOF (Meta Object Facility), todėl modelių transformacijos yra galimos tarp dviejų metamodelių apibrėžiant transformavimo taisykles.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

2.2. Formalizuotas veiklos modelio aprašymas

Kaip jau buvo minėta ankščiau veiklos metamodelis – tai formaliai apibrėžta veiklos modelio struktūra, kuri sudaryta pagal formalizuotą veiklos modelį, atitinkantį bendruosius valdymo teorijos principus [9]. Tai – žinių apie realų pasaulį šaltinis, skirtas verslo procesų reinžinerijai ir IS inžinerijai.

Šiuo metu daugelyje modeliavimo įrankių IS projektavimo metu arba tik iš dalies naudojama informacija, surenkama analizės metu. Todėl tikslinga šias žinias surinkti į saugyklą, kurios sudėtį apibrėžia Veiklos metamodelis. Šio veiklos metamodelio sudėtyje turėtų egzistuoti pagrindinių veiklos modeliavimo metodologijų bei būdų esminiai elementai. Iš šios saugyklos būtų generuoti UML projektinius modelius. Taip veiklos žinių pagrindu automatizuojant IS kūrimo procesą yra sutaupomas tiek projektuotojų, tiek užsakovų darbo laikas, be to, gaunama išsamesnė ir tikslesnė IS projektinė schema, kadangi veiklos modelyje saugomos valdymo požiūriu patikrintos žinios. Taip

veiklos modelis tampa organizacijos žinių saugykla, kuri yra naudojama ne tik žinioms apie organizacijos veiklą kaupti, bet ir kaip priemonė, minimizuojanti IS reinžinerijos darbų apimtį, pakitus organizacijos veiklai [7].



4 pav. Veiklos metamodelio klasių modelis [11].

Pateiksime formalizuotą veiklos modelio (VM) metamodelio aprašą, kuris reikalingas activity 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:

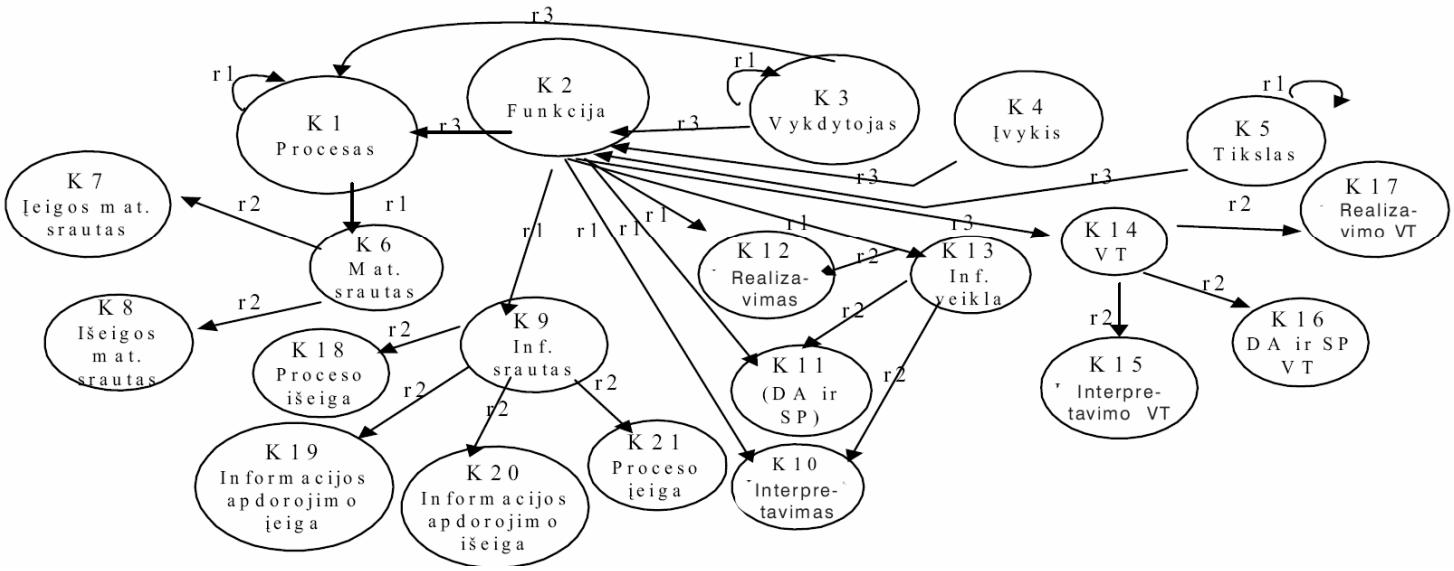
$Kn=<\{an_1, an_2,\ldots,an_k\}, \{mn_1, mn_2,\ldots,mn_l\}>$, kur $\{an_1, an_2,\ldots,an_k\}$ – elemento Kn atributai,

$\{mn_1, mn_2,\ldots,mn_l\}$– 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 5 paveiksle [12].



5 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).

2.3. Veiklos (Activity) diagramos formalaus modelio generavimas

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

Activity modelio formalų aprašymą sudarome remdamiesi, activity diagramos metamodeliu., kuris pateikiamas 7 paveiksle.

Algebrinis activity diagramos M2 aprašomas taip:

• M2=<{K22, K23,...,K42}, {r1, r2, r3}>.
• M2 – Activity modelis,

čia K22 – metaklasė *ActivityPartition,* K23 – metaklasė *ControlNode*

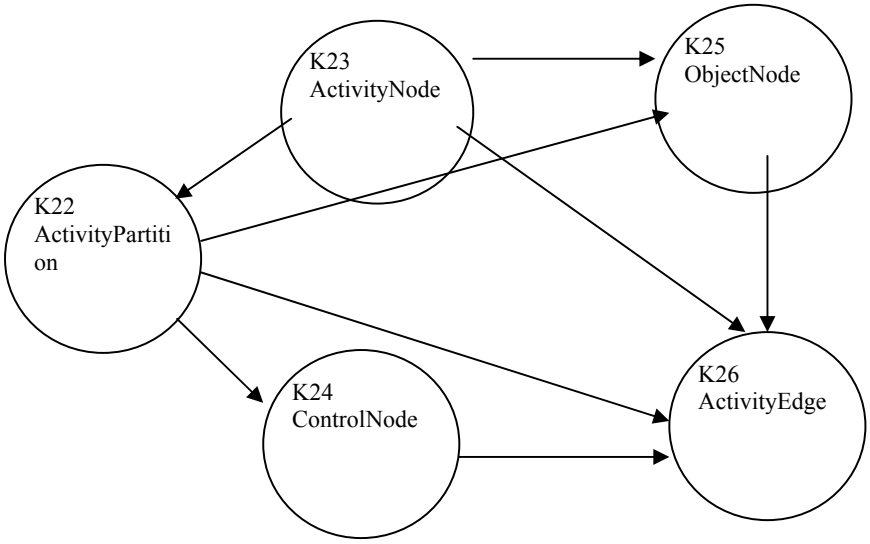
M2=<{K22, K23, ..., K26}, {r3}>

M2 – activity diagramos modelis, K22 – klasė *ActivityPartition* (veiksmų suskirstymas pagal vykdytoją), K23 – klasė *ActivityNode* (žymi veiksmą), K24 – klasė *ControlNode* (taisyklės, srautų kontrolė), K25 – klasė *ObjectNode* (nurodo objektų judėjimą diagramoje – materialius srautus), K26 – klasė *ActivityEdge* (žymi informacinius srautus), r3 -asociacija.

- (K22)r3(K24), (K22)r3(K26), (K22)r3(K25) - klasė ActivityPartition asociacijos ryšiais yra susieta su ControlNode, ActivityEdge ir ObjectNode klasėmis.
- (K23)r3(K22),(K23)r3(K24), (K23)r3(K26) – klasė ActivityNode asociacijos ryšias susieta su ActivityPartition, ControlNode ir ActivityEdge klasėmis.
- (K24)r3(K26) – klasė ControlNode sociacijos ryšiu susieta ActivityEdge klase.

- (K25)r3(K26) - klasė ActivityEdge sociacijos ryšiu susieta ObjectNode klase.

Funkcinė veiklos (activity) diagramos metamodelio schema pateikta 6 paveiksle.



6 pav. Veiklos (Activity) modelio grafinė schema

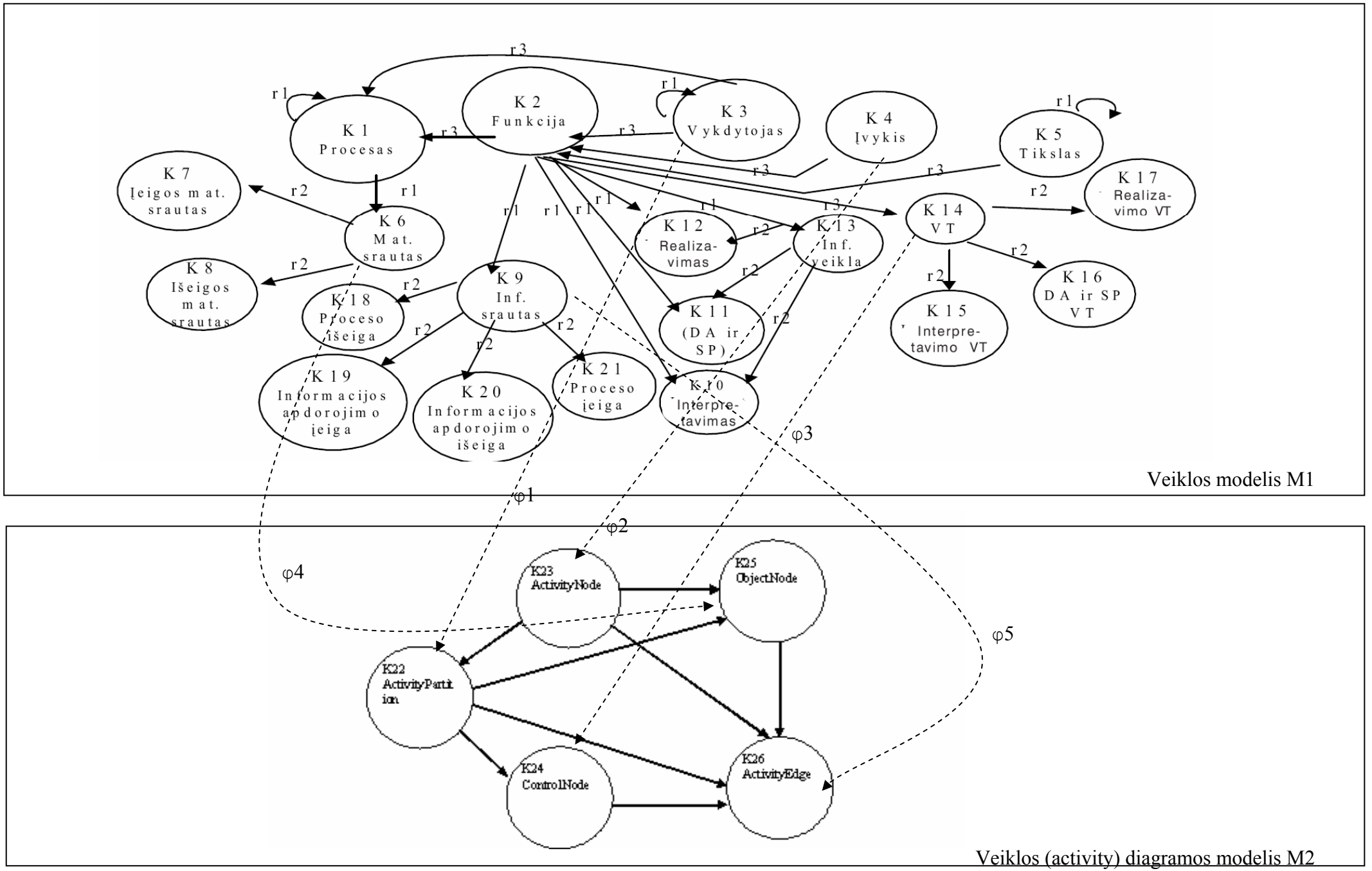
2.4. Veiklos modelio ir veiklos (activity) diagramos modelio sąsaja

Aprašysime veiklos modelio ir ir activity diagramos modelio loginį ryšį nagrinėdami jų klasių modelius. Activity diagramos modelio generavimui būtinos veiklos modelio klasės *Vykdytojas* (K3), *Įvykis* (K4), *Veiklos taisyklės* (K14), *Materialus srautai* (K6) ir *Informaciniai srautai* (K9). Generuojant activity diagramos modelį šios klasės atvaizduojamos į atitinkamas activity diagrmos modelio klases *ActivityPartition* (K22), *ActivityNode* (K23), *ControlNode* (K24), *ObjectNode* (K25) ir *ActivityEdge* (K26). Tai formaliai galima aprašyti aibių atvaizdžiais: φ_1 : K3->K22; φ_2: K4->K23; φ_3: K14->K24; φ_4: K6->K25; φ_5: K9->K26 (2 lentelė). Aibių pavadinimai atitinka klasių pavadinimus.

2 lentelė. Veiklos modelio klasių atvaizdžiai į atitinkamas activity diagramos modelio klases

Veiklos modelio klasių aibės	**Atvaizdis**	**Activity diagramos modelio klasių aibės**	**Formalus aprašymas**
Vykdytojas (K3)	φ_1	ActivityPartition (K22)	K3->K22
Įvykis (K4)	φ_2	ActivityNode (K23)	K4->K23
Veiklos taisyklės (K14)	φ_3	ControlNode (K24)	K14->K24
Materialus srautai (K6)	φ_4	ObjectNode (K25)	K6->K25
Informaciniai srautai (K9)	φ_5	ActivityEdge (K26)	K9->K26

Analizuosime activity diagramos elementų generavimo galimybes, pasinaudodami veiklos modelyje sukauptomis žiniomis. Grafinis lentelės atvaizdavimas pateikiamas 9 paveiksle.



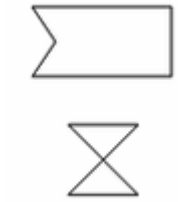
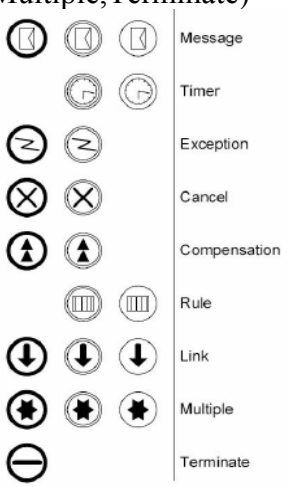
7 pav. Veiklos modelio elementų atvaizdavimas į veiklos (activity) diagramos modelį.

3. Business Process Modeling ir Unified Modeling Language grafinių notacijų palyginimas

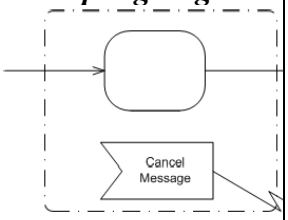
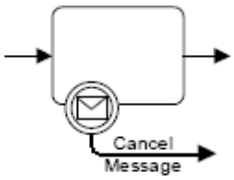
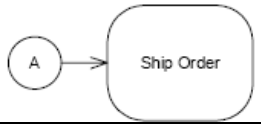
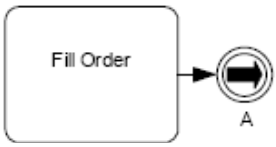
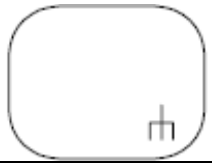
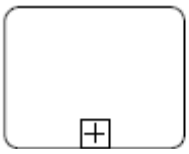
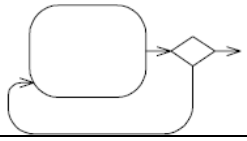
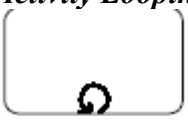
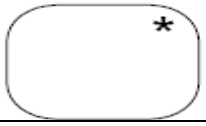
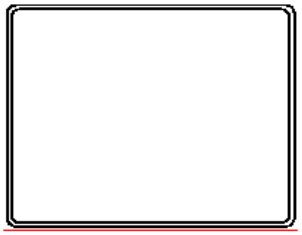
Aukščiau pateiktos formalios veiklos (activity) diagramos ir veiklos modelio sąsajos matome, kad pagrindiniai veiklos modelio elementai gali būti atvaizduoti veiklos (activity) diagramoje. Tačiau be šių pagrindinių elementų veiklos (activity) diagrama turi keletą papildomų ne taip dažnai naudojamų elementų, kurių generavimo galimybes taip pat derėtų aptarti. Šiai analizei atlikti įvertinsime ne tik pateikiama veiklos metamodelį, bet ir veiklos procesų modeliavimo notaciją

(angl. Busness Process Modeling Notation (BPMN)), tai yra darbų sekų (angl. workflow) verslo procesų modeliavimo standartą. Taip pat analizės metu remsimės Stephen A.White darbu, kuriame pateikiamas kai kurių darbų sekų modelio (ang. Workflow) bei veiklos (activity) diagramos palyginimas. Analizės metu darbų sekos (angl. workflow) elementų palyginimą naudosime dėl to, kad būtent šių diagramų pagrindu yra sugeneruojama ir užpildoma veiklos modelio žinių saugykla. O darbų sekos (angl. workflow) dažniausiai modeliuojamos panaudojant grafinio modeliavimo notacijas, tokias kaip UML arba BPM.

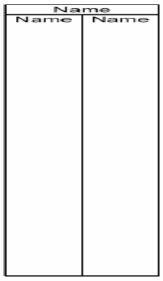
3 lentelė. UML ir BPM grafinių notacijų palyginimas

UML	BPD	Aprašymas
AcceptEventAction 	***Type Dimensijon*** (t.y. Massage, Timer, Error, Cancel,Compensation, Rule, Link, Multiple,Terminate) 	Tai išimtis įprastinės veiklos (activity) diagramos veiklos taisyklėms. AcceptEventAction, tai veiksmas laukiantis specialias sąlygas tenkinančio įvykio pasirodymo BPD grafinė notacija aprašanti įvykio įvykimo priežastį. BPD pateikia daug atvėju aprašančiu, kaip gali būti iššauktas veiksmas. Timer Triger - atskiras atvejis yra procesas pasiekus atitinkama laiko/datos vertė, UML notacijoje žymimas smelio laikrodį primenenčiu elementu.
SendSignalAction 		SendSignalAction yra veiksmas sukuriantis signalą pagal į jį ateinančią informaciją ir perduodantis šį signalą ir perduoda atitinkamam objektui, kur šis signalas gali aktyvuoti veiklos pradžią.

3 lentelė. UML ir BPM grafinių notacijų palyginimas tęsinys

I***nterupting Region*** ir iš jo išeuantis ***Interupting Edge*** 	***Atached to Activity Boudary*** (gali būti panaudojami skirtingi trigeriai) 	Tai veiklos pertraukimo mechanizmas. Veiklos vykdymo metu gavus atitinkamą pranešimą veikla yra nutraukiama ir vykdomas perduodamas sekančiai veiklai, jei tokia egzistuoja.
Activity Edge Connector 	***Within Normal Flow*** 	
CallBehaviorAction 	***Collapsed Sub-process*** 	Sub-proceso detalės diagramoje nėra tiesiogiai matomos. Ženklas veiksme žymi sub-procesą ir informuoja apie esantį žemesnį lygį.
Standart Loop 	***Activity Looping*** 	Atspindi kilpos susidarymą veikloje
Multiple Instances 	***Multiple Instances*** 	Veikla savo veiksmus vykdanti paeiliui.
Transaction 	***Transaction*** 	Sub-procesas užtikrinantis veiklai užsibaigima arba atšaukimą.
Interruptable Region 	***Group*** 	Veiklų sugrupavimas, kuris neįtakoja vykdymo sekos. Naudojamas dokumentacijos arba analizės tikslais.

3 lentelė. UML ir BPM grafinių notacijų palyginimas tęsinys

Expansion Region 	***Loop Type*** 	Activity diagrama panaudodama expension region elementą apjuosiantį veiką sukuria veiklos kopijas. Tuo tarpu BPD šiai funkciajai atlikti pasinaudoja Loop Type atributą.
Padalijimas (angl. Activity Partition) 	***Pool ir Line*** 	ActivityPartition naudojamas siekiant suskirstyti veiksmus pagal tam tikrus juos vienijančius požymius. Gali būti naudojamas paskirstyti veiksmus pagal vykdytoją. BPMN naudojamas pool, kai diagrama apima dvi atskiras veiklas, esybes arba dalyvius, kurie diagramoje turi būti fiziškai atskirti. Line – tai sub-padalinimas, dalinantis pool elementą į dvi dalis.
DataStoreNode 	***Data Store*** 	Duomenys yra pastovųs ir naudojami atsiradus poreikiui, vietoje to kad sandėliuoti laikinus duomenis ir naudotis jai, kai jie yra pasiekiami. Šis objektas kol kas yra BPMN praplėtimu.

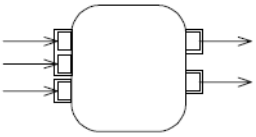
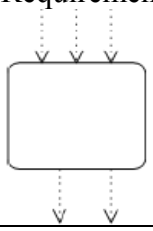
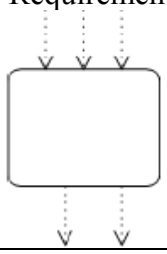
Kaip matome iš BPM ir UML notacijų palyginimo abi modeliavimo specifikacijos yra labai artimos savo pateikiamų sprendimų atžvilgiu. Abi notacijos dalinasi tų pačių žymėjimų atitinkamai sąvokai vaizduoti. Panašumai diagramose matomi dėl bendro jų tikslo – procedūrinio verslo proceso atvaidavimo. Tačiau diagramų grafinėse notacijose galima ižvelgti ir skirtumų, dėl skirtingo galutinio naudojimo: BPMN skirta verslo dalyviams, tuo tarpu UML aprašomas veiklos (activity) diagramos standartas skirtas programinės įrangos procesams modeliuoti.

3.1. Nesugeneruojamų veiklos (activity) diagramos elementų išskyrimas

3.1.1. Dėl notacijos nepilnumo

Notacijų palyginimas parodė, kad naujoji UML 2.0 specifikacija papildyta elementais, kurių atitikmenų BPM notacijoje nėra. Tai pagrinde su objektų srautais susiję elementai.

4 lentelė. UML Parameter Set grafinė notacija

Parameter Set 	Complex Input/Output Requirement 	Kol kas BPM notacija nepalaiko įėjimo ir išėjimo tipų. Pateikiam tik laisva šių UML elementų interpretacija, panaudojant BPMN Task elementą, bei associacion elementą, skirto perduoti informaciją srautą žymintiems objektams iš srauto nežyminčių.
Pins 	Streaming Input/Output Requirement 	

Norint aprašyti šiuos elementus veiklos modelio duomenų saugykloje, sukuriame papildomas klases.

Pin elementas priklauso Object Node veiksmo įėjimo ir išėjimo procesus.

Klasė „PIN“	
Pin_ID	Integer
input	boolen
Atributas	Integer
InfSrID	Integer
Įvykio_id	Integer
MatSrID	Integer
DuomPerd	boolen

Įgyjama reikšmė true arba folse, taip apibūdiant Pin tipą.
Kuriam objektui priklauso
Kuris informacinis srautas ateina
Kuriam įvykiui priklauso.
Kuris materialus srautas ateina.
False nusako, kad pin perduoda duomenis į veiksmą, true ktroliuoja kada jis vykdomas.

Parameter Set

Tai elementas pateikiantis alternatyvių įėjimo ir įšėjimo verčių rinkinį, kuris gali būti naudojamas veiklos elgesiui atvaizduoti. Tai naujas UML 2.0 paskelbtas elementas. Keli objektiniai srautai ieinantys ir paliekantys parameter set elementą tipiškai yra apdorojami panaudojant IR sąlygą, nors kai kuriais atvejais panaudojamas ir ARBA variantas.

Parametro elementas apibrėžia parametrą Parameter Set notacijoje. Šiam elementui aprašyti sukuriame klasę:

Klasė „Parameter Set“	
ParameterSetID	Integer
Parametro ID	Integer
IRsalyga	boolen
FunkcID	Integer

Nusako Parameter Set sudarančius parametrus
Nusakoma koks parametru apjungimas bus naudojamas.
Veikla, kuriai priklauso Parameter Set

Klasė „Parameter“	
Efektas	Text
Parametro ID	Integer
ParameterSetID	Integer

Parametro sukuriama reikšmė

Generacijoje privalo būti įvestas loginis tikrinimas, kad visi ParameterSet parametro ID negali būti lygus kito ParameterSet parametro ID reikšmėms.

Dar vienas BPMN neaprašomų elementų yra ***Exception Handler***.

5 lentelė. UML Exception Handler grafinė notacija

	Šis elementas BPMN neturi savo atitikmens.	Exception Handler nurodo, kokį veiksmą vykdyti toliaus, jei vykdant apsaugotą veiksmą pasirodo išimties sąlyga.

Šis elementas pakeitė UML 1.4 naudotus pertraukimo (break) ir tęsinio (continue) veiksmus.

Klasė „Exception Handler“	
ExHID	Integer
ExSal	Text
Įvyk_ID	Integer
EĮvyk_ID	Integer
Input	boolen
PinID	Integer

Nustatyta sąlygą, kurią aptikęs exception handler pereitų ir jam priklausančio veiksmo vykdymą.
Exception Handler saugomas veiksmas.
Nurodomas veiksmas, kuris bus vykdomas išimties sąlygos atveju.
Nustatomas objektas esantis handler‘io viduje.

Klasė „EIvykis“	
EĮvyk_ID	Integer
EĮvyk_Pav	Text
EĮvyk_sal	Text

Turi būti įvestas loginis tikrinimas, kad rezultato output pin skaičius būtų lygus exception handler įvykyto output pin skaičiui. Exception Handler rezultatai tampa saugomo veiksmo įvykdymo rezulatais.

3.1.2. Dėl duomenų saugyklos nepilnumo

Derėtų paminėti, kad kai kurie UML veiklos diagramos elementai nors ir turi atitikmenį BPM notacijoje, tačiau iš pateiktos darbų sekų (angl. Workflow) pagrindu sugeneruotos eksperimentinės duomenų bazės negali būti sugeneruoti, dėl tam reikalingų duomenų trūkumo. Remiantis pateiktos duomenų bazės struktūra be papildomų klasių įvedimo negali būti sugeneruoti tokie elementai.

Expansion Region bei ***Presentation***

Šio elemento įėjimai sudaro verčių rinkinį, kurių kiekvienai Expansion Region elementas yra įvykdomas po vieną kartą. Naudojant šį elementą įeinančių verčių skaičius nepriklauso nuo išeinančių verčių skaičiaus, tačiau išeinantys elementai turi būti to paties tipo. Regionas veikia kaip filtras atrenkantis jį sudarančių veiklų rezultatus pagal įėjimo reikšmes. Regionui aprašyti pasinaudosime tokia klase:

Klasė „Expansion Region“		
FuncijosID	Integer	Nusako, kokia veikla aprašoma.
InelSk	Integer	Įeinančių elementų skaičius
OutelSK	Integer	Išeinančių elementų skaičius.
InelTipas	Text	Įeinančių elementų tipas
PinID	Integer	
ExRegID	Integer	Regiono vardas sutampa su jį sudarančio elemento vardu.
ExRegPav	Text	Pateikiamas veiksmų vykdymo rėžimas:
Mode	Text	parallel – visi veiksmai nepriklausomi; interative – veiksmai vykdomi pagal elementų seką; strem – verčių aibė vykdoma vienu metu.

Presentation **–** tai Expansion Region elemento veikiančio parallel rėžimu atvejis.

InelSk = OutelSk
Mode = Parallel

ActivityPartition

Actitivy Partition galima pavadinti veiklių grupes tarpusavyje turinčias kažką bendro . Partition padalina veiksmus ir srautus į atskiras sritis, kurios dažnai atspindi organizacinius arba verslo pasidalinimus.

Klasė „Activity partition“	
PadalinimoID	Integer
PadakinimoPov	Text
Dimension	boolen
SubpatID	Integer
DimensionPav	Text
IvykioID	Integer
ObjektoID	Integer
InfSrID	Integer
MatSrID	Integer

Suskirstymo pavadinimas
Ar naudojamas paprastas, ar sudėtinis padalinimas (nustatyta reikšmė false)
Antrinio padalinimo ID

Butina tenkinti sąlygą, kad nei vienas elementas priklausantis vienam padalinimui nepriklausytų kitam.

InteruptingEdge

Tai elementas turintis šaltinį regiono ribose (įvykis) bei imtuvą už regiono ribų. Šiam elementui aprašyti sudarome tokią sąskaitą. Šiam elementui aprašyti sudarome tokią klasę:

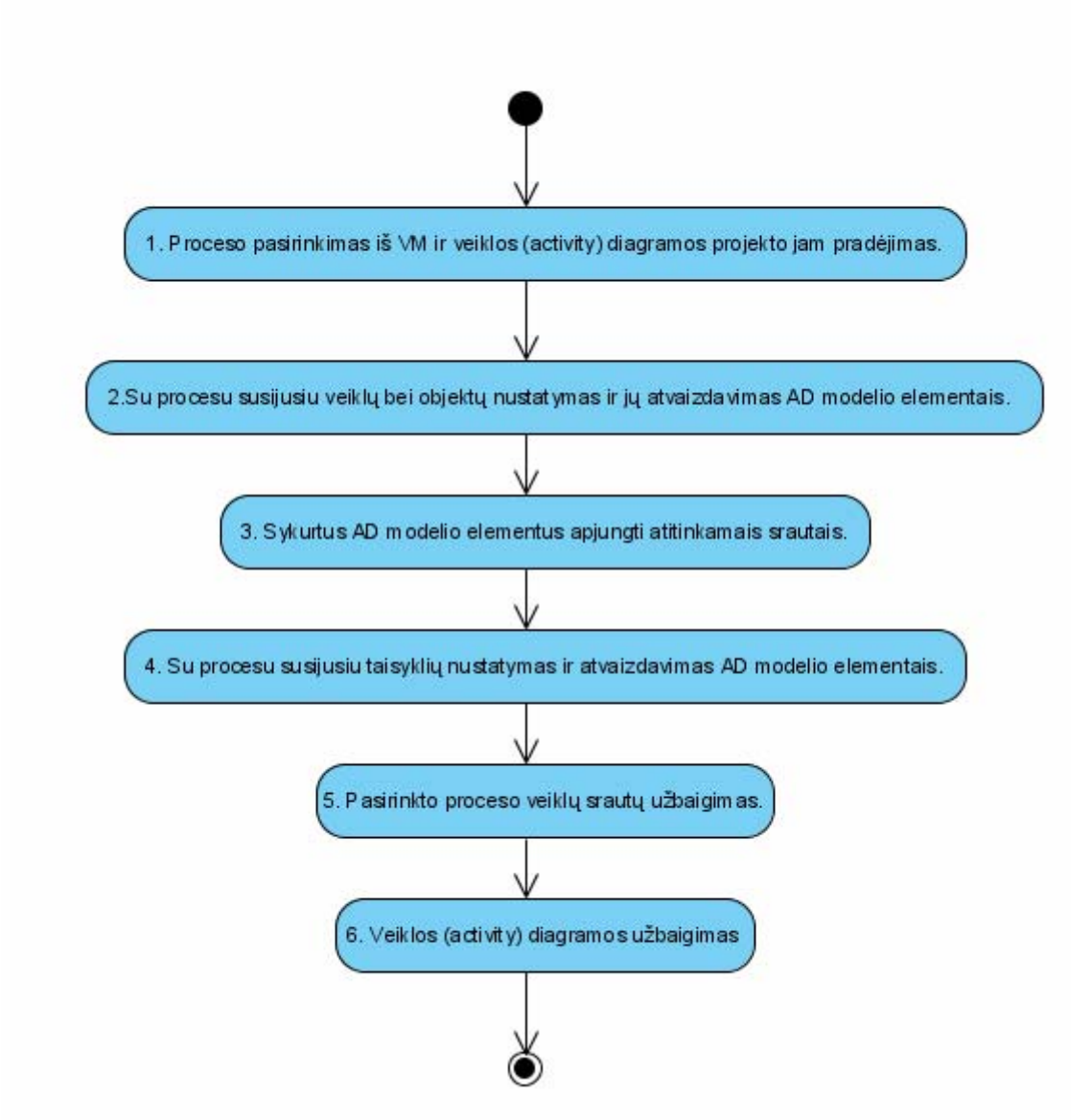
Klasė „InteruptingEdge“	
IvykioID	Integer
RegionoID	Integer
Regiono_Pav	Text
Ivykio_tip	Text

Įvykis sukeliantis InteruptingEdge output = true
Įvykis priimantis InteruptingEdge output = falseI

Šis elementas pagreitina veiklos perėjimą į užbaigimo būseną.

4. Informacinės Sistemos veiklos (activity) modelio generavimo algoritmas

Pateikiamas bedro pavidalo algoritmas UML veiklos (activity) diagramos generavimui.



8 pav. IS veiklos (activity) modelio generavimo algoritmas

1 žingsnis. Proceso pasirinkimas iš VM ir veiklos (activity) diagramos projekto jam pradėjimas. Veiklos (activity) diagramos generavimas pradedamas sistemos vartotojui nurodant vieno iš veiklos modelio saugykloje saugomu proceso vardą (proc_pav). Atlikus proceso pasirinkimą inicijuojamas naujo veiklos (activity) modelio (pažymėkime jį AD1) kūrimas. Naujojo AD projekto pavadinimui priskirimas proceso vardas, t.y., proc_pav->ActivityModelPav, ir inicijuojama AD1 modelio pradžia, sukuriant veiklos (activity) diagramos pradžios tašką (InitialNode).

2 žingsnis. Su procesu susijusių veiklų bei objektų nustatymas ir jų atvaizdavimas AD1 modelio elementais. Pagal vartotojo pasirinktą procesą filtruojant VM saugyklos duomenis yra išrenkami pasirinktam procesui priklausantys priklausantys įvykiai bei materialaus srauto objektai (Įvykis,

MatSrautas). Išrinktieji VM elementai veiklos (activity) modelyje AD1 atvaizduojami: įvykiai -> action, materialaus srauto objektai -> object node elementus. Kiekvienas naujai sukurtas AD1 elementas paveldi pavadinimą iš VM elemento, t.y. action (įvykio_pav), objekt node (pavad).

3 žingsnis. Sukurtus AD1 modelio elementus apjungti atitinkamais srautais. AD1 modelio elementai apjungiami srautus iliustruojančiais simboliais Activity Edge, kurie nustatomi pagal VM saugyklos klasės InfSrautas bei ProcMSr duomenimis.

4 žingsnis. Su procesu susijusių taisyklių nustatymas ir atvaizdavimas AD modelio elementais. Iš VM saugyklos pagal pasirinktą procesą išrenkamos susijusios veiklos taisyklės (ProcFunkc->Realizacija, Interpretacija, Apdorojimas), kurios kuriamame AD1 modelyje atvaizduojamas Control Node elementais: Decision Node, Merge Node, Fork Node, Join Node.

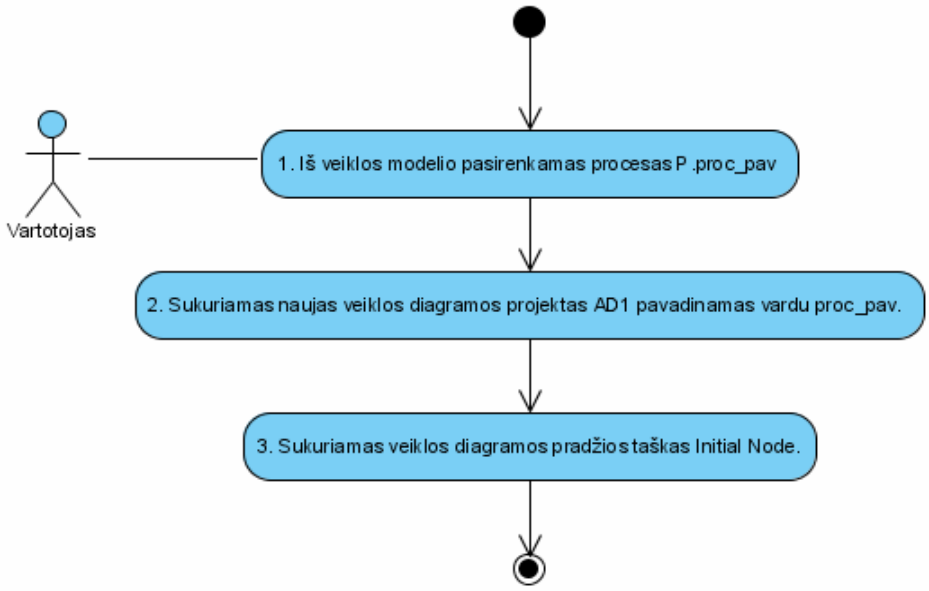
5 žingsnis. Pasirinkto proceso veiklų srauto užbaigimas. Atvaizdavus visus pasirinkto proceso įvykius, bei juos siejančius informacinius bei materialius srautus AD1 modelyje yra atvaizduojamas Activity Final Node simbolizuojanti srauto pabaigą.

6 žingsnis. Veiklos 9activity) diagramos užbaigimas, atliekamas įterpiant Flow Final Node.

4.1. Formalizuotas algoritmo aprašymas

Šiame skyriuje pateikiamas formalizuotas IS veiklos (activity) modelio generavimo veiklos modelio pagrindu algoritmo žingsnių aprašymas.

1 žingsnis.



9 pav. Algoritmo 1 žingsnis

1 žingsnio aprašymas

1. Veiklos diagramos generavimas pradedamas tuomet, kai sistemos vartotojas nurodo konkretų procesą P 9 VM elementą Procesas su atributu proc_pav), kuriam remiantis VM saugyklos duomenimis norima atlikti generavimą.

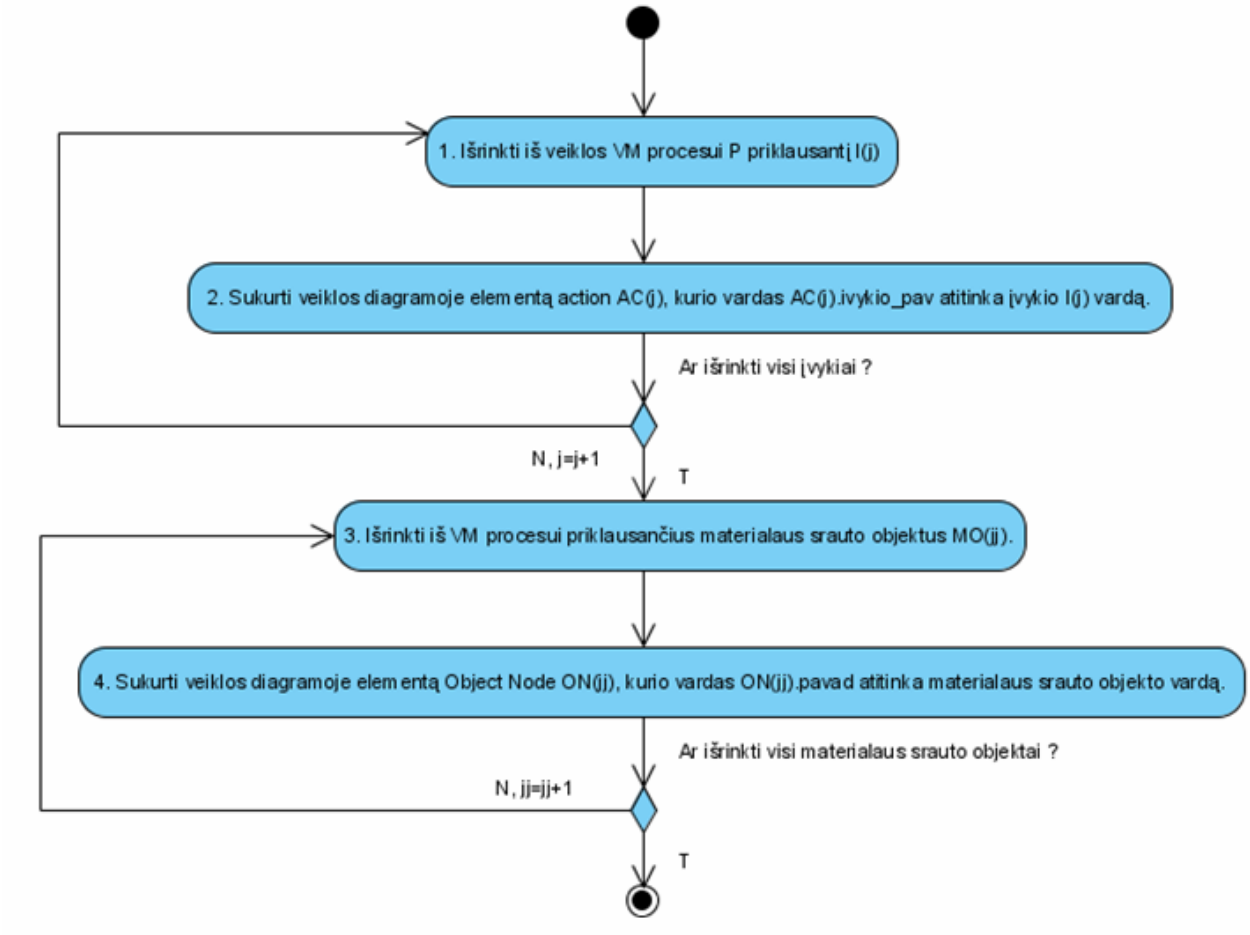
2. Atlikus proceso išrinkimą yra inicijuojamas naujo veiklos diagramos projekto AD1 kūrimas. Projekto pavadinimui priskirimas proceso vardas proc_pav.

Procesas.proc_pav -> ActivityDiagram.ActivityModelPav

3. Veiklos diagramos projekte AD1 sukuriamas veiklos diagramos pradžios taškas Initial Node elementas simbolizuojantis diagramos pradžią.

10 pav. Veiklos (activity) diagrama AD1 po pirmo žingsnio.

2 žingsnis.



11 pav. Algoritmo 2 žingsnis

2 žingsnio aprašymas

1. Iš veiklos modelio išrenkamas įvykis I(j), kuris priklauso vartotojo išrinktam procesui P.

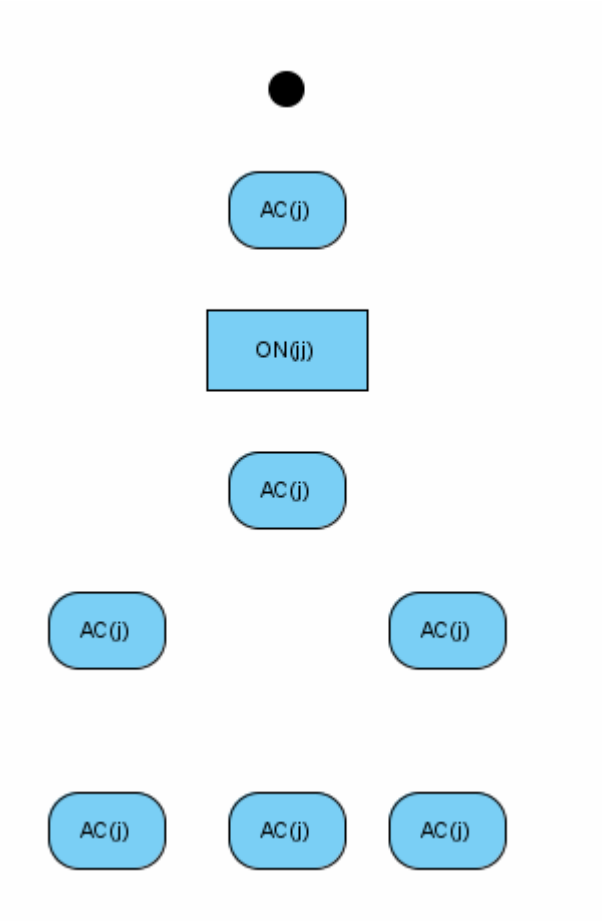
2. Veiklos diagramos modelyje AD1 sukuriamas elementas action AC(j), kurio vardo reikšmė išrinkto įvykio vardo reikšmė I(j).ivykio_pav, bei sukuriamas nuoroda į atitinkamą VM elemento egzempliorių.

VMM.Ivykis -> AMM.AcivityNode

3. Iš veiklos modelio išrenkami materialaus srauto objektai MO(jj), kurie priklauso vartotojo išsirinktam procesui P.

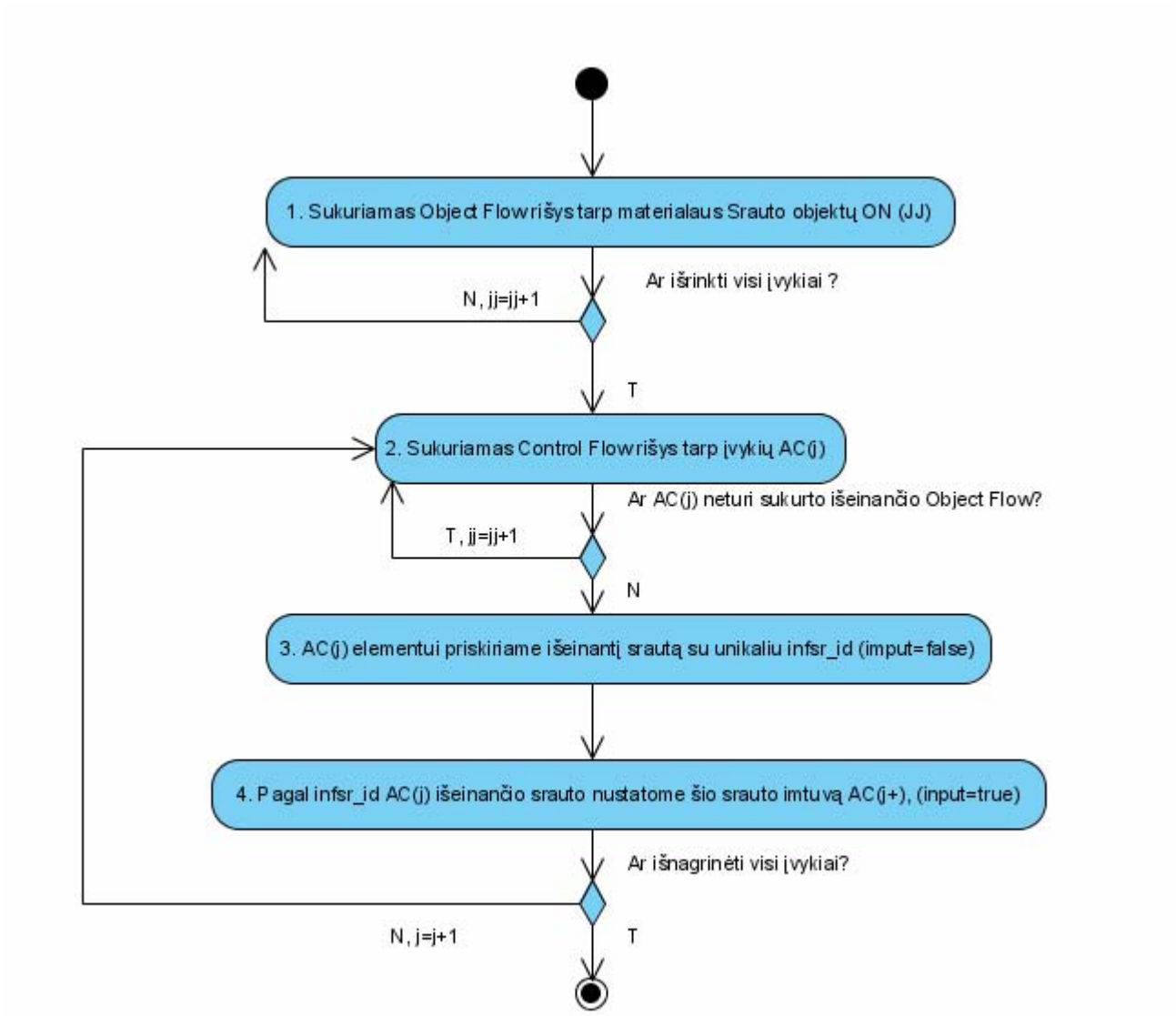
4. veiklos diagrams modelyje sukuriamas elementas ObjectNode ON(jj), kurio vardo reikšmei priskiriama materialaus srauto atributo pavadinimas MO(jj).atributas, ir sukuriama nuoroda į atitinkamą VM elemento egzempliorių.

VMM>MSrAtributai -> AMM.ObjectNode



12 pav. Veiklos (activity) diagrama AD1 po antro žingsnio.

3 žingsnis.

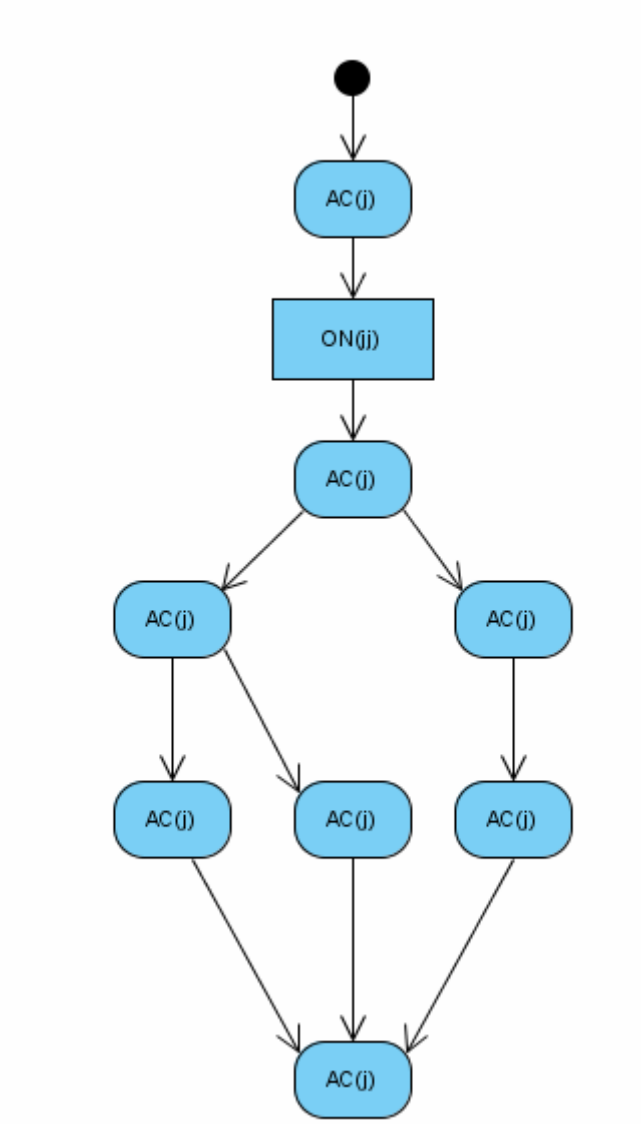


13 pav. Algoritmo 3 žingsnis

3 žingsnio aprašymas

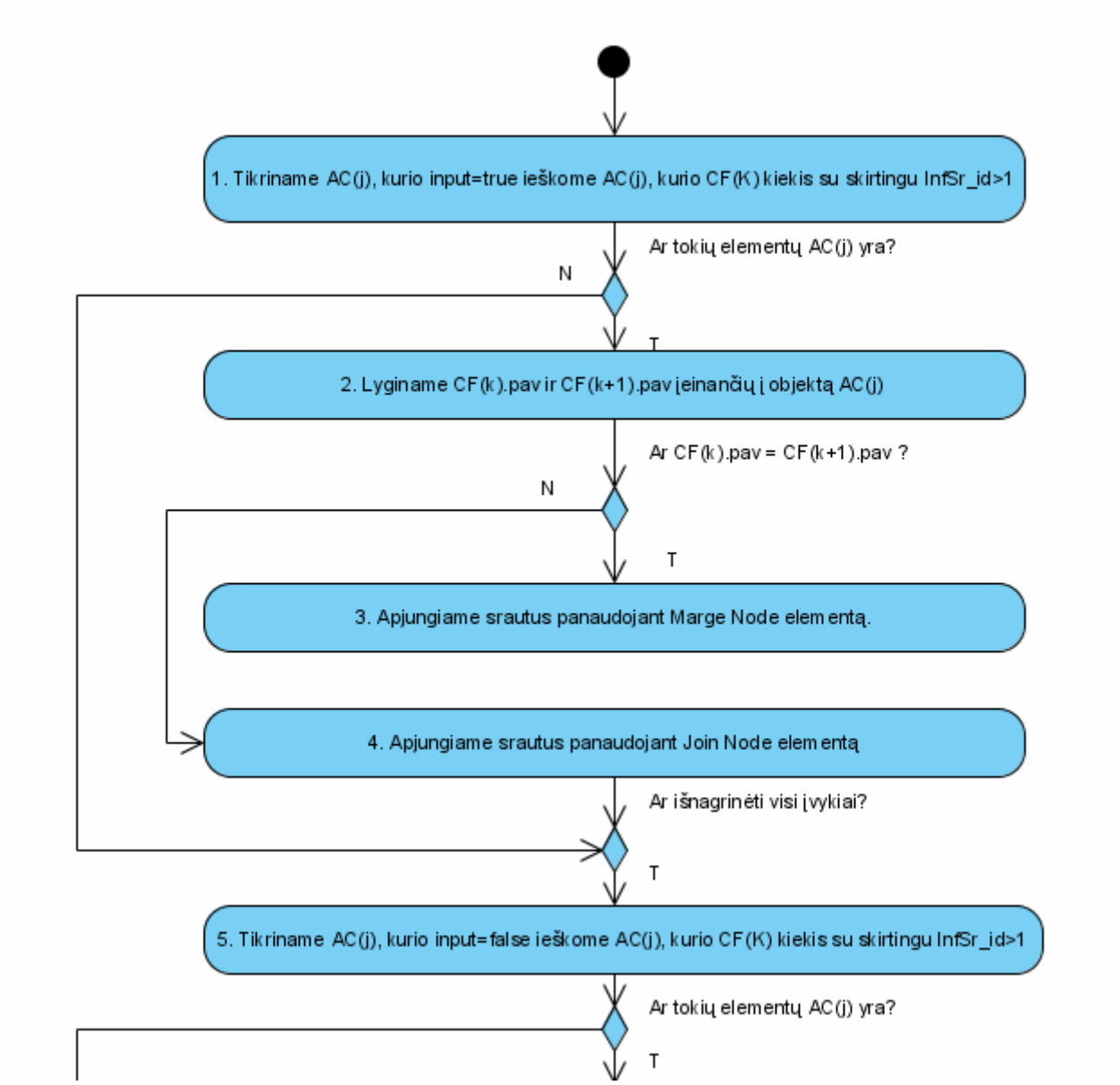
1. Veiklos diagramos AD1 modelyje visiems esantiems materialaus srauto elementams ObjectNode ON(jj) sukuriamas juos apjungiantis materialus srautas ObjectFlow, srautas turi įeiti ir išeiti į Object Node. Tikrinama ar ON(jj) input vertė yra true ir ieškome, kuriam įvykiui AC(J) jis yra priskirtas., t.t. iš kurio įvykio į materialaus srauto objektą išeina Object Flow. Object Flow išeinantis iš objekto patenka į įvykį AC(jj), kurio priskirto ON(jj) input vertė yra false.
2.Veikos diagramos modelyje AD1 tarp objektų action AC(j) sukuriame informacinius srautus ControlFlow. Patiriname ar AC(j) neturi sukurto išeinančio materialaus srauto Object Flow. Jei turi pereiname į 3 žingsnį, jei ne į 4 žingsnį.
3. Jei AC(j) jau turi išeinanti ObjectFlow tipo srautą pereiname prie sekančio AC(j).

4. Jei AC(j) neturi jokio išeinančio srauto, tuomet VM tikriname, kurie informaciniai srautai, pagal infsr_id yra priskiriami šiam įvykiui I9j) atitinkančiam elementui AC(j), kuomet jų input vertė yra false (t.y. atvaizduojami iš AC(j) išeinantys srautai).
5. Pagal unikalų infsr_id nustatome kuriems AC(j) input vertė yra true (t.y. nustatomi įvykiai, į kuriuos yra nukreipiami srautai).

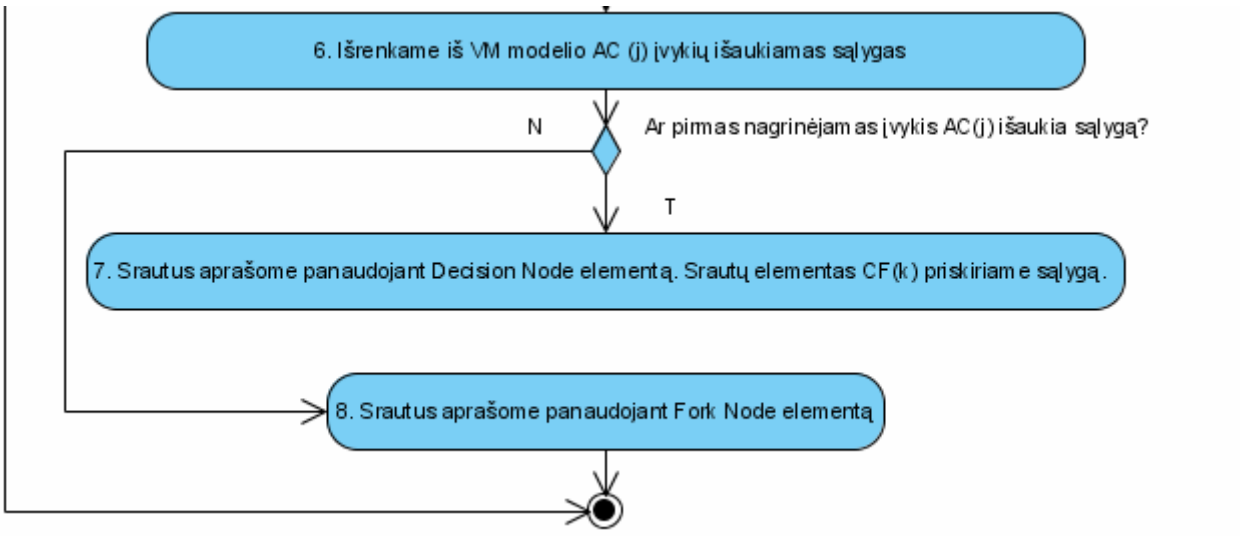


14 pav. Veiklos (activity) diagrama AD1 po trečio žingsnio.

4 žingsnis.



15 pav. Algoritmo 4 žingsnis

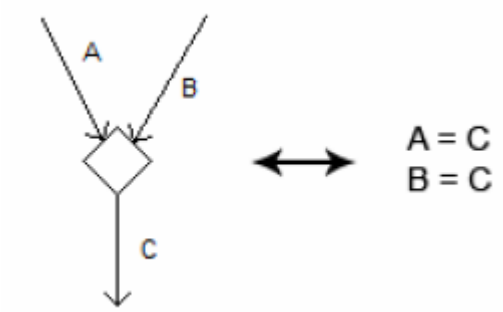


16 pav. Algoritmo 4 žingsnis, tęsinys

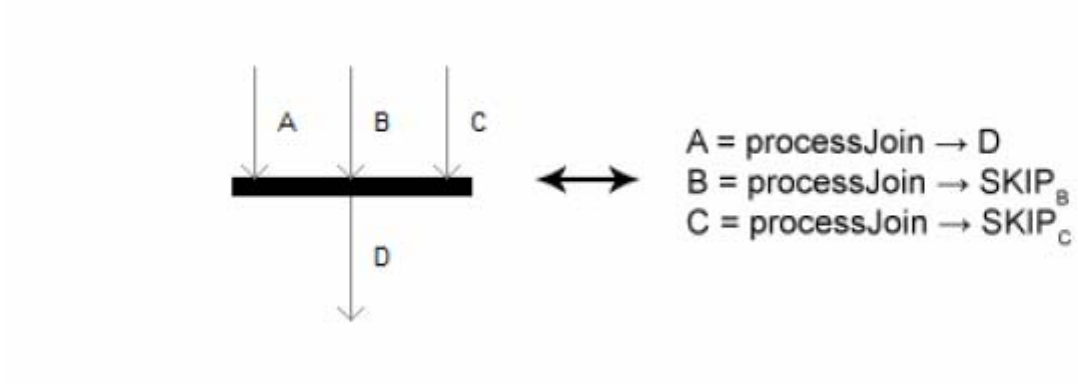
4 žingsnio aprašymas

1. Tikriname AD1 modelio elementus action AC(j), kurių informacinio srauto Control Flow input vertė yra true. Jei į AC(j) įeina daugiau negu vienas Control Flow su skirtingu InfSr_id tikriname, ar Control Flow CF(k).pav = CF(k+1).pav įeinančiam į objektą AC (j). Jei taip pereiname į 2 žingsnį, priešingu atveju į 3 žingsnį.

2. Jei CF(k).pav= CF(k+1).pav, t.y. į action objektą įeina du arba daugiau vienodai nusakomų srautų vadinasi srautus apjungiame panaudojant Merge Node elementą.

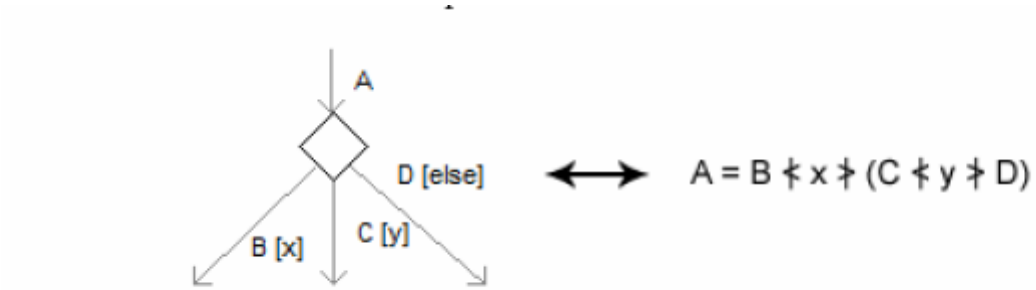


3. Jei CF(k).pav != CF(k+1).pav, t.y. į action objektą įeina du ar daugiau skirtingų srautų, vadinasi srautus apjungiame panaudojant Join Node elementą, į objektą AC(j_ įeinančiu paliekant pirmąjį srautą CF(K).pav.

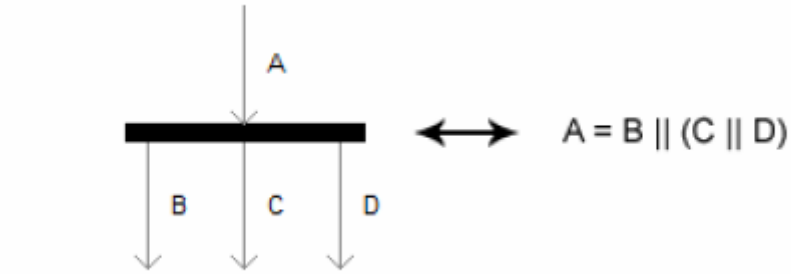


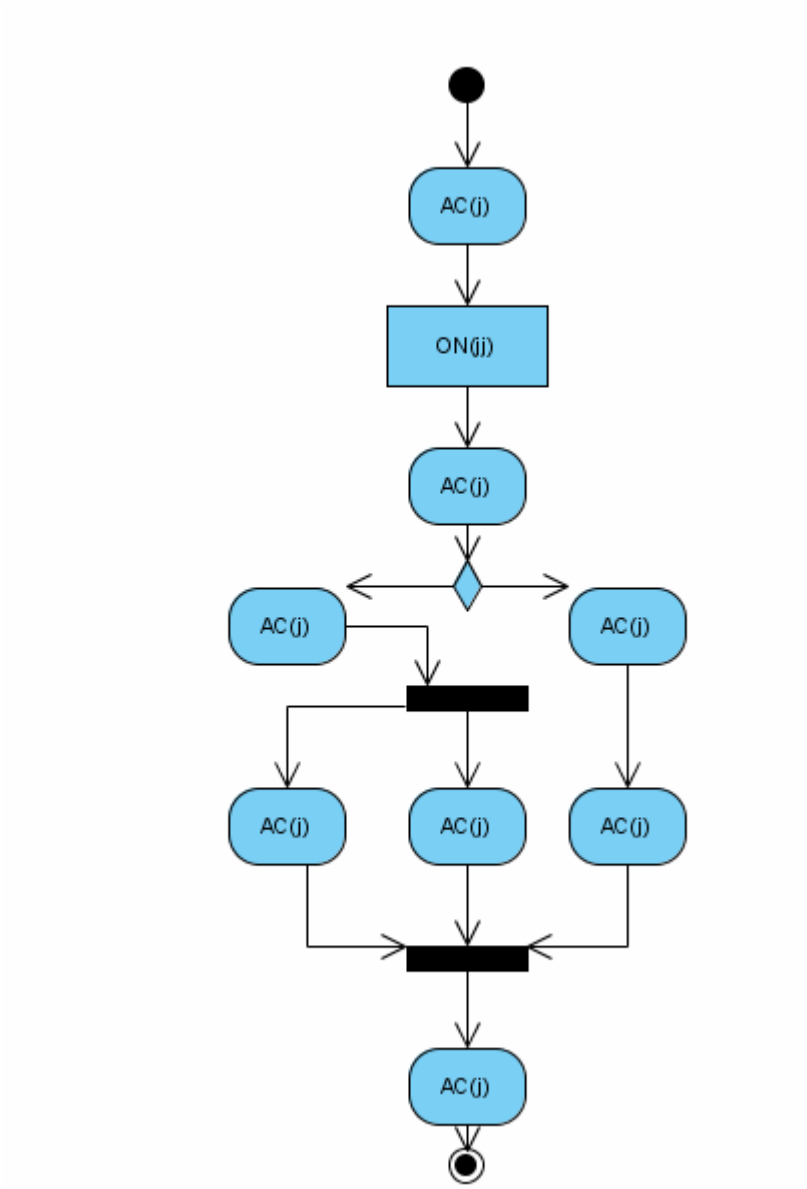
4. Tikriname AD1 modelio elementas action AC(j), kurio informacinio srauto ControlFlow input vertė yra false . Jei iš AC(j) išeina daugiau negu vienas Control Flow su skirtingų InfSr_id tikriname, ar įvykis AC(j) neįšaukia tikrinimo sąlygos. Jei tikrinimo sąlyga yra įšaukiama pereiname prie 5 žingsnio, jei ne prie 6 žingsnio.

5. Įvykis AC(j) išaukia tikrinimo sąlygą. Srautus aprašome panaudojant Decision Node elementą bei nustatome sąlyginių srautų priskirimą elementams, į kuriuos jie įeina.



6. Įvykis AC(j) neįšaukia tikrinimo sąlygos, srautus aprašome panaudojant ForkNode elementą.





(5,6 žingsniais įterpiamų elementų iliustracija)

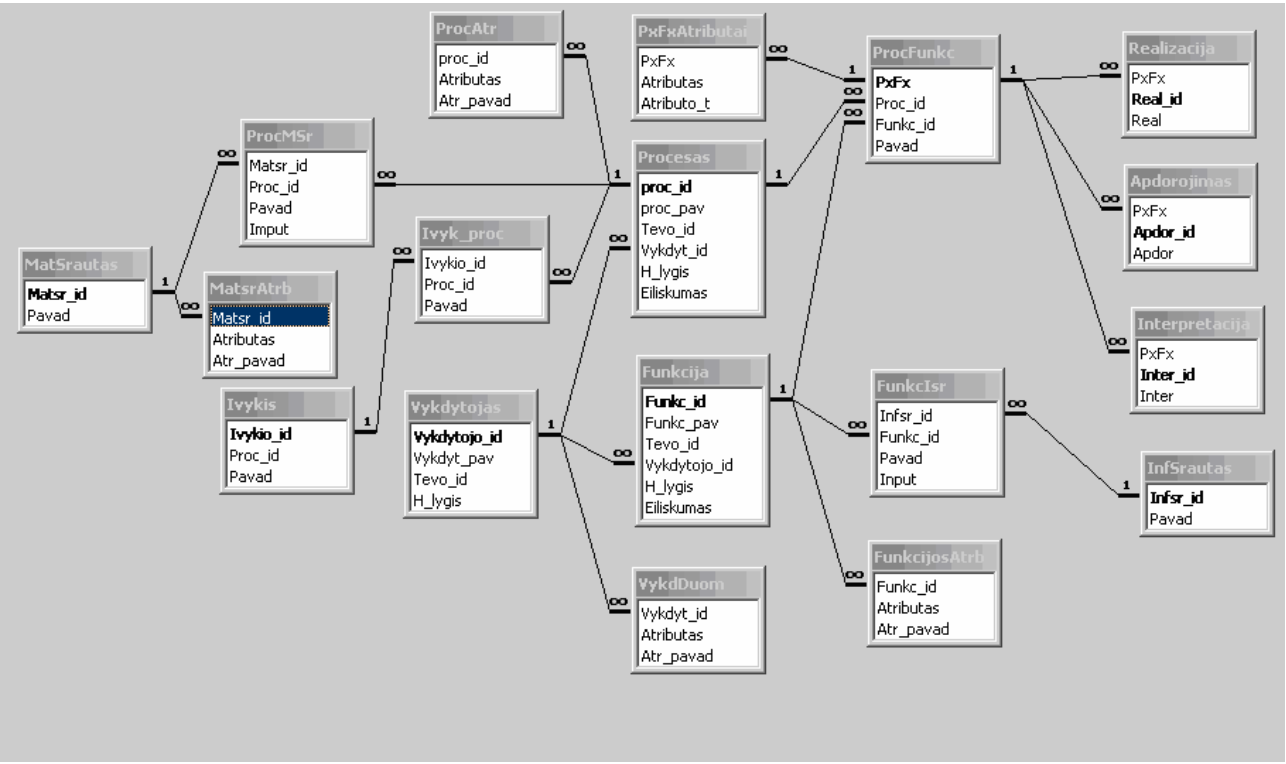
17 pav. Veiklos (activity) diagrama AD1 po ketvirto žingsnio.

5,6 žingsnis.

Veiklos (activity) diagramą užbaigiame įterpiant srauto ir diagramos užbaigimo žymes FinalNode ir Flow Fina Node.

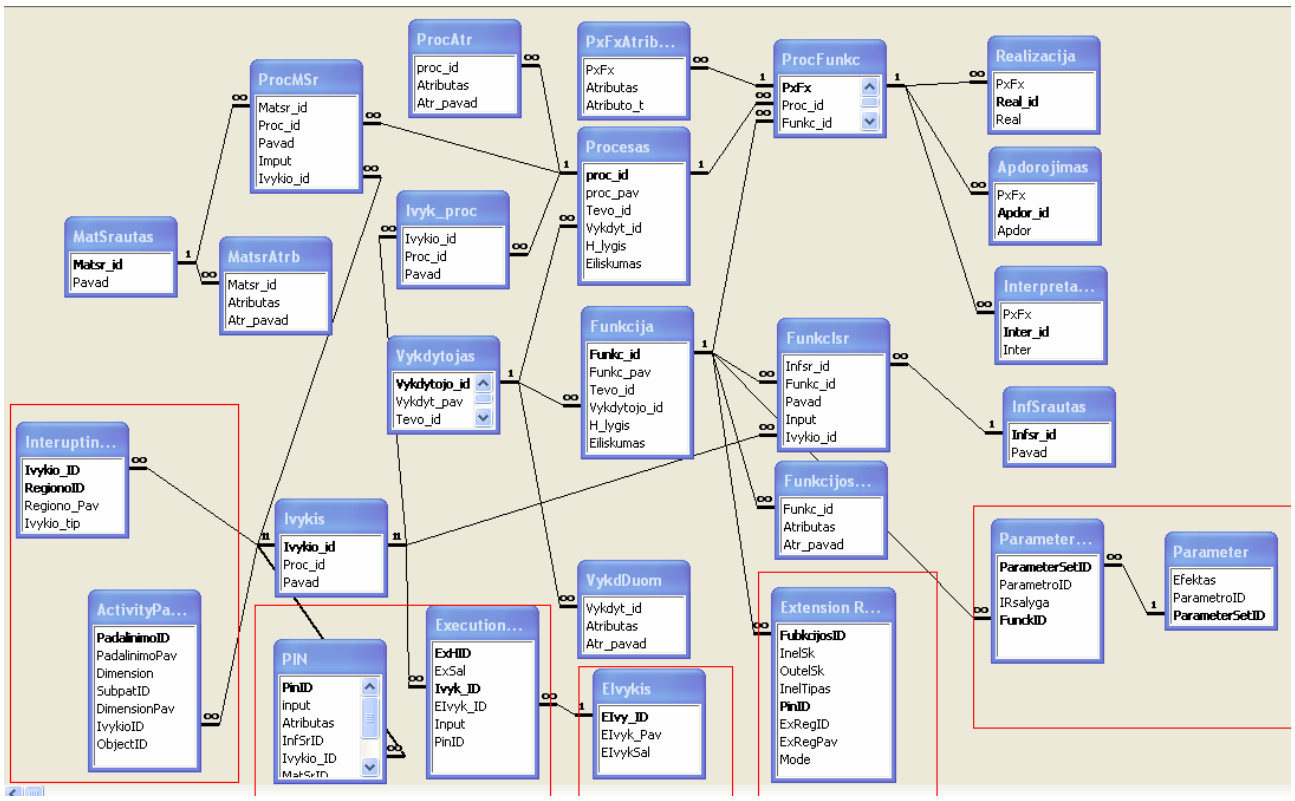
5. Veiklos (activity) diagrmai generuoti naudojama duiomenų struktūra

Experimentinės duomenų bazės pradinė struktūra, kuria remiantis galima sugeneruoti bendrus veiklos (activity) diagramos elementus. Detalesni laukų aprašymai pateikiami 1 priede.



18 pav. Eksperimentinės duomenų bazės struktūra

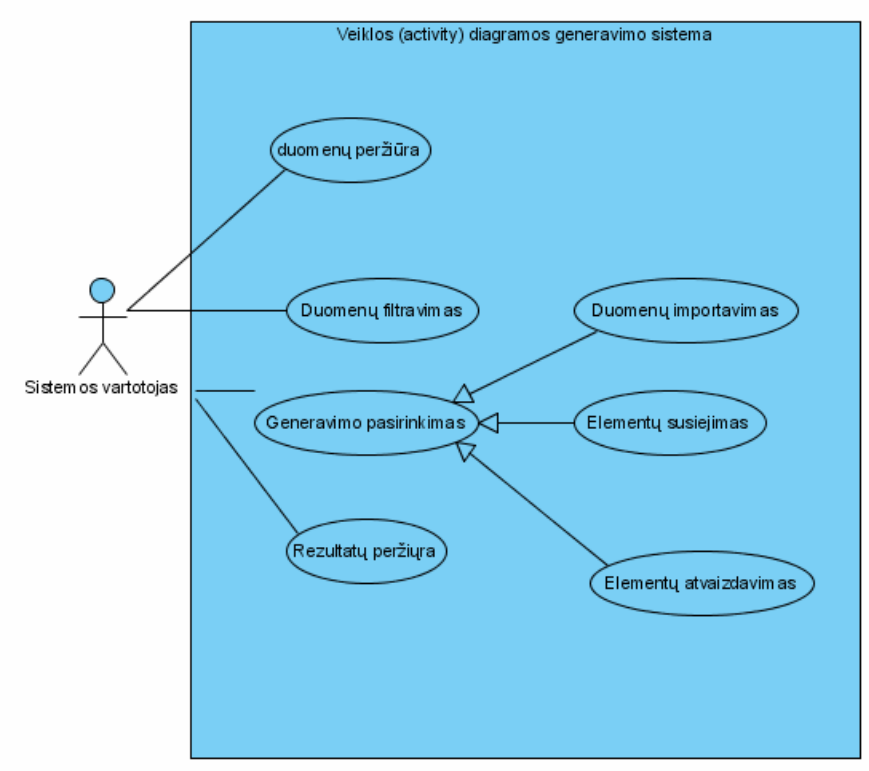
Įvertinus BPM ir UML notacijų panašumus ir skirtumus, sudaromas klasių rinkinys generuoti veiklos (activity) diagramos elementams, kurių darbų sekos (workflow) nepajėgia aprašyti ir tiems, kurie gali būti atvaizduoti abėjose notacijose, tačiau veiklos klasių modelyje atitinkamos klasės neturi.



19 pav. Pakoreguota eksperimentinės duomenų bazės struktūra

6. Veiklos (activity) diagramos generavimo elgsenos modelis

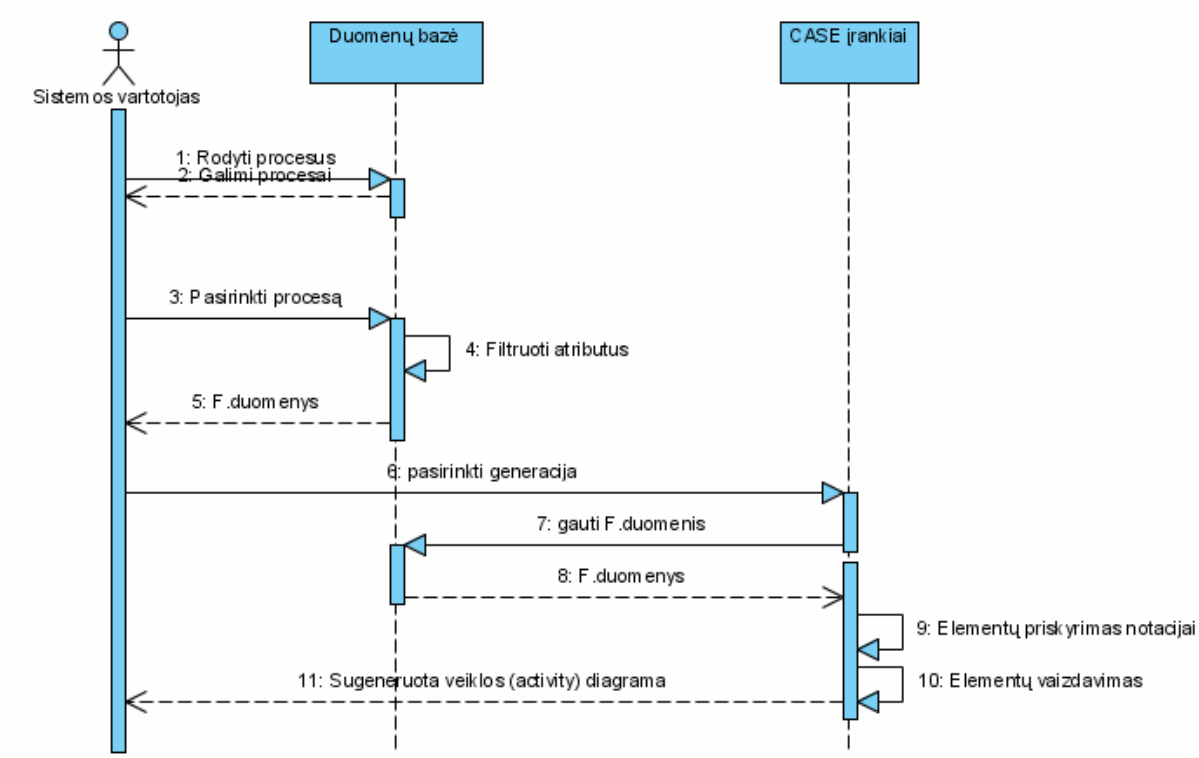
6.1. Panaudojimo atvėjų modelis



20 pav. Veiklos (activity) diagramos generavimo vartojimo atvejų diagrama

Sukurtos prototipinės veiklos (activity) diagramos generavimo sistema yra susieta su duomenų valdymo sistema, leidžiančia vartotojui peržiūrėti patalpintus duomenis, bei filtruoti juos pagal diagramos generacijai pasirinktą procesą. CASE įrankyje pašalinamas interaktyvumo faktorius, kadangi diagramos kūrimo procese vartotojas dalyvauja tik pasirinkdamas diagramos generavimą ir peržiūrėdamas galutinį gautą rezultatą, visas diagramos generavimo procesas atliekamas algoritmo pagalba.

6.2. Sekų diagrama



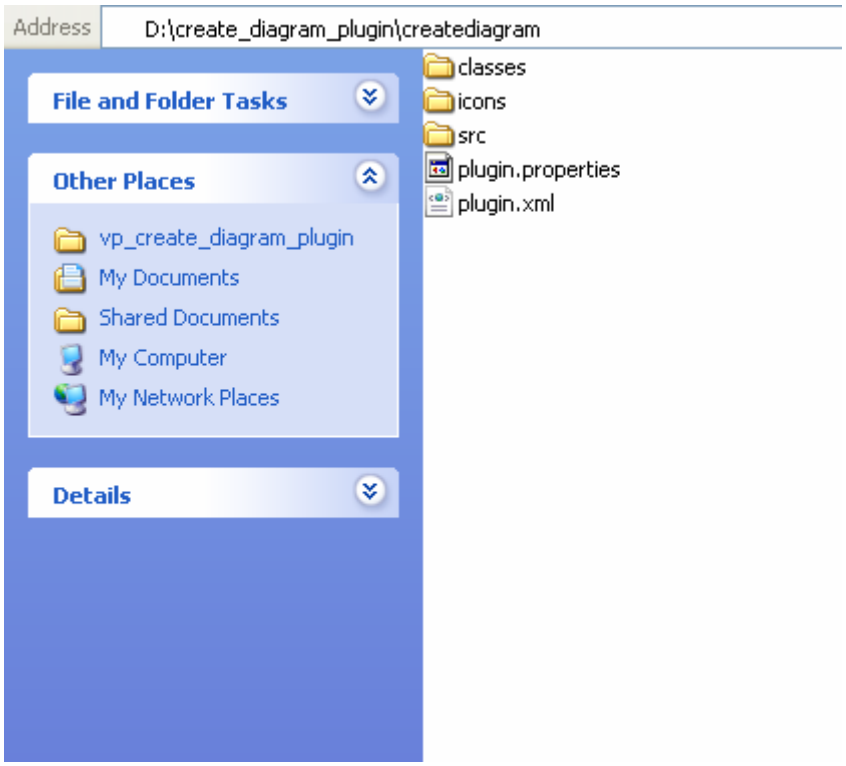
21 pav. Veiklos (activity) diagramos generavimo sekų diagrama

Duomenų valdymo sistemoje vartotojas gali peržiūrėti visus į saugyklą patalpintus procesus, kuriuos galima pasirinkti generavimo procesui vykdyti. Pasirinkus vieną iš galimų procesų ir CASE įrankyje pasirinkus generavimo funkciją, pradedamas fitruotų proceso elementų nuskaitymas iš duomenų bazės. Nuskaitytiems duomenų saugyklos elementams priskiriamos atitinkamos UML veiklos (activity) diagramos notacijos. Atlikus priskyrimo operaciją sugeneruota diagrama yra pateikiama vartotojo vertinimui.

7. Algoritmą realizuojančio įrankio prototipas

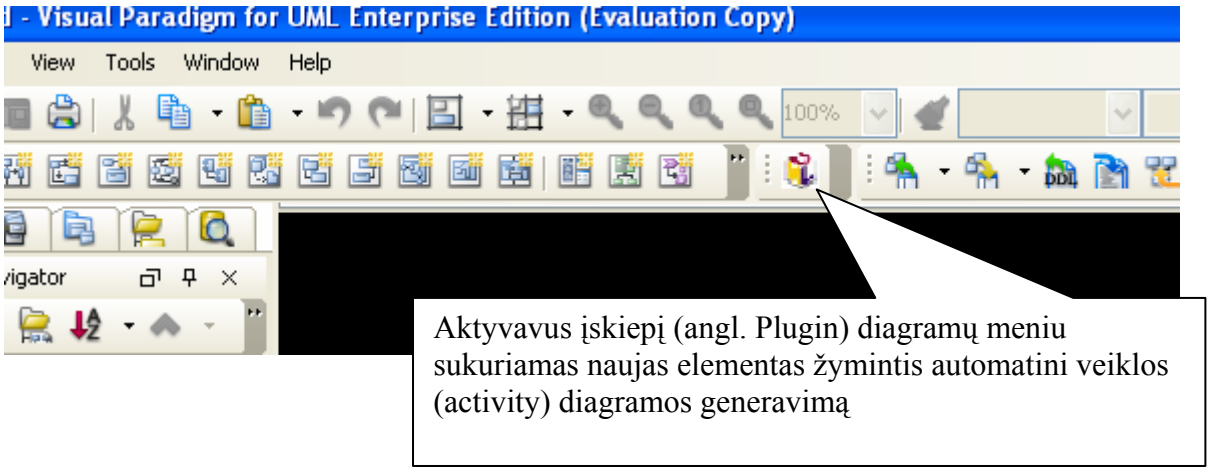
Veiklos (activity) diagramos generavimo veiklos modelio pagrindu prototipui kurti pasirenkamas UML notaciją palaikantis Visual Paradigm CASE įrankis. Prototipas kuriamas įskiepio įdiegimu į CASE įrankį principu.

Sukurti Visual Paradigm įskiepio failai saugomi CASE įrankio įdiegimo direktorijoje.



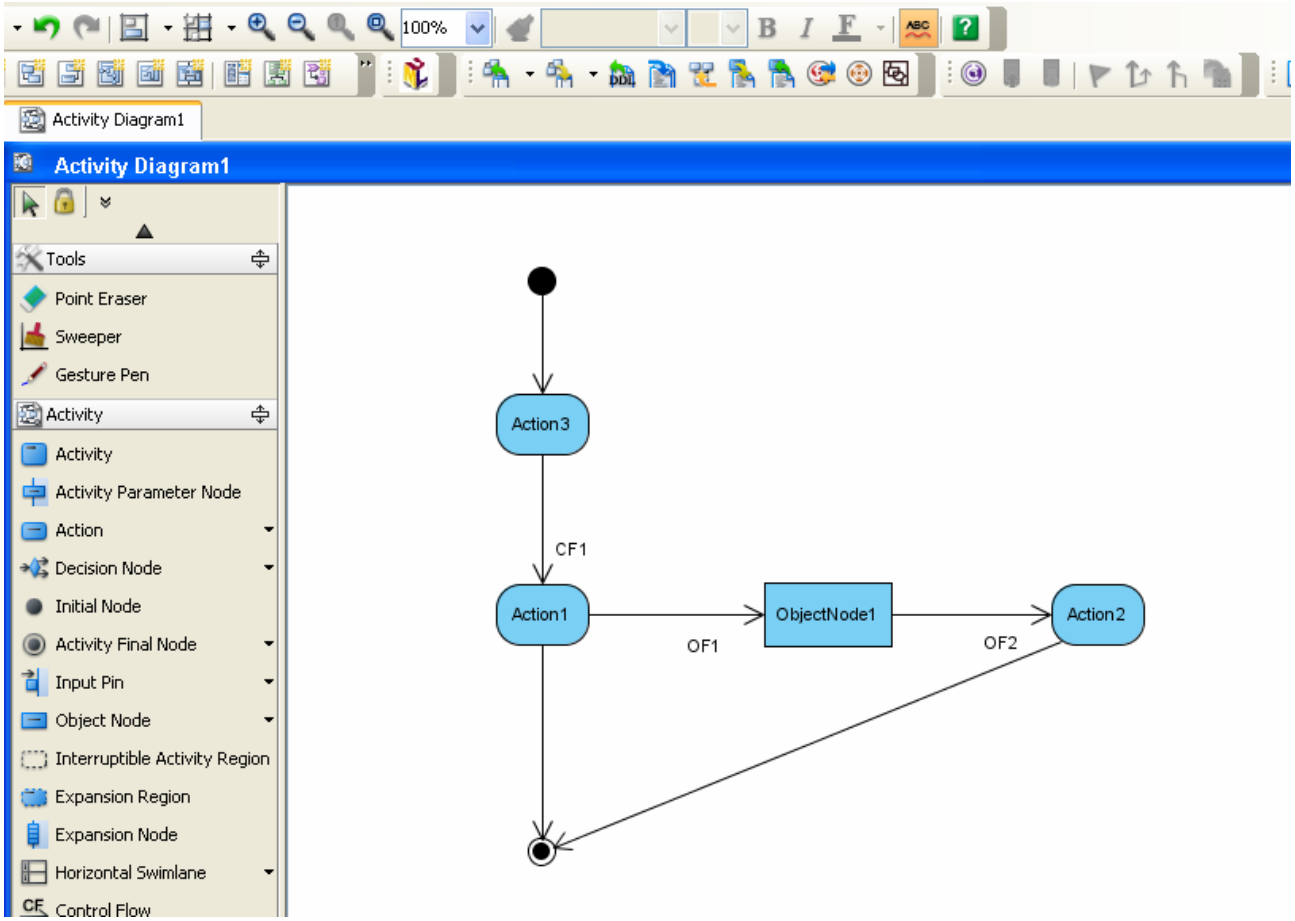
22 pav. Prototipo vykdomieji failai

Aktyvavus sukurtą įskiepį CASE įrankio lange atsiranda naujas įrankių juostos elementas, žymintis naujai sukurtą funkciją.



23 pav. Veiklos (activity) diagramos generavimo funkcija įrankiu juostoje

Pasirinkus naujos funkcijos vykdyma pagal nuskaitytus atrinktus pasirinkto proceso duomenis įvykdžius algoritmą pateikiamas sugeneruotas veiklos (activity) modelis.



24 pav. Veiklos (activity) diagramos generavimo funkcijos realizacija

8. Rezultatų patvirtinimo eksperimentas

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 veiklos (activity) diagramos generavimo algoritmo teisingumą. Patikrinimą atliksime su pavyzdinių duomenų aibe. Pažymėtina, kad iliustruojame ne visų veiklos modelio klasės diagramos lentelių užpildymą, o tik tų, kurios mums bus reikalingos testavimui atlikti.

Žemiau pateiktomis lentelėmis iliustruojame veiklos modelio saugyklos lentelių, reikalingų sistemos veiklos (activity) diagramai kurti testinius duomenis.

Lentelė „Funkcija“
Funkc_id=0001
FunkcPav=“Užsakymo valdymas“
Tėvo_id=0001
Vykdyt_id=0001
H_lygis=1
Eiliškumas=1

Lentelė „Procesas“
Proc_id=0001
Proc_pav=“Prekių užsakymo valdymas “
Tėvo_id=0001
Vykdyt_id=0001
H_lygis=1
Eiliškumas=1

Lentelė „MatSrautas“
Matsr_id=0001
Pavad=“Prekė “

Lentelė „MatSrautas“
Matsr_id=0003
Pavad=“Pinigai “

Lentelė „MatSrautas“
Matsr_id=0002
Pavad=“Dovana “

Lentelė „MatSrautas“
Matsr_id=0004
Pavad=“Saskaita “

Lentelė „MatsAtrb“
Matsr_id=0001
Atributas=1
Atr_pavad=”Pakuotė”

Lentelė „MatsAtrb“
Matsr_id=0003
Atributas=2
Atr_pavad=”Apskaita”

Lentelė „InfSrautas“
Infsr_id=0001
Pavad=“Užsakymas “

Lentelė „MatsAtrb“
Matsr_id=0004
Atributas=2
Atr_pavad=”Apskaita”

Lentelė „InfSrautas“
Infsr_id=0002
Pavad=“Sąmata “

Lentelė „ProcMSr“
Matsr_id=0001
Proc_id=0001
Pav=”Prekė”
imput=true
Įvykio_id=0004

Lentelė „ProcMSr“
Matsr_id=0003
Proc_id=0001
Pav=”Pinigai”
imput=true
Įvykio_id=0003

Lentelė „ProcMSr“
Matsr_id=0002
Proc_id=0001
Pav=”Dovana”
imput=false
Įvykio_id=0006

Lentelė „ProcMSr“
Matsr_id=0004
Proc_id=0001
Pav=”Saskaita”
imput=false
Įvykio_id=0007

Lentelė „Ivykis“
Ivykio_id=0001
Įvykio_pav=“Apibrezti poreiki“
Proc_id=0001

Lentelė „Ivykis“
Ivykio_id=0005
Įvykio_pav=“Pristatyti uzsakyma“
Proc_id=0001

Lentelė „Ivykis“
Ivykio_id=0002
Įvykio_pav=“Priimti uzsakyma“
Proc_id=0001

Lentelė „Ivykis“
Ivykio_id=0006
Įvykio_pav=“Suruosti krovini“
Proc_id=0001

Lentelė „Ivykis“
Ivykio_id=0003
Įvykio_pav=“Nustatyti kaina“
Proc_id=0001

Lentelė „Ivykis“
Ivykio_id=0007
Įvykio_pav=“Suruosti dokumentus“
Proc_id=0001

Lentelė „Ivykis“
Ivykio_id=0004
Įvykio_pav=“Vykdyti uzsakyma“
Proc_id=0001

Lentelė „FunkcISr“
funkc_id=0001
infsr_id=0001
pavad=”Uzsakymas”
imput=false
Įvykio_id=0001

Lentelė „FunkcISr“
funkc_id=0001
infsr_id=0001
pavad=”Uzsakymas”
imput=true
Įvykio_id=0002

Lentelė „FunkcISr“
funkc_id=0001
infsr_id=0002
pavad=”Samata”
imput=false
Įvykio_id=0002

Lentelė „FunkcISr“
funkc_id=0001
infsr_id=0003
pavad=” Transporto uzsakymas”
imput=true
Įvykio_id=0005

Lentelė „FunkcISr“
funkc_id=0001
infsr_id=0003
pavad=”Transporto uzsakymas”
imput=false
Įvykio_id=0006

Lentelė „FunkcISr“
funkc_id=0001
infsr_id=0002
pavad=”Samata”
imput=true
Įvykio_id=0003

Lentelė „FunkcISr“
funkc_id=0001
infsr_id=0003
pavad=” Transporto uzsakymas”
imput=false
Įvykio_id=0007

Lentelė „FunkcISr“
funkc_id=0001
infsr_id=0002
pavad=”Samata”
imput=true
Įvykio_id=0004

1 žingsnis

Tarkime, kad pasirenkame procesą „Prekių užsakymo valdymas“, kuriam ir yra suformuoti eksperimentiniai duomenys. Naujasis veiklos (activity) diagramos projektas tuomet pavadinamas „Prekių užsakymo valdymas“ atliekant tokį duomenų perdavimą tarp duomenų saugyklų.

VMM.Procesas.Proc_pav -> AMM.ActivityDiagram.ActivityModelPav

Atlikus tokį duomenų nuskaitymą funkcijos pagalba pasinaudojant klase „Taisyklė“ į veiklos (activity) diagramos projektą įterpiama InitialNode žymė su pradiniu įvykį inicijuojančiu srautu.

2 žingsnis

Išrenkame procesui „Prekių užsakymo valdymas“, kurio id = 0001 priklausančius įvykius, ir juos atvaizduojame priskiriant įvykio pavadinimą. Tarp duomenų saugyklų įvykdomas toks apsikeitimas duomenimis.

VMM.Ivykis.Proc_id -> AMM.Action.ActionID
VMM.Ivykis.Ivykio_pav -> AMM.Action.ActionPav

Išskiriami tokie proceso „prekių užsakymo valdymas“ įvykiai:
AC(0001).apibrezti poreiki
AC(0002).priimti uzsakyma
AC(0003).nustatyti kaina
AC(0004).vykdyti uzsakyma
AC(0005).pristatyti uzsakyma
AC(0006).suruošti krovini
AC(0007). Suruošti duomenis

Išrenkami pasirinktam procesui priklausantys materialaus srauto objektai. Objektus išrenkame iš lentelės „MatsrAtrb“ ir atvaizduojame į lentelę „Objektas“:
VMM.MatsrAtrb.Matsr_id -> AMM.Objektas.SrautoID
VMM. MatsrAtrb.MatObID -> AMM.Objektas.MatObID
VMM. MatsrAtrb.Atr_pav -> AMM.Objektas.MatObPav

Išskiriami tokie proceso „prekių užsakymo valdymas“ materialaus srauto objektai, kurie atvaizduojami panaudojant Object node notaciją.

ON(1).Pakuote
ON(2).Apskaita

3 žingsnis

Sukuriamas atrinktus materialaus srauto objektus apjungiantis materialus srautas. Čia susiduriame su pirmąja problema materialaus srauto ir įvykio lentelės pasiūlytoje klasių diagramoje nėra susietos atributu leidžiančiu įvykiui priskirti materialų srautą, todėl sukuriame papildomą ryšį tarp lentelių „ProcMSr“ ir įvykio tarpinės lentelės „Ivyk_proc“ bei lentelėje „ProcMSr“ sukuriame papildomą atributą „Įvykio_id“
Atrenkame visus materialaus srauto objektus, kurių materialaus srauto input vertė lentelėje „ProcMSr“ yra true ir atliekame tokį atvaizdavimą tarp saugyklos.

VMM.ProcMSr.Matsr_id -> AMM.Srautas.SrautoID
VMM.ProcMSr.Pav -> AMM.Srautas.SrautoPav

VMM.ProcMSr.Ivykio_id -> AMM.Srautas.ActionID
VMM.ProcMSr.input -> AMM.Srautas.input

Pagal duomenis nustatome, kad sąlyga *input = true* tenkina tokie materialūs srautai:

Matsr_id = 0001	*Matsr_id = 0003*
Pav = "Preke"	*Pav = "Pinigai"*
Ivykio_id = 0004	*Ivykio id = 0003*

Materialialūs srautai, kurių sąlyga *input = false* yra:

Matsr_id = 0002	*Matsr_id = 0004*
Pav = "Dovana"	*Pav = "Saskaita"*
Ivykio_id = 0006	*Ivykio id = 0007*

Pagal atsrinktus duomenis gauname, kad į materialaus srauto objektą:
ON(1).Pakuote įeina materialus srautas, kurio id = 0001, t.y., „Prekė" ir jis yra nukreipiamas iš įvykio, kurio id = 0004, t.y., "Vykdyti uzsakyma".
ON(2).Apskaita įeina materialus srautas, kurio id = 0003, t.y., „Pinigai" nukreipiamas iš įvykio "Nustatyti kaina".
Iš ON(1).Pakuote išeina materialus srautas "Dovana" į įvykį "Paruošti krovinį".
Iš ON(2).Apskaita išeina materialus srautas "Saskaita" į įvykį "Suruošti dokumentus".

Toliasniais veiksmais tarp veiklos (activity) diagramos projekte atvaizduotų įvykių sukuriami informaciniai srautai. Pirmiausia patikriname, ar atvaizduoti įvykiai neturi sukurto išeinančio materialaus srauto, tai atliekame pagal materialių srautų atvaizdavimo etapą, pagal ActioID sąlyga input = true. Šiuos įvykio elementus atmetame kaip jau įvykdytus ir pereiname prie įvykių, kurių ActionID su sąlyga input = false.

Atrenkame visus įvykius, kurių input=false
AC(0001).apibrezti poreiki
AC(0002).priimti uzsakyma
AC(0005).pristatyti uzsakyma
AC(0006).suruosti krovini
AC(0007).suruosti dokumentus

Susiduriame su jau matyta problema, kad tarp įvykio ir informacinio srauto lentelių nėra sukurta ryšio. Todėl sukuriame naują ryšį, o lentelę "FunkcISr" papildome "ivykio_id" atributu. Tuomet galime atlikti tokį atvaizdavimą tarp duomenų saugyklų.

VMM.FunkcISr.infsr_id -> AMM.Srautas.SrautasID
VMM.FunkcISr.pavad -> AMM.Srautas.SrautasPav
VMM.FunkcISr.ivykio_id -> AMM.Srautas.ActionID
VMM.FunkcISr.input -> AMM.Srautas.input

Pagal gautas Srauto.input elemento reikšmes nustatome, kuris įvykis yra informacinio srauto šaltinis, o kuris imtuvas:

Šaltinis	**Imtuvas**
Srauto.input = false	*Srauto.input = true*
SrautoID = 0001 *SrautoPav =Usakymas* *ActionID = 0001*	*SrautoID = 0001* *SrautoPav =Usakymas* *ActionID = 0002*
SrautoID = 0002 *SrautoPav = Srautas* *ActionID = 0002*	*SrautoID = 0002* *SrautoPav = Srautas* *ActionID = 0003*
SrautoID = 0003 *SrautoPav = Transporto uzsakymas* *ActionID = 0006*	*SrautoID = 0002* *SrautoPav = Samata* *ActionID = 0004*
SrautoID = 0003 *SrautoPav = Transporto uzsakymas* *ActionID = 0007*	*SrautoID = 0003* *SrautoPav = Transporto uzsakymas* *ActionID = 0005*

Pagal identifikuotus šaltinius ir imtuvus atvaizduojame informacinius srautus.

4 žingsnis

Tikriname, ar tarp informaciniais srautas apjungtų įvykių nesusidaro taisyklių,t.y., tikriname, ar iš/Į kiekvieno įvykio ActionID išeina/įeina vienas informacinis srautas. Aptikus du ar daugiau srautų, tikriname, kuriai taisyklei jie priklauso.
Iš AC(0004).vykdyti uzsakyma išeina du srautai į AC(0002).priimti uzsakyma ir AC(0003).nustatyti kaina. Srautai gali būti paskirstyti panaudojant DecisionNode arba ForkNode. Čia susiduriame su klausimu, kurioje vietoje būtų patogiausia apibrėžti įvykio vykdymo sąlygą, todėl nusprendžiame sukurti papildomą klasę „Įvykio_sal“

Klasė „Ivykio_sal“	
Sąlygos_id	Integer
Salyga	Text
No_iv_id	Integer
Yes_iv_id	Integer
Įvykio_id	Integer

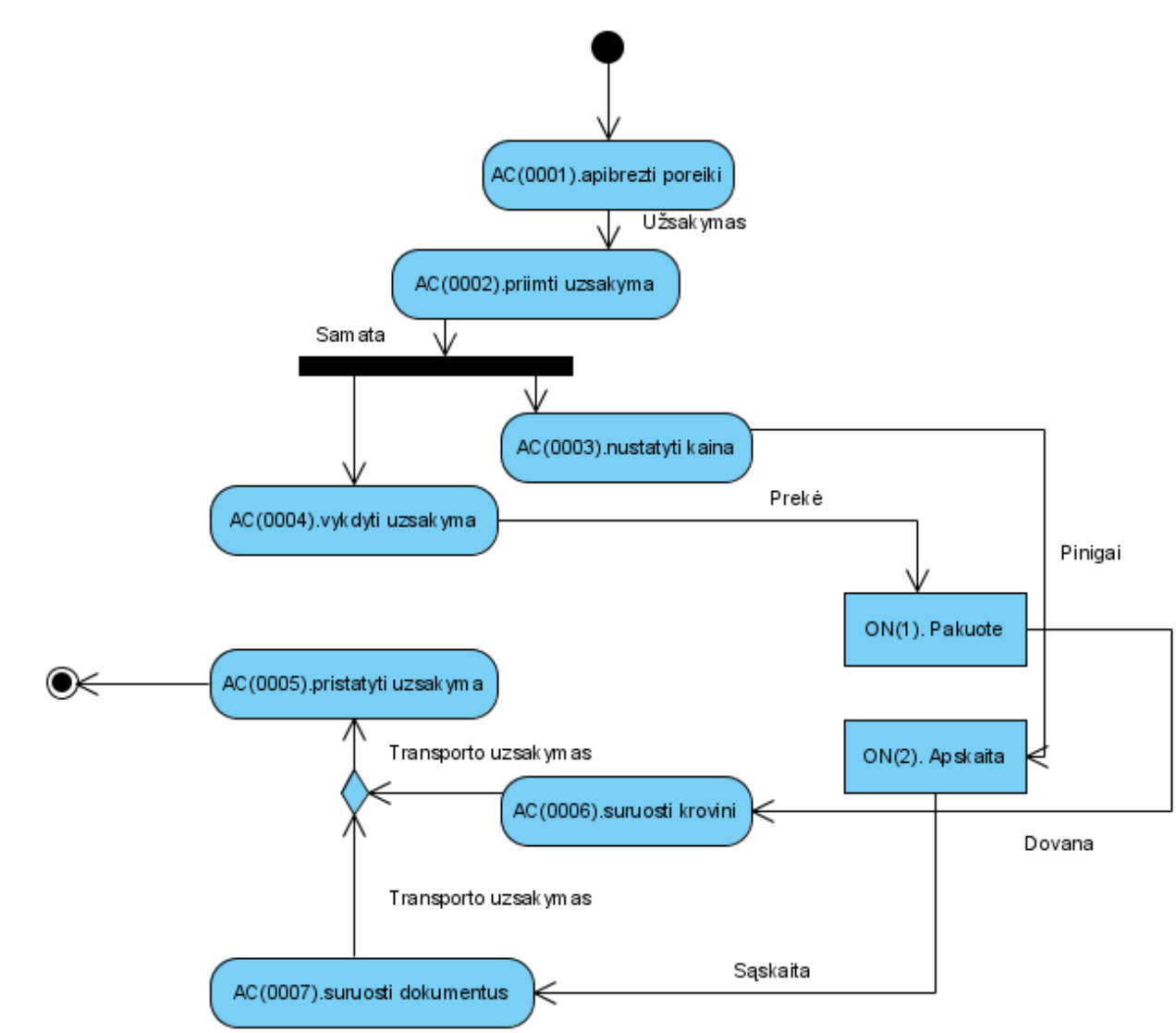
Ši klasė yra reikalinga norint įvykdyti Decision tipo taisykles.

Peržiūrėje pradinius duomenis matome, kad srautų atskirimo sąlyga nėra apibrėžta, vadinasi susidūrėme su Fork Node tipo taisykle, kurios notaciją ir įterpiame į dviejų informacinių srautų vietą. Toliau susiduriame su dviejų informacinių srautų suvedimu į vieną įvykį: srautai iš įvykių AC(0006).suruosti krovini ir AC(0007).suruosti dokumentus suplaukia į įvykį AC(0005).pristatyti užsakymą. Tikriname apjungimo taisykles, šioje situacijoje gali būti panaudota Merge Node arba Join Node apjungimas. Kadangi abu įeinantis srautai yra vienodi jų apjungimui panaudojame Merge Node notaciją.

5 žingsnis

Įvykiams, neturintiems išeinančio srauto priskiriame tuščia informacinį srautą, kurį nukreipiame į Final Node simbolį.

Sugeneruotas rezultatas



25 pav. Sugeneruota pavyzdine veiklos (activity) diagrama

9. Išvados

Darbe analizuojama UML 2.0 Activity diagramos generavimo iš veiklos modelio galimybė. Tai vaizduojama 7 paveiksle, kuriame Veiklos modelio elementai atvaizduojami į UML 2.0 *Activity* diagramos elementus. Kadangi kiekvienos UML 2.0 *Activity* diagramos klasės objektas gali būti atvaizduotas iš veiklos modelio, tai reiškia, jog pasirinktojo veiklos modelio sudėtis yra pakankama šios diagramos generavimui. Atliktas UML ir BPM grafinių notacijų analizė patikslino duomenis gautus sulyginus oraganizacijos ir veiklos (activity) diagramos metamodelius, rezultatas parodė, kad iš pateikto eksperimentos veiklos modelio klasių diagramos galima generuoti tik pagrindinius UML veiklos (activity) diagarmos elementus. Kitų elementų sugeneruoti neleidžia BPM notacijos ribotumas arba veiklos modelio klasių diagramos neišbaigtumas. Su tikslu, kad iš veiklos modelio būtų galima generuoti ne tik esminius veiklos (activity) diagramos elementus, veiklos modelis papildomas atitinkamomis klasėmis.

LITERATŪRA

[1] 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

[2] Gudas S., Lopata A. Žiniomis grindžiama informacijos sistemų inžinerija. // Informacijos mokslai T30, Vilnius: Vilniaus universiteto leidykla, 2004 , p. 90- 98. ISSN 1392- 0561., 2004

[3] Konferencija „informacinės technologijos ir valdymas 2000“ [žiūrėta 2007.01.02] preiga per Internetą: http://209.85.129.104/search?q=cache:eFlWf-FS71YJ:search.delfi.lt/cache.php%3Fid%3D7E1134413043DED9%26m%3Dtxt+40003+%22CEN+ENV+%22&hl=lt&gl=lt&ct=clnk&cd=7

[4] Konferencija „Informacinės technologijos ir valdymas 2002“ [žiūrėta 2007.05.02] preiga per Internetą: http://www.ktu.lt/lt/apie_renginius/konferencijos/2006/k6_02/IT2002/XI_sekcija.pdf

[5] S. Gudas, A. Lopata (2001). Informacijos išteklių identifikavimas veiklos modelio pagrindu [interaktyvus].

[6] A. Lopata. VEIKLOS MODELIŲ SUDĖTIES ANALIZĖ. Informacinės technologijos ir valdymas, 2000.

[7] Unified Modeling Language: Superstructure, version 2.0. [žiūrėta 2007.05.20.] Prieiga per Internetą: http://www.omg.org/docs/formal/05-07-04.pdf

[8] Gudas S., Lopata A. Vartotojo poreikių modelio generavimas veiklos modelio pagrindu. // Informacijos mokslai 2003 , p. 134- 140. ISSN 1392- 0561.

[9] Gudas S., Lopata A. Vartotojo reikalavimų modelio sudarymas žinių saugyklos pagrindu. // Informacijos mokslai 2006 , p. 127- 137. ISSN 1392- 0561.

[10] Gudas S., Skersys T. Klasių modelio generavimas veiklos modelio pagrindu. // Informacijos mokslai 2003 , p. 199- 205. ISSN 1392- 0561.

[11] S. Gudas, A. Lopata, Workflow models based acquisition of enterprise knowledge, INFORMATION TECHNOLOGY AND CONTROL, 2007, Vol.36, No.1A, [žiūrėta 01.10] prieiga per internetą http://itc.ktu.lt/itc361/Gudas361.pdf

[12] 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 01.10] prieiga per internetą http://www-users.cs.york.ac.uk/~kimble/research/UML_Profile_for_Modelling_KBS.pdf

[13] M. RAFFAI, Enterprise application integration, a metamodel approach, [žiūrėta 01.10] prieiga per internetą http://www.sea.uni-linz.ac.at/idimt2007/SessionF.pdf

[14] ACTIVITY DIAGRAM SPECIFICATIONS, [žiūrėta 01.10] prieiga per internetą http://gforge.enseeiht.fr/docman/view.php/34/407/TPC-SPECED-ACTIVITY.pdf
[15] S. A. White, IBM Corporation, Introduction to BPMN, [žiūrėta 01.10] prieiga per internetą http://www.bpmn.org/Documents/Introduction%20to%20BPMN.pdf
[16] S. A. White, IBM Corporation, Process Modeling Notation and workflow paterns“ [žiūrėta 01.10] prieiga per internetą http://www.bpmn.org/Documents/Notations%20and%20Workflow%20Patterns.pdf
[17] Business Process Modeling Notation (BPMN), [žiūrėta 01.10] prieiga per internetą http://www.workflownp.org.uk/fileupload/upload/BPMN-V1.01882004261512.pdf

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 **Vykdytojas** 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

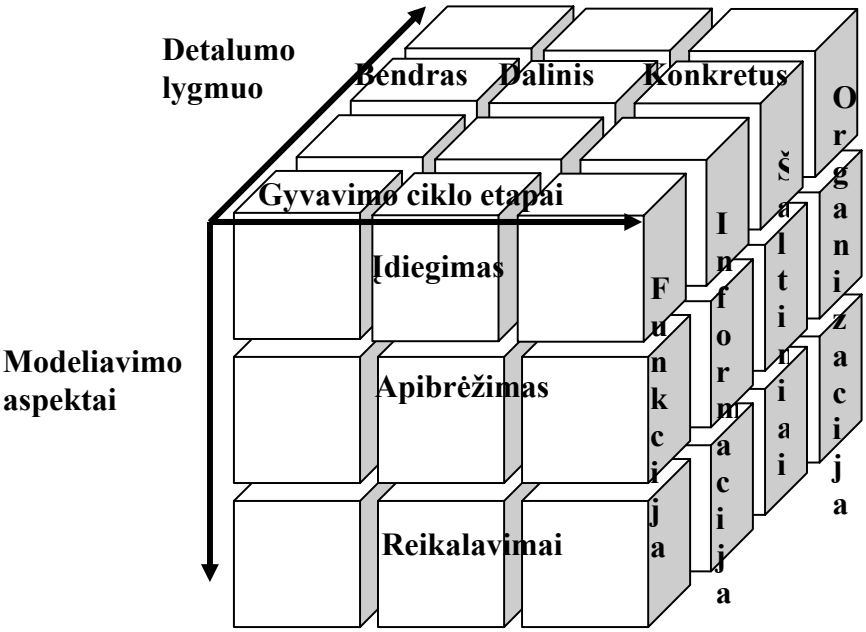
Realizacija		
Atributas	**Tipas**	**Atributų aprašymas**
PxFx	Integer	Procesų ir funkcijų sankirtos ID
Real_id	Integer	Realizacijos unikalus numeris (identifikatorius)
Real	Tekstas	Realizacija

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

CEN tarptautiniai standartai

Veiklos modelis yra IS inžinerijai reikalingų žinių sankaupa. Pasiūlyta veiklos modelio sudėtis grindžiama standartais CEN ENV 12204 , CEN ENV 40003 (dar vadinamu CIMOSA) ir UEML., kuriuos trumpai aptarsime.

CEN ENV 40003 ir CEN ENV 12204 standartus, kuriuose apibrėžti pagrindiniai organizacijos veiklos modeliavimo principai sukūrė Europos standartizacijos komitetas (CEN), bendradarbiaudamas su tarptautine standartizacijos organizacija (ISO). CEN ENV 40003 standartas yra sukurtas CIMOSA 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 (1 pav.)



1 pav. CIMOSA struktūra

CIMOSA pateikia nuoseklią organizacijos modeliavimo metodologiją. Tai į procesus orientuotas modeliavimo būdas, bendru būdu aprašantis visas organizacijos veiklas. Aprašomos veiklos apima gamybinius, valdymo ir administravimo procesus. Kaip matome 1 paveiksle CIMOSA palaiko organizacijos modeliavimo modeliais (duomenų procesais) paremta priartėjimą identifikuojant tris gyvavimo ciklo etapus, tris modeliavimo aspektus ir keturis detalumo lygius

Gyvavimo ciklo lygmuo:

Bendras lygmuo – nurodo į pagrindinių CIMOSA architektūrinių konstrukcijų arba komponentų, apribojimų, taisyklių, sąlygų blokus, funkcijas ir protokolus.

Dalinis lygmuo –dalinai konkrečiai organizacijos kategorijai pritaikomų modelių biblioteka.

Konkretus lygmuo – konkretaus organizacijos darinio modelis sudarytas iš blokų ir dalinių modelių.

Modeliavimo lygis:

CIMOSA pritaikymas organizacijai suskaidytas į tris dalis:

Reikalavimų specifikavimo lygis – organizacijos tikslų suskaidymas, siekiant apibrėžti verslo reikalavimus.

Projekto specifikavimo lygmuo – įvertinami alternatyvūs techniniai sprendimai siekiant pasirinkti geriausią ir specifikuoti optimizuotą į sistemą orientuotą verslo reikalavimų sąrašą.

Įdiegimo aprašymo lygmuo- įdiegti užbaigta CIM sistemą ir visų jos komponentų vaizdavimą, objektus, procesus, veiklas, resursus ir organizacines organizacijos dalis.

Datalumo lygmuo:

Funkcijos – aprašo darbų seką organizacijos funkcijų.

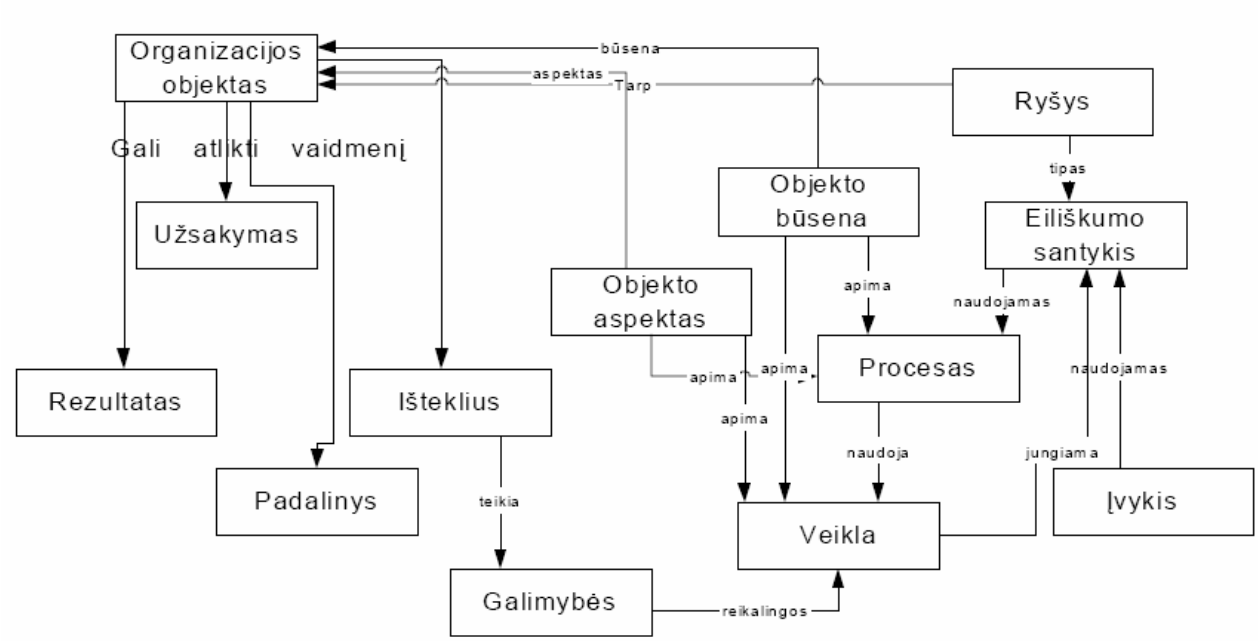
Informacija – aprašo organizacijos funkcijų įėjimus ir išėjimus ir integruotus organizacijos informacinius objektus.

Šaltiniai – aprašo informacijos šaltinius (žmones, mašinas, duomenis pateikiančias programas).

Organizacija – apibrėžia teises ir apribojimus funkcijoms, informacijai ir šaltiniams.

CIMOSA organizacijos veiklos modelio vaizdas atsispindi CIMOSA integruotos infrastruktūros servisuose.

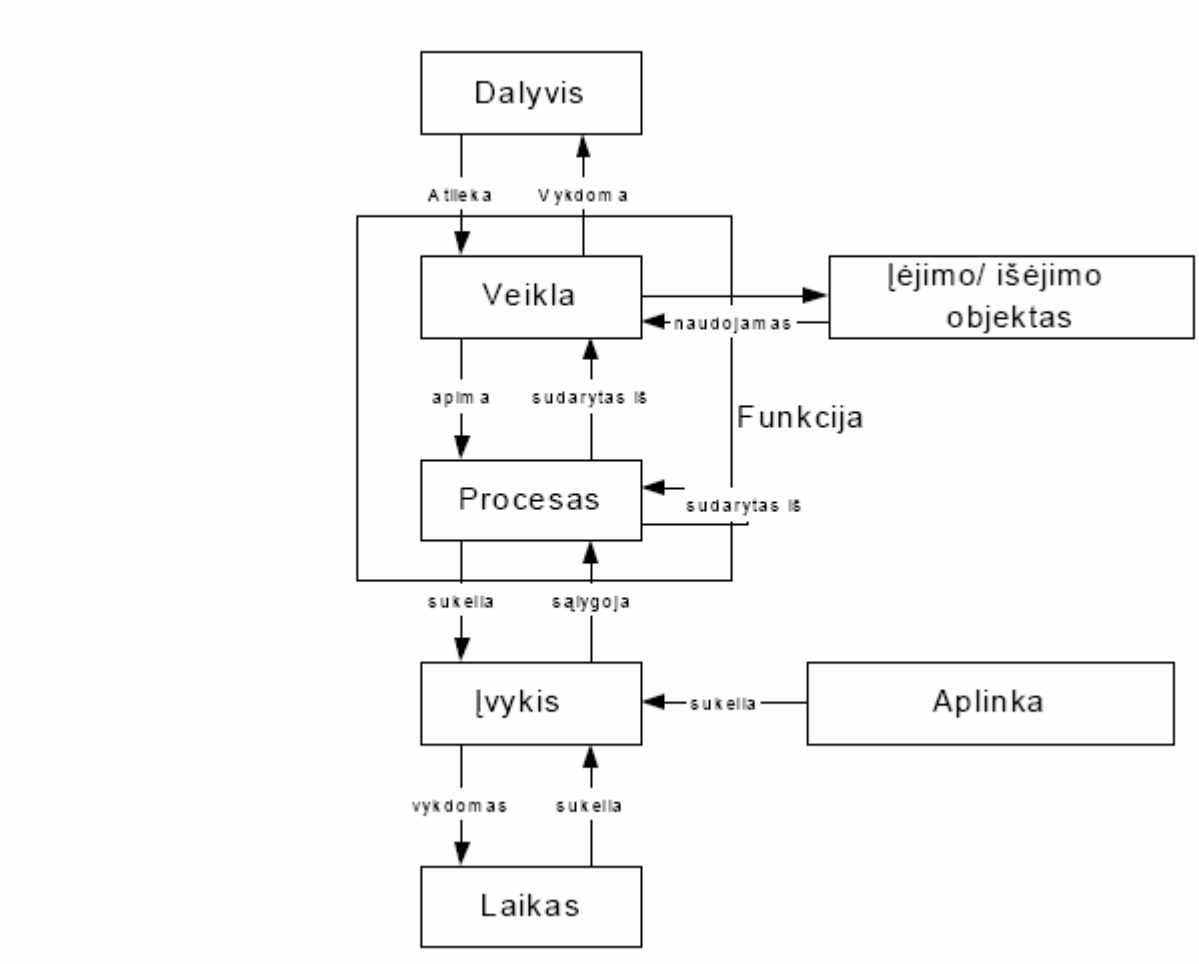
CEN ENV 40003 standarte veiklos modeliavimui būtinas sudėtines dalis (konstruktus) apibrėžia CEN ENV 12204 standartas:



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

ENV 12204 (Advanced Manufacturing Technology - Systems Architecture - Constructs for Enterprise Modelling) standartas apibrėžia 13 modeliavimo kalbų konstrukcijų, kurios yra susijusios su blokais, panaudojamais organizacijos modelio sudarymui, palaiko informacijos įvedimą iš CIMOSA, IEM ir QCIM. Kiekvienas konstrukcija aprašoma panaudojant bendrą apibrėžimo šabloną, aprašymo, antraštės ir struktūros. Ryšiai tarp konstrukcijų (statiniai ir dinaminiai) įtraukiami į aprašymą.

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 , bei kalbos, tokios kaip UEML (Unified Enterprise Modeling Language).



3 pav. UEML principinė schema