Fiziniai mokslai (I310), Technologijos mokslai (E160)

Vadyba
Journal of Management
2011, Nr. 1(18)
ISSN 1648-7974

STUDIJŲ PROGRAMŲ REIKALAVIMŲ INŽINERIJOS SISTEMA: NUO AUTOMATIZUOTO DOKUMENTŲ VALDYMO PRIE MODELIAIS GRINDŽIAMO KŪRIMO PROCESO

Jurij Tekutov[1, 2, 3, 4], Saulius Gudas[1], Vitalijus Denisovas[2, 4], Julija Tekutova[3]
[1]Vilniaus universitetas, [2]Klaipėdos universitetas, [3]Klaipėdos valstybinė kolegija, [4]Vakarų Lietuvos verslo kolegija

Anotacija
Šiame straipsnyje analizuojamas sistemų modeliavimo SysML kalbos standartas, išskiriami skirtumai nuo vieningos objektinės modeliavimo UML kalbos ir parenkamas jas įgyvendinančių CASE priemonių rinkinys, suteikiantis galimybę pereiti nuo į dokumentacijos tvarkymą orientuoto prie modeliais grindžiamo kūrimo proceso (MDA). UML buvo sukurta kaip modeliavimo kalba programinės įrangos kūrimo srityje, tai tarptautinės standartizacijos organizacijos (ISO) visuotinai pripažintas standartas, kurį specifikavo Objektų valdymo grupė (OMG); vėliau atsirado SysML specifikacijos. UML ir SysML kalbų naudojimas sistemų specifikacijai bei projektavimui suteikia daug privalumų, nes tokiu būdu užtikrinama galimybė vertinti ir validuoti sudarytus sistemų modelius, kas palengvina informacijos perdavimą kitiems inžinerijos disciplinų specialistams. Straipsnyje parodytas studijų programų kūrimas kaip modeliais grindžiamas procesas, paremtas SysML ir UML, kuris įgyvendinamas modeliais grindžiama sistemų inžinerijos (MDSE) metodika. Taip pat pateikti kelių instrumentinių programinės įrangos sukurtų CASE priemonių integracijos pavyzdžiai, leidžiantys atvaizduoti skirtingus studijų programų reikalavimų inžinerijos (valdymo) sistemos aspektus: studijų programų panaudojimo atvejų modelis padeda suprasti sistemos veikimo principą; klasių diagrama naudojama studijų programų struktūrai sudaryti (dalykų sudedamųjų dalių ir jų studijų rezultatų analizei); studijų programų reikalavimų išgavimo ir analizės veiklos bei sandaros nustatymo būsenų diagramos iliustruoja sistemos elgsenos (dinaminį) modeliavimą; sistemos architektūros modelis vaizduoja kompiuterizuojamos srities semantiką ir pan. Taip pat pateikta tolesnė minėtos sistemos plėtra, kuri siejama su žiniomis grindžiamų principų taikymu, tokiu būdu studijų proceso ir turinio formavimas vyksta probleminės srities (žinių) modelio pagrindu.
PAGRINDINIAI ŽODŽIAI: mokslas, universitetai, studijos, informatika, programinė įranga, modeliavimas, žinios.

Įvadas

Ankstesniuose darbuose autoriai suprojektavo ir įgyvendino automatizuotą informatikos studijų programų reikalavimų inžinerijos sistemą, grindžiamą reikalavimų dokumentacijos tvarkymo valdymu (reikalavimai saugomi sistemos duomenų bazėje). Šių darbų ciklas remiasi sisteminiu požiūriu ir pagrindinis dėmesys skiriamas studijų programų reikalavimų valdymo sistemai, kuri įgyvendinta CASE programinėmis priemonėmis (pasirinktas *IBM Rational RequisitePro*^{TM} įrankis). Minėta sistema aprobuota ir buvo pritaikyta kelioms studijų programoms kurti bei modernizuoti: pradedant neuniversitetine informatikos studijų programa ir baigiant pagrindinių (bakalauro) bei magistrantūros studijų programomis (Gudas, *et al.* 2009; Denisovas, Gudas, Tekutov 2010). Detalų atskirų reikalavimų inžinerijos sistemos taikymo metodikos etapų ir konkrečių studijų programų nagrinėjimą galima rasti kituose autorių straipsniuose (Denisovas, *et al.* 2009; Gudas, *et al.* 2009).

Kuriant aukščiau minėtą informatikos studijų programų reikalavimų inžinerijos sistemą, jau buvo taikomas struktūrinis-funkcinis metodas: reikalavimams modeliuoti pasirinkta duomenų srautų diagramų DFD (angl. *Data Flow Diagram*) notacija, studijų programų reikalavimų valdymo sistemos funkcijų nustatymui sudaryti IDEF0 (angl. *Integration of computer aided manufacturing DEFinition*) standarto modeliai ir kt. (Denisovas, Gudas, Tekutov ,2010). Tačiau reikėtų pabrėžti, kad informacijos sistemų inžinerijoje naudojami tradiciniai veiklos modeliavimo metodai (IDEF, DFD) neapima svarbių socialinių ir technologinių organizacijos veiklos aspektų, tokių kaip organizacijos strategija, ir jos sąveikos su struktūra, veiklos dalyviais, organizacijos infrastruktūra.

Unifikuota modeliavimo (UML) ir sistemų modeliavimo (SysML) kalbos nuosekliai įgyvendina modeliais grindžiamą sistemų inžinerijos (angl. *Model Based Systems Engrineering* – MBSE) metodiką, leidžiančią kurti modelius visiems kūrimo proceso lygmenims, t. y. operacinius, sistemos ir atskirų komponentų modelius (Hause, 2006). Šiandien objektiniais modeliais paremta MBSE metodika taikoma kaip bendra filosofija organizacijų informacinių sistemų kūrimui, programų ir tinklų inžinerijai, veiklos procesų inžinerijai ir pertvarkymui bei žinių bazėms (Nemuraitė, 2008). Galima išskirti tokio naudojimo privalumus: vizualizavimas (grafinio modelio taikymas), komunikavimo priemonė, galimybė operuoti universaliomis modeliavimo sąvokomis atliekant sistemų projektavimą ir reinžineriją.

Dauguma CASE (angl. *Computer-Aided Systems Engineering*) priemonių, įgyvendinančių MBSE metodikos palaikymą, orientuotos į objektines modeliavimo kalbas. Šiandien vizualaus modeliavimo

kalbos ir jas įgyvendinančios CASE priemonės suteikia programų ir sistemų inžinieriams galimybę pereiti nuo į dokumentacijos tvarkymą orientuoto prie modeliais grindžiamo kūrimo proceso (angl. *Model Driven Architecture* – MDA) (Frankel, 2003; Kleppe, Warmer, Bast, 2003). Moderniausios priemonės leidžia sistemų inžinieriui ne tik aprašyti sistemoje vykstančius procesus dalykinės srities terminais, t. y. pakankamai abstrakčiais SysML modeliais, bet ir sugeneruoti modelį atitinkantį kodą bei atlikti modelio imitavimą (skaičiavimą) (France, Rumpe, 2007; Peak, *et al.* 2007). Tokiu atveju galima kalbėti jau apie vykdomaisiais modeliais grindžiamą kūrimo procesą (angl. *Excecutable Model Driven Architecture* – EMDA) (Johnson, *et al.* 2007). Tačiau dar nėra sukaupta pakankamos SysML taikymo įvairiose dalykinėse srityse patirties, o egzistuojančios CASE priemonės tik dalinai įgyvendina MDA proceso etapus ir dažnai nėra suderinti tarpusavyje.

Tyrimo objektas – studijų programų kūrimo ir atnaujinimo reikalavimų inžinerijos (valdymo) sistema. Straipsnio tikslas – toliau vystant autorių sukurtą studijų programų reikalavimų inžinerijos sistemą, aprašyti ją kaip modeliais ir žiniomis grindžiamą veiklą. Tokiu būdu studijų programų kūrimo procesas tampa daugiau formalizuotas, o sukurti modeliai sudaro prielaidą pereiti prie žiniomis grindžiamų principų.

Straipsnio struktūra tokia: pirmoje dalyje aptariamos modeliais grindžiamos sistemų inžinerijos metodikos įgyvendinančios unifikuotos modeliavimo (UML) ir sistemų modeliavimo (SysML) kalbos; antroje dalyje parodytas studijų programų kūrimas kaip modeliais grindžiamas procesas. Trečioje dalyje pateikiami suformuluoti žiniomis grindžiami studijų programų reikalavimų inžinerijos sistemos principai. Pabaigoje pristatomi ir aptariami pagrindiniai rezultatai bei pateikiamos išvados.

Modeliais grindžiamos sistemų inžinerijos metodikos įgyvendinančių priemonių parinkimas: UML ir SysML kalbų palyginimas

Unifikuota modeliavimo UML (angl. *Unified Modeling Language*) ir sistemų modeliavimo SysML (angl. *System Modeling Language*) kalbos yra vizualaus modeliavimo standartai, prižiūrimi „Objektų valdymo grupės“ (OMG) konsorciumu. SysML buvo sukurta bendradarbiaujant OMG (angl. *Object Management Group*) ir INCOSE (angl. *The International Council on Systems Engineering*), kurios pagrindas buvo UML 2.0. SysML skirta specifikuoti, analizuoti, projektuoti, vertinti ir validuoti sistemas, kurios jungia savyje techninę ir programinę įrangą, duomenis, žmonės (personalą), kūrimo procedūras bei priemones (OMG konsorciumas, 2010). UML ir SysML komponentai:

- susideda iš įvairių grafinių elementų, apjungiamų į diagramas;
- egzistuoja griežtos grafinių elementų panaudojimo taisyklės;
- diagramų tikslas – įvairiais pjūviais aprašyti kuriamą sistemą norimu detalumu, t. y. sudaryti sistemos modelį.

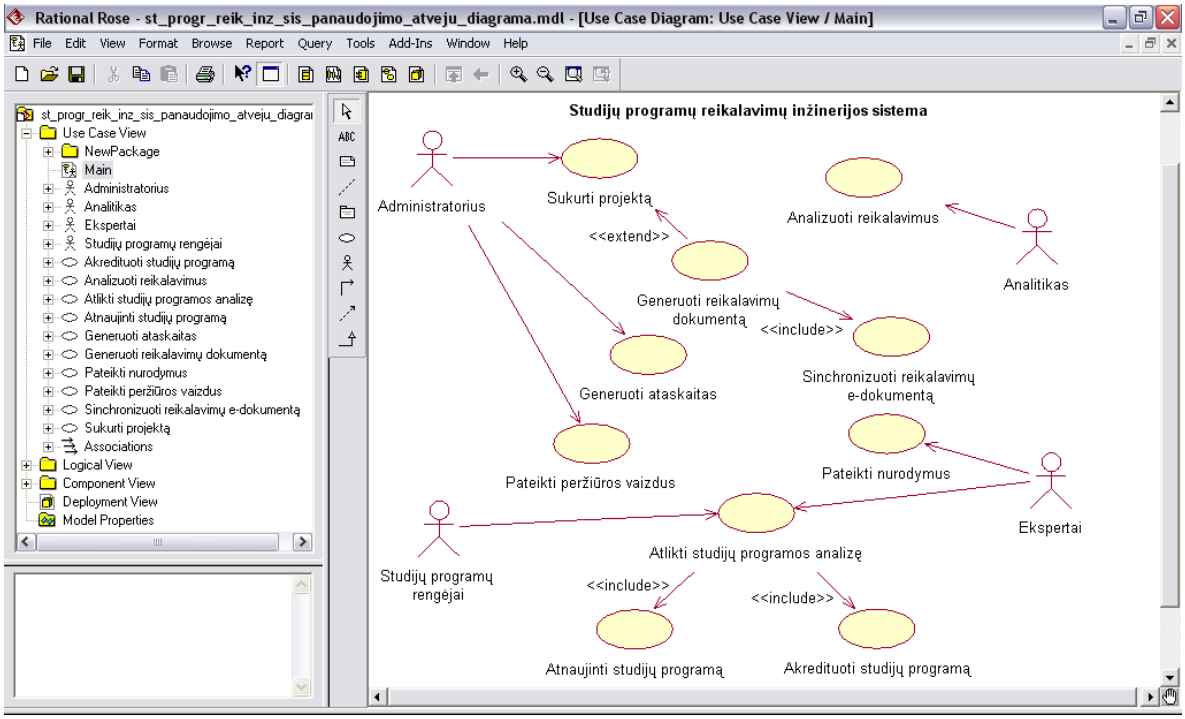
SysML naudoja pagrindinius UML diagramų tipus, tokius kaip panaudojimo atvejai, sekų diagrama, būsenų diagrama, paketų diagrama, o tuo tarpu kitos diagramos modifikuotos, kad atitiktų SysML praplėtimą. SysML‘e yra trys diagramos, skirtos aprašyti struktūrą. Blokų apibrėžimų diagrama panaši kaip esybių-ryšių diagrama, atvaizduoja vienų blokų ryšius su kitais (Рыжов, Иванов, 2009). Vidinės struktūros diagrama skirta detaliau atvaizduoti vidinių blokų kilmę ir ryšius su kitais blokais (pvz., atvaizduoti informacijos srautus tarp blokų). Parametrinė diagrama skirta išgauti lygtis, naudojamas apskaičiuoti tam tikrus bloko parametrus ar atributus (Weilkiens, 2007). Elgsena aprašoma kitomis trimis diagramomis. Sekų ir būsenų diagramos paliktos nemodifikuotos, tokios pačios kaip UML. Veiklos diagrama yra modifikuota UML diagrama. SysML naudoja ne visas UML diagramas, nėra tokių diagramų kaip objektų, komunikavimo, sąveikų peržiūros, laiko skaičiavimo ir išdėstymo. Integruotų sistemų analizei atlikti ir specifikacijoms sudaryti įvesta nauja reikalavimų diagrama (Johnson, *et al.* 2007).

Taigi kiekviena diagrama – požiūris į dalykinę sritį iš tam tikro taško, pabrėžiant tuo momentu svarbiausias išskiriamos analizės srities elementus, juos paverčiant sistemos komponentais ir sujungiant būdingais ryšiais.

Studijų programų kūrimas kaip modeliais grindžiamas procesas

Studijų programų panaudojimo atvejų modelis. Studijų programų reikalavimų inžinerijos sistemos modeliavimas pradedamas nuo panaudojimo atvejų grafinio modelio (angl. *Use Case Model*) sudarymo, aprašančio, kokias funkcijas sistema atlieka išorinio stebėtojo akimis. Šis modelis naudojamas identifikuoti pirminius sistemos elementus ir procesus (Eriksson, *et al.,* 2004), kurie, mūsų atveju, formuoja studijų programą. Pirminiai elementai traktuojami kaip „aktoriai“, o procesai kaip „panaudojimo atvejai“. Tarp skirtingų „panaudojimo atvejų“ gali būti nurodytos dviejų tipų sąsajos: naudoja (angl. *include*), jei vienas veiklos procesas naudoja kito suformuotus rezultatus; išplečia (angl. *extend*), jei vienas veiklos procesas yra kito sudėtyje. Rėmelis (angl. *Boundary*) rodo sistemos ribas. SysML šios diagramos nemodifikavo ir naudoja standartinę UML diagramą.

Sudarytas studijų programų panaudojimo atvejų modelis pateikiamas 1 paveiksle.



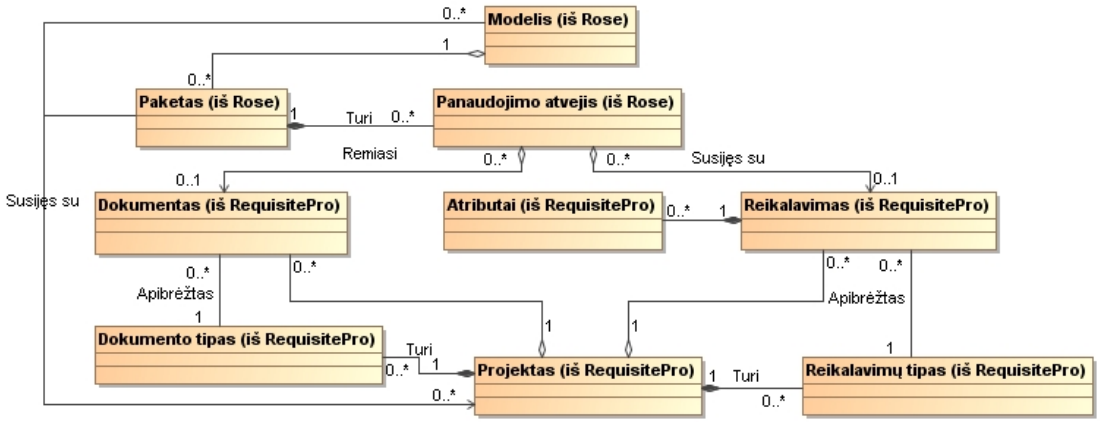
1 pav. Studijų programų panaudojimo atvejų modelis
(sukurtas *IBM Rational Rose Enterprise*^{TM} priemonės aplinkoje)

Reikalavimams specifikuoti esamoje sistemoje taikoma *IBM Rational RequisitePro*^{TM} aplinka (Denisovas, *et al.* 2009), o pereinant prie modeliais grindžiamo kūrimo (Hay 2003), siekiama aptarti panaudojimo atvejų modelio generavimo galimybę *IBM Rational Rose Enterprise*^{TM} priemonėje iš reikalavimų specifikacijos dokumento (*IBM Rational RequisitePro*^{TM} priemonės importuoto projekto).

CASE priemonių integracija įgyvendinant modeliais grindžiamą kūrimą. *IBM Rational Rose Enterprise*^{TM} – CASE priemonė, skirta vizualiam modeliavimui taikant UML kalbą.

Panaudojimo atvejų atvaizde grafiškai modeliuojami projektuojamos sistemos vartotojų (aktorių) ir taikymo (panaudojimo) atvejų santykiai (Gomaa, Olimpiew, 2005). *IBM Rational RequisitePro*^{TM} priemonės pagrindu sukurti reikalavimai su visais savo atributais gali būti tiesiogiai susieti su *IBM Rational Rose Enterprise*^{TM} priemonės panaudojimo atvejų modelio konkrečiu egzemplioriumi (žr. 1 pav.). Taip pat kiekvieną panaudojimo atvejį galima susieti su *IBM Rational RequisitePro*^{TM} dokumentu, kuriame gali būti aprašomi visi jam keliami reikalavimai, jų atributai ir kita papildoma informacija (IBM kompanijos Academic Initiative, 2009).

Pateikiamoje 2 paveiksle klasių diagramoje pavaizduotas *IBM Rational Rose Enterprise*^{TM} ir *IBM Rational RequisitePro*^{TM} CASE priemonių tarpusavio susietumas (programinių paketų komponentų sąsajos).



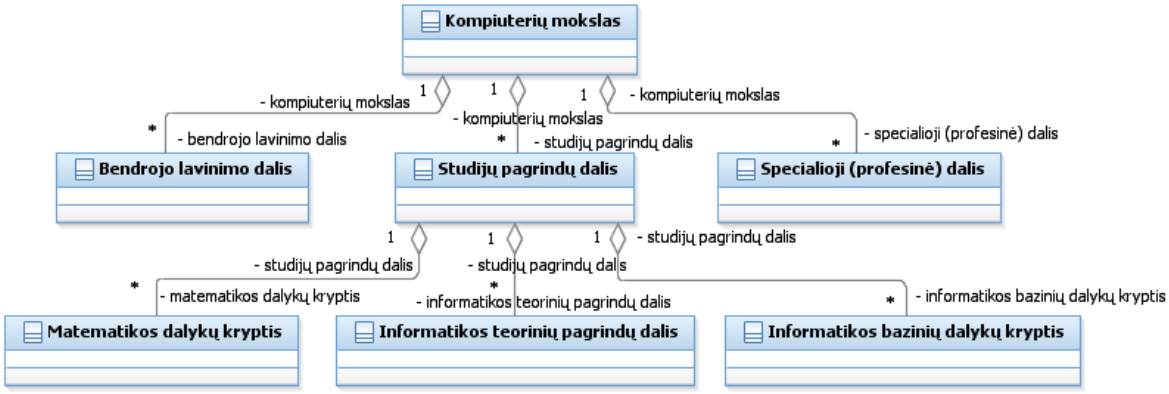
2 pav. *IBM Rational Rose Enterprise*^{TM} ir *IBM Rational RequisitePro*^{TM} CASE priemonių integracijos klasių diagrama (metamodelis)

Analogišku būdu gali būti atliktas integravimas su kitomis CASE priemonėmis, kurios palaiko SysML. Šiame darbe pasirinkta *IBM Rational Software Architect* ™ – integruota architektūros ir kūrimo

priemonė, kuri remiasi modeliais grindžiamu kūrimu, naudojanti UML ar SysML kalbas kuriant aplikacijas (Кватрани, Палистранта 2007; Новичков, Карабанова, 2010; Смит, 2009). Taip pat ši priemonė palaiko integraciją su kitais produktais, tokiais kaip *WebSphere Business Modeler™*, *IBM Rational Clear Case™*, *IBM Rational ClearQuest™* ir *IBM Rational RequisitePro™* (IBM kompanija, 2009).

Detalių studijų programos architektūros komponentų ir studijų rezultatų klasių modelių kūrimas. Bazinis UML statinis modelis – klasių diagrama (angl. *Class Diagram*) naudojama studijų programų struktūrai sudaryti: dalykų (blokų) ir jų studijų rezultatų analizei. Šioje diagramoje stebima agregacija tarp klasių, iš kurių viena ar kelios yra kitos klasės sudedamosios dalys (Eriksson, *et al.* 2004; Sommerville, 2006).

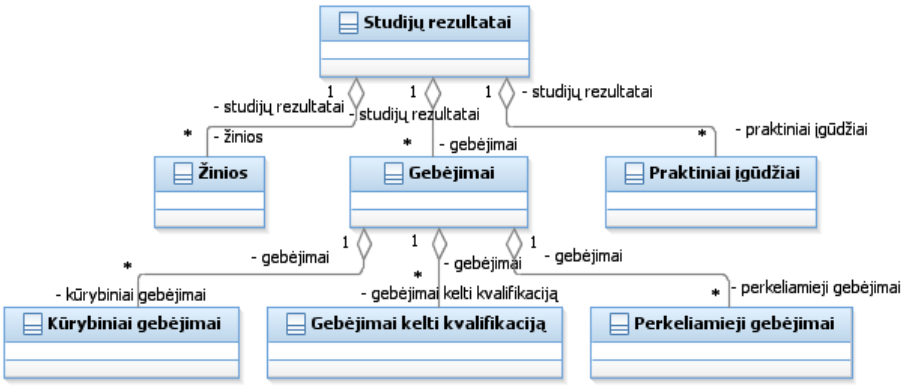
Detalių studijų programos architektūros komponentų klasių diagrama pavaizduota 3 paveiksle.



3 pav. Detalių studijų programos architektūros komponentų klasių diagrama (*IBM Rational Software Architect™* priemonės aplinkoje)

Pagal *Informatikos studijų krypties reglamentą* (Lietuvos Respublikos švietimo ir mokslo ministerija, 2007) studijų programą sudaro trys tikslinės dalys: bendrojo lavinimo dalis, studijų pagrindų dalis (studijų programoje sudaro studijų branduolį) ir specialioji (profesinė) dalis, teikianti gilesnes žinias bei gebėjimus, orientuotus į tolesnę profesinę ar tiriamąją veiklą. Kuriant ar atnaujinant informatikos studijų programas šios dalys turi būti detalizuojamos studijų krypčių (sub-grupių) ir modulių (dalykų) lygmenyse.

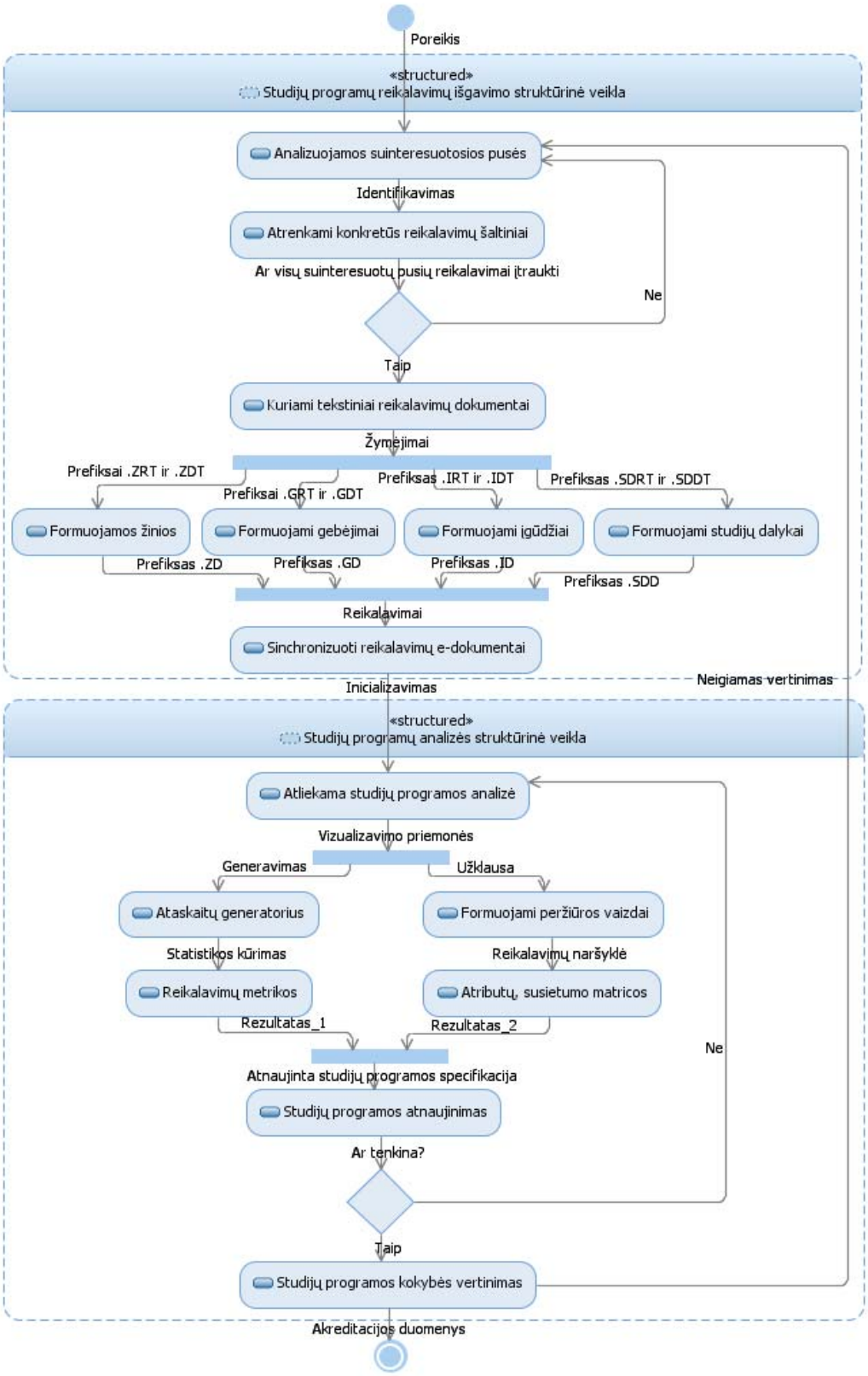
Studijų rezultatai (mokymosi pasiekimai) išreiškiami kaip žinios, gebėjimai ir įgūdžiai. Gebėjimai skirstomi į pažintinius, perkeliamuosius ir praktinius. Profesinė veikla reikalauja atitinkamos kvalifikacijos, susidedančios iš atskirų kompetencijų (pvz., gebėjimo savarankiškai, kokybiškai ir kūrybiškai, t. y. kompetentingai veikti tam tikroje srityje ar profesijoje). Studijų rezultatų klasių diagrama parodyta 4 paveiksle.



4 pav. Studijų rezultatų (mokymosi pasiekimų) klasių diagrama

Taigi kvalifikacija integruotai išreiškia žmogaus žinias, mokėjimus, įgūdžius, nuostatas, kurie dažniausiai įgyjami tam tikru išmokimo būdu. Kita vertus, kvalifikacija parodo „žmogaus tinkamumo tam tikram darbui laipsnį“ (Laužackas, 2008), tai, kad tos įgytos vertybės yra skirtos atlikti tam tikrą veiklą. Jeigu kvalifikacija apibūdina tam tikrai profesijai reikalingos žinios ir gebėjimai (mokėjimai ir įgūdžiai), tuomet yra aišku, kad jų apimtis tam tikros veiklos pradžioje ir po tam tikro laiko gerokai skiriasi. Taigi reali kvalifikacija kinta ir jos kėlimas dažniausiai susijęs ne tiek su naujomis profesinėmis žiniomis, kiek su jų naudojimu atliekant praktinę veiklą ir atsirandančiomis sudėtingesnėmis charakteristikomis.

Studijų programų reikalavimų išgavimo ir analizės veiklos diagrama. Panaudojimo atvejų ir klasių diagramos nerodo veiklos procesų eigos. Todėl veiklos procesams vaizduoti naudojamos veiklos arba scenarijų diagramos (angl. *Activity Diagrams*), skirtos verslo procesų aprašymui (Bock, 2005; Eriksson, *et al.* 2004). Veiklos diagramų elementai yra grafo viršūnės, kurios vaizduoja veiklas (angl. *activities*), ir briaunos (angl. *edges*), kurios vaizduoja valdymo (angl. *control flows*) ar objektų srautus (angl. *object flows*). Veikla reiškia veiksmo vykdymą. Veiksmui pasibaigus, įvyksta perėjimas prie kito veiksmo. Sudaryta studijų programų reikalavimų išgavimo ir analizės veiklos diagrama pavaizduota 5 paveiksle.



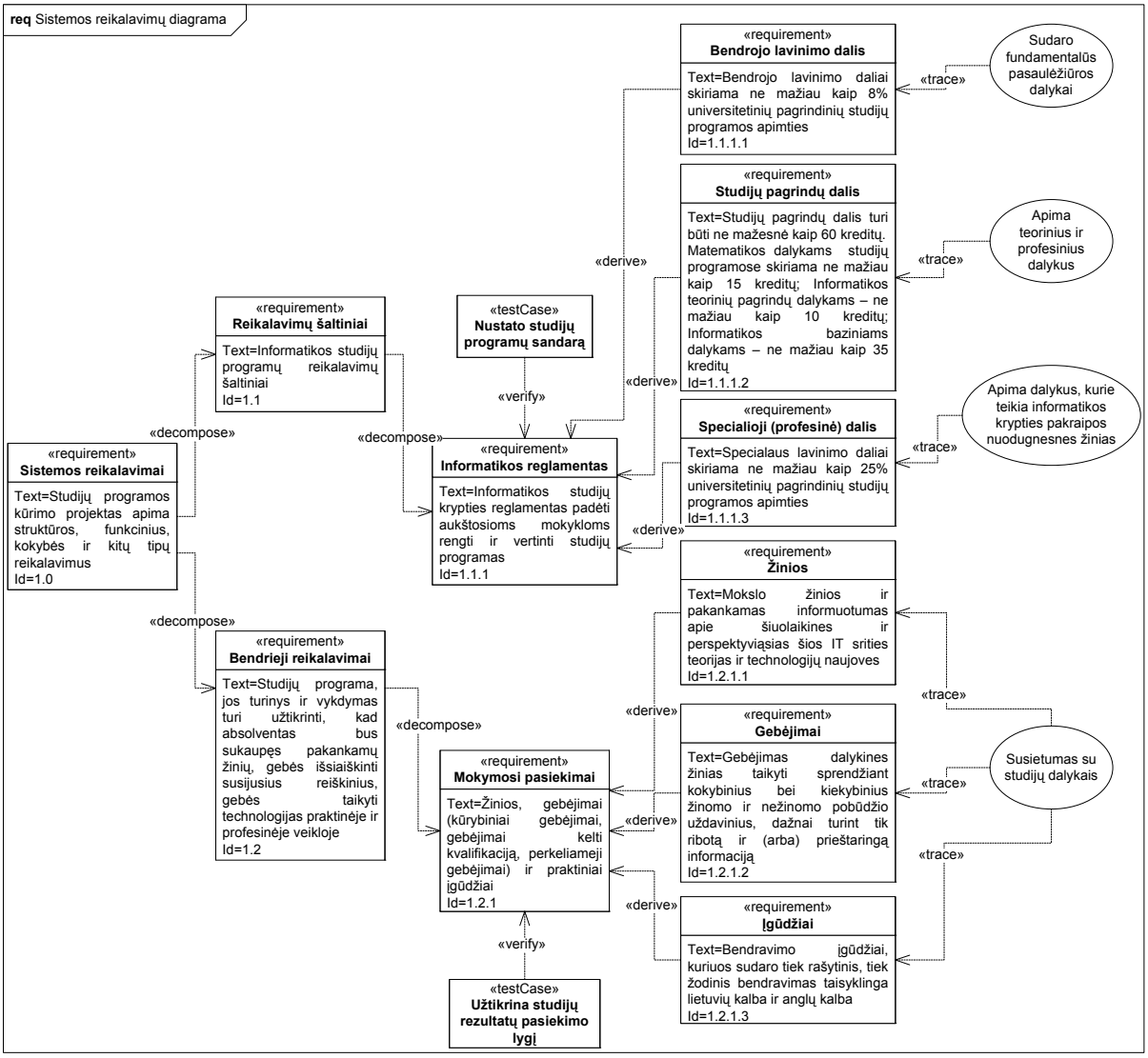
5 pav. Studijų programų reikalavimų išgavimo ir analizės veiklos diagrama

Kaip matome pavaizduotos studijų programų reikalavimų išgavimo ir analizės veiklos diagramos (žr. 5 pav.), suliejimas (angl. *merge*) reiškia paprastą anksčiau išsiskyrusių srautų susiliejimą (pvz., po „atrinktų konkrečių reikalavimų šaltinių“ veiksmo tikrinama, ar visų suinteresuotų pusių reikalavimai įtraukti); išsišakojimas (angl. *fork*) reiškia, kad iš jo išeinantys srautai gali būti vykdomi lygiagrečiai (pvz., priskiriami reikalavimų ir dokumentų tipai); sujungimas (angl. *join*) reiškia, kad išeinantis veiksmas prasideda tik tada, kai įvykdyti visi įeinančių srautų veiksmai (pvz., po visų dokumentų prefiksų suteikimo formuojami patys reikalavimai). Dviejuose suapvalintuose stačiakampiuose vykdoma struktūrinė veikla (angl. *Structured activity*) nuo pradžios (įėjimo duomenų gavimas) iki galo (gaunami išėjimo duomenys).

Studijų programų reikalavimų išgavimo ir analizės veiklos diagrama skirta atvaizduoti duomenų srautus ar kitaip tariant veiklas ir veiksmus (parodyti atskiri žingsniai nagrinėjamoje sistemoje). Tokiu būdu veiklos diagramoje (veiklų grafe) valdymo ir duomenų reikšmės sklinda grafo viršūnėmis, kurios juos apdoroja, perduoda kitoms viršūnėms arba laikinai saugo. Paminėtina, kad SysML veiklos nėra griežtai priskiriamos vienai klasei ar aspektui, viena veikla gali priklausyti kelioms (Bock, 2005).

Reikalavimų diagrama. Nuosekliai įgyvendinant mūsų pasirinkta būdą analizuojant bei modifikuojant studijų programas, reikalavimų išgavimas ir formulavimas yra lemiantys procesai reikalavimų inžinerijoje, todėl šiuo atveju prasminga pasinaudoti sistemų modeliavimo SysML kalbos reikalavimų diagrama (angl. *Requirements Diagram*), kurioje patys reikalavimai gali būti atvaizduoti grafiškai, lentelių pavidalu ar medžio struktūra (Peak, *et al,.* 2007).

SysML kalbos sistemos reikalavimų diagrama pavaizduota 6 paveiksle.



6 pav. SysML kalbos sistemos reikalavimų diagrama

Reikalavimų taksonomiją galima apibrėžti papildomomis reikalavimų stereotipų sub-klasėmis, leidžiančios apibrėžti modelio elementų tipus, kurie tenkintų tam tikrus reikalavimus (Robertson,

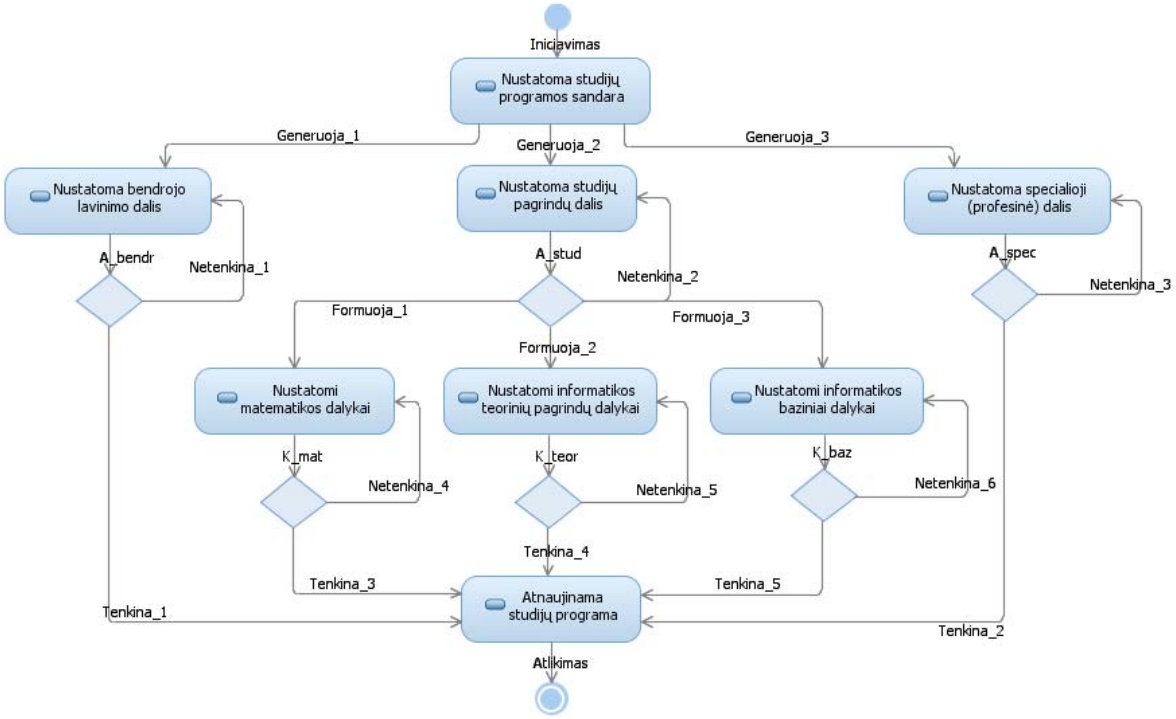
Robertson, 2006). Naudojami tokie stereotipai (Николаев, Зыль, 2006):

- skaidyti (angl. *decompose*) – ryšys tarp reikalavimų ir sub-reikalavimų;
- išvesti (angl. *derive*) – sąryšis tarp dviejų reikalavimų, rodantis, jog vienas iš jų išplaukia iš kito;
- tikrinti (angl. *verify*) – sąryšis tarp reikalavimo ir kontrolinio pavyzdžio (angl. *TestCase*), tikrinantis šio reikalavimo vykdymą;
- susekti (angl. *trace*) – naudojamas sistemos reikalavimų atitikimui ir susietumui.

Reikalavimai gali būti apjungti į blokus, tai padeda atvaizduoti sudėtingesnę sistemą (pvz., bendrojo lavinimo bloką), o taip pat leidžia vykdyti reikalavimų dekompoziciją, t. y. konkretizavimo ir detalizavimo procesus.

Tinkamam studijų programų kūrimo procesui užtikrinti reikalavimai gali pasirodyti kitose diagramose norint atvaizduoti ryšį su reikalavimų diagramoje esančiais elementais.

Studijų programų sandaros nustatymo būsenų diagrama. Laikoma, kad UML ir SysML stipriausia pusė yra struktūrinio sistemos aspekto modeliavimas, tačiau ne mažiau svarbus ir sistemos elgsenos modeliavimas. Kadangi sistemos elgsena ir būsenos, į kurias pereina sistema, priklauso nuo įeinančių duomenų srautų, tai reakcijos į įėjimus reikalavimai taip pat specifikuojami būsenų diagramoje (angl. *State Diagram*) (Кватрани, Палистранта, 2007). Sudaryta studijų programų sandaros nustatymo būsenų diagrama pateikiama 7 paveiksle.



7 pav. Studijų programų sandaros nustatymo būsenų diagrama

Svarbu pažymėti, kad ši diagrama sukurta klasių diagramoje identifikuotoms klasėms (objektams) (žr. 3 pav.), o patys reikalavimai gali būti išgauti iš reikalavimų diagramos (žr. 6 pav.). Atitinkamuose duomenų srautuose atvaizduoti parametrizuojami klasių diagramos atributai. Pvz., informatikos pagrindinių (bakalauro) studijų programoje: A_bendr = „ne mažiau kaip 8 % apimties“; A_stud = „ne mažiau kaip 60 kreditų“; A_spec = „ne mažiau kaip 25 % apimties“; K_mat = „ne mažiau kaip 15 kreditų“; K_teor = „ne mažiau kaip 10 kreditų“; K_baz = „ne mažiau kaip 35 kreditų“.

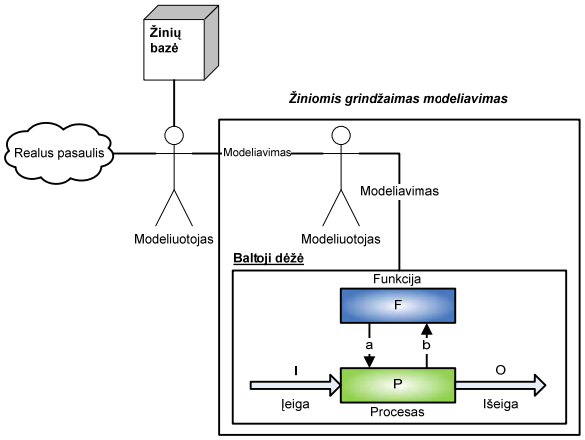
Kaip parodyta 7 paveiksle, studijų programų sandaros nustatymo diagramoje atvaizduotos būsenų sekos, sąlygos, prie kurių pereinama iš vienos būsenos į kitą, o taip pat veiksmai, kurie atliekami esant konkrečioje būsenoje arba perėjimo metu. Tokiu būdu galima atsižvelgti į dinamines (elgsenos) sistemos aspektus.

Žiniomis grindžiami studijų programų reikalavimų inžinerijos sistemos principai

Tolesnė studijų programų reikalavimų inžinerijos sistemos plėtra siejama su žiniomis grindžiamų principų taikymu. Vidinis modeliavimas yra *žiniomis grindžiamas modeliavimas*, kuris gali būti (Gudas, 1991):

- formalizuotas (matematinis) modeliavimas;
- notacija grindžiamas (inžinerinis) modeliavimas.

Žiniomis grindžiamas modeliavimas – tai vidinis modeliavimas, kai modeliuotojas naudojasi žinių baze, kurioje yra mokslinė informacija apie probleminės srities savybes (8 pav.).

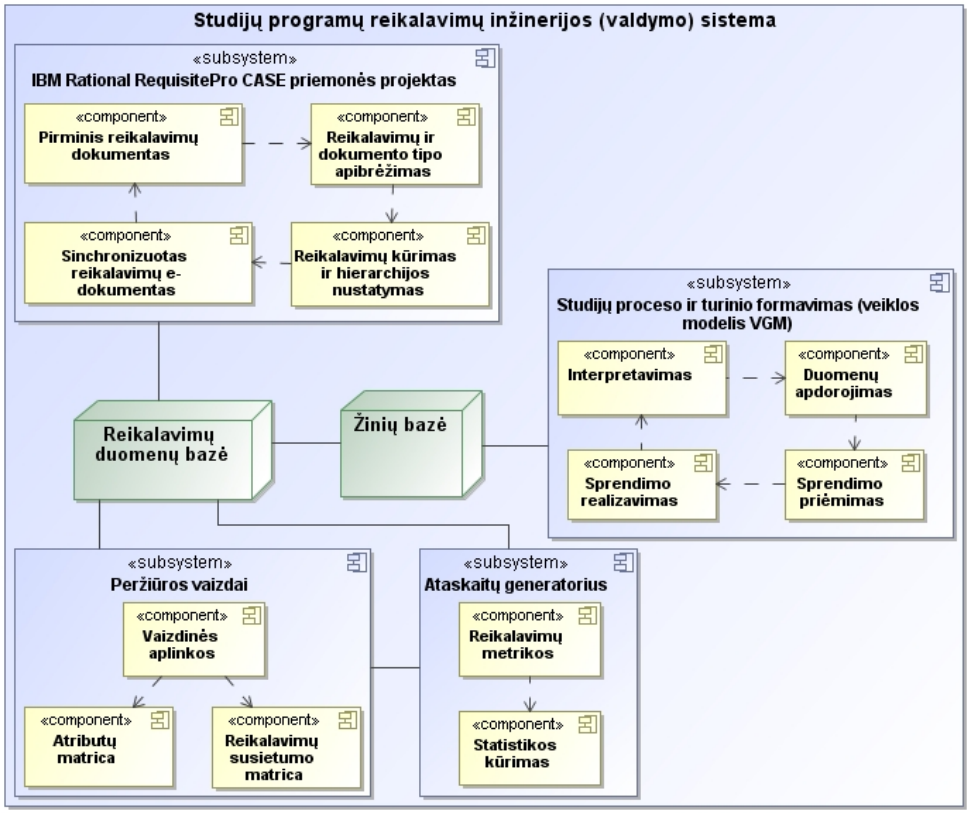


8 pav. Žiniomis grindžiamo modeliavimo principinė schema

Žiniomis grindžiamas modelis yra toks, kuris sudaromas pagal objektyvizuotą (mokslinę) informaciją – mokslo sukauptų žinių apie realybės dėsningumus pagrindu (t. y. vidinio stebėjimo pagrindu), naudojant pasirinktą modelio atvaizdavimo būdą (formalizuotą kalbą arba notaciją). Vidinio modeliavimo atveju analitikas identifikuoja priežastinius veiklos procesų ryšius, t. y. būtinas ir pakankamas veiklos elementų (materialių, informacinių procesų ir struktūrinių informacijos vienetų) dedamąsias ir jų sąveikas (kurios atitinka nagrinėjamos realybės srities dėsningumą).

Studijų programų reikalavimų inžinerijos sistemos architektūra. Architektūra yra fundamentali sistemos organizacija, kurią sudaro jos komponentai, jų ryšiai, aplinka ir sistemos projektavimo bei evoliucijos principai (IEEE 2000). Studijų programų reikalavimų inžinerijos sistemos architektūra yra fundamentali sandara sistemos struktūrai, apima sistemos padalinimą į bendraujančius posistemius. Tam naudojama komponentų (paketų) diagrama (angl. *Component Diagram*).

Pavaizduotas studijų programų reikalavimų inžinerijos sistemos architektūros modelis rodo pagrindinius sistemos komponentus, kaip posistemiai dalinasi duomenimis ir kokias turi tarpusavio sąsajas (9 pav.).



9 pav. Studijų programų reikalavimų inžinerijos sistemos architektūros modelis

Kiekvienas reikalavimų dokumentas priskiriamas vienam iš dokumentų tipų (Zielczynski, 2008). Sukūrus dokumentą, *IBM Rational RequisitePro*^{TM} dinamiškai susieja jį su duomenų baze (DB), kas suteikia galimybę valdyti DB koreguojant ir išsaugant dokumento turinį. Bendri duomenys yra saugomi centinėje reikalavimų DB arba saugykloje ir gali būti pasiekiami visiems posistemiams. Studijų proceso ir turinio formavimas vyksta probleminės srities (žinių) modelio pagrindu, pasirinktas *vertės grandinės modelis* (VGM) pagal M. Porter (Porter, 1998). VGM išreiškia procesinį požiūrį į veiklą. Žinių vadybos srityje taip pat ryškėja tendencija sieti žinių vadybos veiklą su procesiniu požiūriu, šioje srityje įsitvirtino netgi specialus terminas „į veiklos procesus orientuotas žinių valdymas“ (angl. *Business Process Oriented Knowledge Management*). Šio straipsnio autoriai mano, kad prasminga praktiniu veiklos srities modeliavimo metodu pasirinkti vertės grandinės modelį, taikomą žiniomis grįstai veiklai modeliuoti.

Toks modelis atvaizduoja esamu momentu funkcionuojančias veiklas (*funkcijas Fi* ir *procesus Pj*) bei ryšius tarp jų. Pasak S. Gudą (1991), formalizuotas *valdomo veiklos proceso* modelis – elementarus veiklos valdymo ciklas. Tokiu būdu *valdomą veiklos procesą* formalizuotai aprašo struktūra, vadinama *elementariu (veiklos) valdymo ciklu* (EVC) (Gudas, Brundzaitė, 2005). Kiekvienas *valdomas (veiklos) procesas* turi savo struktūrą ir priežastinę tvarką, kurios esmė – grįžtamojo ryšio kontūras, jungiantis į uždarą grandinę (ciklą) šias komponentes: *interpretavimo*, *duomenų apdorojimo*, *sprendimo priėmimo* ir *sprendimo realizavimo* procesus. Žinių bazė (angl. *Knowledge Base*) – žinių rinkinys, išreikštas naudojant tam tikrą formalią žinių vaizdavimo kalbą, žiniomis grindžiamos sistemos (angl. *Knowledge-based System*) dalis (Maskeliūnas, 2006). Žinių bazėje saugomos žinios galėtų būti panaudotos numatytų konkrečių taikomųjų sričių problemoms spręsti.

Išvados

Pateikta studijų programų reikalavimų inžinerijos (valdymo) sistemos evoliucija:

- susietas objektinio modeliavimo kalbų (UML 2.0 ir SysML) bei jas įgyvendinančių CASE programinių priemonių rinkinys suteikia studijų programų rengėjams galimybę pereiti nuo į dokumentacijos tvarkymą orientuoto prie modeliais grindžiamo studijų programų kūrimo proceso;
- galima verifikuoti bei validuoti sudarytus studijų programų reikalavimų inžinerijos sistemos statinius ir dinaminius modelius, o studijų programų kūrimas kaip modeliais grindžiamas procesas, atspindintis tiek duomenų, tiek elgsenos aspektus, padeda sklandžiai pereiti prie *žiniomis grindžiamo* studijų proceso valdymo modelių, leidžiančių valdyti studijų proceso formavimą probleminės srities modelio pagrindu (formalizuotai spręsti studijų programų kūrimo ir turinio tobulinimo problemas).

Taikant *žiniomis grindžiamus principus* galima numatyti autorių sukurtos studijų programų reikalavimų inžinerijos sistemos tolesnę plėtrą:

- praktiniu veiklos srities modeliavimo metodu pasirinkus struktūrizuotą vertės grandinės modelį (VGM) bei sumodeliavus kiekvienos *veiklos valdymo funkcijos Fi* ir *veiklos proceso Pj* porą *(Fi x Pj)* kaip valdomą procesą (elementarų veiklos valdymo ciklą), sudaryti formalią veiklos žinių struktūrą, kuri leistų sukurti žinių bazę, skirtą studijų procesui valdyti ir pertvarkyti į *žiniomis grįstą veiklą*;
- sudaryti apibendrintą studijų proceso valdymo modelį *modifikuoto VGM* pagrindu ir detalius *valdymo funkcijų* bei *procesų* sąveikos modelius, pvz.: *(F1 = „Studijų programų vykdymo valdymas“ ir P1 = „Studijų programų vykdymas aukštosiose mokyklose“)*; *(F2 = „Darbo rinkos reikalavimų nustatymo ir naujų kvalifikacinių reikalavimų formavimo valdymas“ ir P2 = „Darbo rinkos poreikių kitimas ir naujų kvalifikacinių reikalavimų formavimas“)*; *(F3 = „Mokymosi/studijų programų tobulinimo valdymas“ ir P3 = „Studijų programų tobulinimas“)*.

Literatūra

Bock, C. (2005). SysML and UML 2 Support for Activity Modeling Support for Activity Modeling. [Žiūrėta 2011 vasario 23 d.], <http://www.mel.nist.gov/msidlibrary/doc/sysmlactivity.pdf>.

Denisovas, V., Gudas, S., Tekutov, J., Tekutova, J., Brauklytė, I. (2009). Reikalavimų inžinerijos metodų taikymas informatikos pagrindinių studijų programoms tobulinti. *Vadyba,* 1 (14), 123–131.

Denisovas, V., Gudas, S., Tekutov, J., Tekutova, J., Galdikienė, S. (2010). Reikalavimų valdymo sistemos taikymas informatikos krypties neuniversitetinių studijų programų tobulinimui. *Profesinės studijos: teorija ir praktika:*,6, 239–251.

Denisovas, V., Gudas, S., Tekutov, J. (2010). Studijų programų reikalavimų inžinerijos metodas ir informacijos sistema. *Informacijos mokslai,* 53, 106–126.

Eriksson, H.-E., *et al.* (2004). *UML 2 Toolkit.* Wiley Publishing, Indianapolis.

France, R., Rumpe, B. (2007). Model-driven Development of Complex Software: A Research Roadman. In: *Future of Software Engineering at ICSE'07.* 37–54.

Frankel, D. (2003). *Model Driven Architecture: Applying MDA to Enterprise Computing.* John Wiley & Sons.

Gomaa, H., Olimpiew, E. M. (2005). The Role of Use Cases in Requirements and Analysis Modeling. *2nd International Workshop on Use Case Modeling (WUsCaM-05) Use Cases in Model-Driven Software Engineering.* Montego Bay, Jamaika. [Žiūrėta 2011 kovo 23 d.], <http://www.ie.inf.uc3m.es/wuscam-05/>.

Gudas, S. (1991). A framework for research of information processing hierarchy in enterprise. *Mathematics and Computers in Simulation,* 33(4), 281–285.

Gudas, S. (2009). Enterprise knowledge modeling: domains and aspects. *Technological and Economic Development of Economy,* 15, 2, 281–293.

Gudas, S., Brundzaitė, R. (2005). Enterprise knowledge modeling based on modified value chain model. *Information Sciences,* 35, 179–192.

Gudas, S., Denisovas, V., Tekutov, J., Brauklytė, I. (2009). Reikalavimų inžinerijos metodikos įgyvendinimas modernizuojant informatikos magistrantūros studijų programas. *XIV-osios tarpuniversitetinės magistrantų ir doktorantų mokslinės konferencijos „Informacinės technologijos 2009“ pranešimų medžiaga*, 229–234.

Hay, D. C. (2003). *Requirements Analysis: From Business Views to Architecture.* Prentice Hall PTR, New York.

Hause, M. (2006). The SysML Modelling Language. *Fifth European Systems Engineering Conference*, 1–12.

IBM kompanija. (2009). Trial: Rational Software Architect Standard Edition. [Žiūrėta 2011 sausio 17 d.], <http://www.ibm.com/developerworks/downloads/r/rsd/>.

IBM kompanijos Academic Initiative. (2009). Demo: Rational Rose & RequisitePro. [Žiūrėta 2011 sausio 11 d.], <http://www3.software.ibm.com/ibmdl/pub/software/rational/web/demos/viewlets/reqpro/RoseRP_integration/Rose_Integration_viewlet.html>.

IEEE Computer Society. (2000). *IEEE Recommended Practice for Architectural Description of Software Intensive Systems.* IEEE Std. 1471–2000. [Žiūrėta 2011 balandžio 11 d.],

<http://www.win.tue.nl/~johanl/educ/2II45/Lit/software-architecture-std1471-2000.pdf>.

Johnson, T. A., *et al.* (2007). Modeling continuous system dynamics in SysML. *Proceedings of the International Mechanical Engineering Congress and Exposition*, 1–9.

Kleppe, A., Warmer, J., Bast, W. (2003). *MDA Explained: The Model Driven Architecture™:* Practice and Promise. Addison Wesley, Boston.

Laužackas, R. (2008). *Kompetencijomis grindžiamų mokymo/studijų programų kūrimas ir vertinimas.* Vytauto Didžiojo universitetas, Kaunas.

Lietuvos Respublikos švietimo ir mokslo ministerija. (2007). Lietuvos Respublikos švietimo ir mokslo ministro įsakymas 2007 m. gruodžio 22 d. Nr. ISAK-2580 Informatikos studijų krypties reglamentas. [Žiūrėta 2011 sausio 20 d.], <http://www.smm.lt/smt/st_org/docs/st_regl/Informatika%20akt.pdf>.

Maskeliūnas, S. (2006). *Žinių technologijų pagrindinių terminų žodynėlis.* Matematikos ir informatikos institutas, Vilnius.

Nemuraitė, L. (2008). *Informacinių sistemų programinės įrangos projektavimas.* Klaipėdos universiteto leidykla, Klaipėda.

OMG koncorciumas. (2010). Systems Modeling Language specification. [Žiūrėta 2011 kovo 22 d.], <http://www.omg.org/spec/SysML/1.2/PDF/>.

Peak, R. S., *et al.* (2007). Simulation-Based Design Using SysML – Part 2: Celebrating Diversity by Example. *INCOSE Intl. Symposium in San Diego*, 1–22.

Porter, M. E. (1998). *Competitive Advantage:* Creating and Sustaining Superior Performance. Harvard Businesss School Publishing, Boston.

Robertson, S., Robertson, C. J. (2006). *Mastering the Requirements Process.* 2nd edition, Addison-Wesley, New York.

Sommerville, I. (2006). *Software Engineering*, 8th edition. Addison-Wesley, United Kingdom.

Weilkiens, T. (2007). *Systems Engineering with SysML/UML.* Penrose, Burlington.

Zielczynski, P. (2008). *Requirements Management Using IBM Rational RequisitePro.* IBM Press, Boston.

Кватрани, Т., Палистранта, Д. (2007). *Визуальное моделирование с помощью IBM Rational Software Architect и UML.* КУДИЦ-Пресс, Москва.

Николаев, А., Зыль, С. (2006). Визуальное проектирование на основе SysML. [Žiūrėta 2011 vasario 27 d.], <http://www.osp.ru/os/2006/05/2449867>.

Новичков, А., Карабанова, Г. (2010). Моделирование бизнес-процессов с Rational Software Architect. Часть 2. [Žiūrėta 2011 balandžio 22 d.], <http://www.interface.ru/home.asp?artId=23096>.

Рыжов, Д., Иванов, Д. (2009). Процесс разработки программно-аппаратных систем на основе визуального моделирования с использованием SysML/UML. [Žiūrėta 2011 vasario 20 d.], <http://www.interface.ru/home.asp?artId=20894>.

Смит, Б. (2009). Рекомендации по структурированию моделей при помощи ПО IBM Rational. Часть 2. [Žiūrėta 2011 balandžio 22 d.], <http://www.interface.ru/home.asp?artId=22316>.

Трофимов, С. А. (2002). *CASE-технологии: практическая работа в Rational Rose.* Изд. 2-е, Бином-Пресс, Москва.

STUDY PROGRAMME REQUIREMENTS ENGINEERING SYSTEM'S: FROM AUTOMATED DOCUMENT'S MANAGEMENT TO MODEL-BASED DEVELOPMENT PROCESS

S u m m a r y

The Unified Modeling Language (UML) is a modeling language from this field. It has established itself as a worldwide standard. The situation will change with the new Systems Modeling Language (SysML^{TM}). SysML, which is based on UML, is being increasingly used by systems engineers to model systems. As well as providing system requirements, SysML models can be used to define the system architecture to be used by the software engineers. This article has spelled out how SysML to UML can be used together in the newly developed study program requirements engineering system.

SysML provides a general purpose modeling language to support specification, analysis, design and verification of complex systems. These systems may include hardware, software, information, processes, personnel and facilities. SysML reuses a subset of UML 2 and provides additional extensions to satisfy the requirements of the language. The static and structural constructs used in SysML structure diagrams, including the package diagram, block definition diagram, internal block diagram, and parametric diagram. The dynamic, behavioral constructs used in SysML behavioral diagrams, including the activity diagram, sequence diagram, state machine diagram, and use case diagram. Two new diagram types have been added to SysML including the requirement diagram and the parametric diagram. A requirement diagram provides a modeling construct for text-based requirements, and the relationship between requirements and other model elements that satisfy or verify them.

In this article in order to familiarize with the models based Model-Driven System Engineering (MDSE) analyzed models based on systems engineering techniques. Systems engineering concentrates on the definition and documentation of system requirements in the early development phase, the preparation of a system design, and the verification of the system as to compliance with the requirements, taking the overall problem into account: operation, time, test, creation, cost and planning, training and support, and disposal. Systems engineering integrates all disciplines and describes a structured development process, from the concept to the production to the operation phase and finally to putting the system out of operation. It looks at both technical and economic aspects to develop a system that meets the users' needs. This article is devoted to the formulation and implementation of quite novel and interdisciplinary approach to curriculum development that is based on application of systems engineering methods and modern CASE tools. The future improvement of a created system is associated with knowledge-based applications. The presented approach uses the modified Value Chain Model (VCM) to describe the refinement procedure of the problem domain knowledge. Thus, study process parameters derived from VCM can be stored in knowledge databases and later used for solving the problems of expected application areas.

KEYWORDS: higher education, universities, study, computer science, curriculum planning, technology.

Jurij Tekutov. Mokslinis laipsnis: magistras (doktorantas). Darbovietė (-ės): Klaipėdos universitetas, Jūrų technikos fakultetas, Informatikos inžinerijos katedra; Klaipėdos valstybinė kolegija, Technologijų fakultetas, Informacinių technologijų katedra; Vakarų Lietuvos verslo kolegija, Informatikos katedra. Pareigos: lektorius. Mokslinių tyrimų kryptis: žiniomis grindžiami studijų proceso valdymo modeliai. Mokslinės publikacijos: 20 mokslinių straipsnių, 5 mokslinių tezių bei kelių e-metodinių medžiagų virtualioje mokymosi aplinkoje autorius; dalyvavo keliuose tarptautinėse mokslinėse ir respublikinėse jaunųjų mokslininkų konferencijose bei metodinėse-mokomosiose stovyklose. Adresas: Bijūnų g. 17, LT-91225 Klaipėda. Telefonas: 8 (46) 39 89 86. Elektroninis paštas: jurij@ik.ku.lt. Kita informacija apie autorių: nuo 2008 m. studijuoja VU KHF fizinių mokslų srities Informatikos (09 P) krypties doktorantūroje.

Saulius Gudas. Mokslinis laipsnis: daktaras. Darbovietė: Vilniaus universitetas, Kauno humanitarinis fakultetas, Informatikos katedra. Pareigos: profesorius. Mokslinių tyrimų kryptis: žiniomis grindžiama informacijos sistemų inžinerija, tyrimo objektas – kompiuterizuoto IS projektavimo metodai ir technologijos, kurios grindžiamos organizacijų veiklos žinių modeliais. Svarbiausieji darbo rezultatai – sukurtas ir paskelbtas tarptautinėje spaudoje formalus veiklos valdymo informacinių procesų modelis, veiklos informacinės architektūros modelis, veiklos žinių bazės modelis. Mokslinės publikacijos: 150 mokslinių straipsnių, 2 vadovėlių ir virš 10 metodinių priemonių autorius. Atlikti moksliniai tyrimai: tarptautinių ir nacionalinių mokslo bei studijų projektų vadovas, atsakingas vykdytojas. Adresas: Muitinės g 8, LT-44280 Kaunas. Telefonas: 8 (37) 42 25 23. Elektroninis paštas: saulius.gudas@khf.vu.lt. Kita informacija apie autorių: VU KHF dekanas.

Vitalijus Denisovas. Mokslinis laipsnis: daktaras. Darbovietė (-ės): Klaipėdos universitetas, Gamtos ir matematikos mokslų fakultetas, Informatikos katedra; Vakarų Lietuvos verslo kolegija, Informatikos katedra. Pareigos: profesorius. Mokslinių tyrimų kryptis: kompiuterinis modeliavimas, sistemų inžinerija, informatikos studijų programų kūrimas ir modernizavimas. Mokslinės publikacijos: daugiau nei 80 mokslinių straipsnių ir 15 vadovėlių bei metodinių priemonių autorius. Atlikti moksliniai tyrimai: kelių tarptautinių ir nacionalinių mokslo bei studijų projektų vadovas, atsakingas vykdytojas. Adresas: H. Manto g. 84, LT-92294 Klaipėda. Telefonas: 8 (46) 39 88 21. Elektroninis paštas: vitalij@ik.ku.lt. Kita informacija apie autorių: KU Informatikos katedros vedėjas.

Julija Tekutova. Mokslinis laipsnis: magistras. Darbovietė: Klaipėdos valstybinė kolegija, Technologijų fakultetas, Informacinių technologijų katedra. Pareigos: lektorė. Mokslinių tyrimų kryptis: informatikos studijų programų modernizavimas. Mokslinės publikacijos: daugiau nei 10 mokslinių straipsnių, 5 mokslinių tezių autorė; dalyvavo keliuose tarptautinėse mokslinėse ir respublikinėse jaunųjų mokslininkų konferencijose bei metodinėse-mokomosiose stovyklose; vadovavo studentų pranešimams mokslinėse-praktinėse ir tarptautinėse konferencijose. Adresas: Bijūnų g. 10, LT-91223 Klaipėda. Telefonas: 8 (46) 34 62 49. Elektroninis paštas: julija.tekutova@yahoo.com.