KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Inga Pašilskytė

Išplėstoji UML notacija verslo procesams modeliuoti ir specifikuoti

Magistro darbas

Vadovė
doc. dr. L. Nemuraitė

KAUNAS, 2005

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. R. Butleris

Išplėstoji UML notacija verslo procesams modeliuoti ir specifikuoti

Informatikos mokslo magistro baigiamasis darbas

Kalbos konsultantė
Lietuvių k. katedros lekt.
dr. J. Mikelionienė

Vadovas
doc. dr. L. Nemuraitė

Recenzentas
doc. dr. S. Maciulevičius

Atliko
IFM 9/1 gr. stud.
I. Pašilskytė

KAUNAS, 2005

SUMMARY

Visual languages are used for understandability of business analysis, modeling and computerization processes for business analysts as soon as system developers. The advantages of visual modeling are standard notation, unified concepts, and intuitive use. There is no single language suitable for all phases of business process evolution. Three languages were analyzed: UML 2.0, BPMN and BPEL4WS, with regards to their possibilities to represent and execute e-business processes.

It is proposed to extend UML 2.0 with BPMN stereotypes for e-business process modeling with succeeding transformation to execution language BPEL. The exclusive feature of proposed way of modeling lies in integration of business process model with object types of problem domain. In order to test the proposed method, scenario for modeling was prepared and three models were designed: two models using UML 2.0 notation, and one model using UML extended with BPMN stereotypes. Models were compared to show new method advantages.

For implementation of proposed method, UML CASE tool MagicDraw was extended with stereotypes required for the proposed method of modeling. Extended user interface was proposed for specification of business rules governing the e-business process.

Key words: UML, e-business process, BPMN, BPEL

Turinys

Lentelių sąrašas

Paveikslų sąrašas

ĮVADAS

Darbo tyrimo sritis – verslo procesų modeliavimas, kur šiuo metu atliekama nemažai tyrimų, norint išspręsti iškilusias problemas: didelė verslo modeliavimo kalbų aibė, skirtingos kalbos akcentuoja skirtingas verslo modelių savybes [1] ir nėra nė vienos, kuri tenkintų visus reikalavimus; daugelis kalbų yra tekstinio pavidalo ir verslo ekspertams neprieinamos; tinkamą kalbą pasirinkti sunku. Tam turėtų būti naudojama verslo analitikui priimtina vaizdinė kalba, leidžianti aprašyti globalius verslo procesus ir ilgas transakcijas; sąveikas tarp verslo dalyvių; vidinius organizacijų procesus; pranešimus ir verslo dokumentus, susiejant juos su verslo proceso žingsniais; kartu su ja turėtų būti ir išraiškų kalba, leidžianti aprašyti verslo taisykles ir apribojimus. Ši kalba turėtų būti realizuota įrankyje, kuris užtikrintų verslo procesų modelių kūrimą, išsaugojimą ir vykdymą ar perdavimą taip vadinamiems verslo procesų valdymo įrankiams. Yra daug verslo modeliavimo kalbų, kurios gerai įgyvendina kai kuriuos iš šių reikalavimų, tačiau nėra tokios kalbos, kuri tenkintų visus reikalavimus ir užtikrintų e. verslo proceso sukūrimą nuo reikalavimų lygmens modelio iki programinės realizacijos. Išanalizuotos labiausiai paplitusios modeliavimo kalbos: universali modeliavimo kalba – UML 2.0 (*Unified Modeling Language*) [2] [3] ir specializuota verslo procesų kalba – BPMN (*Business Process Modeling Notation*) [4] [5].

UML – viena iš plačiausiai sutinkamų grafinių programinės įrangos projektavimo ir veiklos modeliavimo kalbų. UML 2.0 yra universali kalba, ir verslo procesų modeliavimas yra tik viena jos naudojimo krypčių. Ši modeliavimo kalba nuolat tobulinama, norint palengvinti projektavimo procesą ir detalizuoti modelius iki realizavimo lygio. Naujausioje UML versijoje 2.0 yra didelės galimybės verslo procesams modeliuoti tiek veiklos, tiek vykdomojo kodo lygyje. Veiklos diagramos yra trijų lygių (trečio lygio šiame darbe nenagrinėjamos):

Pirmo lygio diagrama abstrakčiai vaizduoja veiklą.

Antro lygio diagramos gali būti dviejų tipų: tarpinės veiklos (*Intermediate Activities*) diagramos aprašo veiklos grafus, veiklos valdymo bei duomenų srautus; struktūrinės veiklos (*Structured Activities*) leidžia aprašyti darinius, panašius į naudojamus programose.

Trečio lygio diagramose detaliai aprašomi duomenų srautai, parametrai, klaidos ir pan.

Verslo analitikų poreikiams neseniai buvo sukurta verslo procesų modeliavimo notacija BPMN – palyginti naujas modeliavimo standartas, kuris turi didelę grafinių simbolių aibę verslo procesų žingsniams, srautams, trigeriams, klaidoms, kompensacijoms ir pan. vaizduoti. BPMN leidžia aprašyti bendradarbiaujančių organizacijų verslo procesus, kuriuos galima atvaizduoti į programinę realizaciją. Ši kalba turi vienintelio tipo diagramą, kurią galima naudoti įvairiais būdais: vidiniams verslo procesams – orkestruotei; abstraktiems viešiems procesams – choreografijai; globaliems bendradarbiavimo procesams ir įvairioms jų kombinacijoms. Tačiau BPMN turi eilę trūkumų: visų

pirma, tai dar vienas standartas, kurį reikia integruoti su projektavimo kalbomis ir įrankiais; antra, BPMN modeliuose verslo procesai nesusiejami su duomenų modeliais, kas yra būtina, norint generuoti verslo vykdymo programos kodą.

Šiame darbe siūloma e. verslo procesams modeliuoti pritaikyti universalią modeliavimo kalbą UML, papildant ją BPMN stereotipais. UML yra plačiai paplitęs programinės įrangos projektavimo, specifikavimo, realizavimo ir dokumentavimo standartas, todėl verslo procesų modelius galima integruoti su duomenų, paslaugų ir kitais modeliais, kurių reikia išsamiai verslo proceso realizacijai. Be to, daugelis projektuotojų naudoja UML ir efektyviau būtų naudoti vieną universalią notaciją bei CASE įrankius tiek informacinėms sistemoms, tiek kompiuterizuotiems procesams projektuoti.

Šiuo metu sukurta nemažai XML grindžiamų e. verslo kalbų, iš kurių didžiausio pripažinimo ir praktinių taikymų sulaukė BPEL (*Business Process Execution Language*) [6] [7], palaikanti XPath 1.0 funkcijas. Ji skirta formaliai specifikuoti veiklos procesus ir jų sąveikos protokolus. BPEL apibūdinama kaip „orkestruotės" (*orchestration*) kalba, nes ji leidžia aprašyti vienos organizacijos e. verslo procesus, tačiau suderinant kelių organizacijų procesų aprašus, galima apibrėžti ir taip vadinamą „choreografiją" (*choreography*), kuri sinchronizuoja skirtingų dalyvių sąveikų sekas. BPEL aprašytą verslo procesą galima vykdyti BPEL varikliais, kurių realizacijas yra sukūrę IBM, Oracle ir kt. BPEL netenkina dalies reikalavimų, keliamų e. verslo procesams vaizduoti, tačiau tuos reikalavimus būtų galima užtikrinti naudojant verslo analitikui suprantamą aukštesnio lygmens vaizdinę kalbą, kuri leistų atvaizduoti globalius verslo procesus, jų dalyvių choreografijas, ilgas transakcijas ir pan., po to automatiškai generuojant BPEL [8] [9] kodą atskiriems modelyje aprašytiems procesams. Transformavimo procesui į BPEL [10] [11] palengvinti tikslinga sudaryti verslo procesų atvaizdavimo metamodelį. Šaltiniuose [12] [13] [14] aprašytas e. verslo proceso metamodelis, kuris toliau vystomas šiame darbe. Jame vaizduojami transformuoti reikalingi veiklos procesai, sudėtinės veiklos, veiklos partnerio sąsajų tipai, reikalingų kintamųjų aibė, kiekvienam veiksmui atvaizduoti reikalingi tipai ir elementai. Šio metamodelio trūkumas – sąsajos su dalykinės srities modeliu nebuvimas.

Darbo tyrimo objektas – UML 2.0, BPMN ir BPEL notacijos, jų galimybės verslo procesams aprašyti ir kompiuterizuoti. Darbo tikslas – atlikti verslo procesų modeliavimo kalbų analizę ir sudaryti verslo analitikui pritaikytą verslo procesų specifikavimo metodiką, pagal kurią aprašytą verslo procesą būtų galima transformuoti į e. verslo vykdymo programos aprašą.

Darbe sprendžiami uždaviniai: išanalizuoti verslo modelių reikalavimus, tam skirtas UML 2.0, BPMN ir BPEL notacijas; sudaryti universalią modeliavimo metodiką papildant UML 2.0 verslo modeliavimo notacijos BPMN elementais ir susiejant dalykinės srities modelį su veiklos proceso modeliu; aprašyti išplėstosios notacijos atvaizdavimą į BPEL; pritaikyti šią metodiką UML CASE įrankiui ir eksperimentiškai išbandyti sukuriant realaus e. verslo proceso modelį; remiantis įgyta

patirtimi, sudaryti specifikacijų šablonus natūralia ir OCL (*Object Constrain Language*) kalbomis [15] [16] [17], suprojektuoti vartotojo sąsajos eskizą.

BPMN kalba realizuota plačios paskirties įmonių modeliavimo įrankiuose, pavyzdžiui, Popkin Architect, tačiau išsamios sąsajos tarp įvairių notacijų ir kūrimo etapų dar nėra pasiektos. UML 2.0 standartą stengiamasi įdiegti daugelyje įrankių, iš kurių galima paminėti Enterprise Architect Corporate v4.0 [18], MagicDraw [19] [20] ir kt.. UML 2.0 realizacijos yra labai naujos, natūralu, kad jose nemaža klaidų ir ne visos savybės įgyvendintos. Šiame darbe pasirinktas MagicDraw, kadangi jis gerai palaiko tradicinius projektavimo procesus; turi duomenų bazių, XML, WSDL schemų generavimo galimybes; OCL specifikavimo ir sintaksės tikrinimo priemones; plėtimo API; leidžia lengvai įvesti naujus grafinius stereotipus. UML 2.0 notacijos realizacija MagicDraw dar turi trūkumų, tačiau tikimasi, kad jie netrukus bus pašalinti. Vartotojo sąsaja naujam modeliavimo metodui sudaryta išplečiant MagicDraw reikiamais stereotipais ir veiklos taisyklių specifikacijomis.

Darbo struktūra:

1 skyriuje pateikiama vaizdinio modeliavimo kalbų, skirtų verslo procesams modeliuoti, analizė. Analizuojamos universali (*UML*), specializuota verslo (*BPMN*) bei verslo proceso vykdomoji modeliavimo kalbos (*BPEL*). Atliktas UML bei BPMN kalbų palyginimas pagal 23 požymius. Lyginimo metu atskleidžiami abiejų notacijų privalumai ir trūkumai, taip parodant, kad universalios modeliavimo metodikas, kuri tenkintų daugelį reikalavimų nėra. Išanalizuoti įvairūs šiuolaikinio verslo reikalavimai, kuriuos turėtų tenkinti sudaromi e. verslo modeliai. Atlikta populiarių CASE įrankių analizė. Pasiūlyta universalios modeliavimo metodikos bei CASE įrankio sukūrimo galimybė.

2 skyriuje pateikiamas verslo procesų modeliavimo ir specifikavimo metodas UML, BPMN ir BPEL pagrindu. Pasiūloma papildyti UML modeliavimo kalbą BPMN galimybėmis, taip suformuojant naują universalią modeliavimo notaciją, kurioje būtų galima užrašyti e. verslo taisykles bei modelį susieti su dalykine sritimi. Kad modelį būtų galima atvaizduoti į vykdomąją kalbą išanalizuotos BPEL galimybės ir sudaryti elementų atvaizdavimo į šią vykdomąją kalbą šablonai bei pavyzdžiai. Sudaryti ir pateikiami specifikacijų eskizai vartotojui natūralia bei OCL kalbomis.

3 skyriuje pateikiamas atliktas eksperimentas. Sudarytas e. verslo modelio scenarijus. Pasiūlyta universali metodika išbandoma pagal nurodytą scenarijų sudarant eksperimentinį e. verslo modelį. Kad atskleistume naujos modeliavimo metodikos privalumus, modelis sudarytas dviem būdais: naudojant UML notaciją ir naujai siūlomą metodą. Modeliai palyginti tarpusavyje. Kadangi geram e. verslo modeliui sukurti trūksta ne tik metodikos, bet ir įrankio, sudarytas CASE įrankio papildymo eskizas. Buvo pasirinktas MagicDraw UML CASE įrankis, turintis lanksčią vartotojo sąsają bei dideles plėtimo galimybes. Sudaryti elementų specifikacijos formų eskizai, kuriais tikslinga būtų papildyti esamą įrankį bei pateiktas modeliavimo pagal naują eskizą pavyzdys.

Darbo rezultatai buvo pristatyti konferencijoje „Informacinės technologijos 2005" iš konferencijų ciklo „Lietuvos mokslas ir pramonė" [21], vykusioje KTU š. m. sausio mėn. 26 – 27 d. Kaune.

Dar vienas straipsnis priimtas pristatyti tarptautinėje konferencijoje „Informacinės technologijos verslui 2005" [22], kuri vyks VU KHF š. m. gegužės mėn. 27 d. Kaune.

1. VAIZDINIO MODELIAVIMO KALBŲ, SKIRTŲ VERSLO PROCESAMS MODELIUOTI , ANALIZĖ

Verslo procesų modeliavimas yra aktuali, tyrinėjama sritis. Su šia mokslo sritimi susiję daugelis svarbių problemų: didelė verslo modeliavimo kalbų aibė, skirtingos kalbos akcentuoja skirtingas verslo modelių savybes ir nėra nė vienos, kuri tenkintų visus reikalavimus; daugelis kalbų yra tekstinio pavidalo ir verslo ekspertams neprieinamos. Tinkamą kalbą pasirinkti sunku. Be to, trūksta reikiamos modeliavimo metodikos, kuri atitiktų šiuolaikiniams e. verslo modeliams keliamus reikalavimus, kurie nuolat keičiasi. Todėl svarbu atlikti plačiausiai naudojamų verslo procesų modeliavimo kalbų analizę ir sudaryti verslo analitikui pritaikytą verslo procesų specifikavimo metodiką, pagal kurią aprašytą verslo procesą būtų galima transformuoti į e. verslo vykdymo programos aprašą.

1.1. Reikalavimų e. verslo procesams modeliuoti analizė

Reikalavimų e. verslui modeliuoti yra gana daug, jie įvairūs bei nuolat tobulinami pritaikant šiuolaikinio verslo modelių savybėms atvaizduoti. Reikalavimų paskirtis: aprašyti globalius verslo procesus, transakcijas tarp verslo dalyvių, pranešimus bei verslo dokumentus, įvairias verslo taisykles, apribojimus, ilgalaikės informacijos saugojimą, valdymą ir kt.. Žemiau pateikti reikalavimai e. verslui modeliuoti remiasi ankstesniuose magistrų darbuose pateiktomis išvadomis bei papildyti darbo metu iškilusiais svarbiais modeliavimo aspektais.

Anksčiau analizuoti reikalavimai:

Bendradarbiavimo aspektų (globalaus proceso) modeliavimas

Šiuolaikinių verslo procesų modeliuose atskirai vaizduojami bendradarbiavimo procesai ir vidiniai vieno dalyvio procesai. Globalų verslo procesą sudaro bendradarbiavimo sąveikos tarp proceso dalyvių (vaidmenų). Bendradarbiavimo proceso (sąveikų) sekos modelis vadinamas choreografija, o vidinių procesų modeliai – orkestruotėmis. Išsami verslo procesų modeliavimo kalba turi apimti ir globalų bendradarbiavimo vaizdą, ir vidinių procesų valdymą.

Darbų srautų vaizdavimas (reikalavimai choreografijai ar orkestruotei)

Šie reikalavimai apima valdymo srautų ir duomenų srautų vaizdavimą, įvairius jungimų ir išskaidymų tipus, rekursyvų sudėtinių procesų apibrėžimą, sprendimo taškus, trigerius, ciklus ir pan.

Transakcijų valdymas

Koordinuojamų atominių (tenkinančių ACID reikalavimus) ir išplėstų (ilgai trunkančių) transakcijų modeliavimas, kompensuojančių veiksmų, laiko apribojimų atsako gavimui ar siuntimui vaizdavimas.

Išskirtinių atvejų (klaidų) apdorojimas

Turi būti numatyti klaidų aptikimo ir atstatymo veiksmai.

Vykdomasis modelis

Vykdomuoju (*executable*) modeliu suprantamas pilnas dalyvių bendradarbiavimo ir vidinių procesų aprašas, kuris gali būti vienareikšmiškai atvaizduojamas į programos kodą (pvz., BPEL).

Verslo paslaugų sąsajos

Šiuolaikinėse verslo modeliavimo kalbose procesų modelis naudoja laisvai susiejamas verslo paslaugas (*web servisus*), kurias teikia verslo dalyviai. Jos aprašomos WSDL (*Web Services Definition Language*) kalba.

Pranešimų ir verslo dokumentų apibrėžimas

Tai verslo informacinio modelio sudarymas. Vykdomasis modelis neįmanomas be informacinio modelio.

Sutartys ir partnerių profiliai

Dviejų ar daugiau partnerių susitarimai vykdyti numatytas verslo funkcijas.

Pranešimų saugumas ir patikimumas

Patikimas ir saugus pranešimų pristatymas, elektroninis parašas ir autentikavimas.

Audito seka

Verslo dalyvių transakcijų neišsižadėjimo užtikrinimas. Tai reiškia, kad duomenys apie verslo proceso vykdymo žingsnius, jų rezultatus bei vykdytojus turi būti saugomi saugykloje.

Darbo eigoje iškelti reikalavimai:

Laiko elementų modeliavimas

Laiko elementai užduoda tam tikrą laiko ciklą, kurio neviršijus procesas bus tęsiamas. Arba atėjus tam tikram laiko momentui procesas pradedamas.

Verslo taisyklių modeliavimas

Modeliuojant verslo procesus įvairiems proceso elementams turėtų būti užduodamos taisyklės. Procesas tęsiamas jei užduota taisyklė yra tenkinama

Parametrų rinkinių modeliavimas

Verslo procesams turi būti apibrėžiami ir atskiriami įėjimo bei išėjimo parametrai. Įėjimo/išėjimo parametrai turi būti grupuojami į rinkinius, pagal tai kokiu metu jie ateina/išeina.

Modeliuojamo proceso nutraukimas

Modeliuojant procesus turi būti numatyta, kad procesas gali būti nutrauktas savo noru, nebūtinai atsiradus klaidai ar įvykus visam procesui.

Modelio elementų rinkinių sudarymas

Modeliuojant svarbu nustatyti ir sudaryti elementų rinkinius, pvz., turime pabaigų rinkinį, raiškia proceso baigtis gali būti įvairi.

Ilgalaikis informacijos saugojimas ir valdymas

Modeliuojant turi būti suteikta galimybė užfiksuoti ir saugoti ilgalaikę informaciją. Turi būti įvestas ir ilgalaikės informacijos saugojimo buferis, kuris valdytų kelis srautus iš įvairių šaltinių ir nuotolių.

1.2. Verslo modeliavimo kalbos (*BPMN*) analizė

BPMN naujas modeliavimo standartas, suteikiantis galimybę suvokti vidinius verslo procesus grafiškai juos atvaizduojant bei šiuos procesus standartizuojant. BPMN – procesų modeliavimo pagrindas ir metodologija. Šis standartas palengvina žmonėms iš skirtingų įmonių perprasti vieni kitų verslo procesus, sujungti verslus, ar juos suskaidyti į skirtingus. BPMN standartas padeda išsiaiškinti visiškai skirtingų verslo procesų gyvavimo ciklą nuo sukūrimo iki vystymo, realizavimo, vykdymo, analizavimo. Taip pat suteikia galimybę suprasti bendradarbiavimą ir transakcijas tarp organizacijų. BPMN standartas leidžia numatyti galimas gyvavimo ciklo eigos išimtis, klaidas. Patogus aprašyti verslo taisykles, ciklus, sprendimų taškus, trigerius.

BPMN yra vienintelio tipo diagrama, kurią galima naudoti įvairiais pavidalais:

Vidinių verslo procesų diagramos (*Private (internal) business processes*). Tai vidinis įmonės darbų srautų modelis (orkestruotė). Jeigu projektuojamos vidinės verslo diagramos naudojant juostas, tai vidinė diagrama negali išeiti už jos ribų. Jeigu dvi juostos sujungtos, tai reiškia, kad ne jų atskiri elementai, o vidinės diagramos siejasi viena su kita.

Sąveikų procesų diagramos (*Abstract (public) processes*). Jos rodo, kaip sąveikauja skirtingi verslo procesai (choreografija). Į diagramas įtraukiami tik tie procesai kurie veikia vidinių verslo procesų išorėje.

Bendradarbiavimo proceso diagramos (*Collaboration (global) processes*). Šios diagramos parodo globalius procesus. Jos gali būti vaizduojamos be juostų. Veiksmai priklausantys tam pačiam vidiniam procesui šiose diagramose gali būti susieti.

Naudojant BPMN verslo modeliavimo diagramos taip pat gali būti kelių tipų:

- Aukšto lygio vidinių procesų veiklos.
- Detalizuotas vidinis verslo procesas.
- Senas verslo procesas.
- Naujas verslo procesas.
- Detalizuotas verslo procesas su sąsajomis, bei išoriniais verslo dalyviais.
- Du ar daugiau sąveikaujantys verslo procesai.
- Detalizuoto vidinio proceso ryšys su sąsajos procesu.
- Detalizuoto vidinio proceso ryšys su bendradarbiavimo procesu.
- Sąsajos ir bendradarbiavimo procesas.

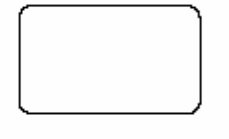
- Du ar daugiau verslo procesų sąveikaujantys su sąsaja
- Du ar daugiau verslo procesų sąveikaujantys su bendradarbiavimu.
- Du ar daugiau verslo procesų sąveikaujantys su sąsajų ir bendradarbiavimo procesais

BPMN notacijos elementai, skirti e. verslui modeliuoti:

Įvykis (*Event*) – tai kas atsitinka verslo proceso metu. Įvykiai nulemia proceso eigą ir dažniausiai turi priežastį. Pagal tai, kaip įvykiai nulemia proceso eigą, BPMN notacijoje yra trys įvykio tipai: pradinis, tarpinis ir pabaigos. Pradžios įvykis vaizduojamas tam tikro proceso pradžioje. Tarpinis įvykis įvyksta tik po to, kai procesas prasidėjęs. Jis turi įtakos proceso eigai, bet nei pradėti, nei nutraukti proceso negali. Pabaigos įvykis vaizduojamas ten, kur baigiasi procesas.

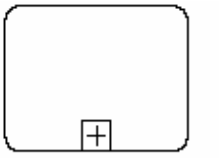
Pradžios įvykis[1] Tarpinis Įvykis Pabaigos Įvykis

Užduotis (*Task*) – atominis veiksmas, kuris toliau neskaidomas.

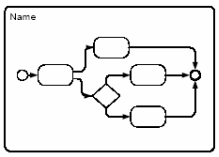


Užduotis

Sudėtinis procesas (*Subprocess*) – procesas, sudarytas iš kitų procesų. Sudėtinis procesas gali būti suspaustas, bet gali būti ir išplėstas. Suspaustas tuomet, kai procesai esantys viduje yra nerodomi, o tik specialus ženkliukas parodo, kad šis procesas yra sudėtinis. Suspausto proceso atveju mums nesvarbus vidus (iš ko jis sudarytas). Išplėstas tai toks, kurio viduje esantys procesai yra vaizduojami ir visiems aiškiai matomi. Išplėstas procesas yra vaizduojamas kaip baltoji, o suspaustas – kaip juodoji dėžė.

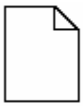


Suspaustas sudėtinis procesas (juodoji dėžė)



Išplėstas sudėtinis procesas (baltoji dėžė)

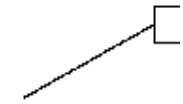
Duomenų objektai (*Data object*) – jie susiję su artefaktais. Duomenų objektai neturi konkretaus poveikio procesui, jie tik aprašo ką procesas veikia.



Duomenų objektai

[1] Taip pateikiami notacijų elementų paveikslai, kurie darbe nenumeruojami

Teksto anotacija (*Text anotation*) – parodo papildomą informaciją apie diagramas.



Teksto anotacija

Trigeriai (*Trigger*) – pradžios, tarpiniai ir pabaigos įvykiai gali turėti trigerius. Trigerių tipai pateikti 1 lentelėje:

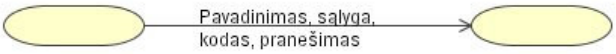
1 Lentelė. Trigerių tipai ir savybės.

Trigeris	Specifikacija
Pranešimo trigeriai (*Message triggers*). 	Pranešimo trigeriai gali būti pradžios, tarpiniai ir pabaigos. Pradžios trigeris pradeda procesą, tarpinis trigeris tęsia procesą, pabaigos praneša apie proceso pabaigą. Pabaigos trigeris veikia, jei buvo tarpinis trigeris.
Laiko trigeriai (*Timer trigger*). 	Laiko trigeriai gali būti tik pradžios ir tarpiniai. Jie užduoda tam tikrą laiko tarpą, kurio neviršijus (arba atėjus tam tikram laikui) procesas bus tęsiamas.
Taisyklių trigeriai (*Rule trigger*). 	Taisyklių trigeriai gali būti tik pradžios arba tarpiniai. Procesas tęsiamas jei užduota taisyklė yra tenkinama.
Ryšio trigeriai (*Link trigger*). 	Ryšio trigeriai gali būti pradžios, tarpiniai ir pabaigos. Ryšio trigeriai naudojami jei procesas gali prasidėti tik pasibaigus kitam procesui, ar pasiekus tarpinę būseną ir kt.
Išsišakojantis trigeris (*Multiple trrigers*). 	Išsišakojantys trigeriai gali būti pradžios, tarpiniai ir pabaigos. Išsišakojantys trigeriai naudojami tada kai gali būti keli įvykio variantai. Procesas paleidžiamas (tęsiamas, baigiamas) kai kuri nors viena iš galimų alternatyvų tenkinama.
Klaidos trigeris (*Exception trriger*). 	Klaidos trigeriai gali būti tik tarpiniai arba pabaigos. Tarpiniai trigeriai užfiksuoja išimtį arba klaidą proceso eigoje, o pabaigos išimties trigeriai sustabdo procesą .
Kompensavimo trigeris (*Compensation trigger*). 	Kompensavimo trigeriai gali būti tik tarpiniai arba pabaigos Šie trigeriai naudojami sugrąžinti procesus pakartotiniam vykdymui.
Atšaukimo trigeris (*Cancel*	Atšaukimo trigeriai gali būti tarpiniai ir pabaigos. Jie

trriger). 	naudojamas tik tuomet, kai procesas yra nutraukiamas savo noru.
Priverstinės pabaigos trigeris (*Terminate trriger*). 	Priverstinės pabaigos trigeriai gali būti tik pabaigos. Jie naudojami tuomet kai įvyko klaida ir procesas nutraukiamas priverstiniu būdu.

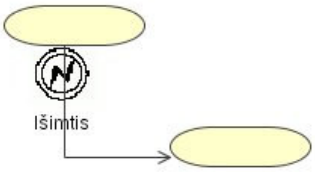
Sekos srautas (*Sequence flow*) – parodo eigą kaip procesai vyks. Sekos srautai gali būti kelių rūšių:

Normalus srautas (*Normal flow*) – toks srautas, kuris prasideda pradžioje ir tęsiasi per alternatyvius kelius iki pabaigos. Normalus srautas gali tęstis esant tam tikroms sąlygoms tik tuomet, jei išpildžius tas sąlygas jis bus tęsiamas iki pabaigos.



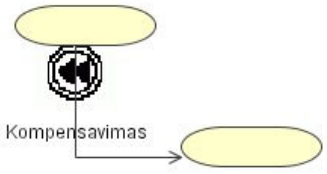
Normalus proceso srautas

Išimtinis srautas (*Exception flow*) – tai toks srautas, kuris atsiranda normalaus proceso eigos išorėje įvykus tarpiniam įvykiui, kuris pakeičia srauto tėkmę



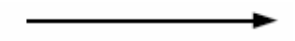
Išimtinis srautas

Proceso kompensavimo srautas (*Compensation association flow*) – tai toks srautas, kuris atsiranda įvykus tarpiniam įvykiui. Esant tokio tipo srautui reiškia, kad šis srautas buvo pradėtas kažkada anksčiau, bet dėl tam tikrų priežasčių buvo nutrauktas.



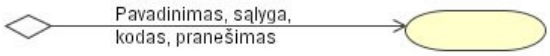
Proceso kompensavimo srautas

Nevaldomas srautas (*Uncontrolled flow*) – tai toks srautas, kuris nėra įtakojamas jokių sąlygų ir neina per sprendimų taškus. Jis teka nuo šaltinio iki galutinio objekto.



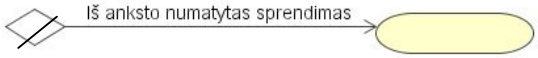
Nevaldomas srautas

Sąlygos srautas (*Conditional flow*) – tai toks srautas, kuris naudojamas, jeigu išpildyta sąlyga. Srautas vaizduojamas su rombu pradžioje jei jis neišeina iš sprendimų taško, priešingu atveju rombo braižyti nereikia.



Sąlygos srautas

Numatytas srautas (*Default flow*) – tai toks srautas, kuris naudojamas tik tuomet, kai netenkinamos jokia kita srauto sąlyga. Šis srautas numatomas iš anksčiau.



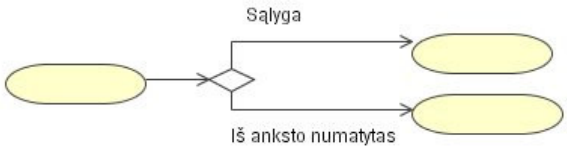
Numatytas srautas

Pranešimų srautas (*Message flow*) – parodo pranešimų srautą verslo dalyvių, kurie pasiruošę tuos pranešimus siųsti arba priimti. BPMN atskiri verslo dalyviai vaizduojami atskiruose juostose.

Pranešimų srautas

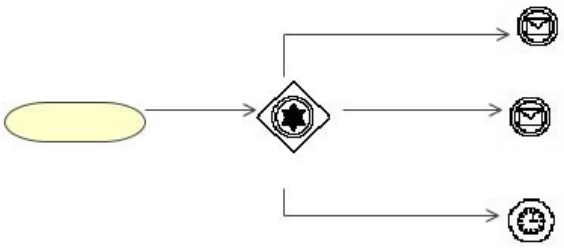
Sprendimas (*Decision*) – tai proceso sekos elementas, kuomet procese galima pasirinkti vieną iš alternatyvių tolesnės eigos sekų. Sprendimai gali būti kelių rūšių:

Duomenimis pagrįstas sprendimas (*Data based decision*). Kelias bus parinktas įvertinus esamas duomenų reikšmes. Jei duomenų reikšmės neatitinka nė vieno iš alternatyvinių kelių tuomet priskiriamas iš anksčiau numatytas (*default*) kelias.



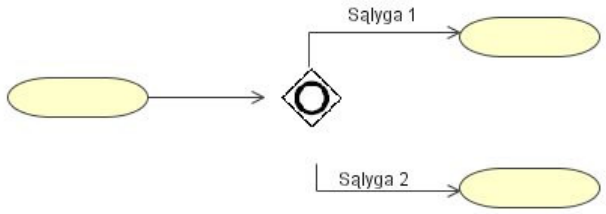
Duomenimis pagrįstas sprendimas

Įvykiais pagrįstas sprendimas *(Event based decision*). Kelio pasirinkimą šiuo atveju apsprendžia tarpinis įvykis. Kuris įvykis yra tenkinamas tas kelias ir pasirenkamas.



Įvykiais pagrįstas sprendimas

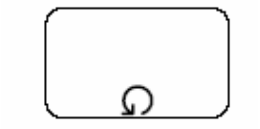
Apimantis sprendimas (*Inclusive decision*). Tai hibridas tarp išsišakojimo ir sujungimo. Priklausomai nuo to kiek yra kelių tiek yra ir tolimesnės eigos kombinacijų.



Apimantis sprendimas

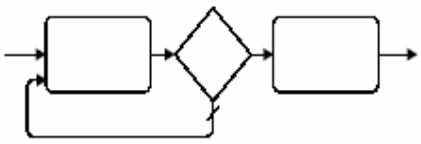
Ciklai (*Loops*) – ciklai gali būti dviejų tipų:

Veiksmų ciklai (*Activity looping*). Užduotys ir procesai bus sustabdyti, jei šie įvykiai įvyks tik vieną kartą.



Veiksmų ciklai

Sekų ciklai (*Sequence flow looping*). Sekų ciklai sudaromi sekos srautą jungiant su prieš tai einančiais objektais. Sekų ciklai naudojami ne tada kai tam tikras įvykis kartojamas keltą kartų, o kai tam tikra įvykių seka kartojama tam tikrą kiekį kartų.



Sekų ciklai

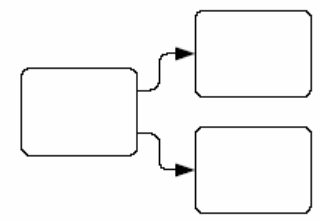
Konteineris (*Pool*) – grafinė riba, kuri atskiria veiksmų sekas vienas nuo kitų.

Konteineris

Juostos (*Lanes*) – tai konteinerio dalis. Jie reikalingi veiklas konteineryje suskirstyti į kategorijas.

Juostos

Išsišakojimas (*Fork*) – Išsišakojimai naudojami tuomet kada vienas kelias gali būti išskaidytas į keletą kelių, ir veiksmai esantys išskaidytuose keliuose gali būti vykdomi lygiagrečiai. Bet kad įvykis toliau tęstųsi visi įvykiai išskaidytuose keliuose turi būti atlikti (ir loginė operacija).

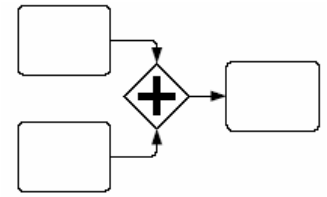


Išsišakojimas

Sujungimas (*Join*) – sujungimai naudojami, kai įvykius vykstančius lygiagrečiai norima sujungti į vieną. Sujungimai yra dviejų rūšių:

IR sujungimas – bus sujungta tik tuomet, kai visi įvykiai bus įvykdyti.

ARBA sujungimas – bus sujungta kai bent vienas iš įvykių bus įvykdytas.



Sujungimas

Grupavimas (*Groups*) – gali būti grupuojami įvairių procesų elementai. Grupavimas proceso eigos nekeičia.

Grupė

Rinkiniai (*Sets*) – naujas, naudingas modeliavimo elementas. Tarkim, kad turime pabaigų rinkinį, raiškia proceso baigtis gali būti įvairi.

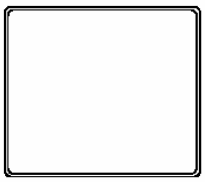
Sprendimų taškas (*Gateway control type*) – kontroliuoja išsiskiriančius sekų srautus. Jis apibrėžia išskaidymą bei sujungimą. Įvairūs sprendimo taškų tipai pateikti 2 lentelėje.

2 lentelė. Sprendimo taškų tipai ir savybės.

Sprendimo taškas	Specifikacija
Išimtinis (*Exclusive*) sprendimų taškas. Duomenimis pagrįstas (*Data based*).	Kai reikalinga srautų kontrolė BPMN naudojamas sprendimų taškas. Atėjus srautui, tikrinama, kuri sąlyga yra išpildoma. Ta atšaka, kurioje sąlyga yra išpildoma tęsiamas srautas. Tarp visų alternatyvių srautų gali būti tik vienas kuris

 Įvykiais pagrįstas (*Event based*). 	užbaigs srautų eigą Šis sprendimas ypatingas tuo, kad lygiagretus išsišakojimas grįstas įvykiais atsirandančiais procese. Šis taškas atlieka papildomą sekų kontrolę. Po šio sprendimų taško seka tarpiniai įvykiai. Kai įvykis atvyksta tikrinama, kuris kelias tenkinamas tuo keliu srautas siunčiamas toliau.
Daugialypis (*Inclusive*) sprendimų taškas (*or*). 	Šis taškas priėmęs ateinantį srautą išskaido į tiek lygiagrečių kelių, kiek sąlygų yra tenkinama. Nuo išplėtimų taško skiriasi tuo, kad galima pasirinkti ne vieną, o nulį ir daugiau išeinančių alternatyvių srautų. Suteikiama galimybė nesirinkti nė vienos alternatyvos.
Sudėtinis (*Complex*) sprendimų taškas. 	Sudėtinis sprendimų taškas BPMN naudojamas, kad būtų galima apibrėžti kiek ateinančiu srautų gali būti tęsiami. Likę srautai užblokuojami sprendimų taške.
Lygiagretus (*Parallel*) sprendimų taškas. 	Lygiagretus išskaidymas – tai vieno kelio išskaidymas į du ir daugiau srautų vykdomų lygiagrečiai.

Transakcija (*Tranzaction*) – veikla, užduotis arba sudėtinis procesas, palaikomas tam tikro protokolo, kuris užtikrina, kad visos dalys įeinančios į transakciją bus arba tikrai įvykdytos, arba nutrauktos.



Transakcija

1.3. Universalios modeliavimo kalbos (*UML 2.0*) analizė

UML modeliavimo kalba gana paplitusi, lengvai išmokstama ir patogi realizuoti, specifikuoti, dokumentuoti. UML modeliavimo kalbai būdinga diagramų įvairovė, todėl ši modeliavimo kalba labai

lanksti ir patogi projektuoti. UML modeliavimo kalboje diagramos skirstomos į tris kategorijas: statines, dinamines bei organizavimo, valdymo. Ši modeliavimo kalba nuolat tobulina ir papildoma. Nauja modeliavimo kalbos UML versija UML 2.0. UML 2.0 suteikia galimybę taip suprojektuoti sistemą, kad ji būtų labai artima realizuojamai sistemai. Šis standartas papildytas elementais, skirtais projektuoti verslo modeliams (duomenų saugykla, daugybė taškų tipų, veiklos suskaidymas, pertraukimas ir kt.). UML 2.0 yra universali kalba, ir verslo procesų modeliavimas yra tik viena jos naudojimo krypčių. Kadangi vienas iš darbo uždavinių yra sulyginti dvi modeliavimo notacijas BPMN ir UML dėmesys bus koncentruojamas į UML 2.0 veiklos diagramas.

Veiklos diagramos yra trijų lygių (trečio lygio šiame darbe nenagrinėjamos):

Pirmo lygio diagrama abstrakčiai vaizduoja veiklą

Antro lygio diagramos gali būti dviejų tipų:

Tarpinės veiklos (*Intermediate Activities*) diagramos aprašo veiklos grafus, veiklos valdymo bei duomenų srautus.

Struktūrinės veiklos (*Structured Activities*) leidžia aprašyti darinius, panašius į naudojamus programose.

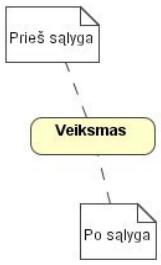
Trečio lygio diagramose aprašomos tokios konstrukcijos kaip briaunos svoris, srautų suskaidymas ir kt.

UML 2.0 veiklos modeliavimo elementai:

Veiksmas (*Action*) – yra esminis vykdomojo funkcionalumo vienetas veikloje. Veiksmas parodo kai kurias transformacijas ar procesus modeliuojamoje sistemoje. Šis elementas turi įeinančių ir išeinančių srautų. Veiksmas nevyksta kol visi įeinantys srautai nėra išpildyti.

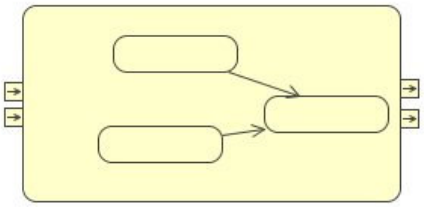
Veiksmas

Veiksmai gali turėti prieš ir po sąlygas:



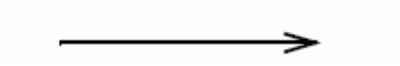
Veiksmas su prieš ir po sąlyga

Veikla (*Activity*) – detalizuoja vykdymo elgseną naudodama kontrolės ir duomenų sekų modelius. Veikla gali būti šalutinių procesų vykdymo pagrindas. Taip pat joje gali būti įtrauktos sekų valdymo konstrukcijos.



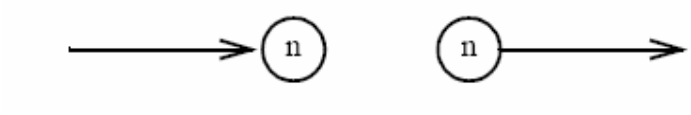
Veikla

Veiklos lankas (*ActivityEdge*) – Tai jungtis tarp dviejų veiklos taškų. Kokia tvarka išdėstytos veiklos ir srautai, tokia tvarka vyksta ir procesas. Seka tęsiasi nuo vienos sekos iki kitos.



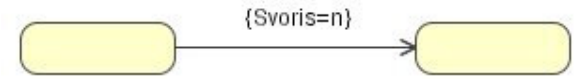
Veiklos lankas

Veiklos lankas gali turėti jungiamąjį elementą



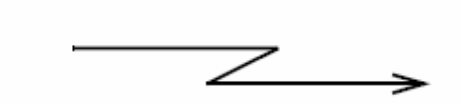
Veiklos lankas su jungiamaisiais elementais

Veiklos lankas su užduotu svoriu:



Svorinis veiklos lankas

Pertraukimų veiklos lankas:



Pertraukimų veiklos lankas

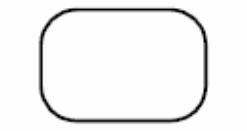
Veiklos pabaigos taškas (*ActivityFinal Node*) – galutinis taškas, kuris užbaigia visas veiklas.

Pabaigos taškas

Veiklos grupė (*Activity group*) – tai veiklos taškų ir ryšių rinkinys.

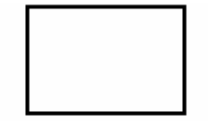
Veiklos taškas (*Activity nodes*) – tai veiklos jungiamųjų žingsnelių abstrakti klasė. Veiklos taškai skirstomi į:

Veiksmų taškus (*Action nodes*)



Veiksmo taškas

Objektų taškus (*Object nodes*)



Objekto taškas

Kontrolės taškus (*Control nodes*)

Sprendimų taškas | išsišakojimų taškas | pradžios taškas | pabaigos taškas | sekos pabaigos taškas

Sąlygos taškai (*Condition nodes*) – taškas, kuris suteikia galimybę pasirinkti tarp duotų alternatyvų.

Pradžios taškas (*Start node*) – taškas, kuris pradeda seką kai yra sužadinamas.

Pradžios taškas

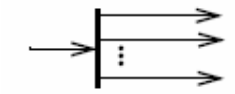
Sprendimų taškas (*Decision node*) – taškas, kuris suteikia galimybę pasirinkti tarp išeinančių srautų.

Sprendimo taškas

Sekos pabaigos taškas (*Flow final node*) – taškas, kuris užbaigia srauto seką.

Sekos pabaigos taškas

Išsišakojimų taškas (*Fork node*) – taškas, kuris išskaido seką į kelias sekas.

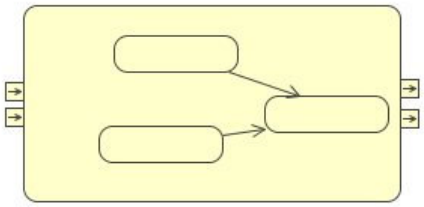


Išsišakojimas

Duomenų saugykla (*Data store node*) – centrinis buferis ilgalaikei informacijai saugoti.

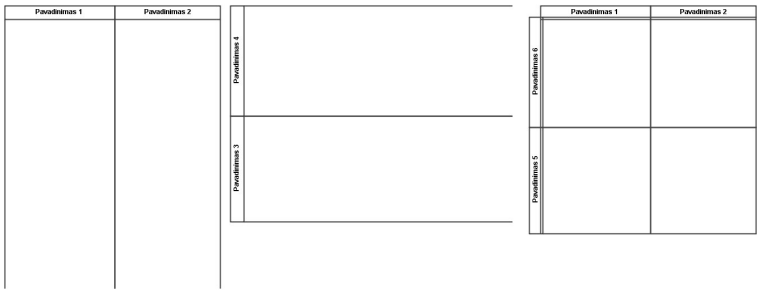
Duomenų saugykla

Veiklos parametrai (*Activity Parametre Node*) – veiklų įėjimo ir išėjimo parametrai.



Veiklos parametrai

Veiklos grupavimas (*Activity partition*) – tai veiklos grupė, kurioje išskirtos bendros charakteristikos. Atskiros dalys gali dalintis komponentais.



Veiklos grupės

Centrinis buferis (*Central buffer node*) – buferis, kuris valdo kelis srautus iš įvairių šaltinių ir nuotolių.

Centrinis buferis

Įvykio priėmimo veiksmas (*Accept evenet action*) – veiksmas, kuris laukia įvykio išpildžius tam tikras reikiamas sąlygas. Gali būti dviejų tipų priėmimo veiksmai: signalo atėjimo ir laiko įvykio.

Signalo įvykio

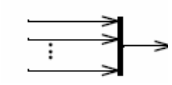
Laiko įvykio

Veiklų rinkinys (*Activity group*) – tai veiklų, sprendimų taškų rinkinys.

Pertraukiamų veiklų rinkinys (*Interupptible activity region*) – veiklų rinkinys, kuris palaiko pertraukimus.

Pertraukiamų veiklų rinkinys

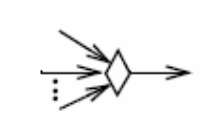
Sujungimų taškas (*Join node*) – taškas, kuris valdo kelis įeinančius srautus.



Sujungimų taškas

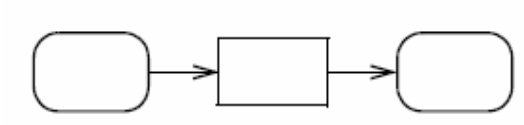
Ciklo viršūnė (*Loop node*) – ši viršūnė valdo ciklus, jų paleidimus testavimus, skaidymus.

Jungimų taškas (*Merge node*) – šis taškas sujungia kelis alternatyvius srautus. Naudojamas srautams sinchronizuoti.



Jungimo taškas

Objektų srautai (*Object flow*) – tai srautai, kurie gali pernešinėti objektus ir duomenis.



Objektų srautai

Objektų viršūnės (*Object nodes*) – tai abstrakčios veiklos viršūnės, kurios aprašo dalį objektų srautų veikloje.

Objektų viršūnės

Objektų aibės

Signalo objektai

1.4. Specializuotos (*BPMN*) ir universalios (*UML 2.0*) kalbos palyginimas

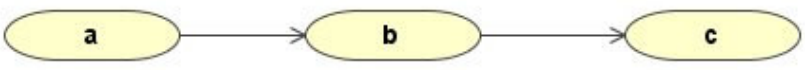
Ieškant vartotojui tinkamiausios e. verslo modeliavimo notacijos išnagrinėtos plačiausiai naudojama universali modeliavimo kalba – UML 2.0 ir specializuota verslo procesų kalba – BPMN. Išanalizuota notacijų paskirtis, elementai, pateiktas jų grafinis vaizdas ir pavyzdžiai. Šiame skyriuje pateiktas UML 2.0 ir BPMN palyginimas pagal 23 įvairius požymius. Lyginant atskleidžiami šių notacijų privalumai bei trūkumai. Kadangi kiekviena iš analizuojamų kabų turi savų privalumų ir trūkumų siūloma praplėsti UML 2.0 specializuotos verslo proceso kalbos galimybėmis ir sudaryti universalią modeliavimo metodiką.

1.4.1. Sekos (*Sequence patterns*)

Sekos – veiklų išdėstymo tvarka, kai vienai veiklai pasibaigus pradedama kita veikla.

BPMN:

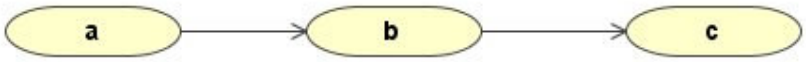
Sekos veiklos proceso diagramose tęsiasi nuo šaltinių iki veiklos pabaigos taško. Seka teka nuo vienos veiklos iki kitos visame procese. BPMN notacijoje sekos kontroliuojamos pagal sprendimų taškų tipus nurodytus, sprendimų taško (rombo) viduje.



Seka

UML:

UML 2.0 veiklos diagramose veiklos susietos srautais. Kokia tvarka išdėstytos veiklos ir srautai, tokia tvarka vyksta ir procesas. Seka tęsiasi nuo vienos sekos iki kitos. UML 2.0 standarto diagramose nėra įtraukiama jokių sąlygų, jokių valdymo kontrolių. Veiklos diagramose daugiau esminių skirtumų nėra. Abiejuose standartuose veiklai žymėti naudojami stačiakampiai užapvalintais kampais. Srautai žymimi rodyklėmis.



Seka

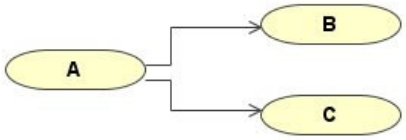
1.4.2. Lygiagretus išskaidymas (*Parallel split*)

Lygiagretus išskaidymas – tai vieno kelio išskaidymas į du ir daugiau lygiagrečiai einančių kelių.

BPMN:

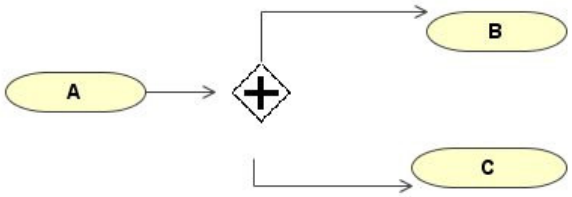
BPMN notacijoje yra trys lygiagretaus išskaidymo būdai:

Pirmas – išskaidymas be jokių apribojimo sąlygų. Išskaidomos šakos neturi jokių kontrolės apribojimų, jokių sprendimų taškų.



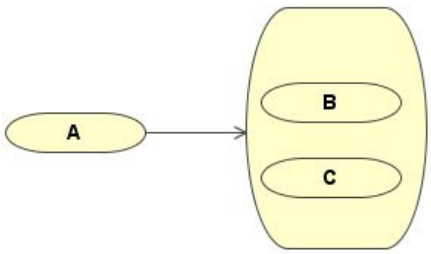
Pirmo BPMN lygiagretaus išskaidymo būdo pavyzdys.

Antras – išskaidant naudojamas lygiagretus sprendimų taškas. Kai tik srautas ateina į sprendimų tašką, jis iškart išsiuntinėjamas per išeinančias sekas.



Antro BPMN lygiagretaus išskaidymo būdo pavyzdys.

Trečias – srautas perduodamas į sudėtinį procesą



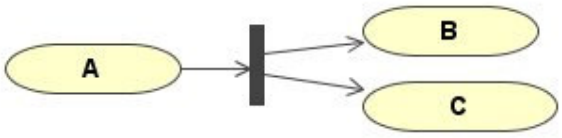
Trečio BPMN lygiagretaus išskaidymo būdo pavyzdys.

UML:

UML 2.0 diagramose lygiagretiems srautams išskaidyti yra naudojamas išskaidymų taškas. Veiklos į kurias eina lygiagretūs srautai gali prasidėti vienu metu.

Antras atvejis – sudėtinio proceso sužadinimas. Kai bent vienas srautas ateis aukštesnis procesas bus sužadinamas.

Išskaidymas abiejuose notacijose yra panašus tik atrodo kiek kitaip.



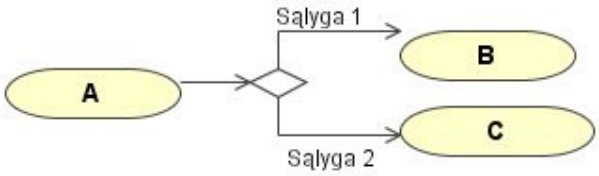
UML 2.0 lygiagretaus išskaidymo pavyzdys

1.4.3. Sprendimo taškas (*Exclusive choise*)

Srautas einantis per sprendimų tašką išskaidomas į kelis alternatyvius srautus, bet kiekvienas iš alternatyvių srautų bus tęsiamas.

BPMN:

Kai reikalinga srautų kontrolė BPMN naudojamas sprendimų taškas. Atėjus srautui, tikrinama, kuri sąlyga yra išpildoma. Ta šaka, kurioje sąlyga yra išpildoma tęsiamas srautas. Tarp visų alternatyvių srautų gali būti tik vienas kuris užbaigs srautų eigą

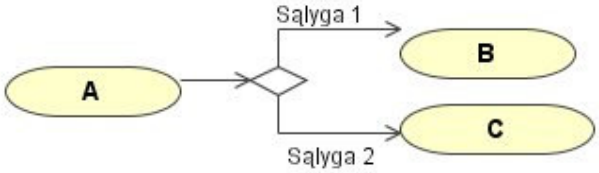


Sprendimų taškas veiklos procesų diagramose

UML:

UML veiklos diagramose alternatyviems keliams kurti naudojamas sprendimo taškas. Sprendimų taške laukiamas srautas. Kai jis atvyksta įvertinamos srauto eigos sąlygos. Srautas tęsiamas ta linkme, kurios sąlyga yra išpildoma.

Taigi, abiejuose notacijose naudojamas sprendimų taškas. Jis net vaizduojamas taip pat, baltu rombu.



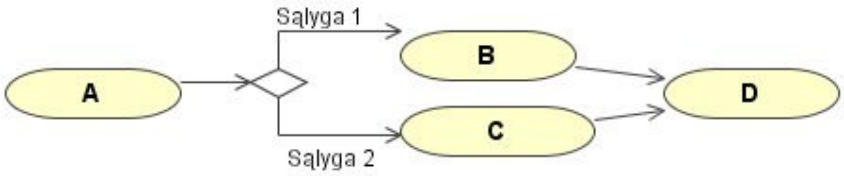
Sprendimų taškas veiklos diagramose

1.4.4. Suliejimo taškas (*Simple merge*)

Suliejimo taškas – tai taškas, kuris sujungia du ar daugiau alternatyvių srautų į vieną srautą.

BPMN:

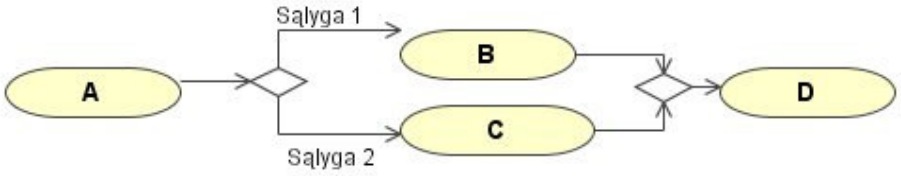
BPMN notacijoje sprendimų srautui sulieti naudojamas suliejimo taškas. Suliejimo taškas sinchronizuoja tik vieno srauto išėjimą iš suliejimo taško. Kai tas srautas ateina jis nedelsiant tęsiamas toliau. Suliejimo taškas gali būti naudojamas ir alternatyviems srautams sulieti. Jei ateis tik vienas srautas, jis bus tęsiamas. Jei ateis daugiau srautų suliejimo taškas veiks kaip diskriminatorius.



Suliejimų taškas veiklos procesų diagramose

UML:

Suliejimo taškas UML diagramose atrodo tai pat kaip sprendimų taškas. Kai srautas ateina į suliejimo tašką jis siunčiamas toliau. Jei ateina keli srautai jie praleidžiami. Keleto srautų praleidimas nėra gerai apgalvotas teisingas veiksmas.



Suliejimų taškas veiklos diagramose

1.4.5. Daugialypis sprendimo taškas (*Multiple choise*)

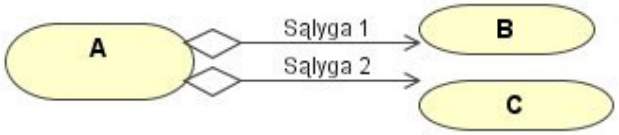
Daugialypis sprendimo taškas nuo išplėtimų taško skiriasi tuo, kad galima pasirinkti ne vieną, o nulį ir daugiau išeinančių alternatyvių srautų. Daugialypis taškas suteikia galimybę nesirinkti nė vienos alternatyvos.

BPMN:

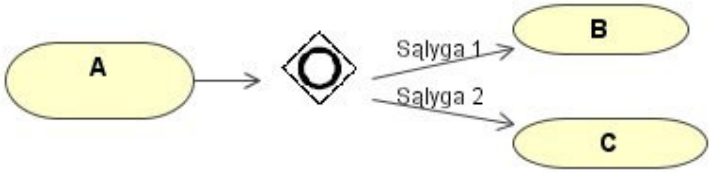
BPMN notacijoje yra du būdai daugialypiam pasirinkimui vaizduoti:

Pirmas būdas kai išeinančių srautų sąlygos rašomos ant pačių srautų. Tuomet pats sprendimų taškas nevaizduojamas, bet vaizduojami maži rombai išeinančio srauto pradžioje. Bent viena iš sąlygų turi būti tenkinama. Tiek srautų bus tęsiama, kiek sąlygų bus išpildyta.

Antras būdas apimančio sprendimų taško naudojimas. Apimantis sprendimų taškas (veikiantis ARBA principu) sužadina tuos srautus, kurių sąlygos yra tenkinamos.



Pirmas būdas veiklos procesų diagramose daugialypiam sprendimo taškui atvaizduoti

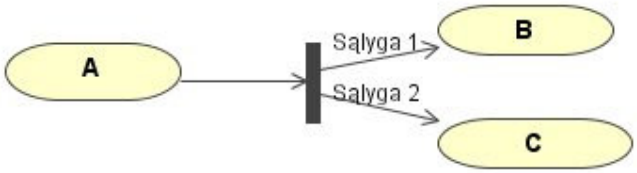


Antras būdas veiklos procesų diagramose daugialypiam sprendimo takui atvaizduoti

UML:

UML diagramose daugialypiam pasirinkimui vaizduoti yra naudojamas išsišakojimų taškas. Šis taškas priėmęs ateinantį srautą išskaido į tiek lygiagrečių kelių, kiek sąlygų yra tenkinama.

Esminis skirtumas tarp UML ir BPMN yra tas, kad BPMN naudojama sprendimų taškų įvairovė, o UML išsišakojimai, kur srautai yra išskaidomi į lygiagrečius srautus. Tai padaro procesą sudėtingesnį.



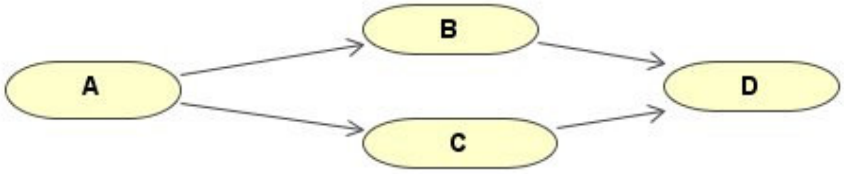
Būdas veiklos diagramose daugialypiam pasirinkimui atvaizduoti

1.4.6. Suliejimo taškas (*Multiple merge*)

Daugelio ateinančių srautų suliejimas be jokios kontrolės. Kiek srautų ateis į veiklą tiek bus tęsiama.

BPMN:

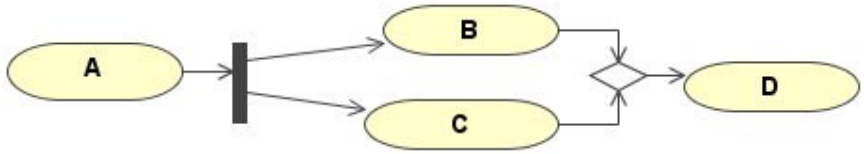
Daugeliui srautų sulieti sprendimų taškas nenaudojamas. BPMN leidžia keliems srautams ateiti į vieną veiklą. Jeigu keli srautai ateina į vieną veiklą tuomet kiekvienam iš jų yra sukuriamas atskiras atvejis.



Daugelio srautų palaikymas veiklos proceso diagramose

UML:

UML diagramose daugeliui srautų sinchronizuoti yra naudojamas suliejimų taškas. Kai tik srautas ateina jis iškart tęsiamas toliau. Tai tik grafinis suliejimas. Suliejimų taškas jokių praeinančių srautų nesustabdys. Suliejimų taškas bus naudojamas daug kartų, tiek kiek bus ateinančių srautų.



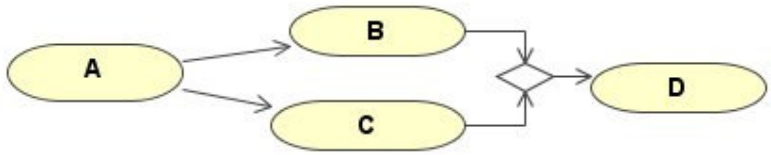
Daugelio srautų palaikymas veiklos diagramose

1.4.7. Diskriminatorius (*Discriminator*)

Diskriminatorius priima ateinančius srautus. Vienas iš ateinančių srautų praleidžiamas, kiti srautai diskriminuojami.

BPMN:

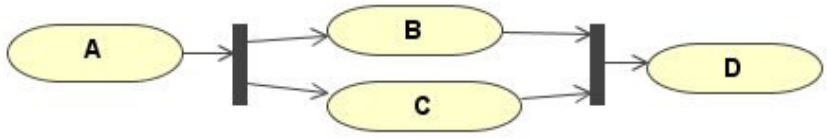
BPMN yra naudojamas eliminuojantis taškas. Šis eliminavimo taškas paleidžia pirmą srautą, o visus kitus uždraudžia.



Diskrimnatorius veiklos proceso diagramose

UML:

UML diagramose naudojamas sujungimas. Sujungimui užduodama sąlyga. Jis laukia ateinančių srautų. Kai pirmas srautas ateina į sujungimą jis siunčiamas toliau, visi kiti srautai sustabdomi.



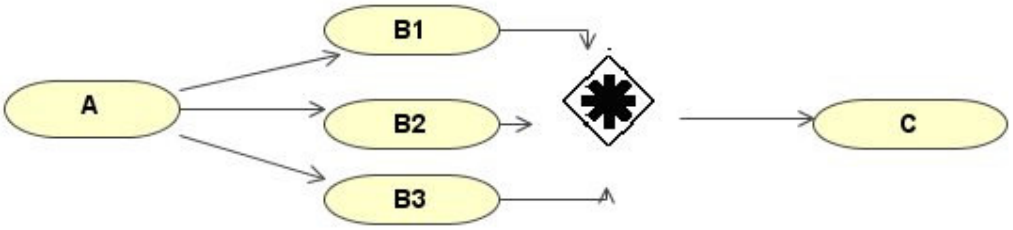
Diskriminatorius veiklos diagramose

1.4.8. Sudėtinis sprendimo taškas (*N out of M join*)

Sudėtinis sprendimų taškas apibrėžia kiek ateinančių srautų gali būti tęsiama

BPMN:

Sudėtinis sprendimų taškas BPMN naudojamas apspręsti kiek ateinančiu srautų gali būti tęsiami. Likę srautai užblokuojami sprendimų taške.

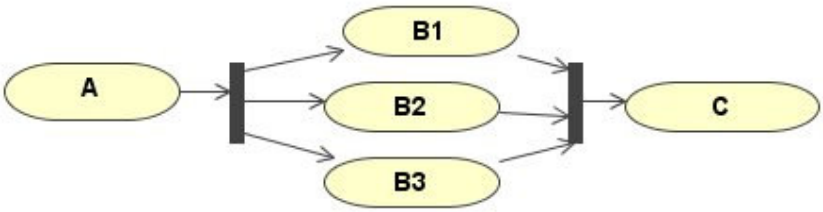


Sudėtinis sprendimų taškas veiklos proceso diagramose

UML:

Veiklos diagramose yra naudojamas sujungimas, kuris jungia ateinančius lygiagrečius srautus. Sujungimų taškui gali būti priskiriama sąlyga, kurioje nurodoma, kiek ateinančių srautų gali būti tęsiami. Kiti srautai užblokuojami sujungimo taške.

Iš esmės šie veiksmai yra panašūs tiek UML tiek BPMN.



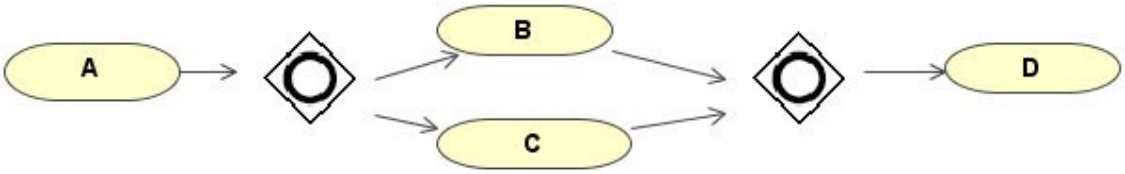
Sudėtinis sprendimų taškas veiklos diagramose

1.4.9. Sinchronizuojantis sujungimo taškas (*Synchronizing merge*)

Šis sujungimas nustato kiek srautų galės praeiti. Tiek srautų kiek nurodoma yra praleidžiami, o kiti uždraudžiami.

BPMN:

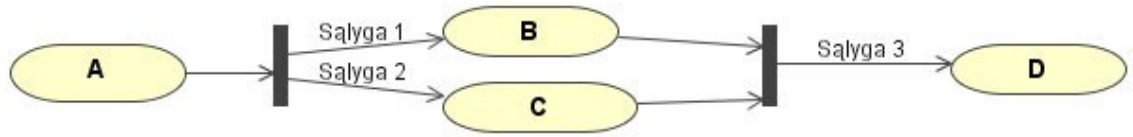
BPMN naudoja apimantį sprendimų tašką, kad sujungtų aukštesnius kelius sukurtus prieš tai einančių apimančių sprendimų taškų



Sinchronizuojantis sujungimas veiklos procesų diagramose

UML:

Srautų kontrolei UML naudoja sujungimus su sąlygomis. Srautai turi atvykti prieš tai kol įvykdoma sąlyga apribojanti srautų tolesnę eigą.



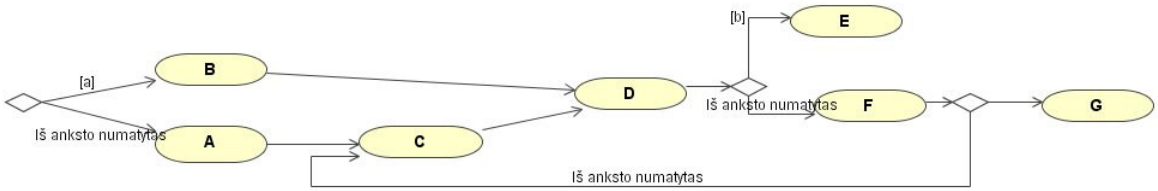
Sinchronizuojantis sujungimas veiklos diagramose

1.4.10. Ciklas (*Cycles*)

Ciklai – nepriklausomų proceso kelių atkartojimas. Ciklų sudarymas leidžia ne vieną įėjimą ir išėjimą.

BPMN:

BPMN notacijoje galima sukurti pasirenkamą ciklą, seką prijungiant prie prieš srautą einančių veiklų.

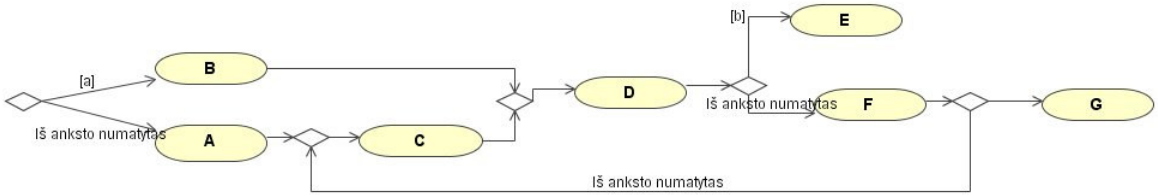


Ciklai veiklos procesų diagramose

UML:

UML 2.0 yra galimybė sukurti pasirenkamą ciklą, kontrolės taškus jungiant prie prieš srautą einančių veiklų.

Abiejuose diagramose ciklų sudarymas yra panašus. Tik tiek, kad UML diagramose ciklai yra jungiami per kontrolės taškus, o BPMN diagramose jokių sprendimo taškų nereikia, nes ir taip galima daugelio srautų atėjimo galimybė.



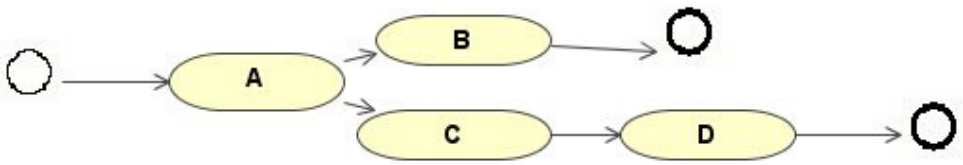
Ciklai veiklos diagramose

1.4.11. Pabaigos taškas (*Implicit termination*)

Pabaigos taškai užbaigia procesą kai jie yra pasiekiami.

BPMN:

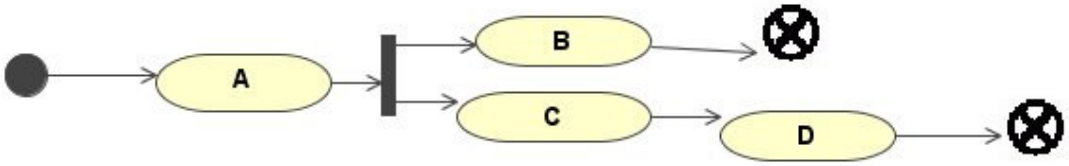
BPMN yra 7 skirtingi pabaigos taškų variantai. Visi išskyrus vieną reiškia besąlygišką srauto kelio nutraukimą. Kiekvienas pabaigos taškas turi vidinį markerį, kuris ir nustato kelio nutraukimo priežastį. Pabaigos įvykis nustatęs, kad kažkokia veikla neprasidėjo ar dar tęsiasi nutrauks procesą.



Pabaigos taškai veiklos proceso diagramose

UML:

Veiklos diagramose pabaigos įvykis parodo, kad tam tikras įvykių srautas pasibaigė. Pabaigos taškas nutrauks procesą net ir tuo atveju, jei kai kurios veiklos neprasidėjusios ar nepasibaigusios.



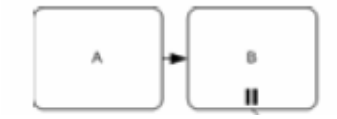
Pabaigos taškai veiklos diagramose

1.4.12. Egzemplioriai (*Multiple instances*)

Šis elementas apibrėžia srautus galinčius praeiti pro veiklą bei išeinančius srautus.

BPMN:

Veiklos procesų diagramose yra elementas, kuris palaiko daug egzempliorių. Tai ciklo veiklos atributas, suteikiantis galimybę sukurti keletą atskirų lygiagrečių atvejų. Galima apibrėžti pastovų skaičių veiklų.

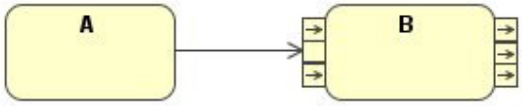


Daugelio egzempliorių palaikymas veiklos proceso diagramose

UML:

Veiklos diagramose daugeliui egzempliorių palaikyti naudojami išplėtimo taškai, atvaizduoti aplink veiklą. Kiek išplėtimo taškų bus, tiek lygiagrečių veiklų bus apibrėžta.

Daugelio egzempliorių palaikymas iš esmės panašus, tik skiriasi atvaizdavimas.



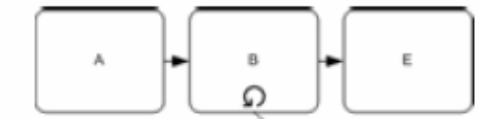
Daugelio egzempliorių palaikymas veiklos diagramose

1.4.13. Egzemplioriai su prioritetu (*Multiple instances with a priori runtime knowledge*)

Šis elementas apibrėžia kiek srautų gali praeiti pro veiklą ir kiek išeinančių srautų bus sukurta. Praleidžiami pirmieji atėję srautai. Kiti uždraudžiami.

BPMN:

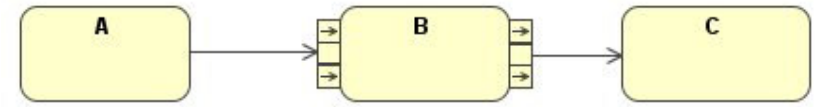
Daugeliui veiklos egzempliorių palaikyti yra naudojamas veiklos ciklo atributas. Ciklo veiklos elementui gali būti užduota sąlyga, kuri apibrėžtų kiek egzempliorių gali būti kuriama. Taip pat šitai elgsenai apibrėžti gali būti naudojamas sprendimų taškas.



Daugelio egzempliorių su prioritetu palaikymas veiklos proceso diagramose

UML:

Veiklos diagramose daugeliui egzempliorių palaikyti naudojami išplėtimo taškai, atvaizduoti aplink veiklą. Kiek išplėtimo taškų bus, tiek lygiagrečių veiklų bus apibrėžta. Įėjimo elementai apibrėžia kiek proceso eigoje bus išėjimų. Šitai elgsenai apibrėžti gali būti naudojamas ir sprendimų taškas.



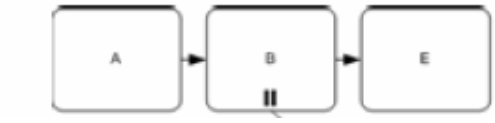
Daugelio egzempliorių su prioritetu palaikymas veiklos diagramose

1.4.14. Egzemplioriai be prioriteto (*Multiple instances whith no a priori runtime knowledge*)

Šis elementas apibrėžia kiek srautų gali praeiti pro veiklą ir kiek išeinančių srautų bus sukurta.

BPMN:

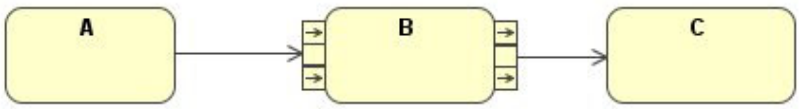
Daugeliui veiklos egzempliorių palaikyti yra naudojamas veiklos ciklo atributas. Ciklo veiklos elementui gali būti užduota sąlyga, kuri apibrėžtų kiek egzempliorių dinamiškai gali būti kuriama. Daugelio egzempliorių sąlyga turi užtikrinti, kad visos proceso kopijos bus užbaigtos iki tol, kol procesas baigsis.



Daugelio egzempliorių be prioriteto palaikymas veiklos proceso diagramose

UML:

Veiklos diagramose daugeliui egzempliorių palaikyti naudojami išplėtimo taškai, atvaizduoti aplink veiklą. Kiek išplėtimo taškų bus, tiek lygiagrečių veiklų bus apibrėžta. Įėjimo elementai apibrėžia kiek dinamiškai proceso eigoje bus išėjimų. Visi išplėtimo taškai turi būti sinchronizuojami kol procesas tęsis.



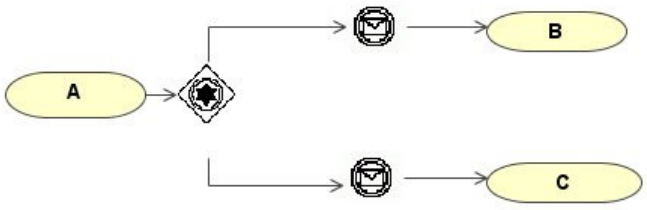
Daugelio egzempliorių be prioriteto palaikymas veiklos diagramose

1.4.15. Atidėtas pasirinkimo taškas (*Deffered choice*)

Atidėtas pasirinkimų taškas atsiranda proceso eigoje. Šis lygiagretus išsišakojimas priklauso nuo įvykių, atsirandančių vykdymo metu.

BPMN:

Veiklos proceso diagramose naudojamas išimtinis duomenimis grįstas sprendimo taškas. Šis sprendimas ypatingas tuo, kad lygiagretus išsišakojimas grįstas įvykiais atsirandančiais procese. Šis taškas atlieka papildomą sekų kontrolę. Po šio sprendimų taško seka tarpiniai įvykiai. Kai įvykis atvyksta tikrinama, kuris kelias tenkinamas, tuo keliu srautas siunčiamas toliau. Kiti įvykiai nepraleidžiami, jie gali būti naudojami kaip laikmačiai.

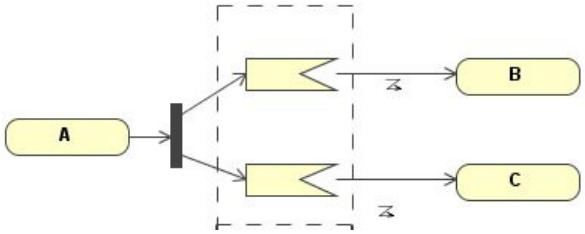


Atidėtas pasirinkimų taškas veiklos proceso diagramose

UML:

Veiklos diagramose tai realizuojama naudojant išsišakojimus ir pertraukimų sritis. Išsišakojimas suskaido kelius į tiek lygiagrečių kelių kiek yra skirtingų pasirinkimų. Lygiagretūs srautai įtraukiami į pertraukimų sritį. Kai ateina signalas jei tenkinama sąlyga tas srautas yra tęsiamas, kiti srautai sunaikinami pertraukimų srities.

BPMN notacijoje suteikiama sprendimo galimybė, o UML visi keliai, išskyrus pirmą atėjusį, sunaikinami.



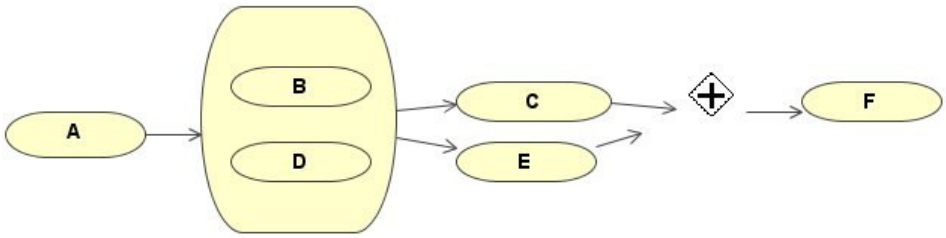
Atidėtas pasirinkimų taškas veiklos diagramose

1.4.16. Dinaminis pasirinkimo procesas (*Interleaved parallel routing*)

Dinaminio pasirinkimo procesas – procesų, kurie gali vykti bet kuriuo metu, rinkinys.

BPMN:

Veiklos procesų diagramose yra galimas dinaminio pasirinkimo procesas. Kiekvienas procesas esantis rinkinio viduje turi sąlyga, kuri turi būti išpildyta, kad procesas vyktų. Specialus procesas turi atlikti ir sekos kontrolę, kad vienu metu vyktų tik vienas specialus procesas.

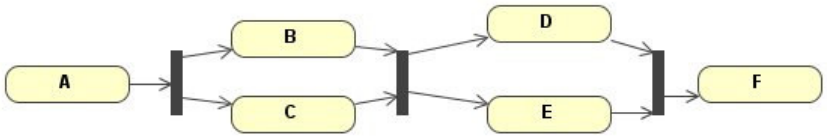


Dinaminio pasirinkimo proceso vaizdavimas veiklos proceso diagramose

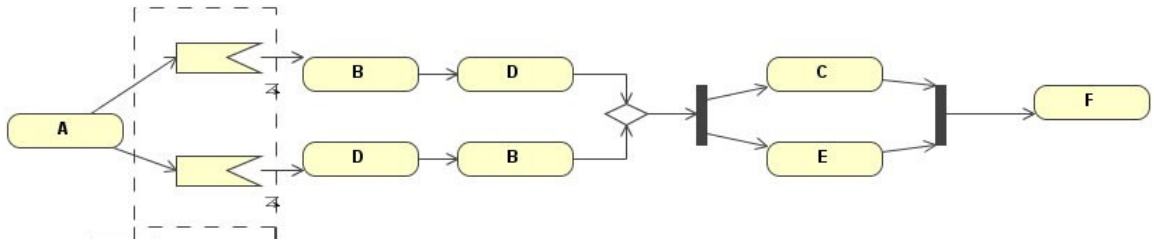
UML:

Veiklos diagramose tokio dinaminio pasirinkimo procesas nėra. Tokį procesą realizuoti galima įvairiais būdais, kaip, pavyzdžiui, specialius įvykius patalpinus tarp išsišakojimo ir sujungimo. Bet kiekvienam procesui reikia tam tikro apribojimo. Taip pat šie procesai vaizduojami kaip lygiagretūs nors negali vykti vienu metu. Sunku atvaizduoti šią sąlygą.

UML veiklos diagramose galimas šio elemento atvaizdavimas ir pertraukimų sritimis ir kopijų kūrimu. Bet diagramos sukurtos tokiu būdu tampa labai sudėtingos ir sunkiai suprantamos.



Pirmas būdas dinaminio pasirinkimo proceso realizavimo veiklos diagramose



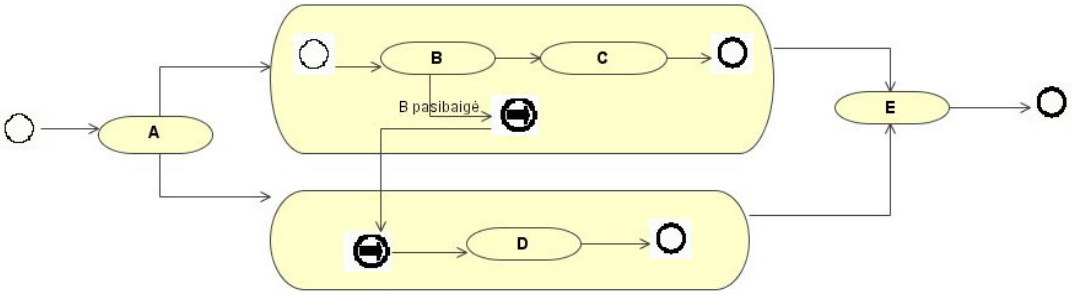
Antras būdas dinaminio pasirinkimo proceso realizavimo veiklos diagramose

1.4.17. Sąsaja (*Milestone*)

Kai normali seka negalima (vienas įvykis gali prasidėti tik kai baigiasi kitas įvykis), tuomet naudojama – sąsaja.

BPMN:

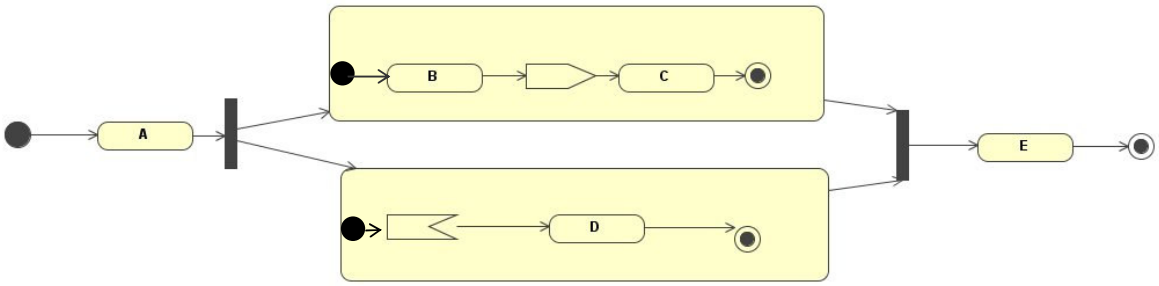
Kai normali seka negalima (vienas įvykis gali prasidėti tik kai baigiasi kitas įvykis), tuomet naudojamas elementas – sąsaja.



Sąsaja veiklos proceso diagramose

UML:

Veiklos diagramose tokiai sekai atvaizduoti yra naudojami: siuntėjo ir gavėjo signalai.



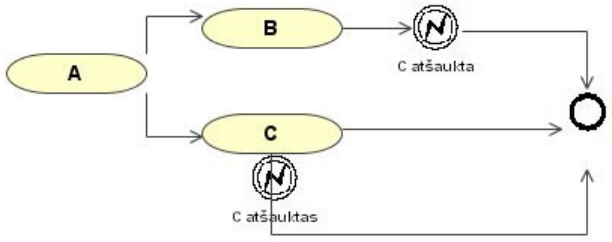
Sąsaja veiklos diagramose

1.4.18. Veiklos nutraukimas (*Cancel activity*)

Veiklos nutraukimas esant tam tikrai nutraukimo sąlygai.

BPMN:

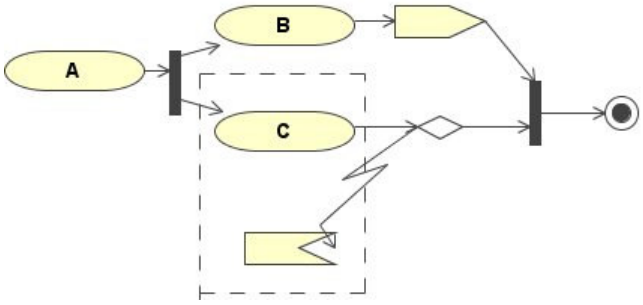
Veiklos proceso diagramose išimtys realizuojamos tarpinėmis veiklomis. Jeigu atsiras trigeris jau prasidėjus veiklai, tai ji bus nutraukiama ir sustabdomas tolimesnis sekos srautas.



Veiklos nutraukimo vaizdavimas veiklos proceso diagramose

UML:

Veiklos diagramose išimtys realizuojamos pertraukimų sritimis, į kurias įeina viena ar daugiau veiklų. Jei ateis signalas, kuris sužadins pertraukimų sritį, tuomet visos veiklos esančios srityje bus nutrauktos.

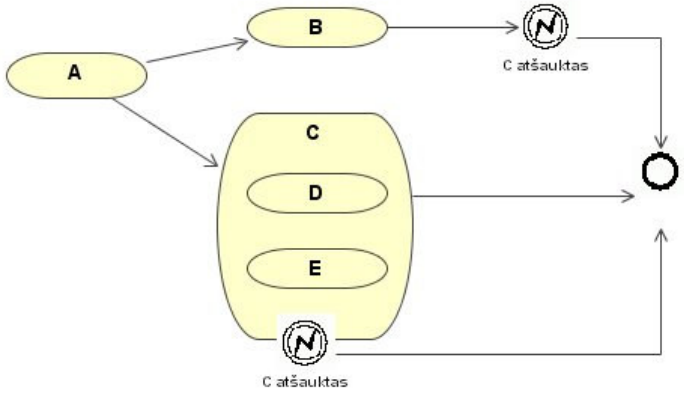


Veiklos nutraukimo vaizdavimas veiklos proceso diagramose

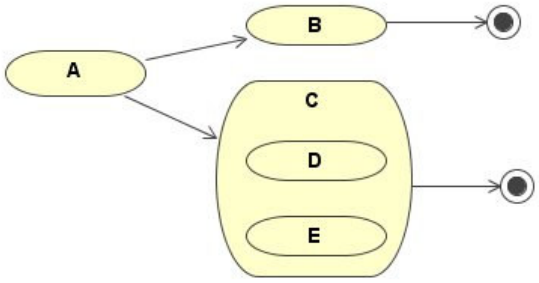
1.4.19. Įvykio nutraukimas (*Cancel case*)

BPMN:

Išimčių realizavimas tarpiniais nutraukimo įvykiais labai panašus į veiklų nutraukimus.



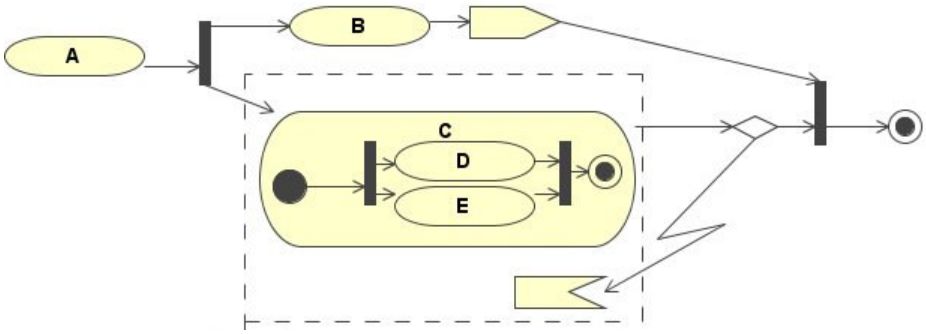
Nutraukimo įvykis veiklos proceso diagramose



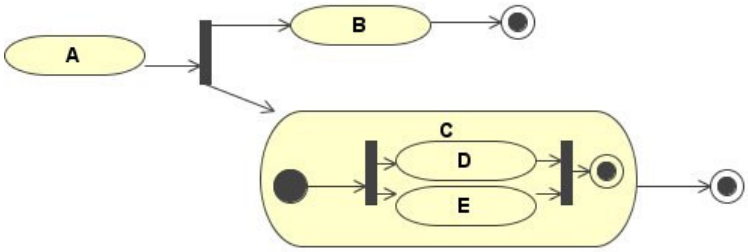
Visiškas proceso nutraukimas veiklos proceso diagramose

UML:

Išimčių realizavimas tarpiniais nutraukimo įvykiais labai panašus į veiklų nutraukimus.



Nutraukimo įvykis veiklos diagramose



Visiškas proceso nutraukimas veiklos diagramose

1.4.20. Dokumentai, naudojami proceso eigoje

BPMN:

Veiklos proceso diagramose dokumentai naudojami kaip duomenų objektai – jie susiję su artefaktais. Duomenų objektai neturi konkretaus poveikio procesui, jie tik aprašo ką procesas veikia.

UML:

Veiklos diagramose dokumentai vaizduojami kaip objektai, kurie objektų srautais gali būti pernešami. Veiklos objektai dalyvauja veiklos procese ir turi įtakos jo eigai.

Svarbus skirtumas tas, kad BPMN veiklos proceso diagramos galima tik paminėti kokie dokumentai yra naudojami. Jei tik užrašomi ir poveikio procesui nedaro. Tuo tarpu UML diagramose yra žymiai didesnės galimybės dokumentams vaizduoti.

1.4.21. Pranešimo trigeriai (*Message triggers*)

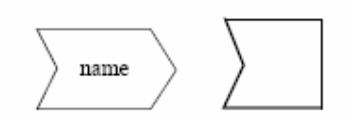
BPMN:

Veiklos proceso diagramose pranešimams realizuoti naudojami pranešimo trigeriai. Pranešimo trigeriai gali būti pradžios, tarpiniai ir pabaigos. Pradžios trigeris paleidžia proceso pradžią, tarpinis trigeris tęsia procesą, o pabaigos trigeris veikia jei buvo tarpinis trigeris, ir jis praneša apie proceso pabaigą.

Pranešimo trigeriai

UML:

UML veiklos diagramose pranešimai siunčiami siutimo signalais, o priimami priėmimo signalais.

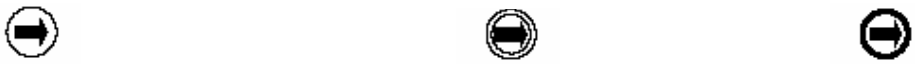


Pranešimo signalai

1.4.22. Sąsajos trigeriai (*Link triggers*)

BPMN:

Veiklos proceso diagramose sąsaja realizuojama per sąsajos trigerius. Sąsajos trigeriai gali būti pradžios, tarpiniai ir pabaigos. Sąsajos trigeriai naudojami jei procesas gali prasidėti tik pasibaigus kitam procesui, ar pasiekus tarpinę būseną ir kt.



Pradžios sąsajos trigeris Tarpinis sąsajos trigeris Pabaigos sąsajos trigeris

UML:

UML 2.0 veiklos diagramose veiklos susietos srautais. Kokia tvarka išdėstytos veiklos ir srautai, tokia tvarka vyksta ir procesas. Seka tęsiasi nuo vienos sekos iki kitos. UML 2.0 standarto diagramose nėra įtrauktos jokios sąlygos, jokios valdymo kontrolės.

1.4.23. Kompensavimo trigeriai (*Compensation triggers*)

BPMN:

Kompensavimas (išlyginimas). Kompensavimo trigeris gali būti tik tarpinis arba pabaigos. Šis trigeris naudojamas procesui gražinti.

Kompensavimo trigeriai

UML:

UML veiklos diagramose kompensavimas gali būti atvaizduojamas tik grįžtamaisiais ryšiais. Specialaus elemento kompensavimui nėra.

1.5. Verslo proceso vykdomosios kalbos (*BPEL*) analizė

BPEL (*Buisiness Process Execution Language*) yra XML pagrįsta kalba, palaikanti XPath 1.0 funkcijas. Ši kalba skirta veiklos procesų formaliai specifikacijai ir verslo procesų sąveikos protokolams. BPEL susieja modelius, palengvina procesų integravimą ir palaiko verslo transakcijas tarp verslo dalyvių. Naudojant BPEL formaliai aprašomas verslo procesas, kad jis kiekviename žingsnyje būtų prieinamas programuotojams. Kaip, pavyzdžiui, užsakymo procesas. Iš ko susideda užsakymas, kokie produktai užsakomi ir kokios šio proceso eigoje gali atsirasti išimtys, klaidos. Visa tai aprašoma procesų sąveikos protokole. BPEL notacijoje paliekama didelė atvaizdavimo laisvė. Atvaizduojant suprojektuotus modelius į BPEL reikalingi realizavimui duomenys gali būti imami iš duomenų modelio sudaryto, pavyzdžiui, UML 2.0 kalba. Taigi, remiantis šia notacija galima išplėsti grafinę proceso apibrėžimo sąsają, kurioje proceso taisyklės būtų susiejamos su dalykinės srities modelio elementais.

1.6. CASE įrankių analizė

UML 2.0 standartą stengiamasi įdiegti daugelyje įrankių, iš kurių galima paminėti Enterprise Architect Corporate v4.0, MagicDraw, IBM Rational Software Architect ir Modeller. BPMN realizavimui tokio įrankio dar nėra. UML 2.0 realizacijos yra labai naujos, natūralu, kad jose nemaža klaidų ir ne visos savybės įgyvendintos. Modeliuoti buvo naudojami MagicDraw 7.0, Enterprise Architect Corporate v4.0 bei MagicDraw 9.0 CASE įrankiai.

MagicDraw 7.2 yra vaizdinė UML modeliavimo priemonė ir CASE įrankis. Šis įrankis skirtas verslo analitikams, programinės įrangos analitikams, programuotojams, inžinieriams ir dokumentacijų kūrėjams. Šis dinamiškas ir įvairialypis vystymo įrankis palengvina orientuotų į objektus sistemų ir duomenų bazių analizę ir projektavimą. Šiame įrankyje galimas kodo (*Java, C#,C++,WSDL, XML Schema, CORBA IDL*) generavimas ir atvirkštinė inžinerija. MagicDraw 7.2 palaiko UML 1.4. Suteikia galimybę modeliuoti visus šio standarto diagramų tipus bei elementus. Sumodeliuotas diagramas galima eksportuoti į XMI, išsaugoti kaip paveikslus. Šis įrankis patogus e. verslo procesams modeliuoti pagal naują sudarytą metodiką, nes įrankyje esančius standartinius stereotipus galima papildyti naujais reikiamais stereotipais, importuojant ir grafinį stereotipo vaizdą. MagicDraw 9.0 tai naujausia MagicDraw versija, kuri jau palaiko UML 2.0 standartą.

Enterprise Architect Corporate v4.0 naujas paketas palaikantis naują standartą UML 2.0. UML 2.0 suteikia galimybę taip suprojektuoti sistemą, kad ji būtų labai artima realizuojamai sistemai. Šis

standartas papildytas elementais, skirtais projektuoti verslo modelius (duomenų saugykla, daugybė taškų tipų, veiklos suskaidymas, pertraukimas ir kt.). Naudojant šį standartą ir notaciją, galima suprojektuoti net sudėtingiausias sistemas nuo pradinio lygmens bei užrašyti įvairias verslo taisykles. Šiuo programiniu paketu galima ne tik modeliuoti, bet ir generuoti kodą bei atlikti atvirkštinę inžineriją, importuoti duomenų bazes iš ODBC duomenų šaltinių, importuoti ir eksportuoti modelius iš ir į XMI.

3 lentelė. CASE įrankių palyginimas

Požymiai	Enterprise Architect Corporate v4.0	MagicDraw 9.0
Naujo standarto UML 2.0 palaikymas	+	+ –
Kodo generavimas	+	+
Atvirkštinė inžinerija	+	+
Eksportavimas į XMI	+	+
Naujų stereotipų kūrimas	–	+
Naujų stereotipų grafinio vaizdo palaikymas	–	+
Laiko apribojimų palaikymas	+	+ –
Galimybė specifikuoti proceso vykdymo taisykles CASE įrankiuose	+ –	+ –
Dalykinės srities galimybės	+ –	+ –

„+“ – palaiko; „–“ – nepalaiko; „+ –“ – iš dalies palaiko.

Abu CASE įrankiai palaiko naują standartą UML 2.0, tik MagicDraw pastebimi kai kurių elementų ir diagramų tipų trūkumai. Abu paketai suteikia galimybę generuoti kodą, atlikti atvirkštinę inžineriją, eksportuoti į vykdomąją kalbą. Šiuose CASE įrankiuose galima specifikuoti proceso vykdymo taisykles, bet tai padaryti gana sunku, nes specialių priemonių tam atlikti nėra. Šiame darbe pasirinktas MagicDraw CASE įrankis, kadangi jis gerai palaiko tradicinius projektavimo procesus; turi duomenų bazių, XML, WSDL schemų generavimo galimybes; OCL specifikavimo ir sintaksės tikrinimo priemones; plėtimo API; leidžia lengvai įvesti naujus grafinius stereotipus. UML 2.0 notacijos realizacija MagicDraw dar turi trūkumų, tačiau tikimasi, kad jie netrukus bus pašalinti.

1.7. Analizės išvados

Ieškant vartotojui tinkamiausios e. verslo modeliavimo notacijos išnagrinėtos plačiausiai naudojama universali modeliavimo kalba – UML 2.0, specializuota verslo procesų kalba – BPMN ir vykdomoji verslo procesų kalba – BPEL. Išanalizuota jų paskirtis; palyginti elementai, pateiktas jų grafinis vaizdas ir pavyzdžiai. Atliktas UML 2.0 ir BPMN palyginimas pagal 23 įvairius požymius, atrasti šių dviejų notacijų privalumai bei trūkumai:

- BPMN yra specializuota verslo procesams vaizduoti ir turi išsamią tam skirtą notaciją
- UML 2.0 yra universali kalba, ir verslo procesų modeliavimas yra tik viena jos naudojimo krypčių. Verslui modeliuoti UML 2.0 turi panašias galimybes, kaip ir BPMN, tačiau UML 2.0 elementų aibė mažesnė

- Pagrindinis BPMN trūkumas – nėra priemonių dalykinei sričiai vaizduoti. Dokumentai gali būti parodyti tik kaip pastabos.
- UML veiklos diagramose dokumentai vaizduojami kaip objektai, kurie dalyvauja veiklos procese ir turi įtakos jo eigai. Greta veiklos diagramų yra išsamus priemonių rinkinys informacinei sistemai projektuoti.
- Abiejose kalbose galima aprašyti verslo taisykles, tačiau jų apdorojimo priemonių ar rekomendacijų nėra.

Kadangi kiekviena iš analizuotų kabų turi savų privalumų ir trūkumų, buvo pasiūlyta idėja praplėsti UML 2.0 specializuotos verslo proceso kalbos galimybėmis ir sudaryti universalią modeliavimo metodiką, taip susiejant dalykinės srities modelį su veiklos proceso modeliu. Kadangi modeliuojant svarbu sukurti ir e. verslo vykdymo specifikaciją aprašytas ir išplėstosios notacijos atvaizdavimas į veiklos procesų vykdomąją kalbą BPEL

Atlikta CASE įrankių pritaikytų grafiniam modeliavimui analizė. Buvo pasirinktas lanksčiausią vartotojo sąsaja turintis įrankis MagicDraw 9.0. Pasiūlyta idėja pasirinktą CASE įrankį papildyti, pritaikant jį universaliai metodikai.

2. VERSLO PROCESŲ MODELIAVIMO IR SPECIFIKAVIMO METODAS UML, BPMN IR BPEL PAGRINDU

Pagrindinė modeliavimo problema yra ta, kad yra daug verslo modeliavimo kalbų, kurios akcentuoja skirtingas verslo modelių savybes ir nėra nė vienos, kuri tenkintų visas savybes. Daugelis kalbų yra tekstinio pavidalo. Sunku pasirinkti tinkamiausią modeliavimo kalbą, nes trūksta reikiamos modeliavimo medžiagos bei universalios metodikos. Nuolat keičiasi reikalavimai verslui modeliuoti. Modeliavimo metodika turėtų atitikti šiuolaikinius e. verslo modeliavimo reikalavimus.

2.1. Universalios modeliavimo metodikos tikslai ir uždaviniai

Darbo tikslas – atlikti verslo procesų modeliavimo kalbų analizę ir sudaryti verslo analitikui pritaikytą verslo procesų specifikavimo metodiką, pagal kurią aprašytą verslo procesą būtų galima transformuoti į e. verslo vykdymo programos aprašą.

E. verslui modeliuoti buvo pasirinktos dvi modeliavimo notacijos UML bei BPMN. Naujausia UML versija 2.0, kurioje išplėstos galimybės verslui modeliuoti. Plačias verslo modeliavimo galimybes palaiko naujas verslo modeliavimo standartas BPMN, kuris palengviną projektavimo kelią nuo projektavimo iki realizavimo, ir projektavimo procesą padaro prieinamą ir aiškų visiems verslo kūrimo dalyviams.

Analizės dalyje buvo išanalizuotos ir palygintos UML 2.0 ir BPMN notacijos pagal įvairius požymius, atrasti šių dviejų notacijų privalumai bei trūkumai. BPMN yra specializuota verslo procesams vaizduoti ir turi išsamią tam skirtą notaciją UML 2.0 yra universali kalba, ir verslo procesų modeliavimas yra tik viena jos naudojimo krypčių. Verslui modeliuoti UML 2.0 turi panašias galimybes, kaip ir BPMN, tačiau UML 2.0 elementų aibė mažesnė. Pagrindinis BPMN trūkumas – nėra priemonių dalykinei sričiai vaizduoti. Dokumentai gali būti parodyti tik kaip pastabos. UML veiklos diagramose dokumentai vaizduojami kaip objektai, kurie dalyvauja veiklos procese ir turi įtakos jo eigai. Greta veiklos diagramų yra išsamus priemonių rinkinys informacinei sistemai projektuoti.

Šiame darbe siūloma:

Sudaryti universalią modeliavimo metodiką, pagal kurią sudarytas bet kuris e. verslo proceso modelis tenkins daugelį e. verslo proceso reikalavimų. Kad tai realizuoti siūloma UML 2.0 papildyti BPMN stereotipais (1 lentelė, 2 lentelė). Apjungus šias dvi notacijas modelis taptu aiškus ir išsamus visiems verslo dalyviams, analitikams, projektuotojams.

Dalykinės srities modelį susieti su veiklos proceso modeliu, kad modelis būtų prieinamas ir programuotojams. Kad tai realizuoti reikėtų išanalizuoti ir aprašyti išplėstosios notacijos atvaizdavimą į BPEL. Atvaizdavimas į BPEL pateiktas vėlesniuose skyriuose.

Pritaikyti šią metodiką UML CASE įrankiui ir eksperimentiškai išbandyti sukuriant realaus e. verslo proceso modelį, parodant naujos modeliavimo metodikos pranašumus.

Sudaryti specifikacijų šablonus natūralia ir OCL kalbomis bei papildyti CASE įrankio veiklos modeliavimo elementų specifikacijas, kurios leistų aprašyti verslo taisyklių naudojamų duomenų modelio elementus. Tokiu būdu sudarytą modelį būtų galima konvertuoti į vykdomąją kalbą.

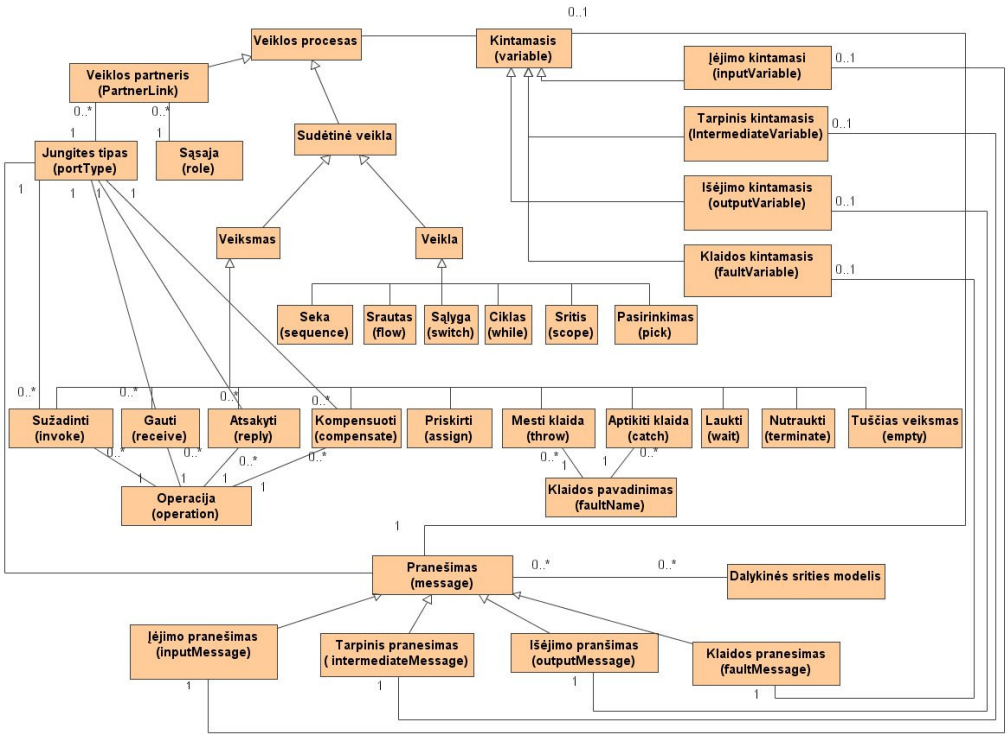
Eksperimentiniam modeliavimui pasirinkta nuotolinio mokymosi sistema. Pagal scenarijų sumodeliuotame trečiame eksperimentiniame modelyje panaudota daugelis BPMN elementų kuriais siūloma papildyti UML 2.0 (1 lentelė, 2 lentelė).

2.2. Verslo modelių atvaizdavimas į vykdomąją (*BPEL*) kalbą

Kadangi galutinis tikslas yra sukurti e. verslo vykdymo programos specifikaciją, buvo išanalizuota plačiausiai naudojama BPEL kalba. Šioje notacijoje paliekama didelė atvaizdavimo laisvė. Tai pat pasiūlyta išplėsti grafinę proceso apibrėžimo sąsają, kurioje proceso taisyklės būtų susiejamos su dalykinės srities modelio elementais. Atvaizduojant suprojektuotus modelius į BPEL reikalingi realizavimui duomenys imami iš duomenų modelio sudaryto UML 2.0, pvz., 3.5 paveikslas.

2.2.1. Papildytas verslo procesų metamodelis

Šiame skyriuje 2.1 paveiksle pateikiamas papildytas veiklos procesų metamodelis, sudarytas apjungiant BPMN, UML 2.0 ir BPEL kalbų savybes. Jis skirtas atvaizdavimo į vykdomąją kalbą procesui palengvinti.



2.1 pav. BPEL metamodelis susietas su dalykinės srities metamodeliu

2.1 paveiksle pateiktas metamodelis buvo sudaromas analizuojant ir papildant anksčiau sudarytus modelius. Metamodelyje vaizduojamas veiklos procesas, kuris susideda iš sudėtinės veiklos, veiklos partnerio bei turi sąsają su kintamųjų aibe. Veiklos procese veikla yra sudėtinė susidedanti iš veiksmo ir įvairių veiklų tipų. Veikla gali būti įvairi: vaizduojama kaip veiklų seka, srautas, sąlyga, sritis, pasirinkimas. Veiksmai priklausomai nuo veiklos proceso elementų gali būti įvairios paskirties: sužadinti, atsakyti, nutraukti ir kt. Kad atvaizduoti kiekvieną veiksmą reikalinga aprašyti tam veiksmui reikalingus tipus ir elementus. Projektavimo etape reikalingi kintamieji susideda iš įvairių tipų pranešimų. Koks pranešimo tipas toks ir kintamojo tipas. Pranešimams duomenys imami iš duomenų modelio. Kintamieji per pranešimus susieja projektuojamą modelį su dalykine sritimi.

2.2.2. Pranešimų, taisyklių, ryšio elementai

Pranešimo (taisyklių, ryšio) trigeris atvaizduojamas į gavimo (*recieve*) elementą Siunčiamas pranešimas sužadina (*invoke*) kažkokį tai įvykį. Jeigu siunčiamas pranešimas yra vienpusis, tai sužadintas įvykis atvaizduojamas tik į sužadinimo (*invoke*) elementą. Jei dvipusis, tuomet siunčiamas atsakas (*reply*) sužadinusiam įvykiui.

Kad atvaizduotume į ankščiau paminėtus elementus, reikia apsirašyti šiuose elementuose naudojamus tipus bei reikalingus elementus. Elementai aprašomi tokia seka:

Apsirašomi visi reikalingi pranešimų tipai. Duomenys pranešimams imami iš UML duomenų modelio (3.4 pav.).

```
<message name="Pranešimo pavadinimas">
    <part name="Pranešimo duomenys" type="xsd:Pranešimo tipas"/>
    ...
    <part name=…/>
</message>
```

Aprašoma jungtis (*port*), per kurią siunčiamas pranešimas. Išėjimo klaidos pranešimų gali ir nebūti, jei pranešimas tik sužadina įvykį:

```
<portType name="Siunčiamo pranešimo jungties pavadinimas">
    <operation name="Operacijos pavadinimas">
    <input message="pos:Įėjimo pranešimo pavadinimas"/>
    <output message="pos:Išėjimo pranešimo pavadinimas"/>
    <fault name="Klaidos pranešimas"/>
              message="pos:Klaidos pranešimo pavadinimas"
    </operation>
</portType>
```

Aprašomas partnerių sąsajos tipas:

```
<plnk:partnerLinkType name="Sąsajos tipo pavadinimas">
   <plnk:role name="Sąsajos pavadinimas">
       <plnk:portType name"Siunčiamo pranešimo jungties pavadinimas">
   </plnk:role>
</plnk:partnerLinkType>
```

Aprašomos sąsajos:

```
<partnerLinks>
 <partnerLink name="Sąsajos pavadinimas">
</partnerLinks>
   partnerLinkType ="Sąsajos tipo pavadinimas"
   myRole="Rolės pavadinimas"
</partnerLinks>
```

Aprašomi siuntimo kintamieji (kiekvienam pranešimui sukuriamas kintamasis):

```
<variables>
    <variable name="Kintamojo vardas"  messageType="lns:Pranešimo
pavadinimas">
</variables>
```

Visi aukščiau paminėti elementai rašomi tiek kartų kiek yra pranešimų, jungčių ir kt. Jie užrašomi vienas paskui kita.

Aprašomas pranešimo siuntimo klaidos elementas:

```
<faultHandlers>
    <catch faultName="Klaidos pranešimo pavadinimas"
        faultVariable="Klaidos kintamojo pavadinimas">
    <reply partnerLink="Sąsajos pavadinimas"
        portType="Siunčiamo pranešimo jungties pavadinimas"
        operation="Operacijos pavadinimas"
        variable="Kintamojo vardas"
        faultName="Klaidos pranešimas"
   </catch>
</faultHandlers>
```

Aprašius visus reikiamus tipus ir elementus užrašomas atvaizdavimas į gavimo (*receive*) elementą, kuris sužadina kitą procesą (*invoke*) , ir jei siuntimas ne vienpusis gauna atsaką (*reply*):

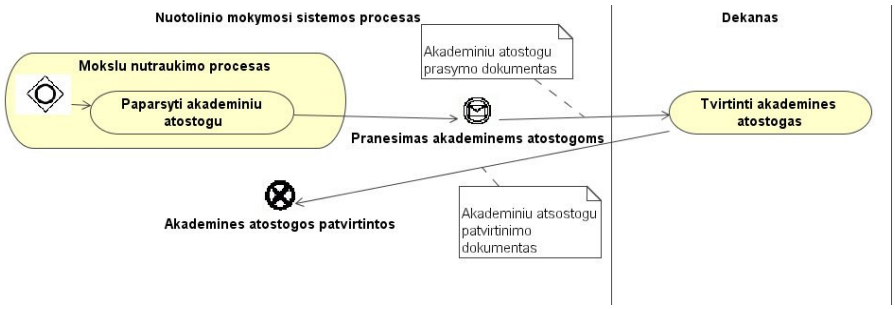
```
<receive partnerLink="Sąsajos pavadinimas"
             portType="Siunčiamo pranešimo jungties pavadinimas"
             operation="Operacijos pavadinimas"
             variable="Kintamojo pavadinimas"
</receive>
<links>
```

```
     <link name="Partnerių sąsajos pavadinimas"/>
</links>
<invoke partnerLink="Sąsajos pavadinimas"
               portType="Gavejo jungties pavadinimas"
               operation="Gavėjo operacija"
               inputVariable="Gavėjo įėjimo duomenys"
               outputVariable="Gavėjo išėjimo duomenys"/>
  <source linkName="Partnerių sąsajos pavadinimas"/>
</invoke>
<reply partnerLink="Sąsajos pavadinimas"
          portType="Siunčiamo pranešimo jungties pavadinimas"
          operation="Operacijos pavadinimas"
          variable="Atsako kintamasis"/>
```

Pranešimas (taisyklė, ryšys) gali būti atvaizduojami į aktyvuotą pranešimą (*onMessage)*, kuris įtraukiamas į įvykį reguliuojantį elementą (*eventHandler*). Aktyvuotas pranešimas (*onMessage*) yra atvaizduojamas analogiškai kaip gavimo (*recieve*) elementas (su tokiais pat atributais).

```
<eventHandlers>
  <onMessage  partnerLink="Sąsajos pavadinimas"
          portType="Siunčiamo pranešimo jungties pavadinimas"
          operation="Operacijos pavadinimas"
          variable="Kintamojo pavadinimas"/>
    veikla
  </onMessage>
</eventHandlers>
```

Veikla gali būti: sužadinimas (*invoke*), klaidos metimas (*throw*) ir kt.



2.2 pav. Pranešimo siuntimo fragmentas iš 3.3 paveikslo modelio

```
<message name="Akademinės atostogos">
    <part name="Vartotojas.Prašymas" type="xsd:dokumentas"/>
    <part name="Vartotojas.vardas" type="xsd:string"/>
    <part name="Vartotojas.pavadė" type="xsd:string"/>
    <part name="Vartotojas.studento_numeris" type="xsd:string"/>
</message>
```

```
<message name=“Pranešimas nepatvirtintas“>
    <part name=”Mokymosi įstaiga.Siuntimo klaida” type=”xsd:string”/>
</message>
<message name=“Akademinės patvirtintos“>
    <part name=”Dekanas.Patvirtinimo dokumentas” type=”xsd:dokumentas”/>
</message>
<portType name=”Prašymo siuntimas”>
    <operation name=”Siųsti prašymą”>
    <input message=“pos: Akademinės atostogas“/>
    <fault name=“Prašymas negalimas“/>
              message=”pos: Pranešimas nepatvirtintas”
    </operation>
</portType>
<portType name=”Prašymo tvirtinimas”>
    <operation name=”Tvirtinti”>
    <input message=“pos: Akademinės atostogas“/>
  </operation>
</portType>
<plnk:partnerLinkType name=“Siuntimas“>
   <plnk:role name=“siuntimo aptarnavimas“>
       <plnk:portType name“Prašymo siuntimas“>
   </plnk:role>
</plnk:partnerLinkType>
<plnk:partnerLinkType name=“Tvirtinimas“>
   <plnk:role name=“patvirtinimo aptarnavimas“>
       <plnk:portType name“Prašymo tvirtinimas“>
   </plnk:role>
</plnk:partnerLinkType>
<partnerLinks>
 <partnerLink name=“Pranešimo siuntimas“>
   myRole=”siuntimo aptarnavimas”
< partnerLinkType =”Siuntimas”>
  myRole=”patvirtinimo aptarnavimas”
</partnerLinks>
<variables>
 <variable name=”Pranešimas apie akademines atostogas”
messageType=”lns:Akademinės atostogos”>
<variable name=”Klaida” messageType=”lns: Pranešimas nepatvirtintas”>
```

```
<variable name=”Akademinės atostogos patvirtintos” messageType=”lns:
Akademinės patvirtintos”>
</variables>
<faultHandlers>
    <catch faultName=“Pranešimas nepatvirtintas“
        faultVariable=“Klaida“>
    <reply partnerLink=“Pranešimo siuntimas“
        portType=”Prašymo siuntimas”
        operation=“Siųsti prašymą”
        variable=“Klaida“
        faultName=“Pranešimas nepatvirtintas“
   </catch>
</faultHandlers
<receive partnerLink=“Pranešimo siuntimas“
             portType=“Prašymo siuntimas“
             operation=“Siųsti prašymą“
             variable=”Pranešimas apie akademines atostogas”
</receive>
<links>
     <link name=”Siųsti–Priimti”/>
</links>
<invoke  name=”Tvortinti akademines atostogas”
              partnerLink=”Siuntimas”
              portType=”Prašymo tvirtinimas”
              operation=”Tvirtinti”
              inputVariable=”Pranešimas apie akademines atostogas”
              outputVariable=”Akademinės atostogos patvirtintos”/>
  <source linkName=”Siųsti_Priimti”/>
</invoke>
```

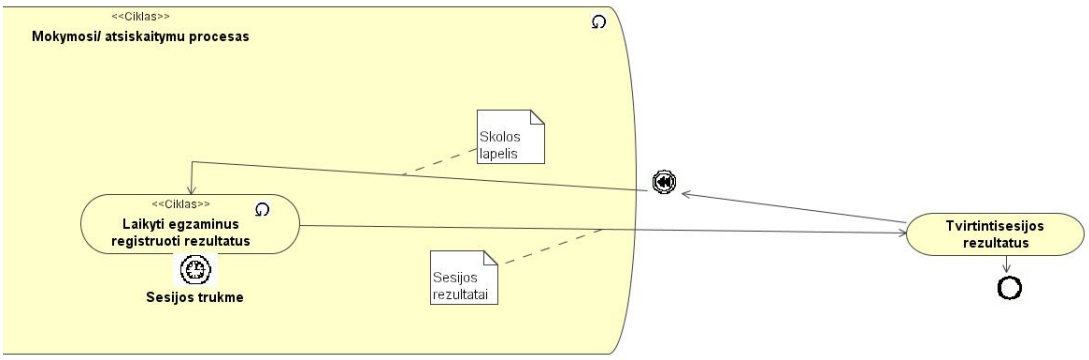
2.2.3. Kompensavimo elementas

Proceso eigoje įvykai gali būti kompensuojami tuomet anksčiau aprašytas pranešimo atvaizdavimo procesas papildomas „*compensationHandler“* elementu, įterpiant jį po „*faultHandlers“* elemento.

```
<compensationHandler>
          veikla
</compensationHandler>
```

Veikla gali būti, pavyzdžiui, kažkokio įvykio sužadinimas (*invoke*):

```
<compensationHandler>
  <invoke partnerLink=”Sąsajos pavadinimas”
              portType=”Sužadinto įvykio jungties pavadinimas”
              operation=”Sužadinto įvykio operacija”
              inputVariable=”Sužadinto įvykio įėjimo duomenys”
             outputVariable=”Sužadinto įvykio išėjimo duomenys”/>
   <source linkName=”Partnerių sąsajos pavadinimas”/>
  </invoke>
</compensationHandler>
```



2.3 pav. Kompensavimo veiksmo fragmentas iš 3.3 paveikslo modelio

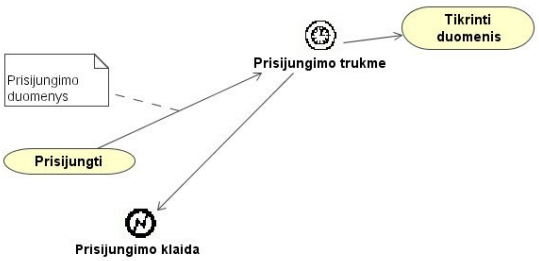
```
<message name=“Nebaigta sesija“>
    <part name=”Dekanas.Skolos lapelis” type=”xsd:Dokumentas”/>
</message>
<message name=“Pabaigta sesija“>
    <part name=”Vartotojas.Sesijos rezultatai” type=”xsd:Dokumentas”/>
</message>
<portType name=”Sesijos patvirtinimas”>
    <operation name=”Rezultatų siuntimas patvirtinimui”>
    <input message=“Pabaigta sesija“/>
    <fault name=“Ne visi egzaminai išlaikyti“/>
              message=”Nebaigta sesija”
    </operation>
</portType>
<plnk:partnerLinkType name=“Siuntimas“>
   <plnk:role name=“Siuntimo aptarnavimas“>
       <plnk:portType name“Sesijos patvirtinimas“>
   </plnk:role>
</plnk:partnerLinkType>
```

```
<partnerLinks>
 <partnerLink name="Rezultatų siuntimas">
</partnerLinks>
   partnerLinkType ="Siuntimas"
</partnerLinks>
<variables>
<variable name="Sesija baigta" messageType=" Pabaigta sesija">
<variable name="Neišlaikytas egzaminas" messageType="Nebaigta sesija">
</variables>
<faultHandlers>
    <catch faultName=" Ne visi egzaminai išlaikyti"
        faultVariable="Neišlaikytas egzaminas">
    <reply partnerLink="Rezultatų siuntimas"
        portType="Sesijos patvirtinimas"
        operation="Rezultatų siuntimas patvirtinimui"
        variable="Pabaigta sesija"
        faultName="Ne visi egzaminai išlaikyti"
   </catch>
</faultHandlers>
<compensationHandler>
  <invoke name="Laikyti egzaminus/registruoti rezultatus"
              partnerLink=" Siuntimas"
              portType="Sesijos patvirtinimas"
              operation="Rezultatų siuntimas patvirtinimui"
              inputVariable="Skolos lapelis"
              outputVariable="Rezultatai"/>
   <source linkName="Rezultatų siuntimas"/>
  </invoke>
</compensationHandler>
```

2.2.4. Klaidos elementas

Klaidos kintamasis atvaizduojamas į klaidos metimo elementą (*throw*), prieš tai aprašius visus kitus reikiamus tipus ir elementus, kaip anksčiau paminėtuose atvaizdavimuose.

```
<throw faultName="Klaidos pavadinimas" faultVariable="Klaidos kintamasis"/>
```



2.4 pav. Klaidos atvaizdavimo fragmentas iš 3.3 paveikslo modelio

Pavyzdžio fragmentas:

```
<message name="Prisijungimas nepavyko">
    <part name="Vartotojas.vardas" type="xsd:string"/>
    <part name="Vartotojas.pavadė" type="xsd:string"/>
    <part name="Vartotojas.studento_numeris" type="xsd:string"/>
</message>
<variables>
    <variable name="Prisijungimo klaida" messageType=" Prisijungimas
nepavyko">
    <variable name="Prisijungimo trukmė" type="integer">
</variables>
<eventHandler>
 <onAlarm for="bpws:getVariableData(Prisijungimo trukmė)='Maxtrukmė'">
       <throw faultName="Prisijungimo registracija baigėsi"
faultVariable="Prisijungimo klaida"/>
 </onAlarm>
</eventHandlers>
```

2.2.5. Priverstinės pabaigos elementas

Priverstinės pabaigos trigeris atvaizduojamas į nutraukimo (*terminate*) elementą Šis elementas parodo, kad bet kuri veikla procese su kuria atliekamas nutraukimo procesas, atsitikus tam tikram įvykiui turi būti išimtinai nutraukiama.

```
<terminate standartiniai atributai>
  Standartiniai elementai
</terminate>
```

2.2.6. Laiko elementas

Laiko trigeris esant normaliai srauto tėkmei gali būti atvaizduojamas trim variantais:

- į laukimo (*wait*) elementą,
- jei laiko trigeris bus laiko/datos (*Time/Date*) atributas – į *wait* atributą *until*
- jei laiko trigeris bus laiko ciklas (*TimeCycle*) atributas – į *wait* atributą *for*

Šis elementas leidžia atidėti proceso veiklą tam tikram laikui arba iki tam tikro laiko momento.

```
<wait (for="ciklo trukme"| until="galutinė data") standartiniai atributai>
        Standartiniai elementai
</wait>
```

Kaip pavyzdžiui:

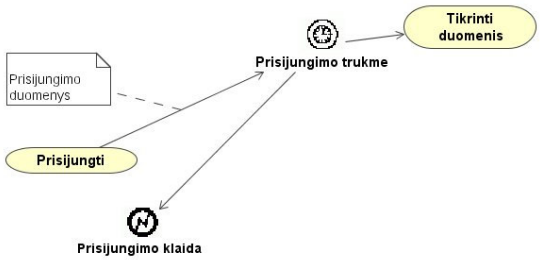
```
 <wait until=" '2004-12-25T18:00' ">
<wait for="T6H">
```

Laiko elementas gali būti atvaizduojamas ir į aktyvuoto pavojaus (*onAlarm*) elementą

- Į įvykį reguliuojantį (*eventHandler*) elementą įtraukiamas aktyvuotas pavojaus (*onAlarm*) elementas.
- jei laiko trigeris bus laiko/datos (*Time/Date*) atributas – į *onAlarm* atributą *until*
- jei laiko trigeris bus laiko ciklas (*TimeCycle*) atributas – į *onAlarm* atributą *for*

```
<onAlarm (for="ciklo trukme"| until="galutinė data")>
        veikla
</onAlarms>
```

Veikla gali būti: sužadinimas (*invoke*), išmetimas (*throw*) ir kt.

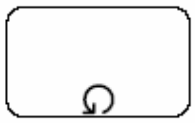


2.5 pav. Laiko elemento atvaizdavimo fragmentas iš 3.3 paveikslo modelio

Pavyzdžio fragmentas:

```
<variable name="Prisijungimo trukmė" type="integer">
 <eventHandlers>
  <onAlarm for ="bpws:getVariableData(Mokymosi įstaiga.Prisijungimo
trukmė)='Maxtrukmė'>
  </onAlarm>
</eventHandlers>
```

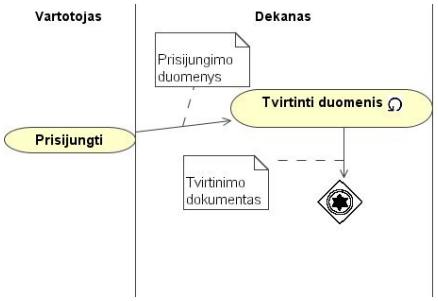
2.2.7. Ciklas



Ciklai atvaizduojami į kol (*while*) elementą Jie naudojami iteratyviai veiklai aprašyti. Iteratyvi veikla yra tęsiama tol, kol nebetenkinama loginė sąlyga. Kad pakeisti ciklo skaitliuką bus kuriama nauja veikla (*assign*), kuri bus dedama pačioje ciklo pabaigoje. Jei ciklo skaitliuko keisti nereikia ši veikla nebūtina.

```
<while conditon="loginė ciklo sąlyga" standartiniai atributai>
   Standartiniai elementai
   Veikla
  <assign>
   <copy>
    <from expression="sąlygos kintamojo pakeitimo išraiška"
    <to variable="kintamojo į kurį įrašoma išraiškos reikšmė pavadinimas"
/>
  </copy>
 </assign>
</while>
```

Jeigu modelyje (3.3 pav.) vietoje prisijungimo laiko elemento įdėtume ciklą:



2.6 pav. Ciklo atvaizdavimo fragmentas iš 3.3 paveikslo modelio

Pavyzdžio fragmentas:

```
<message name="Prisijungimas">
    <part name="Vartotojas.vardas" type="xsd:string"/>
    <part name="Vartotojas.pavadė" type="xsd:string"/>
    <part name="Vartotojas.studento_numeris" type="xsd:string"/>
</message>
<message name="Tvirtinimas">
    <part name="Dekanas.Tvirtinimo dokumentas" type="xsd:dokumentas"/>
</message>
<variables>
```

```
    <variable name=“Prisijungimo bandymų skaičius“ type=”integer”>
    <variable name=”Vartotojo duomenys” messageType=”Prisijungimas“>
    <variable name=”Patvirtinimas” messageType=”Tvirtinimas”>
</variables>
<while  condition=”bpws:getVariableData(Mokymosi įstaiga.Prisijungimų
skaičius)< 3”>
  <sequence>
    <invoke partnerLink=” Prisijungimas”
               portType=”Prisijungimo tvirtinimas”
               operation=”Prisijungti”
               inputVariable=” Vartotojo duomenys”
               outputVariable=”Patvirtinimas”/>
     <source linkName=”Prisijungti_Tvirtinti”/>
    </invoke>
    <assign>
    <copy>
     <from expression="bpws:getVariableData(Prisijungimo bandymų
skaičius)+1"/>
     <to variable="Prisijungimo bandymų skaičius"/>
    </copy>
   </assign>
  </sequence>
</while>
```

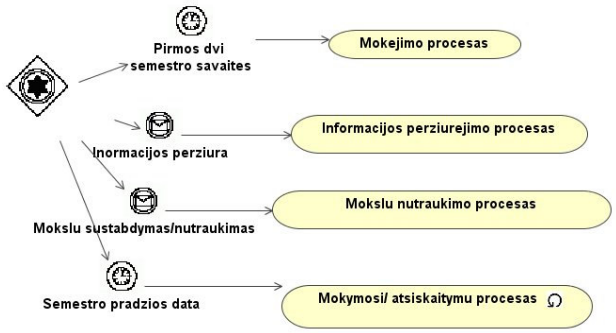
2.2.8. Įvykiais grindžiamas sprendimo taškas

Įvykiais grįstas sprendimų taškas atvaizduojamas į elementą pasirinkimas (*pick*). Pasirinkti elementai apsprendžiami pasirinktų išeinančių sekų. Jeigu sprendimų taško inicijavimo atributas bus nustatytas į „taip“ tada bus kuriamas egzempliorius (*createInstance*). Jei į „ne“ – nebus. Sprendimo elementas su gavimo (*recieve*) užduotimi bus atvaizduojami į aktyvuoto pranešimo (*onMessage*) elementą su pasirinkimu (*pick*). Pranešimo (*onMessage*) elementas yra identiškas gavimo (*recieve*) elementui, kuris buvo analizuojamas anksčiau. O gavimo užduoties atributai bus atitinkamai atvaizduojami į aktyvuoto pranešimo elementus tokius kaip jungimo tipas (*portType*) arba operacija (*operation*). Sprendimai su laiko įvykiais atvaizduojami į aktyvuoto sužadinimo (*onAlarm*) elementą su pasirinkimu (*pick*). Laiko/datos (t*imeDate*) atributai bus atitinkamai atvaizduojami į iki (*until*) elementus tokius kaip jungimo tipas (*portType*) arba operacija (*operation*). Sprendimai su taisyklių

įvykiais atvaizduojami į aktyvuoto pranešimo (*onMessage*) elementą su pasirinkimu (*pick*). Kadangi po šio elemento eina daugiau negu vienas elementas, tai šie elementai yra įtraukiami į seką po to kai jie atvaizduojami.

```
<pick createInstance=”yes|no”>
   <onMessage partnerLink=“Sąsajos pavadinimas“
              portType=“Siunčiamo pranešimo jungties pavadinimas“
              operation=“Operacijos pavadinimas“
              variable=”Kintamojo pavadinimas”/>
         veikla
  <onAlarm (for=“ciklo trukme“| until=”galutinė data”)>
         veikla
  </onAlarms>
 </pick>
```

Veikla gali būti: sužadinimas (*invoke*), klaidos metimas (*throw*) ir kt.



2.7 pav. Sprendimo taško pagrįsto įvykiais atvaizdavimo fragmentas iš 3.3 paveikslo modelio

```
<message name=“Informacijos prašymas“>
    <part name=”Dekanas.Naujienos” type=”xsd:dokumentas”/>
</message>
<message name=“Mokslų sustabdymas“>
    <part name=”Vartotojas. Pasiaškinimo dokumentas”
type=”xsd:dokumentas”/>
</message>
<message name=“Datos paskelbimas“>
    <part name=”Mokymosi įstaiga.Semestro pradzia” type=”xsd:data”/>
</message>
<message name=“Galutinės datos paskelbimas“>
    <part name=”Mokymosi įstaiga.Semestro pradzia+P14D” type=”xsd:data”/>
</message>
<portType name=”Peržiūra”>
    <operation name=”Peržiūrėti info”>
    <input message=“pos:Informacijos prašymas“/>
```

```
    </operation>
</portType>
<portType name=”Nutraukimas”>
    <operation name=”Nutrukti/atidėti mokslus”>
    <input message=“pos:Mokslų sustabdymas“/>
    </operation>
</portType>
<portType name=”Proceso paleidimas”>
    <operation name=”Pradėti semestro darbus”>
    <input message=“pos:Datos paskelbimas“/>
    </operation>
</portType>
<portType name=”Proceso užblokavimas”>
    <operation name=”Sustabdyti semestro darbus”>
    <input message=“pos:Galutinės datos paskelbimas“/>
    </operation>
</portType>
<plnk:partnerLinkType name=“Peržiūrėjimas“>
   <plnk:role name=“Peržiūros aptarnavimas“>
       <plnk:portType name“ Peržiūra“>
   </plnk:role>
</plnk:partnerLinkType>
<plnk:partnerLinkType name=“Nutraukti/Stabdyti“>
   <plnk:role name=“Nutraukimo aptarnavimas“>
       <plnk:portType name“ Nutraukimas“>
   </plnk:role>
</plnk:partnerLinkType>
<plnk:partnerLinkType name=“Pradėti“>
   <plnk:role name=“Paleidimo aptarnavimas“>
       <plnk:portType name“ Proceso paleidimas“>
   </plnk:role>
</plnk:partnerLinkType>
<plnk:partnerLinkType name=“Blokuoti“>
   <plnk:role name=“Blokavimo aptarnavimas“>
       <plnk:portType name“ Proceso užblokavimas“>
   </plnk:role>
</plnk:partnerLinkType>
<partnerLinks>
 <partnerLink name=“Peržiūrėjimas“>
```

```
   myRole=” Peržiūros aptarnavimas”
< partnerLinkType =”Mokymosi Nutraukimas”>
  myRole=” Nutraukimo aptarnavimas”
<partnerLink name=“Paleidimas“>
   myRole=” Paleidimo aptarnavimas”
<partnerLink name=“Blokavimas“>
   myRole=” Blokavimo aptarnavimas”
</partnerLinks>
<variables>
 <variable name=”Informacijos peržiūra” messageType=”lns:Informacijos
prašymas”>
<variable name=”Mokslų sustabdymas/nutraukimas” messageType=”lns:Mokslų
sustabdymas”>
<variable name=”Semestro pradžios data” messageType=”lns:Datos
paskelbimas”>
<variable name=”Semestro pradžios data+ 2sav” messageType=”lns: Galūtinės
datos paskelbimas”>
</variables>
<pick createInstance=”yes”>
     <onAlarm (until=“Semestras.Pradzia+P14D“)>
        <invoke partnerLink=”Blokavimas”
                portType=”Proceso užblokavimas”
                operation=”Sustabdyti semestro darbus”
                inputVariable=”Semestro pradžios data+ 2sav”
        </invoke>
     </onAlarms>
     <onMessage partnerLink=“ Peržiūrėjimas“
              portType=“Peržiūra“
              operation=“Peržiūrėti info“
              variable=”Informacijos peržiūra”/>
     <onMessage partnerLink=“Mokymosi Nutraukimas“
              portType=“Nutraukimas“
              operation=“Nutrukti/atidėti mokslus“
              variable=”Mokslų sustabdymas/nutraukimas”/>
     <onAlarm (for=” Semestras.Pradzia”)>
        <invoke partnerLink=”Paleidimas”
               portType=”Proceso paleidimas”
               operation=”Pradėti semestro darbus”
               inputVariable=”Semestro pradžios data”
```

```
      </invoke>
     </onAlarms>
  </pick>
```

2.2.9. Duomenimis grindžiamas sprendimo taškas

Duomenimis grindžiamas sprendimų taškas atvaizduojamas į elementą jungtuką (*switch*). Sprendimų taško sąlygos bus atvaizduojamos į jungtuko sąlygas. Kadangi po šio elemento eina daugiau negu vienas elementas, tai šie elementai yra įtraukiami į elementų seką (*case*). Iš anksto numatytas (*default*) sprendimas bus atvaizduojamas į jungtuko (*switch*) elementą kitu atveju (*otherwise*).

```
<switch standartiniai atributais>
    Standartiniai elementai
   <case condition = "Išsišakojimo sąlyga(loginis kintamasis)">
         veikla
   </case>
   <otherwise>
          veikla
   </otherwise>
</switch>
```

Veikla gali būti: sužadinimas (*invoke*), klaidos metimas (*throw*), kažkokia įvykių tėkmė (*flow*) ir kt.

Šiuo metu sudarytame eksperimentiniame modelyje, duomenimis grindžiamas sprendimo taškas nėra panaudotas, bet jis yra labai panašus į apimančio sprendimo atvaizdavimą pateiktą žemiau esančiame poskyryje. Skirtumas tas, kad šiuo atveju būtų užpildyta iš anksto numatyta (*otherwise)* šaka, nes būtinai turi būti pasirinktas vienas (ir tik vienas) srautas, kitu atveju priskiriama iš anksto numatyta seka (<*otherwise*> atšakoje). O apimančio sprendimo atveju gali būti pasirinkta daugiau nei viena, arba išvis nepasirinkta tolesnės įvykių tėkmės atšaka.

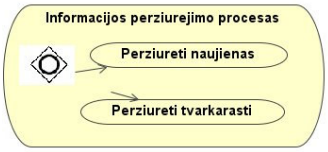
2.2.10. Daugialypis sprendimo taškas

Daugialypis sprendimų taškas atvaizduojamas į elementą jungtuką (*switch*). Sprendimų taško sąlygos bus atvaizduojamos į jungtuko sąlygas. Šis elementas visada turi bent dvi šakas. Viena iš jų iš anksto apspręsta (*default*). Kadangi po šio elemento eina daugiau negu vienas elementas, tai jie yra

įtraukiami į seką (*sequence*) po to kai šie elementai atvaizduojami. Iš anksto numatytas (*default*) sprendimas bus atvaizduojamas į jungtuko (*switch*) elementą kitu atveju (*otherwise*). Iš anksto numatytas kelias bus pasirinktas tik tuo atveju, jeigu jokie kiti keliai nebus pasirinkti. Tai reiškia, kad sekos (*sequence*) elementas turi apimti ir visą srautą ir visus pasirinkimus. Kad sužinoti koks kelias pasirinktas iš anksto numatytas ar ne yra kuriamas stebėjimo (*tracking*) kintamasis.

```
<switch standartiniai atributais>
    Standartiniai elementai
   <case condition = "Išsišakojimo sąlyga(loginis kintamasis)">
          veikla
     <sequence standartiniai atributai>
          veikla
     </sequence>
   </case>
   <otherwise>
          veikla
   </otherwise>
</switch>
```

Veikla gali būti: sužadinimas (*invoke*), metimas (*throw*), įvykių tėkmė (*flow*) ir kt.



2.8 pav. Apimančio sprendimo atvaizdavimo fragmentas iš 3.3 paveikslo modelio

```
<message name="Mokslų sustabdymas">
    <part name="Vartotojas.Pasiaiškinimo dokumentas"
type="xsd:dokumentas"/>
</message>
<message name="Mokslų nutraukimas">
    <part name="Vartotojas.Pasiaiškinimo dokumentas"
type="xsd:dokumentas"/>
</message>
<portType name="Sustabdymo organizavimas">
    <operation name="Pasirinkti mokslų sustabymo meniu">
    <input message="pos: Mokslų sustabdymas"/>
    </operation>
</portType>
<portType name="Nutraukimo organizavimas">
    <operation name="Pasirinkti mokslų nutraukimo meniu">
```

```
    <input message="pos:Mokslų nutraukimas"/>
    </operation>
</portType>
<plnk:partnerLinkType name="Stabdyti">
   <plnk:role name="Stabdymo aptarnavimas">
       <plnk:portType name"Sustabdymo organizavimas">
   </plnk:role>
</plnk:partnerLinkType>
<plnk:partnerLinkType name="Nutraukti">
   <plnk:role name="Nutraukimo aptarnavimas">
       <plnk:portType name"Nutraukimo organizavimas">
   </plnk:role>
</plnk:partnerLinkType>
<partnerLinks>
 <partnerLink name="Mokymosi sustabdymas">
   myRole=" Stabdymo aptarnavimas"
< partnerLinkType ="Mokymosi nutraukimas">
  myRole=" Nutraukimo aptarnavimas"
</partnerLinks>
<variables>
 <variable name="Ruošimasis akademinėms" messageType="lns: Mokslų
sustabdymas">
<variable name="Ruošimasis nutraukimui" messageType="lns:Mokslų
nutraukimas">
</variables>
<switch name="Sumokėta už moksla?">
   <case condition="bpws: getVariableProperty(Dekanas.Sumokėta)=true">
    <sequence>
        <invoke name="Paprašyti akademinių atostogų"
                partnerLink="Mokymosi sustabdymas"
                portType="Sustabdymo organizavimas"
                operation="Pasirinkti mokslų sustabymo meniu"
                inputVariable="Ruošimasis akademinėms"
        </invoke>
        <invoke name="Nutraukti mokslus"
                partnerLink="Mokymosi nutraukimas"
                portType="Nutraukimo organizavimas"
                operation="Pasirinkti mokslų nutraukimo meniu"
                inputVariable="Ruošimasis nutraukimui"
```

```
        </invoke>
   <sequence>
  </case>
   <otherwise>
          <empty/>
   </otherwise>
</switch>
```

2.2.11. Lygiagretus sprendimo taškas

Lygiagretus sprendimo taškas atvaizduojamas į elementą srautas (*flow*). Srautas baigiasi tada kai visos veiklos, įeinančios į tą srautą baigiasi. Sąsaja (*link*) reikalinga srautams sinchronizuoti. Nes viena sąsaja kitai gali būti šaltinis (*source*), kita rezultatas (*target*).

```
<flow standartiniai atributai>
  <links>
    <link name=”Sąsajos pavadinimas”>
  </links>
   veikla
</flow>
```

Veikla gali būti sužadinimas (*invoke*), metimas (*throw*) ir kt.

Šis elementas eksperimentiniame modelyje nenaudojamas, bet nesunkiai atvaizduojamas pagal pateiktą šabloną ir gali būti įtraukiamas į įvairių tipų sprendimo taškų srautus.

2.3. Šablonai vartotojui

Ankstesniuose skyriuose pateikti išanalizuoti verslo modelių reikalavimai, tam skirtos UML 2.0, BPMN ir BPEL notacijos. Pasiūlyta universali modeliavimo metodika papildant UML 2.0 verslo modeliavimo notacijos BPMN elementais ir susiejant dalykinės srities modelį su veiklos proceso modeliu. Aprašytas išplėstosios notacijos atvaizdavimas į BPEL. 2.3 skyriuje remiantis įgyta patirtimi, sudaryti ir pateikti specifikacijų šablonai vartotojui natūralia kalba, padėsiantys suprojektuoti CASE įrankio eskizą.

2.3.1. Įvykiais grindžiamas sprendimo taškas

Veiklos proceso diagramose naudojamas įvykiais grįstas sprendimų taškas. Šis sprendimas ypatingas tuo, kad lygiagretus išsišakojimas grįstas įvykiais atsirandančiais procese. Šis taškas atlieka

papildomą sekų kontrolę. Po šio sprendimų taško seka tarpiniai įvykiai. Kai įvykis atvyksta tikrinama, kuris kelias tenkinamas, tuo keliu srautas siunčiamas toliau.

n – išsišakojimų skaičius.

$a_1, a_2, a_3 \ldots a_n$ – pranešimai, laiko elementai.

$a_{11}, a_{12}, a_{13} \ldots a_{1n}; a_{21}, a_{22}, a_{23} \ldots a_{2n}; \ldots; a_{n1}, a_{n2}, a_{n3} \ldots a_{nn}$ – tipai, reikalingi atitinkamiems elementams aprašyti (kokie tipai ir elementai reikalingi pavaizduota metamodelyje).

$b_1, b_2, b_3 \ldots b_n$ – sekos, einančios iš sprendimo mazgo (procesai, sužadinimai, atsakai, ciklai, srautai ir kt.).

$b_{11}, b_{12}, b_{13} \ldots b_{1n}; b_{21}, b_{22}, b_{23} \ldots b_{2n}; \ldots; b_{n1}, b_{n2}, b_{n3} \ldots b_{nn}$ – tipai, reikalingi atitinkamiems elementams aprašyti (kokie tipai ir elementai reikalingi pavaizduota metamodelyje).

Jeigu a_1 (arba a_2 arba a_3 arba...a_n)
tada b_1 (arba b_2 arba b_3 arba...b_n)

2.3.2. Duomenimis grindžiamas sprendimo taškas

Kai reikalinga srautų kontrolė BPMN naudojamas duomenimis grįstas sprendimų taškas. Atėjus srautui, tikrinama, kuri sąlyga yra išpildoma. Ta atšaka, kurioje sąlyga yra išpildoma tęsiamas srautas. Tarp visų alternatyvių srautų gali būti tik vienas kuris užbaigs srautų eigą

n – išsišakojimų skaičius.

$a_1, a_2, a_3 \ldots a_n$ – duomenys (imami iš duomenų modelio).

$b_1, b_2, b_3 \ldots b_n$ – kintamieji.

$c_1, c_2, c_3 \ldots c_n$ – sekos, einančios iš sprendimo mazgo (procesai, sužadinimai, atsakai, ciklai ir kt.).

$c_{11}, c_{12}, c_{13} \ldots c_{1n}; c_{21}, c_{22}, c_{23} \ldots c_{2n}; \ldots; c_{n1}, c_{n2}, c_{n3} \ldots c_{nn}$ – tipai, reikalingi atitinkamiems elementams aprašyti (kokie tipai ir elementai reikalingi pavaizduota metamodelyje).

d – iš anksto numatytas šaka, kuria einama, jei netenkinamos kitos sąlygos.

$d_1, d_2, d_3 \ldots d_n$ – tipai, reikalingi atitinkamiems elementams aprašyti.

Jeigu $a_1 > | < | = | \geq | \leq b_1$ (arba $a_2 > | < | = | \geq | \leq b_2$ arba $a_3 > | < | = | \geq | \leq b_3$ arba ...
$a_n > | < | = | \geq | \leq b_n$)
tada c_1 (arba c_2 arba c_3 arba...c_n)
kitu atveju d

2.3.3. Daugialypis sprendimo taškas

Šis taškas priėmęs ateinantį srautą išskaido į tiek lygiagrečių kelių, kiek sąlygų yra tenkinama. Nuo išplėtimų taško skiriasi tuo, kad galima pasirinkti ne vieną, o nulį ir daugiau išeinančių alternatyvių srautų. Suteikiama galimybė nesirinkti nė vienos alternatyvos.

n – išsišakojimų skaičius.

$a_1, a_2, a_3 \ldots a_n$ – duomenys (imami iš duomenų modelio).

$b_1, b_2, b_3 \ldots b_n$ – kintamieji.

$c_1, c_2, c_3 \ldots c_n$ – sekos, einančios iš sprendimo mazgo (procesai, sužadinimai, atsakai, ciklai ir kt.).

$c_{11}, c_{12}, c_{13} \ldots c_{1n}; c_{21}, c_{22}, c_{23} \ldots c_{2n}; \ldots; c_{n1}, c_{n2}, c_{n3} \ldots c_{nn}$ – tipai, reikalingi atitinkamiems elementams aprašyti (kokie tipai ir elementai reikalingi pavaizduota metamodelyje).

d – iš anksto numatytas šaka, kuria einama, jei netenkinamos kitos sąlygos.

$d_1, d_2, d_3 \ldots d_n$ – tipai, reikalingi atitinkamiems elementams aprašyti.

Kiek sąlygų yra tenkinama tiek srautų yra tenkinama. Šaka kitu atveju nebūtina, nes galima nesirinkti nei vieno srauto.

Jeigu $(a_1 > | < | = | \geq | \leq b_1)ir(a_2 > | < | = | \geq | \leq b_2)ir(a_3 > | < | = | \geq | \leq b_3)$ir...ir
$(a_n > | < | = | \geq | \leq b_n)$
tada c_1 ir c_2 ir c_3 ir...ir c_n
kitu atveju d

2.3.4. Ciklas

a – ciklo kintamasis.

b – pastovus kintamasis (gali būti iš domenų modelio).

$c_1, c_2, c_3 \ldots c_n$ – veiklų seka (sužadinimas, klaidos metimas atsakas ir kt.).

$c_{11}, c_{12}, c_{13} \ldots c_{1n}; c_{21}, c_{22}, c_{23} \ldots c_{2n}; \ldots; c_{n1}, c_{n2}, c_{n3} \ldots c_{nn}$ – tipai, reikalingi atitinkamiems elementams aprašyti (kokie tipai ir elementai reikalingi pavaizduota metamodelyje).

Gali būti tik viena veikla, bet gali būti ir kelios. Jei ciklo duomenų keisti nereikia, tuomet sumavimas ciklo gale nereikalingas.

Kol a $> | < | = | \geq | \leq$ b
atlikti c_1
atlikti c_2
atlikti c_3
…
atlikti c_n

a=a+1

2.3.5. Laikas

a – to momento data

b – kintamasis

d – konkreti data

$c_1, c_2, c_3 \ldots c_n$ – veiklų seka (sužadinimas, išmetimas atsakas ir kt.).

$c_{11}, c_{12}, c_{13} \ldots c_{1n}; c_{21}, c_{22}, c_{23} \ldots c_{2n}; \ldots; c_{n1}, c_{n2}, c_{n3} \ldots c_{nn}$ – tipai, reikalingi atitinkamiems elementams aprašyti (kokie tipai ir elementai reikalingi pavaizduota metamodelyje).

```
Laukti kol a=b
   atlikti c1
   atlikti c2
   atlikti c3
   …
   atlikti cn

Laukti iki a=d
   atlikti c1
   atlikti c2
   atlikti c3
   …
   atlikti cn
```

2.4. Šablonai OCL kalba.

OCL – išraiškų kalba. Ji naudojama kaip UML dalis, todėl šiame darbe šablonai vartotojui pateikiami ne tik natūralia, bet ir OCL kalba. OCL kalba plečiama ir tobulinama, bet kol kas neturi galimybės užrašyti visų galimų sąlygos ir ciklo sakinių. 2.4. skyriuje pateikti šablonai vartotojui bei pasiūlymai, kaip būtų galima realizuoti tam tikrus ciklo sakinius, kurie šiuo metu OCL kalboje negalimi.

2.4.1. Įvykiais grindžiamas sprendimo taškas

n – išsišakojimų skaičius.

$a_1, a_2, a_3 \ldots a_n$ – pranešimai, laiko elementai.

$a_{11}, a_{12}, a_{13} \ldots a_{1n}; a_{21}, a_{22}, a_{23} \ldots a_{2n}; \ldots; a_{n1}, a_{n2}, a_{n3} \ldots a_{nn}$ – tipai, reikalingi atitinkamiems elementams aprašyti (kokie tipai ir elementai reikalingi pavaizduota metamodelyje).

$b_1, b_2, b_3 \ldots b_n$ – sekos, einančios iš sprendimo mazgo (procesai, sužadinimai, atsakai, ciklai, srautai ir kt.).

$b_{11}, b_{12}, b_{13} \ldots b_{1n}; b_{21}, b_{22}, b_{23} \ldots b_{2n}; \ldots; b_{n1}, b_{n2}, b_{n3} \ldots b_{nn}$ – tipai, reikalingi atitinkamiems elementams aprašyti (kokie tipai ir elementai reikalingi pavaizduota metamodelyje).

```
if a1 (or a2 or a3 or...an)
    then b1 (or b2 or b3 or...or bn)
endif
```

2.4.2. Duomenimis grindžiamas sprendimo taškas

n – išsišakojimų skaičius.

$a_1, a_2, a_3 \ldots a_n$ – duomenys (imami iš duomenų modelio).

$b_1, b_2, b_3 \ldots b_n$ – kintamieji.

$c_1, c_2, c_3 \ldots c_n$ – sekos, einančios iš sprendimo mazgo (procesai, sužadinimai, atsakai, ciklai ir kt.).

$c_{11}, c_{12}, c_{13} \ldots c_{1n}; c_{21}, c_{22}, c_{23} \ldots c_{2n}; \ldots; c_{n1}, c_{n2}, c_{n3} \ldots c_{nn}$ – tipai, reikalingi atitinkamiems elementams aprašyti (kokie tipai ir elementai reikalingi pavaizduota metamodelyje).

d – iš anksto numatytas šaka, kuria einama, jei netenkinamos kitos sąlygos.

$d_1, d_2, d_3 \ldots d_n$ – tipai, reikalingi atitinkamiems elementams aprašyti.

```
if a1>|<|=|≥|≤ b1 (or a2>|<|=|≥|≤ b2 or a3>|<|=|≥|≤ b3 or ...or
      an>|<|=|≥|≤ bn)
  then c1 (or c2 or c3 or...or cn)
  else d
endif
```

2.4.3. Daugialypis sprendimo taškas

n – išsišakojimų skaičius.

$a_1, a_2, a_3 \ldots a_n$ – duomenys (imami iš duomenų modelio).

$b_1, b_2, b_3 \ldots b_n$ – kintamieji.

$c_1, c_2, c_3 \ldots c_n$ – sekos, einančios iš sprendimo mazgo (procesai, sužadinimai, atsakai, ciklai ir kt.).

$c_{11}, c_{12}, c_{13} \ldots c_{1n}; c_{21}, c_{22}, c_{23} \ldots c_{2n}; \ldots; c_{n1}, c_{n2}, c_{n3} \ldots c_{nn}$ – tipai, reikalingi atitinkamiems elementams aprašyti (kokie tipai ir elementai reikalingi pavaizduota metamodelyje).

d – iš anksto numatytas šaka, kuria einama, jei netenkinamos kitos sąlygos.

$d_1, d_2, d_3 \ldots d_n$ – tipai, reikalingi atitinkamiems elementams aprašyti.

Kiek sąlygų yra tenkinama tiek srautų yra tenkinama. Šaka kitu atveju nebūtinas, nes galima nesirinkit nei vieno srauto.

```
if (a1>|<|=|≥|≤ b1)and(a2>|<|=|≥|≤ b2)and(a3>|<|=|≥|≤ b3)and...and
      (an>|<|=|≥|≤ bn)
    then c1 and c2 and c3 and...and cn
    else d
endif
```

2.4.4. Ciklas

OCL kalboje nėra `while` sakinio, sudarant ciklo šabloną vartotojui pateikiamas pavyzdys, kaip būtų galima tai realizuoti `set` ir `iterate` pagalba.

`a` – ciklo kintamasis.

`b` – pastovus kintamasis (gali būti iš domenų modelio).

$c_1, c_2, c_3 \ldots c_n$ – veiklų seka (sužadinimas, klaidos metimas atsakas ir kt.).

$c_{11}, c_{12}, c_{13} \ldots c_{1n}; c_{21}, c_{22}, c_{23} \ldots c_{2n}; \ldots; c_{n1}, c_{n2}, c_{n3} \ldots c_{nn}$ – tipai, reikalingi atitinkamiems elementams aprašyti (kokie tipai ir elementai reikalingi pavaizduota metamodelyje).

```
Set{1,1,...,b}->iterate(
   i:integer; a:integer=0
   if a >|<|=|≥|≤ b
     then {c1 and c2 and c3 and...and cn;
          a=a+1;}
   endif)
```

2.4.5. Laikas

`a` – to momento data.

`b` – reikiama data.

$c_1, c_2, c_3 \ldots c_n$ – veiklų seka (sužadinimas, išmetimas atsakas ir kt.).

$c_{11}, c_{12}, c_{13} \ldots c_{1n}; c_{21}, c_{22}, c_{23} \ldots c_{2n}; \ldots; c_{n1}, c_{n2}, c_{n3} \ldots c_{nn}$ – tipai, reikalingi atitinkamiems elementams aprašyti (kokie tipai ir elementai reikalingi pavaizduota metamodelyje).

```
if a=b
  then c1 and c2 and c3 and...and cn
endif
```

3. EKSPERIMENTAS

Eksperimento dalyje pasiūlyta modeliavimo metodika pritaikoma UML CASE įrankiui. Darbe analizuoti modeliai kuriami su MagicDraw įvedant papildomus stereotipus. Pasiūlyta papildyti MagicDraw veiklos modeliavimo elementų specifikacijas, kurios leistų aprašyti verslo taisyklių naudojamų duomenų modelio elementus. Tokiu būdu sudarytą modelį būtų galima konvertuoti į vykdomąją kalbą.

3.1. Modelis naudojant siūlomą notaciją

Veiklos procesams modeliuoti pasirinkta nuotolinio mokymosi sistema, kuri gerai atspindi e. verslo taisyklių svarbą ir jų užrašymo lankstumą. Eksperimentinio modelio scenarijus:

1 ž . Prisijungimas. Vartotojas jungiasi prie sistemos. Užduodamas prisijungimo laikas. Prisijungimo metu tikrinami ar teisingi vartotojo pateikti duomenys. Jei įvedame duomenis teisingi atliekamas 2 žingsnis. Jei ne išmetamas klaidos pranešimas.

2 ž . Informacijos peržiūra ir redagavimas. Vartotojui prisijungus tikrinama ar sumokėta už mokslą. Jei mokesti už mokslą nesumokėtas tuomet galima atlikti tik 2.1 ir 2.2 žingsnius. Jei mokestis sumokėtas galima atlikti 2.2 , 2.3 ir 2.4 žingsnius.

2.1 ž . Mokėjimo už mokslą procesas. Vartotojas gali sumokėti už mokslą pirmų dviejų semestro savaičių bėgyje.

2.2 ž . Informacijos peržiūrėjimo procesas. Pasirinkus informacijos peržiūros procesą siunčiamas pranešimas ir vartotojos nukreipiamas į naują pasirinkimą kur galima peržiūrėti naujienas ir tvarkaraštį. Jei už mokslą nesumokėta tuomet užduodamas laikas per kurį vartotojas gali peržiūrėti naujienas ir tvarkaraštį. Užduotam laikui pasibaigus vartotojo paklausiama ar jis nori sumokėti už mokslą. Jei vartotojas pageidauja sumokėti už mokslą, tuomet atliekamas 2.1 žingsnis, jei ne atliekamas 3 žingsnis.

2.3 ž . Mokslų nutraukimo procesas. Vartotojui pageidavus mokymosi įstaiga gali nutraukti mokslus 2.3.1 ž. arba išleisti akademinių atostogų 2.3.2 ž..

2.3.1 ž . Prašymas akademinių atostogų. Vartotojas siunčia pranešimą bei prašymo dokumentą dekanui apie akademines atostogas. Dekanas atsiunčia patvirtinimo dokumentą ir vartotojo mokslas laikinai sustabdomas, atliekamas 3 žingsnis.

2.3.2 ž . Prašymas nutraukti mokslus. Vartotojas siunčia pranešimą bei prašymo dokumentą dekanui apie mokslo nutraukimą. Dekanas atsiunčia patvirtinimo dokumentą ir vartotojo mokslas sustabdomas atliekamas 3 žingsnis.

2.4 ž . Mokymosi ir atsiskaitymų procesas. Mokymosi ir atsiskaitymo procesas pradedamas nuo užduoto laiko (semestro pirmos dienos). Mokymosi atsiskaitymo procesas susideda iš semestro

darbų atsiskaitymo 2.4.1, sesijos 2.4.2 ir video konferencijos su dėstytojais 2.4.3. Mokymosi ir atsiskaitymų trukmė – semestro laikas. Atlikus reikiamas operacijas ir panorėjus atsijungti nuo sistemos atliekamas 3 žingsnis.

2.4.1 ž. Semestro darbų atsiskaitymas. Semestro darbų atsiskaitymas susideda iš atsiskaitymų ciklo bei darbų atlikimo ir sudėjimo ciklo. Darbai ir atsiskaitymai negalimi jei neparsiųsta reikiama medžiaga.

2.4.2 ž. Sesija. Savaitę iki sesijos pradžios galima redaguoti sesijos tvarkaraštį. Sesijos metu vyksta egzaminų ir rezultatų registravimo ciklas. Esant neigiamiems rezultatams galimas kompensavimo ciklas, kurio metu pakartotinai laiko egzaminai ir registruojami rezultatai.

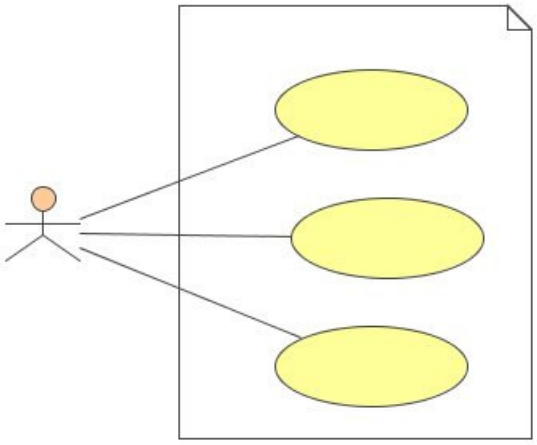
2.4.3 ž. Video konferencija su dėstytojais. Video konferencijos metu galima klausyti paskaitas, bendrauti su dėstytojais ir kt. abiem pusėms patogiu laiku.

3 ž . Atsijungimas. Vartotojai atsijungia savo noru arba atjungiami priverstinai.

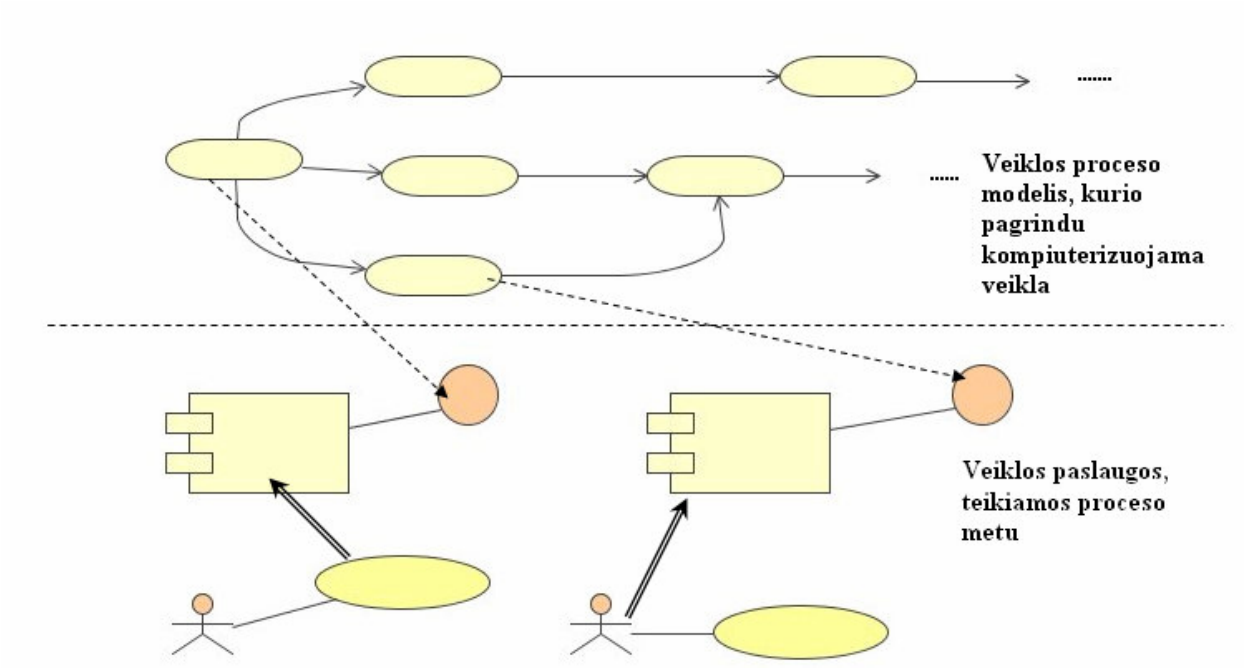
Projektuojant įprastas informacines sistemas, projekto reikalavimai aprašomi panaudojimo atvejų modeliu (3.1 pav.). Projektuojant e. verslo procesus, reikalavimai aprašomi veiklos proceso modeliu, kuris transformuojamas į vykdymo kalbą (pvz., BPEL). Toks projektavimas pavaizduotas 3.2 paveiksle. Vykdomoji kalba aprašo veiklos valdymo logiką ir susieja proceso žingsnius su paslaugomis, kurios tuose žingsniuose naudojamos. Elektroninės paslaugos, realizuojančios proceso elementus, gali būti sukurtos iš anksto arba sukuriamos pagal apibrėžto proceso reikalavimus. Atskiroms, ne sudėtinėms elektroninėms paslaugoms kurti naudojamas įprastas kūrimo procesas, paremtas panaudojimo atvejais.

Taigi kompiuterizuojant veiklą e. verslo procesų modelių pagrindu galimi du keliai:

1. Sukurti proceso programinę įrangą iš esamų paslaugų.
2. Pagal suprojektuotą procesą kurti paslaugas.



3.1 Įprastas informacinės sistemos kūrimas remiantis panaudojimo atvejais



3.2 Informacinės sistemos kūrimas grindžiamas veiklos proceso modeliu

Kad vaizdžiai parodyti modelių sudarymo skirtumus UML 2.0 ir UML 2.0 papildytai BPMN buvo braižomos trys eksperimentinio modelio veiklos diagramos. Trys, todėl, kad UML 2.0 notacija lyginant su ankstesnėmis buvo papildyta dar vienu veiklos diagramų struktūros tipu.

Pirmas modelis – nuotolinės mokymosi sistemos modelis realizuotas UML 2.0 naudojant konteinerius bei juostas pateiktas 3.3 paveiksle (modeliavimas atliktas Enterprise Architect Corporate v4.0).

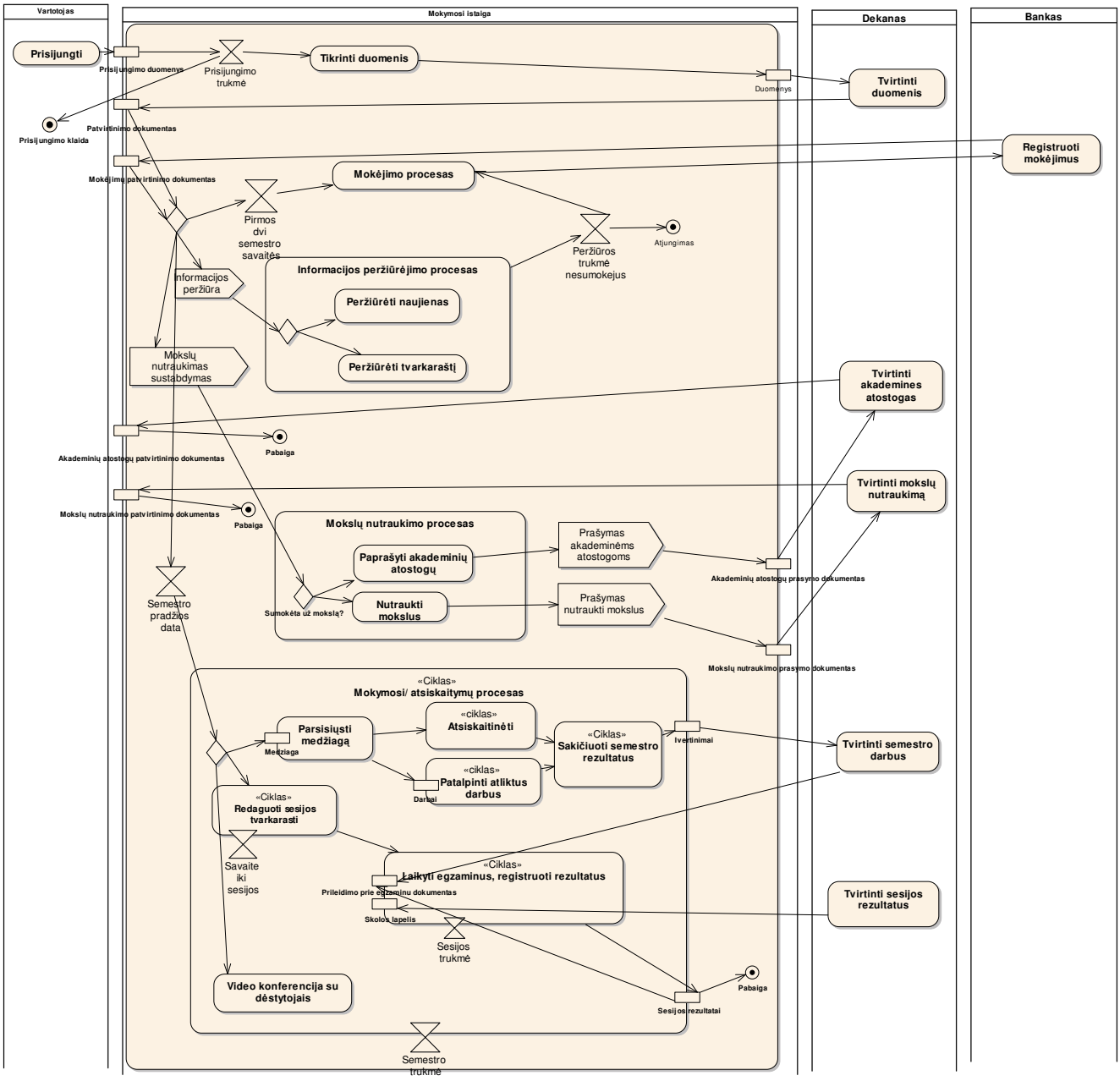
Antras modelis – nuotolinės mokymosi sistemos modelis braižant be juostų ir konteinerių, bet naudojant anotacijas. Naujame UML 2.0 standarte galima pasirinkti verslo dalyvius žymėti kaip juostas arba nenaudoti juostų, bet naudoti anotacijas. Anotacijos žymimos prie kiekvieno veiksmo pavadinimo laužtiniuose skliaustuose nurodant verslo dalyvį, kuris tą veiksmą atlieka (pvz., Prisijungti [Dekanas]). Šis diagramos tipas paprastesnis ir aiškesnis vartotojui. Lengviau identifikuoti kokią veiklą koks darbuotojas kokio proceso eigoje atlieka. Eksperimentinis modelis pateiktas 3.4 paveiksle (modeliavimas atliktas Enterprise Architect Corporate v4.0 paketu).

Trečias modelis – nuotolinės mokymosi sistemos modelis braižomas naudojantis nauja pasiūlyta metodika pateiktas 3.5 paveiksle. Naujos metodikos idėja tokia, kad e. verslo procesas būtų išsamiai aprašomas siūloma UML 2.0 papildyti BPMN elementais (1 lentelė, 2 lentelė). Tai atlikus turėtume universalią modeliavimo kalbą, kuri išsamiai aprašytų veiklos procesus. Tokios sąsajos prototipas įgyvendintas įvedant stereotipus į MagicDraw 9.0.

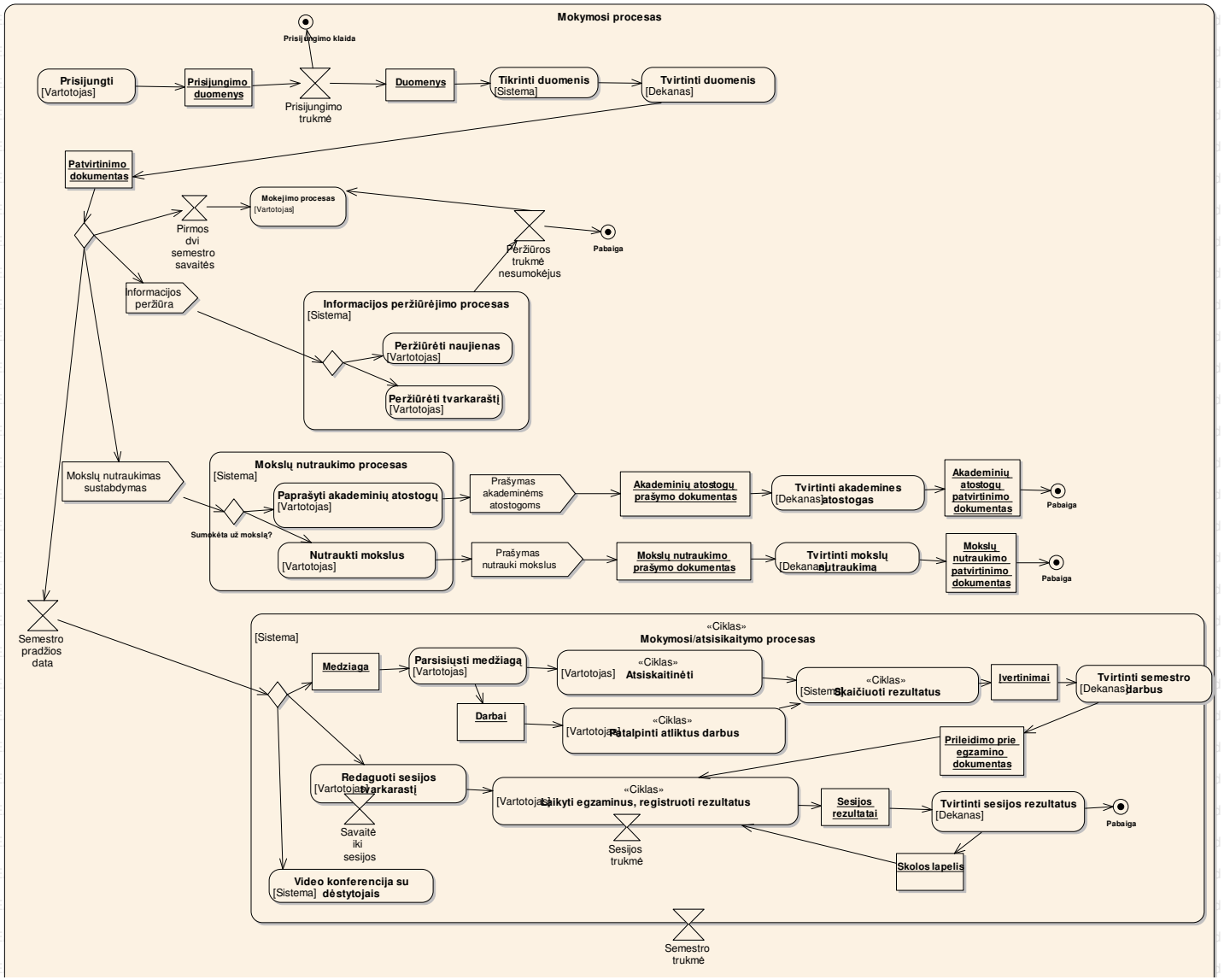
Kodėl BPMN? Tai specializuota verslo procesų kalba. BPMN notacijoje didesnė elementų įvairovė, aiškiau suprantamas modelis bei paprasčiau ir išsamiau aprašomos e. verslo taisyklės. Analizuojant ir lyginant diagramas matoma, kad vien sprendimo taškų, bei trigerių gausa trečiame

modelyje padeda greičiau įsisavinti visą veiklos procesą bei aprašyti veiklos taisykles. Tuo tarpu UML 2.0 visi sprendimų taškai gali būti realizuoti, bet modelis tuomet taptų sudėtingas, sunkiau suprantamas būtų ir sprendimo taškų specifiškumas.

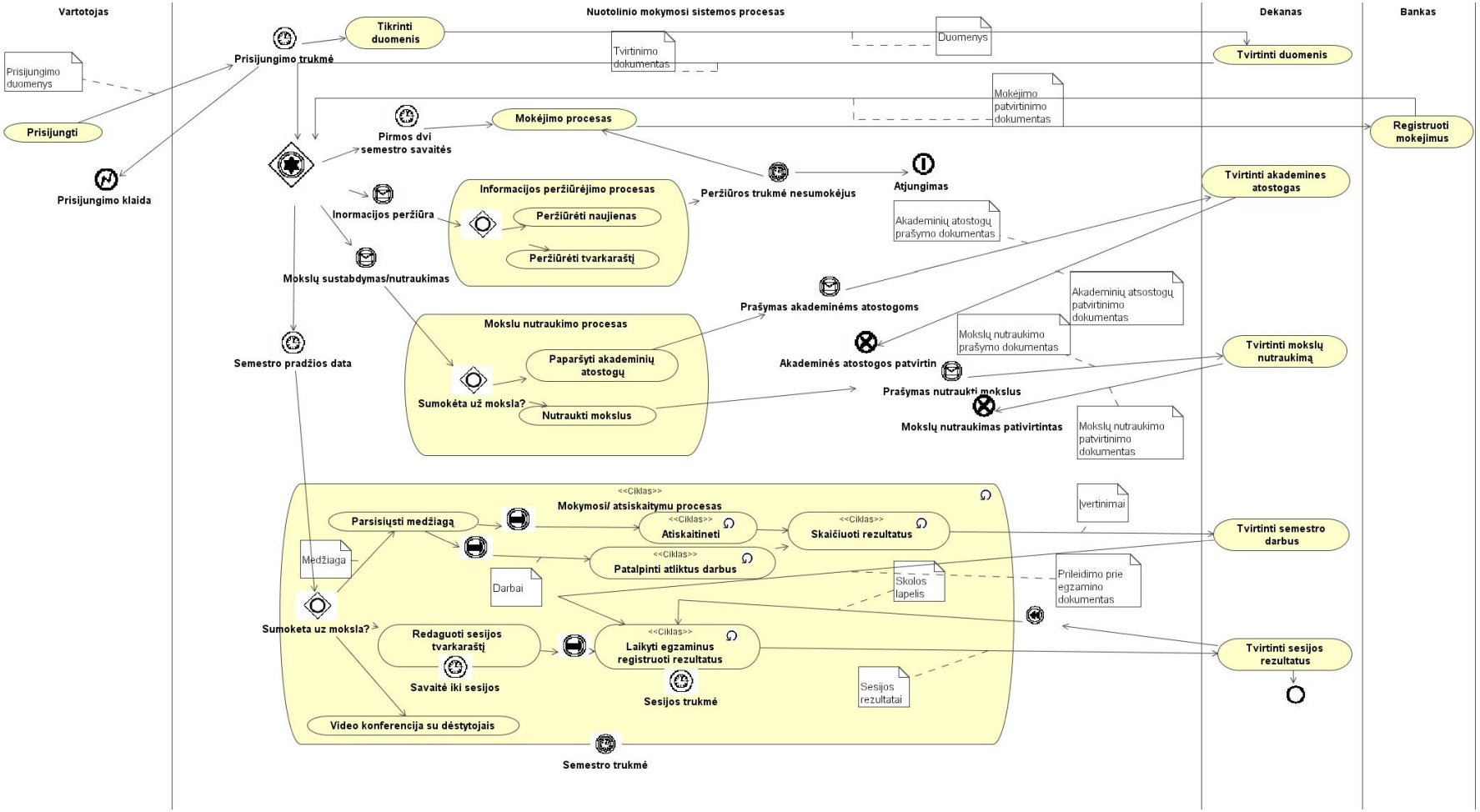
Kodėl UML 2.0? Ši kalba plačiausiai naudojama, be to tai universali kalba kur greta veiklos diagramų yra išsamus priemonių rinkinys informacinei sistemai projektuoti, pvz., 3.7, 3.8 paveikslai. Vienas iš didžiausių BPMN trūkumų, kad nėra priemonių dalykinei sričiai vaizduoti. Dokumentai (reikalingi duomenys) gali būti parodyti tik kaip pastabos. UML veiklos diagramose dokumentai vaizduojami kaip objektai, kurie dalyvauja veiklos procese ir turi įtakos jo eigai. Jie projektuojami analizės klasių diagramomis, pvz., 3.6 paveikslas.



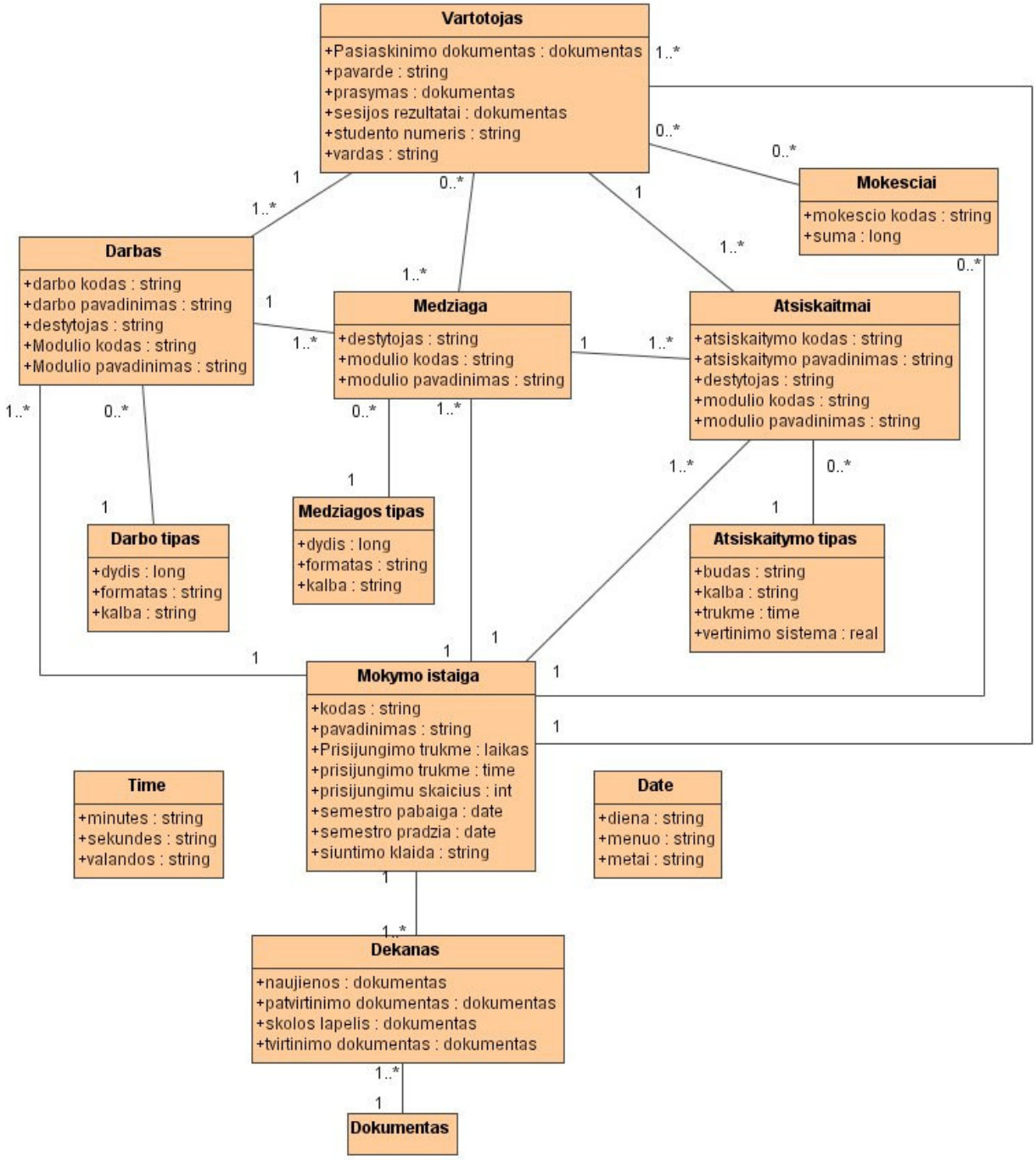
3.3 Nuotolinės mokymosi sistemos veiklos modelis UML 2.0



3.4 pav. Nuotolines mokymosi sistemos veiklos modelis UML 2.0 su anotacijomis

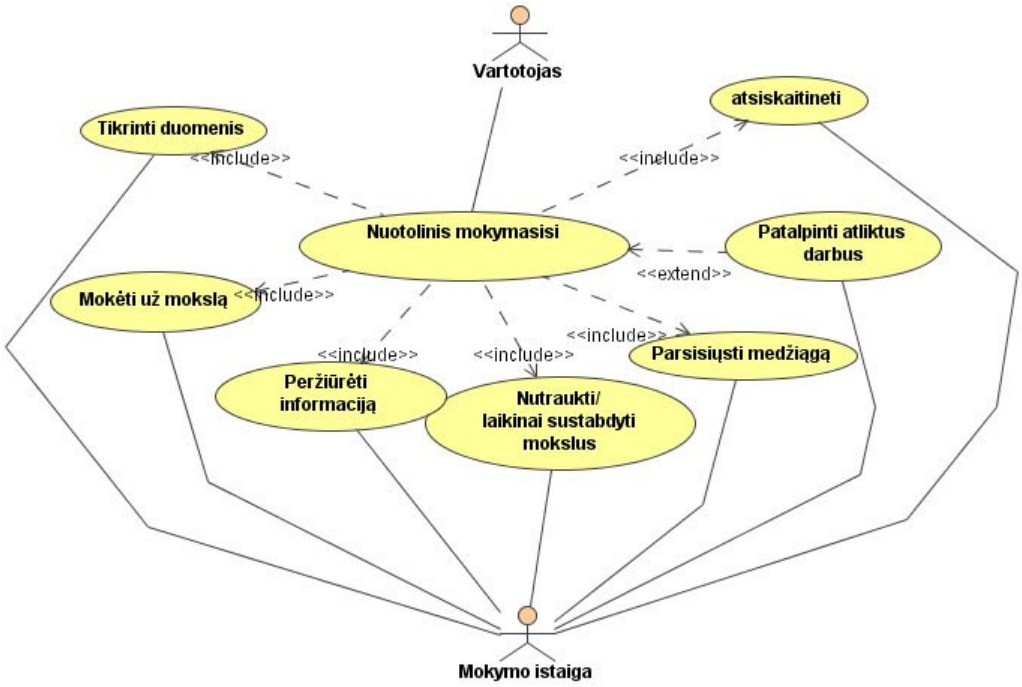


3.5 pav. Nuotolinės mokymosi sistemos modelis sudarytas remiantis universalia metodika



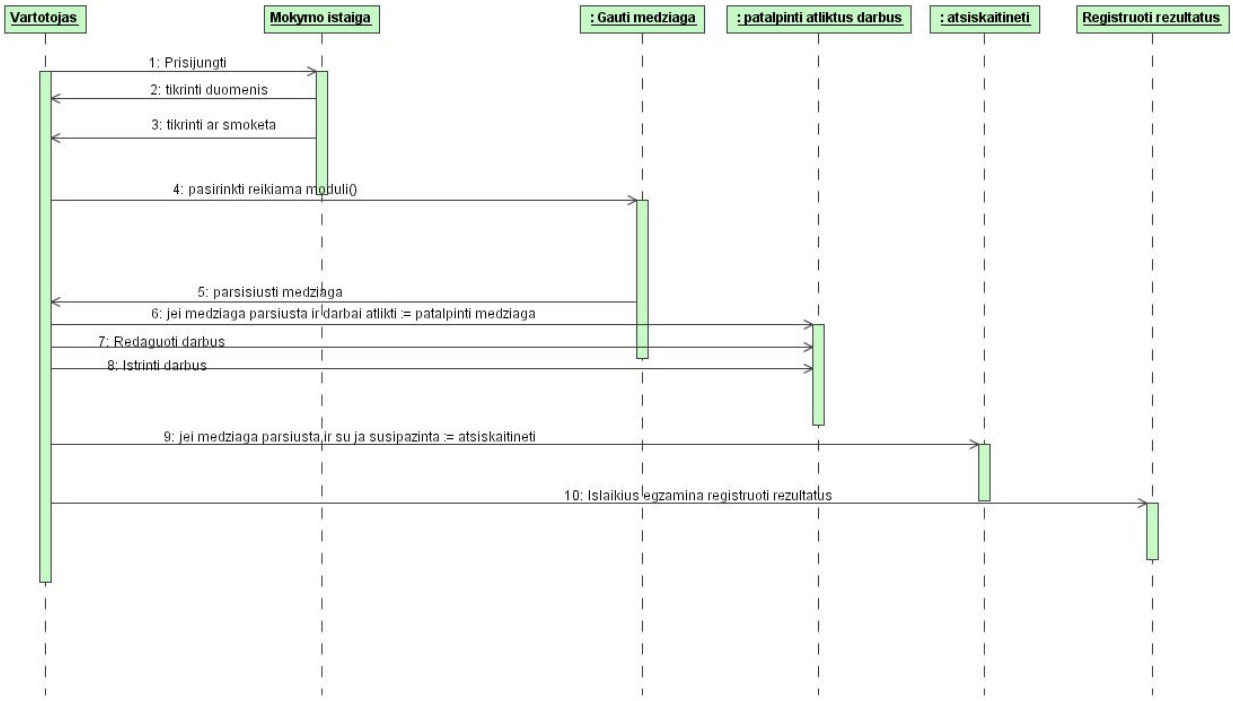
3.6 pav. Duomenų modelis

Greta veiklos diagramų UML 2.0 kalboje yra išsamus priemonių rinkinys informacinei sistemai projektuoti. Kaip, pavyzdžiui, gali būti modeliuojami įvairaus detalizavimo lygio panaudojimo atvejai:



3.7 pav. Mokymosi semestro metu panaudojimo atvejų diagrama

UML 2.0 galima pavaizduoti ir operacijų sekas, kurios vyksta proceso eigoje:



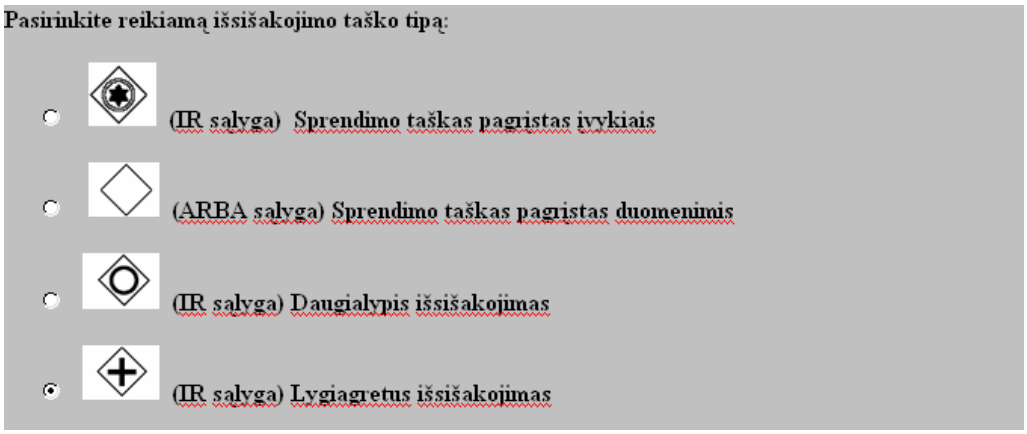
3.8 pav. Mokymosi semestro metu sekų diagrama

Papildžius UML 2.0 BPMN stereotipais (1 lentelė, 2 lentelė) turėtume universalią metodiką, kuri atitiktų daugelį šiuolaikinių reikalavimų ir būtų lengvai prieinama tiek analitikams, tiek projektuotojams, tiek vartotojams.

3.2. Vartotojo sąsajos šablonai

Naudojantis darbo metu sudaryta universalia metodika bei išanalizavus atvaizdavimą į BPEL galima sudaryti CASE įrankio papildymo projektą. Galimas toks modeliavimo scenarijus:

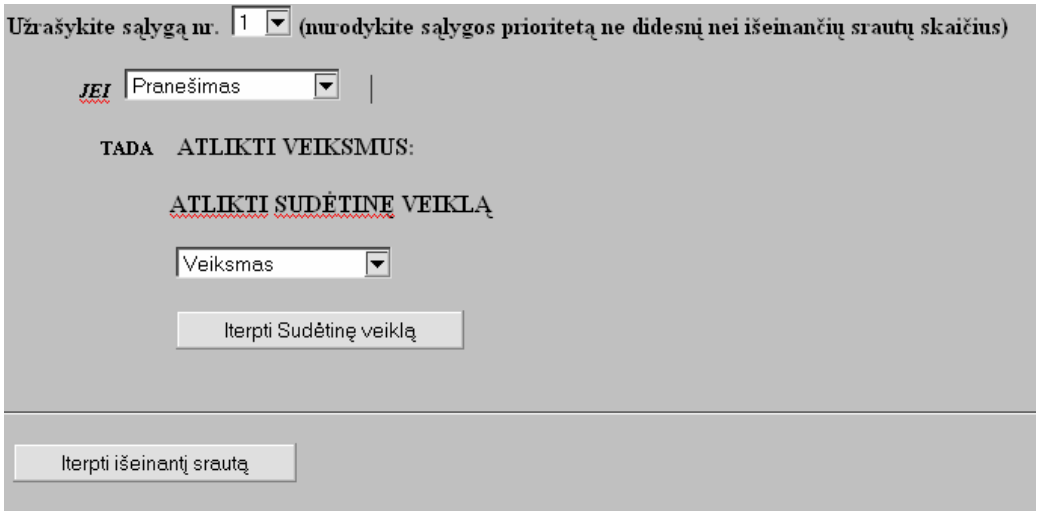
Modeliuojant kiekvienam sprendimo taškui turėtų būti pateikiamas specifikacijos langas, kuriame būtų galima pasirinkti išsišakojimo tipą.



3.9 pav. Sprendimo taškų tipų forma

Pasirinkus išsišakojimo tipą prie kiekvieno išeinančio srauto tikslinga pateikti specifikacijos langus (atitinkamai pagal išsišakojimo tipą), kuriuose galima apibrėžti įvairias taisykles bei sudėtines veiklas.

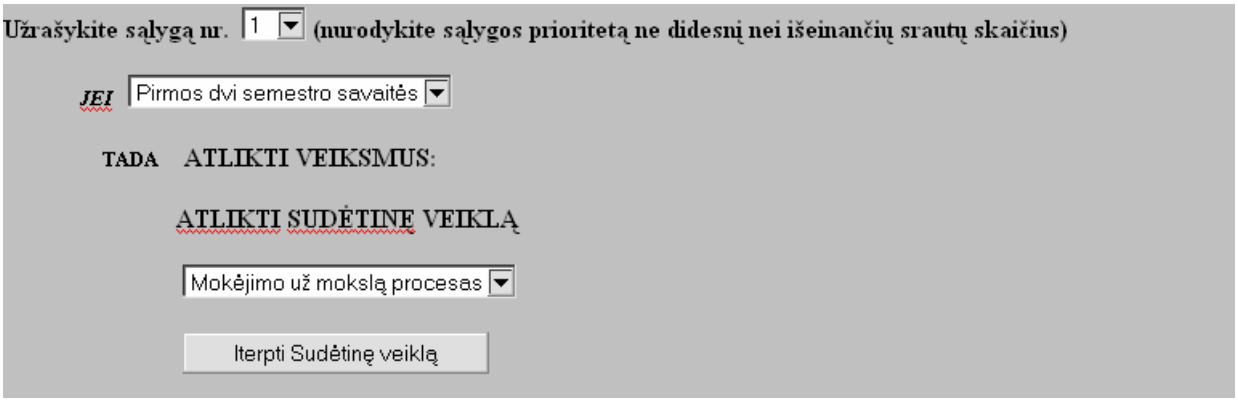
Taisykles sekoms vartotojas galėtų įvesti bet kuria tvarka, tik įvedimo eigoje sąlygai būtų priskiriamas prioritetas. Jei išsišakojimo taškas pasiriktas kaip lygiagretus arba pagrįstas įvykiais, specifikacijos forma galėtų atrodyti taip:



3.10 pav. Lygiagretaus/pagrįsto įvykiais sprendimo taško specifikacijos forma

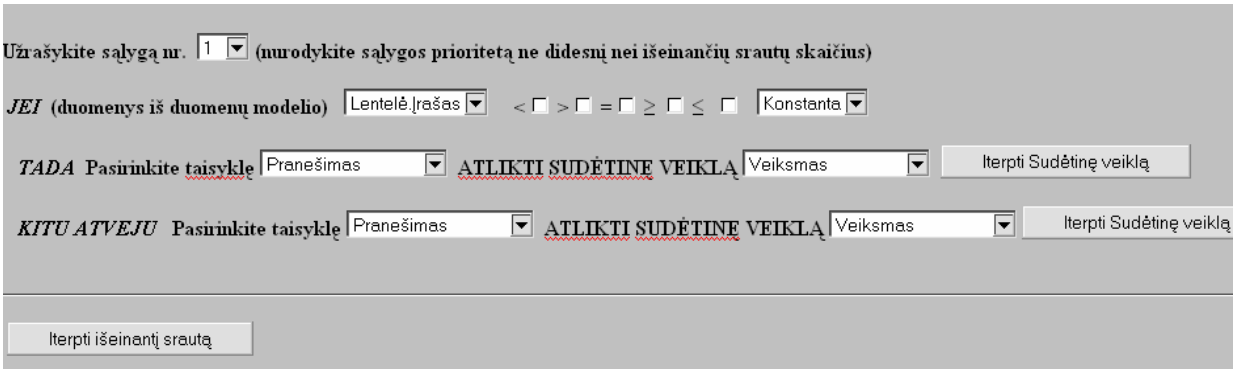
Taisykle gali būti: pranešimas, ryšys, kompensavimas, klaida, priverstinė pabaiga, laiko elementas, tuščias veiksmas. Sudėtine veikla gali būti: veiksmas, ciklas, tuščias veiksmas, išimtis, procesas, grupė, sudėtinis veiksmas.

Taisyklę pasirinkę kaip laiką, o sudėtinę veiklą kaip veiksmą turėtume tokį formos pavyzdį:



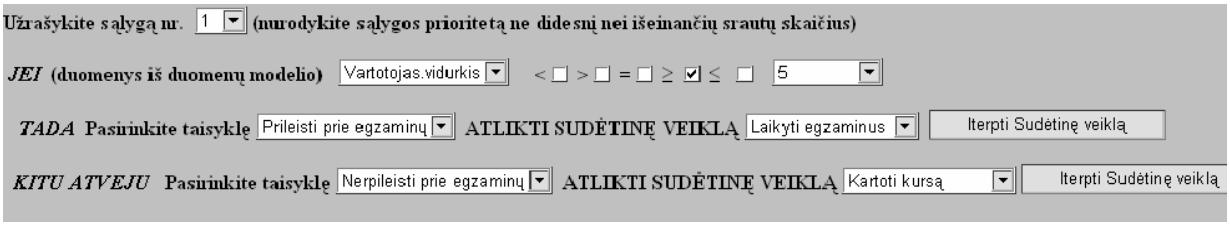
3.10a pav. Lygiagretaus/pagrįsto įvykiais sprendimo taško specifikacijos formos pavyzdys

Jei išsišakojimo taškas būtų pasiriktas kaip pagrįstas duomenimis arba daugialypis išsišakojimas specifikavimo forma atrodytų taip:



3.11 pav. Daugialypio/pagrįsto duomenimis sprendimo taško specifikacijos forma

Taisyklę pasirinkę kaip pranešimą, sudėtinę veiklą kaip procesą turėsime tokį formos pavyzdį:

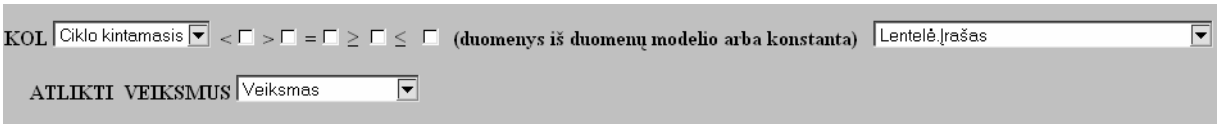


3.11a pav. Daugialypio/pagrįsto duomenimis sprendimo taško specifikacijos formos pavyzdys

Pasirinkus atitinkamą taisyklę ar sudėtinę veiklą vartotojui turėtų būti pateikiamas naujas specifikacijos langas kuriame nurodomi reikiami modeliavimo duomenys. Vartotojui užpildžius

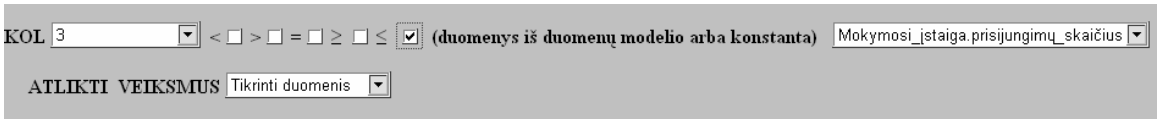
reikiamus laukus, duomenys būtų paimami iš specifikacijos lango ir patalpinami į reikiamus BPEL kodo laukus bei į tam tikras taisyklių lenteles. Patalpinimas į taisyklių lenteles reikalingas tam, kad analogiškas taisykles būtų galima pasirinkti pakartotinai.

Jei sudėtinė veikla/taisyklė būtų pasirinkta kaip ciklas turėtume tokį formos langą:



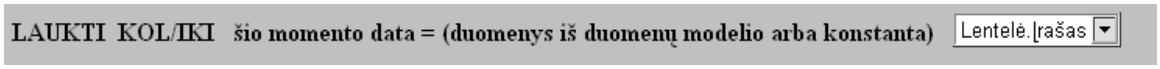
3.12 pav. Ciklo specifikacijos forma

Duomenų tikrinimo prisijungimo metu ciklo užpildyta forma atrodytų taip:



3.12a pav. Ciklo specifikacijos formos pavyzdys

Jei sudėtinė veikla/taisyklė būtų pasirinkta kaip laiko elementas turėtume tokį formos langą:



3.13 pav. Laiko apribojimo specifikacijos forma

Laiko apribojimo iki semestro pradžios datos forma atrodytų taip:

3.13a pav. Laiko apribojimo specifikacijos formos pavyzdys

Jei sudėtinė veikla būtų pasirinkta kaip veiksmas forma būtų tokia:

3.14 pav. Veiksmo specifikacijos forma

Veiksmo, mokėjimas už mokslą, pavyzdys atrodytų taip:

3.14a pav. Veiksmo specifikacijos formos pavyzdys

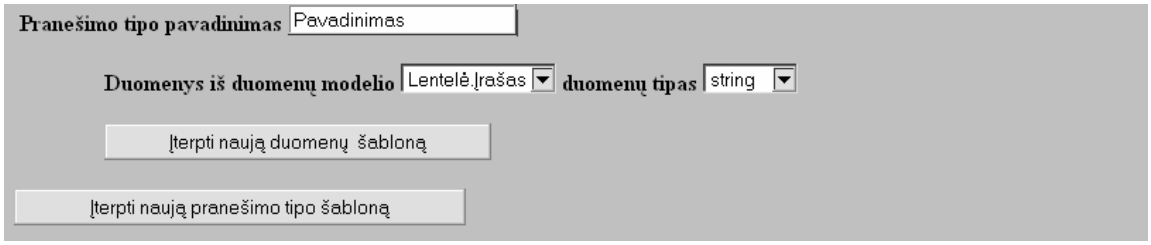
Jei sudėtinė veikla/taisyklė pasirinkta kaip pranešimas, kompensavimas, ryšys, klaida ir kt. tuomet būtų pateiktas langas, kuriame būtų galima arba pasirinkti jau sukurtą taisyklę arba sukurti naują.

3.15 pav. Veiksmo specifikacijos forma

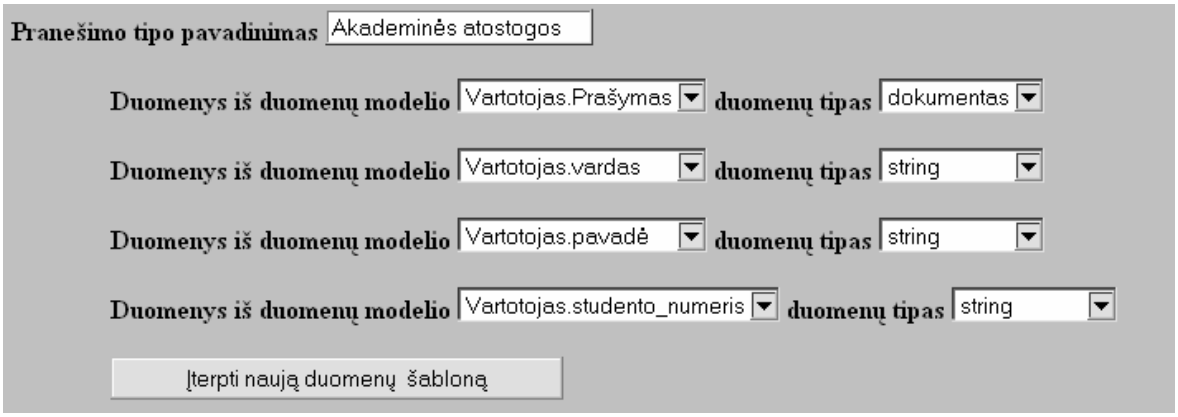
Pasirinkus „Pasirinkti sukurtą veiklos taisyklę“ būtų galima pasirinkti jau sukurtą taisyklę iš specialios duomenų bazės. Pasirinkus „Sukurti naują taisyklę“ būtų pateikiami vienas po kito langai kuriuose vartotojas turėtų užpildyti reikiamus laukus, kad būtų sukurta atitinkamo tipo

taisyklė ir duomenys automatiškai perduodami į BPEL kodą. Žemiau esančiuose paveiksluose pateikiama formų, skirtų pranešimo taisyklės užrašymui, seka. 3.16 paveiksle pirmoji pranešimo tipo kūrimo forma. Ši pranešimo tipo forma užpildoma pirmiausia. Pranešimo tipas susieja veiklos modelį su dalykine sritimi.

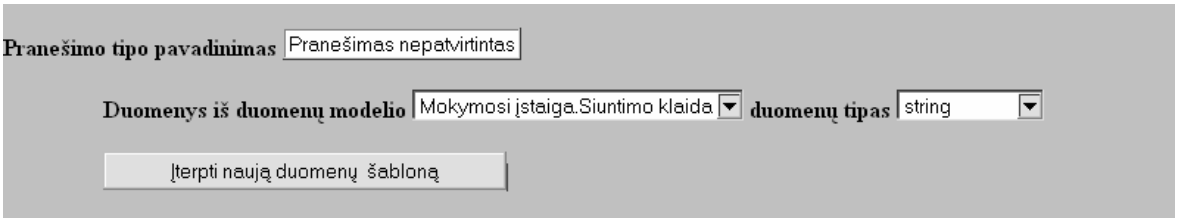
Visame poskyryje po kiekvieno tipo forma pateikta keletas formos užpildymo pavyzdžių.



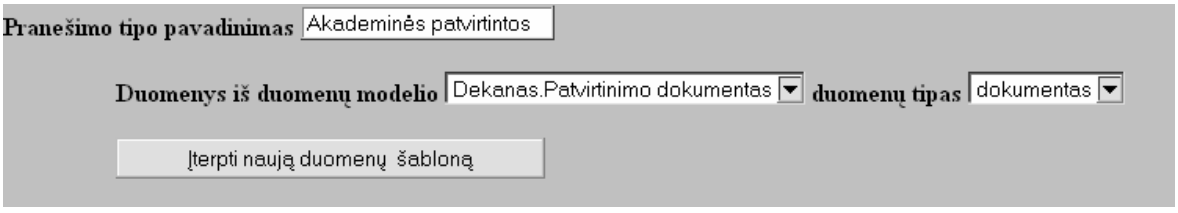
3.16 pav. Pranešimo tipo specifikacijos forma



3.16a pav. Pranešimo tipo formos a pavyzdys

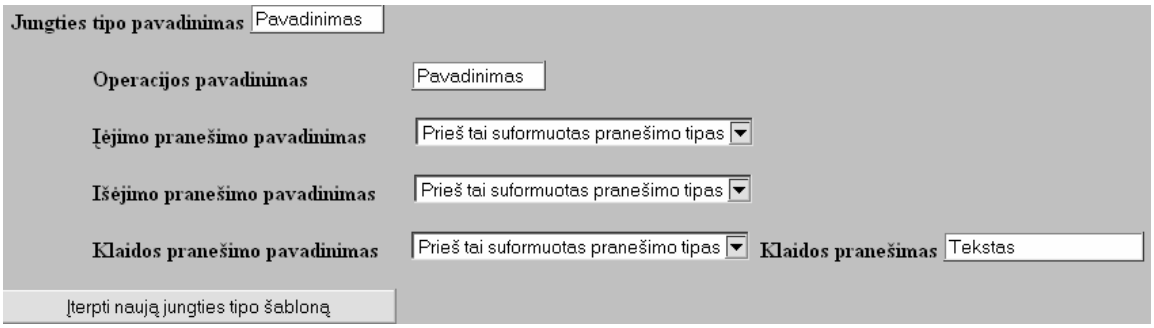


3.16b pav. Pranešimo tipo formos b pavyzdys

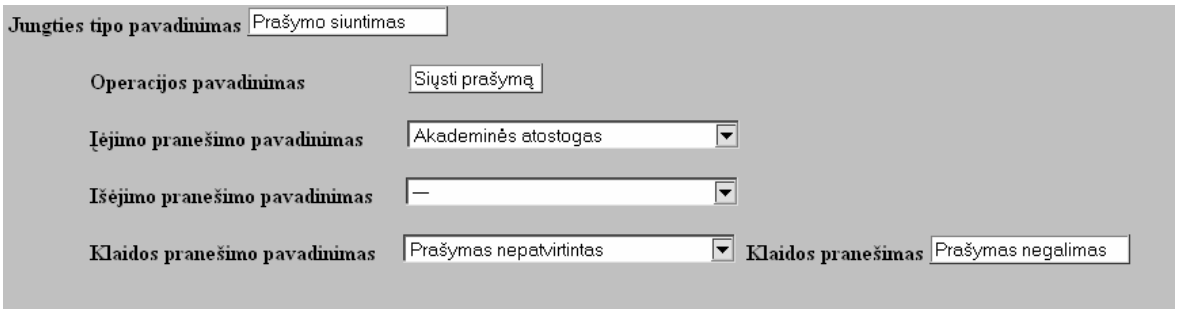


3.16c pav. Pranešimo tipo formos c pavyzdys

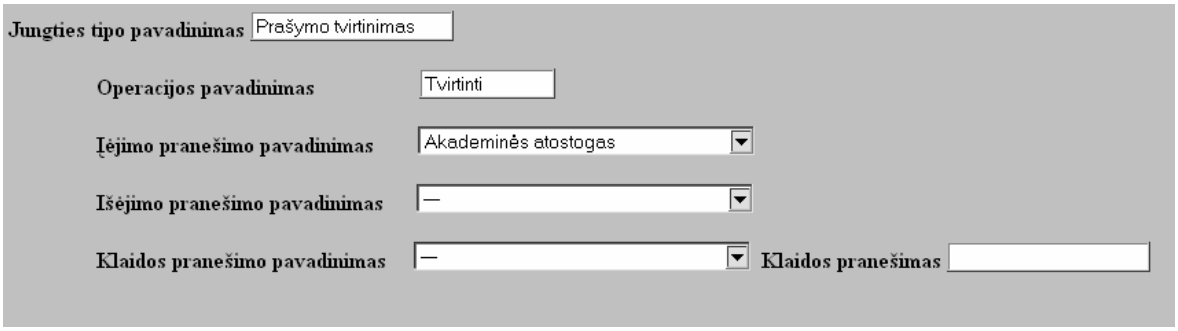
Užpildžius pranešimo tipo formas turėtų būti pateikiami jungties tipo aprašymo langai. Kad užpildyti šiuos langus prieš tai turi būti suformuoti visi reikalingi pranešimų tipai. Visų laukų esančių jungties tipo formoje pildyti nebūtina. Užpildomi tie kurie reikalingi taisyklėms užrašyti.



3.17 pav. Jungties tipo specifikacijos forma

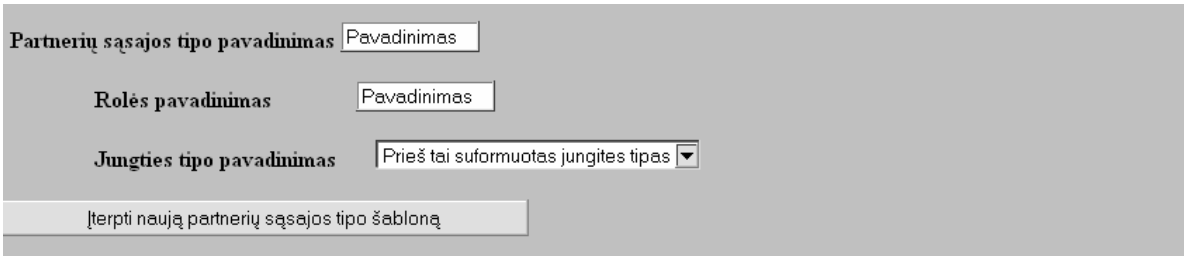


3.17a pav. Jungties tipo specifikacijos formos a pavyzdys

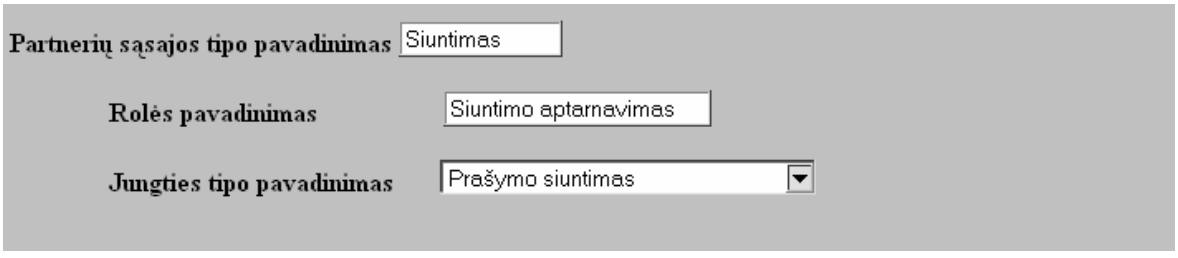


3.17b pav. Jungties tipo specifikacijos formos b pavyzdys

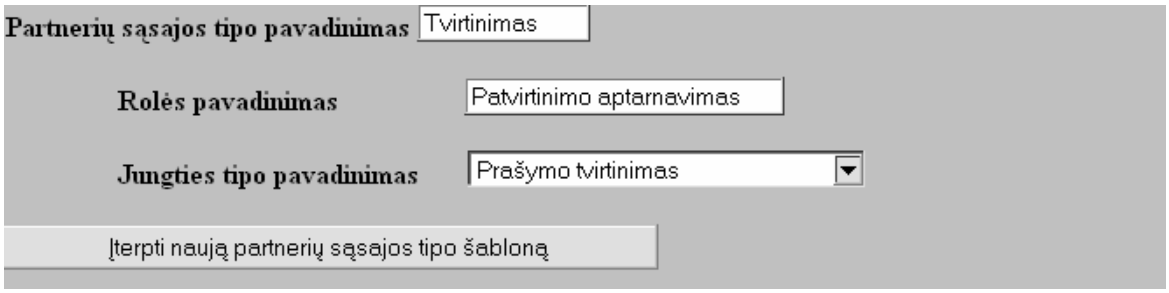
Kai užpildoma jungties tipo forma, vartotojui tikslinga pateikti partnerių sąsajos tipo langus:



3.18 pav. Partnerių sąsajos tipo specifikacijos forma

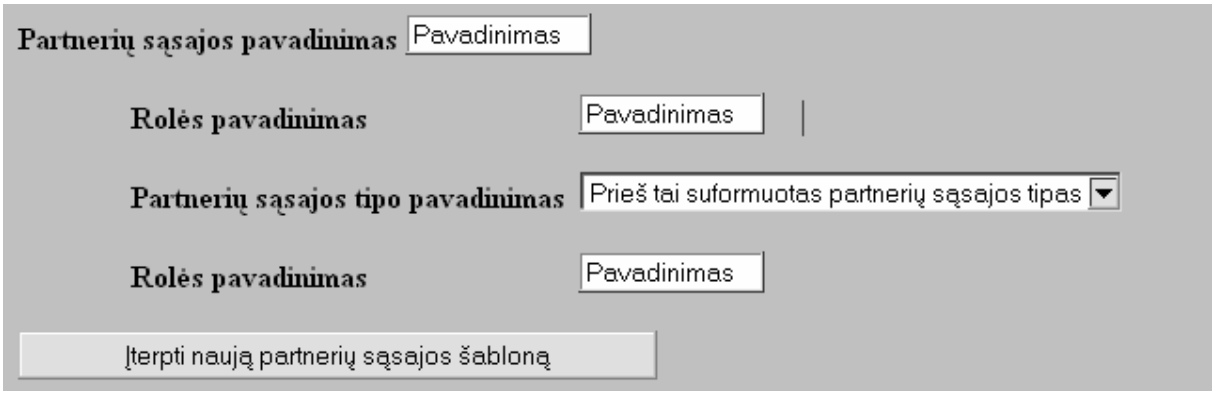


3.18a pav. Partnerių sąsajos tipo specifikacijos formos a pavyzdys

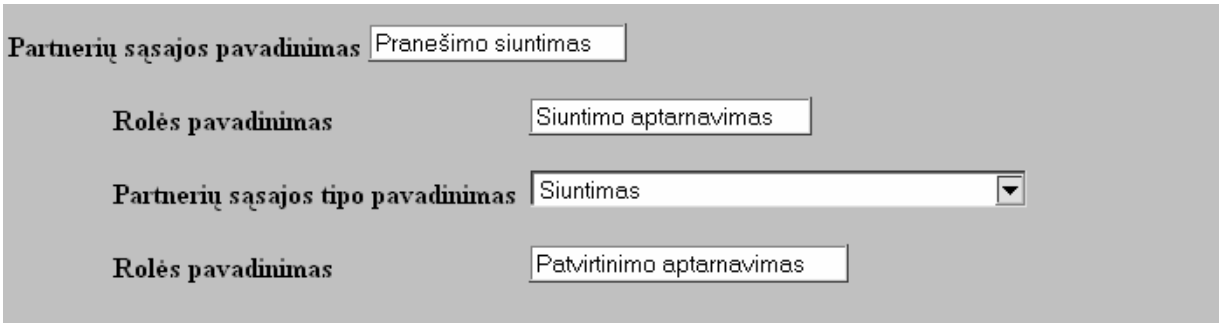


3.18b pav. Partnerių sąsajos tipo specifikacijos formos b pavyzdys

Suformavus partnerių sąsajos tipą reikia aprašyti ir pačią partnerių sąsają:

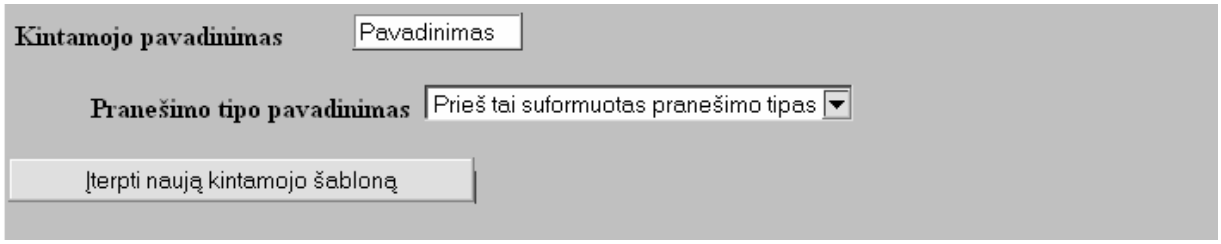


3.19 pav. Partnerių sąsajos specifikacijos forma

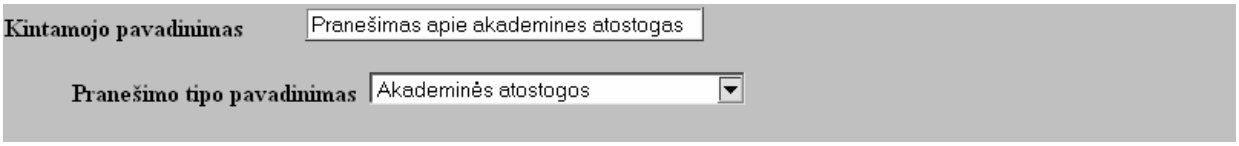


3.19a pav. Partnerių sąsajos specifikacijos formos pavyzdys

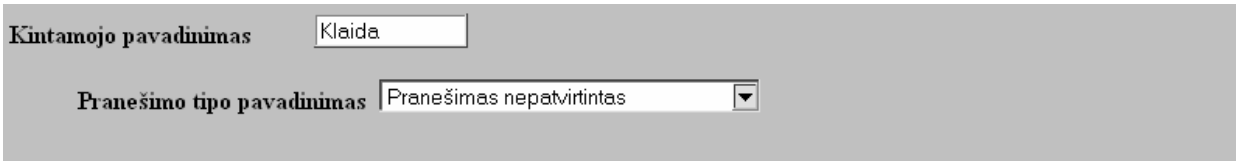
Paskutinis žingsnis kuriant pranešimo taisykles tai kintamųjų formavimas. Kintamieji formuojami kiekvienam taisyklei reikalingam pranešimo tipui, kuris buvo formuojamas 3.16 paveikslo formoje:



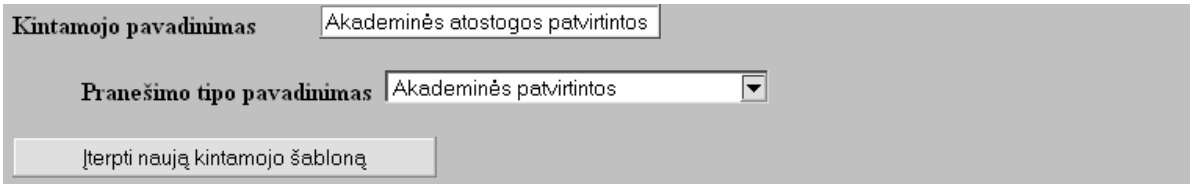
3.20 pav. Kintamojo specifikacijos forma



3.20a pav. Kintamojo specifikacijos formos a pavyzdys



3.20b pav. Kintamojo specifikacijos formos b pavyzdys



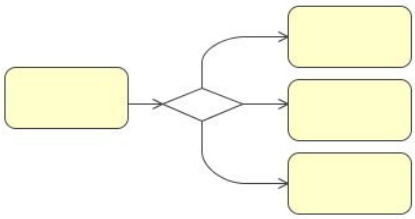
3.20c pav. Kintamojo specifikacijos formos c pavyzdys

Plačiau apie tai kokie duomenys kokia tvarka turi būti užpildomi formose pateikta 2.2, 2.3, 2.4 skyriuose.

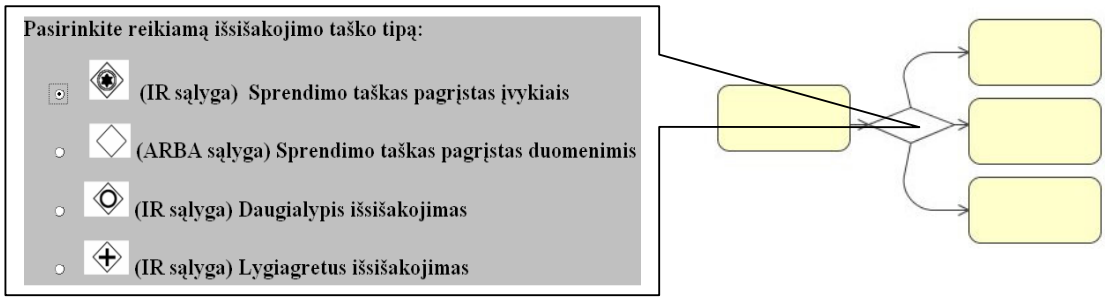
3.3. Modeliavimo CASE įrankis pagal siūlomą modeliavimo metodiką

Šiame skyriuje aprašoma kaip turėtų atrodyti modeliavimo įrankis ir elementų specifikacijos modeliuojant. Tokios sąsajos prototipas įgyvendintas įvedant stereotipus į MagicDraw 9.0 (1 lentelė, 2 lentelė). Visa tai bus atliekama pagal 2.2. poskyryje siūlomus vartotojo sąsajos modeliavimo šablonus.

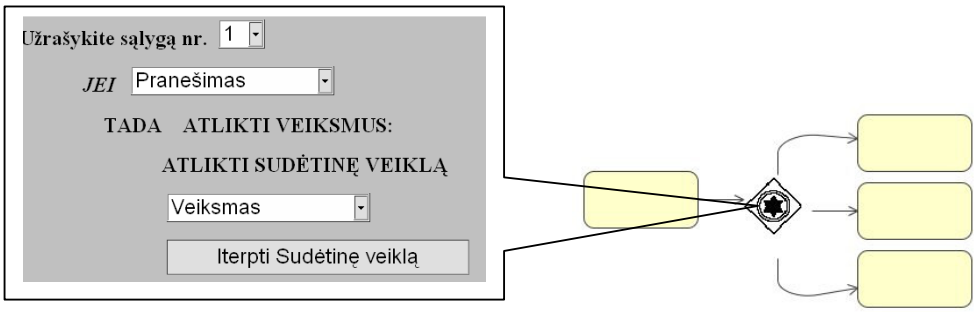
1. Sudarome modelio pirminę struktūrą:



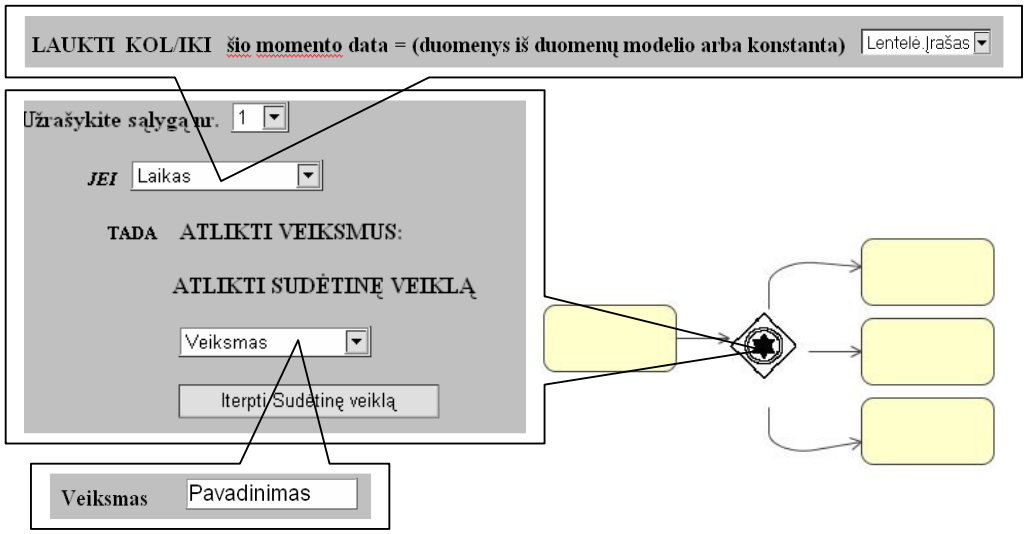
2. Pasirenkame išsišakojimo tipą. Šiuo atveju pasirenkamas sprendimo taškas pagrįstas įvykiais:



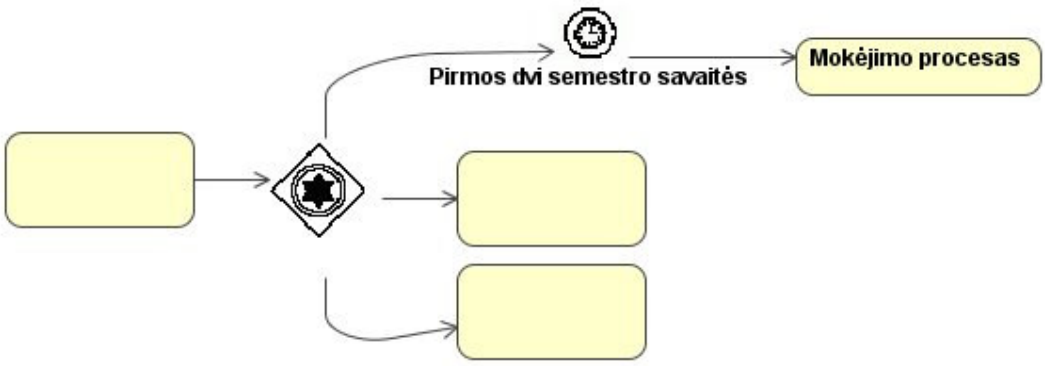
3. Užrašomos išeinančių srautų taisyklės, pasirinkus norimo iššišakojimo specifikacijos langą. Paspaudus reikiamą mygtuką iššoka specifikacijos langas su tokio tipo informacija, koks sprendimo taškas buvo pasirinktas.



4. Priklausomai nuo to kokia taisyklė ir kokia sudėtinė veikla atliekama atitinkamai pateikiami nauji langai, kuriose vartotojas užpildo reikiamą modeliavimo informaciją.



5. Užpildžius specifikaciją turėsime:



6. Analogiškai veiksmai atliekami ir su kitais iššišakojimais bei sprendimų taškais. Užpildžius reikiamus laukus informacija patalpinama į taisyklių lenteles, bei atitinkamai į vykdomosios kalbos BPEL reikiamas kodo vietas. Modeliuojant iš karto formuojama taisyklių duomenų bazė bei BPEL kodas.

IŠVADOS

Norint sukurti universalią modeliavimo metodiką buvo išanalizuotos UML, BPMN ir BPEL kalbos. Pasiūlyta UML 2.0 papildyti BPMN galimybėmis tokiomis kaip laiko elementų vaizdavimas, ciklų vaizdavimas, sprendimo taškų įvairovė, kompensacinių srautų vaizdavimu ir kt. Siūloma notacija ir taisyklių specifikacijos verslo ekspertams, analitikams ir programuotojams leistų efektyviai bendradarbiauti elektroninio verslo procesų, paslaugų ir dokumentų kūrime.

Sukurti papildomi stereotipai ir išbandyti trijų tipų eksperimentiniuose modeliuose: UML 2.0 su anotacijomis, UML 2.0 su konteineriais ir modelis pagal naują metodiką UML 2.0 papildžius BPMN elementais. Modelis, sudarytas pagal naują metodiką, aiškus, pilnai atvaizduotas, taisyklės vaizdžiai ir išsamiai užrašytos.

Kad modelis būtų maksimaliai priartintas prie realizacijos bei dalykinė sritis būtų susieta su veiklos proceso modeliu buvo analizuota vykdomoji kalba BPEL. Pagal pateiktą atvaizdavimo į BPEL metodiką sudaryti apibendrinti specifikacijų eskizai vartotojui natūralia bei OCL kalbomis, padėsiantys sudaryti korektišką veiklos proceso modelį.

Nauja sudaryta notacija CASE įrankyje turėtų papildomas specifikacijas aprašyti verslo proceso taisyklėms, kurios būtų saugomos bazėje ir atvaizduojamos į vykdomosios BPEL kalbos konstrukcijas. Standartinis notacijos pagrindas leistų lengviau susieti verslo procesų modelius tiek su BPEL, tiek su pačių elektroninių paslaugų ir dalykinės srities modeliais. Pastaroji sąsaja užtikrinama siūlomose verslo taisyklių specifikacijose naudojant duomenų modelio elementus.

Šiuo metu intensyviai kuriami CASE įrankiai, siekiantys patenkinti daugybės notacijų, sąsajų ir kodo generavimo reikalavimus. Šiame darbe pasirinktas MagicDraw, kadangi jis gerai palaiko OCL specifikavimo ir sintaksės tikrinimo priemones; plėtimo API; leidžia lengvai įvesti naujus grafinius stereotipus.

Darbe analizuoti modeliai sukurti su MagicDraw įvedant papildomus stereotipus. Buvo pasiūlyta papildyti MagicDraw veiklos modeliavimo elementų specifikacijas, kurios leistų aprašyti verslo taisyklių naudojamų duomenų modelio elementus. Tokiu būdu sudarytą modelį būtų galima konvertuoti į vykdomąją kalbą.

LITERATŪRA

[1] Amsden J., Gardner T., Griffin C., Iyengar S. *Draft UML Profile for Automated Business Processes with a mapping to BPEL 1.0 version 1.1* [interaktyvus] IBM. 2002 gegužė, [žiūrėta 2004 09 21]. Prieiga per internetą: http://dwdemos.dfw.ibm.com

[2] *Unified Modeling Language: Superstructure version 2.0* [interaktyvus] Object Management Group. 2003 sausis, [žiūrėta 2004 11 20]. Prieiga per internetą: http://www.u2-partners.org

[3] Barzdins J., Kalnins A., Celms E. *UML business modeling profile* [interaktyvus] IMCS. [žiūrėta 2005 04 11]. Prieiga per internetą: http://melnais.mii.lu.lv

[4] White S. A. *Buisiness porcess modeling notation* [interaktyvus] IBM Corporation. 2003 rugpjūtis, [žiūrėta 2004 11 25]. Prieiga per internetą: http://bpmi-notation-wg.netfirms.com

[5] White S. A. *Buisiness porcess modeling notation version 1.0* [interaktyvus] IBM Corporation. 2004 kovas, [žiūrėta 2005 01 02]. Prieiga per internetą: http://www.bpmn.org

[6] *Business process execusion language for Web services version 1.1* [interaktyvus] Microsoft, IBM. 2003 gegužė, [žiūrėta 2005 01 18]. Prieiga per internetą: http://www-128.ibm.com/

[7] Clark J., DeRose S. *XML Path Language (XPath) Version 1.0* [interaktyvus]. 2003 gegužė, [žiūrėta 2005 03 21]. Prieiga per internetą: http://www.w3.org

[8] *Business process execution language for web servises* [interaktyvus] Microsoft, IBM. 2003 gegužė, [žiūrėta 2004 12 12]. Prieiga per internetą http://www.oasis-en.org

[9] White S. A. *Process Modeling Notations and Workflow Patterns* [interaktyvus] IBM Corporation. 2003 gruodis, [žiūrėta 2005 04 11]. Prieiga per internetą: http://www.bpmn.org

[10] Mantell K. *From UML to BPEL* [interaktyvus] IBM. 2003 rugsėjis, [žiūrėta 2005 03 21]. Prieiga per internetą: http://www-128.ibm.com

[11] *BPEL Tutorial 3: Manipulating XML Documents in BPEL* [interaktyvus] Oracle. [žiūrėta 2005 04 11]. Prieiga per internetą: www.oracle.com

[12] Biron P. V., Permanente K., Malhotra A. *XML Schema Part 2: Datatypes Second Editon* [interaktyvus]. 2004 lapkritis, [žiūrėta 2005 01 02]. Prieiga per internetą: http://www.w3.org

[13] Simanaitytė K., Nemuraitė L. *Veiklos procesų modeliavimas naudojant veiklos procesų modeliavimo notaciją*: 9 – oji tarpuniversitetinė magistrantų ir doktorantų konferencija [2004m. balandžio 15 d., Kaunas]: pranešimų medžiaga. Kaunas, 2004, p. 254 – 259.

[14] *WS-BPEL* [interaktyvus] *OASIS, WS-BPEL Technical Committee*. [žiūrėta 2005 01 05] Prieiga per internetą: http://www.ebpml.org/bpel4ws.htm

[15] Warmer J., Kleope A. Object *Constraint Language: The Getting Yor Models Read for MDA, Second Editon*. 2003 rugpjūtis.

[16] *UML 2.0 OCL specification* [interaktyvus] Object Management Group. 2003 spalis, [žiūrėta 2005 01 02]. Prieiga per internetą: http://www.omg.org

[17] *UML 2.0 OCL specification* [interaktyvus] OMG. 2003 spalis, [žiūrėta 2005 04 11]. Prieiga per internetą: http://www.omg.org

[18] *EA users manual version 4.0* [interaktyvus] Sparx Systems Pty. Ltd. 1998 – 2004, [žiūrėta 2004 10 10] Prieiga per internetą: http://www.sparxsystems.com.au

[19] *Users manual version 7.2* [interaktyvus] No Magic Inc. 2003 gruodis, [žiūrėta 2004 09 03]. Prieiga per internetą: www.magicdraw.com

[20] *Users manual version 9.0* [interaktyvus] No Magic Inc. 2005 kovas, [žiūrėta 2005 03 02] Prieiga per internetą: www.magicdraw.com

[21] Pašilskytė I., Nemuraitė L. *Verslo procesų modeliavimo kalbų analizė ir specifikavimo priemonių sukūrimas*: Informacinės technologijos 2005 iš konferencijų ciklo „Lietuvos mokslas ir pramonė“ [2005m. sausio 26-27 d., Kaunas]: pranešimų medžiaga. Kaunas, 2005, p. 525 – 533.

[22] Pašilskytė I., Nemuraitė L. *Išplėstoji UML notacija elektroninio verslo procesams modeliuoti ir specifikuoti*: Informacinės technologijos verslui 2005 [2005m. gegužės 27 d., Kaunas]: pranešimų medžiaga.

TERMINŲ IR SANTRUMPŲ ŽODYNAS

UML (*Unified Modeling Language*)	Universalioji modeliavimo kalba.
BPMN (*Business Process Modeling Notation*)	Specializuota verslo procesų modeliavimo notacija.
BPLE4WS/BPEL (*Business Process Execution Language for Web Services*)	Verslo procesų vykdomoji kalba.
ODBC (*Open Database Connectivity*)	Standartinis prisijungimo prie domenų bazių metodas.
XML (eX*tensible Markup Language*)	Išplėstoji žymių kalba.
XPath (*XML Path Language*)	Išplėstosios žymių kalbos funkcijų užrašymo kalba.
CASE (*Computer Aided Software Engineering*)	Kompiuterizuotas programinis įrankis
OCL (*Object Constrain Language*)	Apribojimų kalba. Universalios modeliavimo kalbos dalis.
WSDL (*Web Service Definition Language*)	Tinklo paslaugų užrašymo kalba.
API (*Application Program Interface*)	Plečiama (lengvai pritaikoma) programos sąsaja.
ACID (*Isolation Duration*)	Transakcijų reikalavimai: dalumas, vientisumas, izoliacija, trukmė.
E. verslas	Elektroninis verslas

1 PRIEDAS. Straipsniai

VERSLO PROCESŲ MODELIAVIMO KALBŲ ANALIZĖ IR SPECIFIKAVIMO PRIEMONIŲ SUKŪRIMAS

Inga Pašilskytė, doc. Lina Nemuraitė

Kauno Technologijos universitetas, Informacijos sistemų katedra

Siekiant, kad verslo modeliavimo, projektavimo, analizavimo procesas būtų suprantamas tiek informacinių technologijų specialistams, tiek verslo dalyviams, vis dažniau naudojamos grafinio modeliavimo kalbos. Grafinio modeliavimo privalumai – standartiniai žymėjimai, vieninga terminologija, intuityvus supratimas. Kadangi atvaizduoti verslo procesą visuose jo kompiuterizavimo etapuose vienos kalbos nepakanka, buvo analizuojamos trys veiklos procesų modeliavimo kalbos: UML 2.0, BPMN bei BPEL4WS, gilinantis į jų galimybes elektroninio verslo procesams modeliuoti bei vykdyti. Išsamiam verslo proceso modeliui sudaryti UML 2.0 siūloma papildyti BPMN elementais, sukuriant tam skirtus stereotipus. Taip pat iškeliama idėja išplėsti grafinę proceso apibrėžimo sąsają, kurioje proceso taisyklės būtų susiejamos su dalykinės srities objektais.

1. Įvadas

Atsiradus verslo modeliavimo standartizavimo poreikiui, sukurta daug įvairių modeliavimo standartų, kalbų, kurios akcentuoja skirtingas verslo modelių savybes [1] ir nėra nė vienos, kuri tenkintų visus verslo modeliavimo poreikius. Be to, daugelis kalbų yra tekstinio pavidalo ir verslo ekspertams neprieinamos. Nuolat keičiantis reikalavimams, modeliavimo metodika turi prie jų prisitaikyti. Viena iš plačiausiai naudojamų grafinio modeliavimo kalbų yra UML. ji nuolat tobulinama, kad būtų palengvinamas projektavimo procesas, ir suteikta galimybė jį detalizuoti iki realizuotojų lygio. Naujausia UML versija yra 2.0 [2], kurioje išplėstos galimybės verslui modeliuoti.

Plačias verslo modeliavimo galimybes palaiko naujas verslo modeliavimo standartas BPMN (Business process modelling notation) [3][4], kuris palengviną projektavimo kelią nuo projektavimo iki realizavimo, ir projektavimo procesą padaro prieinamą ir aiškų visiems verslo kūrimo dalyviams.

UML 2.0 versijos veiklos diagramos savo galimybėmis tolygios BPMN, tačiau neturi tokios išbaigtos stereotipų aibės. Tačiau UML turi tų privalumų, kad ji leidžia modeliuoti ne tik verslo procesus, bet ir duomenis bei paslaugas, kurios reikalingos verslo procesams įgyvendinti. Šaime darbe yra siūloma UML 2.0 papildyti BPMN stereotipais, taip modelį priartinant prie realizacijos. BPMN siūloma plati stereotipų aibė supaprastina modeliavimo procesą, padaro jį aikesnį ir lengvesnį.

Siekiant grafinę proceso apibrėžimo sąsają sieti su dalykinės srities objektais buvo pasirinkta BPEL4WS(Business Process Execution Language for Web Services)[5] vykdomoji kalba. Šios vykdomosios kalbos pagalba UML 2.0 BPMN stereotipais papildyti elementai atvaizduojami į BPEL4WS taip susiejant dalykinę sritį su grafiniais veiklos procesais. Analizuojant transformuotą kodą sudaromi šablonai vartotojo procesui palengvinti. Pagal sudarytus šablonus sudaromas išbaigtas struktūrinis veiklos proceso metamodelis.

Šaime darbe trumpai analizuojamos UML 2.0 ir BPMN. Šių notacijų pranašumai ir skirtumai parodomi atvaizduojant to paties proceso modelį UML 2.0 bei UML 2.0 papildžius BPMN stereotipai. Straipsnio pabaigoje trumpai analizuojamas transformavimas į vykdomąją kalbą BPEL4WS ir sudaromas struktūrinis veiklos proceso metamodelis. Modeliavimas atliktas įrankiais Enterprise Architect 4.0[6] bei Magic Draw 7.2[7] . Enterprise Architect 4.0 buvo pasirinktas, todėl, kad jis palaiko UML 2.0, juo patogu braižyti šio standarto modelius. O Magic Draw 7.2 buvo pasirinktas, todėl, kad šiuo įrankiu patogu įvesti naujus stereotipus. Šis įrankis buvo naudojamas UML 2.0 modeliams papildytiems BPMN stereotipais braižyti.

2. Trumpa UML 2.0 ir BPMN notacijų analizė

2.1. UML

UML modeliavimo kalba gana paplitusi, lengvai išmokstama ir patogi realizavimui, specifikavimui, dokumentavimui. UML modeliavimo kalbai būdinga diagramų įvairovė, todėl ši modeliavimo kalba labai lanksti ir patogi įvairiam projektavimui. UML modeliavimo kalboje diagramos skirstomos į tris kategorijas: statines, dinamines bei organizavimo, valdymo. Ši modeliavimo kalba nuolat tobulina ir papildoma. Nauja modeliavimo kalbos UML, versija UML 2.0. UML 2.0 suteikia galimybę taip suprojektuoti sistemą, kad ji būtų labai artima realizuojamai sistemai. Šis standartas papildytas elementais, skirtais projektuoti verslo modeliams (duomenų saugykla, daugybė taškų tipų, veiklos suskaidymas, pertraukimas ir kt.). UML 2.0 yra universali kalba, ir verslo procesų modeliavimas yra tik viena jos naudojimo krypčių. Kadangi vienas iš darbo tikslų yra papildyti šią notaciją naujais veiklos procesų stereotipais, dėmesys koncentruojamas į UML 2.0 veiklos diagramas.

Veiklos diagramos yra trijų lygių (trečio lygio šiame darbe nenagrinėjamos):

Pirmo lygio diagrama abstrakčiai vaizduoja veiklą.

Antro lygio diagramos gali būti dviejų tipų : tarpinės veiklos (IntermediateActivities) diagramos aprašo veiklos grafus, veiklos valdymo bei duomenų srautus; struktūrinės veiklos (StructuredActivities) leidžia aprašyti darinius, panašius į naudojamus programose; trečio lygio diagramose aprašomos tokios konstrukcijos kaip briaunos svoris, srautų suskaidymas ir t.t

3. Veiklos procesų modeliavimo notacija BPMN

BPMN yra naujas modeliavimo standartas. Jis suteikia galimybę suprasti vidinius verslo procesus grafiškai juos atvaizduojant ir suteikia galimybę šiuos procesus standartizuoti. BPMN procesų modeliavimo pagrindas ir metodologija. Ji suteikia galimybę žmonėms iš skirtingų įmonių suprasti vieni kitų verslo procesus, sujungti verslus, ar juos suskaidyti į skirtingus. BPMN standartas padeda suprasti visiškai skirtingų verslo procesų gyvavimo ciklą nuo sukūrimo iki vystymo, realizavimo, vykdymo, analizavimo. Taip pat šis standartas suteikia galimybę suprasti bendradarbiavimą ir transakcijas tarp organizacijų. BPMN standartas leidžia numatyti galimas gyvavimo ciklo eigos išimtis, klaidas. Suteikia galimybę naudoti verslo taisykles, ciklus, sprendimų taškus, trigerius.

BPMN yra vienintelio tipo diagrama, kurią galima naudoti įvairiais pavidalais:

Vidinių verslo procesų diagramos(Private (internal) business processes). Tai vidinis įmonės darbų srautų modelis (orkestruotė). Jeigu projektuojamos vidinės verslo diagramos naudojant juostas , tai vidinė diagrama negali išeiti už jos ribų. Jeigu dvi juostos sujungtos, tai reiškia, kad ne jų atskiri elementai, o vidinės diagramos siejasi viena su kita.

Sąveikų procesų diagramos (Abstract (public) processes). Jos rodo, kaip sąveikauja skirtingi verslo procesai (choreografija). Į diagramas įtraukiami tik tie procesai kurie veikia vidinių verslo procesų išorėje.

Bendradarbiavimo proceso diagramos (Collaboration (global) processes). Šios diagramos parodo globalius procesus. Šios diagramos gali būti vaizduojamos be juostų. Veiksmai priklausantys tam pačiam vidiniam procesui šiose diagramose gali būti susieti.

Naudojant BPMN verslo modeliavimo diagramos taip pat gali būti kelių tipų:

- Aukšto lygio vidinių procesų veiklos.
- Detalizuotas vidinis verslo procesas.
- Senas verslo procesas.
- Naujas verslo procesas.
- Detalizuotas verslo procesas su sąsajomis, bei išoriniais verslo dalyviais.
- Du ar daugiau sąveikaujantys verslo procesai.
- Detalizuoto vidinio proceso ryšys su sąsajos procesu.
- Detalizuoto vidinio proceso ryšys su bendradarbiavimo procesu.
- Sąsajos ir bendradarbiavimo procesas.
- Du ar daugiau verslo procesų sąveikaujantys su sąsaja
- Du ar daugiau verslo procesų sąveikaujantys su bendradarbiavimu.
- Du ar daugiau verslo procesų sąveikaujantys su sąsajų ir bendradarbiavimo procesais.

UML 2.0 daugelis elementų atitinka BPMN notaciją, kaip pavyzdžiui: pradžios pabaigos taškas, sprendimų mazgas, išsišakojimas, veiklos pertraukimas, ciklas, jungimų mazgas ir t.t. Bet BPMN kalba turi daugiau elementų skirtų dar labiau supaprastinti verslo projektavimo procesą. Todėl siūloma UML 2.0 papildyti kai kuriais BPMN elementais.

BPMN notacijos verslo modeliavimo elementai, kuriais siūloma papildyti UML 2.0 notaciją

Trigeriai

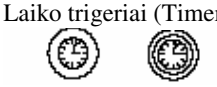
Tarpinis įvykis (Event).

Tarpinis įvykis įvyksta tik po to kai procesas prasidėjęs. Jis turi įtakos proceso eigai, bet nei pradėti, nei nutraukti proceso negali.

Pranešimo trigeriai (Message triggers).

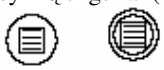
Pranešimo trigeriai gali būti pradžios, tarpiniai ir pabaigos. Pradžios trigeris paleidžia proceso pradžią, tarpinis trigeris tęsia procesą, o pabaigos trigeris veikia jei buvo tarpinis trigeris, ir jis praneša apie proceso pabaigą.

Laiko trigeriai (Timer trigger).



Laiko trigeriai gali būti tik pradžios ir tarpiniai. Jie užduoda tam tikrą laiko ciklą, kurio neviršijus procesas bus tęsiamas.

Taisyklių trigeriai (Rule trigger).



Taisyklių trigeriai gali būti tik pradžios arba tarpiniai. Procesas tęsiamas jei užduota taisyklė yra tenkinama.

Ryšio trigeriai (Link trigger).

Ryšio trigeriai gali būti pradžios, tarpiniai ir pabaigos. Ryšio trigeriai naudojami jei procesas gali prasidėti tik pasibaigus kitam procesui, ar pasiekus tarpinę būseną ir t.t.

Išsišakojantis trigeris (Multiple trrigers).

Išsišakojantis trigeris gali būti pradžios, tarpinis ir pabaigos. Išsišakojantys trigeriai naudojami tada kai gali būti keli įvykio variantai. Procesas paleidžiamas(tęsiamas, baigiamas) kai kuris nors viena iš galimų alternatyvų tenkinama.

Klaidos trigeris (Exception trriger).

Klaidos trigeris gali būti tik tarpinis arba pabaigos. Tarpinis trigeris užfiksuoja išimtį arba klaidą proceso eigoje, O pabaigos išimties trigeris sustabdo procesą .

Kompensavimas trigeris (Compensation trigger).

Kompensavimo trigeris gali būti tik tarpinis arba pabaigos. Šis trigeris naudojamas kai procesą prireikus galima grąžinti atgal.

Atšaukimo trigeris (Cancel trriger).

Atšaukimo trigeris gali būti tarpinis ir pabaigos. Jis naudojamas tik tuomet, kai procesas yra nutraukiamas savo noru.

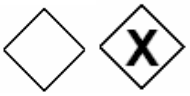
Priverstinės pabaigos trigeris (Terminate trriger).

Priverstinės pabaigos trigeris gali būti tik pabaigos. Jis naudojamas tuomet kai įvyko klaida ir dėl to procesas nutraukiamas priverstiniu būdu.

Sprendimų taškas (Gateway control type)– kontroliuoja išsiskiriančius sekų srautus. Jis apibrėžia išskaidymą bei sujungimą.

Išimtinis(Exclusive) sprendimų taškas.

Duomenimis pagrįstas (Data-based).



Kai reikalinga srautų kontrolė BPMN naudojamas sprendimų taškas. Atėjus srautui, tikrinama, kuri sąlyga yra išpildoma. Ta atšaka, kurioje sąlyga yra išpildoma tęsiamas srautas. Tarp visų alternatyvių srautų gali būti tik vienas kuris užbaigs srautų eigą

Įvykiais pagrįstas(Event-based).

Šis sprendimas ypatingas tuo , kad lygiagretus išsišakojimas grįstas įvykiais atsirandančiais procese. Šis taškas atlieka papildomą sekų kontrolę. Po šio sprendimų taško seka tarpiniai įvykiai. Kai įvykis atvyksta tikrinama, kuris kelias tenkinamas, tuo keliu srautas siunčiamas toliau. Kiti įvykiai nepraleidžiami, jie gali būti naudojami kaip taimeriai.

Apimantis (Inclusive) sprendimų taškas(or).

Šis taškas priėmęs ateinantį srautą išskaido į tiek lygiagrečių kelių , kiek sąlygų yra tenkinama. Nuo išplėtimų taško skiriasi tuo, kad galima pasirinkti ne vieną, o nulį ir daugiau išeinančių alternatyvių srautų. Suteikiama galimybė nesirinkti nė vienos alternatyvos.

Sudėtinis (Complex) sprendimų taškas.

Sudėtinis sprendimų taškas BPMN naudojamas , kad būtų galima apibrėžti kiek ateinančiu srautų gali būti tęsiami. Likę srautai užblokuojami sprendimų taške.

Lygiagretus (Parallel) sprendimų taškas.

Lygiagretus išskaidymas – tai vieno kelio išskaidymas į du ir daugiau kelių einančius lygiagrečiai

Ciklai(Loops) – ciklai gali būti dviejų rūšių :

Veiksmų ciklai (Activity looping).

Veiksmų cikluose užduotys ir procesai bus sustabdyti, jei šie įvykiai įvyks tik vieną kartą.

Sekų ciklai (Sequence flow looping).	Sekų ciklai sudaromi sekos srautą jungiant su prieš tai einančiais objektais. Sekų ciklai naudojami ne tada kai tam tikras įvykis kartojamas keltą kartų, o kai tam tikra įvykių seka kartojama tam tikrą kiekį kartų.

4. Modeliai naudojant UML 2.0 bei BPMN notacijas

Modeliavimui buvo pasirinkta nuotolinio mokymo sistema. Šiame eksperimentiniame modelyje panaudota daugelis elementų, kuriais siūloma papildyti UML 2.0. Kad vaizdžiai parodyti modelio skirtumus tarp UML 2.0, papildytos BPMN stereotipais, ir BPMN, pateiktos sistemos veiklos diagramos naudojant abi notacijas.

Pirmas modelis - nuotolinės mokymosi sistemos modelis nenaudojant juostų ir konteinerių, bet naudojant anotacijas. Šis diagramos tipas paprastesnis ir aiškesnis. Aiškiai matoma kokią veiklą koks darbuotojas kokio proceso eigoje atlieka. Eksperimentinis modelis pateiktas 1 paveiksle.

Antras modelis - nuotolinės mokymosi sistemos modelis realizuotas UML 2.0, papildyta BPMN notacija. Eksperimentinis modelis pateiktas 2 paveiksle.

Eksperimentinio modelio scenarijus:

1ž. Prisijungimas. Vartotojas jungiasi prie sistemos. Užduodamas prisijungimo laikas. Prisijungimo metu tikrinami ar teisingi vartotojo pateikti duomenys. Jei įvesti duomenys teisingi atliekamas 2 žingsnis.

2 ž. Informacijos peržiūra ir redagavimas. Vartotojui prisijungus tikrinama ar sumokėta už mokslą. Jei mokesti už mokslą nesumokėtas tuomet galima atlikti tik 2.1 ir 2.2 žingsnį. Jei mokestis sumokėtas galima atlikti 2.2 , 2.3 ir 2.4 žingsnius.

2.1 ž. Mokėjimo už mokslą procesas. Vartotojas gali sumokėti už mokslą pirmų dviejų semestro savaičių bėgyje.

2.2 ž. Informacijos peržiūrėjimo procesas. Jei už mokslą nesumokėta tuomet užduodamas laikas per kurį vartotojas gali peržiūrėti naujienas ir tvarkaraštį. Užduotam laikui pasibaigus vartotojo paklausiama ar jis nori sumokėti už mokslą. Jei vartotojas pageidauja sumokėti už mokslą t, tuomet atliekamas 2.1 žingsnis, jei ne atliekamas 3 žingsnis.

2.3 ž. Mokslų nutraukimo procesas. Jei vartotojas sumokėjęs reikiamus mokesčius mokymosi įstaigai gali nutraukti mokslus 2.3.1 arba paprašyti akademinių atostogų 2.3.2

2.3.1 ž. Prašymas akademinių atostogų. Vartotojas siunčia pranešimą bei prašymo dokumentą dekanui apie akademines atostogas. Dekanas atsiunčia patvirtinimo dokumentą ir vartotojo mokslas laikinai sustabdomas, atliekamas 3 žingsnis.

2.3.2 ž. Prašymas nutraukti mokslus. Vartotojas siunčia pranešimą bei prašymo dokumentą dekanui apie mokslo nutraukimą. Dekanas atsiunčia patvirtinimo dokumentą ir vartotojo mokslas sustabdomas atliekamas 3 žingsnis.

2.4 ž. Mokymosi ir atsiskaitymų procesas. Mokymosi atsiskaitymo procesas susideda iš semestro darbų atsiskaitymo 2.4.1, Sesijos 2.4.2 ir video konferencijos su dėstytojais 2.4.3. Mokymosi ir atsiskaitymų trukmė - semestro laikas. Atlikus reikiamas operacijas ir panorėjus atsijungti nuo sistemos atliekamas 3 žingsnis.

2.4.1 ž. Semestro darbų atsiskaitymas. Semestro darbų atsiskaitymas susideda iš atsiskaitymų ciklo bei darbų atlikimo ir sudėjimo ciklo. Darbai ir atsiskaitymai negalimi jei neparsiųsta reikiama medžiaga.

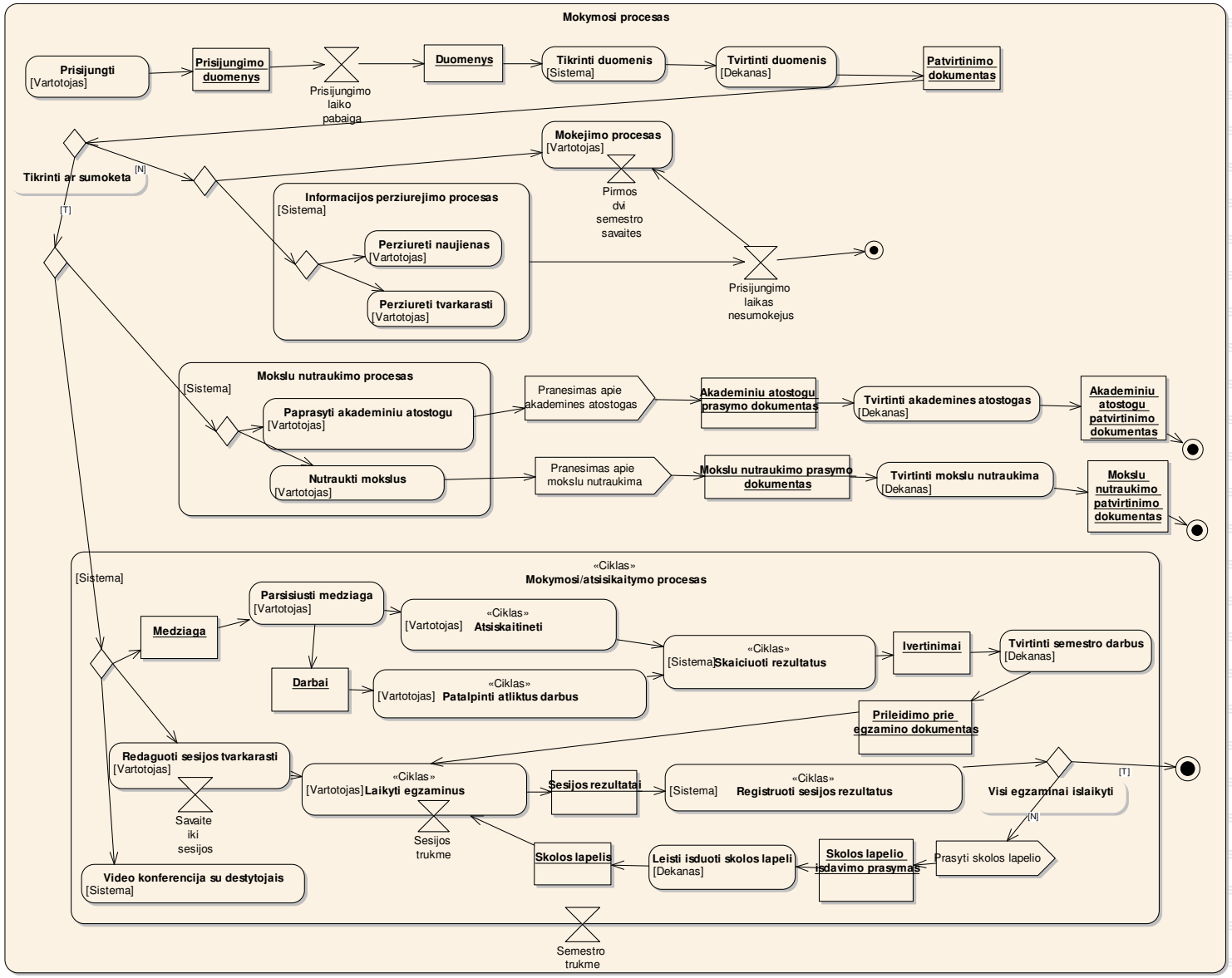
2.4.2 ž. Sesija. Savaitę iki sesijos pradžios galima redaguoti sesijos tvarkaraštį. Sesijos metu vyksta egzaminų ir rezultatų registravimo ciklas. Esant neigiamiems rezultatams galimas kompensavimo ciklas, kurio metu pakartotinai laiko egzaminai ir registruojami rezultatai.

2.4.3 ž. Video konferencija su dėstytojais. Video konferencijos metu galima klausyti paskaitas, bendrauti su dėstytojais ir t.t. abiem pusėms patogiu laiku.

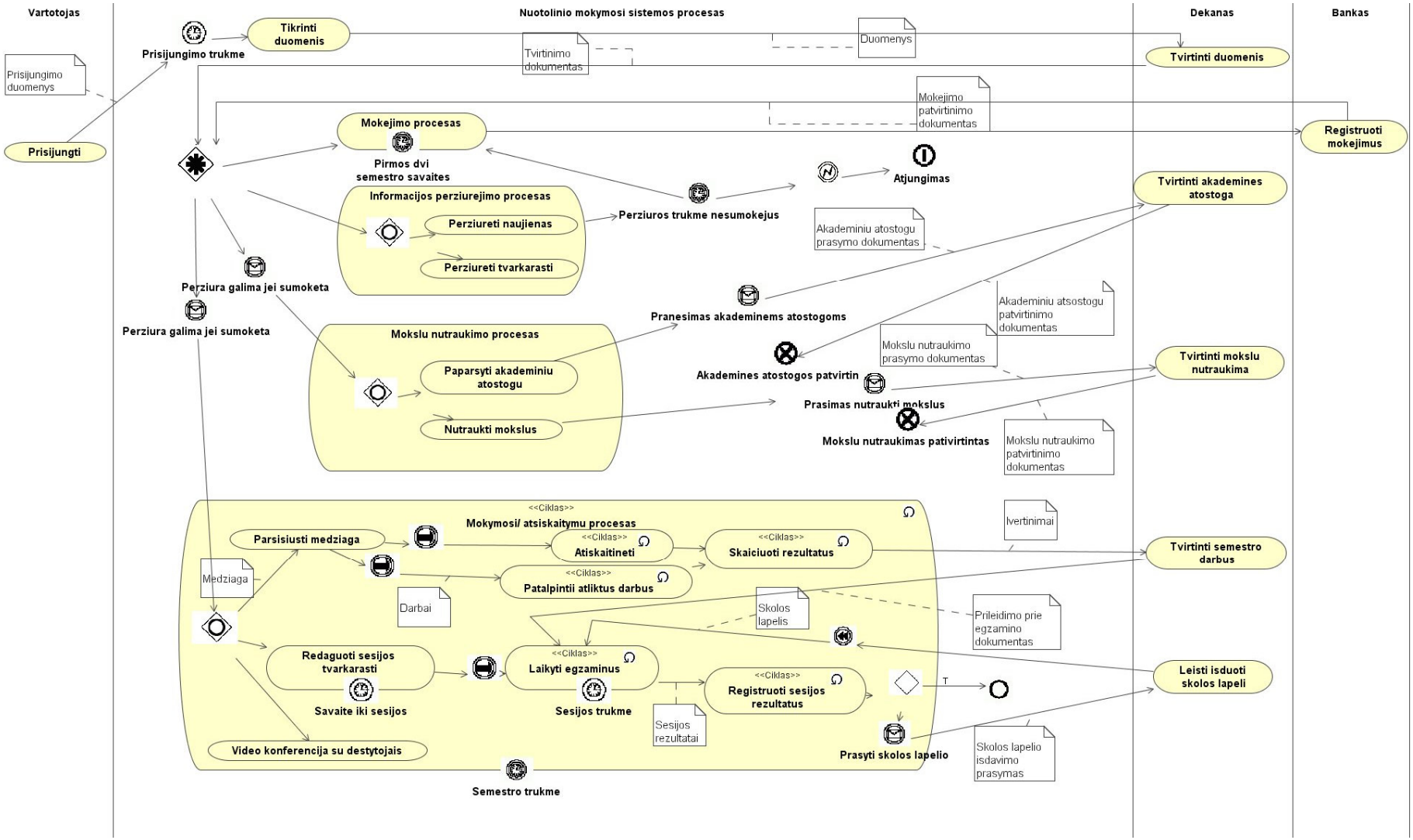
3 ž. Atsijungimas. Vartotojai atsijungia savo noru arba atjungiami priverstinai.

BPMN yra specializuota verslo procesų vaizdavimui ir turi išsamią tam skirtą notaciją UML 2.0 yra universali kalba, ir verslo procesų modeliavimas yra tik viena jos naudojimo krypčių. Verslui modeliuoti UML 2.0 turi panašias galimybes, kaip ir BPMN, tačiau UML 2.0 elementų aibė mažesnė.

Kadangi BPMN notacijoje didesnė elementų įvairovė, aiškiau suprantamas modelis. Kaip matome iš pateiktų diagramų vien sprendimo taškų, bei trigerių gausa padeda greičiau įsisavinti visą veiklos procesą. Tuo tarpu UML 2.0 visi sprendimų taškai gali būti realizuoti, bet modelis tuomet tampa sudėtingesnis ir sunkiau suprantamas sprendimo taškų specifiškumas. Pagrindinis BPMN trūkumas – nėra priemonių dalykinei sričiai vaizduoti. Dokumentai gali būti parodyti tik kaip pastabos. UML veiklos diagramose dokumentai vaizduojami kaip objektai, kurie dalyvauja veiklos procese ir turi įtakos jo eigai. Greta veiklos diagramų UML 2.0 yra išsamus priemonių rinkinys informacinei sistemai projektuoti. Abiejose kalbose galima aprašyti verslo taisykles, tačiau jų apdorojimo priemonių ar rekomendacijų nėra



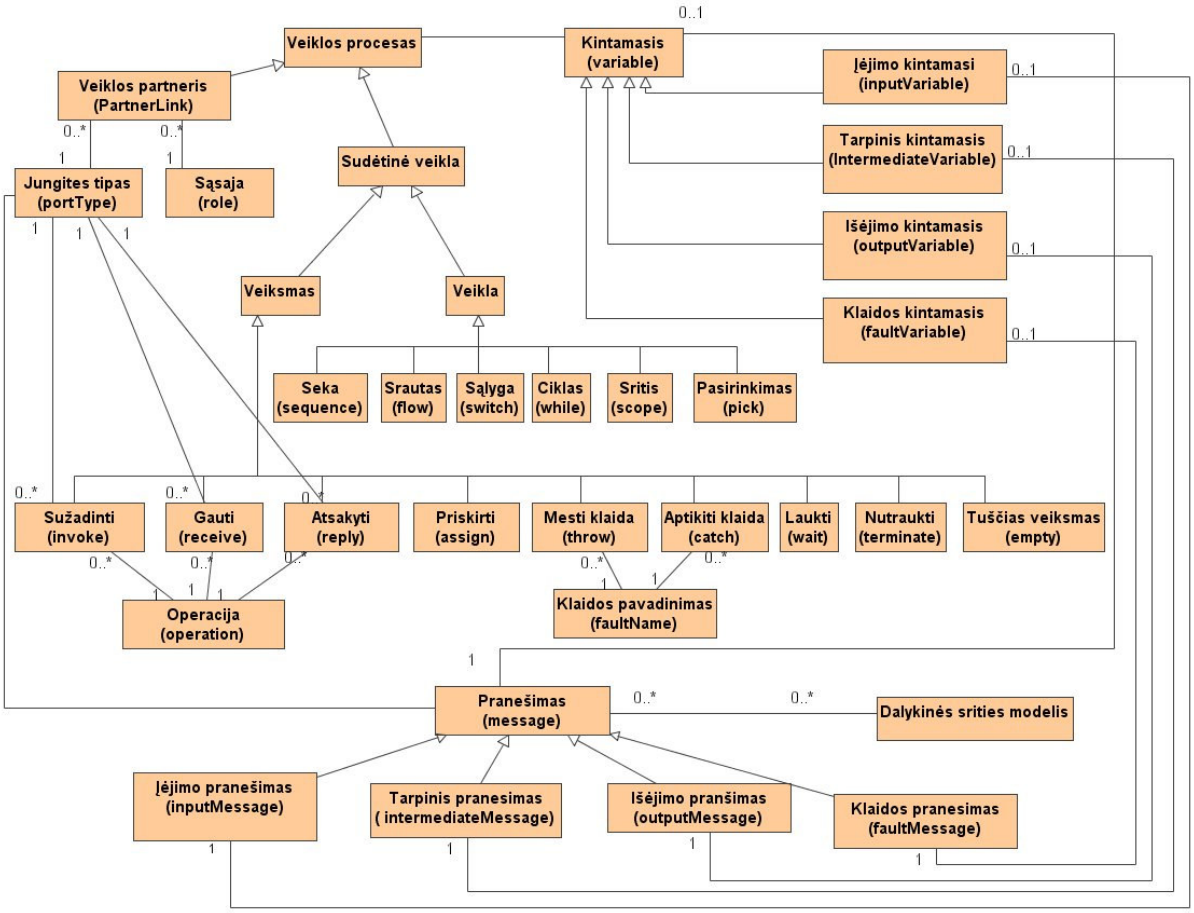
1 pav. Nuotolines mokymosi sistemos veiklos modelis UML 2.0.



2 pav. Nuotolinės mokymosi sistemos modelis BPMN.

5. BPEL metamodelis

Šiame skyriuje **3 pav.** pateikiamas struktūrinių veiklos procesų metamodelis, sudarytas apjungiant BPMN, UML 2.0 ir BPEL4WS kalbų savybes.



3 pav. BPEL metamodelis.

3 pav. pateiktas metamodelis buvo sudaromas analizuojant ir papildant anksčiau sudarytus modelius[8][9]. Metamodelyje vaizduojamas veiklos procesas, kuris susideda iš sudėtinės veiklos, veiklos partnerio bei turi sąsają su kintamųjų aibe. Veiklos procese veikla yra sudėtinė susidedanti iš veiksmo ir įvairių veiklų tipų. Veikla gali būti įvairi: vaizduojama kaip veiklų seka, srautas, sąlyga, sritis, pasirinkimas. Veiksmai priklausomai nuo veiklos proceso elementų gali būti įvairios paskirties: sužadinti, atsakyti, nutraukti ir t.t. Kad atvaizduoti kiekvieną veiksmą reikalinga aprašyti tam veiksmui reikalingus tipus ir elementus. Projektavimo etape reikalingi kintamieji susideda iš įvairių tipų pranešimų. Koks pranešimo tipas toks ir kintamojo tipas. Pranešimams duomenys imami iš duomenų modelio. Kintamieji per pranešimus susieja projektuojamą modelį su dalykine sritimi.

6. Atvaizdavimas Į BPEL4WS

Kadangi siekiama išplėsti grafinę proceso apibrėžimo sąsają, kurioje proceso taisyklės būtų susiejamos su dalykinės srities objektais reikėjo pasirinkti realizacijos kalbą į kuria butų patogu sudarytus modelius transformuoti. Buvo pasirinkta BPEL4WS[10][11] realizacijos kalba. Šioje notacijoje paliekama didelė atvaizdavimo laisvė. Atvaizduojant suprojektuotus modelius į BPEL4WS reikalingi realizavimui duomenys imami iš duomenų modelio sudaryto UML 2.0(dalykinės srities modelis straipsnyje nepateikiamas).

Šiame straipsnyje pateikiamas nedidelis vykdymo į BPEL4WS pavyzdys, kaip pranešimo siuntimas atvaizduojamas į minėtą vykdomąją kalba. Pateikiamas pranešimo atvaizdavimo šablonas ir vieno pranešimo(pranešimas apie akademines atostogas) paimto **2 pav.** (nuotolinės mokymosi sistemos modelis BPMN) pavyzdys.

6.1 Pranešimo vaizdavimo šablonas

Pranešimo (taisyklių, ryšio) trigeris atvaizduojamas į gavimo (receive) elementą Siunčiamas pranešimas sužadina(invoke) kažkokį tai įvykį. Jeigu siunčiamas pranešimas yra vienpusis, tai sužadintas įvykis atvaizduojamas tik į sužadinimo(invoke) elementą. Jei dvipusis, tuomet siunčiamas ataskas(reply) sužadinusiam įvykiui.

Kad atvaizduotume į ankščiau paminėtus elementus, reikia apsirašyti šiuose elementuose naudojamus tipus bei reikalingus elementus. Elementai aprašomi tokia seka:
Apsirašomi visi reikalingi pranešimų tipai. Duomenys pranešimams imami iš UML duomenų modelio:

```
<message name="Pranešimo pavadinimas">
    <part name="Pranešimo duomenys" type="xsd:Pranešimo tipas"/>
    ...
    <part name=…/>
</message>
```

Aprašoma jungtis(port), per kurią siunčiamas pranešimas. Išėjimo klaidos pranešimų gali ir nebūti, jei pranešimas tik sužadina įvykį:

```
<portType name="Siunčiamo pranešimo jungties pavadinimas">
    <operation name="Operacijos pavadinimas">
    <input message="pos:Įėjimo pranešimo pavadinimas"/>
    <output message="pos:Išėjimo pranešimo pavadinimas"/>
    <fault name="Klaidos pranešimas"/>
              message="pos:Klaidos pranešimo pavadinimas"
    </operation>
</portType>
```

Aprašomas partnerių sąsajos tipas:

```
<plnk:partnerLinkType name="Sąsajos tipo pavadinimas">
   <plnk:role name="Sąsajos pavadinimas">
       <plnk:portType name" Siunčiamo pranešimo jungties pavadinimas">
   </plnk:role>
</plnk:partnerLinkType>
```

Aprašomos sąsajos:

```
<partnerLinks>
 <partnerLink name="Sąsajos pavadinimas">
</partnerLinks>
   partnerLinkType =" Sąsajos tipo pavadinimas"
   myRole="Rolės pavadinimas"
</partnerLinks>
```

Aprašomi siuntimo kintamieji (kiekvienam pranešimui sukuriamas kintamasis):

```
<variables>
    <variable name="Kintamojo vardas"  messageType="lns:Pranešimo pavadinimas">
</variables>
```

Visi aukščiau paminėti elementai rašomi tiek kartų kiek yra pranešimų, jungčių ir t.t. Jie užrašomi vienas paskui kita. Aprašomas pranešimo siuntimo klaidos elementas:

```
<faultHandlers>
    <catch faultName=" Klaidos pranešimo pavadinimas"
        faultVariable="Klaidos kintamojo pavadinimas">
    <reply partnerLink="Sąsajos pavadinimas"
        portType=" Siunčiamo pranešimo jungties pavadinimas"
        operation= "Operacijos pavadinimas"
        variable=" Kintamojo vardas"
        faultName="Klaidos pranešimas"
   </catch>
</faultHandlers>
```

Aprašius visus reikiamus tipus ir elementus užrašomas atvaizdavimas į receive elementą, kuris sužadina kitą procesą(invoke) , ir jei reikia(jei ne siuntimas ne vienpusis) gauna atsaką(reply):

```
<receive partnerLink="Sąsajos pavadinimas"
             portType=" Siunčiamo pranešimo jungties pavadinimas"
             operation=" Operacijos pavadinimas"
             variable="Kintamojo pavadinimas"
</receive>
<links>
     <link name="Partnerių sąsajos pavadinimas"/>
</links>
<invoke partnerLink=" Sąsajos pavadinimas"
              portType=" Gavejo jungties pavadinimas"
              operation="Gavėjo operacija"
              inputVariable="Gavėjo įėjimo duomenys"
              outputVariable="Gavėjo išėjimo duomenys"/>
  <source linkName="Partnerių sąsajos pavadinimas"/>
</invoke>
<reply partnerLink="Sąsajos pavadinimas"
          portType="Siunčiamo pranešimo jungties pavadinimas"
          operation="Operacijos pavadinimas"
          variable="Atsako kintamasis"/>
```

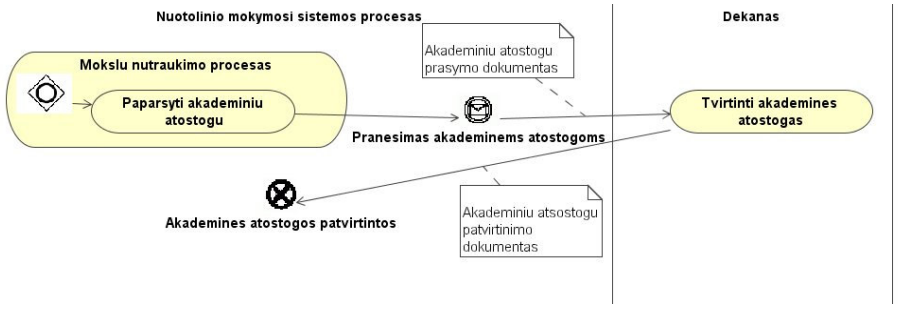
Pranešimas(taisyklė, ryšys) gali būti atvaizduojami į aktyvuotą pranešimą (onMessage), kuris įtraukiamas į įvykį reguliuojantį elementą(eventHandler). Aktyvuotas pranešimas (onMessage) yra atvaizduojamas analogiškai kaip gavimo(recieve) elementas(su tokiais pat atributais).

```
<eventHandlers>
  <onMessage  partnerLink="Sąsajos pavadinimas"
          portType=" Siunčiamo pranešimo jungties pavadinimas"
          operation=" Operacijos pavadinimas"
```

```
          variable="Kintamojo pavadinimas"/>
    veikla
  </onMessage>
</eventHandlers>
```

Veikla gali būti sužadinimas(invoke), klaidos metimas(throw) ir t.t.

7. Pranešimo vaizdavimo pavyzdys



4 pav. Fragmentas iš 2pav. (nuotolinės mokymosi sistemos modelis BPMN) modelio.

Vartotojas siunčia pranešimą apie akademines atostogas dekanui. Iš duomenų modelio vartotojas siuntimui paimami reikalingi duomenys: prašymas(tipo dokumentas), vardas(tipo string), pavardė(tipo string), bei vartotojo studento numeris(tipo string),. Pranešimo siuntimui apsirašomi reikalingi elementai ir tipai tokie kaip: pranešimo tipas, sąsajos tipas, sąsaja, siuntimo partneriai, kintamieji. Jei pranešimas nusiunčiamas be trikdžių ir patvirtinamas(sužadinamas gavėjo procesas), partneris(dakanas) siunčia atsaką(pranešimą apie patvirtinimą) ir patvirtinimo dokumentą. Jei dėl tam tikrų priežasčių dokumentas nepasiekia adresato siunčiamas klaidos pranešimas siuntėjui(vartotojui).

```
<message name="Akademinės atostogas">
    <part name="Vartotojas.Prašymas" type="xsd:dokumentas"/>
    <part name="Vartotojas.vardas" type="xsd:string"/>
    <part name="Vartotojas.pavadė" type="xsd:string"/>
    <part name="Vartotojas.studento_numeris" type="xsd:string"/>
</message>
<message name="Pranešimas nepatvirtintas">
    <part name="Pranešimas.Siuntimo klaida" type="xsd:string"/>
</message>
<message name="Akademinės patvirtintos">
    <part name="Dekanas.Dokumentas" type="xsd:dokumentas"/>
</message>
<portType name="Prašymo siuntimas">
    <operation name="Siųsti prašymą">
    <input message="pos: Akademinės atostogas"/>
    <fault name="Prašymas negalimas"/>
              message="pos: Pranešimas nepatvirtintas"
    </operation>
</portType>
<portType name="Prašymo tvirtinimas">
    <operation name="Tvirtinti">
    <input message="pos: Akademinės atostogas"/>
  </operation>
</portType>
<plnk:partnerLinkType name="Siuntimas">
   <plnk:role name="siuntimo aptarnavimas">
       <plnk:portType name" Prašymo siuntimas">
   </plnk:role>
</plnk:partnerLinkType>
<partnerLinks>
 <partnerLink name="Pranešimo siuntimas">
   myRole="siuintimo aptarnavimas"
< partnerLinkType =" Siuntimas">
  myRole="patvirtinimo aptarnavimas"
</partnerLinks>
<variables>
 <variable name="Pranešimas apie akademines atostogas" messageType="lns:Akademinės atostogos">
<variable name="Klaida" messageType="lns: Pranešimas nepatvirtintas">
<variable name="Akademinės atostogos patvirtintos" messageType="lns: Akademinės patvirtintos">
</variables>
<faultHandlers>
    <catch faultName=" Pranešimas nepatvirtintas"
        faultVariable="Klaida">
    <reply partnerLink=" Pranešimo siuntimas"
        portType=" Prašymo siuntimas"
        operation= "Siųsti prašymą"
        variable=" Klaida"
```

```
        faultName=“ Pranešimas nepatvirtintas“
   </catch>
</faultHandlers
<receive partnerLink=“ Pranešimo siuntimas“
              portType=“ Prašymo siuntimas“
              operation=“ Siųsti prašymą“
              variable=” Pranešimas apie akademines atostogas”
</receive>
<links>
     <link name=”Siųsti-Priimti”/>
</links>
<invoke partnerLink=”Siuntimas”
               portType=” Prašymo tvirtinimas”
               operation=”Tvirtinti”
               inputVariable=” Pranešimas apie akademines atostogas”
               outputVariable=” Akademinės atostogos patvirtintos”/>
  <source linkName=”Siųsti_Priimti”/>
</invoke>
```

8. Išvados

Šiame darbe buvo analizuojamos dvi naujų projektavimo kalbų UML 2.0 ir BPMN savybės. BPMN yra specializuota verslo procesų vaizdavimui ir turi išsamią tam skirtą notaciją UML 2.0 yra universali kalba, ir verslo procesų modeliavimas yra tik viena jos naudojimo krypčių. Verslui modeliuoti UML 2.0 turi panašias galimybes, kaip ir BPMN, tačiau UML 2.0 elementų aibė mažesnė. Pagrindinis BPMN trūkumas – nėra priemonių dalykinei sričiai vaizduoti. Dokumentai gali būti parodyti tik kaip pastabos. UML veiklos diagramose dokumentai vaizduojami kaip objektai, kurie dalyvauja veiklos procese ir turi įtakos jo eigai. Greta veiklos diagramų yra išsamus priemonių rinkinys informacinei sistemai projektuoti. Abiejose kalbose galima aprašyti verslo taisykles, tačiau jų apdorojimo priemonių ar rekomendacijų nėra. Kadangi verslo procesams projektuoti nepakanka vienos kurios notacijos buvo siūloma UML 2.0 papildyti BPMN stereotipais. Kad projektavimo procesai būtų pilnai išbaigtas grafinis modelis turi būti siejamas su dalykinės srities modeliu, todėl buvo pasirinkta viena iš realizavimo kalbų BPEL4WS. Išanalizuotas atvaizdavimas į šia kalbą ir sudarytas metamodelis jungiantis visas anksčiau paminėtas kalbas.

Literatūra

1. „Buisiness porcess modeling notation „ Prieiga per internetą http://dwdemos.dfw.ibm.com/wstk/common/wstkdoc/ettk/wstk/services/demos/uml2bpel/docs/UMLProfileForBusinessProcesses1.1.pdf
2. „Unified Modeling Language: Superstructure version 2.0“ Prieiga per internetą: http://www.u2-partners.org/uml2-submissions.htm
3. Prieiga per internetą: http://bpmi-notation-wg.netfirms.com/Documents/NWG-2002-11-02R1%20BPMN%200-9-1.pdf
4. „Buisiness porcess modeling notation version 1.0 may 3, 2004„ Prieiga per internetą http://www.google.com/search?q=BPMN+V1-0+May+3+2004(1).pdf&sourceid=opera&num=0&ie=utf-8&oe=utf-82
5. „Business process execusion language for Web services version 1.1 “ Prieiga per internetą http://www-128.ibm.com/developerworks/webservices/library/ws-bpel1/Business Process Execution Language for Web Services Version 1_1.htm
6. „Users manual 7,2“ Prieiga per internetą: www.magicdraw.com
7. „Users guide“ Prieiga per internetą: http://www.sparxsystems.com.au/bin/EAUserGuide.pdf
8. Simanaitytė K., Nemuraitė L. Veiklos procesų modeliavimas naudojant veiklos procesų modeliavimo notaciją: 9-oji tarpuniversitetinė magistrantų ir doktorantų konferencija [2004m. balandžio 15 d., Kaunas]: pranešimų medžiaga. Kaunas, 2004, p. 254-259.
9. „WS-BPEL“ Prieiga per internetą: http://www.ebpml.org/bpel4ws.htm
10. „Business process execution language for web services“Prieiga per internetą http://www.oasis-pen.org
11. „XML Schema Part 2: Datatypes Second Editon“ Prieiga per internetą: http://www.w3.org/TR/xmlschema-2

Analysis of Business Process Modelling Languages and Specification Means

Visual languages are used for understability of business analysis, modelling and computerisation processes for business analysts as soon as system developers. The advantages of visual modeling are standard notation, unified concepts, intuitivity of use. There is no single language suitable for all phases of business process evolution. Three languages were analysed: UML 2.0, BPMN bei BPEL4WS, with regards of their possibilities to represent and execute e-business processes. For making comprehensive e-business model, it is proposed to extend UML 2.0 with BPMN stereotypes and to implement them in UML CASE tool. The exclusive feature of proposed way of modelling lies in integration of of business process model with object types of problem domain

IŠPLĖSTOJI UML NOTACIJA ELEKTRONINIO VERSLO PROCESAMS MODELIUOTI IR SPECIFIKUOTI[2]

Inga Pašilskytė, Lina Nemuraitė

KTU, Informacijos sistemų katedra,
Studentų 50 - 308, 51368 Kaunas

Straipsnyje siūloma išplėstoji UML notacija, skirta verslo analitikui, projektuojančiam elektroninio verslo procesus. Ji sukurta į UML 2.0 įvedant verslo modeliavimo kalbos BPMN stereotipus verslo procesams vaizduoti ir papildant jų specifikacijas šablonais verslo taisyklėms aprašyti; verslo taisyklių specifikacijų elementai susiejami su duomenų modelio elementais. Tokio vaizdavimo privalumai – galimybė integruoti proceso aprašą su jame naudojamais duomenų ir tinklo paslaugų modeliais. Sudarytą e.verslo proceso specifikaciją galima tiesiogiai transformuoti į vykdymo kalbą BPEL. Pateikiamas realizacijos prototipas UML CASE įrankyje Magic Draw.

Įvadas

Šio straipsnio tikslas – pasiūlyti paprastą ir efektyvų sprendimą elektroninio verslo procesams modeliuoti. Elektroninio verslui modeliui keliami dideli reikalavimai. Tam turėtų būti naudojama verslo analitikui priimtina vaizdinė kalba, leidžianti aprašyti globalius verslo procesus ir ilgas transakcijas; sąveikas tarp verslo dalyvių; vidinius organizacijų procesus; pranešimus ir verslo dokumentus, susiejant juos su verslo proceso žingsniais; kartu su ja turėtų būti ir išraiškų kalba, leidžianti aprašyti verslo taisykles ir apribojimus. Ši kalba turėtų būti realizuota įrankyje, kuris užtikrintų verslo procesų modelių kūrimą, išsaugojimą ir vykdymą ar perdavimą taip vadinamiems verslo procesų valdymo įrankiams. Yra daug verslo modeliavimo kalbų, kurios gerai įgyvendina kai kuriuos iš šių reikalavimų, tačiau nėra tokios kalbos, kuri tenkintų visus reikalavimus ir užtikrintų e.verslo proceso sukūrimą nuo reikalavimų lygmens modelio iki programinės realizacijos.

Šiuo metu sukurta nemažai XML grindžiamų e.verslo kalbų, iš kurių didžiausio pripažinimo ir praktinių taikymų sulaukė BPEL (Business Process Execution Language) (BPEL4WS, 2004), palaikanti XPath 1.0 funkcijas. Ji skirta formaliai specifikuoti veiklos procesus ir jų sąveikos protokolus. BPEL apibūdinama kaip „orkestruotės" (angl. Orchestration) kalba, nes ji leidžia aprašyti vienos organizacijos e.verslo procesus, tačiau suderinant kelių organizacijų procesų aprašus, galima apibrėžti ir taip vadinamą „choreografiją" (angl. Choreography), kuri sinchronizuoja skirtingų dalyvių sąveikų sekas. BPEL aprašytą verslo procesą galima vykdyti BPEL varikliais, kurių realizacijas yra sukūrę IBM, Oracle ir kt. BPEL netenkina dalies reikalavimų, keliamų e. verslo procesams vaizduoti, tačiau tuos reikalavimus būtų galima užtikrinti naudojant verslo analitikui suprantamą aukštesnio lygmens vaizdinę kalbą, kuri leistų atvaizduoti globalius verslo procesus, jų dalyvių choreografijas, ilgas transakcijas ir pan., po to automatiškai generuojant BPEL kodą atskiriems modelyje aprašytiems procesams.

Verslo analitikų poreikiams neseniai buvo sukurta verslo procesų modeliavimo notacija BPMN (Business Process Modelling Notation) (White, 2004) – palyginti naujas modeliavimo standartas, kuris turi didelę grafinių simbolių aibę verslo procesų žingsniams, srautams, trigeriams, klaidoms, kompensacijoms ir pan. vaizduoti. BPMN leidžia aprašyti bendradarbiaujančių organizacijų verslo procesus, kuriuos galima atvaizduoti į programinę realizaciją. Ši kalba turi vienintelio tipo diagramą, kurią galima naudoti įvairiais būdais: vidiniams verslo procesams – orkestruotei; abstraktiems viešiems procesams – choreografijai; globaliems bendradarbiavimo procesams ir įvairioms jų kombinacijoms. Tačiau BPMN turi eilę trūkumų: visų pirma, tai dar vienas standartas, kurį reikia integruoti su projektavimo kalbomis ir įrankiais; antra, BPMN modeliuose verslo procesai nesusiejami su duomenų modeliais, kas yra būtina, norint generuoti verslo vykdymo programos kodą.

Todėl šiame darbe siūloma e.verslo procesams modeliuoti pritaikyti universalią modeliavimo kalbą UML (angl. Unified Modeling Language) (OMG, 2003), papildant ją BPMN stereotipais. UML yra plačiai paplitęs programinės įrangos projektavimo, specifikavimo, realizavimo ir dokumentavimo standartas, todėl verslo procesų modelius galima integruoti su duomenų, paslaugų ir kitais modeliais, kurių reikia išsamiai verslo proceso realizacijai.

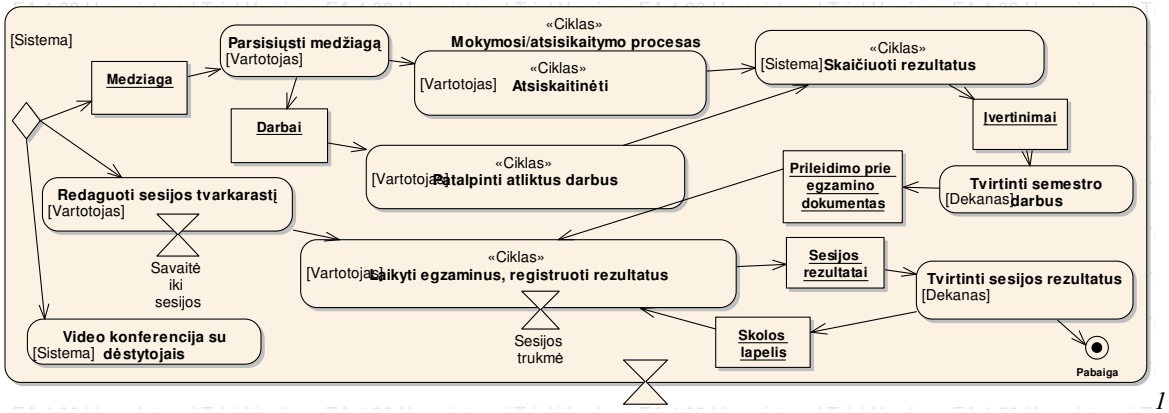
Elektroninio verslo modeliavimo principai ir įrankiai

BPMN yra specializuota verslo procesams vaizduoti ir turi išsamią tam skirtą notaciją. UML 2.0 yra universali kalba, ir verslo procesų modeliavimas yra tik viena jos naudojimo krypčių. Jų galimybės šiuo požiūriu panašios, tačiau UML 2.0 elementų aibė verslo procesams vaizduoti mažesnė. Pagrindinis BPMN trūkumas – nėra priemonių dalykinei sričiai vaizduoti. Duomenys, dokumentai gali būti parodyti tik kaip pastabos. UML veiklos diagramose dokumentai vaizduojami kaip objektai, kurie dalyvauja veiklos procese ir turi įtakos jo eigai; greta veiklos diagramų yra išsamus priemonių rinkinys informaciniams modeliams ir paslaugoms, kurių pagrindu veikia procesas, projektuoti. Abiejose kalbose galima aprašyti verslo taisykles, tačiau nusistovėjusių, plačiai naudojamų priemonių, rekomendacijų joms tikrinti ar transformuoti į kodą trūksta.

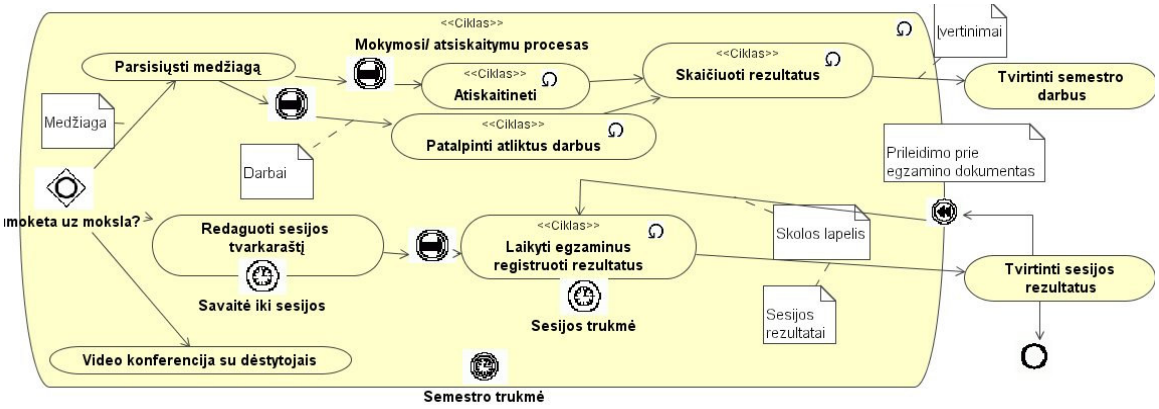
Norint pritaikyti UML 2.0 notaciją e.verslui modeliuoti, ją reikia papildyti BPMN veiklų tipais ir verslo taisyklių aprašais. 1 ir 2 paveiksluose pateiktas veiklos modelio pavyzdys UML 2.0 ir BPMN notacijose. Nauja UML 2.0

[2] Šį darbą remia Lietuvos valstybinis mokslo ir studijų fondas pagal Eureka programos projektą „IT-Europe" (Reg. Nr. 3473)

versija suteikia galimybę specifikuoti vykdomuosius modelius, todėl ji gali tarnauti kaip nuo platformos nepriklausoma vaizdinio programavimo kalba. Joje yra daug papildymų, naudingų projektuoti verslo modeliams: nauji sudėtinių veiklų ir jungimų tipai, duomenų saugyklos ir kt. Veiklos diagramos yra trijų lygių: pirmame lygyje abstrakčiai vaizduojama veikla; antro lygio diagramos gali būti dvejopos: tarpinės veiklos (angl. Intermediate Activities) diagramos aprašo veiklos grafus, valdymo ir duomenų srautus; struktūrinės veiklos (angl. Structured Activities) – programavimo kalboms ekvivalentiškus darinius. Trečiame lygyje aprašomi detalūs veiksmai. UML 2.0 verslo dalyvius žymi anotacijos (pvz., [Dekanas]), objektų simboliai sieja veiklų įėjimus/išėjimus su informacinio modelio elementais (pvz., skolos lapeliais). BPMN atveju informacinius objektus galima aprašyti pastabose, tačiau jos toliau neapdorojamos. Pagrindinis BPMN trūkumas tame, kad nėra sąsajos su duomenų bazių ir elektroninių paslaugų projektavimo priemonėmis. Šį trūkumą lengva įveikti, naudojant UML 2.0, kuri turi abstrakčius atitikmenis visiems BPMN elementams (Pašilskytė, 2005), tačiau jai savo ruožtu trūksta verslo modeliavimo specifikos ir geros metodikos, taip pat paramos verslo taisyklėms specifikuoti. UML turi OCL kalbą išraiškoms specifikuoti, tačiau verslo procesų taisykles patogiau specifikuoti OCL labai artimomis, tačiau šiek tiek skirtingomis XPath išraiškomis.



1 pav. Nuotolinės mokymo sistemos veiklos proceso dalis UML 2.0 notacija

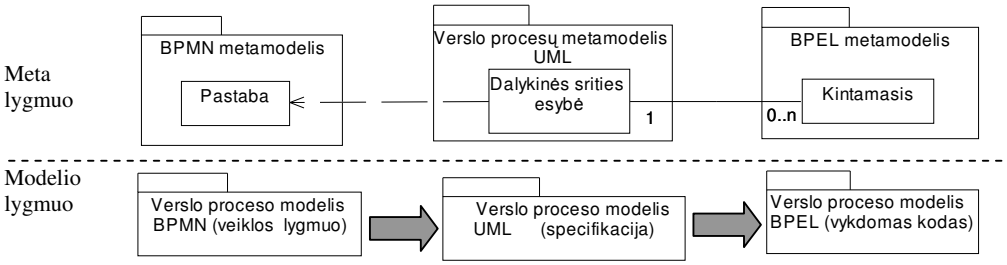


2 pav. Nuotolinės mokymo sistemos veiklos proceso dalis BPMN notacija

BPMN kalba realizuota plačios paskirties įmonių modeliavimo įrankiuose, pavyzdžiui, Popkin Architect, tačiau išsamios sąsajos tarp įvairių notacijų ir kūrimo etapų dar nėra pasiektos. UML 2.0 standartą stengiamasi įdiegti daugelyje įrankių, iš kurių galima paminėti Enterprise Architect Corporate v4.0 (EA Users Manual, 2004), Magic Draw (MagicDraw, 2005), IBM Rational Software Architect ir Modeller (Rational Software Architect, 2005). UML 2.0 realizacijos yra labai naujos, natūralu, kad jose nemaža klaidų ir ne visos savybės įgyvendintos. Šiame darbe pasirinktas Magic Draw, kadangi jis gerai palaiko tradicinius projektavimo procesus; turi duomenų bazių, XML, WSDL schemų generavimo galimybes; OCL specifikavimo ir sintaksės tikrinimo priemones; plėtimo API; leidžia lengvai įvesti naujus grafinius stereotipus. UML 2.0 notacijos realizacija Magic Draw dar turi trūkumų, tačiau tikimasi, kad jie netrukus bus pašalinti.

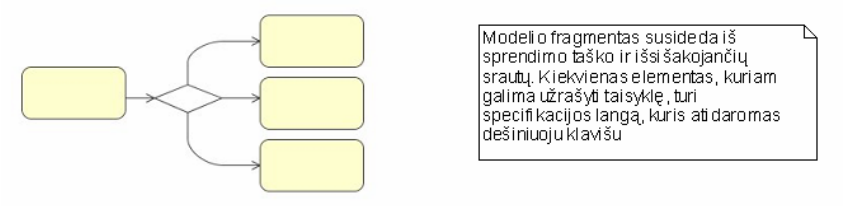
Siekiamos universalaus modeliavimo įrankio savybės

Kaip buvo minėta, šiame darbe siūloma UML 2.0 kalbą papildyti BPMN elementais, išplečiant grafinę UML CASE įrankio sąsają, ir parengti proceso apibrėžimą realizuoti BPEL (Biron, 2004; BPEL4WS, 2003). Atvaizduojant verslo procesų modelius į BPEL, reikalingi duomenų tipai turėtų būti imami iš duomenų modelio – visos galimybės tam yra UML notacijoje ir UML CASE įrankiuose. Taip galima susieti verslo procesų taisykles su dalykinės srities modelio elementais. Principiniai BPMN, UML ir BPEL metamodelių elementų ryšiai parodyti 3 paveiksle (verslo proceso metamodelis detaliau nagrinėjamas (Pašilskytė, 2005)).



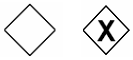
3 pav. UML 2.0, BPMN ir BPEL kalbų metamodelių sąsajos

Toliau pateikiamas pasiūlymas grafinei sąsajai, kuri leistų modeliuoti verslo procesus BPMN stereotipais išplėsta UML 2.0 notacija. Tokios sąsajos prototipas įgyvendintas įvedant stereotipus į MagicDraw 9.0 (visi straipsnyje pateikiami pavyzdžiai sukurti stereotipais papildytu MagicDraw įrankiu). Tobulinant sąsają, reikia sukurti galimybes specifikuoti verslo procesų taisykles – toks aprašas būtų pilnai parengtas automatiškai transformuoti į BPEL (Amsden J., 2002). Toliau pateikiamas verslo taisyklių specifikacijos vaizdas. Verslo taisyklės gali būti rašomos sprendimo taškams, srautams ir veiksmams. Verslo procesas su sprendimo tašku parodytas 4 paveiksle.

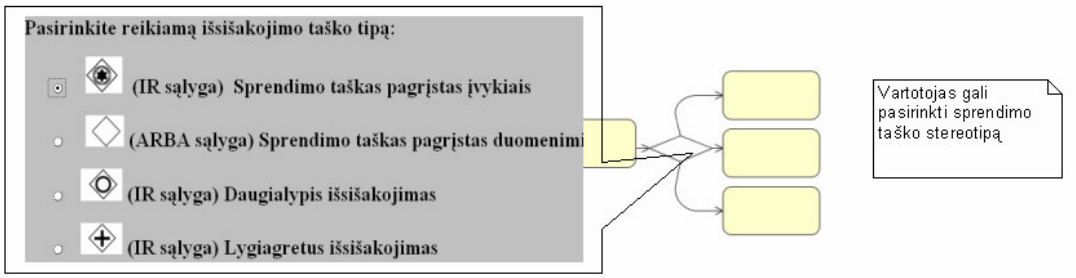


4 pav. Verslo procesas su sprendimo tašku

Pirmiausia tikslinga aprašyti sprendimo tašką, nes nuo sprendimo taško tipo priklauso išeinančių srautų valdymas. Galimi tokie sprendimo taškų variantai:

Duomenimis grindžiamas sprendimo taškas (angl. Data-based)		Duomenimis grindžiamame sprendimo taške atėjus srautui, tikrinama, kuri sąlyga galioja pagal srauto duomenis, ir ta šaka tęsiamas srautas. Iš visų alternatyvių srautų gali būti išrinktas tik vienas
Įvykiais grindžiamas sprendimo taškas (angl. Event-based).		Įvykiais grindžiamame sprendimo taške išėjimo srautą nulemia procese atsirandantys įvykiai.
Apimantis (angl. Inclusive) sprendimų taškas (or)		Apimančiame sprendimų taške srautas išsišakoja į tiek kelių, kiek sąlygų tenkinama (galima nesirinkti nė vienos alternatyvos)
Lygiagretus (angl. Parallel) sprendimų taškas		Lygiagretaus išskaidymo taškas naudojamas, kai vieną srautą reikia išskaidyti į du ir daugiau išeinančių lygiagrečiai vykdomų srautų

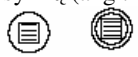
Atidaręs spendimo taško specifikacijos langą, vartotojas išsirenka sprendimo taško tipą:



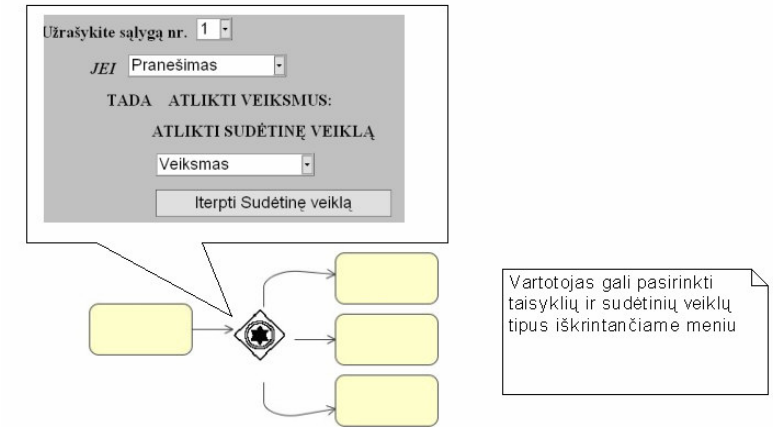
5 pav. Sprendimo taško specifikavimo forma

Pasirenkant sprendimo taško tipą automatiškai parenkamas ir BPEL kodo šablonas, į kurį perkeliama reikiama informacija iš vartotojo sąsajos lango. Tolimesnis žingsnis – srautų taisyklių užrašymas. Srautų taisyklės užrašomos trigeriais, kaip siūloma BPMN specifikacijoje:

Pranešimų trigeriai (angl. Message triggers)	Gali būti pradžios, tarpiniai ir pabaigos pranešimų trigeriai. Pradžios trigeris pradeda procesą, tarpinis – tęsia procesą, pabaigos – praneša apie proceso pabaigą. Pabaigos trigeris veikia, jei buvo tarpinis trigeris.
Laiko (angl. Timer) trigeriai	Laiko trigeriai gali būti tik pradžios ir tarpiniai. Jie užduoda tam tikrą laiko tarpą, kurio neviršijus (arba atėjus tam tikram momentui)

	procesas tęsiamas.
Taisyklių (angl. Rule) trigeriai 	Taisyklių trigeriai gali būti tik pradžios arba tarpiniai. Procesas tęsiamas, jei taisyklė tenkinama.
Ryšio (angl. Link) trigeriai 	Ryšio trigeriai gali būti pradžios, tarpiniai ir pabaigos. Ryšio trigeriai naudojami, jei procesas gali prasidėti tik pasibaigus kitam procesui ar pasiekus tarpinę būseną ir pan.
Klaidų (angl. Exception) trigeriai 	Klaidų trigeriai gali būti tik tarpiniai arba pabaigos. Tarpiniai trigeriai užfiksuoja išimtį arba klaidą proceso eigoje, pabaigos trigeriai sustabdo procesą.
Kompensavimo (angl. Compensation) trigeriai 	Kompensavimo trigeriai gali būti tik tarpiniai arba pabaigos. Jie naudojami tam, kad prireikus procesą būtų galima grąžinti atgal.
Atšaukimo (angl. Cancel) trigeriai 	Atšaukimo trigeriai gali būti tarpiniai ir pabaigos. Jie naudojami tik tuomet, kai procesas nutraukiamas savo noru.
Priverstinės pabaigos (angl. Terminate) trigeriai 	Priverstinės pabaigos trigeriai gali būti tik pabaigos. Jie naudojami tuomet, kai įvyko klaida ir procesas nutraukiamas priverstiniu būdu.

Taisyklės užrašomos specifikacijos lange. Specifikavimo formos šablonas priklauso nuo prieš tai pasirinkto sprendimo taško tipo. Pateiktoje formoje vartotojas gali ne tik aprašyti srauto taisyklę, bet ir po jo einančią sudėtinę veiklą.



6 pav. Įvykiais pagrįsto sprendimo taško specifikavimo forma

UML ir BPMN kalbose galima aprašyti verslo taisykles, tačiau jų apdorojimo priemonių ar rekomendacijų nėra. Formos taisyklių aprašymui sudarytos pagal atvaizdavimo į BPEL šablonus. Aprašant taisykles, naudojami duomenų modelio atributai ir ryšiai – verslo procesų modeliai susiejami su dalykine sritimi.

Paimkime taisyklę, aprašančią laiko elementą. Laiko trigeris gali būti trijų tipų: laukimo (angl. wait); laiko/datos (angl. Time/Date) – *wait* su atributu *until;* laiko ciklas (angl. TimeCycle) – *wait* su atributu *for.* Tarkim, kad taisyklė tokia: „sumokėti už mokslą galima per dvi pirmas semestro savaites“. Laiko apribojimo taisyklės šablonas atrodys taip :

```
<wait (for="ciklo trukme"| until="galutinė data") standartiniai atributai>
       Standartiniai elementai </wait>
```

Užpildytas BPEL kodo šablonas atrodys taip :

```
<wait until=" '2005-02-04T8:00' ">
```

Šablonas natūralia kalba:

```
Laukti kol a = b
atlikti c1
atlikti c2
...
atlikti cn,
čia a – momento data, b – kintamasis, d – konkreti data;
c1,c2,c3...cn – veiklų seka (sužadinimas, išmetimas, atsakas ir t.t.);
c11,c12,c13...c1n;c21,c22,c23...c2n;...;cn1,cn2,cn3...cnn –  tipai, reikalingi atitinkamų elementų
aprašymui.
```

Įrankyje pateikta užpildymo forma turėtų atrodyti taip :

Pateiktame iškrintančiame meniu vartotojas pasirenka reikiamą duomenų modelio lentelę ir reikiamą įrašą. Šio momento datą bei duomenų lentelėje pateiktą datą automatiškai palaiko XPath 1.0 funkcijos. Pateiktas laiko užrašymo pavyzdys yra vienas iš paprasčiausių. Pranešimas užrašomos truputį sudėtingiau, tam reikia kelių formų. Taisyklės patalpinamos į tam skirtą bazę.

Analogiškai užpildoma ir informacija apie sudėtinę veiklą. Sudėtine veikla gali būti veiksmas, ciklas, tuščias veiksmas, išimtis, procesas, grupė, sudėtinis veiksmas. Pagal pasirinktą tipą pateikiama nauja forma, kurioje užpildomi reikiami laukai. Tarkim, sudėtinė veikla yra ciklas. BPEL kodo šablonas atrodys taip:

```
<while conditon="loginė ciklo sąlyga" standartiniai atributai>
   Standartiniai elementai
   Veikla
  <assign>
   <copy>
    <from expression=" sąlygos kintamojo pakeitimo išraiška"
    <to variable="kintamojo į kurį įrašoma išraiškos reikšmė pavadinimas" />
  </copy>
 </assign>
</while>
```

Vartotojo šablonas natūralia kalba:

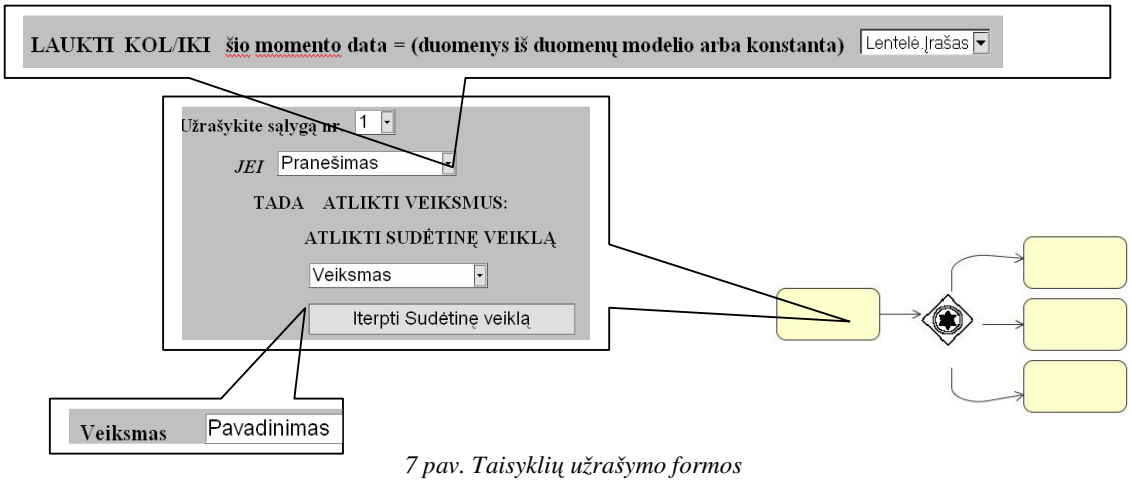
```
Kol a >|<|=|≥|≤ b
   atlikti c1
   atlikti c2
      ...
   atlikti cn
 a=a+1
```

a – ciklo kintamasis,b – pastovus kintamasis (gali būti iš domenų modelio)

$c_1, c_2, c_3 ... c_n$ – veiklų seka (sužadinimas, klaidos metimas atsakas ir t.t.)

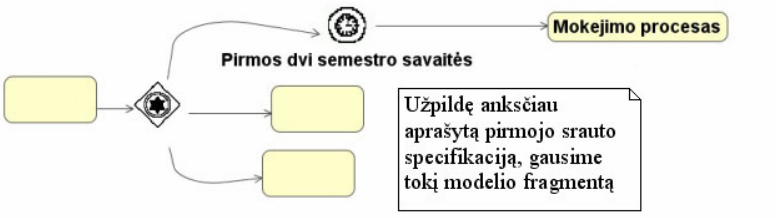
$c_{11}, c_{12}, c_{13} ... c_{1n}; c_{21}, c_{22}, c_{23} ... c_{2n}; ...; c_{n1}, c_{n2}, c_{n3} ... c_{nn}$– tipai, reikalingi atitinkamų elementų aprašymui.

Grafinis vaizdas įrankyje:



7 pav. Taisyklių užrašymo formos

Analogiškai atliekami veiksmai ir su kitais išsišakojimais bei sprendimų taškais. Užpildžius reikiamus laukus, informacija patalpinama į taisyklių lenteles ir BPEL šabloną. Taip formuojamas išsamus verslo modelis ir pildoma taisyklių bazė, iš karto paruošiant BPEL kodą. Suformuotas modelio fragmentas pateiktas 8 paveiksle.



8 pav. Modelio fragmentas suformavus verslo taisyklę

Išvados

Straipsnyje siūloma e.verslo procesus modeliuoti standartine UML kalba, išplėsta specifiniais verslo modelių elementais iš verslo procesų modeliavimo notacijos BPMN. Naudojant tokią kalbą UML CASE įrankyje turi būti sukurti BPMN stereotipai ir papildomos specifikacijos aprašyti verslo proceso taisyklėms, kurios atvaizduojamos į verslo procesų vykdymo kalbos BPEL konstrukcijas.

Standartinis notacijos pagrindas leistų lengviau susieti verslo ir vykdomosios kalbos lygmens procesų modelius su juose naudojamų elektroninių paslaugų ir dalykinės srities duomenų modeliais. Pastaroji sąsaja užtikrinama siūlomose verslo taisyklių specifikacijose naudojant duomenų modelio elementus.

Šiuo metu intensyviai kuriami CASE įrankiai, siekiantys patenkinti daugybės notacijų, sąsajų ir kodo generavimo reikalavimus. Tam perspektyvi Eclipse platforma, kurios įskiepių (angl. plugin) architektūra kada nors leis efektyviai sąveikauti įvairiausioms metodologijoms, tačiau kol kas didžiausias problemas tam kelia įvairių standartų ir versijų nesuderinamumai.

Šiame darbe buvo eksperimentuojama su Magic Draw UML CASE įrankiu, kuris leidžia pereiti nuo vienos notacijos prie kitos įvedant grafinius stereotipus ir turi programavimo sąsają. Siūloma notacija ir taisyklių specifikacijos leidžia efektyviai bendradarbiauti verslo ekspertams, analitikams ir programuotojams elektroninio verslo procesų, paslaugų ir dokumentų kūrime.

Literatūros sąrašas

Amsden, J., et al. Draft UML Profile for Automated Business Processes with a Mapping to BPEL 1.0 version 1.1. 2002 gegužė [žiūrėta 2004 09 21] Prieiga per internetą: http://dwdemos.dfw.ibm.com.

Biron, P.V., Permanente, K., Malhotra A. XML Schema Part 2: Datatypes Second Editon, 2004 lapkritis [žiūrėta 2005 01 02] Prieiga per internetą: http://www.w3.org/TR/xmlschema-2.

BPEL4WS, Business Process Execusion Language for Web Services Version 1.1 Microsoft, IBM, 2003 gegužė [žiūrėta 2005 01 18] Prieiga per internetą : www.128.ibm.com, http://www.oasis-en.org.

EA Users Manual. Version 4.0. Sparx Systems Pty. Ltd., 2004 [žiūrėta 2004 10 10] Prieiga per internetą: http://www.sparxsystems.com.au.

OMG. Unified Modeling Language: Superstructure, Version 2.0. 2003 sausis [žiūrėta 2004 11 20] Prieiga per internetą: http://www.omg.org.

Pašilskytė, I., Nemuraitė, L. Verslo procesų modeliavimo kalbų analizė ir specifikavimo priemonių sukūrimas. Iš: Informacinės technologijos 2005: konferencijos pranešimų medžiaga. T. 2. Kaunas, 2005, p. 525-533.

Rational Software Architect, IBM, 2005 [Žiūrėta 2005 03 15] Prieiga per internetą: http://www-306. ibm. com/software/awdtools/ architect/ swarchitect.

MagicDraw. Users Manual, Version 9.0. No Magic Inc., 2005 kovas [žiūrėta 2005 03 02] Prieiga per internetą: www.magicdraw.com.

White, S.A. Business Process Modeling Notation Version 1.0. IBM Corporation, 2004 kovas.[žiūrėta 2005 01 02] Prieiga per internetą: www.bpmn.org.

Extended UML Notation for Modelling of e-Business Processes

There is no single language suitable for all phases of e-business process development: modelling, transforming to code and execution. For this purpose, the extension of UML 2.0 with BPMN stereotypes and specifications for capturing business process rules is proposed. The exclusive feature of proposed way of modelling lies in integration of business process model with object types of problem domain during specification of business process rules. The prototype for implementation of proposal is analyzed on the top of UML CASE tool Magic Draw.