KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKOS KATEDRA

Dainius Bendikas

PLA modeliavimo metodo panaudojimas programinės įrangos funkcionalumo analizei

Magistro darbas

Darbo vadovas
prof. habil. dr. Henrikas Pranevičius

KAUNAS, 2011

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKOS KATEDRA

Dainius Bendikas

PLA modeliavimo metodo panaudojimas programinės įrangos funkcionalumo analizei

Magistro darbas

Recenzentas
prof. habil. dr. Vytautas Štuikys
2011–05–27

Darbo vadovas
prof. habil. dr. Henrikas Pranevičius
2011–05–26

Atliko
IFM–9/1 gr. stud.
Dainius Bendikas
2011–05–27

KAUNAS, 2011

TURINYS

ĮVADAS

Sparčiai besivystant šiuolaikinėms technologijoms kuo toliau tuo labiau didėja sukuriamų programų skaičius. Sukuriamoms programoms aktualus kokybės klausimas. Kiekviena įmonė kurdama programinės įrangos produktą deklaruoja, kad jis kokybiškas ir darbas su programa pagerins įmonės ar įstaigos, kuriai programinė įranga pateikiama darbą. Kaip gali įmonė pagrįsti šiuos teiginius? Šio darbo tikslas – pasinaudojus PLA modeliavimo metodu parodyti, kaip galima išanalizuoti kuriamą sistemą ir pateikti sistemos darbo charakteristikas. Pagal gautas charakteristikas galimi siūlymai probleminėms sistemos darbo vietoms ir sprendimai, kaip optimizuoti sistemos darbą.

Dauguma įmonių, kurdamos programinę įrangą, naudoja BPMN arba UML modeliavimo kalbas, bet iš esmės jos nenusako sistemos darbo, o tik nurodo, kaip sistema turėtų veikti, tai yra jos padeda aprašyti sistemos darbą. PLA modeliavimo metodu galima aprašyti diskrečias būsenų aibes ir tolydinius laikus, iš kurių galimi analitiniai skaičiavimai. Jie ir nusakys mūsų tiriamos sistemos darbą.

UML (angl. Unified Modeling Language) – tai standartinė grafinė kalba, pritaikyta specifikuoti, vizualizuoti, projektuoti, konstruoti ir dokumentuoti artefaktus, sukuriamus kuriant programų ir kitas sistemas. UML modeliavimo kalba sparčiai populiarėja visame pasaulyje ir yra naudojama daugelio IT specialistų, kurie projektuoja programinę įrangą. UML aprašo sistemų analizės, projektavimo ir realizavimo aspektus.

BPMN (biznio procesų modeliavimo notacija) išsivystė kaip poreikis modeliuoti biznio procesus grafiškai. BPMN – tai OMG standartas, skirtas biznio procesams modeliuoti. Šis standartas aprašo tik vieną – biznio procesų diagramą, kuri labai artima UML veiksmų diagramai.

PLA – atkarpomis tiesinių agregatų formalizmas, naudojamas įvairių sistemų modeliavimui. PLA yra universali formali modeliavimo kalba. Šioje specifikacijoje procesų modeliai išreiškiami tekstinėmis ir matematinėmis išraiškomis. Pagrindinė šio metodo idėja – išskaidyti sistemą į atskirus atkarpomis tiesinius agregatus ir kiekvieną iš jų aprašyti naudojant kontrolines sekas.

Naudojant šį metodą verslo procesams formalizuoti, jie yra atvaizduojami kaip aibė sąveikaujančių agregatų. Agregatas yra objektas, apibrėžtas būsenų aibe Z, įeinančiais signalais X ir išeinančiais signalais Y.

Transformacija yra automatinis galutinio modelio generavimas iš pradinio modelio pagal transformacijos aprašą. Transformacijos aprašas yra transformacijos taisyklių rinkinys, aprašantis, kaip pirminio modelio realizacija tam tikroje kalboje gali būti transformuota į galutinio modelio realizaciją. Transformacijos taisyklė – tai aprašas kaip viena ar kelios konstrukcijos pirminio modelio kalboje gali būti transformuojamos į vieną ar keletą konstrukcijų galutinio modelio kalboje.

Tyrimo objektu pasirinkta nacionalinė verslo registravimo sistema, kuriama ir šiuo metu veikianti Vietname. Vietnamas – tai vis labiau augančios ekonomikos šalis. Kad augimo tempai

nemažėtų, o tik didėtų, stengiamasi pasinaudoti naujausiomis šiuolaikinėmis technologijomis. UNIDO (angl. United Nations Industrial Development Organization) ir Vietnamo vyriausybei vykdant verslo registravimo reformą šalyje yra kuriama nacionalinė verslo registravimo sistema.

NBRS – tai nacionalinė verslo registravimo sistema, kuriama Vietname, norint palengvinti ir struktūrizuoti įmonių užregistravimo, pakeitimo, sustabdymo ar išregistravimo mechanizmus. NBRS suskirstyta į dvi posistemes, tai yra pagrindinė registracija (angl. Basic registration), kuri yra pasiekiama iš 65 Vietname esančių skyrių ir internetinė registracija (angl. Online registration), kuri pasiekiama kiekvienam vartotojui, kuris naudodamasis internetu prisiregistruoja prie NBRS.

NBRS padengia šį funkcionalumą:

- Naujų objektų registracija ir valdymas:
    - Naujų įmonės tipų registracija;
    - Filialų, atstovybių ir verslo vietų registracija.
- Pakeitimų valdymas ir registracija:
    - Įmonės duomenų keitimo registracija;
    - Įmonės sustabdymo registracija;
    - Įmonės atstatymo iš sustabdymo registracija;
    - Įmonės tipo pakeitimo registracija;
    - Įmonės ar kelių įmonių sujungimo registracija;
    - Įmonės išregistravimo registracija.

Norima ištirti nacionalinės verslo registravimo sistemos funkcionalumo charakteristikas ir nusakyti sistemos darbą:

- Vidutinį paraiškos apdorojimo sistemoje laiką;
- Sistemos apkrautumo koeficientą;
- Sistemos naudingumo koeficientą;
- Vidutinį sistemoje esančių paraiškų skaičių.

Darbo tikslai:

- Aprašyti UML, BPMN ir PLA modeliavimo kalbas;
- Išanalizuoti transformacijas tarp modeliavimo kalbų;
- Pasinaudojus PLA modeliavimo metodu suskaičiuoti nacionalinės verslo registravimo sistemos funkcionalumo charakteristikas;
- Pasinaudojus PLA modeliavimo metodo gautom nacionalinės verslo registravimo sistemos funkcionalumo charakteristikom nurodyti silpnąsias sistemos vietas ir pateikti sprendimus sistemos darbui pagerinti.

1. OBJEKTINIS MODELIAVIMAS

1.1. OBJEKTINIO MODELIAVIMO PAGRINDINĖS KONCEPCIJOS

Objektinio modeliavimo svarba ir tikslingumas pagrindžiami taip objektinio modeliavimo metodai gali padidinti modeliuojamų sistemų ir jų komponentų lankstumą ir pakartotinį panaudojimą. Tai leidžia greičiau kurti aukštesnės kokybės sistemas [1].

Toliau šiame skyriuje smulkiau nagrinėjami pagrindiniai objektinio modeliavimo principai. Objektinio modeliavimo paradigma teigia, kad realaus pasaulio sistemas galima modeliuoti turint aibę klasių ir ryšius tarp jų. Siekiant aprašyti ir modeliuoti sritį ir joje esančias sistemas, virš srities abstrakcijų lygmens įvedamas naujas abstrakcijos lygmuo. Pagrindiniai šio aukštesnio abstrakcijos lygmens veikėjai yra klasės – labai abstrakčios struktūros, kurios apima srities duomenis ir su jais atliekamas operacijas. Klasės yra susietos viena su kita sąryšių tinklu. Šie sąryšiai atvaizduoja skirtingus klasifikavimo ir sąveikos tarp srities esybių tipus. Objektinės srities analizės tikslas yra išskaidyti srities sistemą į atskiras klases (objektus) ir ryšius tarp jų.

Objektas yra esybė, kuri turi būseną ir apibrėžtą aibę operacijų. Jos keičia objekto būseną ir atlieka konkrečius veiksmus. Būsena yra objekto savybių rinkinys. Su objektu susietos operacijos teikia servisus (paslaugas) kitiems objektams, kurie jų reikalauja, kai reikia atlikti kokius nors veiksmus. Objektai yra kuriami pagal objektų klasių apibrėžtį. Objekto klasės apibrėžtis naudojama kaip objekto šablonas. Joje yra nurodytos visos savybės bei servisai, kurie turėtų būti susieti su šios klasės objektu.

Objektinio modeliavimo metodologija iš esmės yra evoliucinio projektavimo metodologija, kuri yra ypač orientuota į praktinį projektavimo pakartotiniam naudojimui principo taikymą. Klasės yra specialiai projektuojamos su galimybe vėliau jas išplėsti naudojant paveldėjimo mechanizmą.

Objektinio modeliavimo pagrindiniai metodai yra keturi:

- abstrahavimas;
- koncepcijų atskirtis;
- kompozicija (komponavimas);
- apibendrinimas.

Abstrahavimas – tai srities objektų, jų savybių, struktūros ir elgsenos atvaizdavimas abstraktesniu pavidalu, pavyzdžiui, naudojant klases ir objektus. Koncepcijų atskirtis – tai atskiras srities aspektų specifikavimas ir realizavimas, pavyzdžiui, atskiriant sąsajas nuo konkrečių paslaugų, metodus nuo duomenų ir panašiai. Kompozicija – tai komponentų apjungimas siekiant gauti norimą projektuojamos sistemos elgseną. Komponentai gali būti sujungiami pranešimų perdavimu (asociacija)

arba fiziškai įdedami vienas i kitą (agregacija). Apibendrinimas leidžia nustatyti ir apjungti bendras srities objektų (klasių) savybes, elgseną arba struktūrą naudojant klasių hierarchiją, klasių šablonus, polimorfizmą arba projektavimo šablonus.

Objektinio modeliavimo charakteristikas galima apibendrinti taip [1]:

- Objektai yra realaus pasaulio arba kompiuterinės sistemos būsenų abstrakcijos ir valdo patys save;
- Objektas slepia savo vidinę būseną – taip pasiekiama objektų tarpusavio nepriklausomybė. Nepriklausomus objektus patogu naudoti pakartotinai, skirtingose programose;
- Sistemos funkcijos suvokiamos kaip atskirų objektų teikiamos paslaugos. Bendros duomenų sritys eliminuojamos. Objektai bendrauja perduodami pranešimus;
- Objektai gali būti paskirstyti ir gali būti vykdomi nuosekliai arba lygiagrečiai.

1.2. OBJEKTINIAI MODELIAI IR RYŠIAI

Objektinio modeliavimo metodologijos pagrindas – sistemos objektinis modelis. Modelis yra klasių hierarchijos egzempliorius, sudarytas iš tarpusavyje komunikuojančių objektų. Objektas – unikalus klasės egzempliorius, kuris yra struktūriškai identiškas kitiems tos klasės egzemplioriams. Jis apibrėžia savo teikiamų paslaugų (operacijų, metodų) sąsają, per kurią galima prieiti prie to objekto vidinių duomenų ir būsenos [1]. Objektiniams modeliams būdingi du aspektai:

- Semantinė informacija (semantika) – semantinis modelio aspektas aprašo sistemų kaip loginių konstrukcijų (klasių, asociacijų, būsenų, pranešimų, užduočių) tinklą. Semantinis modelis turi sintaksinę struktūrą, taisykles, nusakančias, kokie modeliai yra sintaksiškai teisingi, ir vykdymo dinamiką.
- Vizualioji pateiktis (notacija) – vizualizuojant modelį, semantinė informacija pateikiama pavidalu, pritaikytu peržiūrinėti ir redaguoti tą informaciją. Modelis pateikiamas žmogui lengvai suprantama forma.

Objektinis srities sistemos modelis gali būti atvaizduotas trijuose abstrakcijos lygmenyse:

1. Realizavimo lygmenyje – klasės atvaizduoja programos kodo dalis, parašytas objektine programavimo kalba;
2. Specifikavimo lygmenyje – klasės specifikuoja sistemos sąsajas aukštesniame abstrakcijos lygmenyje;
3. Konceptualiajame lygmenyje – klasės atvaizduoja abstrakčias tyrimo srities sąvokas, pavyzdžiui, platformas.

Bendrinius objektinius modelius galima aprašyti naudojant polimorfizmą. Polimorfizmas leidžia manipuliuoti su skirtingų klasių objektais žinant tik jų bendras savybes ir nekreipiant dėmesio į konkretų klasės tipą. Tai leidžia vienodai traktuoti bazines klases ir naujas per paveldėjimą sukurtas klases, kurios išplečia bazines klases naujais duomenimis ir operacijomis su jais.

Objektinis srities modelis leidžia konceptualiai apjungti aukštesnį ir žemesnį srities abstrakcijos lygmenis. Objektiniai modeliai pateikia tik abstraktų srities sistemų vaizdą nenurodant konkretaus srities turinio. Šį žemesnio lygmens srities turinį reikia įvesti vėliau, kai objektiniai modeliai yra detalizuojami į srities sistemas [1].

Įvedus du skirtingus abstrakcijos lygmenis, reikia papildomai apibrėžti ir objektinį srities metamodelį, kuris aprašo aukštesniame abstrakcijos lygmenyje specifikuotų modelių transformavimą į žemesnio abstrakcijos lygmens modelius. Objektinio modelio realizacija priklauso nuo apibrėžto objektinio srities metamodelio ir transformacijų taisyklių. Viršutinio (objektinio) abstrakcijų lygmens modeliams aprašyti galima naudoti UML (angl. Unified Modeling Language) diagramas arba objektinio programavimo kalbą. Perėjimas į srities lygmenį gali būti atliktas naudojant įvairius srities kodo generavimo metodus.

Struktūriniams ryšiams tarp klasių ir elgsenos sąveikoms tarp objektų aprašymui objektiniame modelyje naudojami trijų pagrindinių tipų sąryšiai:

- Paveldėjimas;
- Agregacija;
- Asociacija.

Paveldėjimas leidžia poklasiui paveldėti duomenis ir operacijas, apibrėžtas savo tėvinėje klasėje ir papildyti jas papildomais duomenimis ir metodais. Klasės gali būti sutvarkytos klasės hierarchijoje, kur viena klasė (superklasė) yra vienos ar kelių kitų klasių (poklasių) apibendrinimas. Poklasis paveldi atributus ir operacijas iš savo superklasės ir gali būti papildytas naujais metodais ir atributais. Paveldėjimas – tai abstrakcijos mechanizmas, kuris gali būti panaudotas įvairių esybių klasifikavimui. Be to, tai yra pakartotinio panaudojimo mechanizmas tiek projektavimo, tiek programavimo lygmenyje, o klasių hierarchija yra organizacinių žinių apie sritis ir sistemas šaltinis.

Agregacija yra naudojama, kai vienas objektas (konteineris) fiziškai arba konceptualiai turi kitą objektą (komponentą). UML dar apibrėžia stipresnį agregacijos atvejį, vadinamą kompozicija. Šis ryšys stipresnis ta prasme, kad objektas kaip dalis vienu metu gali priklausyti tik vienam sudėtiniam objektui. Be to, jų gyvavimo trukmė sutampa, t.y. objektas – dalis yra sukuriamas ir sunaikinamas vienu metu kaip ir visas sudėtinis objektas [1].

Asociacija atvaizduoja konceptualius ryšius tarp klasių ir yra naudojama pranešimų keitimuisi tarp objektų, pavyzdžiui, kai viena klasė „žino“ apie kitą klasę ir naudoja jos operacijas arba atributus.

1.3. MODELIŲ TRANSFORMACIJŲ TAKSONOMIJA

Kleppe et al. [2] siūlo tokį modelių transformacijos apibrėžimą: Transformacija yra automatinis galutinio modelio generavimas iš pradinio modelio pagal transformacijos aprašą. Transformacijos aprašas yra transformacijos taisyklių rinkinys, aprašantis, kaip pirminio modelio realizacija tam tikroje kalboje gali būti transformuota į galutinio modelio realizaciją. Transformacijos taisyklė – tai aprašas, kaip viena ar kelios konstrukcijos pirminio modelio kalboje gali būti transformuojamos į vieną ar keletą konstrukcijų galutinio modelio kalboje.

Anot [1] modelių transformacija siūlo aprašyti atvaizdavimo ryšius (angl. mappings) tarp modelių sąvokų (t.y. metamodelių elementų) ir modelių rinkinių. Čia atvaizdavimo ryšys apibrėžia atitiktį tarp įvesties ir paskirties modelių elementų. Kadangi modelių sąvoka yra bendresnė negu programos kodo sąvoka, modelių transformacijos apima didesnį skaičių programinės įrangos artefaktų negu programų transformacija, pavyzdžiui, UML modeliai, sąsajos specifikacijos, duomenų schemos ir programų įvesties tekstai.

Modelių transformacija yra vieno ar daugiau pirminių modelių vertimas į paskirties modelį [MDA]. Modelių transformacijas galima apibrėžti trijose lygmenyse: metamodelio, modelio dalių arba egzempliorių [3]. Transformacija metamodelio lygmenyje reiškia tam tikrų modelių elementų, kurie bus transformuojami laikantis nustatytų taisyklių, specifikavimą.

Modelių transformacijoje gali būti išskirtos dvi pagrindinės dalys: transformacija ir transformavimo įrankis. Modelių transformacijos gali būti užrašomos, t.y. specifikuojamos trimis pagrindiniais būdais [4]:

1. Imperatyviai – naudojantis deklaratyviosiomis kalbomis, tiksliai nusakant transformacijos eigą.
2. Deklaratyviai – apibrėžiant atskirų modelio dalių susiejimo taisykles, transformavimo įrankis atsako už jų vykdymo eiliškumą.
3. Mišriai – naudojant abu būdus.

Modelių transformacija yra artimai susijusi su modeliavimo sritimi. Pačias transformacijas taip pat galima modeliuoti naudojant modelius ir metamodelius [5]. Transformacijas galima nagrinėti kaip užfiksuotas vykdomąsias specifikacijas, architektūrinius šablonus ar metamodelius, kuriuos galima taikyti įvairiame kontekste.

Modelių transformacijas galima klasifikuoti kaip endogenines (angl. endogenous) ir egzogenines (angl. exogenous) [4]. Endogeninės transformacijos yra modelių, aprašytų naudojant tą pačią kalbą (notaciją), transformavimas, pavyzdžiui, optimizavimas, faktorizavimas, prastinimas. Egzogeninės transformacijos yra modelių, aprašytų naudojant skirtingas kalbas (notacijas), transformavimas, pavyzdžiui, kodo generavimas, apgrąžos inžinerija, migravimas. Taip pat galima skirti horizontalias

transformacijas, kai išeities ir paskirties modeliai yra to paties abstrakcijos lygmens, ir vertikalias transformacijas, kai išeities ir paskirties modeliai yra skirtingo abstrakcijos lygmens.

Pagal [6], UML modelių transformacijas galima klasifikuoti taip: detalizavimas, kokybės pagerinimas, išplėtojimas (angl. Elaboration), abstrakcijos lygio kėlimas ir projektavimo šablonai.

Remiantis įvairiais architektūriniais transformacijų apibrėžties metodais, modelių transformavimo metodus galima klasifikuoti į tiesioginio modelio manipuliavimo, tarpinio atvaizdavimo ir transformacijų kalbos metodus [7]. Tiesioginio modelių manipuliavimo metode vartotojai gali prieiti prie vidinių modelių atributų reikšmių ir jas keisti. Tarpinio atvaizdavimo metode, modelius galima atvaizduoti standartiniu XML (angl. eXtensible Markup Language) grįstu formatu ir transformavimui naudoti išorinį įrankį.

Transformacijos kalbos metode modelių transformacija yra realizuojama naudojant transformavimo kalbą, kuri turi modelių transformacijų aprašymo ir vykdymo konstrukcijas arba mechanizmus.

Iš MDA (angl. Model Driven Architecture) perspektyvos modelių transformacijos gali būti klasifikuojamos taip [2]:

- Nuo platformos nepriklausančio modelio (angl. Platform Independent Model (PIM)) – nuo platformos priklausančio modelio (angl. Platform Specific Model (PSM)) transformacijos,
- PSM – kodo transformacijos,
- PIM – kodo transformacijos,
- Priderinamos transformacijos – tai yra parametrizuotos transformacijos, aprašytos vidine įrankių scenarijų kalba, kurią galima pritaikyti prie vartotojo reikalavimų.

Anot [1] yra šios svarbiausios modelių transformacijų charakteristikos:

- Automatizavimo laipsnis (rankinis, pusiau automatinis, automatinis);
- Sudėtingumas (jis gali būti įvertintas įvairiais metodais ir įrankiais);
- Tam tikrų savybių išsaugojimas (pvz., atliekant sudalinimą reikia išsaugoti elgseną, kai tuo tarpu struktūra gali būti pakeista).

Taip pat transformacijos charakterizuojamos:

- Kryptiškumu: vienakryptė ar dvikryptė – kaip gali vykti transformacija.
- Kardinalumu: įeinančių ir išeinančių modelių kiekiu.

Modelių transformacijos vykdymo eiga:

1. Naudojant metakalbą specifikuojamos pirminio ir galutinio modelio kalbos, aprašant jų metamodelius.
2. Kuomet metamodeliai specifikuoti aprašoma transformacija tarp jų.

1.4. TRANSFORMAVIMO KALBOS

Informacinėse technologijose transformavimo kalbos yra ypač svarbios. Programinė sistema gali būti suvokiama kaip informacijos transformacijos rinkinys. Pati UNIX[1] sistema gali būti pagrindas naudoti ir kuri transformacijos programas remiantis bazinėmis transformacijų kalbomis GREP, SED, AWK, PERL ir kt.

Labiausiai paplitusios modelių transformacijos kalbos:

- ATL: populiari transformacijų kalba sukurta INRIA[2];
- QVT: OMG[3] standartas modelių transformavimui MOF/QVT trumpai: QVT;
- Beanbag: operacijomis paremta kalba, garantuojanti nuolatinį duomenų augimo vientisumą;
- GReAT: transformacijų kalba prieinama GME;
- MOLA: grafinė aukšto lygmens transformavimo kalba, įgyvendinta Lx pagrindu;
- Tefkat: transformavimo kalba ir modelių transformavimo variklis;
- Kitos – Kermeta, Lx, MT , SiTra, Stratego/XT, VIATRA.

Naudojant modeliavimo paketą MagicDraw UML, transformacija [MD] suteikia modelio ir duomenų migravimo tarp skirtingų UML profilių (egzogeninės) ir tarp skirtingo profilio versijų (endogeninės) transformacijos galimybę. Transformacija yra aprašoma deklaratyviai, grafiškai, naudojant modelį. Transformacija leidžia pakeisti elemento tipą, požymius, požymių tipus. Transformacija yra vienakryptė, skirta vieno pirminio modelio transformavimui į vieną galutinį.

1.5. KALBOS PASIRINKIMAS

Norint pasirikti tinkamiausią modelių transformavimo principą (kalbą ir įrankį) reikia įvertinti aukščiau išdėstytus argumentus. Renkantis gerai apibendrintus pasirinkimo kriterijus [4], reikia atsakyti į pateikiamus klausimus (Kas bus transformuojama? Koks transformacijos tikslas? Ar pakartotinė transformacija bus reikalinga? ir t.t.) ir įvertinti kriterijus.

Be jokių abejonių, tinkamiausias sprendimas yra modeliu paremtos egzogeninės transformacijos kalbos, leidžiančios atlikti transformacijas transformacijos kalbos metode. Transformacijos turi būti gerai palaikančios horizontalias transformacijas ir pakankamai populiarios, jog turėtų išvystytus brandžius transformavimo įrankius. Transformacijų aprašas gali būti ir imperatyvus, ir deklaratyvus.

---

[1] http://www.unix.org/

[2] http://www.inria.fr/

[3] http://www.omg.org/

Tokios transformavimo kalbos ir technologijos yra ATL [ATL] ir QVT [QVT]. Taip pat verta paminėti XSLT [XSLT] nors tai yra egzogenine transformacija modelių atžvilgiu, tačiau dėl glaudžios sąsajos tarp XML ir modelių, kuri yra dažniausia jų saugojimo kalba, XSLT yra ypač paplitusi modelių transformavime. Dėl šio derinio modeliai išeksportuoti į XML dažnai transformuojami XSLT pagalba, o vėliau vėl paverčiami modeliais.

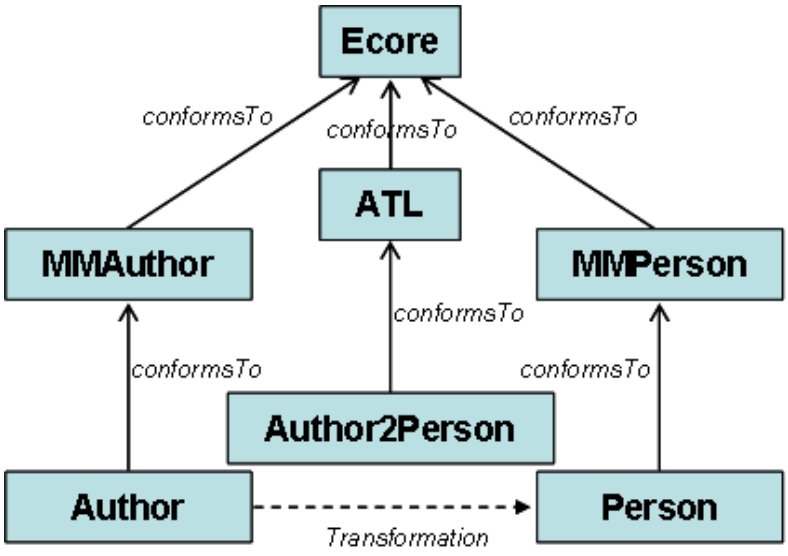
1.5.1. ATL

ATL (angl. ATLAS Transformation Language) yra modelių transformavimo kalba ir įrankių rinkinys. ATL naudojimas modeliais paremtoje inžinerijoje (angl. Model Driven Engineering) tikslas pagaminti daugybę galutinių modelių remiantis pirminiu modeliu. Pagal MDE principą transformacija taip pat aprašoma modeliu.

ATL, įgyvendinta populiarioje Eclipse[4] platformoje, kuri suteikia papildomas galimybes, sintaksės validavimą ir kitas galimybes bei transformacijų bibliotekas.

Šiuo metu yra keletas metamodeliavimo technologijų. ATL palaiko MOF (angl. Meta Object Facilities) apibrėžiamą OMG ir Ecore metamodelį [8]. Tai reiškia, jog ATL gali transformuoti metamodelius specifikuotus MOF ir Ecore.

ATL transformacijos pavyzdys yra pateiktas 1 paveiksle.



1 pav. ATL transformacijos pavyzdys

Šaltinis: [http://wiki.eclipse.org/ATL/Concepts]

Ši schema apžvelgia ATL transformaciją (Author2Person), kuri leidžia generuoti Person modelį atitinkantį MMPerson metamodelį iš Author modelio, o šis atitinka MMAuthor metamodelį. ATL

[4] http://www.eclipse.org/

aprašyta transformacija atitinka ATL metamodelį. Šiuo atveju transformacija yra egzogenine nors metamodeliai aprašyti naudojanti Ecore semantiką, tačiau transformacija atliekama tarp skirtingų metamodelių modelių.

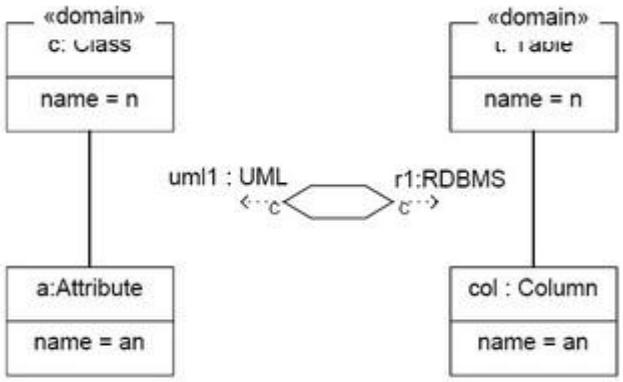
1.5.2. QVT

QVT (angl. Query/Views/Transformation) palaiko mišrią transformacijos užrašymo specifikaciją: deklaratyvią ir imperatyvią. Pagal [9] QVT yra OMG standartas – kalba transformacijų aprašymui MDA kontekste. Tai yra vienas iš svarbiausių modelių transformacijų standartų.

Užklausa – tai išraiška išrinkti modelį. OCL (angl. Object Constraint Language) yra viena iš žymiausių kalbų, skirta užklausoms. Modelis visiškai kilęs iš kito modelio yra vadinamas vaizdu. Dažniausiai vaizdas nėra saugomas ir negali būti keičiamas nepriklausomai nuo pradinio modelio. Programa, kuri iš pradinio modelio sukuria galutinį, vadinama transformacija.

QVT pavyzdys:

```
Relation UML2Rel {
        checkonly domain uml1
                c:Class {name = n, attribute = a:Attribute{name = an}}
        checkonly domain r1
                t:Table {name = n, column = col:Column{name = an}}
}
```



2 pav. QVT transformacija ir klasių į realecinių duomenų bazių modelį
Šaltinis: [http://visual-languages.blogspot.com/2007/11/mof-qvt.html]

2 paveikslas atvaizduoja klasių diagramos transformacijos į reliacinę duomenų schemą, transformacijos tarp klasės ir lentelės aprašas. QVT palyginus su ATL yra ganėtinai jauna kalba.

QVT principai:

- pirminis ir galutinis modeliai gali atitikti MOF;

- transformacijos aprašas yra modelis, taigi taip pat atitinka MOF metamodelį, t.y. abstrakti QVT sintakse atitinka MOF versijos 2.0 metamodelį.

Detaliau QVT integruoja OCL versijos 2.0 standartą ir išplečia jį iki imperatyvaus. QVT nusako net tris DSL kalbas (angl. Domain Specific Languages):

- Ryšių (angl. Relations);
- Centrinę (angl. Core);
- Operacinių (angl. Operational) sąsajų.

Šios kalbos yra suskirstytos į lygmenis. Ryšių ir centrinė yra deklaratyvios kalbos skirtinguose abstrakcijos lygmenyse su tarpusavio sąsajomis. Ryšių kalba turi tekstinę ir grafinę sintaksę. Operacinė sąsajų kalba yra imperatyvinė kalba, praplečianti ryšių ir centrinę kalbas. Operacinės kalbos sintaksės konstrukcijos randamos imperatyviose kalbose (ciklai, sąlygos, ir kt.).

Taip pat svarbi specifikacijos dalis yra mechanizmas, vadinamas QVT/Juoda dėžė (angl. QVT/BlackBox), skirtas iškviesti transformacijas, aprašytas kitomis kalbomis (XSLT, XQuery).

Šiuo metu QVT palaiko modelio – modelio transformacijas. Modelio į tekstą transformacijos yra už QVT specifikacijos ribų. Detalūs QVT principai aprašomi OMG standarte [QVT].

1.5.3. QVT IR ALT PALYGINIMAS

Tarp ATL ir QVT daug bendro: abi naudoja OCL, abi nusakomos MOF metamodeliu, abi palaiko XMI kaip pirminio modelio ir galutinio rezultato formatus. Tačiau šiuo metu ATL yra viena labiausiai QVT standartą atitinkančių atviro kodo transformacijos kalbų. Bet dėl savo komponentiškumo ATL turi potencialą būti lengviau plečiama už QVT standarto ribų. Praplečiantys ATL atvirojo kodo įrankiai: AMMA[5], AMW[6], AM3[7], TCS[8], ir t.t.

Pagal [9] ATL galima būtų traktuoti kaip skirtą spręsti daugiau problemų, nei QVT, kuri labiausiai orientuojasi į PIM ir PSM. Pavyzdžiui, AMMA platformoje turėtų būti įmanoma transformuoti binarinius failus, kuomet QVT apsiriboja „XMI į XMI transformacijomis“ su apribotu praplėtimu.

Iš architektūrinės puses QVT ir ATL lyginamos [10] pagal keletą kategorijų: abstraktumas, paradigma, kryptingumas, kardinalumas, ir t.t.. Rezultatai rodo, jog ATL ir QVT yra suderinami ir

---

[5] http://atlanmod.emn.fr/ammaroot/

[6] Del Fabro D. M., Bézivin J., Jouault F., Valduriez P., Applying Generic Model Management to Data Mapping. Journées Bases de Données Avancées, 2005, Saint Malo, France.

[7] http://www.eclipse.org/gmt/am3/

[8] http://www.eclipse.org/gmt/tcs/

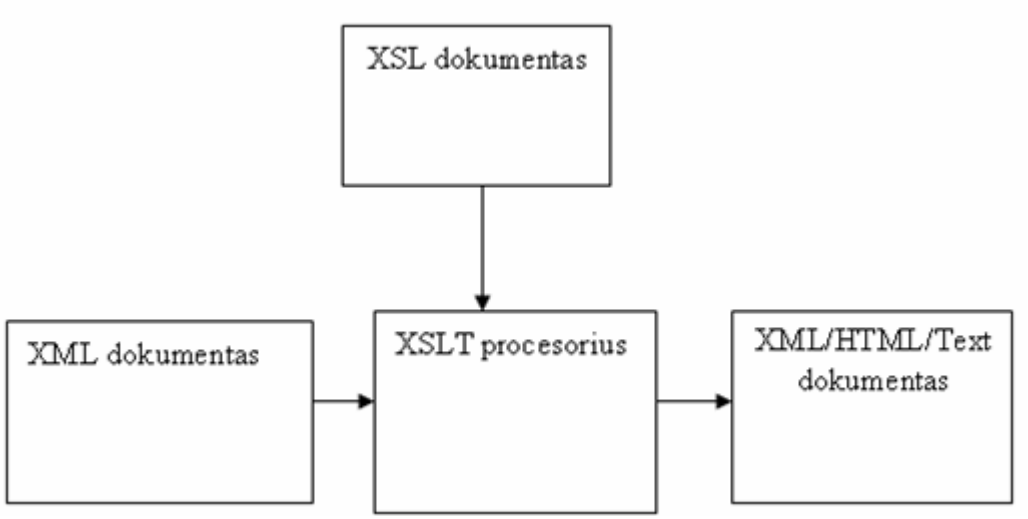
galimos tarpusavio transformacijos. Tad pasirinkus vieną iš šių transformavimo sprendimų būtų prieinama kitos įrankių bazė, bibliotekos, sprendimų analizės, standartų palaikymas, ir t.t.

1.5.4. XSLT

XSLT (angl. eXtensible Stylesheet Language Transformations) – kalba, aprašanti XML dokumento transformaciją į HTML (angl. Hyper text Markup Language) dokumentą arba į kitokios struktūros XML dokumentą. Ši kalba neretai naudojama tinklalapių turiniui tiesiogiai iš XML duomenų generuoti, tačiau tai tik vienas iš daugelio jos pritaikymų. XSLT naudoja XML sintaksę.

Nėra iki galo sutarta, ar šią kalbą galima laikyti pilnaverte programavimo kalba, tačiau ja įmanoma realizuoti sudėtingus algoritmus.

Transformavimo prielaidos tarp MOF metamodelio (BPMN) ir PLA, naudojantis XSLT yra išdėstytos [11]. Tiek BPMN ir UML modelius, tiek ir PLA modelį galima aprašyti XML formatu, tad galima taikyti tokią transformavimo kalbą, kuri vieną XML dokumentą transformuotų į kitą XML dokumentą. Tokia transformavimo kalba yra XSLT (angl. Extensible Stylesheet Language Transformations). XSLT yra sukurta naudojimui kaip XSL (angl. Extensible Stylesheet Language) kalbos dalis. 3 paveiksle parodytas bendras XSLT naudojimo atvejis.



3 pav. Bendras XSLT naudojimo atvejis

Šaltinis: sudarė autorius

2. SISTEMŲ FUNKCIONAVIMO FORMALIZAVIMAS AGREGATINIU METODU

2.1. ATKARPOMIS TIESINIAI AGREGATAI

Aprašomos sistemos [12] būsenų aibėje S išskiriama baigtinė pagrindinių būsenų aibė $I = \{0,1,2,\ldots,s\}$. Šios aibės elementai $v \in I$ yra pagrindinės agregato būsenos. Kiekvienai pagrindinei

būsenai priskiriamas sveikas neneigiamas skaičius $\| v \|$, kuris vadinamas būsenos rangu, bei išgaubtas daugiakampis $Z^{(v)}$, nusakytas $\| v \|$ išmatavimų Euklido erdvėje. Sakoma, kad būsenų aibę $Z = \bigcup_{v \in I} Z^{(v)}$ sudaro poros $(v, z^{(v)})$; čia $v \in I$; čia $z^{(v)} \in Z^{(v)}$. $z^{(v)}$ vadinamos papildomomis agregato koordinatėmis.

Pradiniu laiko momentu t_0 agregatas yra būsenoje $z(t_0) = (v, z^{(v)}(0))$; čia $z^{(v)}(0) \in Z^{(v)}$. Jei nėra įėjimo signalo, kai $t > t_0$, taškas $z^{(v)}(t)$ juda srityje $Z^{(v)}$ tol, kol pasiekia šios srities kontūrą. Laiko momentas t_1, kai pasiekiamas kontūras, vadinamas atraminiu.

Daugiakampio $Z^{(v)}$ kontūras aprašomas lygtimis:

$$\sum_{i=1}^{\|v\|} \gamma_{ji}^{(v)} z_i^{(v)} + \gamma_{j0}^{(v)} = 0, \; j = 1, \ldots, m(v);$$

čia $m(v);$ – kontūrų skaičius;

$z_i^{(v)}$ – vektoriaus $z^{(v)}$ dedamosios, $i = 1, \ldots, \| v \|$;

$\gamma_{ji}^{(v)}$ – sistemos parametrais apibūdinami dydžiai.

Atraminiu laiko momentu agregato pagrindinė būsena kinta iš v į v', o agregato papildomos koordinatės įgyja reikšmę $z^{(v')}(t_1) = Z^{(v')}$.

Papildomos koordinatės kinta srityje $Z^{(v')}$ tol, kol nepatenka ant šios srities kontūro. Tam įvykus, agregatas vėl keičia būseną.

Atkarpomis tiesinio agregato būsenai patekus į srities kontūrą, generuojamas išėjimo signalas $y \in Y$; čia Y – išėjimo signalų aibė, kuri yra analogiška aibei Z. Išėjimo signalo struktūra

$y = (\lambda, y^{(\lambda)});$

čia λ – išėjimo signalo diskrečioji dalis;

$y^{(\lambda)}$ – išėjimo signalo papildomų koordinačių vektorius, priklausantis nuo λ:

$y^{(\lambda)} = (y_1^{(\lambda)}, \ldots, y_{r(\lambda)}^{(\lambda)});$

Jei laiko momentu $t*$ į agregatą siunčiamas įėjimo signalas, tai agregato papildomos koordinatės nustoja kitusios ir agregato būsena akimirksniu pereina į kitą tos pačios ar kitos srities $Z^{(v'')}$ tašką.

Įėjimo signalas turi struktūrą, analogišką išėjimo signalo struktūrai, t.y.

$x = (\mu, x^{(\mu)}).$

Įėjimo signalo atėjimo momentu agregatas siunčia išėjimo signalą ir šis momentas taip pat yra atraminis. Toliau taškas $z^{(v'')}(t)$ srityje $Z^{(v'')}$ juda taip pat, kaip buvo aprašyta anksčiau.

Kai nėra įėjimo signalų, atkarpomis tiesinio agregato būsenos kitimas aprašomas tokiomis lygtimis:

$$v(t) = v = const; \frac{dz^{(v)}}{dt} = -\alpha^{(v)};$$

čia $\alpha^{(v)}$ – pastovus vektorius, turintis pavidalą

$$\alpha^{(v)} = (\alpha_1^{(v)}, \ldots, \alpha_{m(v)}^{(v)}).$$

Vektorine forma diferencialinės lygties sprendinys $z^{(v)}(t)$ gali būti užrašytas taip:

$$z^{(v)}(t) = z^{(v)}(0) + \alpha^{(v)}(t - t_0).$$

Spręsdami papildomų koordinačių kitimo ir sričių kontūrų lygtis, kartu galime apskaičiuoti ir laiko momentus, kai papildomų koordinačių reikšmės patenka į kontūrus. Pavyzdžiui, pirmasis atraminis laiko momentas

$$t_1 = \min_j [t : z^{(v)}(0) + \alpha^{(v)}(t - t_0) \in \bigcup_{i=1}^{m(v)} Z_j^{(v)}, t > t_0]$$

arba

$$t_1 = \min_j \left\{ t : t > t_0, \sum_{i=0}^{\|v\|} \gamma_{ji}^{(v)} \left[z_{ji}^{(v)}(0) + \alpha_i^{(v)}(t - t_0 \right] + \gamma_{j0}^{(v)} = 0 \right\}$$

Minimumas yra ieškomas indeksų $j = 1, \ldots, m(v)$ aibėje.

Jeigu pažymėsime

$$\tau_j = \left. -\left(\sum_{i=1}^{\|v\|} \gamma_{ji}^{(v)} z_i^{(v)}(0) + \gamma_{j0}^{(v)} \right) \right/ \sum_{i=1}^{\|v\|} \gamma_{ji}^{(v)} \alpha_i^{(v)}$$

ir

$$\tau = \min \{ \tau_j : \tau_j > 0 \},$$

tai laiko momentas, kai pirmą kartą pasiekiamas kontūras, yra

$$t_1 = t_0 + \tau.$$

Patekus į kontūrą, nauja pagrindinė būsena v' apibrėžiama tikimybių skirstiniu P_1, priklausančiu tik nuo būsenos $z(t_1)$. Norint apibrėžti papildomų koordinačių vektorių $z^{(v')}$, sudaromas pagalbinis vektorius η, kurio skirstinys nepriklauso nuo proceso istorijos, o priklauso tik nuo v ir $z^{(v)}$.

Papildomų koordinačių vektorius apskaičiuojamas:

$$z^{(v')} = \left(z_*^{(v)}, \eta \right) \times L_z^{(vv')}$$

čia $L_z^{(vv')}$ – matrica, kurios elementai priklauso nuo v, v' ir $z_*^{(v)}$;

$$z_*^{(v)} = z^v(\tau).$$

Kai ateina įėjimo signalas $(\mu, x^{(\mu)})$, naują pagrindinę būseną v'' nusako tikimybių skirstinys P_2, kuris priklauso nuo v, $z_*^{(v)}$, v'' ir μ.

Papildomų koordinačių vektoriui $z^{(v'')}$ apibrėžti sudaromas papildomas vektorius ξ, kurio pasiskirstymas nepriklauso nuo proceso istorijos, o priklauso nuo v, $z_*^{(v)}$ ir μ.

$$z^{(v'')} = \left(z_*^{(v)}, \xi, x^{(\mu)}\right) \times L_x^{(v,v'')};$$

čia $L_x^{(v,v'')}$ – matrica, kurios elementai priklauso nuo v, $z_*^{(v)}$, v'' ir μ.

Išėjimo signalas formuojamas, kai $z^{(v)}$ pasiekia kontūrą arba ateina įėjimo signalas (laiko momentu t').

Pirmuoju atveju

$$\lambda = \lambda\left(v, v', z_*^{(v)}\right),$$

$$y^{\lambda} = \left(z_*^{(v)}, \eta\right) \times L_y^{(vv')}$$

Matricos $L_y^{(vv')}$ elementai priklauso nuo v, $z_*^{(v)}$, v'.

Antruoju atveju:

$$\lambda = \lambda\left(v, v'', z^{(v)}(t'), \mu\right),$$

$$y^{\lambda} = \left(z^{(v)}(t'), \xi, x^{(\mu)}\right) \times L_y^{(vv'')}.$$

Matricos $L_y^{(vv'')}$ elementai priklauso nuo v, $z^{(v)}(t')$, v'' ir μ.

2.2. ATKARPOMIS TIESINIŲ AGREGATŲ FUNKCIONAVIMO GRAFINIS ILIUSTRAVIMAS

Atkarpomis tiesiniai agregatai priklauso automatų modelių klasei. Kaip ir automatas, atkarpomis tiesinis agregatas aprašomas nurodant būsenų aibę *Z*, įėjimo signalų aibę *Y* bei perėjimo operatorių (atvaizdavimą) *H* ir išėjimo operatorių *G*. Tačiau atkarpomis tiesinis agregatas turi nemažai ypatybių, skiriančių šį modelį nuo automatinių modelių.

Agregato funkcionavimas stebimas laiko momentų $t \in T$ aibėje, t.y. agregato būsena $z \in Z$ yra laiko funkcija $z(t)$. Atkarpomis tiesinio agregato būsenos struktūra yra tokia pat kaip ir atkarpomis tiesinio Markovo proceso, t.y.

$$z(t) = (v(t), z_v(t));$$

čia $v(t)$ – diskrečioji būsenos dedamoji,

$z_v(t)$ – tolydi būsenos dedamoji.

Bendruoju atveju

$v(t) = \{v_1(t), v_2(t), \ldots, v_m(t)\}, z_v(t) = \{z_{v1}(t), z_{v2}(t), \ldots, z_{vk}(t)\};$

čia t_i

$v_i(t)$ – i–oji diskrečiosios dedamosios koordinatė,

$z_{vi}(t)$ – i–oji tolydžiosios dedamosios koordinatė.

Kai nėra įėjimo signalų, agregato būsenos kinta taip:

$v(t) = const, \frac{dz_v(t)}{dt} = -\alpha_v;$

čia $\alpha_v = (\alpha_{v1}, \alpha_{v2}, \ldots, \alpha_{vk})$ – pastovusis vektorius.

Agregato būsena gali pakisti tik dviem atvejais: kai į agregatą siunčiamas įėjimo signalas arba kai viena iš tolydžiosios dedamosios koordinačių įgyja tam tikrą reikšmę.

3. UML ANALIZĖ

3.1. UML APRAŠYMAS

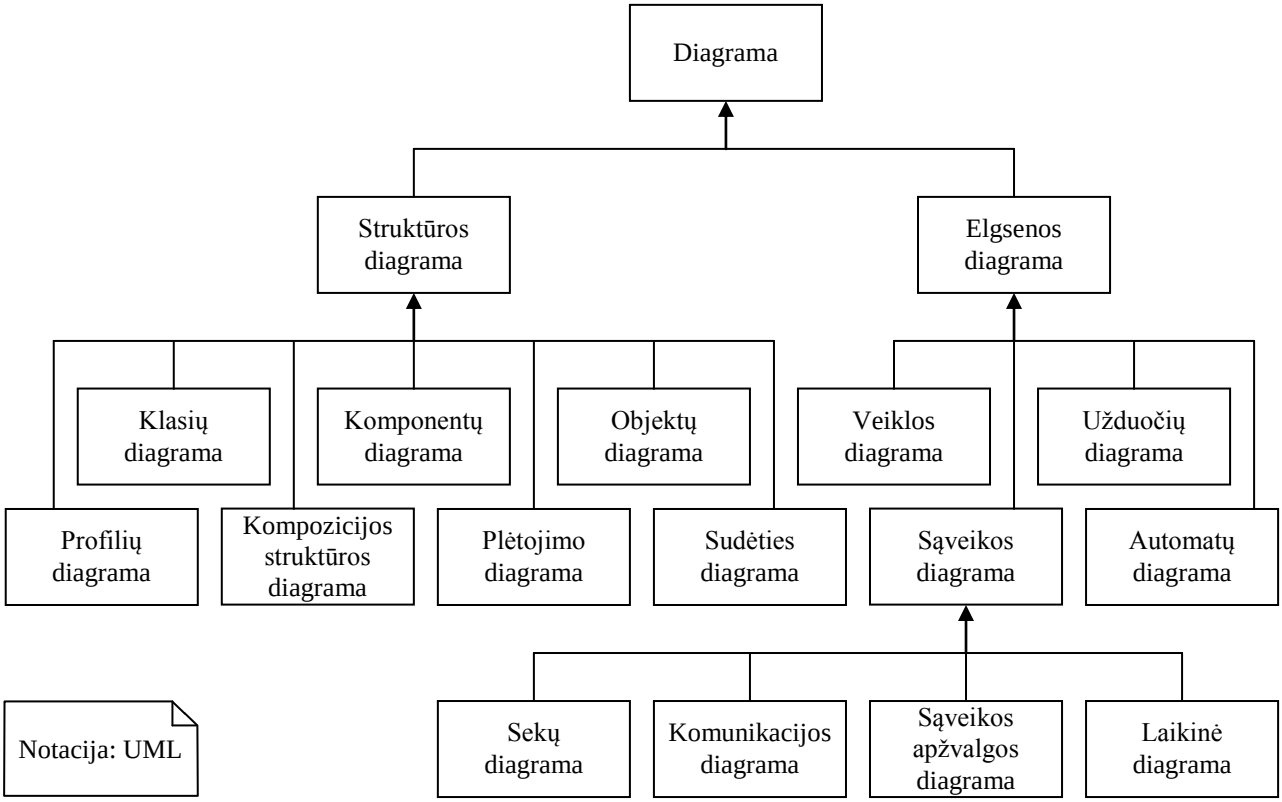
UML (angl. Unified Modeling Language) – tai standartinė grafinė kalba, pritaikyta specifikuoti, vizualizuoti, projektuoti, konstruoti ir dokumentuoti artefaktus, sukuriamus kuriant programų sistemas ir kitas sistemas. UML modeliavimo kalba sparčiai populiarėja visame pasaulyje ir yra naudojama daugelio IT specialistų, kurie projektuoja programinę įrangą. UML aprašo sistemų analizės, projektavimo ir realizavimo aspektus. Sistemos yra abstrakčiai atvaizduojamos modeliais naudojant gerai apibrėžtą sąvokų žodyną ir taisykles. Sistemų modeliai gali būti aprašomi tiksliai, nedviprasmiškai ir išbaigtai naudojant UML diagramas. UML yra vizuali kalba, turinti grafinę notaciją, skirtą įvairių programinės įrangos architektūros aspektų modeliavimui. UML modeliai leidžia greičiau ir lengviau suprasti programinės įrangos struktūrą ir veikimo principus, todėl yra efektyviai naudojami programinės įrangos architektūros dokumentavimui bei projektavimo sprendimų aptarimui. UML gali pateikti daug projektuojamos sistemos vaizdų, pasitelkdama įvairias struktūrines ir elgsenos diagramas.

UML diagramos naudojamos trims skirtingiems tikslams:

- modeliuoti realaus pasaulio sistemas;
- sistemoms specifikuoti ir projektuoti koncepciniu ir architektūriniu lygmenimis;
- kuriamoms sistemoms realizuoti eskiziniu ir detaliuoju lygmenimis.

3.2. UML NOTACIJA

UML versijoje 2.2 pateikiama 14 diagramų tipų, kurios suskirstytos į dvi kategorijas. Septyni diagramų tipai apibrėžia struktūrinę informaciją, likę septyni – pagrindinius elgsenos tipus, įskaitant penkis diagramų tipus, kurie apibrėžia skirtingus sąveikos aspektus. Šios diagramos gali būti suskirstytos į hierarchines kategorijas kaip parodyta 4 paveiksle.



4 pav. UML versijos 2.2 klasių diagrama

Šaltinis: [http://www.thefullwiki.org/Unified_Modeling_Language]

Projektuojant skirtingos UML diagramos yra naudojamos skirtingiems tikslams. Labai svarbu žinoti, kad UML pateikia tik bendro pobūdžio notaciją, kuri turi būti pritaikyta proceso vystymo metu. Taip pat UML versija 2.2 neriboja UML elementų tipų naudojimo konkrečiam diagramų tipui. Tai reiškia, kad kiekvienas UML elementas gali būti naudojamas beveik visuose diagramų tipuose. Ši lanksti savybė iš dalies buvo ribojama UML versijoje 2.0. UML profiliai gali apibrėžti papildomus diagramų tipus arba diagramos gali būti išplėstos papildomomis notacijomis.

Laikantis inžinerinio braižymo tradicijų, UML diagramose leidžiama rašyti komentarus ar užrašus, paaiškinančius naudojimo tikslus ar apribojimus.

3.3. STRUKTŪROS DIAGRAMOS

Struktūros diagramos pristato sistemos struktūrą ir yra plačiai naudojamos dokumentuojant programinės įrangos sistemų architektūrą. Struktūros diagramos apibrėžia tai, kas turi būti modeliuojamoje sistemoje:

- Statinės struktūros diagrama (arba klasių diagrama) (angl. Class diagram) paaiškina sistemos struktūrą, nurodydama sistemos klases, jų atributus ir ryšius tarp klasių;
- Komponentų diagrama (angl. Component diagram) vaizduoja kaip programinė įranga (sistema) yra padalinta į komponentus ir parodo šių komponentų tarpusavio ryšius;
- Kompozicijos struktūros diagrama (angl. Composite structure diagram) paaiškina vidinę klasės struktūrą ir šios struktūros galimą bendradarbiavimą;
- Plėtojimo diagrama (angl. Deployment diagram) naudojama modeliuojant sistemos techninę įrangą ir šios įrangos artefaktų išsidėstymą;
- Objektų diagrama (angl. Object diagram) parodo pilną arba dalinį suprojektuotos sistemos struktūrinį vaizdą atskirais laiko mementais;
- Sudėties diagrama (angl. Package diagram) apibūdina kaip sistema yra padalinta į logines grupuotes nurodant priklausomybes tarp šių grupuočių;
- Profilių diagrama (angl. Profile diagram) funkcionuoja metamodelio lygmenyje, parodydama stereotipus kaip klases su «stereotype» stereotipu ir profilius, kaip pakuotes su «profile» stereotipu. Išplėtimo ryšiai (vientisa linija su užpildyta rodykle) nurodo, kurį metamodelio elementą papildo stereotipas.

Klasių diagrama ko gero yra labiausiai žinoma UML diagrama. Pasinaudojant klasėmis ir ryšiais tarp jų nurodomi kuriamos sistemos struktūriniai aspektai. Klasės gali turėti atributus ir operacijas. Taip pat sąsajos bei apibendrinimo ryšiai leidžia sukurti objektiškai orientuotas hierarchijas. UML klasių diagramos atvaizduoja sistemos statinę struktūrą: sistemos objektus ir statinius ryšius tarp jų. Jos naudojamos modeliuoti probleminės srities sąvokas (pvz., analizuojant reikalavimus), modeliuoti posistemes ir sąsajas (pvz., kuriant sistemą), modeliuoti klases (pvz., taikant objektinį projektavimą).

Klasių diagramose naudojamos tokios sąvokos:

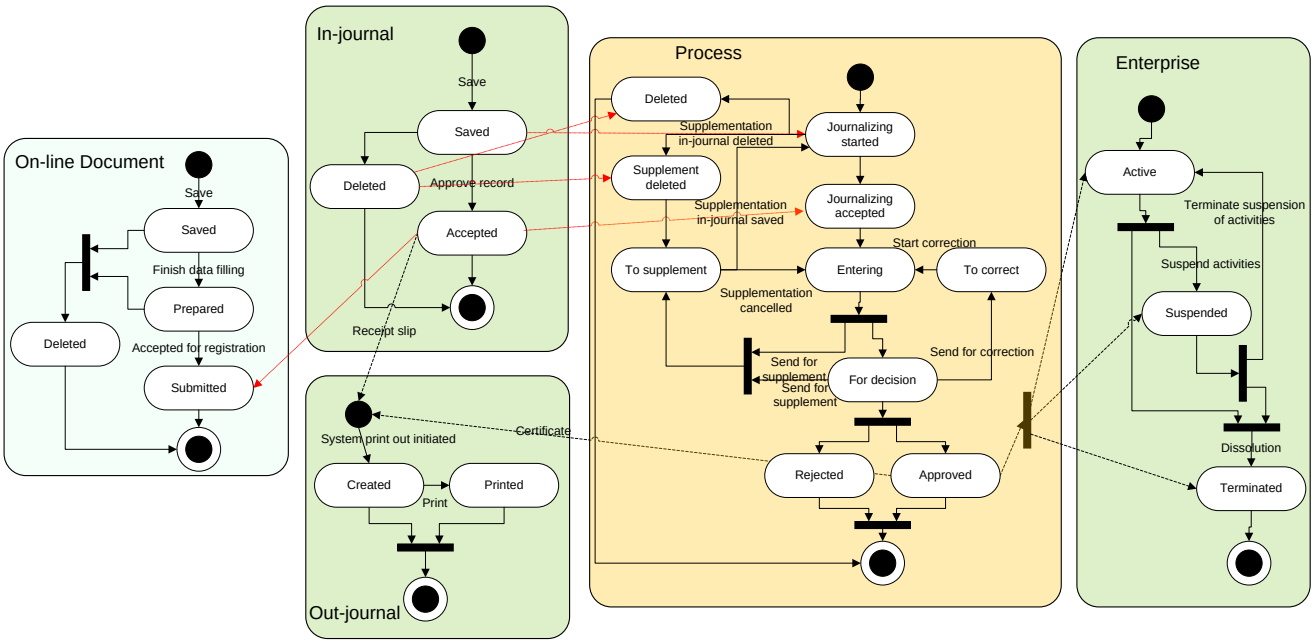
- Objektas – tai realaus pasaulio „daiktas“, atliekantis tam tikrus apibrėžtus veiksmus;
- Klasė – tai objektų, atliekančių panašius veiksmus abstrakcija, paprastai turinti kintamuosius (atributus) ir metodus (procedūrinį kodą);
- Atributas – tai vardinė klasės savybė, aprašanti reikšmių diapazoną, kurį gali įgyti klasės būsena;

- Metodas – tai objekto teikiama paslauga. Apribojimas – tai taisyklė, apibrėžianti galimas objektų, klasių, ryšių būsenas.

3.4. ELGSENOS DIAGRAMOS

Elgsenos diagramos yra naudojamos pavaizduoti programinės įrangos (sistemos) elgseną ir funkcionalumą. Elgsenos diagramos paaiškina tai, kas turi nutikti sistemos modeliavimo metu:

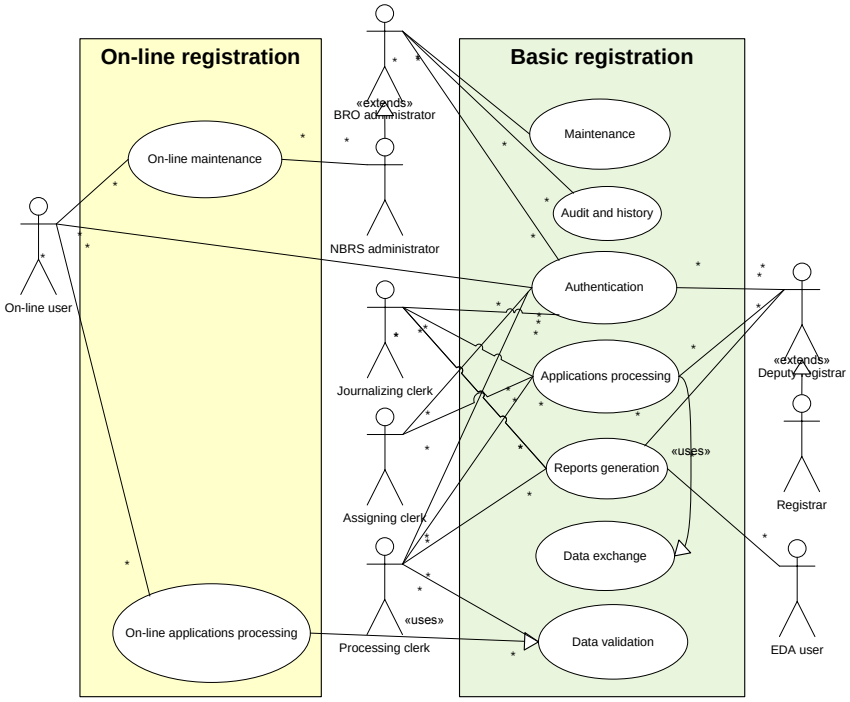
- Veiklos diagrama (angl. Activity diagram) pristato verslo ir veiklos sistemos komponentų darbo eigos žingsnius. Veiklos diagrama rodo bendrą valdymo eigą;
- Automatų diagrama (angl. State machine diagram) standartizuoja daugelio sistemų notaciją, nuo kompiuterių iki verslo procesų. Gali būti naudojama modeliuoti tinklo komunikacijų protokolams;



5 pav. Automatų diagrama NBRS projektui

Šaltinis: sudarė autorius

- Užduočių diagrama (arba panaudojimo atvejų diagrama) (angl. Use case diagram) parodo funkcionalumą, kuriuo aprūpinama sistema, remiantis tuo, kokiems atvejams skirta naudoti sistema ir kas bus sistemos vartotojas.



6 pav. Užduočių diagrama NBRS projektui

Šaltinis: sudarė autorius

Sistemos elgsenos modeliavimas aukščiausiu abstrakcijos lygmeniu prasideda nuo panaudos atvejų ir susijusių su sistema aktorių identifikavimo. Tai aprašoma panaudos atvejų diagramomis. Detalesnei elgsenos specifikacijai naudojamos sąveikos ir veiklos diagramos. Sąveikų diagramomis specifikuojamas bendravimas žinutėmis tarp sistemos komponentų. Visos šios diagramos kartu gali pateikti detalią kuriamos sistemos specifikaciją įvairiu abstrakcijos lygmeniu. Sąveikos diagramos apibūdina vieno sistemos objekto dinaminį elgesį kaip būsenų kaitą ir yra naudojamos parodyti visas būsenas, kuriose gali pabuvoti tam tikras objektas per savo gyvavimo ciklą. Sąveikos diagramos taip pat parodo, kokie įvykiai sužadina būsenos pasikeitimus.

3.5. SĄVEIKOS DIAGRAMOS

Sąveikos diagramos yra elgsenos diagramų sutrumpinimas, apibrėžiantis valdymo ir duomenų tėkmę tarp modeliuojamos sistemos elementų:

- Komunikacijos diagrama (angl. Communication diagram) parodo sąveiką tarp objektų ar dalių pagal iš eilės esančius pranešimus. Pranešimai pristato informacijos kombinacijas, paimtas iš Klasių, Sekų ir Panaudojimo atvejų diagramų, kurios nurodo sistemos statinę struktūrą ir jos dinaminę elgseną;
- Sąveikos apžvalgos diagrama (angl. Interaction overview diagram) yra veiklos diagrama, kurioje mazgai nurodo sąveikos diagramas;

- Sekų diagrama (angl. Sequence diagram) parodo kaip objektai komunikuoja tarpusavyje, naudojant pranešimų sekos terminus. Taip pat rodo objektų, susijusių su šiais pranešimais, gyvavimo trukmę;
- Laikinė diagrama (angl. Timing diagram) yra specifinė sąveikos diagrama, kurioje daugiausia dėmesio skiriama laiko apribojimams.

3.6. UML KALBA

UML kalba nesiejama nei su kokia nors konkrečiąja objektinio projektavimo metodika, nei su konkrečiu gyvavimo ciklo modeliu, nei su kokia nors programų sistemos dalykine architektūra ar kokia nors programavimo kalba. Tai universali koncepcinio modeliavimo ir objektinio projektavimo kalba, kurią galima naudoti kaip bet kurios konkrečiosios objektinio projektavimo metodikos komponentą.

UML kalbos aprašas formalizuotas. Kalbai aprašyti naudojamas formalizmas vadinamas UML metamodeliu. Šiuo formalizmu aprašyta kalbos semantika ir klausimai, susiję su UML vartojimu CASE sistemose (pavyzdžiui, mainų formatai).

UML kalba nėra grindžiama kuriuo nors vienu klasikiniu koncepcinio modeliavimo formalizmu. Iš tiesų tai rinkinys kelių tarpusavyje susietų kalbų, kiekviena iš kurių grindžiama kitu koncepcinio modeliavimo formalizmu. Kadangi UML kalbos notacija yra grafinė, tai kiekviena iš kalbų vartojama kito tipo diagramoms sudaryti. Kalbos suprojektuotos taip, kad su jomis būtų patogu dirbti CASE sistemose, pavyzdžiui, grafiniai žymenys suprojektuoti taip, kad juos būtų galima išskleisti arba suglausti kompiuterio ekrane.

3.7. MODELIŲ SPECIFIKAVIMAS NAUDOJANT UML

UML leidžia aprašyti tris skirtingus sistemos modelius: funkcinį modelį (sistemos elgsena vartotojo požiūriu), objektinį modelį (sistemos struktūra) ir dinaminį modelį (vidinė sistemos elgsena).

Modelis yra realaus pasaulio objekto supaprastintas atvaizdavimas. Aukšto lygmens specifikavimo notacijos ir abstrakcijos įvedimas leidžia išryškinti svarbiausius modelio aspektus. Gero modelio sukūrimas yra būtinas kuriamos programinės įrangos kokybei užtikrinti. Sistemos modelis padeda vizualizuoti, specifikuoti, konstruoti ir dokumentuoti kuriamą sistemą, padeda atskleisti ir suprasti įvairius kuriamos sistemos aspektus, padeda sistemų projektuotojams bendrauti tarpusavyje, leidžia išvengti pernelyg didelio sudėtingumo poveikio.

Grafinio modeliavimo privalumai, lyginant su formaliosiomis/abstrakčiosiomis specifikacijomis (kur naudojama specialiai tam tikslui sukurta srities kalba), tekstine programa arba aprašymu natūralia kalba yra šie [1]:

- Lengviau suprasti sudėtingas sistemas. Kompiuterinių sistemų sudėtingumas nuolat auga. Projektuotojams sunku „suturėti“ galvoje visas didelės sistemos produkto „plonybes“, atpažinti atskirų jo dalių sąveikos atvejus ir numatyti galimas tos sąveikos pasekmes;
- Supaprastinamas bendravimas tarp srities žinovų, analitikų, projektuotojų ir programuotojų. Vietoje srities terminologijos (žargono) arba prie konkrečios programavimo kalbos pririštų abstrakcijų naudojama viena universali sistemų modeliavimo kalba, kuri turi būti aiški, vienareikšmė ir paprasta, kad esant reikalui, srities žinovas galėtų ją sparčiai išmokti;
- Grafinis modeliavimas leidžia atskirti sistemos atvaizdą aukštame abstrakcijos lygmenyje (architektūrą) nuo jos realizavimui naudojamos kalbos. Toks atskyrimas leidžia ją realizuoti naudojant skirtingas kalbas bei technologijas;
- Dėka geresnio srities sistemų suvokimo, gaunamos programos, bibliotekos ir komponentai, kuriuos lengva plėtoti ar naudoti pakartotinai;
- Modeliuojant galima iš anksto, dar nepradėjus fizinės sistemos realizacijos, pastebėti ribines sistemos darbo sąlygas ir „kritines vietas“. Tai leidžia ištaisyti programas pirminėje projektavimo proceso stadijoje ir padidinti kuriamų sistemų patikimumą;
- Esant visuotinai priimtam modeliavimo standartui, galima kurti pagalbines priemones ir įrankius, kurie automatizuoja dalį darbo, atliekamo kuriant ir plėtojant kompiuterinius produktus.

Šiandien rinkoje galima rasti daug įvairių UML modeliavimo įrankių, pavyzdžiui: IBM Rational Rose, UMLStudio, MagicDraw ir kitų. Skirtingi įrankiai gali lengvai keistis UML modeliais. Pasikeitimą užtikrina XMI standartas ir XML paremta aprašymo kalba, kuri leidžia išsaugoti UML modelių diagramų duomenis pernešamu ir lengvai nuskaitomu formatu. Trumpai apžvelgsime kelis UML įrankius [1]:

- Rational Rose[9] – tai galingas komercinis UML įrankis, skirtas programinės įrangos projektavimui. Jis leidžia specifikuoti kuriamą sistemą įvairiais abstrakcijos lygmenimis, palaiko automatinį UML aprašų transformavimą į populiariausias programavimo kalbas (C++, Java, Visual Basic, Ada ir kitas). Rational Rose UML modeliavimo įrankis turi Rose Scripting kalbą, kuri sukurta Basic kalbos pagrindu ir yra skirta generatoriams kurti. Ji leidžia automatizuoti Rational Rose paketo specifines funkcijas ir atlikti veiksmus, kurie nepasiekiami vartotojui per Rational Rose paketo sąsają;

[9] http://www.ibm.com/developerworks/rational/products/rose/

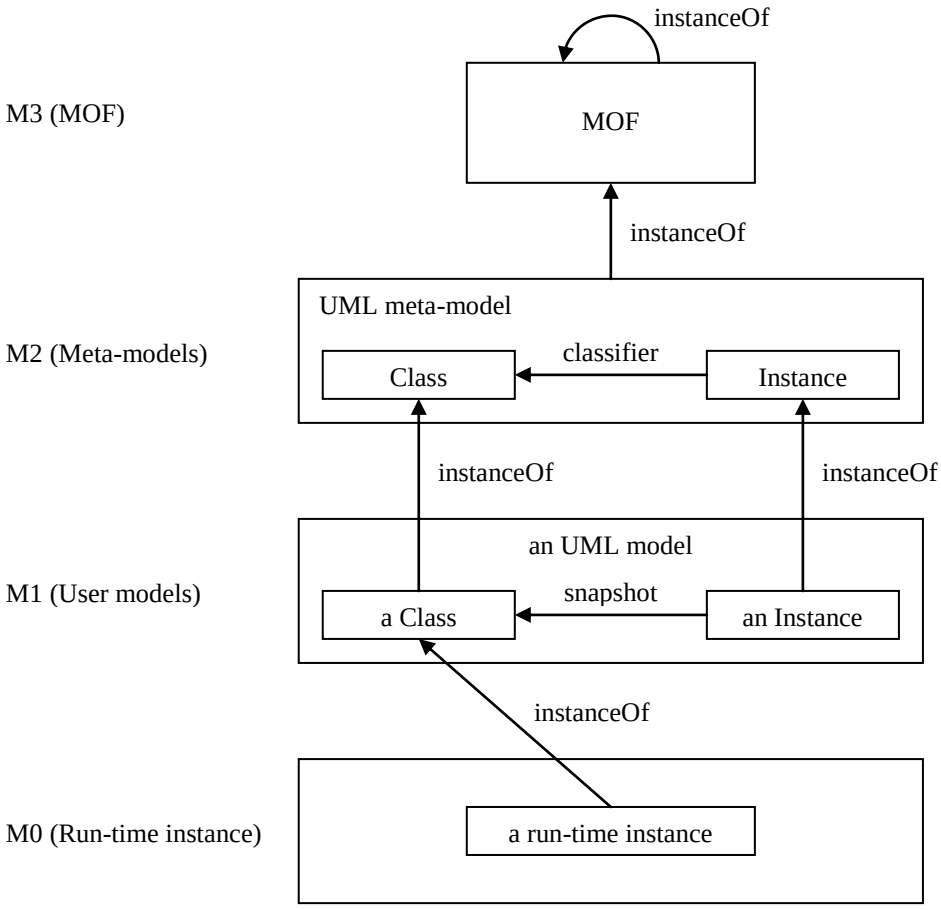
- MagicDraw[10] – programinės įrangos modeliavimo priemonė, veikianti Windows, Linux, Mac ir kitose labiausiai paplitusiose aplinkose. Šio produkto kūrėjai stengiasi šį įrankį padaryti kuo universalesnį, kad jį naudotų ne tik IT specialistai, bet ir srities vartotojai, įmonių ir projektų vadovai;
- Visual Paradigm[11] – UML modeliavimo įrankis, sukurtas plačiam vartotojų ratui: programinės įrangos inžinieriams, sistemų analitikams, verslo procesų analitikams, sistemų architektams. Šis įrankis palaiko naujausius UML ir Java kalbų standartus ir integruojamas į populiariausias Java integruotąsias programų kūrimo aplinkas.

3.8. UML METAMODELIO ARCHITEKTŪRA

MOF (angl. Meta–Object Facility) yra meta–modeliavimo architektūra, kurią pasiūlė OGM (angl. Object Management Group) [MOF]. MOF yra rekomenduojama kaip pagrindinė MDA (angl. Model-Driven Architecture) modeliavimo kalbos apibrėžimo technologija. Daugelis kitų OMG specifikacijų yra susijusios su MOF. Apibrėžiant UML architektūrą, kaip ir kitoms OMG specifikacijoms, MOF nustato keturis modelio lygmenis [MOF], nurodytus 7 paveiksle. Lygmenų reikšmės laikui bėgant keitėsi. Čia pristatoma UML versijos 2.0 Infrastruktūros Bibliotekos (angl. UML 2.0 Infrastructure Library) projekcija [UML].

[10] http://www.magicdraw.com/

[11] http://www.visual-paradigm.com/product/vpuml/



7 pav. Meta Object Facility (MOF) lygmenys

Šaltinis: [http://eleed.campussource.de/archive/1/81]

Žemiausią lygmenį M0 sudaro vykdomosios instancijos (angl. run–time instances). Šios instancijos laikomos modeliuojamo realaus pasaulio dalimi. Taipogi, M0 neturėtų būti vertinamas kaip modelio lygmuo. Vartotojas modeliuoja M0 instancijas, esančias M1 lygmenyje. Klasės ir jų instancijos kartu sudaro bendrą modelio lygmenį. Instancijos, esančios M1 (pvz. an Instance) yra vadinamos nuotraukomis (angl. snapshots). Jos atstovauja M0 elementus.

Modeliai, esantys M1, yra išreikšti modeliavimo kalba, kuri yra apibrėžiama meta-modelio priemonėmis, esančiomis M2 lygmenyje (pvz. UML Metamodelis). Galiausiai, viršutinį (aukščiausią) lygmenį M3 sudaro modelių iš M2 modelis, kitaip vadinamas metamodeliu, MOF modeliu arba paprasčiausiai MOF. Sakoma, jog MOF apibrėžia kalbą, leidžiančią kurti modeliavimo kalbas. MOF yra modeliuojama ta pačia kalba, kuria yra sudaryta. Ši savybė vadinama savireflekcija (angl. self-reflective).

Vienas iš reikalavimų, keliamų MOF, yra aprūpinti reflekcines aplikacijas modeliais ir įranga, kuri reikalinga modelių apsikeitimui (angl. model interchange). Tačiau tai įmanoma tik M1, M2 ir M3 lygiams. MOF semantika, dar žinoma kaip MOF abstrakti topografija (angl. abstract mapping) [MOF], apibrėžia M1 elementų struktūrą kaip egzistuojančius objektus su slotais, kurie turi reikšmes ir yra

susiję su kitais objektais per ryšius. Kadangi M3 lygmens modelis yra atspindintis M3 ir M2 lygmenis, M1 lygmens elementai atitinka tą pačia struktūrą. Taipogi visi elementai trijuose viršutiniuose lygmenyse dalijasi bendra struktūra ir gali būti traktuojami vienodai.

Verta pažymėti, jog pagrindinis MOF modelio tikslas yra apibrėžti modeliavimo kalbas M2 lygmenyje. Todėl MOF yra specifinės paskirties kalba. Tačiau dėl istorinių priežasčių dažniausiai pagrindinė MOF paskirtis yra modeliavimo kalba, apibrėžta kaip UML poaibis. Dabartinė praktika rodo, jog modeliuojama tik abstrakti kalbų sintaksė. Tam, kad MOF taptų kalba apibrėžiančia kitas kalbas, ji turi aprūpinti pirmos klasės (angl. firstclass) konstruktus, apibrėžiančius gerai žinomų kalbų elementus, tokius kaip tipai, instancijos, instanciniai ryšiai ir t.t.

3.9. UML 2.0 METAMODELIS

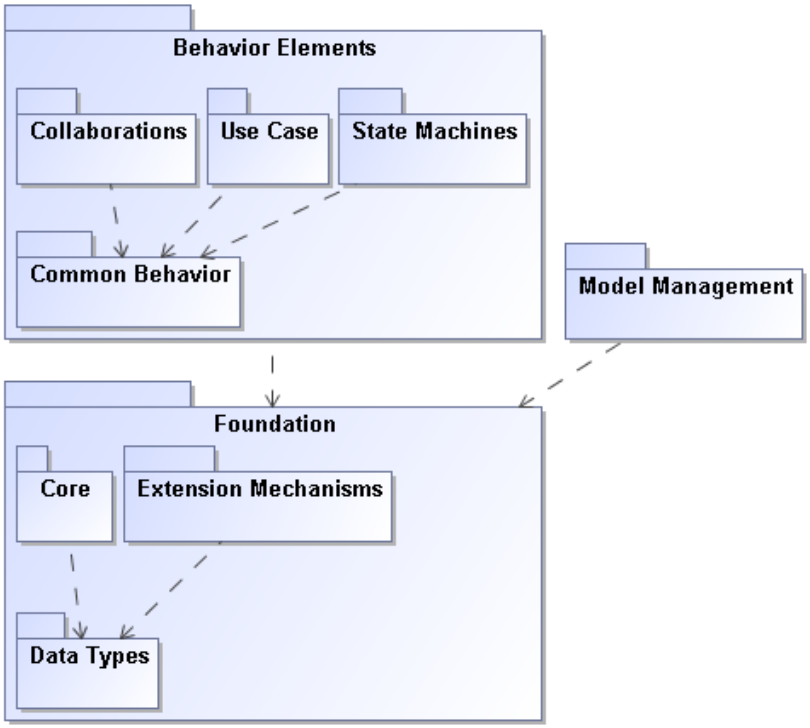
UML yra formaliai apibūdinamas naudojant metamodelį, kuris yra loginio UML konstrukto modelis. Metamodelis yra savaime išreiškiamas per UML. Tai metacirkuliarinio programos interpretatoriaus pavyzdys. Tai reiškia, kad kalba apibrėžiama tais pačiais terminais, kaip ir šios kalbos metamodelis. Tačiau procesas nėra visiškai žiedinis. Tik nedidelis UML poaibis yra naudojamas apibrėžti metamodelį. Iš esmės šis fiksuojamas apibrėžimo taškas galėtų būti įvadas į labiau apibendrintą/bazinį apibrėžimą. Tačiau praktiškai toks gilinimasis yra nereikalingas [21].

Kiekviena semantinio dokumento dalis (sekcija) turi klasių diagramą, parodančią dalį metamodelio. Tekstinis metamodelio klasės aprašymas yra apibrėžtas šioje sekcijoje kartu su apibrėžimo atributais (savybėmis) ir ryšiais. Elementų apribojimų sąrašas išreikštas natūralia kalba ir OCL (angl. Object Constraint Language). UML konstruktų dinaminės schemos tekstinis aprašymas taip pat apibrėžtas šioje sekcijoje. Dėl to dinaminė semantika yra neformali, tačiau formalus aprašymas būtų nepraktiškas ir beveik nesuvokiamas, sunkiai skaitomas. Pastabos užrašytos atskiroje sekcijoje (skyriuje), kuri remiasi semantiniu skyriumi ir metamodelių klasių simbolių planu.

3.10. METAMODELIO STRUKTŪRA

Metamodelis yra dalijamas į tris pagrindinius komplektus (angl. packages) pavaizduotus 8 paveiksle [MD]:

- Pagrindinis (angl. Foundation) komplektas apibrėžia statišką UML struktūrą;
- Elgsenos elementų (angl. Behavior Elements) komplektas apibrėžia dinamišką UML struktūrą;
- Modelio valdymo (angl. Model Management) komplektas apibrėžia organizacinę UML modelio struktūrą.



8 pav. UML metamodelio paketo struktūra

Šaltinis: [http://www.jot.fm/issues/issue_2010_03/article5/index.html]

3.11. PAMATINIS PAKETAS

Pamatinis komplektas (paketas) turi keturis subkomplektus:

- branduolys;
- duomenų tipai;
- išplėtimo mechanizmas.

Branduolio (angl. Core) komplektas apibrėžia pagrindinius nekintančius UML konstruktus. Juos sudaro klasifikatoriai, jų savybės ir ryšiai. Jų turinys apima požymius, operacijas, metodus ir parametrus, o ryšiai apima apibendrinimus, asociacijas, priklausomybės sritis. Kai kurios abstrakčios metaklasės yra apibrėžiamos apibendrinamais elementais, vardų erdve ir modelio elementas. Šis komplektas taip pat apibrėžia šabloną ir įvairias priklausomų subklasių rūšis bei komponentus, mazgus, komentarus.

Duomenų tipo (angl. Data Types) komplektas apibūdina duomenų tipų klases, naudojamas metamodelyje.

Išplėtimo mechanizmo (angl. Extension Mechanisms) komplektas apibūdina apribojimus, stereotipus ir žyminio įvertinimo mechanizmus.

3.12. PRIELAIDOS UML MODELIO TRANSFORMAVIMUI

UML veiklos diagramos naudojamos kaip pagrindinė grafinė proceso modeliavimo notacija. Dėl šios priežasties šiame skyriuje bus aptariami rekomenduojami veiklos diagramų poaibiai modeliuojant procesus ir pasiūlytas veiklos diagramų profilis, naudojamas apibrėžiant procesą. Keletas veiklos diagramų profilių, apibrėžiančių eigos procesus, jau yra pasiūlyti kitų autorių, tačiau nei vienas tyrinėtų metodų neatskleidžia visų privalumų keliamiems reikalavimams.

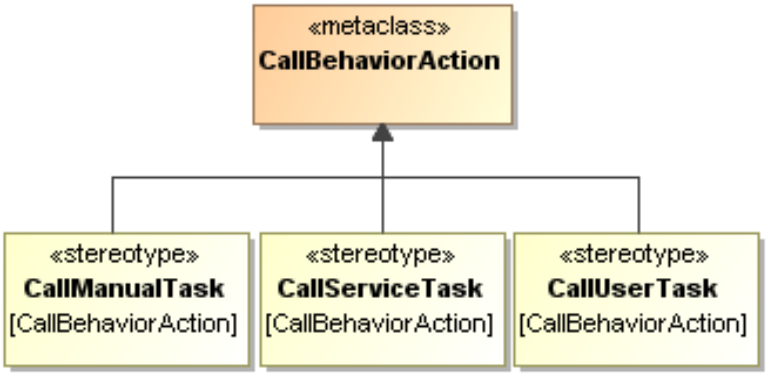
3.13. VEIKLOS DIAGRAMŲ POAIBIŲ APIBRĖŽIMAS EIGOS PROCESUOSE

Veiklos diagramų notacija turi daugumą funkcijų, kurios nedaro jokios įtakos eigos apibrėžimui, nes jie buvo įtraukti visiškai skirtingiems tikslams (apibrėžiant įterptųjų sistemų vykdomuosius procesus). Tačiau [13] pateikia eilę rekomenduojamų veiklos diagramų reikalavimų eigos apibrėžimui, išlaikančių originalią veiklos diagramų elementų semantiką. Stereotipai prijungia trūkstamas savybes ir apribojimus, kurių reikalauja eigos praktika (arba paaiškina semantiką tose vietose, kur originali semantika nepakankamai tiksli). Kadangi eigos (angl. workflows) tampa integraline sudėtingų verslo procesų dalimi, atsiranda būtinybė apibrėžti darbo srautus tarp skirtingų organizacijų.

Darbe apibūdinamas keleto susijusių eigų orkestravimas naudojant keletą veiklos diagramų (kiekvienam dalyviui), kurios transformuojamos į UML diagramas, apimančias kelis procesus (angl. pools). Paveiksle 20 parodytas paprastų bendradarbiaujančios darbo eigos proceso apibrėžimų pavyzdys, kuris nurodo keturias veiklos diagramas – pareiškėjo, žurnalizuojančio klerko, vykdančiojo klerko ir registratoriaus diagramas. Diagramose parodoma paraiškos įmonės įregistravimui kelias nuo pradinio taško iki galinio.

3.14. VEIKLOS DIAGRAMŲ PRAPLĖTIMO PROFILIS

Kadangi BPMN palaiko daugiau užduočių (angl. Task) tipo elementų nei UML veiklos diagrama turi veiksmo (angl. Action) tipų, konkretiems procesams atvaizduoti gali būti panaudojamas UML praplečiantis profailas, kaip siūloma [13]. Taip garantuojamas UML veiksmų ir BPMN užduočių atitikimas.



9 pav. Profilio veiklos diagramos praplėtimui

Šaltinis: sudarė autorius

4. BPMN ANALIZĖ

4.1. BPMN APRAŠYMAS

BPMN (biznio procesų modeliavimo notacija) išsivystė kaip poreikis modeliuoti biznio procesus grafiškai. Šis poreikis kilo 2001 metais, kuomet BPMI organizacija pradėjo kuri BPML (biznio procesų modeliavimo kalba) standartą. BPML laikui bėgant tapo BPEL (biznio procesų simuliavimo kalba), kuri vystėsi lygiagrečiai BPMN [18].

Pirmoji oficiali BPMN versija buvo išleista OMG grupės 2004 m.[18].

BPMN – tai OMG standartas, skirtas biznio procesams modeliuoti. Šis standartas aprašo tik vieną – biznio procesų diagramą, kuri labai artima UML veiksmų diagramai. Stephen‘as A. White‘as detaliai išanalizavęs 21 skirtingą darbų sekų realizavimo būdą (angl. workflow pattern), įrodė, kad 20 darbų sekų variantų gali būti sumodeliuoti ir su UML veiklos diagrama. Autorius kaip esminius skirtumus tarp procesų ir veiklos diagramų išskiria skirtingus konceptų pavadinimus, skirtingą kai kurių elementų notaciją ir skirtingą teorinį pagrindą. Šie faktoriai suteikia Biznio procesų diagramai privalumų modeliuojant biznio procesus. Autorius linkęs manyti, kad ateityje veiksmų diagrama ir jos plėtinys biznio procesų diagrama taps viena universalaus panaudojimo diagrama [14].

4.2. BPMN NOTACIJA

BPMN notacijos elementai suskirstyti į keturias pagrindines kategorijas [BPMN]:

- srauto objektai;
- jungiantys objektai;
- takeliai;
- artefaktai.

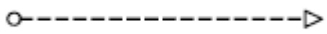
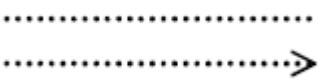
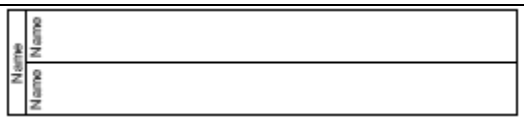
Srauto objektai yra pagrindiniai grafiniai elementai, apibrėžiantys veiklos procesų elgseną. Srauto objektai skirstomi į įvykius, veiklas ir vartus. Srauto objektai gali būti jungiami trijų tipų jungiančiaisiais objektais: seka, pranešimu arba asociacija [BPMN].

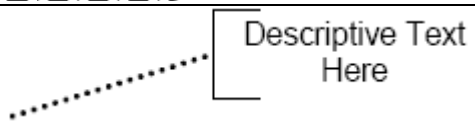
Taip vadinamieji takeliai, pirminiai modelio elementai, skirstomi į pulus ir takus, o artefaktai į duomenų objektus, grupes ir pastabas.

Visi šie elementai yra pagrindiniai BPMN notacijos elementai. Kiekvienas jų dar skirstomas į smulkius, konkrečius elementus. Pagrindiniai BPMN elementai, jų aprašymas ir notacija pateikiama lentelėje žemiau.

1 lentelė. Pagrindiniai BPMN elementai

Šaltinis: [http://bpmnhandbook.com/01_specs/BPMN_20_spec.pdf]

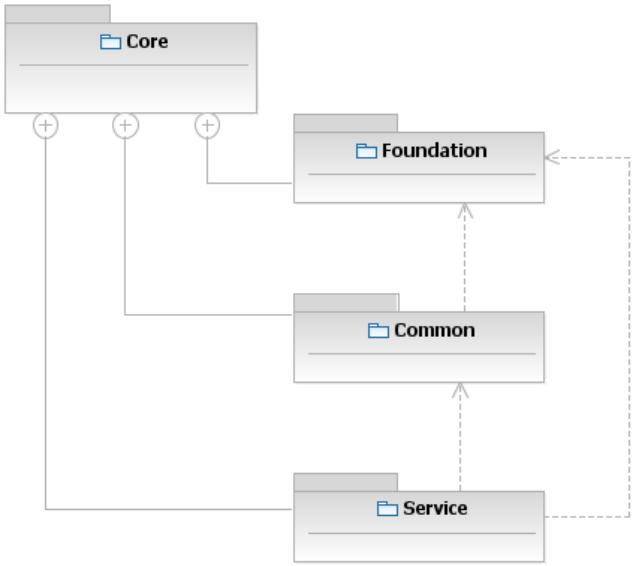
Pavadinimas	Notacija	Aprašas
Įvykis		Įvykis yra kažkas, kas nutinka veiklos procese. Įvykiai įtakoja proceso eigą ir rezultatą. BPMN įvykiai vaizduojami apskritimu su įvairiais atvaizdas jo viduryje. Pagal tai, kaip jie įtakoja proceso eigą, įvykiai skaidomi į pradžios, tarpinius ir pabaigos.
Veikla		Veikla dažniausiai apibrėžiama kaip darbas, kurį atlieka įmonė. Veiklos skirstomos į procesus, subprocesus ir užduotis.
Vartai		Vartai naudojami sekos išskyrimui į keletą sekų arba keleto sekų susiejimui į vieną.
Seka		Seka parodo, kokiu eiliškumu procese bus vykdomos veiklos, kurias šis ryšio elementas apjungia.
Pranešimas		Pranešimo ryšys naudojamas parodyti dviejų proceso dalyvių tarpusavio sąveiką.
Asociacija		Asociacija naudojama susieti informacijai su srauto objektais. Ji gali būti kryptinė, su nukreipiančia rodykle arba nekryptinė – be rodyklės.
Pulas		Pulas skirtas atvaizduoti proceso dalyviui.
Takelis		Takelis naudojamas suskirstyti veikloms, esančioms viename pule.

Pavadinimas	Notacija	Aprašas
Duomenų objektas		Duomenų objektas skirtas atvaizduoti atskirų veiklų įeigoms ir išeigoms ir jokios įtakos veiksmų sekai procese neturi.
Grupė		Elementas skirtas veikloms grupuoti.
Pastaba		Elementas skirtas parodyti papildomai informacijai apie tam tikrą objektą.

4.3. BPMN 2.0 METAMODELIS

BPMN 2.0 versijoje atlikti techniniai ir notaciniai pakeitimai. Pagrindiniai notaciniai pakeitimai apima naujas choreografijos ir pokalbio (angl. conversation) diagramas. Be šių pakeitimu, atsirado nauji nepertraukiantieji įvykiai (angl. non-interrupting Events) ir įvykio paprocesiai (angl. Event Sub-Processes). Pagrindiniai techniniai pakeitimai apima formalaus metamodelio atsiradimą, išreikštą klasių diagramomis, apsikeitimo formatų atnaujinimus ir XSLT transformacijas tarp XMI ir XSD formatų [BPMN].

BPMN 2.0 branduolys susideda iš bendrinių (angl. Common), pamatinių (angl. Foundation) ir paslaugų (angl. Service) konceptų (10 pav.).



10 pav. BPMN konceptų grupės

Šaltinis: [http://bpmnhandbook.com/01_specs/BPMN_20_spec.pdf]

Pamatiniai BPMN konceptai – tai baziniai struktūros komponentai, aprašantys pagrindinę BPMN modeliavimo semantiką. Dauguma pamatinių konceptų yra abstraktūs ir tiesiogiai modelyje nenaudojami.

BPMN 1.2 notacijos elementai buvo suskirstyti į keturias pagrindines kategorijas [BPMN]:

- Srauto objektai;
- Jungiantys objektai;
- Takeliai;
- Artefaktai.

BPMN 2.0 visos šios elementų grupės taip pat egzistuoja, tačiau skirstymas į grupes kur kas platesnis. Srauto, jungiantys elementai, takeliai ir artefaktai naujajame BPMN metamodelyje talpinami tarp bendrinių BPMN elementų.

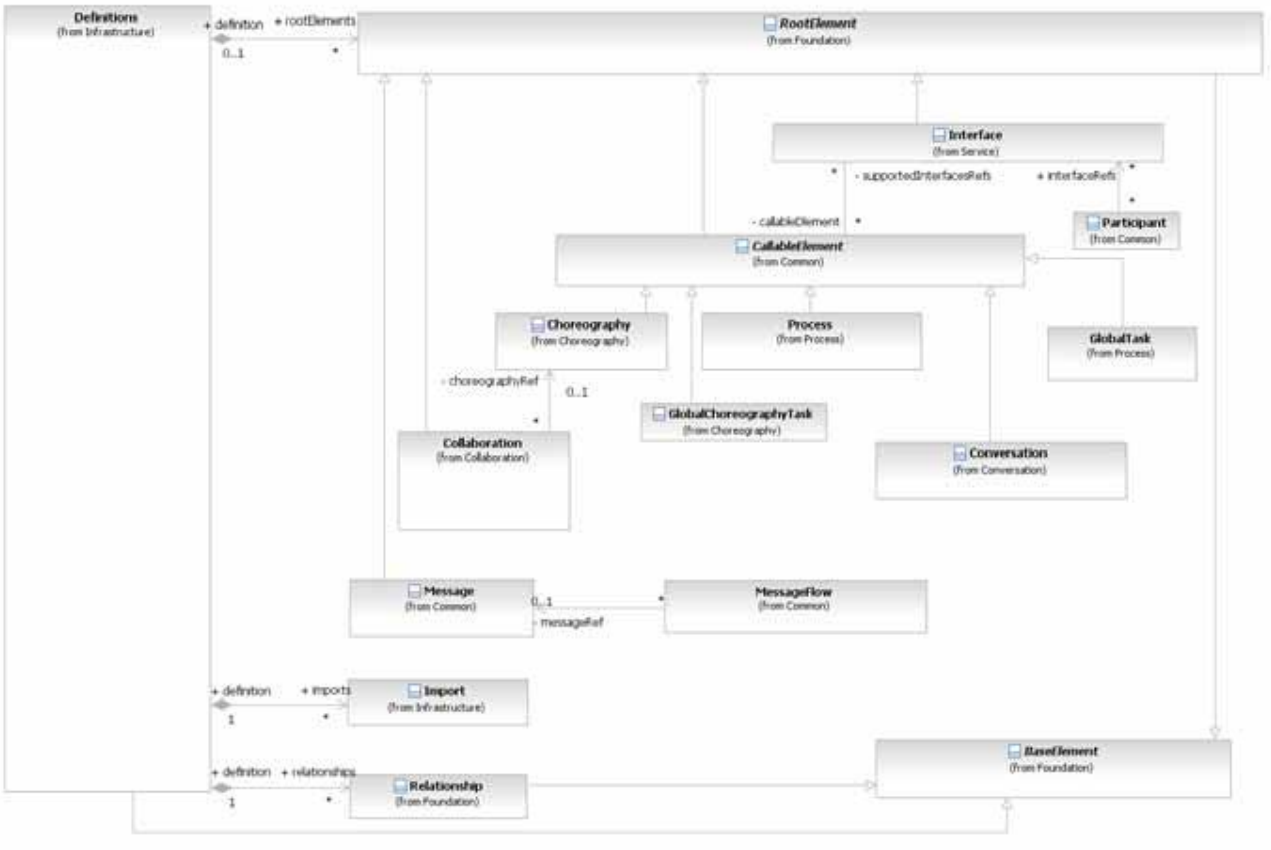
BPMN branduolio elementai (tiek BPMN 2.0, tiek BPMN 1.2) yra aibė elementų, kurios pakanka primityviems verslo procesams modeliuoti. Kita aibė BPMN elementų yra skirta sudėtingoms situacijoms ir imituojamiems verslo procesams modeliuoti. Toks elementų suskirstymas realizuoja BPMN tikslus [18]:

- Lengvas verslo žmonių BPMN notacijos įsisavinimas;
- Imituojamų verslo procesų modeliavimas;
- Teigiama, jog BPMN notacija gali būti tokio sudėtingumo, kokio reikia tam tikrai problemai spręsti.

4.4. BPMN PAMATINIAI ELEMENTAI

Pamatinių elementų paskirtis, kaip jau minėta, yra atlikti tam tikrus konstrukcinius vaidmenis metamodelyje. Šios grupės elementų metamodelis pateikiamas paveiksle žemiau.

Pats pagrindinis BPMN elementas, yra šakninis elementas (angl. Root Element). Iš šio elemento tiesiogiai paveldimi sąsajos (angl. interface), kviečiamasis (angl. callable), bazinis (angl. Based), pranešimo (angl. Message) ir ansamblio arba bendradarbiavimo (angl. Collaboration) elementai [BPMN].



11 pav. BPMN metamodelis: pamatiniai elementai

Šaltinis: [http://bpmnhandbook.com/01_specs/BPMN_20_spec.pdf]

Kaip matome 11 paveiksle, sąsajos elementas yra sąryšis tarp kviečiamojo elemento, kurio specializacijos yra procesas (angl. Process), choreografija (angl. Choreography) globali užduotis (angl. Global Task), pokalbis (angl. Conversation) ir dalyvio (angl. participant). Kaip matysime vėlia, tai viena esminių sąsajų BPMN notacijoje [BPMN].

Keletas konkrečių konceptų, atvaizduotų pamatinių elementų metamodelyje, analizuojami detaliau tolesniuose darbo skyriuose.

4.5. PRIELAIDOS BPMN MODELIO TRANSFORMAVIMUI

Kaip vieną iš konceptualių naujovių BPMN, verta paminėti tris pagrindines procesų kategorijas [18]:

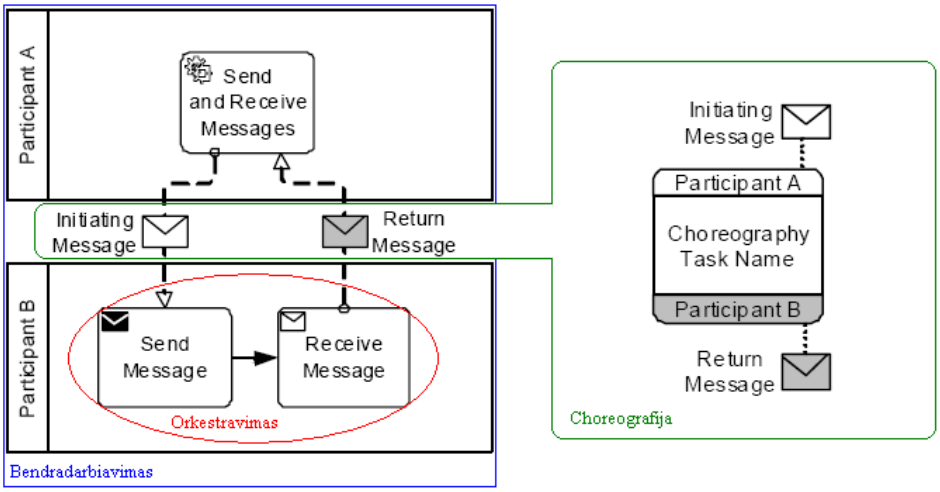
- Orkestravimą (angl. Orchestration);
- Choreografiją (angl. Choreography);
- Bendradarbiavimą (angl. Collaboration).

Orkestravimo modelių paskirtis, atvaizduoti vieną koordinuojamą perspektyvą, t.y. atvaizduoti specifinį organizacijos žvilgsnį į procesą. Šio tipo procesai atvaizduoja atskiro dalyvio (angl. participant) sąsają su aplinka [18].

Choreografija aprašo sąveikų tarp dalyvių seką. Ji egzistuoja šalia pulo arba tarp dviejų pulų. Kiekvienas žingsnis choreografijoje aprašo dviejų ar daugiau dalyvių bendravimą. Choreografija dažniausiai derinama su bendradarbiavimu [BPMN].

Bendradarbiavimas – tai dviejų daugiau pulų, reprezentuojančių dalyvius, sąveika. Pulai gali būti tušti arba atvaizduoti savyje procesą (orkestravimą).

Visos trys procesų kategorijos viename pavyzdyje – bendradarbiavimo diagramoje – pateikiamos paveiksle žemiau.



12 pav. BPMN procesų kategorijos

Šaltinis: [http://bpmnhandbook.com/01_specs/BPMN_20_spec.pdf]

Procesų kategorijos nustatymas dažnai susilaukdavo diskutuojančių pusių konflikto, nes BPMN 1.2 nepateikė aiškaus notacinio šių kategorijų atskyrimo. Tačiau, BPMN 2.0 versija tokį atskyrimą turi [18].

4.6. BPMN PROCESAS

Procesas BPMN aprašo veiklų seką organizacijoje, kuri turi tikslą atlikti tam tikrą darbą. Proceso elementas rotacinio simbolio neturi, o diagramose atvaizduojamas kaip srauto elementų, pvz., veiklų, įvykių, stuomenų objektų ir pan rinkinys. BPMN procesas gali būti bet kokio lygio procesas nuo proceso visos organizacijos mastu iki vieno žmogaus vykdomo proceso. Paties žemiausio lygio (atominiai) procesai gali būti sugrupuoti siekiant bendro tikslo [18].

Procesas yra iškviečiamas elementas, o tai BPMN notacijoje reiškia elementą, kuris gali būti panaudojamas dar kartą. Taigi procesas gali būti iškviečiamas kito proceso viduje. Tokiu būdu procesai dekomponuojami į procesų hierarchiją.

Kito proceso viduje esantis procesas, paprocesis ar užduotis, panaudojama dar kartą, yra veiklos. Veikla specifikuojama kaip darbas proceso viduje. Veikla turi duomenų įeigą, išeigą, resursus ir tam tikrais atvejais inicijuojantį įvykį [14].

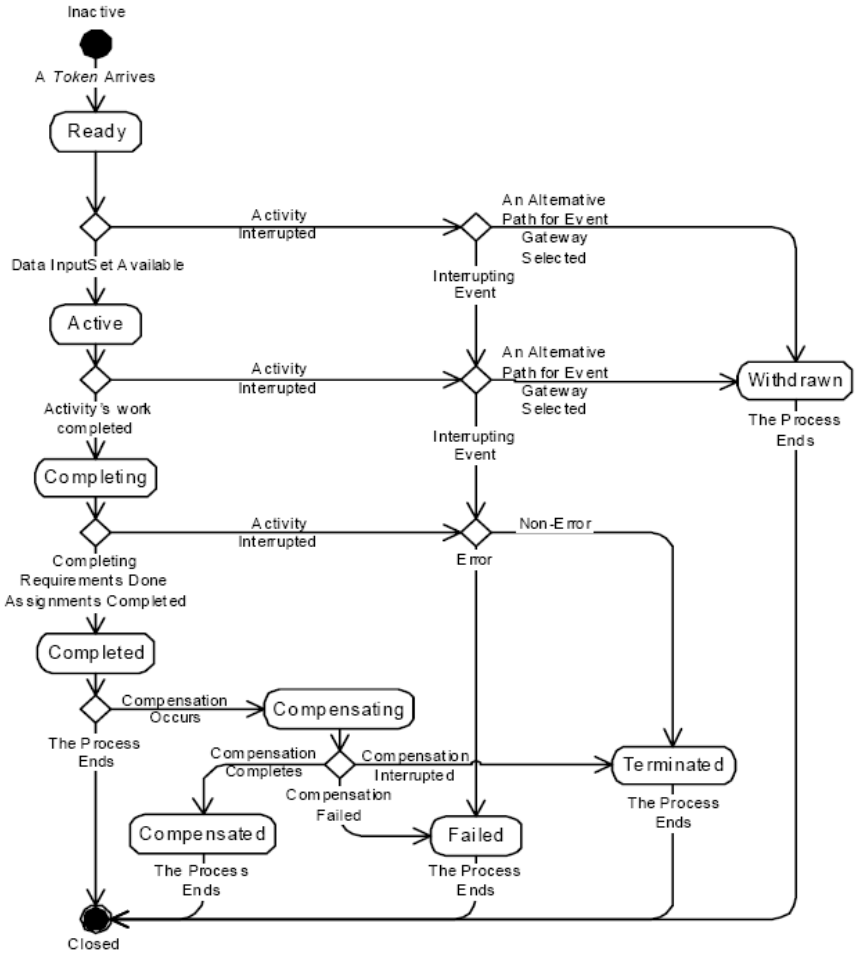
Procesai BPMN 2.0 notacijoje skirstomi į [BPMN]:

- privačius nevykdomus;
- privačius vykdomus;
- atvirus (angl. public).

Vidiniai procesai – tai procesai, kurie vykdomi vienoje organizacijoje. Šio tipo procesai dar vadinami orkestravimo procesais. Priešingai vidiniams, išoriniai procesai skirti atvaizduoti vidinių procesų sąveiką su kitais procesais arba dalyviais per tam tikras sąsajas (angl. interfaces).

Duomenų apsikeitimas tarp procesų vyksta pranešimais. Pranešimai perneša duomenis tarp dviejų ar daugiau procesų ar dalyvių. Turi BPMN terminologijoje yra choreografija.

BPMN veiklos procesas, be jau minėto tipo, gali turėti daugybę kitų atributų. Vienas iš jų – būsena. Procesas tam tikru laiko momentu gali turėti vieną iš apibrėžtų būsenų: neaktyvus, pasiruošęs, užsidaręs savyje, aktyvus, nutrauktas, nepavykęs, besibaigiantis, pasibaigęs, kompensuojantis, kompensuotas, uždarytas. Proceso būsenų diagrama pateikiama žemiau.



13 pav. BPMN proceso būsenų diagrama

Šaltinis: [http://bpmnhandbook.com/01_specs/BPMN_20_spec.pdf]

Tokias pačias būsenas gali įgyti ir veiklos specializacijos.

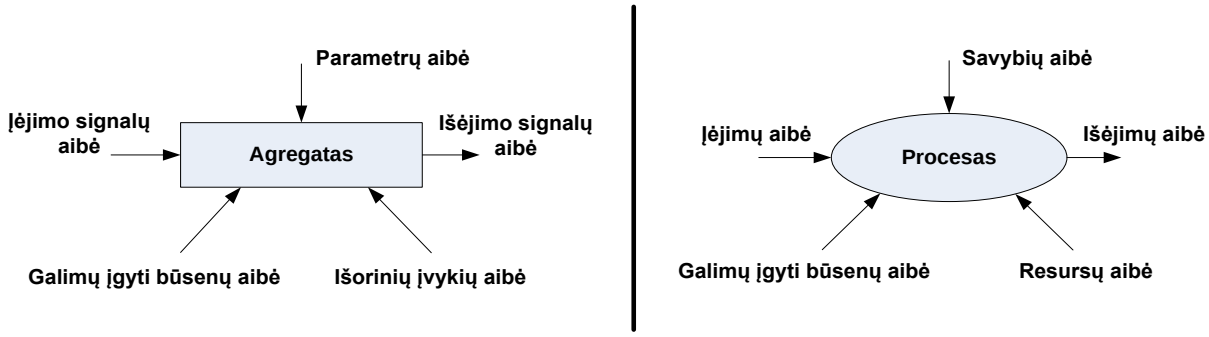
Procesas būna pasirengimo būsenoje, kol yra aktyvuojamas įeigos srauto. Tuomet procesas pereina į aktyvaus proceso būseną, jei įeigoje yra duomenų. Priešingu atveju procesas nutraukiamas. Po aktyvaus procesas pereina į baigiamąją būseną, o po jos į pasibaigusią. Po pabaigos procesas gali būti kompensuojamas, jei įvyksta kompensacijos įvykis ir kompensuotas arba neįvykęs, jei kompensacija nepavyksta [BPMN].

Procesas taip pat gali būti atšaukiamas, nutraukiamas arba neįvykęs dėl tam tikrų įvykių, pvz., 13 pav. klaidos [BPMN].

4.7. BPMN PROCESAS IR PLA AGREGATAS

Procesas ir agregatas turi daug tarpusavio panašumų. Šiame skyrelyje PLA agregatas ir BPMN procesas palyginami konceptualiai.

Agregatas aprašomas kaip tam tikrą funkciją atliekantis sudėtingos mašinos mazgas arba kelios sujungtos mašinos bendram darbui atlikti [19]. Kaip jau buvo minėta, procesas aprašo veiklų seką organizacijoje, kuri turi tikslą atlikti tam tikrą darbą.

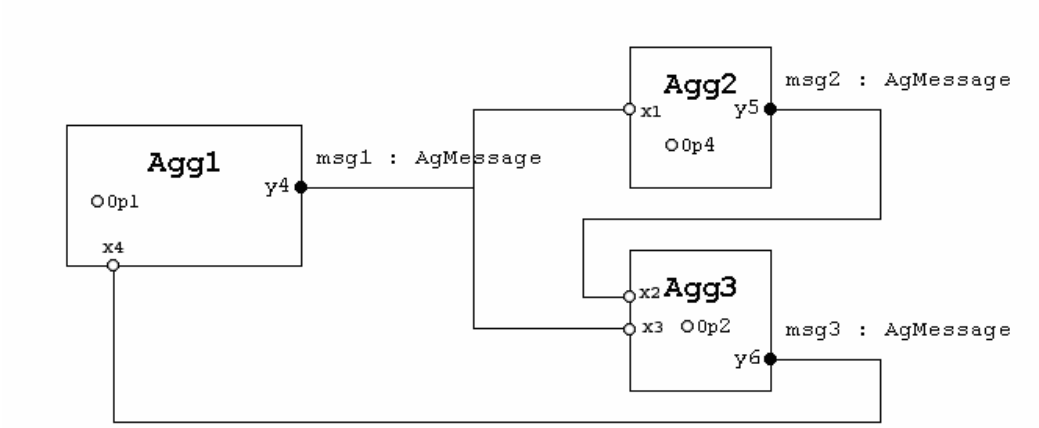


14 pav. Procesas ir agregatas

Šaltinis: [http://bpmnhandbook.com/01_specs/BPMN_20_spec.pdf]

Agregatas ir procesas (14 pav.) turi įėjimus ir išėjimus, galimas įgyti būsenas. Agregatas turi jį apibūdinančių parametrų aibę, tuo tarpu procesas – savybių aibę. Priešingai nei procesas, agregatas neaprašo reikalingų resursų, o procesas išorinių įvykių aibės.

Kitas svarbus transformacijos momentas – ryšiai tarp procesų. Šie ryšiai BPMN vadinami pranešimais. Tarp agregatų esantys ryšiais, nuo vieno agregato išėjimo iki kito agregato įėjimo, vadinami kanalais ir gali būti atvaizduoti agregatų diagramoje [17].



15 pav. Kanalai tarp agregatų

Šaltinis: [http://bpmnhandbook.com/01_specs/BPMN_20_spec.pdf]

15 paveikslėlyje pavaizduota agregatų diagrama. Joje yra aprašyti trys agregatai „Agg1“, „Agg2“, „Agg3“, jų įėjimo ir išėjimo signalai, kanalai, jungiantys agregatus, bei jais perduodami pranešimai. Taip pat pavaizduotas kanalas jungiantis iš karto tris agregatus.

Apibendrinant galima teigti, jog konceptualiai procesas ir agregatas yra artimi vienas kitam. Tai leidžia transformuoti biznio procesus į agregatus tiesiogiai.

4.8. BPMN MODELIO FIZINĖ IŠRAIŠKA

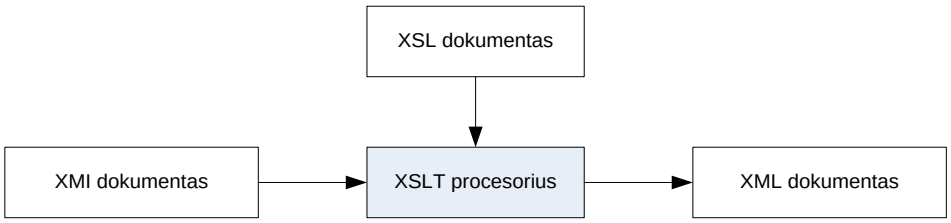
BPMN meta modelis pagrįstas MOF (angl. Meta Object Facility). Šiuo atžvilgiu BPMN meta modelis yra lygiagretus UML meta modeliui. Tai BPMN išskiria iš daugelio kitų OMG standartų, pvz., SysML ar UPDM, kurie yra UML kalbos plėtiniai. BPMN nėra UML kalbos plėtinys, todėl yra nesuderinamas tarp OMG standartų.

MOF meta modelis suteikia galimybę BPMN modelius saugoti XMI formatu. BPMN nuo 2.0 versijos, taip pat pasižymi diagramų apsikeitimo aprašu (DI). DI suteikia galimybę keistis ne tik modeliu, bet ir diagramomis tarp skirtingų gamintojų BPMN įrankių [BPMN].

BPMN 2.0 specifikacija pateikia detalias XSD schemas kiekvienam BPMN elementui. Pavyzdžiui, paprocesio XSD schema pateikiama žemiau:

```
<xsd:element name="subProcess" type="tSubProcess" substitutionGroup="flowElement"/>
<xsd:complexType name="tSubProcess">
    <xsd:complexContent>
        <xsd:extension base="tActivity">
            <xsd:sequence>
                <xsd:element ref="flowElement" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

XMI formatas yra specialus XML dokumentas, kuris naudojant XSLT (angl. Extensible Stylesheet Language Transformations) transformacijų kalbą, gali būti nesunkiai transformuojamas į bet kokį kitą XML fizine išraiška pagrįstą formatą [19].



16 pav. XSLT transformacijų principinė schema

Šaltinis: sudarė autorius

XSLT yra sukurta naudojimui kaip XSL (angl. Extensible Stylesheet Language) kalbos dalis. Paveiksle 16 parodytas bendras XSLT naudojimo atvejis. XSLT procesoriui yra perduodamas XML dokumentas, kuris bus transformuojamas, ir XSL dokumentas, kuris nurodo, kaip transformuotas dokumentas atrodys. Rezultatas – naujas dokumentas (dažniausiai XML, HTML ar tekstinio tipo dokumentas).

4.9. BPMN TRANSFORMACIJA Į PETRI TINKLUS

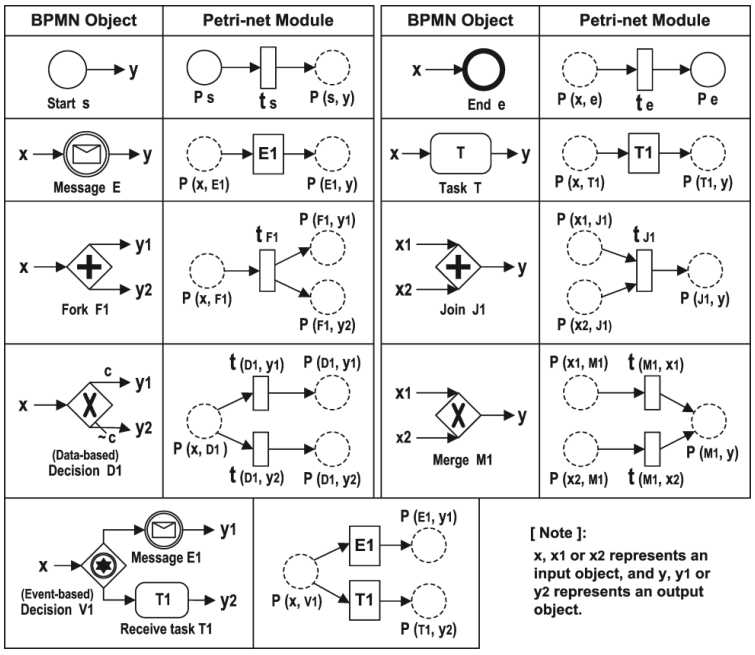
Mokslininkų Remco M. Dijkman, Marlon Dumas ir Chun Ouyang pasiūlyta taisyklingų BPMN modelių transformacija į Petri tinklus apima pažymėtų ir nepažymėtų parėjimų (angl. transitions) panaudojimą. Pažymėti parėjimai taikomi apibūdinti BPMN įvykiams ir užduotims, o nepažymėti (angl. silent) – vidiniams veiksmams, kurie nematomi išoriniams vartotojams [20].

Paveiksle 17 pateikiami BPMN bendrinių elementų atitikmenys Petri tinkluose. Užduotys ir tarpiniai įvykiai atitinka perėjimus su vienu įėjimu ir vienu išėjimu. Perėjimai, paženklinti vardu, reiškia užduoties arba įvykio įvykdymą. Pradžios ir pabaigos įvykiai, priešingai nei tarpiniai, atitinka tylius perėjimus [20].

Visi vartai, išskyrus įvykiu pagrįstus išskiriančius vartus, atitinka smulkius Petri tinklų modulius su tyliais perėjimais, kurie aprašo maršrutizavimo elgseną. Įvykiu pagrįsti vartai atitinka įvykio ir užduočių perėjimus, kur visi perėjimai turi įvykti, kad srauto žymė atsidurtų išėjime [20].

Pozicijos, kurių rėmeliai pažymėti brūkšniais, žymi, kad jų pritaikymas nėra unikalus pateikiamo pavyzdžio atveju. Tam tikra prasme šios žymės artimos sekos srautams BPMN notacijoje. Išimtis – tik įvykiu pagrįsti vartai [20].

Sub-procesų transformavimui į BPMN nenaudojama jokia speciali notacija. Sub-proceso viduje vykstantys įvykiai, užduotys ir kt. transformuojama pagal tas pačias taisykles.



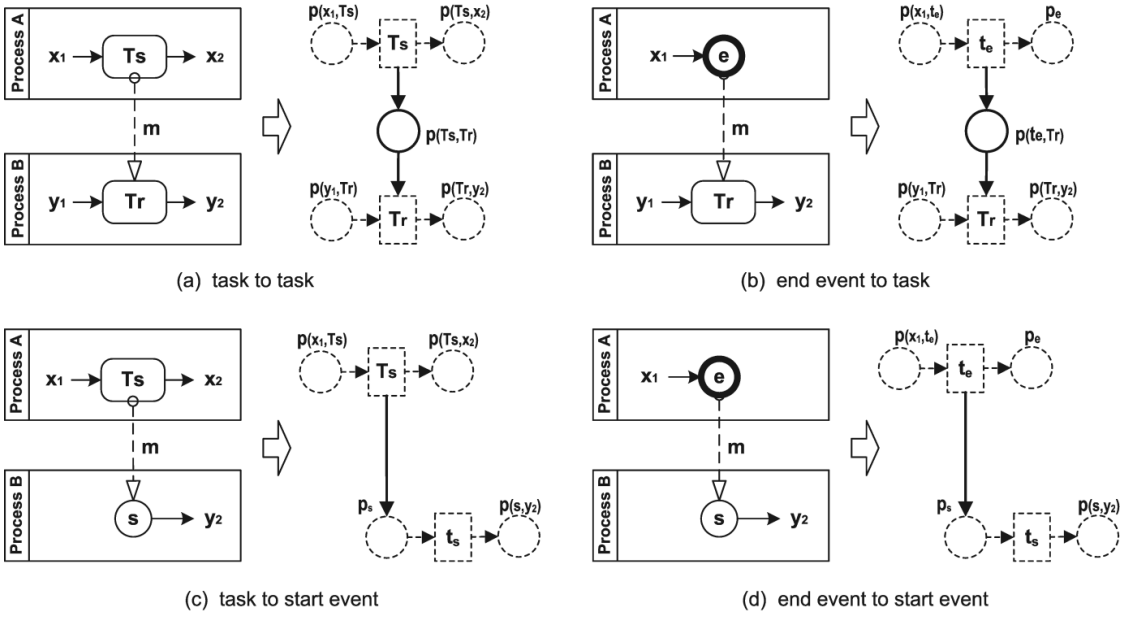
17 pav. BPMN bendrinių elementų atitikmenys Petri tinkluose

Šaltinis: [http://bpmnhandbook.com/01_specs/BPMN_20_spec.pdf]

Pranešimo srautas, rodantis transakcijas tarp dviejų dalyvių BPMN notacijoje, abstrakčiu atveju transformuojamas į poziciją tarp siuntimo ir priėmimo užduočių parėjimų. Atskiras atvejis yra

pranešimas sujungtas su pradžios įvykiu. Tai parodo, kad procesas bus pradėtas, vos tik pranešimas bus gautas. Šiuo atveju siunčianti pozicija yra įvykio tipo [20].

Keturi skirtingi transformavimo atvejai pateikiami paveiksle žemiau. Svarbu tai, kad visais atvejais užduoties perėjimas gali būti keistinas į tarpinio įvykio perėjimą nekeičiant pačios taisyklės.



18 pav. BPMN transformavimo į Petri tinklus atvejai

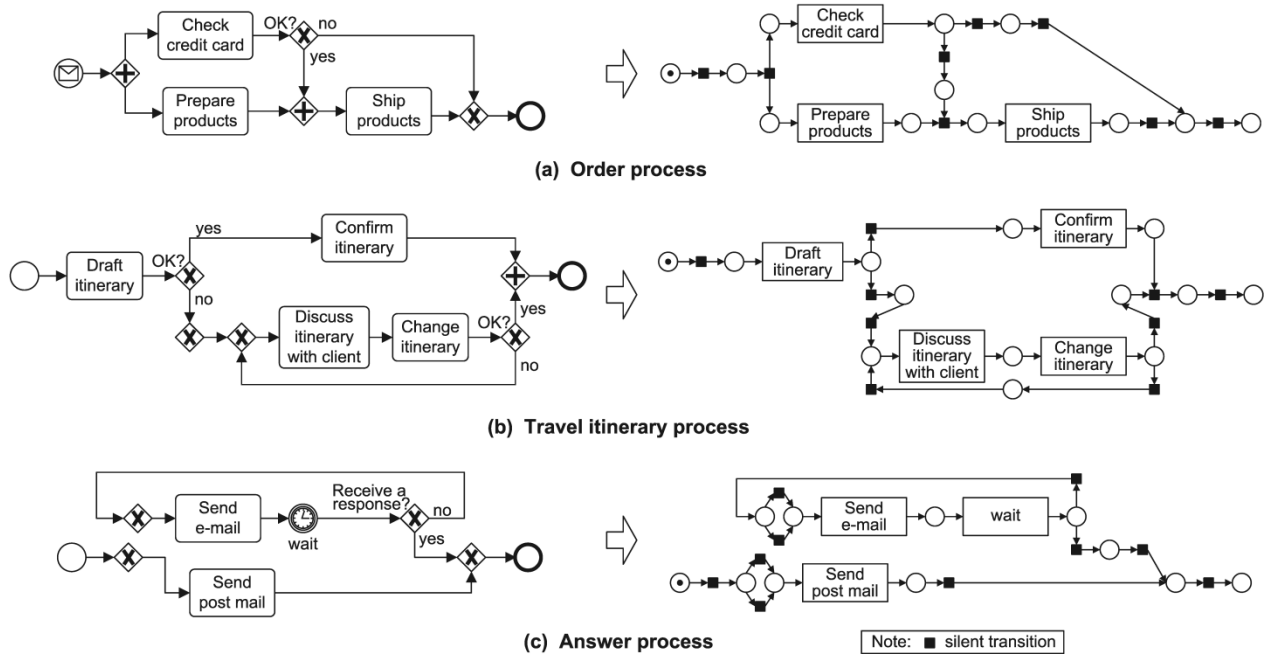
Šaltinis: [http://bpmnhandbook.com/01_specs/BPMN_20_spec.pdf]

Remiantis aukščiau pateiktomis taisyklėmis, transformuojami trys pavyzdžiai. Tai užsakymo procesas (a), kelionės maršruto parinkimo procesas (b) ir atsakymo į elektroniniu paštu siunčiamą pranešimą procesas (c).

Pirmuoju atveju lygiagrečiai patikrinama kreditinė kortelė ir paruošiami produktai. Jei kortelė priimta, produktai išsiunčiami pirkėjui, priešingu atveju procesas baigiamas. Pažymėtina, kad procesas pradedamas nuo pranešimo gavimo.

Antrasis atvejis pradedamas kelionės maršruto parinkimu. Jei maršrutas tenkina, jis patvirtinamas, priešingu atveju maršrutas aptariamas su klientu ir pakeičiamas. Jei pakeistas maršrutas netenkina, ieškoma kito maršruto, kuris tenkintų klientą. Tokį maršrutą radus, procesas baigiamas.

Trečiuoju atveju siunčiamas laiškas paštu ir elektroniniu paštu. Laiškas elektroniniu paštu kartojamas kas tam tikrą laiką, jeigu nesulaukiama atsakymo.



19 pav. BPMN transformacijos į Petri tinklus pavyzdžiai

Šaltinis: [http://bpmnhandbook.com/01_specs/BPMN_20_spec.pdf]

Kaip matyti 19 paveiksle, transformacija iš nesudėtingų BPMN modelių į Petri tinklus nėra itin sudėtinga. Transformuoti procesai lengvai perskaitomi, neprarandama daug informacijos. Paveiksle juodais kvadratais žymimi tylūs perėjimai.

Pasiūlytas transformavimo metodas nepadengia tam tikrų sudėtingų atvejų, kurie susiję su Petri tinklų ribotu panaudojimu. Dėl šios priežasties sukurti YAWL tinklai, išplečiantys Petri tinklus lygmens požymiais.

5. UML IR BPMN TRANSFORMAVIMAS Į PLA

5.1. UML IR BPMN PALYGINIMAS

UML veiklos diagramos ir BPMN pateikia lengvai skaitomą grafinę eigos proceso (angl. workflow process) notaciją. Kai kurie autoriai teigia, kad veiklos diagramos ir BPMN turi daug panašumų [14], todėl OMG pateikia naują iniciatyvą [15] [BMI], kuri apjungia UML veiklos diagramas ir BPMN pagal vieną integruotą metamodelį, nors tiksli transformacija tarp jų nėra paprastas uždavinys. Tyrimas [13] parodo ryšius tarp UML veiklos diagramų ir BPMN notacijos eigos proceso pavyzdžiu.

Siejančių ryšių tarp veiklos diagramų ir BPMN analizė (daugiausia konceptualiame lygmenyje, nors ir pakankamai išsami valdymo šablonų požiūriu) atlikta [16] [RFP]. Tačiau toliau bus pateikiami išplėstiniai ryšiai tik su PLA susijusiomis sąvokomis.

Toliau analizuosime BPMN ir UML veiklos diagramos panašumus procesams aprašyti ir transformavimo į PLA galimybes. Bus naudojamas reikalingas elementų kiekis proceso aprašymui ir parodomos sąsajos tarp BPMN bei UML veiklos diagramos elementų.

Siekiant palyginti BPMN ir UML veiklos diagramas, vienas svarbiausių etapų yra sudaryti transformacijos žemėlapį, kuris apibrėžtų BPMN ir UML veiklos atitikmenis procesų aprašymui, o vėliau juos būtų galima naudoti transformuojant į PLA. Transformacijos žemėlapis pateikiamas 2 lentelėje žemiau.

2 lentelė. UML veiklos ir BPMN konceptų transformacijos žemėlapis
Šaltinis: sudarė autorius remiantis [13], [14]

UML veiklos (angl. Activity) koncertas	BPMN koncertas
Veikla (angl. Acitivity)	Procesas (angl. Process)
Veiksmas (angl. Action)	Paprocesis (angl. Subprocess)
Elgsenos iššaukimo veiksmas (angl. Call Bevavior Action)	Iškviečiama veikla (angl. Call Activity)
Įvykio pagavimo veiksmas (angl. Accept Event Action)	Įeinantis pranešimas (angl. Message)
Signalo išsiuntimo veiksmas (angl. Send Signal Action)	Išeinantis pranešimas (angl. Message)
Įvykio pagavimo veiksmas (angl. Accept Event Action)	Tarpinis pagavimo įvykis (angl. Intermediate Catching Event)
Signalo išsiuntimo veiksmas (angl. Send Signal Action)	Tarpinis išsiuntimo įvykis (angl. Intermediate Throwing Event)
Veiksmas (angl. Action)	Užduotis (angl. Task)
Objektas (angl. Object Node)	Duomenų objetas (angl. Data Object)
Valdymo srautas (angl. Control Flow)	Sekos srautas (angl. Sequance Flow)
Veiklos dalis (angl. Activity Partition)	Pulas (angl. Pool)
Sprendimas, apjungimas (angl. Decision, Merge)	Vartai (angl. Gateway)
Pradžios mazgas (angl. Initial Node)	Pradžios įvykis (angl. Start Event)
Veiklos pabaiga (angl. Activity Final) Srauto pabaiga (angl. Flow Final)	Pabaigos įvykis (angl. End Event)
Veikla, veiksmas	Choreografijos užduotis (angl. Choreography)
Duomenų srautas (angl. Object Flow)	Duomenų asociacija (angl. Data Association)
Rankinės užduoties iššaukimas (angl. Call Manual Task)	Rankinė užduotis (angl. Manual Task)
Serviso užduoties iššaukimas (angl. Call Service Task), Operacijos iššaukimas (angl. Call Operation)	Serviso užduotis (angl. Service Task)
Vartotojo užduoties iššaukimas (angl. CallUserTask)	Vartotojo užduotis (angl. User Task)
Įėjimo/Išėjimo taškas (angl. InputPin/OutputPin)	Pranešimas (angl. Message)
Pertraukiamas veiklos regionas (angl. Interruptible Activity Region) ir išimties tvarkiklis (angl. Exception Handler)	Įterptinis paprocesis (angl. Embedded Subprocess) ir pranešimo pertraukimo įvykis (angl. Message Intermed Event)

5.2. UML VEIKLOS MODELIO TRANSFORMAVIMAS Į PLA

Siekiant atlikti transformaciją iš UML veiklos modelio į PLA, vienas svarbiausių etapų yra sudaryti transformacijos žemėlapį, kuris apibrėžtų UML ir PLA atitikmenis. Transformacijos žemėlapis pateikiamas 3 lentelėje žemiau.

3 lentelė. UML ir PLA transformacijos žemėlapis
Šaltinis: sudarė autorius remiantis [13]

UML veiklos (angl. Activity) konceptas	PLA atitikmuo
Veikla (angl. Acitivity)	Agregatas
Veiksmas (angl. Action)	Vidinis įvykis
Elgsenos iššaukimo veiksmas (angl. Call Bevavior Action)	Išorinis įvykis
Signalas (angl. Signal)	Įėjimo signalas
Signalas (angl. Signalas)	Išėjimo signalas
Įvykio pagavimo veiksmas (angl. Accept Event Action)	Išorinis įvykis
Signalo išsiuntimo veiksmas (angl. Send Signal Action)	Išorinis įvykis
Veiksmas (angl. Action)	Vidinis įvykis
Objektas (angl. Object Node)	Atitikmens nėra
Valdymo srautas (angl. Control Flow)	Atitikmens nėra
Veiklos dalis (angl. Activity Partition)	Agregatas tik tuo atveju, jeigu orkestravimas pule yra paslėptas „juoda dėžė“
Sprendimas, apjungimas (angl. Decision, Merge)	Atitikmens nėra
Pradžios mazgas (angl. Initial Node)	Išorinis įvykis (jei apibrėžtas), Neapibrėžtas (angl. None) įvykis atitikmens neturi
Veiklos pabaiga (angl. Activity Final), Srauto pabaiga (angl. Flow Final)	Vidinis įvykis (jei apibrėžtas), Neapibrėžtas (angl. None) įvykis atitikmens neturi
Veikla, veiksmas	Vidinis įvykis tuo atveju, jei choreografija traktuojama kaip procesas
Duomenų srautas (angl. Object Flow)	Atitikmens nėra
Rankinės užduoties iššaukimas (angl. Call Manual Task)	Vidinis įvykis
Serviso užduoties iššaukimas (angl. Call Service Task), Operacijos iššaukimas (angl. Call Operation)	Vidinis įvykis
Vartotojo užduoties iššaukimas (angl. Call User Task)	Vidinis įvykis
Įėjimo/Išėjimo taškas (angl. InputPin/OutputPin)	Įėjimo signalas, Išėjimo signalas
Pertraukiamas veiklos regionas (angl. Interruptible Activity Region) ir išimties tvarkiklis (angl. Exception Handler)	Atitikmens nėra

5.3. BPMN MODELIO TRANSFORMAVIMAS Į PLA

Siekiant atlikti transformaciją iš BPMN į PLA, vienas svarbiausių etapų yra sudaryti transformacijos žemėlapį, kuris apibrėžtų BPMN ir PLA atitikmenis. Sudarant transformacijų žemėlapį, itin svarbu atsižvelgti į galimus BPMN panaudojimo atvejus, t.y. orkestraciją, choreografiją ir bendradarbiavimą.

Orkestracijos atveju aprašomas vienas procesas. Jis gali būti transformuojamas į PLA agregatą, o jo vidinės užduotys ir kiti įvykiai į PLA įvykius. Taigi orkestracijos atveju turime vieną agregatą.

Bendradarbiavimo atveju galimi keli tarpusavyje bendradarbiaujantys procesai. Jei bendradarbiaujantys procesai yra orkestracijos, transformavimas nesikeičia nuo orkestracijos atvejo, tačiau, jei bent vienas procesas yra taip vadinama „juodoji dėžė“, pulas, kuris ją reprezentuoja, turėtų būti transformuojamas į agregatą.

Choreografijos transformavimas gali būti dvejopas. Kadangi choreografija tiesiogiai neaprašo orkestracijos, o tik sąveiką tarp dviejų dalyvių, choreografiją tarp dviejų dalyvių galima traktuoti kaip atskirą procesą, o drauge ir PLA agregatą. Kitas galimas pasirinkimas – choreografijos apskritai netransformuoti, tačiau tokiu atveju prarandama dalis BPMN specifikuotos informacijos.

Transformacijos žemėlapis pateikiamas lentelėje žemiau.

4 lentelė. BPMN ir PLA transformacijos žemėlapis
Šaltinis: sudaryta autoriaus remiantis [19]

BPMN koncertas	PLA atitikmuo
Procesas (angl. Process)	Agregatas
Paprocesis (angl. Subprocess)	Vidinis įvykis
Pulas (angl. Pool)	Agregatas (angl. Aggregate). Transformuojama tik tuo atveju, jei pulas yra uždaras, t.y. reprezentuoja orkestraciją, kurioje vykstantys įvykiai ir užduotys yra nežinomi. Tai agregatas su viena būsena, įėjimo ir išėjimo signalais.
Iškviečiama veikla (angl. Call Activity)	Vidinis įvykis
Įeinantis pranešimas (angl. Message)	Įėjimo signalas
Išeinantis pranešimas (angl. Message)	Išėjimo signalas
Tarpinis išsiuntimo įvykis (angl. Intermediate Throwing Event)	Išorinis įvykis
Užduotis (angl. Task)	Vidinis įvykis
Duomenų objektas (angl. Data Object)	Atitikmens nėra
Sekos srautas (angl. Sequance Flow) (nuo vienos užduoties/paprocesio iki kitos). Jeigu yra keletas sekos srautų pertrauktų tam tikrais įvykiais, jie visi traktuojami kaip vienas transformuojant.	Valdymo seka
Vartai (angl. Gateway)	Sąlyga perėjimo operatoriuje
Pradžios įvykis (angl. Start Event)	Atitikmens nėra
Kraštinis įvykis (angl. Boundary Event)	Sąlyga perėjimo operatoriuje
Pabaigos įvykis (angl. End Event)	Vidinis įvykis (jei apibrėžtas). Neapibrėžtas

BPMN koncertas	PLA atitikmuo
	(angl. None) įvykis atitikmens neturi.
Duomenų asociacija (angl. Data Association)	Atitikmens nėra
Procesas (angl. Process)	Agregatas

Dar vienas galimas atvejis – bendradarbiavimas be pranešimų. Tokiu atveju procesai bendrauja per įvykius. Vieno proceso viduje įvykęs įvykis įtakoja kitą procesą, jį inicijuoja, baigia ir pan. Tokiu atveju rezultatas PLA – du agregatai, nesusieti kanalais tarpusavyje, abu savo išorinių įvykių aibėse turintys įvykius, kurių yra įtakojami. Jei tai pranešimo siuntimo įvykis, išimtinai galima jį traktuoti kaip pranešimo srautą, o drauge ir sujungti agregatus kanalu.

6. NACIONALINĖ VERSLO REGISTRAVIMO SISTEMA

6.1. NACIONALINĖS VERSLO REGISTRAVIMO SISTEMOS APRAŠYMAS

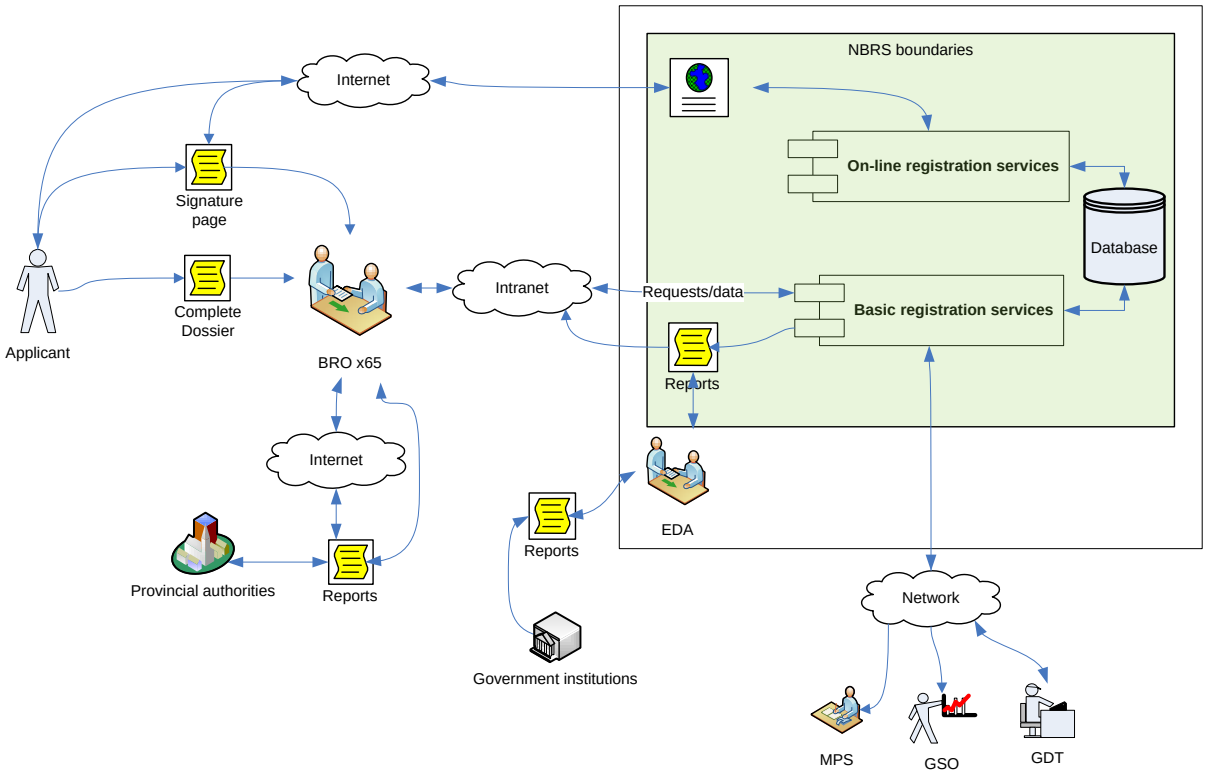
NBRS – tai nacionalinė verslo registravimo sistema, kuriama Vietname, norint palengvinti ir struktūrizuoti įmonių užregistravimo, pakeitimo, sustabdymo ar išregistravimo mechanizmus. NBRS suskirstyta į dvi posistemes: tai yra pagrindinė registracija (angl. Basic registration), kuri yra pasiekiama iš 65 Vietname esančių skyrių, ir internetinė registracija (angl. Online registration), kuri pasiekiama kiekvienam vartotojui, kuris naudodamasis internetu prisiregistruoja prie NBRS.

NBRS padengia šį funkcionalumą:

- Naujų objektų registracija ir valdymas:
  - Naujų įmonės tipų registracija;
  - Filialų, atstovybių ir verslo vietų registracija.
- Pakeitimų valdymas ir registracija:
  - Įmonės duomenų keitimo registracija;
  - Įmonės sustabdymo registracija;
  - Įmonės atstatymo iš sustabdymo registracija;
  - Įmonės tipo pakeitimo registracija;
  - Įmonės ar kelių įmonių sujungimo registracija;
  - Įmonės išregistravimo registracija.

6.2. NACIONALINĖS VERSLO REGISTRAVIMO SISTEMOS APŽVALGA

Eskizinė projekto diagrama trumpai apžvelgia NBRS architektūra, funkcionalumo modulius, vartotojus sistemoje ir duomenų bazę. Tolesniuose šio dokumento skyriuose bus aprašyta kiekviena dalis išsamiau.



20 pav. Eskizinė projekto diagrama
Šaltinis: sudarė autorius

Pareiškėjas atneša verslo registravimo dokumentus į vieną iš skyrių. Visi 65 skyriai turi priėjimą prie sistemos per internetą, kuri yra centinėje dalyje Hanojuje, EDA padalinyje. Klerkas, gavęs dokumentus, įregistruoja juos drauge ir prašymą įregistruoti įmonę žurnale.

Taip pat žmogus gali užpildyti duomenis per internetu prieinamą (angl. on-line) registraciją, tai yra per NBRS posistemę. Tokiu būdu sumažinamas apkrautumas pačių klerkų, kurie turėtų užpildyti visą informaciją apie įmonės įregistravimą. Tuo pasirūpina pats žmogus. Jeigu norima užregistruoti įmonę iš internetu prieinamos (angl. on-line) sistemos reikia pereiti patvirtinimo mechanizmus. Taip užtikrinama, kad pateikti duomenys bus labiau korektiški ir nebus atmesti skyriaus registratoriaus, kuriam bus pateikta paraiška užregistruoti.

Visa informacija apie paraišką, tokia kaip įmonės vardas, akcininkai, partneriai ir t.t., užpildomi NBRS. Šie žingsniai gali būti praleisti, jei paraiška pildoma pasinaudojant internetu prieinama (angl. on-line) registracijos paslauga. Be to, priedai, tai yra nuskenuoti failai, prisegami prie NBRS ir nereikia jų skenuoti pačiam skyriaus darbuotojui. Teisinius ir loginius patikrinimus sistema vykdo automatiškai. Klerkas užpildo duomenis ir siunčia juos dėl sprendimo registratoriui.

Registratorius patvirtina registraciją, paraiškos atitikimą nustatytiems teisės aktams. Vyriausybinės įstaigos užprašomas įmonės kodas GDT ir jai priskiriamas. Verslo registracijos

liudijimas yra generuojamas NBRS. Bendradarbiaujančioms institucijoms GDT, MPS ir GSO yra pranešama apie naujos įmonės įregistravimą ar informacijos apie įmonę pasikeitimą.

Registratorius atmeta registraciją, jei pateikta informacija dokumentuose neatitinka pateiktų teisės aktų, kurie galioja norimam įregistruoti įmonės tipui. Atmetimo atveju nurodoma sprendimo atmesti priežastis, kuri yra įrašomi į NBRS. Atmetimo dokumentas gali būti atspausdinamas.

Skyriaus vartotojai turi priėjimą prie ataskaitų. Jų pagalba galima sužinoti apie naujas registracijas, pasikeitimus ir kitą informaciją, kuri įvyko sistemoje. Tai gana patogios ir daug resursų nereikalaujančios užklausos, kurios gana išsamiai parodo NBRS pasikeitimus skyriaus ribose.

7. NACIONALINĖS VERSLO REGISTRAVIMO SISTEMOS TRANSFORMACIJA

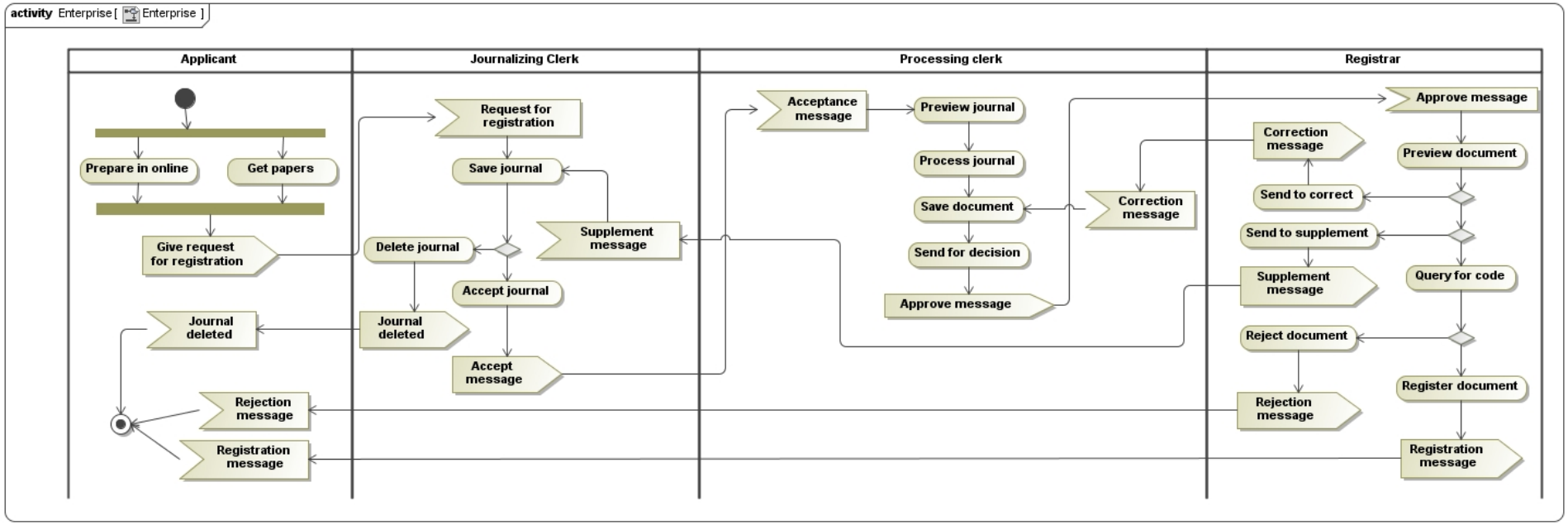
7.1. NACIONALINĖS VERSLO REGISTRAVIMO SISTEMOS UML TRANSFORMAVIMAS Į PLA

Transformacijos pavyzdžiu imamas NBRS paraiškos procesas, kuris apibūdina įmonės įregistravimo kelią Vietname.

Vartotojas inicijuoja procesą surinkdamas duomenis apie norimą įkurti įmonę. Surinkimo būdai yra du. Tai yra užpildant internetiniame puslapyje duomenis ir sugeneravus pradinį dokumentą apie duomenų užpildymą ir jį atnešus į skyrių galima pradėti įmonės užregistravimą. Kitas būdas surinkti visus popierinius dokumentus, juos atgabenti į skyrių ir juos pateikti skyriaus darbuotojui, kuris dirba vieno langelio principu. Vartotojo atėjimą užregistruoja žurnalizuojantis klerkas ir užveda žurnalą, kitaip vadinamą paraišką. Žurnale išsaugoma pradinė informacija apie norimą atlikti veiksmą, kokią operaciją norima atlikti (nauja registracija, pakeitimas, išregistravimas), taip pat pradinė įmonės informacija: įmonės pavadinimas, tipas ir kas pateikė duomenis, tai yra koks pateikusio duomenis vartotojo vardas, pavardė, adresas ir kontaktinė informacija. Priėmus žurnalą jis patvirtinamas ir sugeneruojamas dokumentas, kuris įduodamas vartotojui, kaip faktas, kad jis užvedė paraišką įmonės įregistravimui. Paraiška patvirtinama ir keliauja pas vykdantįjį klerką, kuris peržiūrėjęs žurnalo duomenis pradeda pildyti dokumentą. Užpildęs paraiškoje visus duomenis, reikalingus įmonės įregistravimui, vykdantysis klerkas siunčia paraišką registratoriui. Kartu su paraiška vykdantysis klerkas teikia rekomendaciją atmesti arba patvirtinti paraiškos duomenis. Registratorius, gavęs paraiškos duomenis, juos peržiūri, vyriausybinės organizacijos užklausia įmonės kodo. Gavus įmonės kodą, paraišką įregistruoja arba atmeta. Registratorius taip pat gali siųsti paraišką pataisyti vykdomajam klerkui arba siųsti papildyti žurnalizavusiam klerkui. Vykdantysis klerkas, gavęs paraišką taisyti, perskaito komentarus, kuriuos užpildė registratorius nurodydamas, kodėl iniciavo duomenų

pakeitimo procedūrą. Pakoregavęs duomenis vykdantysis klerkas siunčia atgal paraišką registratoriui vėl nurodęs savo sprendimą dėl duomenų įregistravimo arba išregistravimo. Jei registratorius siunčia papildyti, paraiška pakliūva pas žurnalizuojantį klerką. Žurnalizuojantis klerkas užklausia trūkstamų duomenų vartotoją, kuris užvedė paraišką įmonei užregistruoti. Duomenų pateikimo rekomenduojamas laikas – 15 dienų, po kurių žurnalizuojantis klerkas, niekaip nesusisiekęs su vartotoju, gali atmesti papildymo veiksmą ir sprendimą palikti registratoriui.

Kaip matome, sistemoje yra trys vartotojai: tai yra žurnalizuojantis klerkas, vykdantysis klerkas ir registratorius. Kiekvienas atlieka savo užduotis ir negali vykdyti kito vartotojo užduočių, tai yra vykdantysis klerkas negali priimti sprendimo dėl įregistravimo ar atmetimo, tai atlieka tik registratorius. Įregistravus yra sukuriamas sertifikatas, kuris atiduodamas vartotojui. Vartotojas informuojamas apie įmonės įregistravimą.



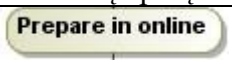
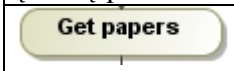
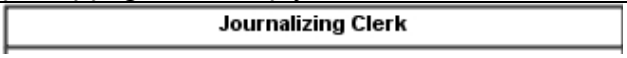
21 pav. NBRS paraiškos įregistravimo procesas

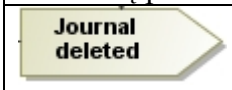
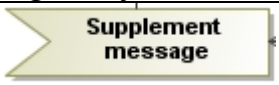
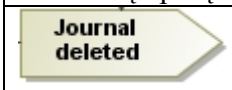
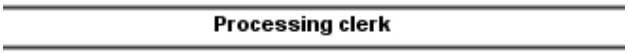
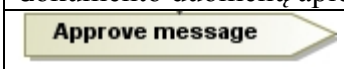
Šaltinis: sudarė autorius

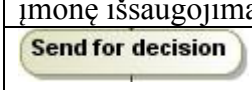
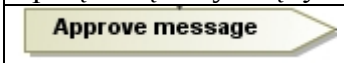
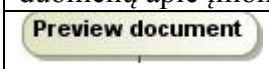
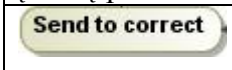
Remiantis sudarytu transformacijos iš UML veiklos į PLA žemėlapiu, sudaryta konkretaus pavyzdžio transformacijos atitikmenų lentelė.

5 lentelė. UML ir PLA transformacijos atitikmenys

Šaltinis: sudarė autorius

UML elementas	PLA atitikmuo
 Dalyvis pareiškėjas. Pavyzdyje pareiškėjas atvaizduotas Veiklos dalyje (angl. Activity Partition). Pareiškėjas užpildo duomenis ir juos atneša į skyrių.	Pareiškėjas atitikmens PLA neturi, tačiau proceso, kurį vykdo pareiškėjas atitikmuo yra agregatas *„Pareiškėjas“*.
 Pranešimas apie informacijos siuntimą į skyrių (BRO)	Išėjimo signalas y_1
 Pranešimas apie žurnalo duomenų ištrynimą	Įėjimo signalas x_1
 Pranešimas apie dokumento duomenų apie įmonę atmetimas	Įėjimo signalas x_2
 Pranešimas apie dokumento duomenų apie įmonę įregistravimą	Įėjimo signalas x_3
 Dokumento duomenų apie įmonę paruošimas NBRS posistemės metodu	Vidinis Įvykis e''_1
 Dokumentų duomenų apie įmonę surinkimas ne per NBRS posistemę	Vidinis Įvykis e''_2
 Siuntimo į skyrių įvykis	Išorinis Įvykis e'_1
 Žurnalo duomenų ištrynimo įvykis	Išorinis Įvykis e'_2
 Dokumento duomenų apie įmonę atmetimo įvykis	Išorinis Įvykis e'_3
 Dokumento duomenų apie įmonę įregistravimo įvykis	Išorinis Įvykis e'_4
 Dalyvis klerkas. Pavyzdyje žurnalizuojantis klerkas atvaizduotas Veiklos dalyje (angl. Activity Partition). Žurnalizuojantis klerkas vykdo dokumentų įmonės įregistravimui pateikimo procesą.	Žurnalizuojantis klerkas atitikmens PLA neturi, tačiau proceso, kurį vykdo žurnalizuojantis klerkas atitikmuo yra agregatas *„Žurnalizavimas*“.

UML elementas	PLA atitikmuo
 Pranešimas apie žurnalo duomenų priėmimą	Išėjimo signalas y_1
 Pranešimas apie žurnalo duomenų ištrynimą	Išėjimo signalas y_2
 Pranešimas apie prašymo registracijai užklausimą	Įėjimo signalas x_1
 Pranešimas apie prašymą dokumento duomenų apie įmonę papildymui	Įėjimo signalas x_2
 Žurnalo duomenų išsaugojimas	Vidinis Įvykis e''_1
 Žurnalo duomenų ištrynimas	Vidinis Įvykis e''_2
 Žurnalo duomenų priėmimas	Vidinis Įvykis e''_3
 Žurnalo duomenų priėmimo įvykis	Išorinis Įvykis e'_1
 Prašymo dokumento duomenų apie įmonę papildymo įvykis	Išorinis Įvykis e'_2
 Prašymo dokumento duomenų apie įmonę registracijai įvykis	Išorinis Įvykis e'_3
 Žurnalo duomenų ištrynimo įvykis	Išorinis Įvykis e'_4
 Dalyvis vykdantysis klerkas. Pavyzdyje vykdantysis klerkas atvaizduotas Veiklos dalyje (angl. Activity Partition). Vykdantysis klerkas vykdo dokumento peržiūrą ir duomenų pildymą, bei suteikia rekomendaciją registratoriui įmonės duomenų įregistravimui arba atmetimui.	Vykdantysis klerkas atitikmens PLA neturi, tačiau proceso, kurį vykdo dalyvis, atitikmuo yra agregatas *„Vykdymas“*.
 Pranešimas apie žurnalo duomenų priėmimą	Įėjimo signalas x_1
 Pranešimas apie prašymą dokumento duomenų apie įmonę taisymui	Įėjimo signalas x_2
 Pranešimas apie dokumento duomenų apie įmonę patvirtinimą	Išėjimo signalas y_1

UML elementas	PLA atitikmuo
 Žurnalo duomenų peržiūra	Vidinis Įvykis e''_1
 Užvedimas dokumento įmonės duomenims pildyti	Vidinis Įvykis e''_2
 Dokumento duomenų apie įmonę išsaugojimas	Vidinis Įvykis e''_3
 Dokumento duomenų apie įmonę siuntimas registratoriui patvirtinimui arba atmetimui	Vidinis Įvykis e''_4
 Žurnalo duomenų priėmimo įvykis	Išorinis Įvykis e'_1
 Prašymo dokumento duomenų apie įmonę taisymo įvykis	Išorinis Įvykis e'_2
 Dokumento duomenų apie įmonę patvirtinimo įvykis	Išorinis Įvykis e'_3
 Dalyvis Registratorius. Pavyzdyje registratorius atvaizduotas Veiklos dalyje (angl. Activity Partition). Registratorius vykdo dokumento įregistravimą ar atmetimą.	Registratorius atitikmens PLA neturi, tačiau proceso, kurį vykdo dalyvis, atitikmuo yra agregatas *„Registravimas“*.
 Pranešimas apie dokumento duomenų apie įmonę patvirtinimą	Įėjimo signalas x_1
 Pranešimas apie prašymą dokumento duomenų apie įmonę taisymui	Išėjimo signalas y_1
 Pranešimas apie prašymą dokumento duomenų apie įmonę papildymui	Išėjimo signalas y_2
 Pranešimas apie dokumento duomenų apie įmonę atmetimą	Išėjimo signalas y_3
 Pranešimas apie dokumento duomenų apie įmonę įregistravimą	Išėjimo signalas y_4
 Dokumento duomenų apie įmonę peržiūra	Vidinis Įvykis e''_1
 Siųsti koreguoti dokumento duomenis apie įmonę	Vidinis Įvykis e''_2
 Siųsti papildyti dokumento	Vidinis Įvykis e''_3

UML elementas	PLA atitikmuo
duomenis apie įmonę	
 Užklausti įmonės kodo dokumento duomenims apie įmonę	Vidinis Įvykis e''_4
 Atmesti dokumento duomenis apie įmonę	Vidinis Įvykis e''_5
 Įregistruoti dokumento duomenis apie įmonę	Vidinis Įvykis e''_6
 Pranešimo apie dokumento duomenų apie įmonę patvirtinimo įvykis	Išorinis Įvykis e'_1
 Pranešimo apie prašymą dokumento duomenų apie įmonę taisymui įvykis	Išorinis Įvykis e'_2
 Pranešimo apie prašymą dokumento duomenų apie įmonę papildymui įvykis	Išorinis Įvykis e'_3
 Pranešimo apie dokumento duomenų apie įmonę atmetimo įvykis	Išorinis Įvykis e'_4
 Pranešimo apie dokumento duomenų apie įmonę įregistravimo įvykis	Išorinis Įvykis e'_5

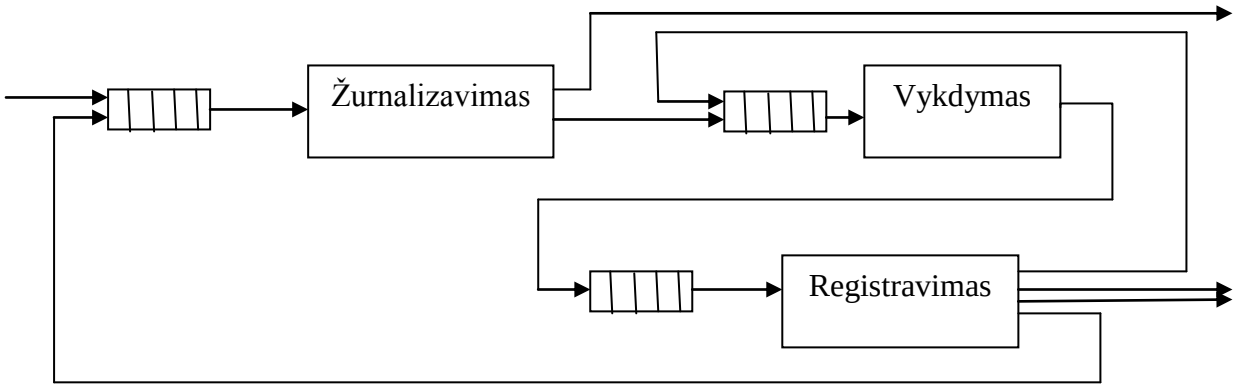
Taigi apibendrinę, gauname keturis agregatus: *„Pareiškėjas“*, *„Žurnalizavimas“*, *„Vykdymas“* ir *„Registravimas“*, kuriems žemiau sukursime konceptualų modelį. Kaip matome transformacija lengvai padėjo iš vieno modelio pereiti prie kito modelio, kurio dėka paskaičiuosime sistemos funkcines charakteristikas ir atliksime jų analizę.

7.2. NACIONALINĖS VERSLO REGISTRAVIMO SISTEMOS KONCEPTUALUSIS MODELIS

Aptarnavimo sistemą NBRS sudaro keturios agregatinės specifikacijos. Agregatinės specifikacijos struktūrinė schema parodyta paveiksle žemiau. Visos keturios agregatinės specifikacijos yra vienkanalės. Paraiška, aptarnauta žurnalizavimo agregate, pereina į antrąjį vykdymo agregatą. Paraiška, aptarnauta vykdymo agregate, pereina į registravimo agregatą. Paraiška, esanti registravimo agregate, gali pereiti į žurnalizavimo agregatą arba į vykdymo agregatą pagal kai kurias sąlygas. Tai yra, jei paraiška įregistruoti dokumentą siunčiama papildyti, tai ji pereina į pirmąjį žurnalizavimo

agregatą arba jei paraiška įregistruoti dokumentą siunčiama pataisyti, tai ji pereina į vykdomąjį agregatą.

Paraiškų aptarnavimo strategija visuose agregatuose yra vienoda ir yra atsitiktinis dydis, tai yra nelabai svarbu, kuri paraiška užvedama pirmoji, nors prioritetas suteikiama FIFO strategijai.

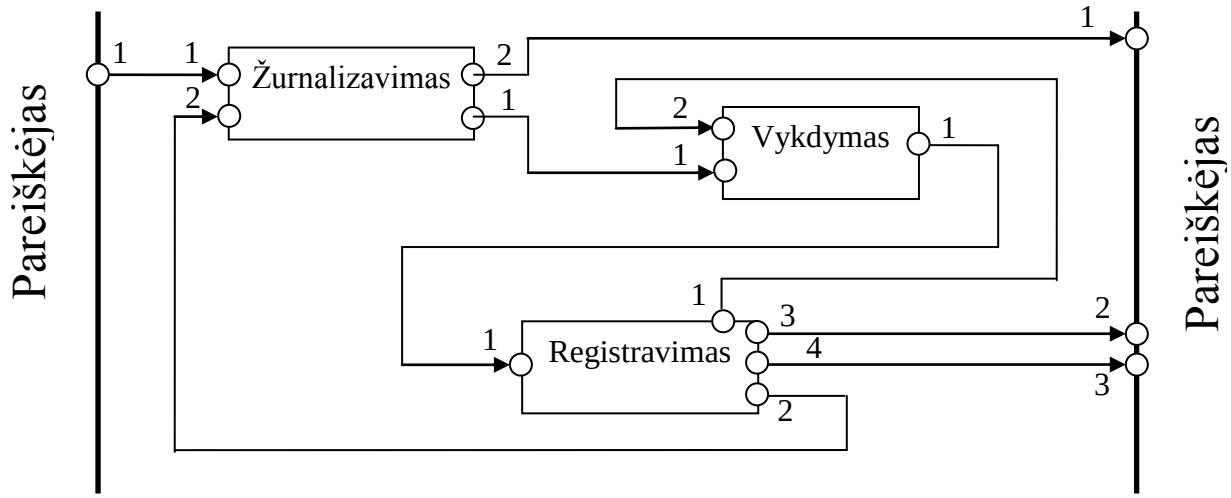


22 pav. NBRS konceptualus modelis

Šaltinis: sudarė autorius

Sudarytas modelis turi leisti įvertinti šias charakteristikas:

1. Vidutinį paraiškos apdorojimo sistemoje laiką;
2. Kiekvieno agregato vidutinį apdorojimo laiką;
3. Sistemos apkrautumo koeficientą;
4. Kiekvieno agregato apkrautumo koeficientą;
5. Sistemos naudingumo koeficientą;
6. Kiekvieno agregato naudingumo koeficientą;
7. Vidutinį sistemoje esančių paraiškų skaičių;
8. Kiekvieno agregato vidutinį esančių paraiškų skaičių.



23 pav. Agregatų sujungimo schema

Šaltinis: sudarė autorius

Agregato „*Pareiškėjas*“ specifikacija

1. Įėjimo signalų aibė $X = \{x_1, x_2, x_3\}$,

   kur x_1 – pranešimas apie paraiškos žurnalo ištrynimą;

   x_2 – pranešimas apie paraiškos atsisakymą įregistruoti;

   x_3 – pranešimas apie paraiškos dokumentui įregistruoti įregistravimą.

2. Išėjimo signalų aibė $Y = \{y_1\}$,

   kur y_1 – Pareiškėjo užklausa į skyrių.

3. Išorinių įvykių aibė $E' = \{ e'_1, e'_2, e'_3, e'_4 \}$,

   kur e'_1 – išsiųsta užklausa;

   e'_2 – atėjo signalas x_1;

   e'_3 – atėjo signalas x_2;

   e'_4 – atėjo signalas x_3.

4. Vidinių įvykių aibė $E'' = \{ e''_1, e''_2 \}$,

   kur e''_1 – paruošimo NBRS posistemėje pabaiga;

   e''_2 – paruošimo be NBRS pabaiga.

5. Valdymo sekos

   $e''_1 \rightarrow \{\xi_{1j}\}$, j=1... ∞, kur ξ_{1j} – paruošimo NBRS posistemėje trukmė;

   $e''_2 \rightarrow \{\xi_{2j}\}$, j=1... ∞, kur ξ_{2j} – paruošimo be NBRS trukmė;

6. Diskrečioji būsenos dedamoji

   $v(t_m) = \{sa(t_m), s(t_m), sn(t_m), sr(t_m), {}^{\#}Q(t_m) \}$,

   čia – $sa(t_m)$ – į NBRS atėjusių paraiškų skaičius laiko tarpe $[t_0, t_m]$;

$s(t_m)$ – patvirtintų paraiškų skaičius laiko tarpe $[t_0, t_m]$;

$sn(t_m)$ – ištrintų paraiškos žurnalų skaičius laiko tarpe $[t_0, t_m]$;

$sr(t_m)$ – atmestų paraiškų skaičius laiko tarpe $[t_0, t_m]$;

${}^{\#}Q(t_m)$ – paraiškų, laukiančių vykdymo eilė, laiko momentu t_m;

7. Tolydžioji būsenos dedamoji

$Z_v(t_m) = \{w(e_1'', t_m), w(e_2'', t_m)\}$.

kur $w(e_1'', t_m)$ – paraiškos formavimo pabaigos laiko momentas;

$w(e_2'', t_m)$ – paraiškos duomenų rinkimo pabaigos laiko momentas.

8. Pradinė būsena

$sa(t_0) = 0$; $s(t_0) = 0$; $sn(t_0) = 0$; $sr(t_0) = 0$; ${}^{\#}Q(t_m) = 0$;

$w(e_1'', t_m) = \infty; w(e_2'', t_m) = \infty;$

PRAD = 0; PAB = 0;

Pagalbiniai kintamieji:

PRAD – paraiškos aptarnavimo pradžia;

PAB – paraiškos aptarnavimo pabaiga.

9. Perėjimo ir išėjimo operatoriai

$H(e_1')$: / Į NBRS siunčiama paraiška/

$${}^{\#}Q(t_{m+1}) = {}^{\#}Q(t_m) + 1;$$

$$sa(t_{m+1}) = sa(t_m) + 1;$$

$$PRAD = t_m;$$

$$w(e_1'', t_{m+1}) = t_m + \xi_{1j};$$

$$w(e_2'', t_{m+1}) = t_m + \xi_{2j};$$

$G(e_1')$: y_1 – Pareiškėjo užklausa į skyrių.

$H(e_2')$: / Iš žurnalizavimo agregato pranešimas apie paraiškos žurnalo ištrynimą/

$$sn(t_{m+1}) = sn(t_m) + 1;$$

$${}^{\#}Q(t_{m+1}) = {}^{\#}Q(t_m) - 1;$$

$$PAB = t_m;$$

$G(e_2')$: $Y = \varnothing$.

$H(e_3')$: / Iš registravimo agregato pranešimas apie paraiškos atmetimą /

$$sr(t_{m+1}) = sr(t_m) + 1;$$

$${}^{\#}Q(t_{m+1}) = {}^{\#}Q(t_m) - 1;$$

$PAB = t_m;$

$G(e'_3)$: $Y = \varnothing.$

$H(e'_4)$: / Iš registravimo agregato pranešimas apie paraiškos įregistravimą /

$s(t_{m+1}) = s(t_m) + 1;$

$^{\#}Q(t_{m+1}) = ^{\#}Q(t_m) - 1;$

$PAB = t_m;$

$G(e'_4)$: $Y = \varnothing.$

$H(e''_1)$: / Paruošimas NBRS posistemėje /

$w(e''_1, t_{m+1}) = \infty;$

$G(e''_1)$: $Y = \varnothing.$

$H(e''_2)$: / Paruošimo be NBRS /

$w(e''_2, t_{m+1}) = \infty;$

$G(e''_2)$: $Y = \varnothing.$

Agregato *„Žurnalizavimas“* specifikacija

1. Įėjimo signalų aibė X = {x_1, x_2},
   kur x_1 – pranešimas iš agregato *„Pareiškėjas“*;
   x_2 – pranešimas apie prašymą dokumento duomenų apie įmonę papildymui.
2. Išėjimo signalų aibė Y = {y_1, y_2},
   kur y_1 – Žurnalo duomenų priėmimo pranešimas;
   y_2 – Žurnalo duomenų ištrynimo pranešimas.
3. Išorinių įvykių aibė E′ = { e'_1, e'_2, e'_3, e'_4 },
   kur e'_1 – Žurnalo duomenų priėmimo siuntimas;
   e'_2 – Dokumento duomenų apie įmonę prašymo papildyti gavimas;
   e'_3 – Dokumento duomenų apie įmonę prašymo registracijai gavimas;
   e'_4 – Žurnalo duomenų ištrynimas.
4. Vidinių įvykių aibė E″ = { e''_1, e''_2, e''_3 },
   kur e''_1 – Žurnalo duomenų išsaugojimo pabaiga;
   e''_2 – Žurnalo duomenų ištrynimo pabaiga;
   e''_3 – Žurnalo duomenų priėmimo pabaiga.
5. Valdymo sekos

$e_1'' \rightarrow \{\xi_{1j}\}$, j=1... ∞, kur ξ_{1j} – Žurnalo duomenų išsaugojimo trukmė;

$e_2'' \rightarrow \{\xi_{2j}\}$, j=1... ∞, kur ξ_{2j} – Žurnalo duomenų ištrynimo trukmė;

$e_3'' \rightarrow \{\xi_{3j}\}$, j=1... ∞, kur ξ_{3j} – Žurnalo duomenų priėmimo trukmė.

6. Diskrečioji būsenos dedamoji

$v(t_m) = \{ja(t_m), j(t_m), jn(t_m), {}^{\#}Q(t_m)\}$,

čia – $ja(t_m)$ – į žurnalizavimą atėjusių paraiškų skaičius laiko tarpe $[t_0, t_m]$;

$j(t_m)$ – patvirtintų paraiškų skaičius laiko tarpe $[t_0, t_m]$;

$jn(t_m)$ – ištrintų paraiškos žurnalų skaičius laiko tarpe $[t_0, t_m]$;

${}^{\#}Q(t_m)$ – paraiškų, laukiančių aptarnavimo eilė, laiko momentu t_m;

7. Tolydžioji būsenos dedamoji

$Z_v(t_m) = \{w(e_1'', t_m), w(e_2'', t_m), w(e_3'', t_m)\}$.

kur $w(e_1'', t_m)$ – paraiškos išsaugojimo laiko momentas;

$w(e_2'', t_m)$ – paraiškos ištrynimo laiko momentas;

$w(e_3'', t_m)$ – paraiškos patvirtinimo laiko momentas.

8. Pradinė būsena

$ja(t_0) = 0$; $j(t_0) = 0$; $jn(t_0) = 0$; ${}^{\#}Q(t_0) = 0$;

$w(e_1'', t_m) = \infty; w(e_2'', t_m) = \infty; w(e_3'', t_m) = \infty;$

PRAD = 0; PAB = 0;

Pagalbiniai kintamieji:

PRAD – paraiškos aptarnavimo pradžia;

PAB – paraiškos aptarnavimo pabaiga.

9. Perėjimo ir išėjimo operatoriai

$H(e_1'')$: / Išsaugomas žurnalo duomenys /

${}^{\#}Q(t_{m+1}) = {}^{\#}Q(t_m) + 1;$

$PRAD = t_m;$

$ja(t_{m+1}) = ja(t_m) + 1;$

$w(e_1'', t_{m+1}) = t_m + \xi_{1j};$

$G(e_1'')$: $Y = \varnothing.$

$H(e_2'')$: / Ištrinami žurnalo duomenys /

${}^{\#}Q(t_{m+1}) = {}^{\#}Q(t_m) - 1;$

$PAB = t_m;$

$jn(t_{m+1}) = jn(t_m) + 1;$

$w(e_2'', t_{m+1}) = t_m + \xi_{2j}$;

$G(e_2''):$ $Y = \varnothing$.

$H(e_3''):$ / Priimami žurnalo duomenys /

${}^{\#}Q(t_{m+1}) = {}^{\#}Q(t_m) - 1$;

$\mathrm{PAB} = \mathrm{t_m}$;

$j(t_{m+1}) = j(t_m) + 1$;

$w(e_3'', t_{m+1}) = t_m + \xi_{3j}$;

$G(e_3''):$ y_1 – paraiška patvirtinta.

$H(e_1'):$ / Pranešimas siuntimas dėl žurnalo duomenų priėmimą /

$G(e_1'):$ $Y = \varnothing$.

$H(e_2'):$ / Pranešimo gavimas dėl prašymo dokumento duomenų apie įmonę papildymui /

$w(e_1'', t_{m+1}) = \infty$;

$w(e_2'', t_{m+1}) = \infty$;

$w(e_3'', t_{m+1}) = \infty$;

$G(e_2'):$ $Y = \varnothing$.

$H(e_3'):$ / Pranešimo gavimas dėl prašymo dokumento duomenų apie įmonę registracijai /

$w(e_1'', t_{m+1}) = \infty$;

$w(e_2'', t_{m+1}) = \infty$;

$w(e_3'', t_{m+1}) = \infty$;

$G(e_3'):$ $Y = \varnothing$.

$H(e_4'):$ / Pranešimas siuntimas dėl žurnalo duomenų ištrynimo /

$G(e_4'):$ $Y = \varnothing$.

Agregato *„Vykdymas“* specifikacija

1. Įėjimo signalų aibė $X = \{x_1, x_1\}$,
   kur x_1 – Pranešimas apie žurnalo duomenų priėmimą;
   x_2 – Pranešimas apie prašymą dokumento duomenų apie įmonę taisymui.
2. Išėjimo signalų aibė $Y = \{y_1\}$,
   kur y_1 – Pranešimas apie dokumento duomenų apie įmonę patvirtinimą.

3. Išorinių įvykių aibė $E' = \{e'_1, e'_2, e'_3\}$,

   kur e'_1 – Žurnalo duomenų priėmimo įvykis;

   e'_2 – Prašymo dokumento duomenų apie įmonę taisymo įvykis;

   e'_3 – Dokumento duomenų apie įmonę patvirtinimo įvykis.

4. Vidinių įvykių aibė $E'' = \{e''_1, e''_2, e''_3, e''_4\}$,

   kur e''_1 – Žurnalo duomenų peržiūros pabaiga;

   e''_2 – Užvedimo dokumento įmonės duomenims pildyti pabaiga;

   e''_3 – Dokumento duomenų apie įmonę išsaugojimo pabaiga;

   e''_4 – Dokumento duomenų apie įmonę siuntimo registratoriui pabaiga.

5. Valdymo sekos

   $e''_1 \rightarrow \{\xi_{1j}\}$, j=1... ∞, kur ξ_{1j} – Žurnalo duomenų peržiūros trukmė;

   $e''_2 \rightarrow \{\xi_{2j}\}$, j=1... ∞, kur ξ_{2j} – Dokumento užvedimo įmonės duomenims pildyti trukmė;

   $e''_3 \rightarrow \{\xi_{3j}\}$, j=1... ∞, kur ξ_{3j} – Dokumento duomenų apie įmonę išsaugojimo trukmė;

   $e''_4 \rightarrow \{\xi_{4j}\}$, j=1... ∞, kur ξ_{4j} – Dokumento duomenų apie įmonę siuntimo registratoriui trukmė.

6. Diskrečioji būsenos dedamoji

   $v(t_m) = \{da(t_m), d(t_m), {}^{\#}Q(t_m)\}$,

   čia – $da(t_m)$ – į duomenų vykdymą atėjusių paraiškų skaičius laiko tarpe $[t_0, t_m]$;

   $d(t_m)$ – perduotų registratoriui dokumentų skaičius laiko tarpe $[t_0, t_m]$;

   ${}^{\#}Q(t_m)$) – paraiškų, laukiančių aptarnavimo eilė, laiko momentu t_m;

7. Tolydžioji būsenos dedamoji

   $Z_v(t_m) = \{w(e''_1, t_m), w(e''_2, t_m), w(e''_3, t_m), w(e''_4, t_m)\}$.

   kur $w(e''_1, t_m)$ – Žurnalo duomenų peržiūros laiko momentas;

   $w(e''_2, t_m)$ – Dokumento užvedimo įmonės duomenims pildyti laiko momentas;

   $w(e''_3, t_m)$ – Dokumento duomenų apie įmonę išsaugojimo laiko momentas;

   $w(e''_4, t_m)$ – Dokumento duomenų apie įmonę siuntimo registratoriui laiko momentas.

8. Pradinė būsena

   $da(t_0) = 0$; $d(t_0) = 0$; ${}^{\#}Q(t_0) = 0$;

   $w(e''_1, t_m) = \infty; w(e''_2, t_m) = \infty; w(e''_3, t_m) = \infty; w(e''_4, t_m) = \infty;$

   PRAD = 0; PAB = 0;

   Pagalbiniai kintamieji:

PRAD – paraiškos aptarnavimo pradžia;

PAB – paraiškos aptarnavimo pabaiga.

9. Perėjimo ir išėjimo operatoriai

$H(e_1''):$ / Atėjo patvirtina paraiška peržiūrai /

$w(e_1'', t_{m+1}) = t_m + \xi_{1j};$

$G(e_1''):$ $Y = \varnothing.$

$H(e_2''):$ / Dokumento užvedimas įmonės duomenims pildyti /

${}^{\#}Q(t_{m+1}) = {}^{\#}Q(t_m) + 1;$

$PRAD = t_m;$

$da(t_{m+1}) = da(t_m) + 1;$

$w(e_2'', t_{m+1}) = t_m + \xi_{2j};$

$G(e_2''):$ $Y = \varnothing.$

$H(e_3''):$ / Dokumento duomenų apie įmonę išsaugojimas /

$w(e_3'', t_{m+1}) = t_m + \xi_{3j};$

$G(e_3''):$ $Y = \varnothing.$

$H(e_4''):$ / Dokumento duomenų apie įmonę siuntimas registratoriui /

${}^{\#}Q(t_{m+1}) = {}^{\#}Q(t_m) - 1;$

$\mathrm{PAB} = \mathrm{t_m};$

$d(t_{m+1}) = d(t_m) + 1;$

$w(e_4'', t_{m+1}) = t_m + \xi_{4j};$

$G(e_4''):$ y_1 – Pranešimas apie dokumento duomenų apie įmonę patvirtinimą.

$H(e_1'):$ / Pranešimo gavimas dėl žurnalo duomenų priėmimo /

$w(e_1'', t_{m+1}) = \infty;$

$w(e_2'', t_{m+1}) = \infty;$

$w(e_3'', t_{m+1}) = \infty;$

$w(e_4'', t_{m+1}) = \infty;$

$G(e_1'):$ $Y = \varnothing.$

$H(e_2'):$ / Pranešimo gavimas dėl prašymo dokumento duomenų apie įmonę taisymui /

$w(e_1'', t_{m+1}) = \infty;$

$w(e_2'', t_{m+1}) = \infty;$

$w(e_3'', t_{m+1}) = \infty;$

$w(e_4'', t_{m+1}) = \infty;$

$G(e_2'):$ $Y = \varnothing.$

$H(e_3'):$ / Pranešimo siuntimas dėl dokumento duomenų apie įmonę patvirtinimą /

$G(e_3'):$ $Y = \varnothing.$

Agregato *„Registravimas“* specifikacija

1. Įėjimo signalų aibė X = {x_1},

   kur x_1 – Pranešimas apie dokumento duomenų apie įmonę patvirtinimą.

2. Išėjimo signalų aibė Y = { y_1, y_2},

   kur y_1 – Pranešimas apie prašymą dokumento duomenų apie įmonę taisymui;

   y_2 – Pranešimas apie prašymą dokumento duomenų apie įmonę papildymui;

   y_3 – Pranešimas apie dokumento duomenų apie įmonę atmetimą;

   y_4 – Pranešimas apie dokumento duomenų apie įmonę įregistravimą.

3. Išorinių įvykių aibė E′ = { e_1', e_2', e_3', e_4', e_5' },

   kur e_1' – Pranešimo apie dokumento duomenų apie įmonę patvirtinimą gavimas;

   e_2' – Pranešimo apie prašymą įmonės dokumento duomenų taisymui siuntimas;

   e_3' – Pranešimo apie prašymą įmonės dokumento duomenų papildymui siuntimas;

   e_4' – Pranešimo apie įmonės dokumento duomenų atmetimą siuntimas;

   e_5' – Pranešimo apie įmonės dokumento duomenų įregistravimą siuntimas;

4. Vidinių įvykių aibė E″ = { e_1'', e_2'', e_3'', e_4'', e_5'', e_6'' },

   kur e_1'' – Dokumento duomenų apie įmonę peržiūros pabaiga;

   e_2'' – Siuntimo koreguoti dokumento duomenų apie įmonę pabaiga;

   e_3'' – Siuntimo papildyti dokumento duomenis apie įmonę pabaiga;

   e_4'' – Užklausimo įmonės kodo dokumento duomenims apie įmonę pabaiga;

   e_5'' – Atmetimo dokumento duomenų apie įmonę pabaiga;

   e_6'' – Įregistravimo dokumento duomenų apie įmonę pabaiga.

5. Valdymo sekos

   $e_1'' \rightarrow \{\xi_{1j}\}$, j=1... ∞, kur ξ_{1j} – Dokumento duomenų apie įmonę peržiūros trukmė;

   $e_2'' \rightarrow \{\xi_{2j}\}$, j=1... ∞, kur ξ_{2j} – Siuntimo koreguoti dokumento duomenų apie įmonę trukmė;

$e''_3 \rightarrow \{\xi_{3j}\}$, j=1... ∞, kur ξ_{3j} – Siuntimo papildyti dokumento duomenis apie įmonę trukmė;

$e''_4 \rightarrow \{\xi_{4j}\}$, j=1... ∞, kur ξ_{4j} – Užklausimo įmonės kodo dokumento duomenims apie įmonę trukmė;

$e''_5 \rightarrow \{\xi_{5j}\}$, j=1... ∞, kur ξ_{5j} – Atmetimo dokumento duomenų apie įmonę trukmė;

$e''_6 \rightarrow \{\xi_{6j}\}$, j=1... ∞, kur ξ_{6j} – Įregistravimo dokumento duomenų apie įmonę trukmė.

6. Diskrečioji būsenos dedamoji

   $v(t_m) = \{ra(t_m), r(t_m), rr(t_m), rc(t_m), rs(t_m), {}^{\#}Q(t_m)\}$,

   čia – $ra(t_m)$ – į agregatą *„Registravimas“* atėjusių paraiškų skaičius laiko tarpe $[t_0, t_m]$;

   $r(t_m)$ –įregistruotų dokumentų skaičius laiko tarpe $[t_0, t_m]$;

   $rr(t_m)$ –atmestų dokumentų skaičius laiko tarpe $[t_0, t_m]$;

   $rc(t_m)$ –nusiųstų pataisyti dokumentų skaičius laiko tarpe $[t_0, t_m]$;

   $rs(t_m)$ –nusiųstų papildyti dokumentų skaičius laiko tarpe $[t_0, t_m]$;

   ${}^{\#}Q(t_m)$) – paraiškų, laukiančių aptarnavimo eilė, laiko momentu t_m;

7. Tolydžioji būsenos dedamoji

   $Z_v(t_m) = \{ w(e''_1,t_m), w(e''_2,t_m), w(e''_3,t_m), w(e''_4,t_m), w(e''_5,t_m), w(e''_6,t_m)\}$.

   kur $w(e''_1,t_m)$ – Dokumento duomenų peržiūros laiko momentas;

   $w(e''_2,t_m)$ – Siuntimo koreguoti duomenis laiko momentas;

   $w(e''_3,t_m)$ – Siuntimo papildyti duomenis momentas;

   $w(e''_4,t_m)$ – Užklausimo įmonės kodo laiko momentas;

   $w(e''_5,t_m)$ – Atmetimo dokumento duomenų laiko momentas;

   $w(e''_6,t_m)$ – Įregistravimo dokumento duomenų laiko momentas.

8. Pradinė būsena

   $ra(t_0) = 0$; $r(t_0) = 0$; $rr(t_0) = 0$; $rc(t_0) = 0$; $rs(t_0) = 0$; ${}^{\#}Q(t_0) = 0$;

   $w(e''_1,t_m) = \infty; w(e''_2,t_m) = \infty; w(e''_3,t_m) = \infty; w(e''_4,t_m) = \infty;\ w(e''_5,t_m) = \infty; w(e''_6,t_m) = \infty;$

   PRAD = 0; PAB = 0;

   Pagalbiniai kintamieji:

   PRAD – paraiškos aptarnavimo pradžia;

   PAB – paraiškos aptarnavimo pabaiga.

9. Perėjimo ir išėjimo operatoriai

   $H(e''_1)$: / Atėjo įregistravimo paraiška peržiūrai /

   ${}^{\#}Q(t_{m+1}) = {}^{\#}Q(t_m) + 1;$

$PRAD = t_m$;

$ra(t_{m+1}) = ra(t_m) + 1$;

$w(e_1'', t_{m+1}) = t_m + \xi_{1j}$;

$G(e_1''):$ $Y = \varnothing$.

$H(e_2''):$ / Siuntimas koreguoti dokumento duomenis apie įmonę /

${}^{\#}Q(t_{m+1}) = {}^{\#}Q(t_m) - 1$;

$rc(t_{m+1}) = rc(t_m) + 1$;

$w(e_2'', t_{m+1}) = t_m + \xi_{2j}$;

$G(e_2''):$ y_1 – Pranešimas apie prašymą dokumento duomenų apie įmonę taisymui.

$H(e_3''):$ / Siuntimas papildyti dokumento duomenis apie įmonę /

${}^{\#}Q(t_{m+1}) = {}^{\#}Q(t_m) - 1$;

$rs(t_{m+1}) = rs(t_m) + 1$;

$w(e_3'', t_{m+1}) = t_m + \xi_{3j}$;

$G(e_3''):$ y_2 – Pranešimas apie prašymą dokumento duomenų apie įmonę papildymui.

$H(e_4''):$ / Užklausimas įmonės kodo dokumento duomenims apie įmonę /

$w(e_4'', t_{m+1}) = t_m + \xi_{4j}$;

$G(e_4''):$ $Y = \varnothing$.

$H(e_5''):$ / Atmetimas dokumento duomenų apie įmonę /

${}^{\#}Q(t_{m+1}) = {}^{\#}Q(t_m) - 1$;

$\mathrm{PAB} = \mathrm{t_m}$;

$rr(t_{m+1}) = rr(t_m) + 1$;

$w(e_5'', t_{m+1}) = t_m + \xi_{5j}$;

$G(e_5''):$ y_3 – Pranešimas apie dokumento duomenų apie įmonę atmetimą.

$H(e_6''):$ / Įregistruojami dokumento duomenys apie įmonę /

${}^{\#}Q(t_{m+1}) = {}^{\#}Q(t_m) - 1$;

$\mathrm{PAB} = \mathrm{t_m}$;

$r(t_{m+1}) = r(t_m) + 1$;

$w(e_6'', t_{m+1}) = t_m + \xi_{6j}$;

$G(e_6''):$ y_4 – Pranešimas apie dokumento duomenų apie įmonę įregistravimą.

$H(e_1')$: / Pranešimo gavimas dėl dokumento duomenų apie įmonę patvirtinimo /

$w(e_1'', t_{m+1}) = \infty;$

$w(e_2'', t_{m+1}) = \infty;$

$w(e_3'', t_{m+1}) = \infty;$

$w(e_4'', t_{m+1}) = \infty;$

$w(e_5'', t_{m+1}) = \infty;$

$w(e_6'', t_{m+1}) = \infty;$

$G(e_1')$: $Y = \varnothing.$

$H(e_2')$: / Pranešimo siuntimas dėl prašymo dokumento duomenų apie įmonę taisymo /

$G(e_2')$: $Y = \varnothing.$

$H(e_3')$: / Pranešimo siuntimas dėl prašymo dokumento duomenų apie įmonę papildymo /

$G(e_3')$: $Y = \varnothing.$

$H(e_4')$: / Pranešimo siuntimas dėl dokumento duomenų apie įmonę atmetimo /

$G(e_4')$: $Y = \varnothing.$

$H(e_5')$: / Pranešimo siuntimas dėl dokumento duomenų apie įmonę įregistravimo /

$G(e_5')$: $Y = \varnothing.$

7.3. NACIONALINĖS VERSLO REGISTRAVIMO SISTEMOS AGREGATŲ REZULTATŲ LENTELĖS

Pagal pateiktą konceptualų modelį apskaičiuoti užduotus sistemos reikalavimus buvo pasirinktas intervalas nuo 2010 metų gruodžio 1 dienos iki gruodžio 31 dienos. Buvo sekami dienos pabaigoje užfiksuoti paraiškų skaičiai kiekviename agregate. Pagrindinis tyrimo reikalavimas, kad paraiškos, pakliuvusios į sistemą, aptarnavimo laikas būtų ne ilgesnis nei 7 dienos. Žinoma, galimos ir išlygos, nes paraiškos vykdymas gali priklausyti ne tik nuo sistemos vartotojų, bet ir nuo pareiškėjo greičio pateikti prašomus duomenis į prašymą papildyti duomenis apie įmonę. Taip pat norima apskaičiuoti kiekvieno agregato apkrautumą, kad būtų galima nustatyti, kuriame žingsnyje paraiškos vykdymas užtrunka ilgiausiai, taip nustatant kritinius taškus, kuriuose būtų galima pakeisti vykdymo sekas arba pasiūlyti naujus sprendimus. Taip pat skaičiuojami naudingumo koeficientas ir apkrautumo koeficientas visai sistemai ir kiekvienam agregatui atskirai.

Naudingumo koeficientas – tai apdorotų paraiškų skaičius agregate padalintas iš į agregatą paduotų paraiškų skaičiaus pridėjus buvusių paraiškų skaičių tiriamame laiko periode T_m. Naudingumo koeficientą žymime η.

Pasinaudojus naudingumo koeficientu galima spręsti apie agregato produktyvumą laiko momente ir jo skirtumus užduotame intervale nuo 2010 metų gruodžio 1 dienos iki gruodžio 31 dienos.

Apkrovos koeficientas – tai paraiškos apdorojimo laikas agregate padalintas iš paraiškos maksimalaus buvimo agregate laiko tiriamame laiko periode T_m. Apkrautumo koeficientą žymime *APK*.

Pirmasis tiriamasis agregatas *„Pareiškėjas“*, tai bendrasis paraiškos kelias nuo pradinio taško išsaugojimo iki pat galino taško, tai yra išregistravimo, atmetimo ar paraiškos žurnalo ištrynimo. Agregato *„Pareiškėjas“* naudingumo koeficiento laiko periode $[T_{m-1};T_m]$ apskaičiavimo formulė yra:

$$\eta(T_m)=\frac{sr(T_m)+sn(T_m)+s(T_m)}{sa(T_m)+^{\#}Q(T_{m-1})};$$

Agregato *„Pareiškėjas“* naudingumo koeficiento laiko periode $[T_{m-1};T_m]$ apskaičiavimo formulė yra:

$$APK(T_m)=1-\frac{\sum_{i=1}^{sa(T_m)}(PRAD-T_{m-1})+\sum_{i=1}^{sr(T_m)}(T_m-PAB)+\sum_{i=1}^{sn(T_m)}(T_m-PAB)+\sum_{i=1}^{s(T_m)}(T_m-PAB)}{sa(T_m)\cdot(T_m-T_{m-1})+^{\#}Q(T_{m-1})\cdot(T_m-T_{m-1})};$$

Intervalas $[T_{m-1};T_m]$ lygus vienai parai.

6 lentelė. Agregato „Pareiškėjas“ rezultatai
Šaltinis: sudarė autorius

T_m	sa(T_m)	sn(T_m)	sr(T_m)	s(T_m)	#Q(T_m)	η(T_m)	APK(T_m)	Avgn(T_m)	Avgr(T_m)	Avga(T_m)	Avg(T_m)
2010.11.31	0	0	0	0	0	0	0	0	0	0	0
2010.12.01	1200	13	12	240	935	0,22	0,40	19 min 19 s	1 h 8 min 55 s	1 h 2 min 56 s	1 h 1 min 4 s
2010.12.02	1264	9	43	398	1749	0,20	0,62	3 h 22 min 29 s	11 h 22 min 29 s	7 h 47 min 14 s	8 h 2 min 30 s
2010.12.03	1273	20	21	496	2485	0,18	0,71	8 h 21 min 18 s	21 h 7 min 43 s	15 h 17 min 45 s	15 h 15 min 56 s
2010.12.04	137	3	2	62	2555	0,03	0,96	5 h 38 min 33 s	2 d. 18 h 52 min 40 s	1 d. 38 min 35 s	1 d. 1 h 3 min 11 s
2010.12.05	16	0	1	4	2566	0,00	1,00	-	4 d. 6 h 10 min 56 s	1 d. 23 h 33 min 40 s	2 d. 10 h 29 min 7 s
2010.12.06	1270	25	41	661	3109	0,19	0,75	2 d. 1 h 13 min 52 s	2 d. 11 h 2 min 35 s	2 d. 7 h 3 min 8 s	2 d. 7 h 4 min 38 s
2010.12.07	1279	13	33	678	3664	0,16	0,78	1 d. 2 h 43 min 23 s	3 d. 16 h 17 min 13 s	2 d. 8 h 31 min 57 s	2 d. 9 h 26 min 40 s
2010.12.08	1138	31	19	704	4048	0,16	0,80	16 h 45 min 40 s	2 d. 16 h 8 min 50 s	3 d. 1 h 45 min 44 s	2 d. 23 h 10 min 35 s
2010.12.09	1303	24	57	768	4502	0,16	0,80	8 h 45 min 6 s	3 d. 12 h 7 min 45 s	2 d. 12 h 49 min 5 s	2 d. 12 h 54 min 40 s
2010.12.10	1107	15	64	915	4615	0,18	0,82	13 h 46 min 29 s	2 d. 15 h 27 min 53 s	3 d. 4 h 4 min 46 s	3 d. 2 h 19 min 37 s
2010.12.11	88	0	20	84	4599	0,02	0,98	-	2 d. 21 h 15 min 50 s	3 d. 7 h 46 min 41 s	3 d. 5 h 45 min 22 s
2010.12.12	26	0	9	17	4599	0,01	0,99	-	3 d. 21 h 27 min 9 s	2 d. 14 h 55 min 16 s	3 d. 1 h 29 min 23 s
2010.12.13	1324	11	58	967	4887	0,17	0,80	3 d. 17 h 10 min 15 s	5 d. 14 h 40 min 34 s	4 d. 1 h 49 min 1 s	4 d. 3 h 47 min 19 s
2010.12.14	1306	18	55	871	5249	0,15	0,82	1 d. 16 h 10 min 49 s	4 d. 13 h 47 min 35 s	3 d. 21 h 35 min 15 s	3 d. 21 h 30 min 48 s
2010.12.15	1247	21	73	879	5523	0,15	0,83	2 d. 20 h 45 min 55 s	4 d. 19 h 2 min 40 s	4 d. 5 h 56 min 20 s	4 d. 6 h 12 min 22 s
2010.12.16	1213	19	56	896	5765	0,14	0,84	1 d. 7 h 35 min 21 s	4 d. 15 h 59 min 18 s	4 d. 4 h 21 min 54 s	4 d. 3 h 41 min 23 s
2010.12.17	1096	20	59	909	5873	0,14	0,86	2 d. 11 h 37 min 39 s	3 d. 21 h 15 min 16 s	3 d. 18 h 49 min 14 s	3 d. 18 h 20 min 4 s
2010.12.18	150	0	27	190	5806	0,04	0,97	-	4 d. 5 h 9 min 25 s	6 d. 7 h 53 min 6 s	6 d. 1 h 34 min 24 s
2010.12.19	13	1	13	81	5724	0,02	0,99	11 d. 45 min 9 s	10 d. 17 h 6 min 33 s	9 d. 14 min 33 s	9 d. 6 h 20 min 43 s
2010.12.20	1504	34	73	1172	5949	0,18	0,81	1 d. 23 h 59 min 32 s	7 d. 12 h 47 min 5 s	4 d. 19 h 1 min 11 s	4 d. 20 h 59 min 29 s
2010.12.21	1479	18	81	1004	6325	0,15	0,83	14 h 16 s	4 d. 13 h 11 min 36 s	4 d. 9 h 21 min 55 s	4 d. 8 h 9 min 20 s
2010.12.22	1434	22	64	970	6703	0,14	0,84	1 d. 17 min 58 s	5 d. 8 h 58 min 24 s	4 d. 3 h 47 min 29 s	4 d. 3 h 59 min 14 s
2010.12.23	1474	12	80	1225	6860	0,16	0,83	1 d. 3 h 52 min 7 s	5 d. 7 h 11 min 24 s	4 d. 10 h 3 min 2 s	4 d. 10 h 37 min 20 s
2010.12.24	1288	21	73	891	7163	0,12	0,87	3 d. 19 h 25 min 46 s	4 d. 6 h 42 min 12 s	3 d. 12 h 32 min 49 s	3 d. 14 h 2 min 22 s
2010.12.25	246	1	15	120	7273	0,02	0,98	3 d. 18 h 41 min 58 s	9 d. 8 h 21 min 20 s	4 d. 21 h 44 min 18 s	5 d. 9 h 17 min 56 s
2010.12.26	14	0	25	28	7234	0,01	1,00	-	5 d. 15 h 15 min 1 s	7 d. 3 h 12 min 42 s	6 d. 10 h 14 min 56 s
2010.12.27	987	22	58	937	7204	0,12	0,88	2 d. 19 h 58 min 51 s	10 d. 16 h 1 min 36 s	5 d. 11 h 1 min 57 s	5 d. 16 h 47 min 49 s
2010.12.28	1656	23	88	1335	7414	0,16	0,83	1 d. 21 h 31 min 55 s	7 d. 4 h 28 min 54 s	5 d. 8 h 47 min 37 s	5 d. 10 h 7 min 40 s
2010.12.29	1528	12	94	1066	7770	0,13	0,85	4 d. 15 h 27 min 3 s	8 d. 11 h 38 min 36 s	5 d. 7 h 2 min 35 s	5 d. 13 h 1 min 37 s
2010.12.30	1129	27	106	1171	7595	0,15	0,86	2 d. 11 h 50 min 14 s	5 d. 10 h 25 min 7 s	4 d. 11 h 5 min 21 s	4 d. 12 h 26 s
2010.12.31	1704	13	86	1239	7961	0,14	0,84	16 h 24 min 35 s	6 d. 19 h 40 min 20 s	4 d. 6 h 49 min 21 s	4 d. 9 h 53 min 39 s
Iš viso:	**30893**	**448**	**1506**	**20978**	**5281**	**0,12**	**0,84**	**1 d. 17 h 5 min 49 s**	**5 d. 6 h 37 min 37 s**	**3 d. 22 h 58 min 42 s**	**4 d. 15 s**

Vidutinis paraiškų skaičius agregate per dieną intervale $[T_{m-1};T_m]$:

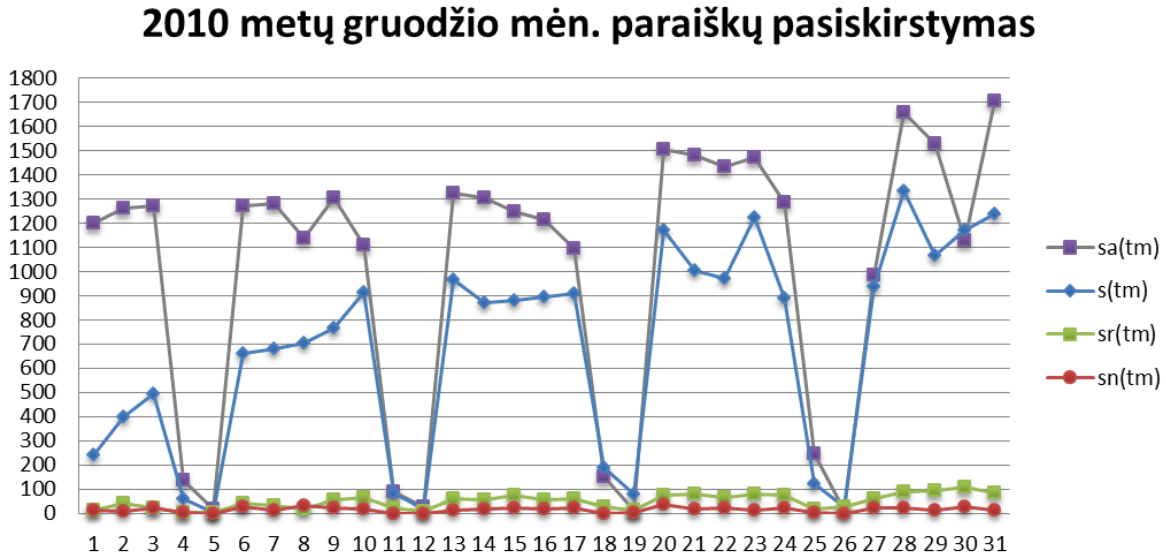
$$\frac{30893}{31} \approx 996{,}55;$$

Per dieną į agregatą patenka vidutiniškai 997 paraiškos. Iš jų per dieną vidutiniškai intervale $[T_{m-1};T_m]$ apdorojama:

$$\frac{448+1506+20978}{31} = \frac{22548}{31} \approx 727{,}35;$$

Vidutinis paraiškos apdorojimo laikas agregate 4 dienos. Pagal NBRS specifikaciją vykdymo laikas neturėtų užtrukti ilgiau nei 7 dienas. Kaip matome, pagal vidutinius rezultatus agregatas dirba korektiškai ir produktyviai.

Pagal pateiktus rezultatus nubraižome paraiškų pasiskirstymo grafiką laike. Kaip matome, paraiškų skaičius pastoviai kyla ir daugiau paraiškų įregistruojama, nei jų atmetama ar žurnalai ištrinami. Iš to galime daryti išvadą, kad agregatas dirba korektiškai ir žmogiškų klaidų faktorius yra ganėtinai mažas. Kitas pastebėjimas, kad daugiausia paraiškų apdorojama darbo savaitės pradžioje ir jos viduryje, o savaitės pabaigoje jaučiamas apkrautumo sumažėjimas. Kaip matome iš rezultatų agregatas savaitgalį visiškai neapkrautas ir vykdomos tik atsitiktinės paraiškos.

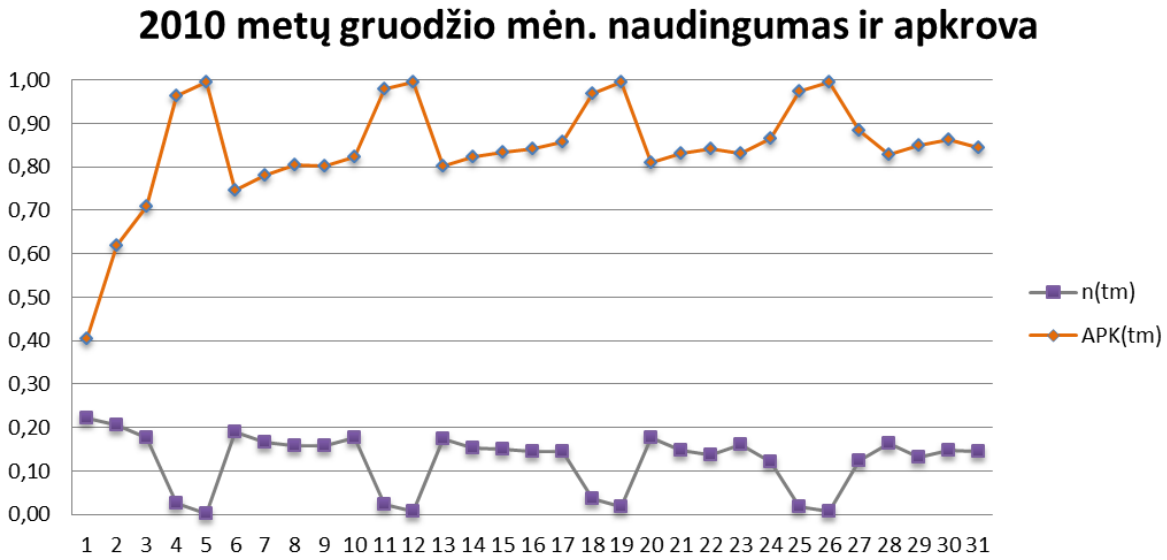


24 pav. Paraiškų pasiskirstymas NBRS

Šaltinis: sudarė autorius

Pagal pateiktus rezultatus apskaičiavome sistemos vidutinį naudingumo koeficientą. Jis yra apie 0,12, o vidutinis apkrautumo koeficientas – 0,84. Žemiau pateikta naudingumo ir apkrovos koeficiento kitimas laiko intervale nuo 2010 metų gruodžio 1 dienos iki gruodžio 31 dienos. Kaip matome, agregato apkrautumas pastoviai didėja, savaitgaliais pasiekia aukščiausią tašką, nes paraiškos, esančios

eilėje, nėra vykdomos. Iš naudingumo koeficiento matyti, kad darbo dienomis agregatas dirba pastoviai ir stabiliai.



25 pav. NBRS naudingumo ir apkrovos koeficientų kitimas

Šaltinis: sudarė autorius

Antrasis tiriamasis agregatas *„Žurnalizavimas“*, tai paraiškos kelias nuo pradinio taško išsaugojimo iki paraiškos žurnalo priėmimo arba žurnalo ištrynimo. Agregato *„Žurnalizavimas“* naudingumo koeficiento laiko periode $[T_{m-1};T_m]$ apskaičiavimo formulė yra:

$$\eta(d_m)=\frac{jn(T_m)+j(T_m)}{ja(T_m)+^{\#}Q(T_{m-1})};$$

Agregato *„Pareiškėjas“* naudingumo koeficiento laiko periode $[T_{m-1};T_m]$ apskaičiavimo formulė yra:

$$APK(T_m)=1-\frac{\sum_{i=1}^{ja(T_m)}(PRAD-T_{m-1})+\sum_{i=1}^{jn(T_m)}(T_m-PAB)+\sum_{i=1}^{j(T_m)}(T_m-PAB)}{ja(T_m)\cdot(T_m-T_{m-1})+^{\#}Q(T_{m-1})\cdot(T_m-T_{m-1})};$$

7 lentelė. Agregato „Žurnalizavimas“ rezultatai
Šaltinis: sudarė autorius

T_m	$ja(T_m)$	$jn(T_m)$	$j(T_m)$	$^{\#}Q(T_m)$	$\eta(T_m)$	$APK(T_m)$	$Avgn(T_m)$	$Avga(T_m)$	$Avg(T_m)$
2010.11.31	0	0	0	0	0	0	0	0	0
2010.12.01	1211	13	1170	28	0,98	0,01	19 min 19 s	1 min 48 s	1 min 59 s
2010.12.02	1277	9	1235	61	0,95	0,03	3 h 22 min 29 s	3 min 39 s	5 min 5 s
2010.12.03	1291	20	1239	93	0,93	0,06	8 h 21 min 18 s	5 min 14 s	13 min 7 s
2010.12.04	145	3	140	95	0,60	0,40	5 h 38 min 33 s	57 min 25 s	1 h 3 min 19 s
2010.12.05	23	0	19	99	0,16	0,82	-	34 s	34 s
2010.12.06	1297	25	1272	99	0,93	0,07	2 d. 1 h 13 min 52 s	50 min 23 s	1 h 46 min 21 s
2010.12.07	1314	13	1269	131	0,91	0,08	1 d. 2 h 43 min 23 s	5 min 3 s	21 min 16 s
2010.12.08	1180	31	1148	132	0,90	0,10	16 h 45 min 40 s	24 min 42 s	50 min 30 s
2010.12.09	1385	24	1335	158	0,90	0,10	8 h 45 min 6 s	8 min 55 s	18 min 2 s
2010.12.10	1211	15	1182	172	0,87	0,12	13 h 46 min 29 s	7 min 1 s	17 min 17 s
2010.12.11	122	0	121	173	0,41	0,59	-	10 min 9 s	10 min 9 s
2010.12.12	30	0	28	175	0,14	0,86	-	24 s	24 s
2010.12.13	1464	11	1440	188	0,89	0,11	3 d. 17 h 10 min 15 s	23 min 27 s	1 h 3 min 49 s
2010.12.14	1431	18	1394	207	0,87	0,12	1 d. 16 h 10 min 49 s	26 min 8 s	56 min 32 s
2010.12.15	1391	21	1364	213	0,87	0,13	2 d. 20 h 45 min 55 s	9 min 55 s	1 h 12 min 20 s
2010.12.16	1377	19	1333	238	0,85	0,14	1 d. 7 h 35 min 21 s	9 min 45 s	36 min 14 s
2010.12.17	1253	20	1229	242	0,84	0,16	2 d. 11 h 37 min 39 s	21 min 30 s	1 h 18 min 26 s
2010.12.18	191	0	189	244	0,44	0,56	-	24 s	24 s
2010.12.19	29	1	26	246	0,10	0,90	11 d. 45 min 9 s	55 s	9 h 49 min 13 s
2010.12.20	1669	34	1611	270	0,86	0,13	1 d. 23 h 59 min 32 s	7 min 36 s	1 h 6 min 57 s
2010.12.21	1638	18	1601	289	0,85	0,15	14 h 16 s	8 min 32 s	17 min 46 s
2010.12.22	1640	22	1604	303	0,84	0,15	1 d. 17 min 58 s	8 min 55 s	28 min 31 s
2010.12.23	1631	12	1589	333	0,83	0,16	1 d. 3 h 52 min 7 s	3 min 13 s	15 min 43 s
2010.12.24	1488	21	1464	336	0,82	0,18	3 d. 19 h 25 min 46 s	20 min 39 s	1 h 37 min 56 s
2010.12.25	293	1	282	346	0,45	0,54	3 d. 18 h 41 min 58 s	19 s	19 min 33 s
2010.12.26	28	0	27	347	0,07	0,93	-	35 s	35 s
2010.12.27	1129	22	1105	349	0,76	0,24	2 d. 19 h 58 min 51 s	34 min 50 s	1 h 53 min 46 s
2010.12.28	1901	23	1865	362	0,84	0,16	1 d. 21 h 31 min 55 s	6 min 24 s	39 min 36 s
2010.12.29	1754	12	1715	389	0,82	0,18	4 d. 15 h 27 min 3 s	28 min 37 s	1 h 14 min 53 s
2010.12.30	1242	27	1201	403	0,75	0,24	2 d. 11 h 50 min 14 s	7 min 37 s	1 h 26 min 23 s
2010.12.31	2065	13	2030	425	0,83	0,17	16 h 24 min 35 s	12 min 6 s	18 min 17 s
Iš viso:	**34100**	**448**	**33227**	**231**	**0,72**	**0,28**	**1 d. 17 h 5 min 49 s**	**14 min 15 s**	**46 min 52 s**

Vidutinis paraiškų skaičius agregate lygus vidutiniškai 1100 paraiškos per dieną intervale $[T_{m-1};T_m]$.

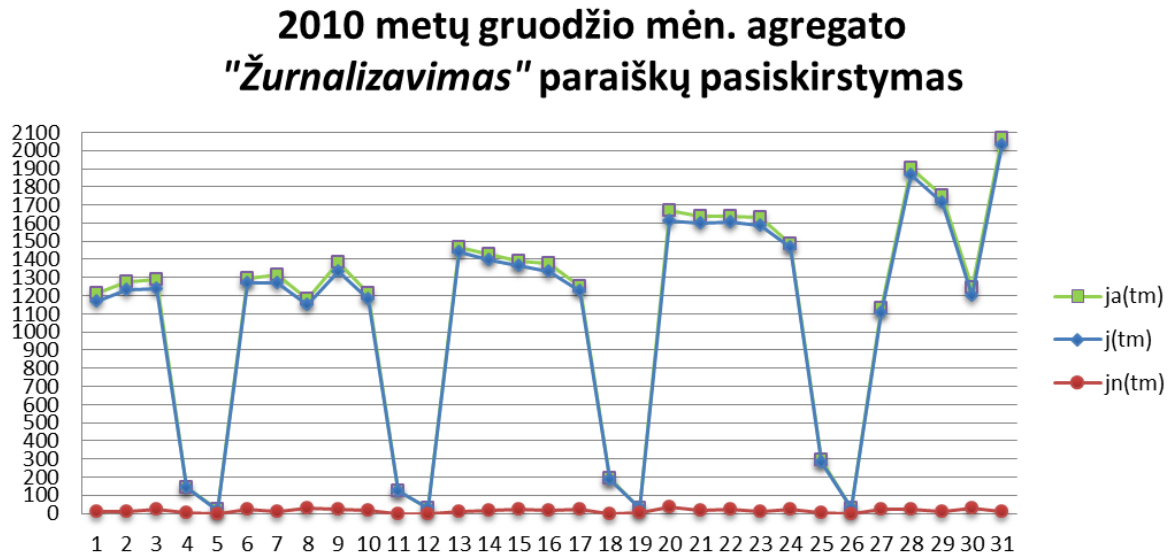
$$\frac{34100}{31}=1100;$$

Iš visų patekusių į agregatą paraiškų vidutiniškai per dieną intervale $[T_{m-1};T_m]$ apdorojama:

$$\frac{448+33227}{31}=\frac{33675}{31}\approx 1086{,}29;$$

Kaip matome, vartotojai, kurie priskirti prie žurnalizavimo, per dieną apdoroja 1086 paraiškas. Tai gana aukštas rezultatas. Vidutinis paraiškos užsibuvimo laikas agregate beveik 1 val. Vidutinis apkrautumas – 231 paraiškos agregate per dieną.

Pagal pateiktus rezultatus nubraižome *„Žurnalizavimas“* agregate paraiškų pasiskirstymo grafiką laike. Kaip matome, paraiškų skaičius pastoviai kyla. Mėnesio gale į agregatą atėjusių paraiškų skaičius tapo mažesnis nei patvirtintų paraiškų skaičius. Iš to galime daryti išvadą, kad taip buvo pasistengta sumažinti eilėje stovinčių paraiškų skaičių. Kitas pastebėjimas, kad daugiausia paraiškų apdorojama darbo savaitės pradžioje. Kaip matome iš rezultatų, agregatas savaitgalį visiškai neapkrautas ir patvirtinamos atsitiktinės paraiškos.

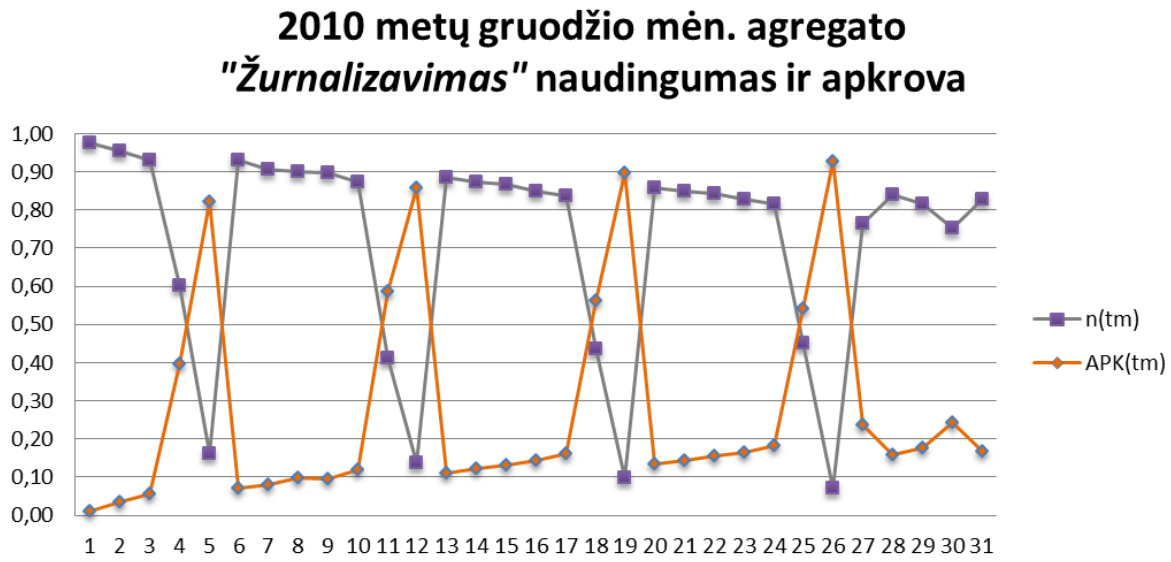


26 pav. Paraiškų pasiskirstymas „Žurnalizavimas“ agregate

Šaltinis: sudarė autorius

Pagal pateiktus rezultatus apskaičiavome agregato *„Žurnalizavimas“* vidutinį naudingumo koeficientą. Jis yra apie 0,72 o vidutinis agregato apkrautumo koeficientas – 0,28. Žemiau pateikta naudingumo ir apkrovos koeficiento kitimas laiko intervale nuo 2010 metų gruodžio 1 dienos iki gruodžio 31 dienos agregate *„Žurnalizavimas“*. Kaip matome, agregato apkrautumas pastoviai didėja, savaitgaliais pasiekia aukščiausia tašką, nes paraiškos esančios eilėje nėra vykdomos, o mėnesio gale

agregate apkrautumas sumažėjo, nes labai sumažintas paraiškų kiekis. Iš naudingumo koeficiento matyti, kad darbo dienomis agregatas dirba pastoviai ir stabiliai, o mėnesio gale agregato naudingumas pats didžiausias.



27 pav. agregato „Žurnalizavimas“ naudingumo ir apkrovos koeficientų kitimas

Šaltinis: sudarė autorius

Trečiasis tiriamasis agregatas *„Vykdymas“*, tai paraiškos kelias nuo paraiškos pateikimo vykdančiajam klerkui iki duomenų užpildymo ir pateikimo registratoriui įregistruoti. Agregato *„Vykdymas“* naudingumo koeficiento laiko periode $[T_{m-1};T_m]$ apskaičiavimo formulė yra:

$$\eta(T_m)=\frac{d(T_m)}{da(T_m)+{}^{\#}Q(T_{m-1})};$$

Agregato *„Vykdymas“* naudingumo koeficiento laiko periode $[T_{m-1};T_m]$ apskaičiavimo formulė yra:

$$APK(T_m)=1-\frac{\sum_{i=1}^{da(T_m)}(PRAD-T_{m-1})+\sum_{i=1}^{d(T_m)}(T_m-PAB)}{da(T_m)\cdot(T_m-T_{m-1})+{}^{\#}Q(T_{m-1})\cdot(T_m-T_{m-1})};$$

8 lentelė. Agregato „Vykdymas“ rezultatai

Šaltinis: sudarė autorius

T_m	$da(T_m)$	$d(T_m)$	${}^{\#}Q(T_m)$	$\eta(T_m)$	$APK(T_m)$	$Avg(T_m)$
2010.11.31	0	0	0	0	0	0
2010.12.01	464	400	64	0,86	0,93	23 min 56 s
2010.12.02	728	669	123	0,84	0,88	1 h 34 min 9 s
2010.12.03	756	731	148	0,83	0,84	2 h 59 min 42 s
2010.12.04	228	184	192	0,49	0,56	1 h 7 min 36 s
2010.12.05	236	190	238	0,44	0,50	28 min 43 s
2010.12.06	983	949	272	0,78	0,80	8 h 50 s
2010.12.07	1181	1167	286	0,80	0,81	5 h 32 min 45 s
2010.12.08	1196	1198	284	0,81	0,81	6 h 8 min 59 s
2010.12.09	1392	1355	321	0,81	0,82	5 h 51 min 43 s

2010.12.10	1088	1128	281	0,80	0,78	5 h 25 min 48 s
2010.12.11	226	202	305	0,40	0,42	3 h 2 min 55 s
2010.12.12	88	73	320	0,19	0,21	14 h 15 min 12 s
2010.12.13	1284	1279	325	0,80	0,80	11 h 4 min 52 s
2010.12.14	1360	1355	330	0,80	0,81	6 h 38 min 51 s
2010.12.15	1406	1420	316	0,82	0,81	5 h 23 min 28 s
2010.12.16	1389	1328	377	0,78	0,80	4 h 54 min 45 s
2010.12.17	1321	1319	379	0,78	0,77	5 h 26 min 13 s
2010.12.18	609	510	478	0,52	0,57	2 h 52 min 9 s
2010.12.19	295	282	491	0,36	0,37	7 h 8 min 48 s
2010.12.20	1624	1653	462	0,78	0,77	12 h 48 min 19 s
2010.12.21	1633	1621	474	0,77	0,77	8 h 8 min 31 s
2010.12.22	1385	1427	432	0,77	0,75	7 h 28 min 5 s
2010.12.23	1564	1528	468	0,77	0,77	6 h 28 min 55 s
2010.12.24	1358	1353	473	0,74	0,74	6 h 38 min 58 s
2010.12.25	729	639	563	0,53	0,57	4 h 2 min 3 s
2010.12.26	196	184	575	0,24	0,25	6 h 34 min 33 s
2010.12.27	1309	1298	586	0,69	0,69	11 h 33 min 51 s
2010.12.28	1676	1769	493	0,78	0,75	11 h 36 min 40 s
2010.12.29	1752	1723	522	0,77	0,77	8 h 30 min 16 s
2010.12.30	1768	1775	515	0,78	0,77	6 h 59 min 39 s
2010.12.31	1524	1527	512	0,75	0,75	7 h 12 min 40 s
Iš viso:	**32748**	**32236**	**374**	**0,69**	**0,70**	**7 h 8 min 30 s**

Vidutinis paraiškų skaičius agregate per dieną intervale $[T_{m-1};T_m]$:

$$\frac{32748}{31} \approx 1056{,}39;$$

Per dieną į agregatą paduodama vidutiniškai 1056 paraiškos, iš jų vidutiniškai per dieną intervale $[T_{m-1};T_m]$ apdorojama:

$$\frac{32236}{31} \approx 1039{,}87;$$

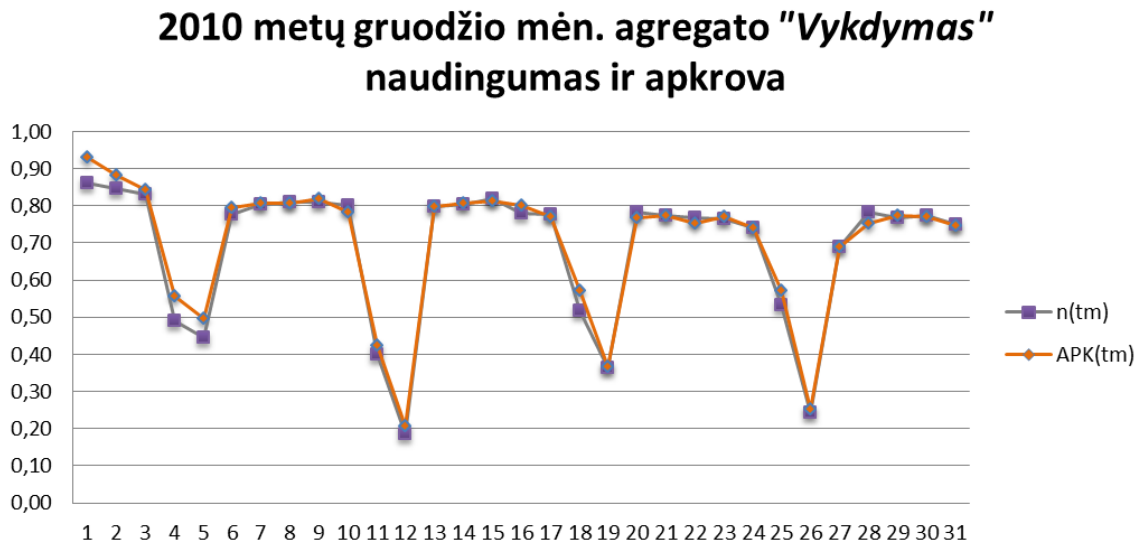
Kaip matome, vartotojai, kurie priskirti prie vykdymo, per dieną apdoroja 1040 paraiškas. Tai gana aukštas rezultatas, kaip ir žurnalizavimo agregate, todėl galima daryti išvadas, kad paraiška ilgai neužsibūna šiuose agregatuose ir greitai vykdoma žurnalizuojančio ir vykdančiojo klerkų. Vidutinis paraiškos užsibuvimo laikas agregate 7 val., tai yra vykdymas baigiamas per 1 darbo dieną. Vidutinis apkrautumas – 374 paraiškos agregate per dieną. Tai yra mažiau už žurnalizavimo agregato, nes iš žurnalizavimo agregato paraiška gali ir nepatekti į vykdymo agregatą, nes gali būti, kad paraiškos žurnalas bus ištrintas.

Pagal pateiktus rezultatus nubraižome *„Vykdymas“* agregate paraiškų pasiskirstymo grafiką laike. Kaip matome iš *„Žurnalizavimas“* agregato atėjusių paraiškų į *„Vykdymas“* agregatą skaičius nuolat kyla. Taip pat pastebime, jog atėjusių paraiškų skaičius yra beveik vienodas pabaigtų vykdyti paraiškų skaičiui. Iš to galima spręsti, kad naudingumo koeficientas bus labai aukštas. Kitas pastebėjimas, kad daugiausia paraiškų apdorojama darbo savaitės pradžioje. Kaip matome iš rezultatų, agregatas savaitgalį visiškai neapkrautas ir registratoriui spręsti pasiunčiamos atsitiktinės paraiškos.

28 pav. Paraiškų pasiskirstymas „Vykdymas“ agregate

Šaltinis: sudarė autorius

Pagal pateiktus rezultatus apskaičiavome agregato *„Vykdymas“* vidutinį naudingumo koeficientą. Jis yra apie 0,69 o vidutinis agregato apkrautumo koeficientas – 0,70. Žemiau pateikta naudingumo ir apkrovos koeficiento kitimas laiko intervale nuo 2010 metų gruodžio 1 dienos iki gruodžio 31 dienos agregate *„Vykdymas“*. Kaip matome, agregato apkrautumas pastoviai mažėja, savaitgaliais pasiekia žemiausią tašką, nes agregate paraiškos vykdomo kuo greičiau ir darbas vyksta net savaitgaliais. Iš naudingumo koeficiento matyti, kad darbo dienomis agregatas dirba pastoviai ir stabiliai ir yra aukščiausias iš visų agregatų. Paraiškos šiame agregate neužsibūna.



29 pav. agregato „Vykdymas“ naudingumo ir apkrovos koeficientų kitimas

Šaltinis: sudarė autorius

Ketvirtasis tiriamasis agregatas *„Registravimas“*, tai paraiškos kelias nuo paraiškos užpildytų duomenų, tokių kaip įmonės akcininkai, kapitalas ir daugelis kitų, kurių pildomas kiekis priklauso nuo įmonės tipo, pateikimo registratoriui iki registratoriaus sprendimo dėl įregistravimo ar atmetimo priėmimo. Rekomendacija, ką daryti registratoriui, yra pasiūloma ir vykdančiojo klerko, kuris pildė įmonės duomenis. Taip pat suradęs neatitikimus registratorius pagal neatitikimus gali pasiųsti paraišką vykdančiajam klerkui pataisyti, o jei neatitikimai didesni ir reikia papildomų dokumentų iš pareiškėjo, gali pasiųsti papildyti žurnalizuojančiam klerkui.

Agregate nenagrinėjamas laikas nuo paraiškos atėjimo laiko į agregatą iki paraiškos siuntimo koreguoti ar paraiškos siuntimo papildyti, nes tie laikai įeina į prieš tai skaičiuojamus agregatus. Šiame agregate nagrinėjama nuo pradinio taško, tai yra paraiškos atėjimo, iki galutinio taško – paraiškos įregistravimo ar paraiškos atmetimo. Po paraiškos patvirtinimo, pagal paraiškos tipą, įmonės informacija yra įregistruojama arba pakeičiami duomenys apie įmonę, arba įmonės informacija išregistruojama.

Agregato *„Registravimas“* naudingumo koeficiento laiko periode $[T_{m-1};T_m]$ apskaičiavimo formulė yra:

$$\eta(T_m) = \frac{rr(T_m) + r(T_m) + rc(T_m) + rs(T_m)}{ra(T_m) + {}^{\#}Q(T_{m-1})};$$

Agregato *„Registravimas“* naudingumo koeficiento laiko periode $[T_{m-1};T_m]$ apskaičiavimo formulė yra:

$$APK(T_m) = 1 - \frac{\sum_{i=1}^{ra(T_m)}(PRAD - T_{m-1}) + \sum_{i=1}^{rr(T_m)}(T_m - PAB) + \sum_{i=1}^{r(T_m)}(T_m - PAB) + \sum_{i=1}^{rc(T_m)}(T_m - PAB) + \sum_{i=1}^{rs(T_m)}(T_m - PAB)}{ra(T_m)\cdot(T_m - T_{m-1}) + {}^{\#}Q(T_{m-1})\cdot(T_m - T_{m-1})};$$

9 lentelė. Agregato „Registravimas“ rezultatai
Šaltinis: sudarė autorius

T_m	$ra(T_m)$	$rr(T_m)$	$r(T_m)$	$rc(T_m)$	$rs(T_m)$	$^{\#}Q(T_m)$	$\eta(T_m)$	$APK(T_m)$	$Avga(T_m)$	$Avgr(T_m)$	$Avg(T_m)$
2010.11.31	0	0	0	0	0	0	0	0	0	0	0
2010.12.01	400	12	240	14	19	115	0,71	0,88	2 min 54 s	22 min 10 s	21 min 15 s
2010.12.02	669	43	398	34	31	278	0,65	0,76	2 h 22 min 19 s	3 h 3 min 56 s	2 h 59 min 53 s
2010.12.03	731	21	496	55	34	403	0,60	0,67	1 h 6 min 4 s	6 h 39 min 29 s	6 h 25 min 57 s
2010.12.04	184	2	62	14	14	495	0,16	0,26	56 s	7 h 42 min 46 s	7 h 28 min 20 s
2010.12.05	190	1	4	10	13	657	0,04	0,17	24 s	3 min 49 s	3 min 8 s
2010.12.06	949	41	661	63	57	784	0,51	0,57	8 h 36 min 23 s	1 d. 1 h 35 min 12 s	1 d. 35 min 42 s
2010.12.07	1167	33	678	126	129	985	0,50	0,57	3 h 34 min 25 s	18 h 4 min 6 s	17 h 23 min 44 s
2010.12.08	1198	19	704	133	247	1080	0,51	0,56	5 h 27 min 43 s	17 h 49 min 2 s	17 h 29 min 33 s
2010.12.09	1355	57	768	123	216	1271	0,48	0,55	5 h 26 min 46 s	17 h 11 min 15 s	16 h 22 min 35 s
2010.12.10	1128	64	915	140	246	1034	0,57	0,53	1 h 48 min 11 s	22 h 21 min 55 s	21 h 1 min 16 s
2010.12.11	202	20	84	12	46	1074	0,13	0,15	10 min 13 s	1 d. 35 min 59 s	19 h 54 min 6 s
2010.12.12	73	9	17	3	8	1110	0,03	0,05	1 min 25 s	1 d. 2 h 47 min 51 s	17 h 31 min 47 s
2010.12.13	1279	58	967	160	216	988	0,59	0,58	11 h 58 min 42 s	1 d. 14 h 8 min 1 s	1 d. 12 h 39 min 13 s
2010.12.14	1355	55	871	151	266	1000	0,57	0,59	8 h 32 min 45 s	1 d. 23 min 35 s	23 h 27 min 6 s
2010.12.15	1420	73	879	160	200	1108	0,54	0,59	6 h 16 min 24 s	22 h 57 min 57 s	21 h 41 min 9 s
2010.12.16	1328	56	896	165	320	999	0,59	0,59	8 h 40 min 22 s	17 h 15 min 13 s	16 h 44 min 56 s
2010.12.17	1319	59	909	125	232	993	0,57	0,59	3 h 4 min 27 s	16 h 43 min 23 s	15 h 53 min 28 s
2010.12.18	510	27	190	36	145	1105	0,26	0,31	1 h 28 min 15 s	22 h 26 min 52 s	19 h 50 min 16 s
2010.12.19	282	13	81	35	94	1164	0,16	0,17	1 d. 9 h 32 min 56 s	1 d. 4 h 44 min 54 s	1 d. 5 h 24 min 44 s
2010.12.20	1653	73	1172	159	245	1168	0,59	0,60	18 h 15 min 10 s	1 d. 6 h 57 min 52 s	1 d. 6 h 13 min 9 s
2010.12.21	1621	81	1004	173	249	1282	0,54	0,58	9 h 1 min 23 s	20 h 17 min 50 s	19 h 27 min 20 s
2010.12.22	1427	64	970	139	232	1304	0,52	0,54	8 h 10 min 31 s	21 h 1 min 30 s	20 h 13 min 47 s
2010.12.23	1528	80	1225	170	261	1096	0,61	0,59	7 h 2 min 31 s	20 h 18 min 33 s	19 h 29 min 45 s
2010.12.24	1353	73	891	135	223	1127	0,54	0,56	1 h 57 min 1 s	16 h 15 min 31 s	15 h 10 min 31 s
2010.12.25	639	15	120	20	49	1562	0,12	0,26	1 d. 9 h 21 min 42 s	21 h 13 min 36 s	22 h 34 min 30 s
2010.12.26	184	25	28	5	14	1674	0,04	0,08	12 min 2 s	1 d. 11 h 30 min 27 s	18 h 51 min 12 s
2010.12.27	1298	58	937	153	273	1551	0,48	0,48	2 d. 4 h 25 min 12 s	1 d. 17 h 12 min 21 s	1 d. 17 h 51 min 35 s
2010.12.28	1769	88	1335	141	330	1426	0,57	0,57	20 h 2 min 58 s	1 d. 3 h 44 min 2 s	1 d. 3 h 15 min 32 s
2010.12.29	1723	94	1066	165	322	1502	0,52	0,56	7 h 18 min 44 s	23 h 9 min 48 s	21 h 52 min 44 s
2010.12.30	1775	106	1171	120	256	1624	0,50	0,55	2 h 21 min 13 s	20 h 37 min 24 s	19 h 6 min 24 s
2010.12.31	1527	86	1239	160	313	1353	0,57	0,54	7 h 15 min 48 s	19 h 47 min 51 s	18 h 59 min 3 s
Iš viso:	**32236**	**1506**	**20978**	**3099**	**5300**	**1075**	**0,44**	**0,49**	**9 h 20 min 8 s**	**22 h 17 min 10 s**	**21 h 25 min 8 s**

Vidutinis paraiškų skaičius agregate per dieną intervale $[T_{m-1};T_m]$:

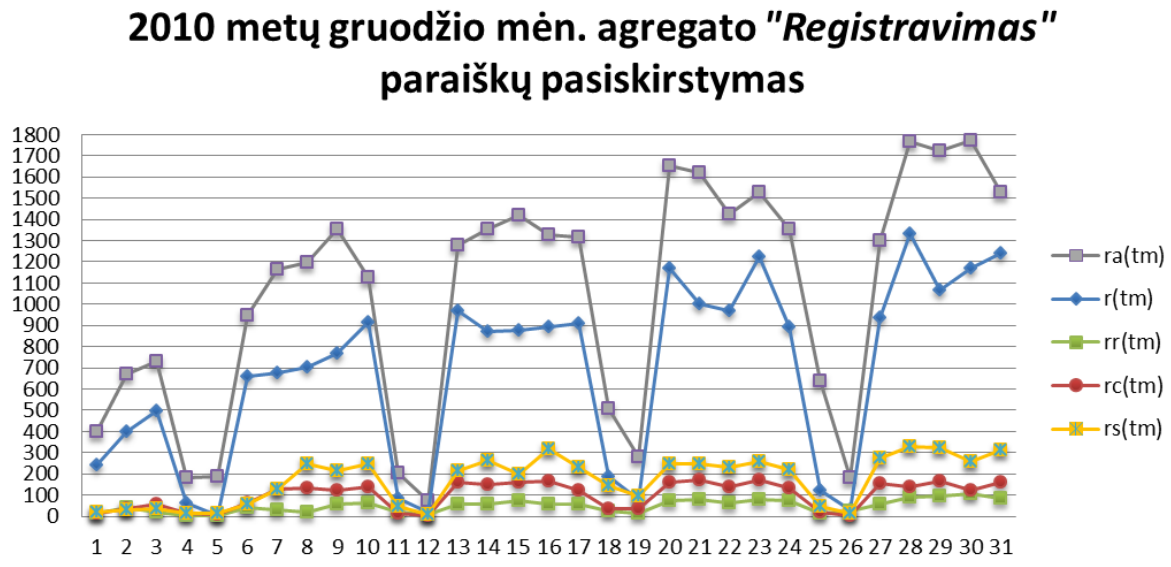
$$\frac{32236}{31} \approx 1039{,}87;$$

Per dieną į agregatą paduodama vidutiniškai 1040 paraiškos, iš kurių per dieną vidutiniškai intervale $[T_{m-1};T_m]$ apdorojama:

$$\frac{1506+20978}{31} = \frac{22484}{31} \approx 725{,}29;$$

Kaip matome, vartotojai, kurie priskirti prie registravimo, per dieną apdoroja 725 paraiškas. Tai žemiausias rezultatas lyginant su kitais agregatais. Vidutinis paraiškos užsibuvimo laikas agregate 21 val., tai yra registravimas baigiamas per 1 parą ir užtrunka ilgiausiai palyginus su prieš tai buvusiais agregatais. Viena iš priežasčių ta, kad registratorius turi patvirtinti paraišką. Duomenys kruopščiai tikrinami prieš priimant galutinį sprendimą. Dar viena iš lėtumo priežasčių tai užklausos įmonės kodo suteikimui į vyriausybinę organizaciją (GDT) prašymas. Vidutinis apkrautumas – 1075 paraiškos agregate per dieną.

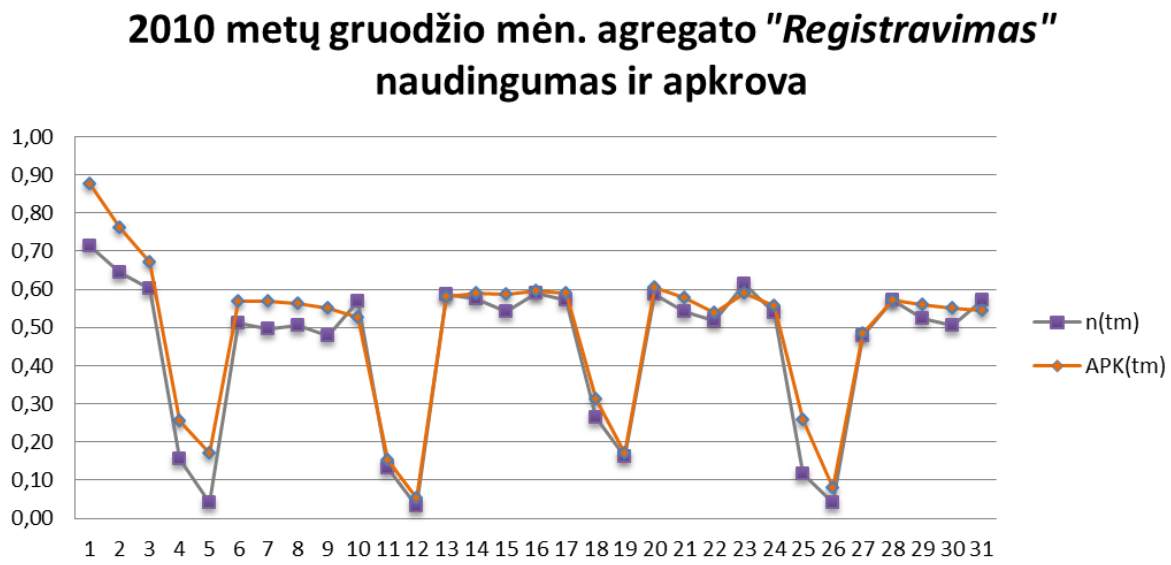
Pagal pateiktus rezultatus nubraižome *„Registravimas“* agregate paraiškų pasiskirstymo grafiką laike. Kaip matome, vykdomų paraiškų skaičius pastoviai kyla. Taip pat pastebime, jog atėjusių paraiškų skaičius yra beveik vienodas pabaigtų vykdyti paraiškų skaičiui. Iš to galima spręsti, kad naudingumo koeficientas bus labai aukštas. Kitas pastebėjimas, kad daugiausia paraiškų apdorojama darbo savaitės pradžioje. Kaip matome iš rezultatų, agregatas savaitgalį visiškai neapkrautas ir registratorius apdoroja nebaigtas savaitės pradžioje paraiškas.



30 pav. Paraiškų pasiskirstymas „Registravimas“ agregate

Šaltinis: sudarė autorius

Pagal pateiktus rezultatus paskaičiavome agregato *„Registravimas“* vidutinį naudingumo koeficientą. Jis yra apie 0,44 o vidutinis agregato apkrautumo koeficientas – 0,49. Žemiau pateikta naudingumo ir apkrovos koeficientų kitimas laiko intervale nuo 2010 metų gruodžio 1 dienos iki gruodžio 31 dienos agregate *„Registravimas“*. Kaip matome, agregato apkrautumas pastoviai mažėja, savaitgaliais pasiekia žemiausią tašką, nes agregate paraiškos vykdomo kuo greičiau. Darbas vyksta net savaitgaliais, kadangi registratorius stengiasi pabaigti nebaigtus savaitės darbus. Iš naudingumo koeficiento matyti, kad darbo dienomis agregatas dirba pastoviai ir stabiliai.

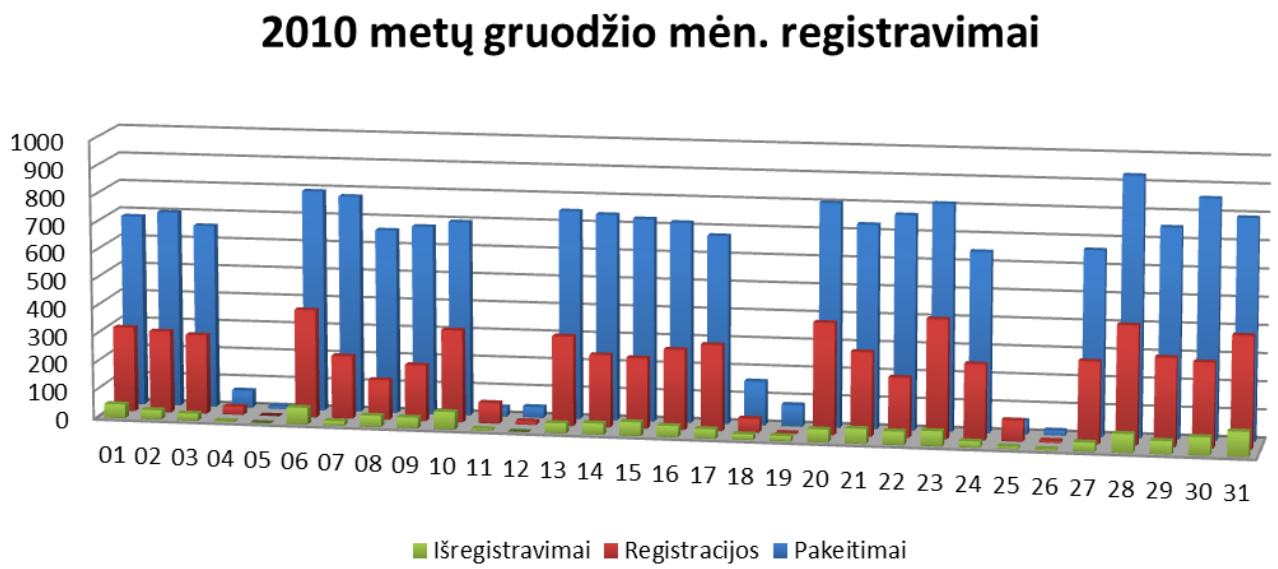


31 pav. agregato „Registravimas“ naudingumo ir apkrovos koeficientų kitimas

Šaltinis: sudarė autorius

7.4. NACIONALINĖS VERSLO REGISTRAVIMO SISTEMOS AGREGATŲ REZULTATŲ IEŠKOTŲ CHARAKTERISTIKŲ APIBENDRINIMAS

Pagal tiriamo mėnesio paraiškų skaičių buvo gauti gruodžio mėnesio rezultatai. Paveiksle žemiau parodyta, kokio tipo paraiškos buvo įregistruotos per gruodžio mėnesį. Kaip matome, pirmauja įmonės duomenų pakeitimo paraiškos tipas, o nauji įmonės įregistravimai užima antrą vietą. Paskutinė vieta – tai įmonės išregistravimas iš NBRS. Bendrai per mėnesį patvirtintos 35135 paraiškos, iš kurių 25739 pakeisti įmonės duomenis, 8292 paraiškos naujai įmonei įregistruoti ir 1104 paraiškos įmonei išregistruoti iš NBRS.



32 pav. 2010 metų gruodžio mėnesio paraiškų registravimo tipai
Šaltinis: sudarė autorius

Pagal gautus rezultatus žemiau pateikta lentelė visų ieškomų charakteristikų rezultatų. Kaip matome, naudingiausiai dirba *„Žurnalizavimas“* agregatas ir jo apkrautumas mažiausias, tai yra paraiškos šiame agregate vykdomos gana greitai ir ilgai agregate neužsibūna. Mažiausias naudingumas *„Registravimas“* agregate, nors apkrautumas ir nėra labai didelis. To priežastis gali būti, kad daug paraiškų registratorius siunčia papildyti į *„Žurnalizavimas“* agregatą arba pataisyti į *„Vykdymas“* agregatą. O didžiausias apkrautumas *„Vykdymas“* agregate, nes visas paraiškos užpildymas ir paraiškos duomenų koregavimas vyksta šiame agregate ir tai trunka gana ilgai. Apskaičiavę agregatų laikų aptarnavimo vidurkį be bendro agregato *„Pareiškėjas“* gauname laiką, panašų į agregato *„Pareiškėjas“* vidutinį aptarnavimo laiką. Tai leidžia teigti, kad mūsų skaičiavimai teisingi. Nors agregato *„Pareiškėjas“* naudingumo koeficientas ir nedidelis, o apkrautumo koeficientas aukštas, agregato vidutinis aptarnavimo laikas atitinka užduotą kriterijų, kad paraiškos aptarnavimo laikas turėtų būti 7 darbo dienos.

10 lentelė. Bendri agregatų rezultatai
Šaltinis: sudarė autorius

Agregatas	$^{\#}Q_{vid}$	η_{vid}	APK_{vid}	Avg_{vid}
„Žurnalizavimas“	231	0,72	0,28	46 min 52 s
„Vykdymas“	374	0,69	0,70	7 h 8 min 30 s
„Registravimas“	1075	0,44	0,49	21 h 25 min 8 s
„Pareiškėjas“	5281	0,12	0,84	4 d. 15 s

IŠVADOS

Šiame magistriniame darbe buvo išsikelti šie tyrimo tikslai:

1. Verslo procesų modelius, specifikuotus grafine UML (angl. Unified Modeling Language) notacija, transformuoti į formalius PLA (angl. Piece-Linear Aggregates) modelius, naudojantis sudarytomis transformavimo taisyklėmis ir pasirinktomis technologijomis;

2. Pasinaudojus verslo procesų modelių transformacija NBRS (angl. National Business Registration Sytem) konceptualaus modelio sukūrimas;

3. Pasinaudojus PLA (angl. Piece-Linear Aggregates) modeliavimo metodu įvertinti funkcines sistemos charakteristikas, kurios parodo NBRS darbą;

Užsibrėžti tikslai buvo pasiekti. Tai yra realizuotas verslo procesų modelių, specifikuotų grafine UML notacija, transformacija į formalius PLA modelius. Tyrimas parodė, kad nesunku pasinaudojus transformacijos galutinio modelio generavimu iš pradinio modelio pagal transformacijos aprašą, pereiti nuo vieno modelio prie kito.

Sukurtas konceptualus modelis, nagrinėjantis keturis agregatus: *„Pareiškėjas“, „Žurnalizavimas“, „Vykdymas“* ir *„Registravimas“*. Kiekviename agregate buvo apskaičiuotos norimos surasti funkcinės sistemos charakteristikos, kurios parodė visos sistemos bei atskirų sistemos dalių paraiškos apdorojimo darbo laiką, agregato naudingumą ir apkrautumą. Buvo įrodyta, kad pasinaudojus PLA (angl. Piece-Linear Aggregates) modeliavimo metodu nesunku suskaičiuoti sistemos charakteristikas ir pasinaudojus šiais skaičiavimais nesudėtinga rasti sistemos kritinius taškus.

Pagrindinis NBRS (angl. National Business Registration Sytem) reikalavimas buvo, kad paraiškos apdorojimo darbo laikas nebūtų ilgesnis nei septynios dienos. Galutiniai skaičiavimai parodė, kad sistema dirba sparčiau nei tikėtasi. Tai yra paraiškos apdorojimas sistemoje vidutiniškai užtrunka keturias dienas.

Pasinaudojus PLA (angl. Piece-Linear Aggregates) modeliavimo metodu galima analizuoti sistemos darbą, pamatyti kritinius taškus, kurie sistemos darbą lėtina. To negalima padaryti UML (angl. Unified Modeling Language) ar BPMN (angl. Business Process Modelling Notation) notacijomis.

LITERATŪRA

[1] Štuikys V., Damaševičius R.. Modelių ir programų abstrakčios transformacijos: mokomoji knyga. Vilnius, Lithuania, 2008. 227 p.

[2] Kleppe A., Warmer J., Bast W. MDA Explained – The Model-Driven Architecture: Practice and Promise. Addison-Wesley Professional, 2003. 192 p.

[3] Koch N. Transformation Techniques in the Model Driven Development Process of UWE. Workshop at ICWE‘06, Palo Alto, USA, 2006. p. 11-14.

[4] Mens T., Van Gorp P. A Taxonomy of Model Transformation. In Proceedings of International Workshop on Graph and Model Transformation, Tallinn, Estonia, 2005. p. 125-142.

[5] Gogolla M., Lindow A., Richters M., Ziemann P. Metamodel Transformation of Data Models. Proceedings of UML‘2002 Workshop in Software Model Engineering. Technical report, Nantes, France, 2002. 8 p.

[6] Lano K. Design for Change: Advanced Systems Design with Java, UML and MDA. Butterworth-Heinemann, 2005. 416 p.

[7] Sendal S., Kozaczynski W. Model Transformation: the Heart and Soul of Model Driven Software Development. Software, IEEE. 2003. p. 42-45.

[8] Budinsky F., Steinberg D., Ellersick R., Grose T. J. Eclipse Modeling Framework, 1st Edition. Addison-Wesley Professional, 2004. 720 p.

[9] Kurtev I. State of the Art of QVT: A Model Transformation Language Standard. Third International Symposium of Applications of Graph Transformations with Industrial Relevance. Kassel, Germany, 2007. p. 377-393.

[10] Jouault F. Kurtev I. On the Architectural Alignment of ATL and QVT. Proceedings of ACM Symposium on Applied Computing, Model Transformation Track. Dijon, France, 2006. p. 1188-1195.

[11] Slaninaitė L. Verslo procesų, aprašytų BPMN notacija, transformavimas į PLA modelius. Magistro tezės, Kaunas, Lithuania, 2006.

[12] Pranevičius H. Kompiuterinių tinklų protokolų formalusis specifikavimas ir analizė: agregatinis metodas. Technologija, 2005. p. 29-42.

[13] Vitolins A., Kalnins V. Use of UML and Model Transformations for Workflow Process Definitions. Technika, Vilnius, Lithuania, 2006. p. 3-15.

[14] White, S. A. Process Modeling Notations and Workflow Patterns. IBM Corporation, 2006. 24 p.

[15] Watson A. OMG's New Modeling Specifications, European Conference on Modelling Foundations and Applications. Nuremberg, Germany, 2005. Keynote speech.

[16] Wohed P., Van Der Aalst W. M. P., Dumas M., Ter Hofstede A. H. M., Russell N. Pattern-based Analysis of BPMN – An extensive evaluation of the Control-flow, the Data and the Resource Perspectives. Springer-Verlag Berlin Heidelberg, 2005. p. 63-78.

[17] Packevičius Š., Kazla A., Pranevičius H. Extension of PLA Specification for Dynamic System Formalization. Proceedings of Information Technology and Control. Kaunas, Lithuania, 2006. p. 235-242.

[18] White S. A., Miers D. BPMN Modeling and Reference Guide. Future Strategies Inc., Lighthouse Pt., Florida, USA, 2008. 226 p.

[19] Kulbaitytė L. BPMN notacija aprašytų verslo procesų modelių transformavimas į PLA modelius. Magistro tezės, Kaunas, Lithuania, 2008.

[20] Dijkman R. M., Dumas M., Ouyangl C. Formal Semantics and Analysis of BPMN Process Models using Petri Nets. Technical report, Brisbane, Australia, 2007.

[21] Rumbaugh J., Jacobson I., Booch G. The Unified Modeling Language Reference Manual. Addison-Wesley Longman, 1999. 568 p.

The use of PLA modeling method for software performance analysis

SUMMARY

Research area – applying formal methods in describing and modeling of the complex systems functionalities together with verification and validation of designed specifications. The research is based on the use of Piece-Linear Aggregates, which allows creating modules for analyses of the system functionality and system correctness.

The purpose of this work is transformation of UML activity model into PLA model in order to assess the characteristics of system functionality. These characteristics can show how system is operating and which parts of the system can be improved.

PLA – Piece-Linear Aggregates formalism is used for modeling various systems. PLA is universal formal modeling language. In this specification process models are expressed in textual and mathematical expressions. The main idea of this approach - divide the system into piece linear aggregates each of which describe the use of control sequences.

Using this method the formalized business processes are displayed as a set of interacting aggregates. The aggregate is an object, defined by a set of states Z, incoming signals X and outgoing signals Y.

The final result of the research is the assessment the National Business Registration System in the Vietnam. Assessment allows to evaluate characteristics of the system functionality using PLA modeling method and identify the weakest point in the system in order to propose possible improvement and optimization methods.

SANTRUMPŲ IR TERMINŲ ŽODYNAS

[ATL] Elipse project M2M: ALT Documentation. http://www.eclipse.org/m2m/atl/doc/

[BMI] Business Modeling & Integration Domain Task Force. http://bmi.omg.org/

[BPMN] OMG Document: BPMN Specification versions 1.0, 1.2, 2.0. http://www.omg.org/spec/BPMN/1.0/, http://www.omg.org/spec/BPMN/1.2/, http://www.omg.org/spec/BPMN/2.0/, http://www.bpmn.org/documents.htm/

[BRO] Provincial Business Registration Office

[DB] Database

[EDA] Enterprise Development Agency

[EE] Enterprise Edition

[GDT] General Department of Taxation of the Socialist Republic of Vietnam

[GSO] General Statistics Office of the Socialist Republic of Vietnam

[HW] Hardware

[ISP] Internet Service Provider

[LAN] Local Area Network

[MD] MagicDraw architecture tools: http://www.magicdraw.com/magicdraw_resources/

[MDA] OMG Document: MDA Guide version 1.0.1. http://www.omg.org/cgi-bin/doc?omg/03-06-01/

[MPI] Ministry of Planning and Investment of the Socialist Republic of Vietnam

[MPS] Ministry of Public Security of the Socialist Republic of Vietnam

[MOF] OMG Document: MOF Specification version 2.0. http://www.omg.org/mof/

[MS] Microsoft Company

[NBRS] National Business Registration System

[QVT] OMG Document: QVT Specification version 1.0. http://www.omg.org/spec/QVT/1.0/

[RFP] OMG Document: Business Process Definition Metamodel RFP. http://xml.coverpages.org/OMG-03-01-06.pdf

[UML] OMG Document: UML Infrastructure Specification version 2.1.2. http://www.omg.org/spec/UML/2.1.2/

[UNIDO] United nations industrial development organization http://www.unido.org

[VPN] Virtual Private Network

[XSLT] W3C Document: W3C Recommendations for XSL Transformations (XSLT) version 1.0. http://www.w3.org/tr/xslt/