KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

VERSLO INFORMATIKOS KATEDRA

Živilė Janušauskaitė

Verslo sistemų REA modelio analizė, panaudojant agregatinę schemą ir loginį programavimą

Magistro darbas

Darbo vadovė

doc. R. Misevičienė

Kaunas, 2006

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKOS KATEDRA

Živilė Janušauskaitė

Verslo sistemų modelio analizė, panaudojant agregatinę schemą ir loginį programavimą

Magistro darbas

Kalbos konsultantė
Lietuvių k. katedros
dėst. I. Mickienė
2006-05-22

Vadovė
doc. R. Misevičienė
2006-05-29

Recenzentas
doc. R. Butleris
2006-05-29

Atliko
IFM/0-1 gr. stud.
Živilė Janušauskaitė
2006-05-22

Kaunas, 2006

Turinys

Analysis business systems REA model using aggregate schema and logic programming

Summary

This work presents business process analysis methodology which consists of presentation of the business processes created on the ground of the Resource - Event - Agent model by means of Piece–Linear Aggregate approach. The aggregate specification is analyzing using first order predicate logic while checking correctness by resolution method using logic programming based language Prolog. The work is concluded with concrete example of analysis of Resource - Event - Agent model based business process using the aggregate approach.

The novelty of this work

PLA (Piece-Linear Aggregate) model and the software tools, created on the ground of PLA (Piece-Linear Aggregate), are used the first time for business processes analysis that is defined using REA formalism. The use of such integrated models allows performing the automated analysis of general and individual properties (completeness, deadlock freeness, termination or cyclic behavior, boundedness) of defined business processes.

The main results are achieved:

- The methodology, that consists of presentation of the business processes created on the ground of the Resource - Event - Agent model by means of Piece–Linear Aggregate approach.
- Verification and validation of general and individual properties by using PLA and PROLOG language approach designed system that executes the analysis of aggregate specification.
- Implementing internal accounting controls as constraints in relational algebra, SQL and PROLOG language.
- Concrete example of analysis of Resource - Event - Agent model based business process using the aggregate approach.

Įvadas

Pagrindinė problema, su kuria dažniausiai susiduria apskaitos informacijos sistemų kūrėjai – kaip sukurti, tokią informacijos sistemą, kuri pilnai ir patikimai aprūpintų apskaitos informacija įvairius jos vartotojus valdančius įmonės ūkinę veiklą bei vykdančius jos kontrolę. Dauguma verslo sistemų yra labai didelės ir sudėtingos, kad sistemų kūrėjai visapusiškai galėtų įvertinti jų veiklą. Todėl būtina sudaryti įmonės veiklą atspindinčius modelius, kurie supaprastintų realios įmonės veiklos aprašymą.

Vienas tokių modelių yra **R**esursų-**Į**vykių-**A**gentų (REA) modelis, kuris taikomas įmonės verslo procesų aprašymui. Šis modelis naudojamas atvirų sistemų sąveikavimo standarto ISO/IEC 14662 funkciniame lygmenyje. Minėtą standartą patvirtino Tarptautinė prekybos vystymo ir elektroninio verslo organizacija UN/CEFACT 2003 metais. Literatūroje REA modelis yra dažnai sutinkamas ir yra svarbus informacijos sistemų semantiniam modeliavimui, tačiau jų teisingumo analizei skirta gana nedaug dėmesio.

Magistriniame darbe naudojamas vienas iš sudėtingų sistemų aprašymo formalizmų – atkarpomis tiesinių agregatų modelis – PLA (piece-linear aggregate). Agregatinio metodo panaudojimas suteikia priemones ne tik formalių specifikacijų sudarymui, bet ir jų modeliavimui bei teisingumo analizei.

Šiame darbe pasiūlyta metodika, pagal kurią siūloma, REA modelio pagrindu sudarytų, verslo procesų analizei, panaudoti programines priemones, sukurtas agregatinio formalizmo pagrindu. Analizės metu verslo procesų REA modelis užrašomas panaudojant agregatinį formalizmą, o sudarytos specifikacijos bendrosios savybės tikrinamos naudojant PLA analizės priemones.

Darbo tikslas:

Sudaryti verslo procesų analizės metodiką, kuri atliktų REA (Resource-Event-Agent) modelio pagrindu sudarytų verslo procesų analizę, panaudojant programines priemones, sukurtas PLA (piece-linear aggregate). formalizmo pagrindu. Sudaryta agregatinė specifikacija analizuojama, taikant pirmos eilės predikatų logiką. Jos užrašymo teisingumas tikrinamas, naudojant loginio programavimo kalbą – PROLOG.

Darbe sprendžiami uždaviniai:

- Sukurti metodiką, kuri realizuotų sąryšį tarp REA (Resource-Event-Agent) ir PLA (piece-linear aggregate) modelių.
- Atlikti bendrųjų ir individualiųjų savybių analizę panaudojant PLA (Piece-Linear Aggregate) ir PROLOG kalbos pagrindu sukurtą sistemą, atliekančią agregatinių specifikacijų analizę.

- Individualių savybių užrašymui taikyti vidaus apskaitos kontrolės apribojimus, naudojant reliacinę algebrą, SQL ir PROLOG.
- Iliustruojant metodiką, užrašyti pardavimo ciklo REA (Resource-Event-Agent) modelio pavyzdį, taikant PLA (Piece-Linear Aggregate) struktūrą ir atlikti pavyzdžio agregatinės specifikacijos analizę.

Darbo turinys.

Kadangi darbe pateikta verslo proceso analizės metodika, sudaryta iš verslo procesų aprašymo, taikant REA (Resource-Event-Agent) modelį ir PLA (piece-linear aggregate) formalizmą, *pirmame skyriuje* pateikiama verslo srities ontologija dviem lygiais (vertės sistemos lygmuo ir pridėtinės vertės lygmuo). Parodomi REA (Resource-Event-Agent) modelio sudarymo principai, vyraujantys ryšiai bei pagrindiniai objektai. Šiame skyriuje taip pat pateikiamas trumpas aprašymas vieno iš sudėtingų sistemų aprašymo formalizmų – PLA (piece-linear aggregate) – atkarpomis tiesinių agregatų modelis. Toliau aprašoma, kaip agregatinės specifikacijos bendrosios savybės analizuojamos PROLOG kalbos pagrindu sukurtoje sistemoje. Šios sistemos principai bei pirmos eilės predikatų logikos sakiniai, naudojami agregatinės specifikavimui, išdėstyti *pirmame skyriuje*.

Verslo procesų analizės metodika, susidedanti iš REA (Resource-Event-Agent) modelio pagrindu sudarytų verslo procesų aprašymo, panaudojant PLA (Piece-Linear Aggregate) formalizmą, pateikta *antrame skyriuje*. Čia apibrėžta verslo procesų modeliavimo ir analizės esmė, parodyti sąryšiai tarp REA (Resource-Event-Agent) ir PLA (Piece-Linear Aggregate) modelių bei aprašyta verslo procesų modelio transformacija į PLA.

Individualių savybių užrašymui taikomi apskaitos sistemų vidaus kontrolės apribojimai. Specifikacijos individualiųjų savybių užrašymas, taikant reliacinę algebrą, SQL ir PROLOG pateikta *trečiame skyriuje*.

Iliustruojant metodiką, *ketvirtame skyriuje* užrašytas pardavimo ciklo REA (Resource-Event-Agent) modelio pavyzdys, taikant PLA (Piece-Linear Aggregate) struktūrą ir atlikta sistemos agregatinės specifikacijos analizė.

Darbo pabaigoje pateikti rezultatai ir išvados.

Prieduose pateiktas programos kodas, 2005-2006 metų respublikinėse konferencijose „*Informacinės technologijos*“ pristatyti pranešimai.

Darbo naujumas:

Verslo procesų, kuriems aprašyti naudojamas REA formalizmas, analizei pirmą kartą panaudotas PLA modelis bei jo pagrindu sukurtos programinės priemonės. Toks integruotas modelių panaudojimas leidžia atlikti aprašomų verslo procesų automatizuotą

bendrųjų (pilnumas, aklaviečių nebuvimas, apribojimų tenkinimas, pertekliškumas) ir individualiųjų savybių tikrinimą.

Darbo rezultatai:

- Sukurta metodika, kuri realizuoja sąryšį tarp REA (Resource-Event-Agent) ir PLA (piece-linear aggregate) modelių.
- Atlikta bendrųjų ir individualiųjų savybių analizė, naudojant PLA (Piece-Linear Aggregate) ir PROLOG kalbos pagrindu sukurtą sistemą, atliekančią agregatinių specifikacijų analizę.
- Individualių savybių užrašymui pritaikyti vidaus apskaitos kontrolės apribojimai, naudojant reliacinę algebrą, SQL ir PROLOG.
- Iliustruojant metodiką, užrašytas pardavimo ciklo REA (Resource-Event-Agent) modelis, taikant PLA (Piece-Linear Aggregate) struktūrą ir atlikta sistemos agregatinės specifikacijos analizę.

Pagrindiniai darbo rezultatai buvo pristatyti šiose respublikinėse konferencijose:

- Janušauskaitė Ž., Muralis L., Misevičienė R. „*Įmonės verslo sistemos ontologija*“. „*Informacinės technologijos*“, Kaunas, 2005, sausio 26d..
- Pranevičius H., Misevičienė R., Janušauskaitė Ž.. „*Verslo procesų, aprašytų REA (Resource-Event-Agent) metodu, analizė panaudojant agregatinį formalizmą*“. „*Informacinės technologijos*“, Kaunas, 2006, sausio 26d..

1. Verslo procesų modelių ir jų teisingumo analizės metodų apžvalga

1.1. Įvadas

Pagrindinė problema, su kuria dažniausiai susiduria apskaitos informacijos sistemų kūrėjai – kaip sukurti, tokią informacijos sistemą, kuri pilnai ir patikimai aprūpintų apskaitos informacija įvairius jos vartotojus valdančius įmonės ūkinę veiklą bei vykdančius jos kontrolę. Dauguma verslo sistemų yra labai didelės ir sudėtingos, kad sistemų kūrėjai visapusiškai galėtų įvertinti jų veiklą. Todėl būtina sudaryti įmonės veiklą atspindinčius modelius, kurie supaprastintų realios įmonės veiklos aprašymą.

Nauji verslo projektai remiasi e-verslo informacinių sistemų kūrimu. Čia egzistuoja du modelių tipai: e-verslo modeliai ir procesų modeliai. E-verslo modelis, susijęs su resursų vertės mainais tarp verslo partnerių. Procesų modelyje dėmesys skiriamas operaciniams ir procedūriniams aspektams, realizuojant procesus. Nors šie du modeliai yra panašūs, turi ir skirtumų.

E-verslo modeliai yra aprašomi naudojant standartinius modeliavimo metodus, pvz., UML [7], IDRF [7].

Berghpltz‘as [3] parodė, kaip galima du (verslo ir proceso) modelius integruoti. Integravimas naudingas tuo, kad apjungiami skirtingi vartotojo požiūriai ir vidinių ryšių modeliavimo supratimas. Teorinis pagrindas remiasi REA (Resource-Event-Agent) struktūra [12]. Ši struktūra buvo sukurta, siekiant pavaizduoti ekonominius mainus. REA (Resource-Event-Agent) buvo suprantamas kaip apskaitos sistemoms taikoma struktūra. Vėliau buvo išvystytas į verslo srities ontologiją. Nors REA (Resource-Event-Agent) modelį buvo pasiūlyta taikyti, modeliuojant verslo sistemas, iš tikrųjų atlikti tik keli REA (Resource-Event-Agent) analizės tyrimai [6].

Verslo proceso modeliavimo ir imitavimo integravimas teikia didelę reikšmę verslo procesų analizei [2]. Naudojant imitavimą, modeliuojant ir analizuojant verslo procesus, galima mažinti įkainius arba didinti našumą ir sėkmės galimybę, pakartotinai atkuriant verslo procesų projektus.

Šiame darbe taikoma integruota verslo procesų modeliavimo ir imitavimo metodika – PLA (Piece-Linear Aggregate) – skirta verslo procesų imitavimui, o REA (Resource-Event-Agent) naudojama verslo procesų modeliams sudaryti. PLA (Piece-Linear Aggregate) plačiai taikomas verslo procesų specifikacijų kūrime, modeliavime, imitavime ir analizėje [19]. Remiantis PLA (Piece-Linear Aggregate) principais buvo sukurti automatizuoti analizės įrankiai [1].

Vienoje iš tokių priemonių naudojama pirmos eilės predikatų logika, sudarant specifikacijas, bei loginio programavimo kalba – PROLOG, konstruojanti vykdomą aprašą agregatinei specifikacijos analizei (Pranevičius, 2003). Taikant aprašytus analizės būdus, naudojamas pasiekiamų būsenų metodas. Ši analizė paremta globalios būsenos sąvoka. Taikant šį metodą sudaromas pasiekiamumo grafas ir atliekama bendrųjų ir individualiųjų savybių analizė.

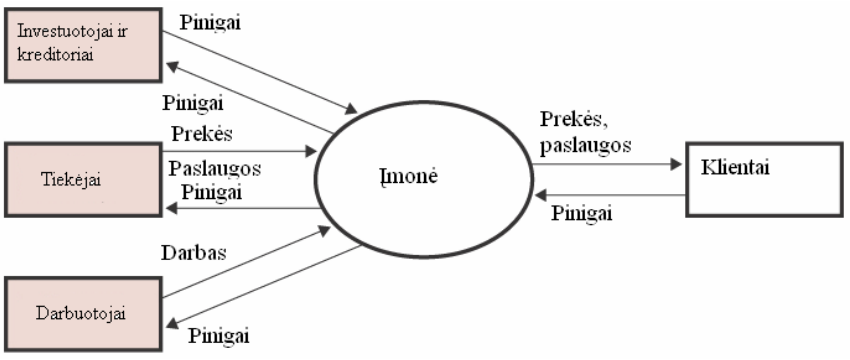
1.2. Verslo srities procesų ontologija

Šiame skyriuje pateikiama konceptuali REA (Resource-Event-Agent) modelio struktūra, aptariama REA (Resource-Event-Agent) ontologija dviem lygiais (value system, value chain). Parodomi REA (Resource-Event-Agent) modelio sudarymo principai, vyraujantys ryšiai bei pagrindiniai objektai.

Verslo srities ontologija apibrėžia objektus, egzistuojančius verslo pasaulyje. Ontologija apima komunikavimą, pasidalijimą ir informacijos pakartotinį panaudojimą. Šiuolaikinėms informacijos sistemoms ir e-versle šios sąvokos yra labai svarbios.

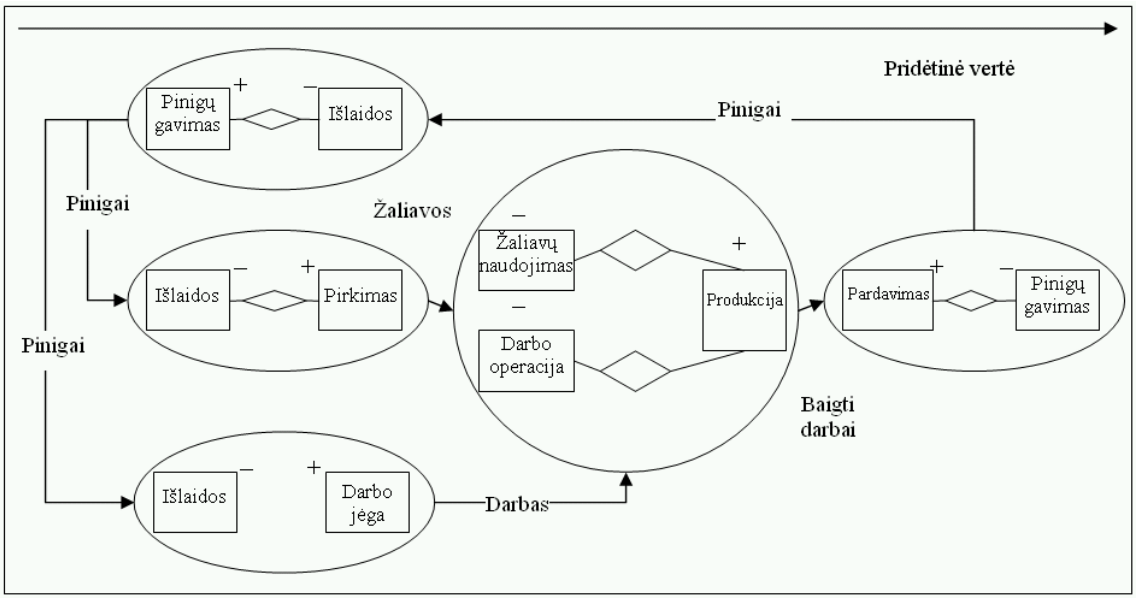
Šiame darbe konceptuali REA (Resource-Event-Agent) modelio [12], [15] struktūra taikoma kaip verslo srities ontologija. REA (Resource-Event-Agent) ontologijoje [6] pasiūlytas požiūris modeliuoti įmones, taikant vertės sistemos (value system), pridėtinės vertės (value chain), bei verslo proceso lygmenis.

Vertės sistemos modelyje analizuojami įmonės ryšiai su vidaus ir išorės partneriais [10]. Kitaip tariant parodomas įmonės resursų vertės kitimas vidinių ir išorinių partnerių atžvilgiu (žr. 1 pav.)



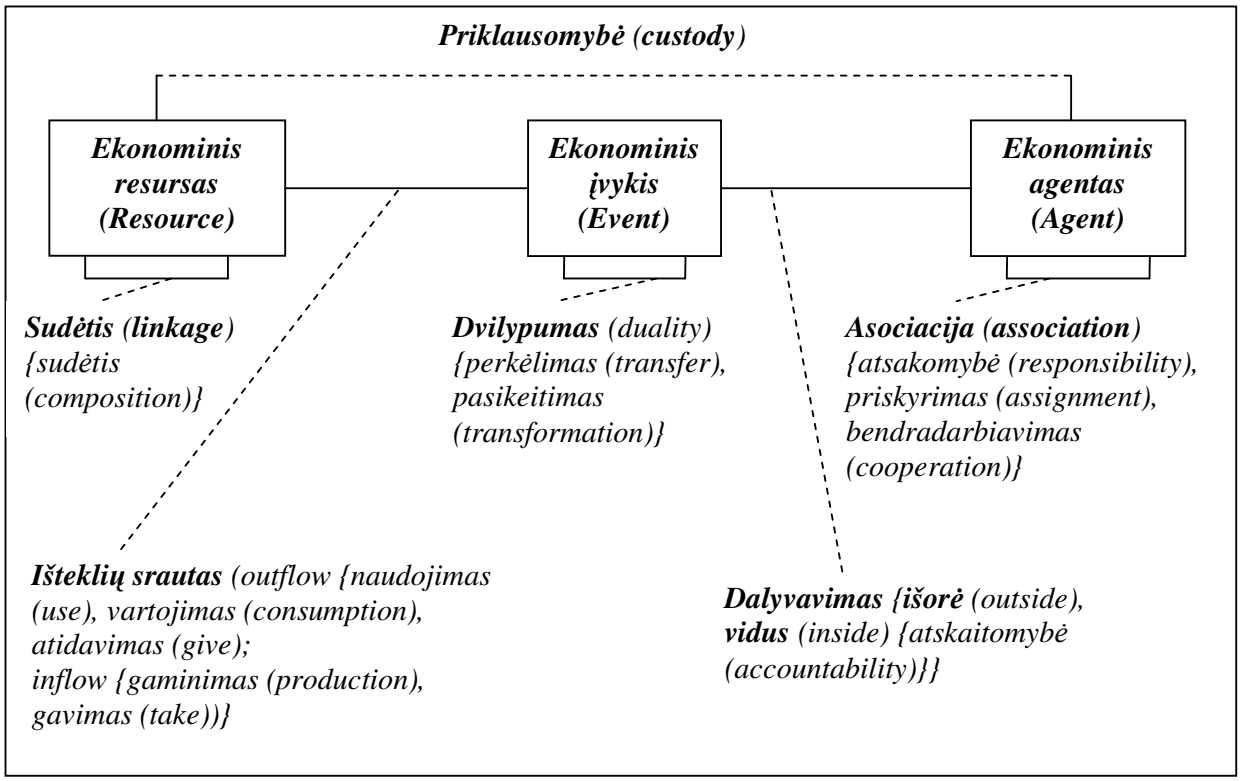
1 pav. Vertės sistemos modelio struktūra [8]

Pridėtinės vertės modelis parodo, kaip vienos rūšies resursai virsta kitais. (žr. 2 pav.).



2 pav. Pridėtinės vertės modelis, taikant REA (Resource-Event-Agent) [8]

Verslo proceso lygį sudaro trys pagrindiniai komponentai (ekonominis resursas, ekonominis įvykis ir ekonominiai agentai) ir ryšiai tarp jų (išteklių srautas, dvilypumas, dalyvavimas, asociacija, priklausomybė ir sudėtis) (žr. 3 pav.).



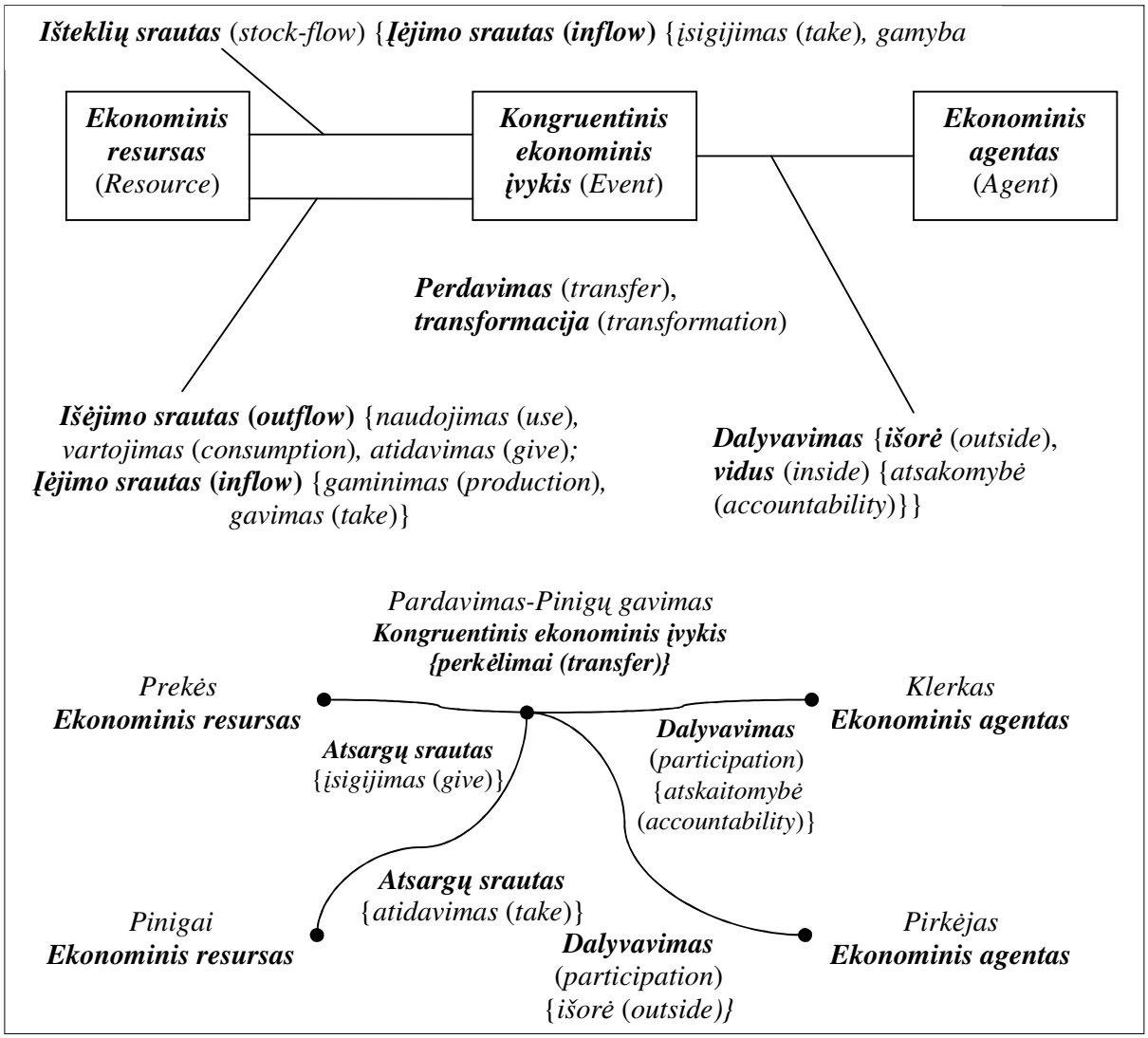
3 pav. Verslo proceso lygmuo REA (Resource-Event-Agent) modelyje [8]

Ekonominis įvykis yra verslo informacijos sistemos pagrindinis elementas, lemiantis įėjimo ir išėjimo resursų srautus. Įėjimo srauto resurso pavyzdžiu galėtų būti pinigai, gauti už parduotas prekes, kurios yra išėjimo srauto resurso pavyzdys.

Dvilypumo ryšys egzistuoja tarp įvykių ir sąlygoja resursų įėjimo ir išėjimo srautus. Kartais tokie mainai gali būti vadinami **kongruentiniais** (congruent exchange). Skirtumas tarp dvilypumo ir kongruentinio ekonominių įvykių yra tas, kad antruoju atveju įvykiai vyksta tuo pat metu (žr. 3 pav.).

Fisher (1906) išskyrė du dvilypumo tipus: **pasikeitimas** (transformation) ir **perkėlimas** (transfer). Galima sakyti, kad įvyko pasikeitimas (transformation), kai tam tikras resursas proceso metu virsta kitu, pvz. iš molio nulipdomi ąsočiai. Šis tipas dažniausiai naudojamas gamyboje. Perkėlimo (transfer) sąvoka aiškinama resurso atidavimu išorei (žr. 3 pav.). Pavyzdžiui, nulipdyti ąsočiai parduodami, t.y. tampa pirkėjo nuosavybe.

Ekonominis resursas – tai statinis elementas, aprašantis tam tikrų objektų kiekį. Resursus lemia ekonominiai įvykiai. Ryšys tarp ekonominių resursų ir įvykių vadinamas **išteklių srautu** (Stock-Flow). 4 pav. išskiriami penki išteklių srautų tipai: **naudojimas** (use), **vartojimas** (consumptin), **atidavimas** (give), **gamyba** (production) ir **gavimas** (take). Resursas yra sunaudojamas arba suvartojamas įvykus pasikeitimui (transformation), t.y tam tikros žaliavos gamybos (production) metu virsta produkcija. Sunaudojimas (use) nuo suvartojimo (consume) skiriasi tuo, kad pirmuoju atveju resursas pakeičia savo formą, o antruoju – sumažėja jo kiekis. Pardavimo metu atiduodame (give) gaminius ir gauname (take) pinigus. Tas pats resursas gali dalyvauti keliuose ryšiuose, pvz. mašina pirmiausia yra įsigijama (take), po to ji naudojama (use) gamyboje (production) ir, pagaliau, gali būti parduota (give).



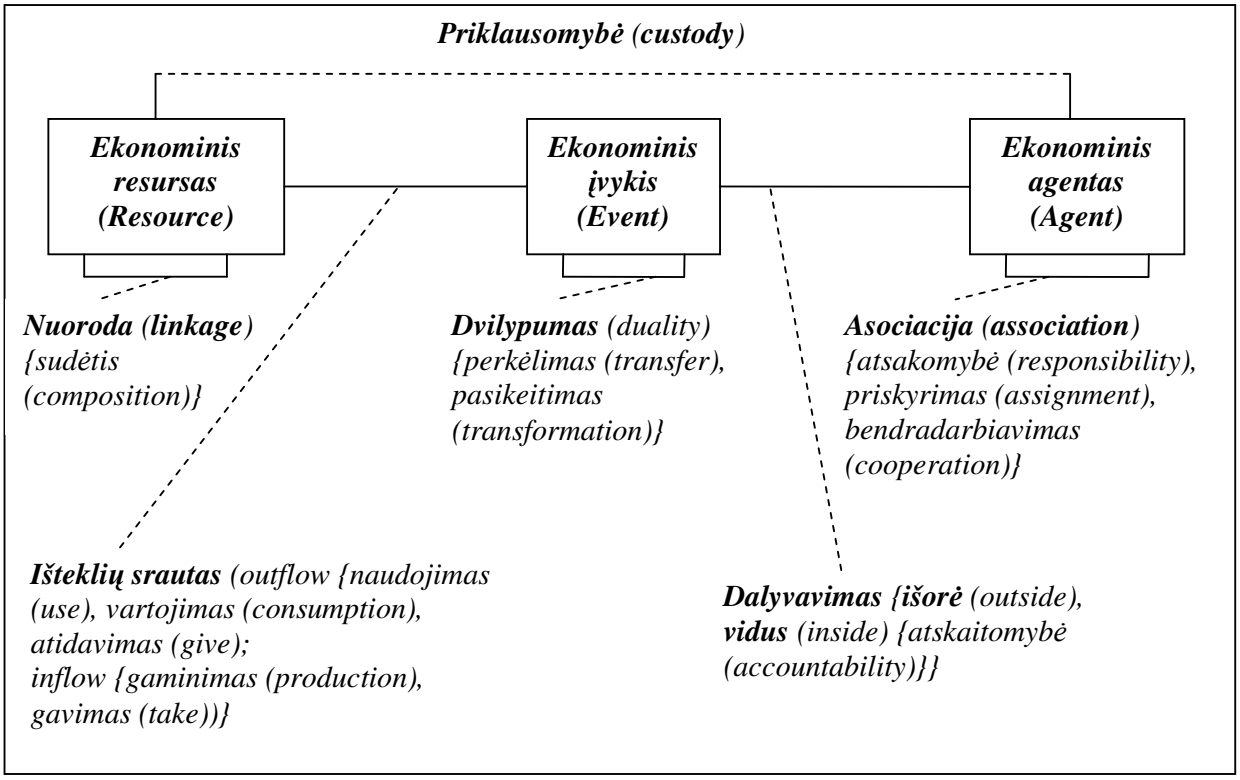
4 pav. Kongruentiniai ekonominiai mainai [8]

Agentas – tai trečiasis REA (Resource-Event-Agent) pagrindinis elementas, nurodantis asmenis arba organizacijas, dalyvaujančias ekonominiuose įvykiuose. Skiriami **vidiniai** (inside) ir **išoriniai** (outside) agentai. Vidiniu gali būti organizacijos, įmonės darbuotojas. Išoriniu dažniausiai vadinamas organizacijos klientas. Ryšys tarp vidinių/išorinių agentų ir ekonominių įvykių vadinamas **dalyvavimu** (participation)**. Atskaitomybė** (accountability) – dalyvavimo (participation) ryšio tipas, kuris nurodo vidinio agento pareigas įmonėje, ar organizacijoje, pvz. sandėlininkas yra atskaitingas už sandėlyje esančias prekes ar atsargas.

Ryšys tarp įvykių ir agentų vadinamas **valdymu** (control). Kitaip tariant, ekonominis įvykis kontroliuoja ekonominį įvykį, pvz., pardavimą ir atitinkamus išeinančio srauto resursus, pvz., prekes.

Bendravimas (association), sąsaja (linkage), priklausomybė (custody)

5 pav. pavaizduoti papildomi ryšiai, kurie sukuria priklausomybes tarp agentų (**bendravimas** (association)), resursų (**sąsaja** (linkage)) ir tarp resursų bei agentų (**priklausomybė** custody)).



5 pav. Nuoroda, priklausomybė, asociacija [8]

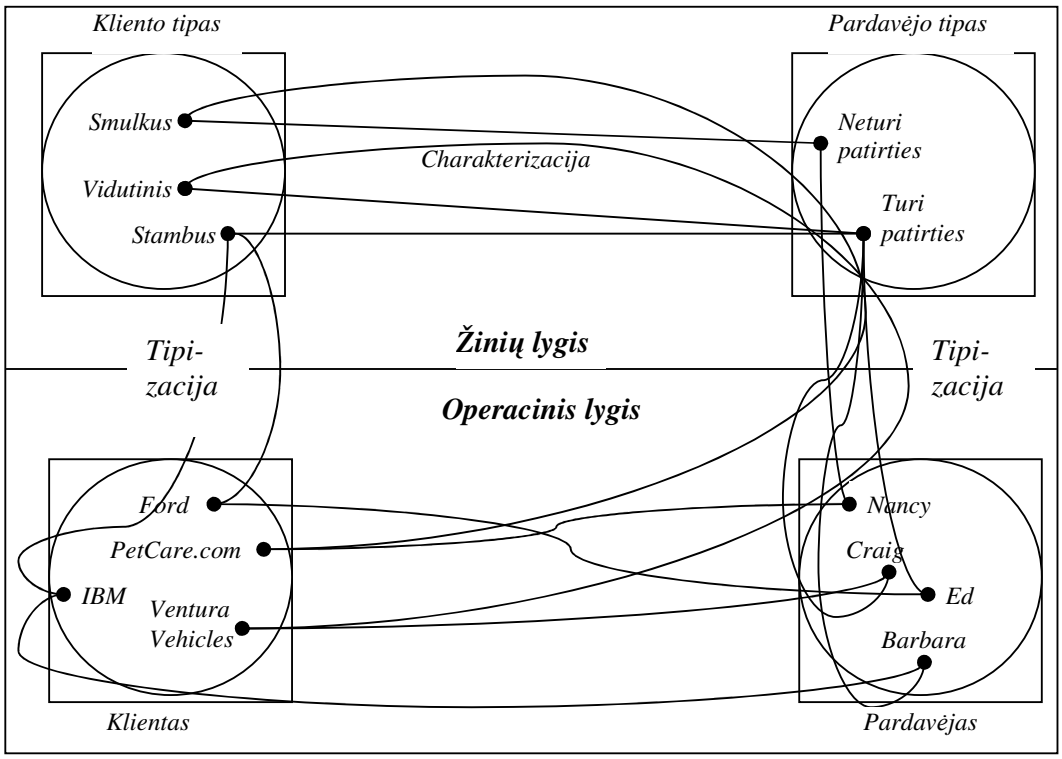
Bendravimo (association) ryšys aprašo priklausomybes tarp agentų. Skiriami trys šio ryšio tipai: ***atsakomybė*** (responsibility), ***priskyrimas*** (assigment), ***bendradarbiavimas*** (cooperation). Pirmąjį tipą – ***atsakomybė*** **(**responsibility) – McCarthy (1982) apibūdina taip: „Atsakomybė (responsibility) rodo, kad aukštesnio lygio vienetai kontroliuoja ir yra atsakingi už žemesnio lygio veiksmus.“ [12]. Kitais žodžiais tariant, atsakomybė rodo darbuotojų hierarchiją tam tikroje organizacijoje, įmonėje, pvz. toks ryšys galėtų egzistuoti tarp direktoriaus, vadybininko, sandėlininko ir kasininko. **Priskyrimo** (assignment) ryšys nusako priklausomybes tarp vidinio ir išorinio agentų, pvz., pardavėjas yra atsakingas už pirkėją arba supirkėjas – už tiekėjus. **Bendradarbiavimo** (cooperation) ryšys rodo priklausomybes tarp išorinių agentų (pvz. „tam tikras vienos organizacijos klientas gali būti kitos organizacijos (pardavėjo) partneris“).

Sąsajos (linkage) ryšys rodo priklausomybes tarp resursų. Svarbus šio ryšio tipas yra **composite** (sudėtis). Toks ryšys sieja tam tikrų resursų visumą, pvz. asmeninio kompiuterio dalys – monitorius, kietasis diskas, klaviatūra, pelė – gali būti aprašomos kaip atskiros dalys.

Priklausomybės (custody) nurodo vidinį agentą, susijusį su tam tikru resursu, pvz., toks ryšys galėtų būti tarp sandėlininko ir prekių, saugomų sandėlyje. Šiuo atveju prekės yra priklausomos nuo sandėlininko.

Įsipareigojimas (Commitment)

McCarthy išplėtė verslo sistemų REA (Resource-Event-Agent) modelį įvesdamas naują sąvoką – **įsipareigojimai** (commitment) (žr. 6 pav.). Ijiri (1975) įsipareigojimus (commitment) apibūdino, kaip „susitarimą vykdyti ekonominį įvykį tiksliai nustatytoje ateityje, kuris sumažins arba padidins resursų kiekį.“. **Reciprocol** (abipusis**) –** tai papildomas ryšys, jungiantis du atitinkamus įsipareigojimus (commitment). Tiksliau šį ryšį apibūdina **ekonominis susitarimas** (economic agreement). Egzistuoja du susitarimų tipai: **kontraktas** (contract) ir **tvarkaraštis** (shedule). Perkėlimas (transfer) vykdo kontraktą (contract), o pasikeitimas (transformation) – tvarkaraštį (shedule). Prisimenant perkėlimo (transfer) ir pasikeitimo (transformation) apibrėžimus galime teigti, kad atsargų įsigijimui naudojamas kontraktas (contract), gamybai – tvarkaraštis (shedule).



6 pav. Charakteristikos (Characteristics) ir strategijos (Policies) [8]

REA (Resource-Event-Agent) ontologijjos pagrindiniai principai:

- Kiekvienam resursui egzistuoja bent vienas resursų įėjimo – *inflow* – ir išėjimo – *outflow* – srautus lemiantis įvykis. Ir atvirkščiai: įėjimo (*inflow*) ir išėjimo (*outflow*) įvykiai turi daryti įtaką tam tikriems resursams.

- Kiekvienas įvykis, susijęs su resurso sumažėjimu turi būti sujungtas su įvykiu, lemiančiu resurso padidėjimą.

- Kiekvienas tam tikras mainų pavyzdys yra susietas su vidiniu ir išoriniu agentais.

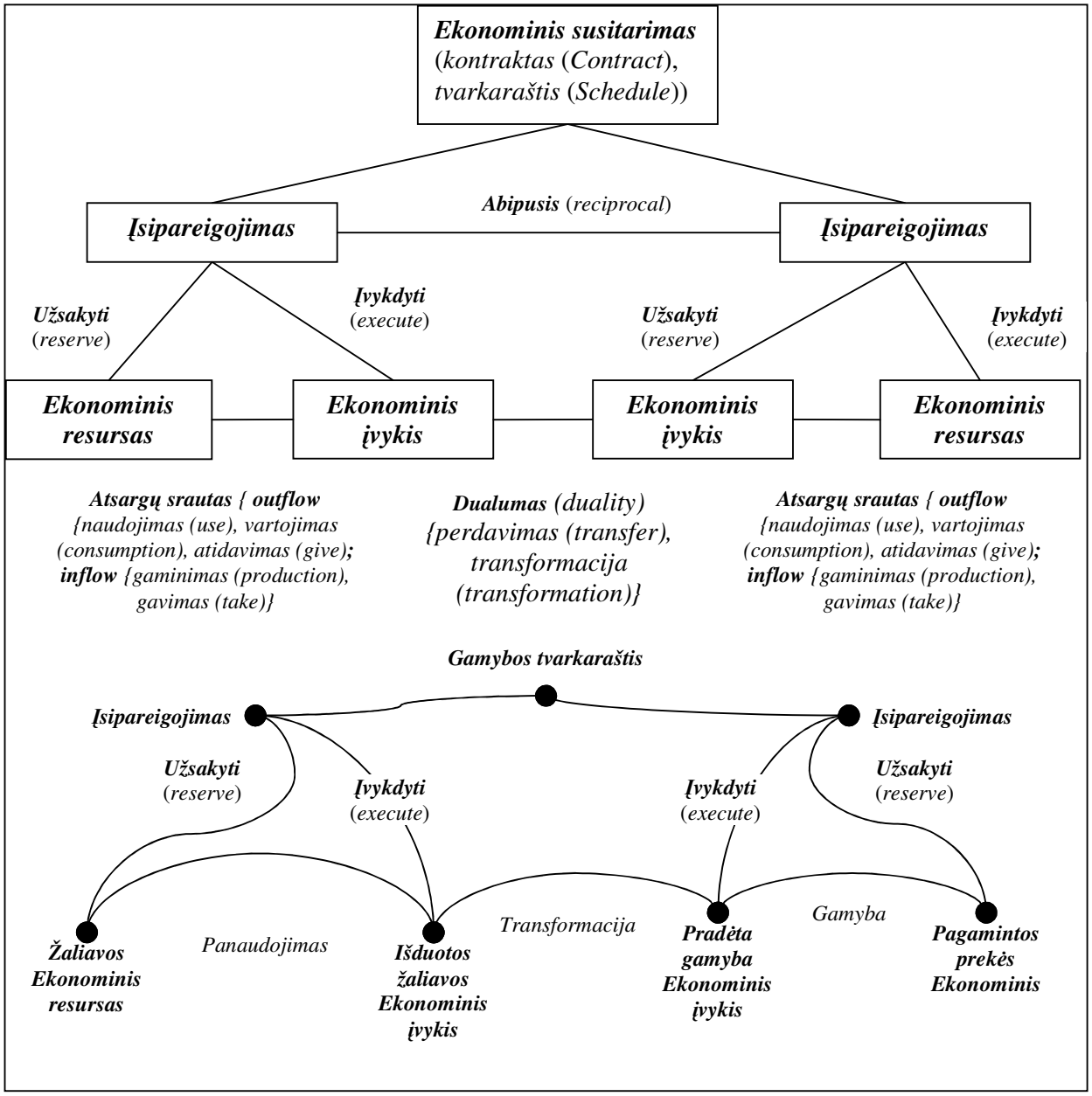
Pirmasis principas aprašo mainų sekos modeliavimą tam tikroje organizacijoje. Antrasis užtikrina teisingas išvardintų mainų konfigūracijas. Trečiasis – garantuoja mainų dalyvavimą tarp konkuruojančių ekonominių verslų.

Žinių infrastuktūra

Įvaizdžio tipas (Type Image)

REA (Resource-Event-Agent) ontologijoje **įvaizdžio tipas (type image)** skirtas pavaizduoti ekonominio reiškinio neapčiuopiamą struktūrą. Įvaizdžių tipo konstravimui naudojama **tipizacija** (typification) (Smith and Smith (1977), Sakai (1981), Brodie (1981) and Odell (1994)).

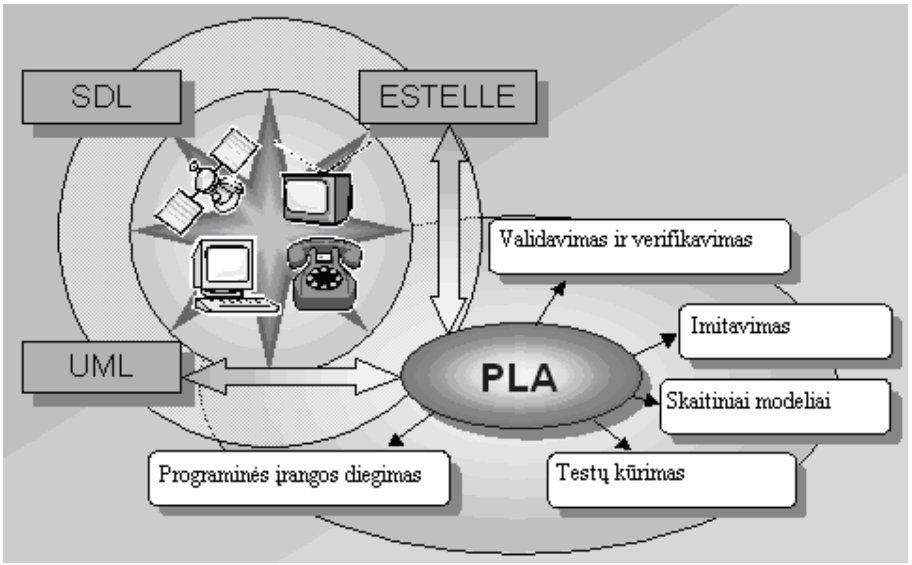
Tipizacija (typification) – tai bendrų sąvokų išraiška konkrečiais vaizdais. Naudojamos klasės: *resursas-resurso tipas* ir a*gentas-agento tipas*. 6 pav. parodyta tipizacija (*typification*) ir charakterizavimas (*characterization*). Tipizacija (*typification*) leidžia aprašyti kiekvieną klientą (organizaciją) kaip smulkų, vidutinį ar stambų. Viršutinė diagramos dalis iliustruoja charakteristikas (*characterization*), kur susiję du skirtingi apibrėžimų (*definition*) tipai. Charakteristikos yra puiki priemonė pateikti bendras strategijas (*policies*) arba valdymo (*control*) procedūras. Strategija (*policy*) pavaizduota 6 pav., rodo, kad patyrę pardavėjai aptarnauja vidutines ir dideles organizacijas, o neturintys patirties – smulkias. Strategijos (*policies*) gali būti naudojamos stebėti tam tikrus ryšius tarp objektų, pvz., užtikrinti, kad tinkamą kvalifikaciją turintis darbuotojas aptarnautų atitinkamą klientą (žr. 6 pav.). Trys charakterizacijos konjunkcijos kategorijos: *agento tipas – agento tipas*; *agento tipas – resurso tipas*; *resurso tipas – resurso tipas*.



7 pav. Įsipareigojimai ir susitarimai (Commitments ir Agreements) [8]

1.3. Verslo procesų modeliavimas ir imitavimas

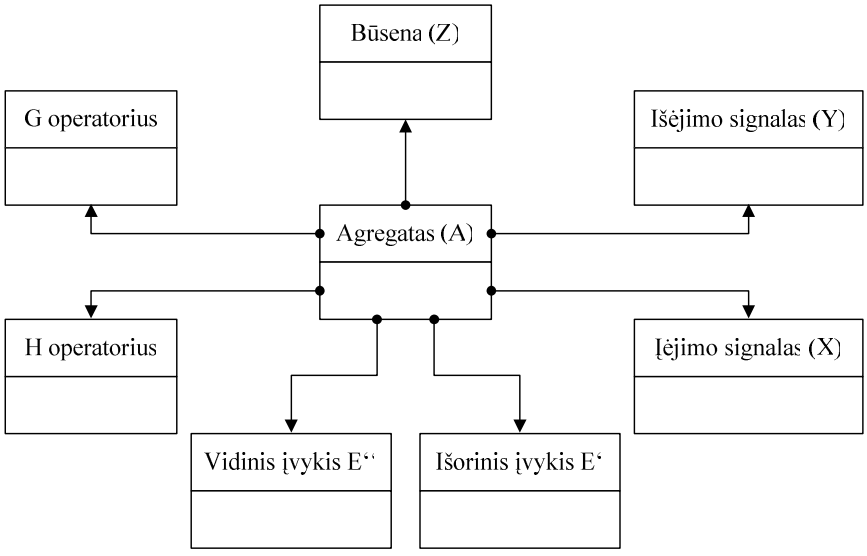
Kaip ir kitos verslo srities ontologijos, REA (Resource-Event-Agent) neturi formalaus aprašo, kuris būtų naudingas jį analizuoti. Todėl šiame darbe naudojami verslo proceso modeliavimui ir imitavimui skirti įrankiai, sukurti PLA (Piece-Linear Aggregate) formalizmo pagrindu, verslo procesų ontologijos formaliam vaizdavimui ir analizei atlikti.



8 pav. MoSIS paskirstytoms sistemoms

PLA (Piece-Linear Aggregate) plačiai taikomas sudarant verslo sistemų specifikacijas, jas modeliuojant, imituojant ir analizuojant [14], [17]. Remiantis sukurtais PLA (Piece-Linear Aggregate) principais, buvo sukurti automatinės analizės įrankiai (žr. 8 pav.).

MoSIS (Modeling, simulation, analysis and implementation suite) skirta paskirstytoms sistemoms, pavaizduota 4 paveiksle, leidžia atlikti kūrimo ir analizės fazes, naudojant formalius metodus. Pagrindinė šios sistemos savybė yra ta, kad visos analizės ir kūrimo fazės paskirstytų sistemų yra atliekamos, remiantis vienintele matematine schema: PLA (Piece-Linear Aggregate). Sistemą sudaro posistemės, kurios atlieka tokias užduotis: sistemos modelių, aprašytų SDL, ESTELLE, UML kalbomis transformacija į PLA (Piece-Linear Aggregate); PLA formalios specifikacijos analizė; imitavimas; testo kūrimas; automatizuotos programinės įrangos diegimas.



9 pav. Agregatų elementai

Agregatiniu požiūriu sumodeliuota sistema suprantama tarpusavyje sąveikaujančių agregatų visuma. Agregatinę sistemą sudaro įėjimo ir išėjimo signalų aibė, būsenų aibė, perėjimo ir išėjimo operatoriai.

Kiekvienas agregatas A yra sistema $A = \langle X, Y, E, Z, H, G \rangle$.

Tokioje sistemoje signalai x_i ($i = \overline{1,N}$, $x_i \in X$, čia $X = \{x_1, x_2, \ldots, x_N\}$) ateina iš įvesties sąveikos taškų, o signalai y_i $i = \overline{1,M}$, $y_i \in Y$, čia $Y = \{y_1, y_2, \ldots, y_N\}$, gaunami iš išvesties sąveikos taškų. Signalai x_i ir y_i gali būti paprasti arba sudėtingi.

Agregato įvykį e_i, ($i = \overline{1,F}, e_i \in E$ čia $E = \{e_{1,}, e_2, \ldots, e_F\}$) sudaro du nesikertantys poaibiai $E = E' \cup E''$.

Poaibį $E' = \{e'_{1,}, e'_2, \ldots, e'_N\}$ sudaro įvykiai $e'_i, i = \overline{1,N}$, kurie įvyksta dėl įėjimo signalų atėjimo iš aibės X. Poaibio E' įvykiai yra vadinami išoriniais ir vienareikšmiai atitinka poaibį E', čia $X \to E'$.

Poaibis $E'' = \{e''_{1,}, e''_2, \ldots, e''_M\}$ yra vadinamas vidinių įvykių poaibiu. Aibės E'' įvykiai fiksuoja operacijos pabaigą. Kiekvienam vidiniam įvykiui e''_i, priskiriama valdymo seka $\{\xi_i^{(j)}, j = \overline{1,\infty}\}$, kuri aprašo operacijos trukmę.

Agregato funkcionavimas patikrinamas laiko momentų aibe $T = \{t_{1,}, t_2, \ldots, \}$, kada įvyksta vienas ar keli įvykiai, įtakojantys agregato būsenos pasikeitimą.

Agregato būseną $z(t_m) = v(t_m) \cup z_v(t_m)$ aprašo diskreti $v(t_m) = \{d_i,\ i = \overline{1,S}\}$ ir tolydi $z_v(t_m) = \{w(e''_1, t_m), \ldots, w(e''_M, t_m)\}$ koordinatės. Tolydi koordinatė $w(e''_i, t_m)$ apibrėžia laiko momentus, kada agregate galimi vidiniai įvykiai. Momentas $w(e''_i, t_m)$ turi neapibrėžtą reikšmę (∞) jei momentu t_m operacija nėra vykdoma:

$$w(e''_i, t_m) = \begin{cases} < \infty, & \text{jei operacija vyksta laiko momentu } t_m, \\ \infty, & \text{priešingu atveju.} \end{cases}$$

Perėjimo ir išėjimo operatoriai aprašo agregato funkcionavimą. Perėjimo operatorius $H(e''_i)$ apibrėžia naują agregato būseną:

$z(t_{m+1}) = H[z(t_m), e_i]$, čia $e_i \in E$.

Išėjimo signalas y_i gali būti siunčiamas, tik tada, kai pasirodo įvykis $e_i \in E$.

Išėjimo operatorius *G* apibrėžia išėjimo signalus: $y(t_{m+1}) = G[z(t_m), e_i]$.

Sumodeliuota sistema yra sąveikaujanti. Agregatinėje sistemoje agregatai sujungti kanalais (žr. 9. pav.). Kiekvienas kanalas jungia įėjimo ir išėjimo taškų porą: $S = \langle A_1, A_2, \ldots, A_n, R \rangle$, čia $R : XP \to YP$ – baigtinė kanalų aibė.

1.4. Loginio programavimo panaudojimas agregatinių specifikacijų teisingumo tikrinimui

Darbe agregatinės specifikacijos bendrosios savybės analizuojamos PROLOG kalbos pagrindu sukurtoje sistemoje [1]. Panaudojant šią sistemą, agregatinė specifikacija ir analizuojamos savybės užrašomos pirmos eilės predikatų logikos poaibio (Horno) sakiniais. Pagal pateiktą metodiką agregatinė specifikacija ir įrodinėjamos savybės užrašomos žemiau nurodytomis formulėmis. Loginės formulės apibrėžia agregato būsenas, perėjimo bei išėjimo operatorius, priklausančius nuo vidinių ir išorinių įvykių.

Perėjimo ir išėjimo operatorių specifikavimui vartojami šie predikatai:

1) QX_name (input_p, input_v, w_1, ..., w_f, d_1, ..., d_s) –

tai predikatas, aprašantis agregato, kurio vardas *name*, būseną, kai įvyksta išorinis įvykis ir į polių *input_p* ateina įėjimo signalas, turintis reikšmę *input_v*; w_j yra agregato būsenos tolydinė, o d_i - diskretinė koordinatės ($w_j \in W_j$; $d_i \in D_i$; $j := 1, \ldots, f$; $i := 1, \ldots, s$).

2) QY_name (output_p, output_v, w_1, ..., w_f, d_1, ..., d_s) –

tai predikatas, analogiškai kaip ir pirmasis predikatas, aprašantis agregato būseną, kuriai esant per išėjimo polių *output_p* yra išsiunčiamas išėjimo signalas, turintis reikšmę *output_v*.

3) QW_name (w_1, ..., w_f, d_1 , ..., d_s) –

tai predikatas, aprašantis agregato, kurio vardas *name*, būseną, kai įvyksta vidinis įvykis.

Aukščiau paminėtieji predikatai toliau naudojami agregato perėjimo ir išėjimo operatorių užrašymui predikatų logikos terminais. Išskiriamos tokios keturios situacijos:

1) Įvykus išoriniam įvykiui, pasikeičia būsenos koordinatės bei siunčiamas išėjimo signalas:

$$QX\_name (input\_p, input\_v, w_1, \ldots, w_f, d_1, \ldots, d_s) \wedge P (w_1, \ldots, w_f, d_1 , \ldots, d_s) \rightarrow$$

$$QY\_name (output\_p, output\_v, next\_w_1, \ldots, next\_w_f, next\_d_1, \ldots, next\_d_s);$$

2) Įvykus išoriniam įvykiui, pasikeičia būsenos koordinatės, bet išėjimo signalas nėra siunčiamas:

$$QX\_name (input\_p, input\_v, w_1, \ldots, w_f, d_1, \ldots, d_s) \wedge P (w_1, \ldots, w_f, d_1, \ldots, d_s) \rightarrow$$

$$QW\_name (next\_w_1, \ldots, next\_w_f, next\_d_1, \ldots, next\_d_s).$$

3) Įvykus vidiniam įvykiui, pasikeičia būsenos koordinatės bei siunčiamas išėjimo signalas:

$$QW\_name (w_1, \ldots, w_f, d_1, \ldots, d_s) \wedge P (w_1, \ldots, w_f, d_1, \ldots, d_s) \rightarrow$$

$QY\_name\ (\ output\_p,\ output\_v,\ next\_w_1,\ ...,\ next\_w_f,\ next\_d_1,\ ...,\ next\_d_s\ ).$

4) Įvykus vidiniam įvykiui, pasikeičia būsenos koordinatės, bet išėjimo signalas nėra siunčiamas:

$QW\_name\ (\ w_1,\ ...,\ w_f,\ d_1,\ ...,\ d_s\ ) \wedge P\ (\ w_1,\ ...,\ w_f,\ d_1,\ ...,\ d_s\ ) \rightarrow$

$QW\_name\ (\ next\_w_1,\ ...,\ next\_w_f,\ next\_d_1,\ ...,\ next\_d_s\ ).$

Agregato, turinčio pavadinimą *name_1*, išėjimo poliaus *output_p* sujungimas su agregato, turinčio pavadinimą *name_2*, įėjimo poliumi *input_p* užrašomas tokia logine formule:

$QY\_name\_1\ (\ output\_p,\ output\_v,\ w_{11},\ ...,\ w_{1f},\ d_{11},\ ...,\ d_{1s}\ ) \rightarrow$

$QX\_name\_2\ (\ input\_p,\ input\_v,\ w_{21},\ ...,\ w_{2f},\ d_{21},\ ...,\ d_{2s}\ ).$

Žemiau pateikiamos formulės, kurios aprašo nagrinėjamas globalines bei individualias agregatų sistemos savybes:

1. Patikimumo tikrinimas pradedamas pradinėje būsenoje:

$Q\_global\ (\ w_{1P1},\ ...,\ w_{1Pf1},\ d_{1P1},\ ...,\ d_{1Ps1},\ ...,\ w_{nP1},\ ...,\ w_{nPfn},\ d_{nP1},\ ...,\ d_{nPsn}\ ).$

2. Tikrinamas galinės būsenos pasiekiamumas:

$Q\_global\ (\ w_{1T1},\ ...,\ w_{1Tf1},\ d_{1T1},\ ...,\ d_{1Ts1},\ ...,\ w_{nT1},\ ...,\ w_{nTfn},\ d_{nT1},\ ...,\ d_{nTsn}\ ).$

3. Tikrinamos tolydžiųjų ir diskrečiųjų koordinačių viršutinė bei apatinė ribos (minėtąsias ribas aprašo predikatas *D*):

$Q\_global\ (\ w_{11},\ ...,\ w_{1f1},\ d_{11},\ ...,\ d_{1s1},\ ...,\ w_{n1},\ ...,\ w_{nfn},\ d_{n1},\ ...,\ d_{nsn}\ ) \wedge D.$

4. Tikrinama, ar specifikacijoje aprašytos visos reakcijos į galimus sistemos įvykius:

$QX\_name\ (\ input\_p,\ input\_v,\ w_1,\ ...,\ w_f,\ d_1,\ ...,\ d_s\ ) \wedge input\_v \in X.$

5. Tikrinama, ar nėra statinių aklaviečių:

$Q\_global\ (\ 0,\ ...,\ 0,\ d_{11},\ ...,\ d_{1s1},\ ...,\ 0,\ ...,\ 0,\ d_{n1},\ ...,\ d_{nsn}\ ).$

6. Tikrinamas invariantas užrašomas predikatu *I*:

$Q\_global\ (\ w_{11},\ ...,\ w_{1f1},\ d_{11},\ ...,\ d_{1s1},\ ...,w_{n1},\ ...,\ w_{nfn},\ d_{n1}\ ,\ ...,\ d_{nsn}\ ) \wedge I.$

Loginio programavimo kalboje PROLOG [1] problemos aprašymui naudojami Horno sakiniai (**loginė dalis**), o sprendimas (**valdančioji dalis**) vykdomas, remiantis atgaliniu (*backward chaining*) išvedimo mechanizmu. Šiame darbe yra siūloma PROLOG kalbą panaudoti agregatinių specifikacijų bei jų savybių analizei.

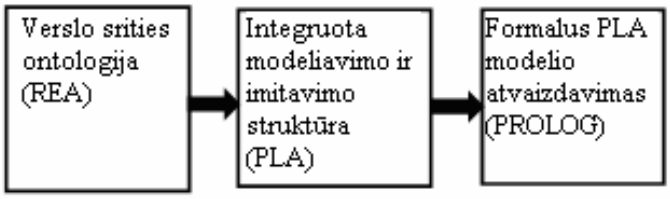
Agregatinė specifikacija užrašyta pirmos eilės predikatų logikos formulėmis, užrašoma PROLOG kalbos sakiniais, ir pastaroji specifikacija taps vykdoma. PROLOG

programos vykdymas valdomas atitinkamu išvedimo metodu, kuris leidžia analizuoti aprašytąsias savybes.

2. REA verslo procesų užrašymas naudojant PLA

2.1. Verslo procesų modeliavimo ir analizės esmė

Verslo procesų analizės metodiką, sudaro REA (Resource-Event-Agent) modelio pagrindu sudarytų verslo procesų užrašymą panaudojant PLA (Piece-Linear Aggregate) formalizmą. Toliau sudaryta agregatinė specifikacija analizuojama taikant pirmos eilės predikatų logiką. Jos užrašymo teisingumas tikrinamas, naudojant loginio programavimo kalbą – PROLOG. Šios metodikos esmė pavaizduota (žr. 10 pav.). Pirmame žingsnyje PLA (Piece-Linear Aggregate) modelis turi būti sukurtas REA (Resource-Event-Agent) struktūrai. Antrame žingsnyje yra naudojama ankstesniuose skyriuose aprašyta paskirstytų sistemų modeliavimo ir analizės sistema MoSIS.



10 pav. REA (*Resource-Event-Agent*) *metodika taikant PLA* (*Piece-Linear Aggregate*)

Pimojo metodologijos žingsio sąryšiai aprašyti toliau.

2.2. REA ir PLA modelių ryšiai

Tegul REA (Resource-Event-Agent) sistema aprašoma kaip:

S_{REA}=*(resursai, įvykiai, ryšiai, būsenos),*

o PLA (Piece-Linear Aggregate) sistema:

S_{PLA}=*(agregatai, kanalai, įėjimai, išėjimai, įvykiai, būsenos, perėjimai)*

Ryšiai tarp S_{REA} ir S_{PLA} sistemų :

T_1: $Agentai_{REA} \rightarrow Agregatai_{PLA}$

T_2:$Ryšiai\ tarp\ agentų_{REA} \rightarrow Kanalai\ tarp\ agregatų_{PLA}$

T_3: $Resursai_{REA} \rightarrow Įėjimai, Išėjimai_{PLA}$

T_4: $Įvykiai_{REA} \rightarrow Įvykiai_{PLA}$

Čia $Įvykiai_{PLA} = E^{Išorinis} \cup E^{Vidinis}$ ir $Įvykiai_{REA} = E^{Išorė} \cup E^{Vidus}$

T_5: $Būsenos_{REA} \rightarrow Būsenos_{PLA}$, *čia* $Būsena_{PLA}=W \cup D$, $E^{vidinis}{}_{PLA} \rightarrow W$ *ir* $D_{PLA} \rightarrow$*Resursų būsena,* $Būsena_{REA}=W \cup D$, $E^{vidinis}{}_{REA} \rightarrow W$ *ir* $D_{REA} \rightarrow$*Resursų būsena.*

T_6: $Perėjimas_{REA} \rightarrow Perėjimas_{PLA}$,

čia $Perėjimas_{PLA}$: $Įvykis_{PLA} \times Būsena_{PLA} \rightarrow Būsena_{PLA}$

ir $Perėjimas_{REA}$: $Įvykis_{REA} \times Būsena_{REA} \rightarrow Būsena_{REA}$.

2.3.REA transformavimas į PLA

Plačiau aprašysime aukščiau pateiktus modelių ryšius.

Taikant agregatinį požiūrį, sumodeliuota sistema yra aprašoma sąveikaujančių agregatų visuma. Agregatinė sistema S, sudaryta iš n elementų yra:

S = (A1 , A2 , ..., An , R), čia A1 , A2 , ..., An yra sistemos agregatai; R atitinka kanalų, jungiančių agregatus, aibę.

Kiekvienas agregatas A yra sistema:

A = (X , Y , E' , E", Z , H , G), čia:

X yra įėjimo signalų aibė;

Y yra išėjimo signalų aibė;

Z yra išplėstų baigtinių būsenų aibė: $Z = D \cup W$, čia D D = { d1 , d2, ... , dp } yra diskrečių koordinačių aibė ir W = { w1 , w2, ... , wf } yra tolydžių koordinačių aibė;

E' yra išorinių įvykių klasė. Šie įvykiai siejami su įėjimo signalais ($E' \rightarrow X$);

E" vidinių įvykių klasė. Šie įvykiai siejami su tolydžiosiomis koordinatėmis ($E" \rightarrow W$);

$H : E \times Z \rightarrow Z$ yra būsenų perėjimo operatorius, čia $E = E' \cup E"$;

$G : E \times Z \rightarrow Y$ yra išėjimo operatorius.

Transformuojant REA teiginius į PLA, reikia aprašyti visus rinkinius (žr. 1 lentelę).

1 lentelė

REA transformavimas į PLA

Aibės	REA taikymas	PLA taikymas
A	Aprašomi du kontaktuojantys agentai A_i ir A_j (ryšys tarp agentų $r_{ij} \in R$).	Aprašomi du agregatai A_i,A_j , sujungti kanalais $r_{ij} \in R$.
R	Kontaktuodami jie siunčia bendravimo dokumentus.	Kontaktuodami agregatai vienas kitam siunčia signalus.
Y **X**	Vienas agentas A_i siunčia dokumentą y_i ($y_i \in Y$) tuo tarpu kitas agentas A_j gauna dokumentą x_i ($x_i \in X$).	Vienas iš agregatų A_i siunčia signalą y_i ($y_i \in Y$). Tuo tarpu kitas agregatas A_j gauna signalą x_i ($x_i \in X$).
E'	Gautas dokumentas iš išorinio agento sukelia vidinio agento įvykį e'_i ($e'_i \in E'$).	Iš išorinio agregato gautas signalas inicijuoja išorinį agregato įvykį e'_i ($e'_i \in E'$).
E"	Vidinio agento įvykiai sukelia kitų vidinių agentų įvykius $e"_i$ ($e" \in E"$).	Vidinių įvykių operacijos inicijuoja kitų vidinių įvykių operacijas $e"_i$ ($e"_i \in E"$).
W	Įvykio pasirodymo laiko momentą aprašo koordinatė w_i ($w_i \in W$).	Įvykio pasirodymo laiko momentą aprašo valdymo koordinatė w_i ($w_i \in W$).
Z	Įvykiai gali veikti resursų (prekių ar	Įvykiai gali veikti būsenos koordinačių

	pinigų) kiekio pasikeitimus.	pasikeitimus.
D	Informaciją apie resursus aprašo diskreti koordinatė d_i ($y_i \in D$).	Informaciją apie resursus aprašo diskreti koordinatė d_i ($y_i \in D$).
H	Diskrečių koordinačių pasikeitimai priklauso nuo įvykių. Jų priklausomybes aprašo H išraiškos Pasibaigus įvykiui, išoriniam agentui gali būti siunčiamas dokumentas y_i, ($y_i \in$ Y).	Diskrečių koordinačių pasikeitimai priklauso nuo įvykių. Jų priklausomybes aprašo H operatorius. Pasibaigus operacijai, išoriniam agregatui gali būti siunčiamas signalas y_i, ($y_i \in$ Y).
G	Dokumento formavimą aprašo G išraiška.	Signalo formavimą aprašo G operatorius.

3. Apskaitos sistemos vidaus kontrolės individualiųjų savybių tikrinimas

Individualių savybių užrašymui gali būti taikomi apskaitos sistemų vidaus kontrolės apribojimai. Šiame skyriuje pateiktas specifikacijos individualiųjų savybių užrašymas, taikant reliacinę algebrą, SQL ir PROLOG.

3.1. Apskaitos sistemos vidinė kontrolė

Reliacinės duomenų bazės naudojamos saugoti ir valdyti didelius įrašų kiekius. Šie įrašus sieja bendra struktūra ir ryšiai tarp jų [11].

Norint užtikrinti saugomų duomenų kiekį reikia užtikrinti sistemos patikimumą. Siekiant įvertinti jos patikimumą atliekama sistemos analizė bei sistemoje užrašomos vidinės kontrolės.

Teisingai aprašytos vidinės kontrolės gali prižiūrėti tikslius ir patikimus duomenis, pagerinti operacinį efektyvumą bei prisidėti prie valdymo strategijų.

Spartus informacinių technologijų (IT) augimas nekeičia vidinių kontrolių poreikio verslo sistemoms. IT integracija į verslo procesus išplečia IT sistemų bei IT paremtų vidinių kontrolių vaidmenį. Pripažįstant, IT reikšmę vidinėms kontrolėms ir metodą, kuriuo turėtų būti paremtas patikrinimas, ASB (Auditing Standards Board) 2001m. išleido SAS Nr. 94 „*The Effect of Information Technology on the Auditor's Consideration of Internal Control in a Financial Statement Audit*“. Čia numatyta teikti kursus apie IT vidinių kontrolių supratimą ir valdymo rizikos įvertinimą.

Vidinis valdymas – tai procesas, užtikrinantis sistemos patikimumą, efektyvumą bei našumą [11]. Šios sąvokos teisinga interpretacija svarbi tiek, vadybininkams, tiek, projektuotojams, tiek vartotojams.

Vidinių kontrolių komponentai.

Remiantis 1992m. parengta ataskaita „Internal Control – An Integrated Framework“, kurią sudarė COSO (*Committee of Sponsoring Organizations*), išskiriami penki vidinių kontrolių komponentai (1) valdymo aplinka, (2) rizikos sumažinimas, (3) valdymo veiklos, (4) informacija ir komunikacija bei (5) stebėjimas (*monitoring*). Šiame darbe dėmesys koncentruojamas į vidines kontroles [11].

3.2. Individualiųjų savybių panaudojimas, taikant reliacinę algebrą, SQL ir PROLOG

Vidinės apskaitos kontrolės gali būti išreikštos apribojimais. Jei duomenys saugomi RDBMS (reliacinės duomenų bazių valdymo sistemose) šios vidinės kontrolės gali būti realizuotos, naudojant apribojimus, teiginius ir trigerius SQL kalboje. Toliau pateikta vidinės kontrolės, aprašytos reliacinėje algebroje, SQL kalboje bei PROLOG [11].

Šiuolaikinės duomenų bazės dažniausiai paremtos reliaciniu modeliu. Šis modelis plačiai naudojamas, nes ryšio (*relation*) sąvoką lengva interpretuoti ir naudoti. Reliacinį modelį sudaro dviejų lygių lentelė. Žemiau pateikta lentelė sudaryta iš atributų (vardas, adresas, darbuotojo ID ir pajamų mokesčio procentas (PVM)).

1 lentelė. Darbuotojai

Vardas	*Adresas*	*ID*	*darbLikutis*
Vidas Vidaitis	*Jonavos 12b*	*6666666666*	*15*
Antanas Visažinis	*Pagėgių 1-45*	*5555555555*	*15*
Petras Vingis	*Saulės 5-52*	*4444444444*	*15*

Panašiai atrodys ir kita atlyginimų lentelė su atitinkamais atributais.

2 lentelė. Atlyginimai

DarbuotojoID	*Atlyginimas*	*Data*
6666666666	*326,62 Lt*	*2006.03.24*
5555555555	*390,62 Lt*	*2006.03.24*
4444444444	*390,62 Lt*	*2006.03.24*

Reliacinė schema aukščiau aprašytoms darbuotojų ir atlyginimų lentelėms aprašoma taip:

Darbuotojai(vardas, adresas, ID, darbLikutis)

Atlyginimai(darbuotojoID, atlyginimas, data)

I. Vidinių kontrolių aprašymas reliacinėje algebroje.

Reliacinę algebrą sudaro formali sistema, skirta ryšiams sudaryti, naudojant tokias operacijas, kaip **apjungimas** (*union*), **sankirta** (*intersection*), **skirtumas** (*difference*) ir specialios reliacinės operacijos, tokios kaip **išrinkimas** (*selection*), **projektavimas** (*projection*) ir **apjungimas** (*join*) [11]. Vidiniai apribojimai reliacinėje algebroje taikomi atributams arba jų rinkiniams. Apribojimai aprašo sąlygas, kurias turi tenkinti duomenys. Žemiau pateikti pavyzdžiai, iliustruojantys vidinės apskaitos kontrolių aprašymą reliacinėje algebroje.

1. Vidinis apribojimas, kuris neleidžia mokėti atlyginimus netikriems darbuotojams.

Reliacinė schema:

MokamiAtlyginimai (darbuotojoID, suma)

Darbuotojai (vardas, adresas, ID, darbLikutis)

Jei lentelę *Darbuotojai* sudaro visi teisėti darbuotojai,

$$\pi_{darbuotojoID}(MokamiAtlyginimai) \subseteq \pi_{ID}(Darbuotojai)$$

arba,

$$\pi_{darbuotojoID}(MokamiAtlyginimai) - \pi_{ID}(Darbuotojai) = \phi$$

2. Vidinė kontrolė neleidžianti atlikti transakcijų su ne įmonės darbuotojais.

Reliacinė schema:

Klientas(klientID, klientVardas, klientAdres, saskNr, patvirtintasSaskNr)

Užsakymas(uzsID, klientID, produktoID, uzsData)

Vidinis apribojimas:

$$\pi_{klientID}(Uzsakymas) \subseteq \pi_{klientID}(Klientas)$$

arba,

$$\pi_{klientID}(Uzsakymas) - \pi_{klientID}(Klientas) = \phi$$

3. Sąskaitos likučio dydžio apribojimas.

Reliacinė schema:

Klientas(klientID, klientVardas, klientAdres, saskNr, patvirtintasSaskNr)

Įvesime apribojimą, kuris neleidžia sąskaitoje viršyti 5mln. sumos.

$$\sigma_{saskNr > 5000000}(Klientas) = \phi$$

4. Apribojimas kredito departamento pareigūno, tvirtinančio sąskaitos likutį, rangui riboti.

Leistinas atitinkamas departamento pareigūno rangas suteikiant kreditą yra pageidaujama vidinė kontrolės strategija.

Reliacinė schema:

Klientas(klientID, klientVardas, klientAdres, saskNr, patvirtintasSaskNr)

SaskDepPareig(vardas, rangas, idarbMetai)

II. Vidinių apribojimų aprašymas, naudojant SQL kalbą.

Jei duomenys saugomi reliacinėje duomenų valdymo sistemoje, vidinės kontrolės gali būti aprašytos apribojimais (*constraints*), teiginiais (*assertions*), trigeriais (*triggers*) [4]. Populiariausios RDBMS sistemos palaiko vientisus semantinius apribojimus (teiginius, trigerius), tam kad apsaugoti nuo klaidingų duomenų nesuderinamumo su realaus pasaulio verslo taisyklėmis. Šiuos apribojimus galima aprašyti naudojant SQL kalbą. Skirtingai nei reliacinėje algebroje, SQL kalboje užrašyti apribojimai nusako sąlygas, kurias turi atitikti duomenys. Apribojimo sąlyga įvedama, kai apriboti elementai yra pakeisti. Kai apribojimas yra aprašytas, modifikacijos duomenų bazėje pažeidžiančios tą apribojimą yra negalimos.

Vidinės kontrolės SQL kalboje.

1. Vidinis apribojimas, kuris neleidžia mokėti atlyginimus netikriems darbuotojams.

```
CREATE TABLE MokamiAtlyginimai (
darbuotojoID CHAR(10),
suma REAL,
mokData DATE,
FOREIGN KEY dartuotoID REFERENCES Darbuotojai(ID));
```

Pastaba. Raktas čia nusako apribojimą nemokėti atlyginimo tiems darbuotojams, kurių darbuotojoID reikšmė nesutampa su lentelės Darbuotojas atributu ID.

2. Vidinė kontrolė neleidžianti atlikti transakcijų su ne įmonės darbuotojais.

```
CREATE TABLE Uzsakymas(
uzsID CHAR(10) UNIQUE NOT NULL,
klientID CHAR(10),
produktoID CHAR(10) REFERENCES Produktas(PID),
uzsData DATE);
FOREIGN KEY klientID REFERENCES Klientas(custID));
```

Pastaba. Raktas, aprašytas paskutinėje eilutėje, leidžia tik tas operacijas, kur *klientID* reikšmė sutampa su lentelės *Klientas* atributo *klientID* reikšme.

3. Sąskaitos likučio dydžio apribojimas.

```
CREATE TABLE Klientas (
klientID CHAR(10) UNIQUE NOT NULL,
klientVardas CHAR(30),
klientAdresas VARCHAR(255),
saskLik REAL CHECK (saskLik <= 5000000),
sakjLikPatvirt CHAR(10));
```

Pastaba. Aukščiau pateiktą apribojimą aprašo penkta eilutė.

III. Vidinių apribojimų aprašymas, naudojant PROLOG kalbą

1. Vidinis apribojimas, kuris neleidžia mokėti atlyginimus netikriems darbuotojams.

```
id(123456).
id(234567).

darbuotojas(123456, 1200, 15).
darbuotojas(234567, 1500, 15).
darbuotojas(456789, 1000, 15).

atlyginimas:-
write('Moketi atlyginimus siems darbuotojams: '), nl,
darbuotojas(A, B, C),
```

```
id(X),
A==X,
write(A), write(', '), write(B), write(', '), write(C), nl,
fail.
```

2. Vidinė kontrolė neleidžianti atlikti transakcijų su ne įmonės darbuotojais.

```
productID(1111).
productID(2222).
productID(4444).
uzs(1, 654321, 1111, 10).
uzs(2, 765432, 2222, 20).
uzs(3, 987654, 3333, 15).

uzsakymas:-
write('Pateikti uzsakymai: '), nl,
%- write('UzsID '), write('klientID '), write('productID '), write('Data '), nl,
uzs(A, B, C, D),
A\==[],
productID(G),
C==G,
write(A), write(', '), write(B), write(', '), write(C), write(', '), write(D), nl,
fail.
```

3. Sąskaitos likučio dydžio apribojimas.

```
klientas(123456, 'Jonas', 'Studentu 9-51', 100000).
klientas(234567, 'Petras', 'Baltu 56A', 200000).
klientas(456789, 'Bronius', 'Saules 24B', 600000).
creditline:-
write('Siu klientu saskaitos likutis nevirsyja nustatytos sumos: '), nl,
klientas(A, B, C, D),
A\==[],
D=<500000,
write(A), write(', '), write(B), write(', '), write(C), write(', '), write(D), nl,
fail.
```

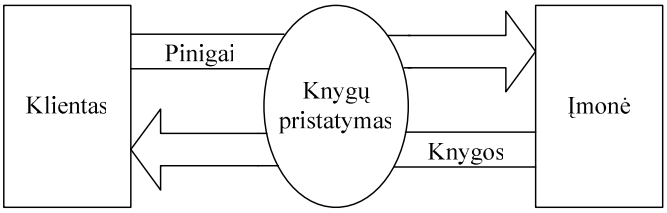
4. Verslo procesų modeliavimo ir analizės metodologijos pavyzdys

Iliustruojant metodiką, šiame skyriuje užrašytas pardavimo ciklo REA (Resource-Event-Agent) modelio pavyzdys, taikant PLA (Piece-Linear Aggregate) struktūrą ir atlikta sistemos agregatinės specifikacijos analizė.

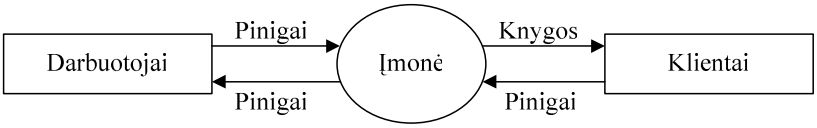
4.1. Pavyzdžio REA modelis

Šis pavyzdys aprašo pardavimo verslo procesą, naudojant elektroninį katalogą. *Pirkėjas* užsisako knygas iš minėto katalogo. Šiuo atveju *Pirkėjas,* nevykdamas į parduotuvę, gali peržiūrėti *Pardavėjo* siūlomas prekes. *Pirkėjas* siunčia *Užsakymą Pardavėjui*. Jei užsakymas priimamas, *Pardavėjas* pristato knygas *Pirkėjui*. Nagrinėjamame pavyzdyje prekiaujama tik vienos rūšies knygomis. Vaizduojamas pardavimo procesas be apmokėjimo įvykio.

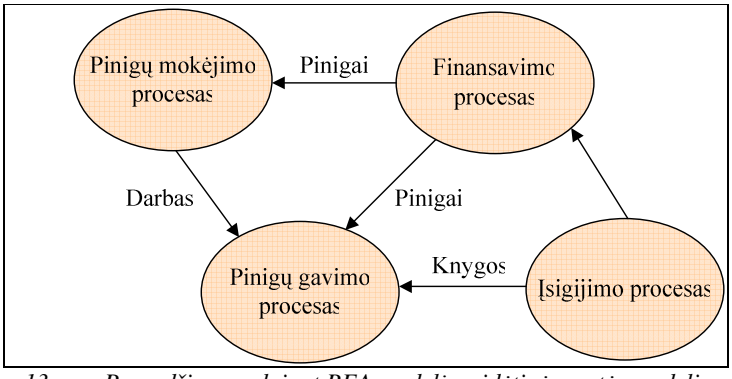
11-15 paveiksluose pavaizduotas vertės sistemos, pridėtinės vertės, proceso modeliai bei veiklų diagrama.



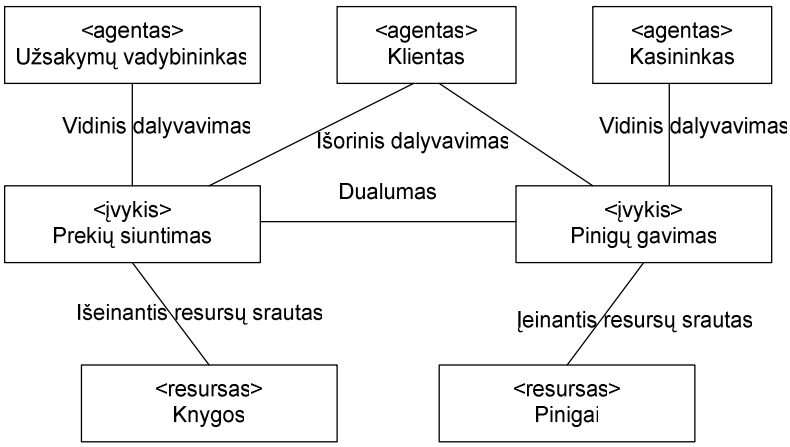
11 pav. Pavyzdžio verslo procesas



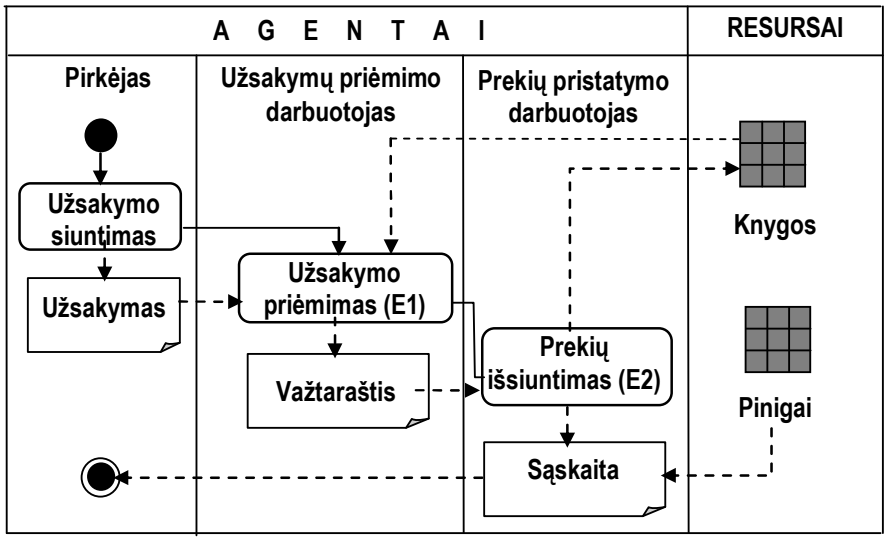
12 pav. Pavyzdžio, naudojant REA modelį, sistemos lygmuo



13 pav. Pavyzdžio, naudojant REA modelį, pridėtinės vertės modelis



14 pav. Pavyzdžio, naudojant REA modelį, verslo proceso lygmuo



15 pav. Pavyzdžio veiklų diagrama (be pinigų gavimo)

4.2.REA modelio transformavimas į PLA

Pavyzdžio transformacijos pateiktos 2 lentelėje.

Pavyzdžio REA modelio transformavimas į PLA

2 lentelė

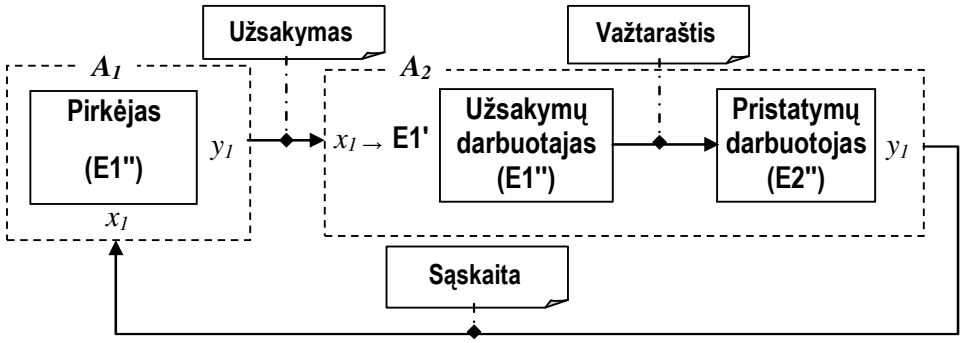
Aibės	REA taikymas	PLA taikymas
A	Aprašomi du kontaktuojantys agentai A_i ir A_j (ryšys tarp agentų r_{ij} $\in$R).	Aprašomi du agregatai A_i,A_j , sujungti kanalais r_{ij} $\in$R.
R	Kontaktuodami jie siunčia bendravimo dokumentus.	Kontaktuodami agregatai vienas kitam siunčia signalus.
Y **X**	Vienas agentas A_i siunčia dokumentą y_i (y_i $\in$Y) tuo tarpu kitas agentas A_j gauna dokumentą x_i (x_i $\in$X).	Vienas iš agregatų A_i siunčia signalą y_i (y_i $\in$Y). Tuo tarpu kitas agregatas A_j gauna signalą x_i (x_i $\in$X).
E'	Gautas dokumentas iš išorinio agento sukelia vidinio agento įvykį e'_i (e'$_i$ $\in$E'). Pavyzdžiui, gautas	Iš išorinio agregato gautas signalas inicijuoja išorinį agregato įvykį e'_i (e'$_i$ $\in$E').

	dokumentas *Užsakymas* iš išorinio agento inicijuoja *užsakymo priėmimo* įvykį.	Pavyzdžiui, gautas signalas apie užsakymą inicijuoja išorinį įvykį – užsakymo priėmimas.
E”	Vidinio agento įvykiai sukelia kitų vidinių agentų įvykius $e”_i$ (e” $\in$E”).	Vidinių įvykių operacijos inicijuoja kitų vidinių įvykių operacijas $e”_i$ (e”$_i \in$E”).
W	Įvykio pasirodymo laiko momentą (datą) aprašo koordinatė w_i ($w_i \in$W).	Įvykio pasirodymo laiko momentą aprašo valdymo koordinatė w_i ($w_i \in$W).
Z	Įvykiai gali veikti resursų (prekių ar pinigų) kiekio pasikeitimus.	Įvykiai gali veikti būsenos koordinačių pasikeitimus (pvz., informacija apie prekes ar pinigus).
D	Informaciją apie resursus aprašo diskreti koordinatė d_i ($y_i \in D$) (prekių kiekis arba pinigų kiekis).	Informaciją apie resursus aprašo diskreti koordinatė d_i ($y_i \in D$) (pvz., prekių kiekis ar pinigų kiekis).
H	Diskrečių koordinačių pasikeitimai priklauso nuo įvykių. Jų priklausomybes aprašo H išraiškos (pvz., prekių kiekis sandėlyje sumažėjo). Pasibaigus įvykiui, išoriniam agentui gali būti siunčiamas dokumentas y_i, ($y_i \in$Y).	Diskrečių koordinačių pasikeitimai priklauso nuo įvykių. Jų priklausomybes aprašo H operatorius (pvz., prekių kiekis sandėlyje sumažėjo). Pasibaigus operacijai, išoriniam agregatui gali būti siunčiamas signalas y_i, ($y_i \in$Y).
G	Dokumento formavimą aprašo G išraiška.	Signalo formavimą aprašo G operatorius.

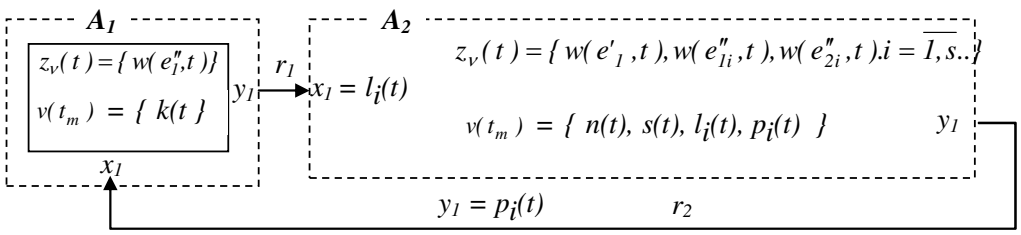
4.3.Pavyzdžio PLA (Piece-Linear Aggregate) modelis

Agregatinę specifikaciją sudaro sistema $S = (A_1 , A_2, R)$, čia A_1 , A_2 yra sistemos agregatai; R atitinka du kanalus (r_1 ir r_2), jungiančius agregatus (žr. 4-5 pav.).

Žemiau pateiktas formalus agregatų aprašymas.



16 pav. REA (Resource-Event-Agent) modelio transformacija į PLA (Piece-Linear Aggregate)



17 pav. Agregatinė sistema

Agregato A1 formalus aprašymas

1. Įėjimų aibė: $X = \{ x_1 \}$.
2. Išėjimų aibė: $Y = \{ y_1 \}$.
3. Išorinių įvykių aibė $E' = \{e'_1\}$;

   e'_1 - įvykis, aprašantis sąskaitos atėjimo įvykį;
4. Vidinių įvykių aibė $E'' = \{e''_1\}$,

   čia e''_1 - įvykis, aprašantis užsakymo formavimą,
5. Valdymo sekos:

   $e''_1 \rightarrow \tau_1, \tau_2, \ldots$, čia τ_i aprašo i-ojo įvykio trukmę.
6. Diskrečioji būsenos $v(t_m) = \{ k(t_m) \}$ komponentė, čia

   $k(t_m)$ - knygų kiekis užsakyme (momentu t)
7. Tolydžioji būsenos komponentė:

   $z_v(t) = \{w(e''_1, t)\}$, čia

   $w(e''_1, t)$ aprašo momentą, kai prasideda įvykis.
8. Pradinė būsena:

   $w(e''_1, t_0) = t_0 + \tau_1$; $k(t_m) = 0$.
9. Perėjimo operatoriai:

   $H(e'_1)$: .

   $H(e''_1)$:
   $k(t_m) = 1 + random(5);$
   $w(e''_1, t_m) = t_m + \tau_m.$

   $G(e''_1)$:
   $y_1(t_m) = k(t_m).$

Formalus agregato A2 aprašymas:

1. Įėjimų aibė: $X = \{ x_1 \}$.
2. Išėjimų aibė: $Y = \{ y_1 \}$.
3. Išorinių įvykių aibė $E' = \{e'_1\}$;

   e'_1 - įvykis, aprašantis užsakymo atėjimą į A2 agregatą;
4. Vidinių įvykių aibė $E'' = \{ e''_{11}, e''_{21}, e''_{12}, e''_{22}, \ldots, e''_{1s}, e''_{2s} \}$,

   čia:

e''_{1i} - įvykis, aprašantis i-ojo užsakymo vykdymo pradžią;

e''_{2i} - įvykis, aprašantis i-ojo užsakymo vykdymo pabaigą (prekių pristatymo įvykis);

5. Valdymo sekos:

   $e''_{1i} \to \xi_i$, čia ξ_i reiškia užsakymo operacijos vykdymo trukmę;

   $e''_{2i} \to \zeta_i$, čia ζ_i reiškia prekių pristatymo operacijos vykdymo trukmę.

6. Diskrečioji būsenos $v(t_m) = \{n(t), s(t), l_i(t), p_i(t)\}$ komponentė, čia

   $n(t)$ – prekių kiekis sandėlyje (laiko momentu t);

   $s(t)$ – užsakymų kiekis (laiko momentu t);

   $l_i(t)$ – i-ojo užsakymo prekių kiekis (laiko momentu t);

   $p_i(t)$ – i-ojo užsakymo vertė (laiko momentu t).

7. Tolydžioji būsenos komponentė:

   $z_v(t) = \{w(e''_{11}, t), w(e''_{21}, t), w(e''_{12}, t), w(e''_{22}, t), ..., w(e''_{1s}, t), w(e''_{2s}, t)\}$, čia

   $w(e''_{1i}, t)$ - laiko momentas, kai i-asis užsakymas bus pradėtas vykdyti;

   $w(e''_{2i}, t)$ - laiko momentas, kai i-asis užsakymas bus baigtas vykdyti (pradėtas prekių pristatymas).

8. Pradinė būsena:

   $n(t_0) = 35$; $s(t_0) = 0$;

   $p_i(t_0) = 0$; $l_i(t_0) = 0$; $i = \overline{1,s}$

   $w(e''_{1i}, t_0) = \infty$;

   $w(e''_{2i}, t_0) = \infty$.

9. Perėjimo operatoriai:

   $H(e'_1)$:

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

   $l_{s(t_m)} = x_1;$

   $$n(t_m) = \begin{cases} n(t_{m-1}) - l_{s(t_m)}, \text{ jeigu } n(t_{m-1}) - l_{s(t_m)} >= 0; \\ n(t_{m-1}), \text{ priešingu atveju}; \end{cases}$$

   $$w(e''_{1s(t_m)}, t_m) = \begin{cases} t_m + \xi_m, \text{ jeigu } n(t_{m-1}) - l_{s(t_m)} >= 0; \\ \infty, \text{ priešingu atveju}; \end{cases}$$

$H(e''_{1i});\ i = \overline{1,s}:$

$w(e''_{2i}, t_m) = t_m + \zeta_m;$

$p_i(t_m) = 2.5 * l_i(t_{m-1});$

$H(e''_{2i});\ i = \overline{1,s}:$

$s(t_m) = s(t_{m-1}) - 1.$

$G(e''_{2i});\ i = \overline{1,s}:$

$y_1(t_m) = p_i(t_m)$

Sujungimai tarp agregatų yra pateikti 3 lentelėje.

3 lentelė

Agregatų sujungimai

Kanalas	*Agregatas_i*	*Išėjimo signalas*	*Agregatas_j*	*Įėjimo signalas*
1.	AG1	y_1	AG2	x_1
2.	AG2	y_1	AG1	x_1

4.4.Pavyzdžio aprašymas panaudojant Prolog kalbą

4.4.1. Perėjimo operatorių užrašymas

Išorinių įvykių perėjimo operatoriai

Sistema, kurios pagrindu iliustruojamos žinių bazės sudedamosios dalys, yra sudaryta iš vieno agregato ir joje negalimi išoriniai įvykiai, todėl nepateiksime atitinkamo PROLOG sakinio.

Vidinių įvykių perėjimo operatoriai

```
qt1(W1,W2L,W3L,List,S,N,X6,L) :-
    W1=:=1,
    LL is 10,
    N >= LL,!,
    procnr(1),
    X7 is X6+1,
    itraukti(W2L,1,W2Ln),
    itraukti(List,LL,Listn),
    Sn is S+1,
    Nn is N-LL,
    itraukti(L,qt1(W1,W2L,W3L,List,S,N),Ln),!,
    qw(W1,W2Ln,W3L,Listn,Sn,Nn,X7,Ln),
    !.
```

Šis PROLOG sakinys teigia, kad įvykus išoriniam įvykiui, t.y. pasirodžius užsakymui, keičiasi sistemos būsenos koordinačių reikšmės:

– Sąrašas *W2L* (šiame sąraše vienetukais pažymimos į sistemą atėję užsakymai) atitinkamai papildomas vienetuku.

– Sąrašo *W3L* turinys lieka nepakitęs (nei vienas užsakymas nepradėtas aptarnauti).

– Sąrašas *List* papildomas į sistemą atėjusių užsakymų skaičiumi *LL* (kiekvienas jo elementas – tai skaičius, nurodantis tam tikro užsakymo prekių kiekį).

– Diskrečioji koordinatė *Sn* (į sistemą atėjusių užsakymų skaičius) padidinama vienetu, o *Nn* (prekių kiekis sandėlyje) sumažinama dydžiu *LL*, nusakančiu atėjusio užsakymo prekių (knygų) skaičių.

```
qt_Ice3(W2L,W3L,List,S,N,X6):-
     W3L\==[],!,
    procnr(6),
```

```
    X7 is X6+1,
    ismesti1(1,W3L,W3Ln),
    pirm(L,List),
    ismesti1(L,List,Listn),
    Sn is S-1,
    Nn is N+L,
     qw_Ice(W2L,W3Ln,Listn,Sn,Nn,X7),
     !.
```

Šis PROLOG sakinys teigia, kad įvykus vidiniam įvykiui, t.y. baigus aptarnauti vieną užsakymą, keičiasi sistemos būsenos koordinačių reikšmės:

– Iš sąrašo *W3L* (šiame sąraše vienetukais pažymimos aptarnaujami užsakymai) atitinkamai išmetamas vienas vienetukas.

– Sąrašo *List* turinys sumažinamas aptarnautų užsakymų skaičiumi *L*.

– Diskrečioji koordinatė *Sn* (į sistemą atėjusių užsakymų skaičius) sumažinama vienetu, o *Nn* (prekių kiekis sandėlyje) padidinama dydžiu *L*, nusakančiu aptarnautų užsakymų skaičių.

4.4.2. Agregatų sujungimo aprašymas

```
qy_Env(Out,W1,X6):-
    procnr(7),
    retract(busena_Env(__)),
    asserta(busena_Env(W1)),
    busena_Ice(W2L,W3L,List,S,N),
    qx_Ice(Out,W2L,W3L,List,S,N,X6),
    !.
```

4.4.3. Analizės aksiomų užrašymas

Bendrųjų savybių užrašymas

Aklavietė

```
bb(X6):-
    busena_Env(0),
    busena_Ice([],[],[],S,N),
    X7 is X6+1,
    nl,
    rod(X6),
    write('-->'),
    write('Aklaviete'),
 !.
```

Čia turi būti nagrinėjama, ar nėra tokios situacijos, kai visos tolydžiosios koordinatės lygios nuliui, nes tokiu atveju pasiekiamų būsenų medis nebeturi kur šakotis, t.y. “pasikabina”.

Apribojimų sąlyga

Čia turi būti nagrinėjama, ar diskrečiosios koordinatės priklauso užduotai reikšmių aibei.

Šiuo atveju diskrečiųjų koordinačių kitimo diapazonas nėra nurodytas (netikrinama apribojimų sąlyga), todėl nepateiksime atitinkamo PROLOG sakinio.

Pilnumo sąlyga

Čia turi būti nagrinėjama, ar pilnai aprašyta reakcija į visus vidinius ir išorinius įvykius.

Šiuo atveju pilnumo sąlyga nėra tikrinama, todėl nepateiksime atitinkamo PROLOG sakinio.

Pabaigiamumas

Čia turi būti nagrinėjama, ar sistemai baigus funkcionuoti, t.y. suformavus visą pasiekiamų būsenų grafą, patenkama į užsiduotą galinę būseną.

Šiuo atveju sistema po tam tikro laiko turi grįžti į savo pradinę būseną, t.y.:

```
ein_busena(1,[],[],[],0,35).
```

Dinaminis ciklas

Čia turi būti nagrinėjama, ar yra toks pasiekiamų būsenų medžio šakojimas, kuriame patenkama į begalinį ciklą.

Šiuo atveju dinaminio ciklo susidarymas nėra tikrinamas, todėl nepateiksime atitinkamo PROLOG sakinio.

Pertekliškumo savybė

```
qt1(W1,W2L,W3L,List,S,N,X6,L) :-
    W1=:=1, !,
    procnr(2),
    X7 is X6+1,
    itraukti(L,qt1(W1,W2L,W3L,List,S,N),Ln),!,
    qw(0,W2L,W3L,List,S,N,X7,Ln),
    !.
procnr(X):-proc(X),!; assertz(proc(X)),!.
pasalintiproc:-
    retract(proc(X)),!,
    pasalintiproc.
pasalintiproc:-!.
```

Čia turi būti nagrinėjama, ar sistemos darbo metu buvo pavartotos visos procedūros ir nėra nei vienos perteklinės.

4.4.4. Individualiosios arba invarianto savybės

Čia turi būti nagrinėjamos įvairios papildomos agregatinės sistemos sąlygos.

Šiuo atveju individualiosios arba invarianto savybės nėra tikrinamos, todėl nepateiksime atitinkamų PROLOG sakinių.

4.4.5. Pradinė būsena

```
ein_busena(1,[],[],[],0,35)
```

Nagrinėjamos sistemos pradinė būsena apibrėžia, kad pradiniu laiko momentu:

- į sistemą gali ateiti naujas užsakymas;
- sistemoje nėra nei vieno užsakymo;
- nei vienas užsakymas nėra aptarnaujamas;
- prekių (knygų) kiekis sandėlyje lygus 35.

4.4.6. Pasiekiamų būsenų grafo formavimo sakiniai

```
qz(G1,G2L,G3L,X6,Nr) :-
      G1=:=1,!,
      X7 is X6+1,
      r(W1,W2L,W3L,List,S,N,Nr),
      ein_b(Nr),
      nl,rod(X6),write('-->'),
      write(Nr), write(' + '), write(W1), write(' - '),
      write(W2L), write(' - '), write(W3L), write(' - '),
      write(List), write(' - '), write(S), write(' - '), write(N),
      nl,rod(X7),write('|'),
      nl,rod(X7),write('W1'),
      nl,rod(X7),write('|'),
      asserta(qr(0,G2L,G3L,X6,Nr)),
      qt_Env(W1,X6),!.


bb(X6):-
     busena_Env(W1),
     busena_Ice(W2L,W3L,List,S,N),
     r(W1,W2L,W3L,List,S,N,Nr),!,
     X7 is X6+1,
     nl,
     rod(X6),     write('-->'),     write(Nr),
     write(' * '), write(W1),  write(' - '), write(W2L),
     write(' - '), write(W3L), write(' - '), write(List),
     write(' - '), write(S), write(' - '), write(N),
     !.
```

Aukščiau pateikti du PROLOG sakiniai atlieka sistemos pasiekiamų būsenų grafo formavimą ir spausdinimą į ekraną. Atitinkamai pirmasis sakinys spausdina naują sistemos būseną, kuri žymima “+”, o antrasis spausdina pasikartojančią būseną, kuri jau yra pasiekiamų būsenų grafe, ją pažymėdamas “*”.

Vykdant PROLOG kalba parašytą programą, papildomai turi būti naudojamos kitos taisyklės, jungiančios ją į vieną visumą (žr. PROLOG kalba parašytą programą, pateiktą prieduose).

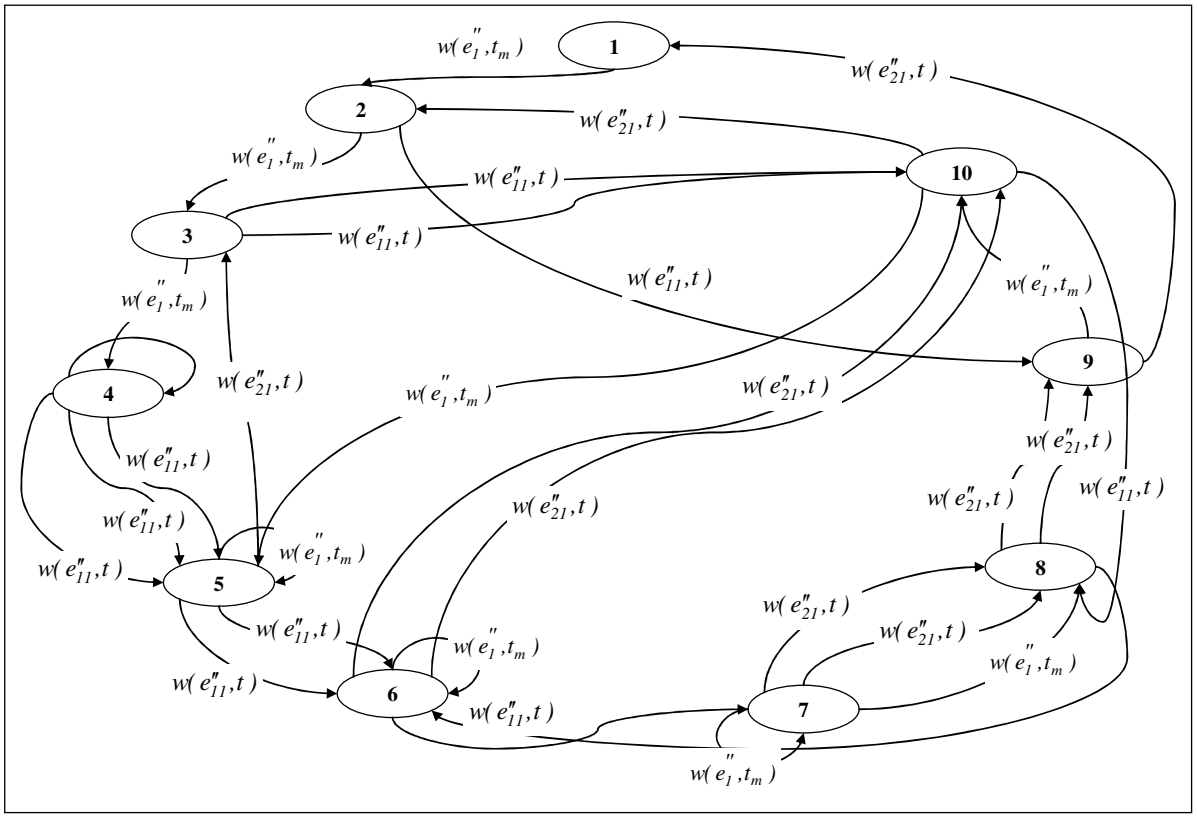
4.5.Pavyzdžio analizės rezultatai

Aukščiau pateikto pavyzdžio specifikacija buvo išanalizuota.

Analizei buvo pasirinktas pasiekiamų būsenų metodas. Jis paremtas visų galimų būsenų generavimu. Specifikacijos analizė atlikta patikrinant bendrąsias savybes: 1) pilnumas, 2) aklaviečių nebuvimas, 3) apribojimų tenkinimas, 4) pertekliškumas.

Atlikus analizę paaiškėjo, kad aprašyta galutinė būsena buvo pasiekta ir atitiko tikrinamų savybių reikalavimus. Tai rodo, kad sudaryta specifikacija yra teisinga.

Pasiekiamų būsenų grafo fragmentas pateiktas 18 paveiksle. Be to, rodyklėms suteikti vardai atitinka įvykius, kurie inicijavo būsenų pasikeitimus.



18 pav. Pasiekiamų būsenų grafo fragmentas

5. Rezultatai ir išvados

Gauti rezultatai:

1. Sukurta metodika, kuri realizuoja sąryšį tarp REA (Resource-Event-Agent) ir PLA (piece-linear aggregate) modelių.

2. Atlikta bendrųjų ir individualiųjų savybių analizė, naudojant PLA (Piece-Linear Aggregate) ir PROLOG kalbos pagrindu sukurtą sistemą, atliekančią agregatinių specifikacijų analizę.

3. Individualių savybių užrašymui pritaikyti vidaus apskaitos kontrolės apribojimai, naudojant reliacinę algebrą, SQL ir PROLOG.

4. Iliustruojant metodiką, užrašytas pardavimo ciklo REA (Resource-Event-Agent) modelis, taikant PLA (Piece-Linear Aggregate) struktūrą ir atlikta sistemos agregatinės specifikacijos analizę.

Išvados:

1. Verslo proceso modeliavimas, imitavimas ir analizė yra alternatyvūs scenarijai verslo procesų reinžinerijoje. Tyrimo rezultatai parodė, kad REA (Resource-Event-Agent) modelis ir PLA (piece-linear aggregate) formalizmas yra naudingi kuriant realias verslo sistemas.

2. Sukurta metodika, parodo sąryšį tarp REA (Resource-Event-Agent) ir PLA (piece-linear aggregate) modelių. Tai reiškia, kad, sudarant formalias specifikacijas ir atliekant analizę galima taikyti REA ir PLA. Be to, programinės įrangos priemonės, kurios remiasi PLA formalizmu leidžia atlikti automatizuotą bendrųjų (pilnumas, aklaviečių nebuvimas, apribojimų tenkinimas, pertekliškumas) ir individualiųjų savybių tikrinimą.

3. Darbe pateikta metodika turėtų būti tyrinėjama toliau. Vėlesniuose tyrimuose turėtų būti ryšys, tarp verslo proceso modelio bei PLA formalizmo, vystomas ir automatizuotas.

6. Literatūros sąrašas

1. Adventure in Prolog [interaktyvus]. Stow, Massachusetts, Dennis Merritt, 1996 balandis. - [žiūrėta 2006-03-10]. Prieiga per internetą: <http://www.amzi.com/AdventureInProlog/>

2. BARJIS JOSEPH, ILKOV ILIAN G. Integrating Business Process Modeling and Simulation. [žiūrėta 2006-07-04]. Prieiga per internetą: http://www.scs.org/getDoc.cfm?id=1112

3. BERGHOLTZ MARIJA; JAYAWEEARA PRASAD; JOHANNESSON PAUL; WOHED PETIA. Process Models and Business Models – a Unified Framework. [žiūrėta 2006-07-04]. Prieiga per internetą: http://openebxml.sourceforge.net/information/papers/ecomo02stockholm.pdf

4. BORCH E. AND STEFANSEN CH. Evaluating the REA enterprise ontology from an operational perspective. 2004, kovas [žiūrėta 2006-07-04]. Prieiga per internetą: http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-125/paper3-title-abstract.pdf.

5. DAIGLE, R.J.; ARNOLD V. Analysis of the research productivity of AIS faculty. International Journal of Accounting Information Systems. 106-122. 2000.

6. DUNN CH. L.; MCCARTHY W. E. The REA Accounting Model: Intellectual Heritage and Prospects for Progress. The Journal of Information Systems, 1997, p. 31-51.

7. GORDIJN JAAP, AKKERMANS HANS, VLIET HANS. Business Modelling is not Process Modelling. [žiūrėta 2006-07-04]. Prieiga per internetą: http://www.cs.vu.nl/ ~gordijn/ecomo-gordijn.pdf

8. GEERTS G.; MCCARTHY W.E. An ontological analysis of the economic primitives of the extended-REA enterprise information architecture. International Journal of Accounting Information Systems, (3):1–16. 2002.

9. GUIDO L. GEERTS; WILLIAM E. MCCARTHY. The Ontological Foundation of REA Enterprise Information Systems. November 1999, March 2000, August 2000.

10. GUIDO L. GEERTS; WILLIAM E. MCCARTHY. Modeling business enterprises as value-added process hierarchies with resource-event-agent object templates. In: Sutherland J , Patel D, Casanave C, Hollowell G, Miller J, editors. *Business object design and implementation*. London: Springer-Verlag, 1997a; 94-113

11. KINSUN TAM. Implementing internal accounting controls as constraints in RDBMN and XML. USA. 2002.

12. MCCARTHY W.E. The REA accounting model: A generalized framework for accounting systems in a shared data environment. *The Accounting Review, LVII(3)*, p. 554–578. 1982.

13. MISEVICIUS P.V.; MISEVICIENE R. Model of accounting systems for enterprises. International Conference „*Modelling and simulation of business systems*, p. 334-338. 2003.

14. MISEVICIUS P. V.; MISEVICIENĖ R. Application of aggregate method and logic programming for machining process planning validation and verification. Mechanical engineering of the Baltic Region. Collection of research papers of the Baltic Association of Mechanical engineering experts No 3. Kaliningrad, Publishing house KSTU, P. 74-77. 2005.

15. POELS G.; MAES A.; GAILLY F.; PAEMELEIRE R. The Pragmatic Quality of Resurces –Event- Agents diagrams: An Experimental Evaluation. Ghent University. Working Paper. 2004.

16. PRANEVICIUS H. Aggregate Approach for Specification and Analysis of Computer Network Protocols / Pranevicius H., Pilkauskas V., Chmieliauskas A., Kaunas, Technologija, 1994. P. 19-52.

17. PRANEVICIUS, H. Formalization and simulation of business process. *Modelling and simulation of business systems. International conference*, p. 198- 202. 2003.

18. PRANEVICIUS, H., MISEVICIUS P. V., MISEVICIENĖ R. Transformation of aggregate specifications to the predicate logic models. The International Workshop on Harbour, Maritme and Multimodal Logistics Modelling and Simulation, p. 378-384. 2003

19. Visual Prolog Version 5.0: Language Tutorial / Prolog Development Center A/S. Copenhagen, Denmark, 1996.

7. Priedai

Pardavimo ciklą aprašančios sistemos PROLOG programos kodas

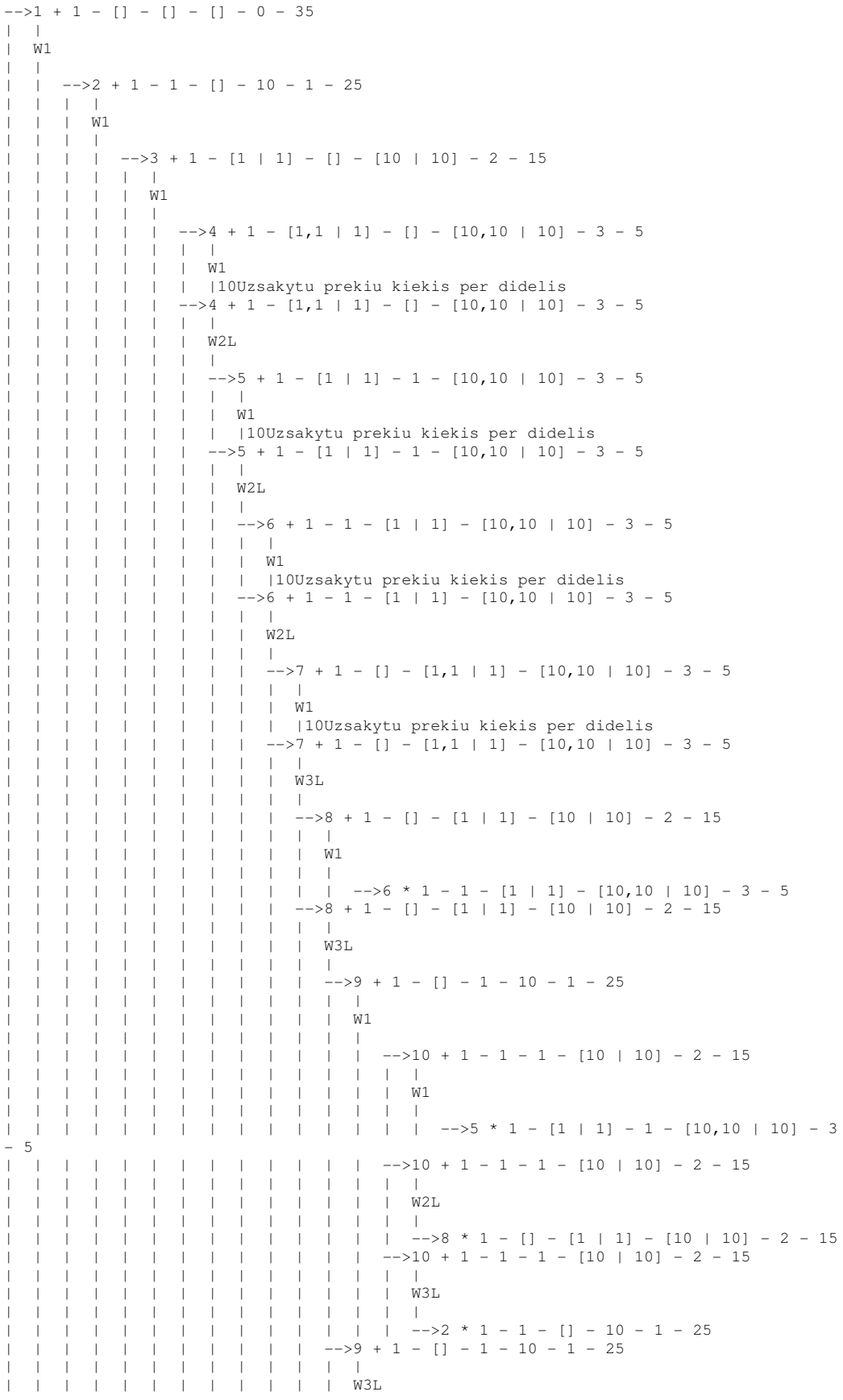
```
seed(13).
random(R,N) :-
   retract(seed(S)),
   N is (S mod R) +1,
   NewSeed is (125*S+1) mod 4096,
   asserta(seed(NewSeed)), !.

itraukti([],L,L).
itraukti(L1,L2,[L2|L1]).

ismesti1(L,L,[]).
ismesti1(_,[_|L],L):-!.

pirm(A,[A|L]):-!.
pirm(A,A):-!.
rod(0):-!.
rod(N):-N1 is N-1, write('|  '), rod(N1).

procnr(X):-proc(X),!;assertz(proc(X)),
!.

qt_Env(W1,X6) :-
   W1=:=1,!,
   procnr(1),
   X7 is X6+1,
   L is 10,
   qy_Env(L,W1,X7),
        !.

qt_Env(W1,X6) :-
   write(2),nl,
   qw_Env(1,X6), !.

qt_Env(W1,X6) :-
   write('Neivykdyta pilnumo salyga qt_Env'), !.

qx_Ice(Inp,W2L,W3L,List,S,N,X6) :-
   N >= Inp,!,
   procnr(3),
   X7 is X6+1,
   L is Inp,
   itraukti(W2L,1,W2Ln),
   itraukti(List,L,Listn),
   Sn is S+1,
   Nn is N-L,
   qw_Ice(W2Ln,W3L,Listn,Sn,Nn,X7),
   !.

% ----------------- individualios savybes tikrinimas ---------

qx_Ice(Inp,W2L,W3L,List,S,N,X6) :-
   write(Inp),
   N < Inp,!,
   write('Uzsakytu prekiu kiekis per didelis'), !.

qx_Ice(Inp,W2L,W3L,List,S,N,X6) :-
   procnr(4),
   write(4), nl,
   qw_Ice(W2L,W3L,List,S,N,X6), !.
```

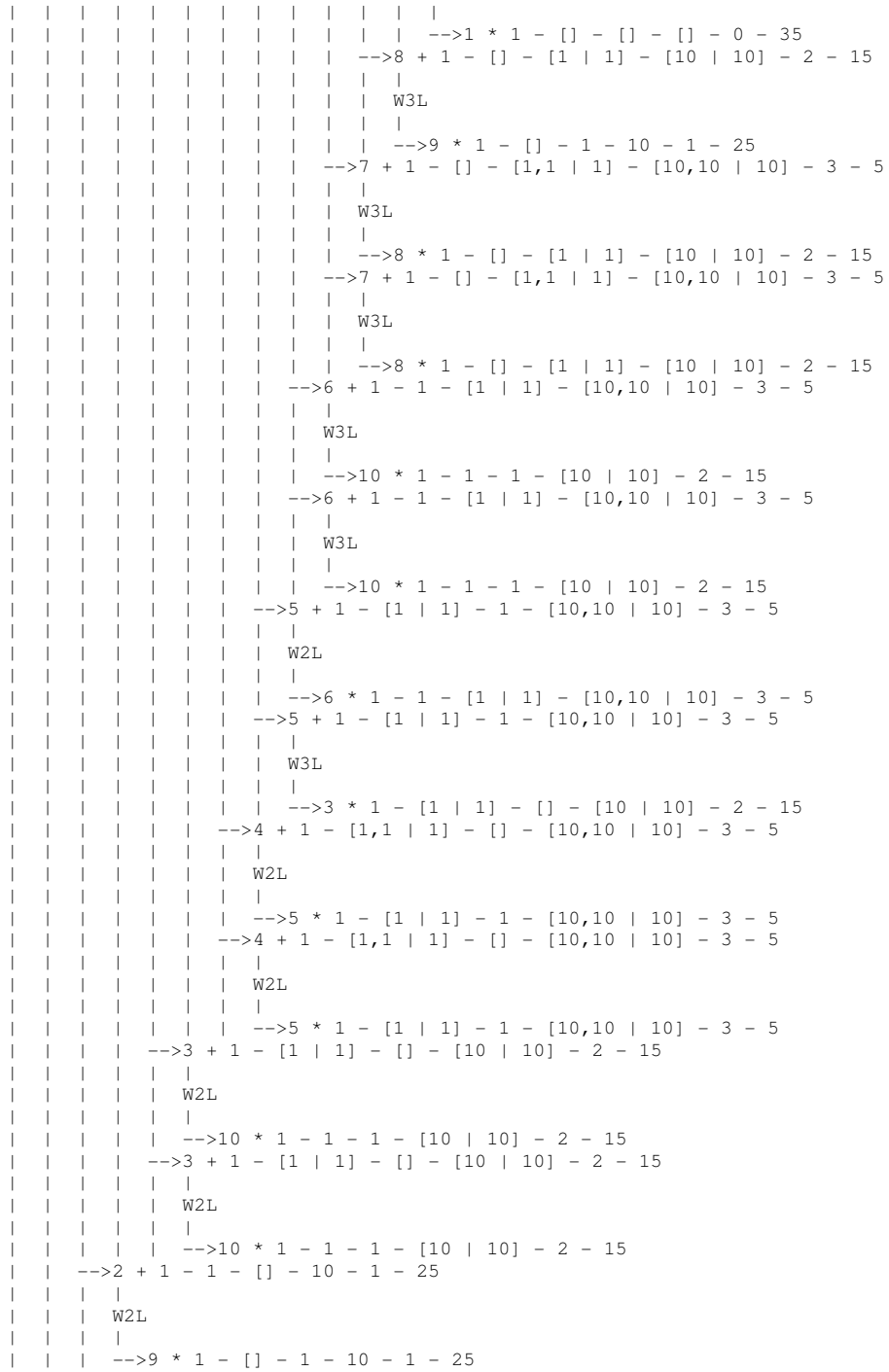
```
qx_Ice(Inp,W2L,W3L,List,S,N,X6) :-
   write('Neivykdyta pilnumo salyga qx_Ice'), !.

qt_Ice2(W2L,W3L,List,S,N,X6):-
   procnr(5),
   W2L\==[],!,
   itraukti(W3L,1,W3Ln),
   ismesti1(1,W2L,W2Ln),
   X7 is X6+1,
   qw_Ice(W2Ln,W3Ln,List,S,N,X7),
   !.

qt_Ice3(W2L,W3L,List,S,N,X6):-
   W3L\==[],!,
   procnr(6),
   X7 is X6+1,
   ismesti1(1,W3L,W3Ln),
   pirm(L,List),
   ismesti1(L,List,Listn),
   Sn is S-1,
   Nn is N+L,
   qw_Ice(W2L,W3Ln,Listn,Sn,Nn,X7),
   !.

qt_Ice2(W2L,W3L,List,S,N,X6):-
   write('Neivykdyta pilnumo salyga qt_Ice'), !.
qt_Ice3(W2L,W3L,List,S,N,X6):-
   write('Neivykdyta pilnumo salyga qt_Ice'), !.

 /*Sujungimu aprasymas*/

qy_Env(Out,W1,X6):-
   procnr(7),
   retract(busena_Env(__)),
   asserta(busena_Env(W1)),
   busena_Ice(W2L,W3L,List,S,N),
   qx_Ice(Out,W2L,W3L,List,S,N,X6),
   !.

bb(X6):-
   busena_Env(0),
   busena_Ice([],[],[],S,N),
   X7 is X6+1,
   nl,
   rod(X6),
   write('-->'),
   write('Aklaviete'), !.

%-----------------------------
bb(X6):-
   busena_Env(0),
   busena_Ice([],[],[],S,N),
   N > L,
   X7 is X6+1,
   nl,
   rod(X6),
   write('-->'),
   write('Klaida: uzsakytu prekiu kiekis didesnis uz prekiu kieki
sandelyje'), !.

%-----------------------------

bb(X6):-
```

```
   busena_Ice(W2L,W3L,List,S,N),
   N > 35,!,
   X7 is X6+1,
   nl,
   rod(X6),
   write('-->'),
   write('Neivykdyta apribojimu salyga'), !.

bb(X6):-
   busena_Env(W1),
   busena_Ice(W2L,W3L,List,S,N),
   r(W1,W2L,W3L,List,S,N,Nr),!,
   write(10), nl,
   X7 is X6+1,
   nl,
   rod(X6),
   write('-->'),
   write(Nr), write(' * '), write(W1), write(' - '), write(W2L),
write(' - '), write(W3L), write(' - '), write(List), write(' - '),
write(S), write(' - '), write(N),
   !.

bb(X6):-
   busena_Env(W1),
   busena_Ice(W2L,W3L,List,S,N),
   r(_,_,_,_,_,_,Nr),
   Nr1 is Nr+1,
   asserta(r(W1,W2L,W3L,List,S,N,Nr1)),
   asserta(qr(W1,W2L,W3L,X6,Nr1)),
   !.

ein_b(Nr) :-
   r(W1,W2L,W3L,List,S,N,Nr),
   retract(busena_Env(_)),
   retract(busena_Ice(_,_,_,_,_)),
   asserta(busena_Env(W1)),
   asserta(busena_Ice(W2L,W3L,List,S,N)),
   !.

qz(G1,G2L,G3L,X6,Nr) :-
   G1=:=1,!,
   write(13), nl,
   X7 is X6+1,
   r(W1,W2L,W3L,List,S,N,Nr),
   ein_b(Nr),
   nl,rod(X6),write('-->'),
   write(Nr), write(' + '), write(W1), write(' - '),
   write(W2L), write(' - '), write(W3L), write(' - '),
   write(List), write(' - '), write(S), write(' - '),    write(N),
   nl,rod(X7),write('|'),
   nl,rod(X7),write('W1'),
   nl,rod(X7),write('|'),
   asserta(qr(0,G2L,G3L,X6,Nr)),
   qt_Env(W1,X6),
   !.

qz(G1,G2L,G3L,X6,Nr) :-
   G2L \== [], !,
   X7 is X6+1,
   r(W1,W2L,W3L,List,S,N,Nr),
   ein_b(Nr),
   ismesti1(1,G2L,G2Ln),
   nl,rod(X6),write('-->'),
```

```
   write(Nr), write(' + '), write(W1), write(' - '),
   write(W2L), write(' - '), write(W3L), write(' - '),
   write(List), write(' - '), write(S), write(' - '),    write(N),
   nl,rod(X7),write('|'),
   nl,rod(X7),write('W2L'),
   nl,rod(X7),write('|'),
   asserta(qr(G1,G2Ln,G3L,X6,Nr)),
   qt_Ice2(W2L,W3L,List,S,N,X6),
   !.

qz(G1,G2L,G3L,X6,Nr) :-
   G3L \== [], !,
   X7 is X6+1,
   r(W1,W2L,W3L,List,S,N,Nr),
   ein_b(Nr),
   ismesti1(1,G3L,G3Ln),
   nl,rod(X6),write('-->'),
   write(Nr), write(' + '), write(W1), write(' - '),
   write(W2L), write(' - '), write(W3L), write(' - '),
   write(List), write(' - '), write(S), write(' - '),    write(N),
   nl,rod(X7),write('|'),
   nl,rod(X7),write('W3L'),
   nl,rod(X7),write('|'),
   asserta(qr(G1,G2L,G3Ln,X6,Nr)),
   qt_Ice3(W2L,W3L,List,S,N,X6),!.
   qz(_,_,_,_,_) :- !.
   qw_Env(W1,X6) :-
   retract(busena_Env(_)),
   asserta(busena_Env(W1)),
   bb(X6),
   !.

qw_Ice(W2L,W3L,List,S,N,X6):-
   retract(busena_Ice(_,_,_,_,_)),
   asserta(busena_Ice(W2L,W3L,List,S,N)),
   bb(X6),
   !.

/* Pradines agregatu busenos35*/
%busena_Env(1).
%busena_Ice([],[],[],0,35).
%act_St(0).

/*vyk:-
   bb(0),
   retract(qr(W1,W2L,W3L,X6,Nr)),
   qz(W1,W2L,W3L,X6,Nr),
   fail.*/

main:- vyk.

ism:- qr(W1,W2L,W3L,X6,Nr),
retract(qr(W1,W2L,W3L,X6,Nr)),!,
qz(W1,W2L,W3L,X6,Nr),!, ism.

ism:-!.

vyk:- asserta(r(0,[],[],[],0,0,0)),
asserta(busena_Env(1)),
asserta(busena_Ice([],[],[],0,35)),
!,
bb(0),
ism.
```

Pardavimo ciklą aprašančios sistemos pasiekiamų būsenų grafas




-->1 * 1 - [] - [] - [] - 0 - 35
-->8 + 1 - [] - [1 | 1] - [10 | 10] - 2 - 15
W3L
-->9 * 1 - [] - 1 - 10 - 1 - 25
-->7 + 1 - [] - [1,1 | 1] - [10,10 | 10] - 3 - 5
W3L
-->8 * 1 - [] - [1 | 1] - [10 | 10] - 2 - 15
-->7 + 1 - [] - [1,1 | 1] - [10,10 | 10] - 3 - 5
W3L
-->8 * 1 - [] - [1 | 1] - [10 | 10] - 2 - 15
-->6 + 1 - 1 - [1 | 1] - [10,10 | 10] - 3 - 5
W3L
-->10 * 1 - 1 - 1 - [10 | 10] - 2 - 15
-->6 + 1 - 1 - [1 | 1] - [10,10 | 10] - 3 - 5
W3L
-->10 * 1 - 1 - 1 - [10 | 10] - 2 - 15
-->5 + 1 - [1 | 1] - 1 - [10,10 | 10] - 3 - 5
W2L
-->6 * 1 - 1 - [1 | 1] - [10,10 | 10] - 3 - 5
-->5 + 1 - [1 | 1] - 1 - [10,10 | 10] - 3 - 5
W3L
-->3 * 1 - [1 | 1] - [] - [10 | 10] - 2 - 15
-->4 + 1 - [1,1 | 1] - [] - [10,10 | 10] - 3 - 5
W2L
-->5 * 1 - [1 | 1] - 1 - [10,10 | 10] - 3 - 5
-->4 + 1 - [1,1 | 1] - [] - [10,10 | 10] - 3 - 5
W2L
-->5 * 1 - [1 | 1] - 1 - [10,10 | 10] - 3 - 5
-->3 + 1 - [1 | 1] - [] - [10 | 10] - 2 - 15
W2L
-->10 * 1 - 1 - 1 - [10 | 10] - 2 - 15
-->3 + 1 - [1 | 1] - [] - [10 | 10] - 2 - 15
W2L
-->10 * 1 - 1 - 1 - [10 | 10] - 2 - 15
-->2 + 1 - 1 - [] - 10 - 1 - 25
W2L
-->9 * 1 - [] - 1 - 10 - 1 - 25

ĮMONĖS VERSLO SISTEMOS ONTOLOGIJA

Živilė Janušauskaitė, Laisvūnas Muralis ir Regina Misevičienė,

KTU, Informatikos fakultetas, Verslo informatikos katedra,

Studentų 56-301, 3028 Kaunas

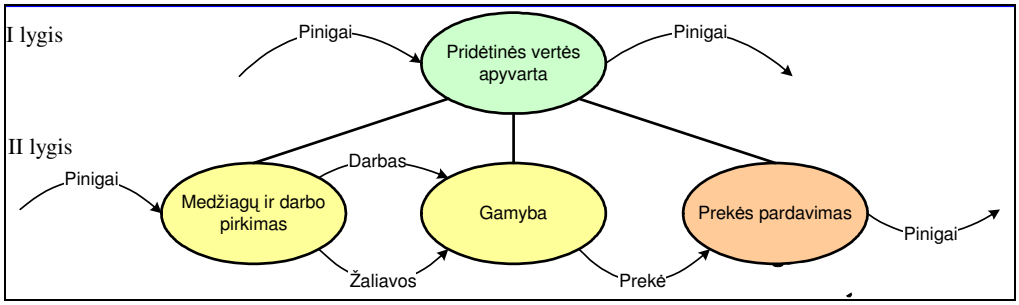
Straipsnyje aprašoma įmonių informacijos sistemų ontologija pagal REA (**R**esource-**E**vent-**A**gent) modelį, kurį išplėtojo Geerts ir McCarthy (2000). Aptariamos dvi REA ontologijos kategorijos: fiziniai objektai ir abstrakcijos, kurias pasiūlė Sowa [3]. Išskiriami trys jų lygiai. Pirmame nusakomi statiniai ir dinaminiai objektai. Antrame aprašoma, kaip pirmojo lygio kategorijos yra susietos su antro lygio objektais. Trečio lygio tikslas – apibrėžti ir logiškai pagrįsti antro lygio kategorijose esančius ir tarpusavyje susietus primityvus. Straipsnyje parodyta, kaip abstrakčios kategorijos naudojamos, aiškinant kai kuriuos fizinius elementus (tipizavimas) arba abstrakčius (charakterizavimas).

1 Įvadas

Neseniai, ontologijos tapo pagrindiniu tyrimo objektu dirbtinio intelekto ir žinių valdymo srityse. Skiriamos dvi ontologijų rūšys: nepriklausomos (subject-independent) ir tam tikros srities (domain ontologies) ontologijos. *Nepriklausomomis* vadinamos aukštesnio lygio (upper-level) ir apibrėžia sąvokas, bendras visoms sritims. Jų tikslas aprašyti sąvokas, kurios taikomos daugelyje sričių, pvz., laikas ir erdvė. *Srities* ontologijos apibrėžia sąvokas, skirtas konkrečiai taikymo sričiai. Šiame darbe apibrėžiama įmonių informacijų sistemų ontologija pagal REA modelį, kurį išplėtojo Geerts ir McCarthy (2000).). Aptariamos dvi REA ontologijos kategorijos: fiziniai objektai ir abstrakcijos, kurias pasiūlė Sowa [3]. Išskiriami trys jų lygiai. Pirmame nusakomi statiniai ir dinaminiai objektai. Antrame aprašoma, kaip pirmojo lygio kategorijos yra susietos su antro lygio objektais. Trečio lygio tikslas – apibrėžti ir logiškai pagrįsti antro lygio kategorijose esančius ir tarpusavyje susietus primityvus. Straipsnyje parodyta, kaip abstrakčios kategorijos naudojamos, aiškinant kai kuriuos fizinius elementus (tipizavimas) arba abstrakčius (charakterizavimas).

2 Įmonės verslo scenarijus: REA modelis

Iš esmės visos įmonės dirba vienodai (žr. 19 pav.) [1]. Randamas pradinis kapitalas ir įvedamas į ekonominių santykių su kitais rinkos dalyviais (tiekėjais ar samdomais darbuotojais) grandinę, kiekvieną kartą keičiant ekonominį resursą (pavyzdžiui, pinigai) į kitą, jo manymu, didesnę vertę turintį resursą. Vertė nustatoma, pagal produkto ar paslaugos firmos klientams patrauklias savybes.



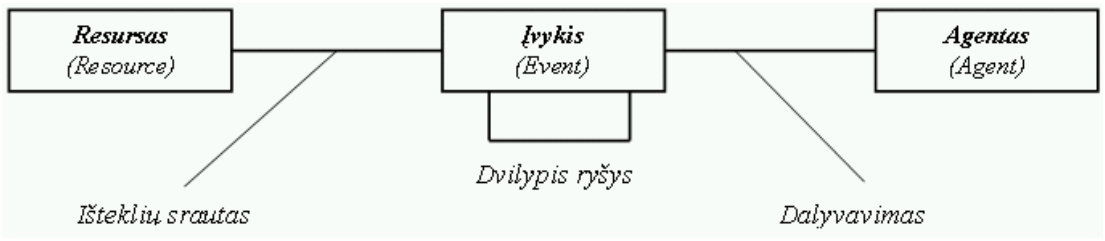
19 pav. Įmonės verslo scenarijus

Pagrindinis procesas „Pagrindinės vertės apyvarta“ (pirmas lygis) susideda iš trijų antro lygio procesų, kurių kiekvienas turi ekonominius resursus įėjime ir išėjime.

Šie procesai apskaitoje vadinasi:

- įsigijimo ciklas – pinigai keičiami į darbą ir žaliavas;
- apdirbimo ciklas – darbas ir žaliavos virsta į baigtą produkciją;
- pelno ciklas – baigta produkcija vėl keičiama į pinigus.

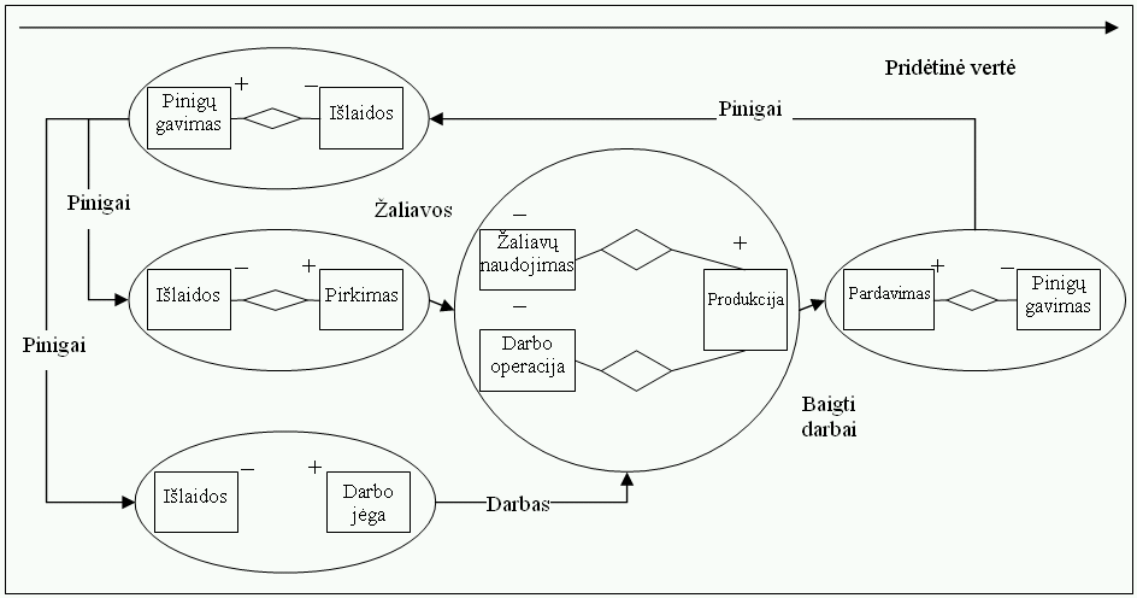
Šiems verslo procesams aprašyti **McCarthy** [2,4] pasiūlė REA (**R**esource(resursas)-**E**vent(įvykis)-**A**gent(agentas)) modelį.



20 pav. Supaprastintas REA modelis

20 paveiksle REA modelis išreikštas atskirais objektais ir ryšiais tarp jų, panaudojus UML (Unified Modeling Language) notaciją. REA parodo tris būdingus objektus, dalyvaujančius mainuose: *įvykius*, *resursus* (kurie yra mainų komponentai) ir tarpininkaujančius *agentus*. *Ekonominis įvykis* yra verslo informacijos sistemos pagrindinis elementas, įtakojantis įėjimo ir išėjimo resursų srautus. Ekonominio įvykio rezultatas yra arba resursų įplaukos, arba jų panaudojimas. Natūralias mainuose esančias neatskiriamas tarpusavio operacijas vaizduoja *dvilypis ryšys* tarp ekonominio įvedimo (resursų didinimo) ir ekonominio išvedimo (resursų mažinimo) įvykių. *Išteklių srautas* paaiškina ryšį tarp ekonominių resursų ir ekonominių įvykių (panaudojimas, tiekimas, gamyba). *Dalyvavimo* ryšys aprašo agentus, dalyvaujančius ekonominiame įvykyje.

Įmonės verslo scenarijus, kuris buvo parodytas 19 paveiksle, REA modelyje aprašomas kaip pavaizduota 21 paveiksle. Šiame modelyje medžiagų ir darbo išteklių pirkimo, gamybos ir pardavimo verslo procesai pavaizduoti kaip REA objektų visuma. Kiekviename verslo procese yra prieaugio įvykis (išteklių gavimas, pažymėtas + ženklu), kuris susijęs su išlaidų įvykiu (išteklių vartojimas, pažymėtas - ženklu).



21 pav. Įmonės verslo REA modelis

3 REA ontologija

Įmonės ontologijos tikslas – apibrėžti bendrus ekonominius reiškinius, vykstančius įmonės komercinėje veikloje. Pagal Sowa [3] skiriamos dvi ontologijų kategorijos: *fiziniai objektai* ir *abstrakcijos*. Fiziniai objektai aprašo realius reiškinius, o abstrakcijos yra informacinės struktūros, kurios charakterizuoja fizinius objektus.

Fiziniai objektai pasižymi tęstinumo (statiniai objektai) arba vyksmo (dinaminiai objektai) savybėmis. *Tęstinumas* – ilgalaikis objektas, kuris pasižymi statiniais atributais, įgalinančiais įvairiose jo formose ir laiko etapuose atpažinti, kaip tą pačią esybę. *Vyksmas* apibūdinamas, kaip procesas arba įvykis, esantis nuolat kintančioje laiko atžvilgiu būsenoje ir identifikuojamas pagal savo buvimo vietą tam tikruose laiko intervaluose.

Objektų apibrėžimui išskiriami trys lygiai. Pirmame lygyje išskiriami objektai, nepriklausantys nuo bet kokių išorinių ryšių. Pavyzdžiui, žmonių poaibis (vyras arba moteris), kas paprastai yra intuityviai ir aiškiai suvokiama (pirmas lygis), nepriklausantis nuo bet kokių išorinių ryšių. *Moters* tipas gali būti atpažintas pagal savo individualias savybes, nepaisant įvairių ryšių su kitomis esybėmis. To paties individo klasifikacija gali būti traktuojama ir kitais aspektais – įvedami konceptualūs tipai *Mama*, *Žmona*, tačiau, klasifikacija pagal šiuos tipus priklauso nuo išorinių ryšių (antras lygis) su kitomis esybėmis, tokiomis, kaip *Vaikas*, *Vyras*. Trečias lygis apibrėžiamas tarpine grandimi, užtikrinančia ryšius tarp pirmo ir antro lygio objektų. Mūsų atveju motinystė

apima kelis dalykus, tokius, kaip gimdymas, auklėjimas, kurie riša motiną su vaiku. Žmoną ir vyrą sieja santuokiniai ryšiai.

McCarthy [1] apibrėžė REA ontologines kategorijas, kurios pateiktos 4 lentelėje.

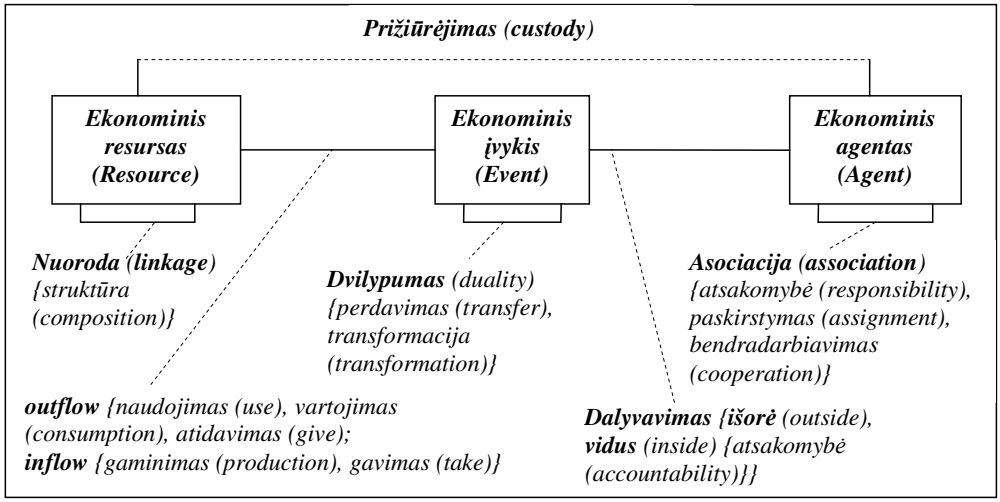
Pirmame lygyje apibrėžiami nepriklausomi objektai. Objektai klasifikuojami į statinius – *Agentas (A)* ir *Resursas(R)* ir dinaminius objektus – *Ekonominis įvykis (E)* ir *Įsipareigojimas (C)*. Tradicinės McCarthy [1] suformuluotos REA modelio kategorijos buvo išplėstos *Įsipareigojimo* kategorija. Pastaroji skiriasi nuo *Įvykio*, nes reikalauja skirtingų įsipareigojimų iš biznio partnerių. *Įsipareigojimo* pavyzdys galėtų būti vietų viešbutyje rezervavimas.

Abstrakčios kategorijos (4 lentelės dešinėje) aprašo fizines kategorijas (4 lentelės kairiuose stulpeliuose). Pirkėjo tipo (*Agento tipas - AT*) pavyzdys – *didelės kredito teikimo rizikos pirkėjas* ir *mažos kredito teikimo rizikos pirkėjas*. Apmokėjimo (*Įvykio tipas- ET*) tipo pavyzdys yra apmokėjimo tipas (*Įvykio tipas*). Apmokėjimas, pavyzdžiui, gali būti kreditine kortele.

4 lentelė. REA ontologijos kategorijos

	Fizinės kategorijos		Abstrakčios kategorijos	
	Statiniai objektai	Dinaminiai objektai	Statiniai objektai	Dinaminiai objektai
1 lygis	*Agentas (A)* *Resursas (R)*	*Ekonominis įvykis* (E) *Įsipareigojimas (C)*	*Agento tipas (AT)* *Resurso tipas (RT)*	*Įvykio tipas (ET)* *Įsipareigojimo tipas (CT)*
2 lygis ***(Ryšių)***	*Asociacija (A-A)* *Prižiūrėjimas (A-R)* *Nuorodos (R-R)*	*Išteklių srautas (E-R)* *Dvilypumas (E-E)* *Atskaitomybė (E-A)* *Vykdymas (C-E)* *Dalyvavimas (C-A)* *Rezervavimas (C-R)* *Abipusiškumas (C-C)*	*Tipizavimas (A-AT); (R-RT)* *Charakterizavimas (AT-AT); (AT-RT) (RT-RT)*	*Tipizavimas (E-ET); (C-CT)* *Charakterizavimas (ET-RT); (ET-ET); (ET-AT); (CT-ET); (CT-AT); (CT-RT); (CT-CT)*
3 Lygis ***(Tarpinis)***	*Struktūra (atsakomybė, partnerystė, konfigūracija)*	*Situacija (mainai, patvirtinimas, užsakymas, planavimas)*	*Priežastis(segmentacija, politika, pakeičiamumas, papildomumas, konfigūracija)*	*Tikslai (standartizacija, politika, strategija)*

Antrame lygyje aprašoma, kaip pirmo lygio kategorijos yra susietos su antro lygio objektais. Išskiriami trys statiniai: *Asociacija (A-A)*, *Prižiūrėjimas(A-R)* ir *Nuoroda (R-R)* ir dinaminiai: *Išteklių srautas (E-R), Dvilypumas(E-E), Atskaitomybė (E-A), Vykdymas (C-E), Dalyvavimas (C-A), Rezervavimas (C-R), Abipusiškumas (C-C)* ryšiai (žr. 2 pav.).



22 pav. Ryšių kategorijos

Prižiūrėjimas pavyzdžiu galėtų būti ryšys tarp sandėlio darbuotojo ir inventoriaus, už kurį jis atsakingas. *Nuorodos* jungimo pavyzdys – ryšys tarp produkcijos atskirų komponentų ir jų būsenos baigtame gaminti produkte.

Asociacija (association) nusako statinius ryšius tarp agentų. Skiriami trys šio ryšio tipai: *atsakomybė* (responsibility), *priskyrimas* (assigment), *bendradarbiavimas* (cooperation). *Atsakomybę* McCarthy [4] apibūdina taip: „aukštesnio lygio vienetai kontroliuoja ir yra atsakingi už žemesnio lygio veiksmus“ (pvz. toks ryšys galėtų egzistuoti tarp direktoriaus, vadybininko, sandėlininko ir kasininko). *Priskyrimo* ryšys nusako priklausomybes tarp vidinio ir išorinio agentų (pvz. „pardavėjas yra atsakingas už pirkėją arba supirkėjas – už

tiekėjus“). *Bendradarbiavimas* aprašo priklausomybes tarp išorinių agentų (pvz. „tam tikras vienos organizacijos klientas gali būti kitos organizacijos (pardavėjo) partneris“).

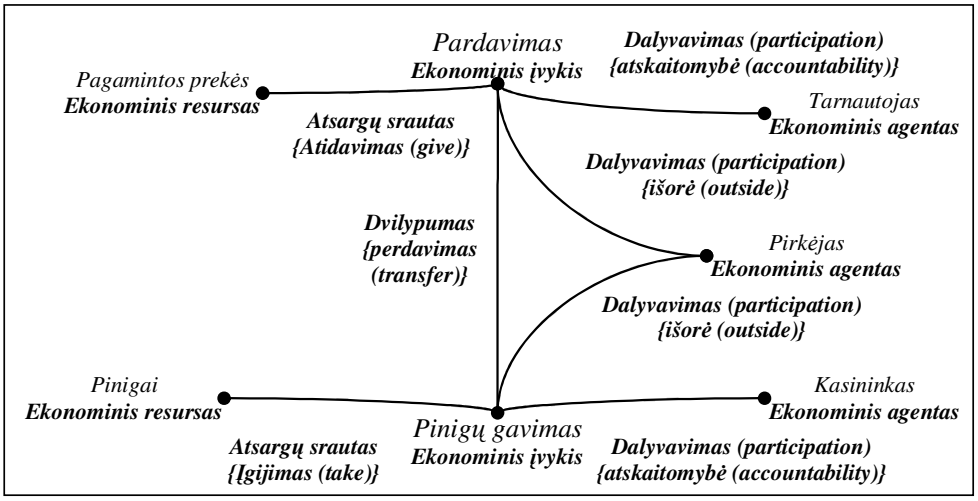
Nuoroda (Linkage) aprašo priklausomybes tarp resursų. Svarbus šio ryšio tipas yra *sudėtis* (composite). Toks ryšys nusako resursą, kaip visumą kitų resursų (pvz. asmeninio kompiuterio dalys – monitorius, kietas diskas, klaviatūra, pelė – gali būti aprašomos atskiromis dalimis).

Prižiūrėjimas (Custody) nurodo vidinį agentą, atsakingą už tam tikrą resursą (pvz., toks ryšys galėtų būti tarp sandėlininko ir prekių, saugomų sandėlyje).

Dvilypumo (Duality) ryšys, nusako ryšį tarp dviejų ekonominių įvykių. Fisher [2] išskyrė du dvilypumo tipus: *pasikeitimas* (transformation) ir *perkėlimas* (transfer). Galime sakyti, kad įvyko pasikeitimas, kai tam tikras resursas proceso metu virsta kitu (pvz. iš molio nulipdomi ąsočiai). Šis tipas dažniausiai naudojamas gamyboje. *Perkėlimo* sąvoka aiškinama resurso atidavimu išorei (pvz., nulipdyti ąsočiai parduodami, t.y. tampa pirkėjo nuosavybe).

Ryšys tarp ekonominių resursų ir įvykių vadinamas *atsargų srautu* (Stock-Flow). išskiriami penki atsargų srautų tipai: *naudojimas* (use), *vartojimas* (consumptin), *atidavimas* (give), *gamyba* (production) ir *gavimas* (take). Resursas yra naudojamas įvykus pasikeitimui, t.y tam tikros žaliavos gamybos (production) metu virsta gauta produkcija. *Naudojimas* nuo *vartojimo* skiriasi tuo, kad pirmuoju atveju resursas pakeičia savo formą, o antruoju – sumažėja jo kiekis. Pardavimo metu *atiduodame* gaminius ir *gauname* pinigus. Tas pats resursas gali dalyvauti keliuose ryšiuose (pvz. mašina pirmiausia yra įsigijama (take), po to ji panaudojama (use) gamyboje (production) ir, pagaliau, gali būti parduota (give)).

Ryšių kategorijų pavyzdys, atspindintis mainus tarp parduodamų prekių ir gaunamų pinigų, parodytas 23 paveiksle.



23 pav. Prekių pardavimo REA modelis

Abstrakčios kategorijos naudojamas aiškinant kai kuriuos fizinius elementus (*Tipizavimas*), arba abstrakčius (*Charakterizavimas*). 24 paveiksle parodyti *tipizavimas* ir *charakterizavimas*.

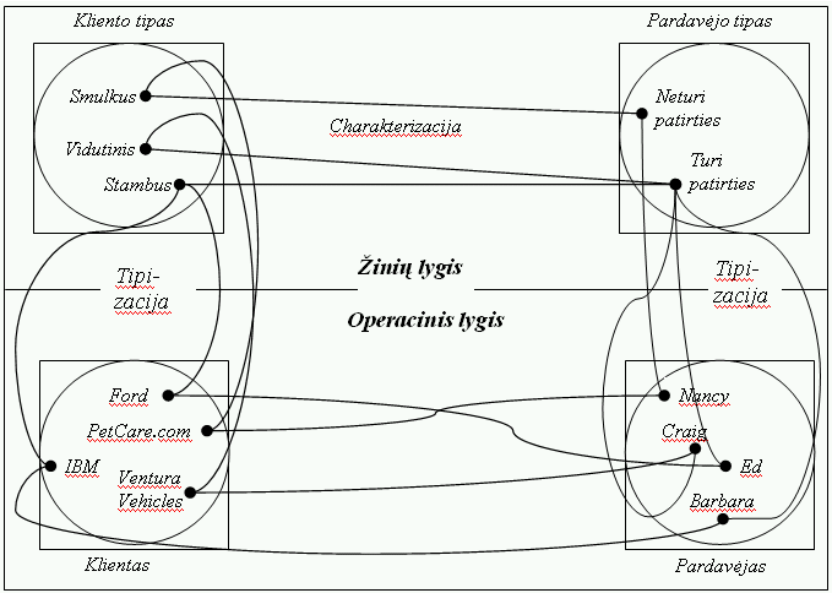
Tipizavimas padeda įtraukti fizinius objektus į abstrakčias informacijos struktūras (pavyzdžiui, aprašyti pirkėją, kaip mažą, vidutinę, ar didelę įmonę). Naudojamos klasės: *Resource-ResourceType ir Agent–AgentType.*

Charakterizavimas gerai atspindi įmonės politiką (24 pav., rodo, kad patyrę pardavėjai aptarnauja vidutines ir dideles organizacijas, o neturintys patirties – smulkias). Naudojamos trys charakterizacijos kategorijos: *AgentType-AgentType, AgentType-ResourceType, ir ResourceType-ResourceType*.

Trečio lygio tikslas – aiškiai apibrėžti ir logiškai pagrįsti antro lygio kategorijose esančius ir tarpusavyje susietus primityvus.

Struktūros kategorijos, pavyzdžiui riša tam tikros įmonės vadovą ir su turtu, ir su dirbančiais įmonėje žmonėmis bei paaiškina, kodėl dvi išorinės įmonės vienijasi ir formuoja asociaciją.

Situacija apibrėžia pagrindines priežastis, kodėl kai kurie elementai (*Resursas*, *Įvykis* ir *Agentas*) yra susieti tokiais ryšiais, kaip *išteklių* srautas, *atskaitomumas* ir *dvilypumas*: arba tarp dviejų nepriklausomų agentų vykdomi apsikeitimai rinkoje, arba reikia pereiti prie proceso, apsiribojus darbu be agento.



24 pav. Tipizavimo ir charakterizavimo pavyzdžiai

Priežasties kategorijos numato resursų ir agentų grupavimą į tokias abstrakčias kategorijas, kaip „aukšto atsparumo produktai“, arba „nemokūs užsakovai“ pagrindimą arba „kodėl įmonės linkusios derinti resursus tarpusavyje“.

Tikslas pagrindžia ekonominių įvykių aprašymą, kaip žaliavos išeigą, o įsipareigojimus, kaip žaliavos pareikalavimą, nes tai leidžia numatyti terminus, resursų kiekius ir kokių agentų reikės suplanuotam įvykiui. Šie nuspėjami veiksniai įgalina formuoti politiką ir strategiją, kurios pasireiškia įvairių tipų biznio procesuose, kai vadovai bando nustatyti geriausią optimalų resursų ir žmonių paskirstymą.

4 Baigiamosios pastabos

Šiuo metu tikslingos REA realizacijos labai mažai. Tokios kompanijos, kaip PricewaterhouseCoopers ir IBM įdiegę REA standartus, kaip architektūrinius kriterijus sudarant buhalterines sistemas, o SES Software pritaikė juos savoje biznio objektų sudarymo architektūroje BOMA. Vis dėlto nei viename iš išvardintų atvejų REA modelis nerealizuotas pilnai. REA yra aukštos kokybės apskaitos modelis, bet jo populiarumą lėmė apskaitos profesionalų konservatyvus požiūris. Atsiradus ERP sistemoms ir vystantis objektinei architektūrai, artėja buhalterinių infrastruktūrų, orientuotų į vertės grandinę realizacija (REA principu). Tiekimo grandinės bendradarbiaujant Internetu ir Internetinė prekyba gali rasti didesnę REA modelio pritaikymų sferą ir būti palankiau vertinamas. Daugeliu atveju REA modelis yra daug geresnis, nei kiti konkuruojantys semantiniai modeliai.

5 Literatūros sąrašas

[1] Guido L. Geerts, William E. McCarthy. The Ontological Foundation of REA Enterprise Information Systems. November 1999, March 2000, August 2000.

[2] Guido L. Geerts, William E. McCarthy. Modeling business enterprises as value-added process hierarchies with resource-event-agent object templates. In: Sutherland J , Patel D, Casanave C, Hollowell G, Miller J, editors. *Business object design and implementation*. London: Springer-Verlag, 1997a; 94-113.

[3] Sowa J. Conceptual structures: Information processing in mind and machine. Reading, MA: Addison-Wesley, 1984.

[4] William E. McCarthy The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment, The Accounting Review (July 1982) pp. 554-78

6 Summary

The ontology of enterprise business system

The paper describes ontology of REA enterprise information systems, that Geerts and McCarthy (2000) expanded. There are defined the two categories by Sowa [3] of REA ontology: physical objects and abstractions. There are three levels of them. The first level defines static and dynamic objects. The second presents how first level categories are related with objects of the second level. The third level objective is to define and logically valid relative primitives in the second level categories. The paper describes how abstracts categories are used to explain some physical elements (typification) and abstracts (characterization).

Respublikinės konferencijos „Informacinės technologijos“ 2-ojo pranešimo medžiaga (2006 m.)

Verslo procesų, aprašytų REA metodu, analizė panaudojant agregatinį formalizmą

Herikas Pranevičius, Regina Misevičienė, Živilė Janušauskaitė

KTU, Informatikos fakultetas, Verslo informatikos katedra

Studentų g. 56-301, LT-3028 Kaunas

Įmonės verslo procesams aprašyti taikomas REA modelis, kuris naudojamas atvirų sistemų sąveikavimo standarto ISO/IEC 14662 funkciniu lygmeniu. Minėtą standartą patvirtino Tarptautinė prekybos vystymo ir elektroninio verslo organizacija UN/CEFACT 2003 metais. Pagal REA modelį sudaromi formalūs verslo procesų aprašymai, tačiau tokių aprašymų teisingumo analizei reikia panaudoti programines priemones. Šiame straipsnyje pasiūlyta REA modelio pagrindu sudarytų specifikacijų analizei panaudoti programines priemones, sukurtas agregatinio formalizmo pagrindu.
Analizę sudaro verslo procesų transformavimas į agregatinį modelio aprašą bei sudarytos specifikacijos bendrųjų savybių tikrinimas.

1 Įvadas

Pagrindinė problema, su kuria dažniausiai susiduria apskaitos informacijos sistemų kūrėjai, – kaip sukurti tokią informacijos sistemą, kuri pilnai ir patikimai aprūpintų apskaitos informacija įvairius jos vartotojus, valdančius įmonės ūkinę veiklą bei ją kontroliuojančius. Dauguma verslo sistemų yra labai didelės ir sudėtingos, dėl to sistemų kūrėjams keblu visapusiškai įvertinti jų veiklą. Todėl būtina sudaryti įmonės veiklą atspindinčius modelius, kurie supaprastintų realios įmonės veiklos aprašymą.

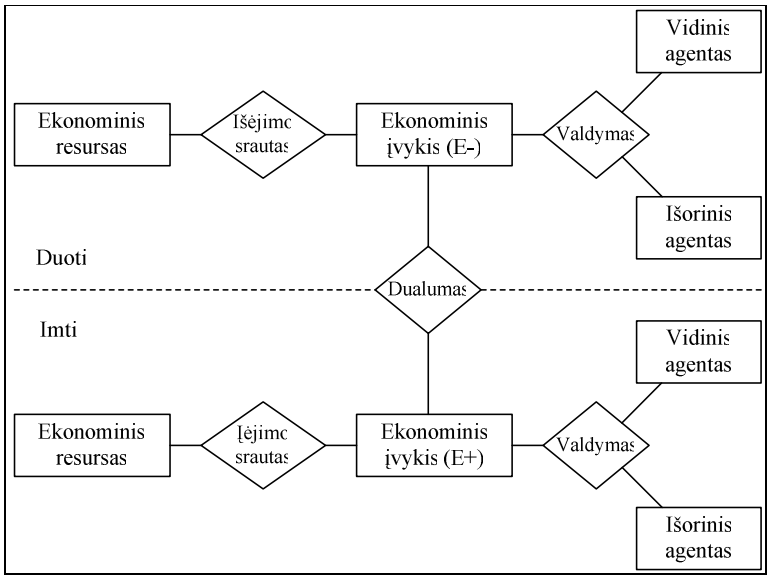
Vienas iš tokių modelių yra **R**esursų-**Į**vykių-**A**gentų (REA) modelis, kuris taikomas įmonės verslo procesams aprašyti [5]. Šis modelis naudojamas atvirų sistemų sąveikavimo standarto ISO/IEC 14662 funkciniu lygmeniu. Minėtą standartą patvirtino Tarptautinė prekybos vystymo ir elektroninio verslo organizacija UN/CEFACT 2003 metais. Literatūroje REA modelis minimas ir aptariamas dažnai – jis yra svarbus informacijos sistemų semantiniam modeliavimui [2, 4, 14, 15], tačiau modelio taikymas sistemų teisingumo analizei nagrinėtas palyginti mažai [1, 3, 8, 13].

Šiame straipsnyje naudojamas vienas iš sudėtingų sistemų aprašymo modelių – vadinamasis PLA (piece-linear aggregate) [9] formalizmų-atkarpomis tiesinių agregatų modelis. Agregatinio metodo panaudojimas suteikia priemones ne tik formalių specifikacijų sudarymui, bet ir jų modeliavimui bei teisingumo analizei [6, 7, 10, 11, 12]. Agregatinėje specifikacijoje analizuojama sistema aprašoma kaip tarpusavyje sąveikaujančių agregatų visuma (1 pav.), o valdančių sekų metodas naudojamas aprašyti agregatų funkcionavimui. Tokių agregatinių sistemų analizei dažnai naudojamas pasiekiamumo metodas. Pagal šį metodą yra sudaromas agregatinės sistemos globalių būsenų orientuotas grafas. Šio grafo viršūnės - tai agregatinės sistemos globalios būsenos, o rodyklės parodo galimus perėjimus tarp jų. Grafo viršūnėse ir yra tikrinamos agregatinės sistemos savybės: 1) pilnumas; 2) statinių aklaviečių nebuvimas; 3) pertekliškumas; 4) dinaminių aklaviečių susidarymas; 5) apribojimų tenkinimas bei kitos. Individualių savybių tikrinimui užrašomi invariantai.

Šiame straipsnyje pasiūlyta REA modelio pagrindu sudarytų verslo procesų analizei panaudoti programines priemones, sukurtas agregatinio formalizmo pagrindu. Analizės metu verslo procesų REA modelis užrašomas panaudojant agregatinį formalizmą, o sudarytos specifikacijos bendrosios savybės tikrinimos naudojant PLA analizės priemones.

2 REA MODELIS

Straipsnyje taikysime **R**esursų-**Į**vykių-**A**gentų (REA) modelį (pasiūlytą McCarthy [5]).



25 pav. REA modelis pagal [5]

REA modelis (1 pav) išreiškiamas atskirais objektais ir ryšiais tarp jų, panaudojus UML (Unified Modeling Language) notaciją. REA parodo tris būdingus objektus, dalyvaujančius mainuose: *įvykius*, *resursus* (kurie yra mainų komponentai) ir tarpininkaujančius *agentus*. *Ekonominis įvykis* yra verslo informacijos sistemos pagrindinis elementas, veikiantis įėjimo ir išėjimo resursų srautus. Ekonominio įvykio rezultatas yra arba resursų įplaukos, arba jų panaudojimas. Natūralias mainuose esančias neatskiriamas tarpusavio operacijas vaizduoja *dvilypis ryšys* tarp ekonominio įvedimo (resursų didinimo) ir ekonominio išvedimo (resursų mažinimo) įvykių. *Išteklių srautas* paaiškina ryšį tarp ekonominių resursų ir ekonominių įvykių (panaudojimas, tiekimas, gamyba). *Dalyvavimo* ryšys aprašo agentus, dalyvaujančius ekonominiame įvykyje.

REA modelio panaudojimas verslo procesų specifikavimui giliau analizuotas straipsnyje [17].

3 REA TRANSFORMAVIMAS Į PLA

Šiame skyriuje REA modelio sakinius transformuosime į PLA.

Taikant agregatinį požiūrį, sumodeliuota sistema yra aprašoma sąveikaujančių agregatų visuma. Agregatinė sistema S, sudaryta iš n elementų yra:

$S = (A_1 , A_2 , ..., A_n , R)$, čia $A_1 , A_2 , ..., A_n$ yra sistemos agregatai; R atitinka kanalų, jungiančių agregatus, aibę.

Kiekvienas agregatas A yra sistema:

$A = (X , Y , E' , E'', Z , H , G)$, čia:

X yra įėjimo signalų aibė;

Y yra išėjimo signalų aibė;

Z yra išplėstų baigtinių būsenų aibė: $Z = D \cup W$, čia D $D = \{ d_1 , d_2, ... , d_p \}$ yra diskrečių koordinačių aibė ir $W = \{ w_1 , w_2, ... , w_f \}$ yra tolydžių koordinačių aibė;

E' yra išorinių įvykių klasė. Šie įvykiai siejami su įėjimo signalais ($E' \rightarrow X$);

E'' vidinių įvykių klasė. Šie įvykiai siejami su tolydžiosiomis koordinatėmis ($E'' \rightarrow W$);

$H : E \times Z \rightarrow Z$ yra būsenų perėjimo operatorius, čia $E = E' \cup E''$;

$G : E \times Z \rightarrow Y$ yra išėjimo operatorius.

Transformuojant REA teiginius į PLA, reikia aprašyti visus rinkinius (žr. 1 lentelę).

REA transformavimas į PLA

1 lentelė

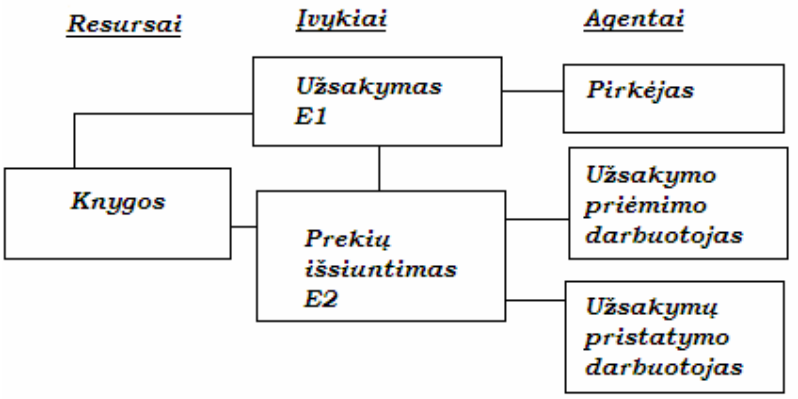
Aibės	REA taikymas	PLA taikymas
A	Aprašomi du kontaktuojantys agentai A_i ir A_j (ryšys tarp agentų $r_{ij} \in$R).	Aprašomi du agregatai A_i,A_j , sujungti kanalais $r_{ij} \in$R.

R	Kontaktuodami jie siunčia bendravimo dokumentus.	Kontaktuodami agregatai vienas kitam siunčia signalus.
Y **X**	Vienas agentas A_i siunčia dokumentą y_i ($y_i \in$Y) tuo tarpu kitas agentas A_j gauna dokumentą x_i ($x_i \in$X).	Vienas iš agregatų A_i siunčia signalą y_i ($y_i \in$Y). Tuo tarpu kitas agregatas A_j gauna signalą x_i ($x_i \in$X).
E'	Gautas dokumentas iš išorinio agento sukelia vidinio agento įvykį e'_i (e'$_i \in$E'). Pavyzdžiui, gautas dokumentas *Užsakymas* iš išorinio agento inicijuoja *užsakymo priėmimo* įvykį.	Iš išorinio agregato gautas signalas inicijuoja išorinį agregato įvykį e'_i (e'$_i \in$E'). Pavyzdžiui, gautas signalas apie užsakymą inicijuoja išorinį įvykį – užsakymo priėmimas.
E"	Vidinio agento įvykiai sukelia kitų vidinių agentų įvykius $e"_i$ (e" $\in$E").	Vidinių įvykių operacijos inicijuoja kitų vidinių įvykių operacijas $e"_i$ (e"$_i \in$E").
W	Įvykio pasirodymo laiko momentą (datą) aprašo koordinatė w_i ($w_i \in$W).	Įvykio pasirodymo laiko momentą aprašo valdymo koordinatė w_i ($w_i \in$W).
Z	Įvykiai gali veikti resursų (prekių ar pinigų) kiekio pasikeitimus.	Įvykiai gali veikti būsenos koordinačių pasikeitimus (pvz., informacija apie prekes ar pinigus).
D	Informaciją apie resursus aprašo diskreti koordinatė d_i ($y_i \in D$) (prekių kiekis arba pinigų kiekis).	Informaciją apie resursus aprašo diskreti koordinatė d_i ($y_i \in D$) (pvz., prekių kiekis ar pinigų kiekis).
H	Diskrečių koordinačių pasikeitimai priklauso nuo įvykių. Jų priklausomybes aprašo H išraiškos (pvz., prekių kiekis sandėlyje sumažėjo). Pasibaigus įvykiui, išoriniam agentui gali būti siunčiamas dokumentas y_i, ($y_i \in$Y).	Diskrečių koordinačių pasikeitimai priklauso nuo įvykių. Jų priklausomybes aprašo H operatorius (pvz., prekių kiekis sandėlyje sumažėjo). Pasibaigus operacijai, išoriniam agregatui gali būti siunčiamas signalas y_i, ($y_i \in$Y).
G	Dokumento formavimą aprašo G išraiška.	Signalo formavimą aprašo G operatorius.

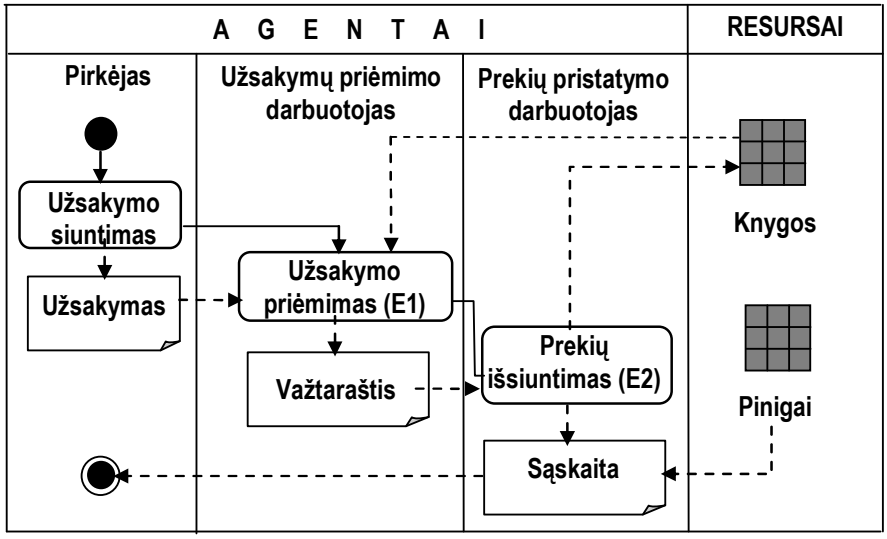
4 REA VERSLO PROCESO, NAUDOJANT PLA, ANALIZĖS PAVYZDYS

a. PAVYZDŽIO APRAŠAS, TAIKANT REA MODELĮ

Šis pavyzdys aprašo pardavimo verslo procesą, naudojant elektroninį katalogą. *Pirkėjas* užsisako knygas iš minėto katalogo. Šiuo atveju *Pirkėjas,* nevykdamas į parduotuvę, gali peržiūrėti *Pardavėjo* siūlomas prekes. *Pirkėjas* siunčia *Užsakymą Pardavėjui*. Jei užsakymas priimamas, *Pardavėjas* pristato knygas *Pirkėjui*. Nagrinėjamame pavyzdyje prekiaujama tik vienos rūšies knygomis. Vaizduojamas pardavimo procesas be apmokėjimo įvykio. Klasių bei veiklų diagramos pavaizduotos 2-3 paveiksluose.



26 pav. REA klasių diagrama

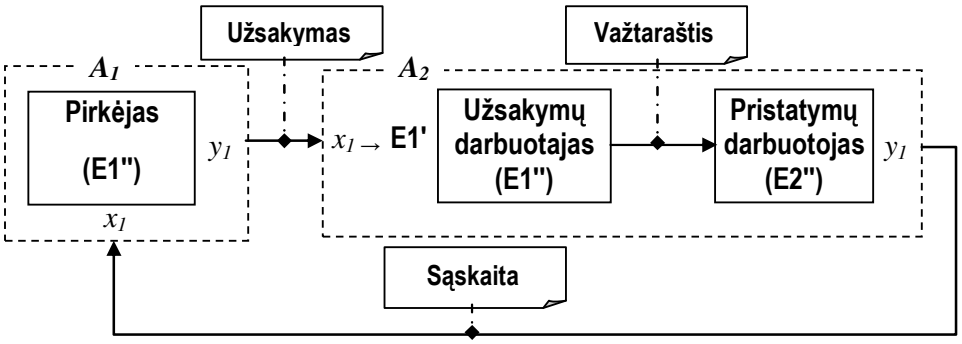


27 pav. Detali veiklų diagrama

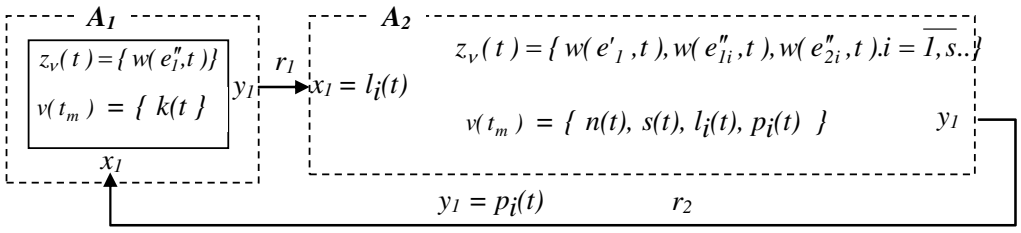
4.2 AGREGATINĖ SPECIFIKACIJA

Agregatinę specifikaciją sudaro sistema $S = (A_1 , A_2, R)$, čia A_1 , A_2 yra sistemos agregatai; R atitinka du kanalus (r_1 ir r_2), jungiančius agregatus (žr. 4-5 pav.).

Žemiau pateiktas formalus agregatų aprašymas.



28 pav. REA transformacija į PLA



29 pav. Agregatinė sistema

Agregato A1 formalus aprašymas

1. Įėjimų aibė: $X = \{ x_1 \}$.
2. Išėjimų aibė: $Y = \{ y_1 \}$.
3. Išorinių įvykių aibė $E^{'} = \{e'_1\}$;

   e'_1 - įvykis, aprašantis sąskaitos atėjimo įvykį;
4. Vidinių įvykių aibė $E^{''} = \{e''_1\}$,

   čia e''_1 - įvykis, aprašantis užsakymo formavimą,

5. Valdymo sekos:

   $e_1'' \rightarrow \tau_1, \tau_2, \ldots$, čia τ_i aprašo i-ojo įvykio trukmę.

6. Diskrečioji būsenos $v(t_m) = \{ k(t_m) \}$ komponentė, čia

   $k(t_m)$ - knygų kiekis užsakyme (momentu t)

7. Tolydžioji būsenos komponentė:

   $z_v(t) = \{ w(e_1'', t) \}$, čia

   $w(e_1'', t)$ aprašo momentą, kai prasideda įvykis.

8. Pradinė būsena:

   $w(e_1'', t_0) = t_0 + \tau_1 ; k(t_m) = 0$.

9. Perėjimo operatoriai:

   $H(e_1'):$.

   $H(e_1''):$

   $k(t_m) = 1 + random(5);$

   $w(e_1'', t_m) = t_m + \tau_m.$

   $G(e_1''):$

   $y_1(t_m) = k(t_m).$

Formalus agregato A2 aprašymas:

2. Įėjimų aibė: $X = \{ x_1 \}$.

3. Išėjimų aibė: $Y = \{ y_1 \}$.

4. Išorinių įvykių aibė $E' = \{ e'_1 \}$;

   e'_1 - įvykis, aprašantis užsakymo atėjimą į A2 agregatą;

5. Vidinių įvykių aibė $E'' = \{ e''_{11}, e''_{21}, e''_{12}, e''_{22}, \ldots, e''_{1s}, e''_{2s} \}$,

   čia:

   e''_{1i} - įvykis, aprašantis i-ojo užsakymo vykdymo pradžią;

   e''_{2i} - įvykis, aprašantis i-ojo užsakymo vykdymo pabaigą (prekių pristatymo įvykis);

6. Valdymo sekos:

   $e''_{1i} \rightarrow \xi_i$, čia ξ_i reiškia užsakymo operacijos vykdymo trukmę;

   $e''_{2i} \rightarrow \zeta_i$, čia ζ_i reiškia prekių pristatymo operacijos vykdymo trukmę.

7. Diskrečioji būsenos $v(t_m) = \{ n(t), s(t), l_i(t), p_i(t) \}$ komponentė, čia

   $n(t)$ – prekių kiekis sandėlyje (laiko momentu t);
   $s(t)$ – užsakymų kiekis (laiko momentu t);
   $l_i(t)$ – i-ojo užsakymo prekių kiekis (laiko momentu t);
   $p_i(t)$ – i-ojo užsakymo vertė (laiko momentu t).

8. Tolydžioji būsenos komponentė:

   $z_v(t) = \{ w(e''_{11}, t), w(e''_{21}, t), w(e''_{12}, t), w(e''_{22}, t), \ldots, w(e''_{1s}, t), w(e''_{2s}, t) \}$, čia

   $w(e''_{1i}, t)$ - laiko momentas, kai i-asis užsakymas bus pradėtas vykdyti;

$w(e''_{2i}, t)$ - laiko momentas, kai i-asis užsakymas bus baigtas vykdyti (pradėtas prekių pristatymas).

9. Pradinė būsena:

$$n(t_0) = 35;\ s(t_0) = 0;$$

$$p_i(t_0) = 0;\ l_i(t_0) = 0; i = \overline{1,s}$$

$$w(e''_{1i}, t_0) = \infty;$$

$$w(e''_{2i}, t_0) = \infty.$$

10. Perėjimo operatoriai:

$$H(e'_1):$$

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

$$l_{s(t_m)} = x_1;$$

$$n(t_m) = \begin{cases} n(t_{m-1}) - l_{s(t_m)}, jeigu\ n(t_{m-1}) - l_{s(t_m)} >= 0; \\ n(t_{m-1}), priešingu\ atveju; \end{cases}$$

$$w(e''_{1s(t_m)}, t_m) = \begin{cases} t_m + \xi_m, jeigu\ n(t_{m-1}) - l_{s(t_m)} >= 0; \\ \infty, priešingu\ atveju; \end{cases}$$

$$H(e''_{1i});\ i = \overline{1,s}:$$

$$w(e''_{2i}, t_m) = t_m + \zeta_m;$$

$$p_i(t_m) = 2.5 * l_i(t_{m-1});$$

$$H(e''_{2i});\ i = \overline{1,s}:$$

$$s(t_m) = s(t_{m-1}) - 1.$$

$$G(e''_{2i});\ i = \overline{1,s}:$$

$$y_I(t_m) = p_i(t_m)$$

Sujungimai tarp agregatų yra pateikti 2 lentelėje.

Agregatų sujungimai

2 lentelė

Kanalas	*Agregatas_i*	*Išėjimo signalas*	*Agregatas_j*	*Įėjimo signalas*
1.	AG1	y_1	AG2	x_1
2.	AG2	y_1	AG1	x_1

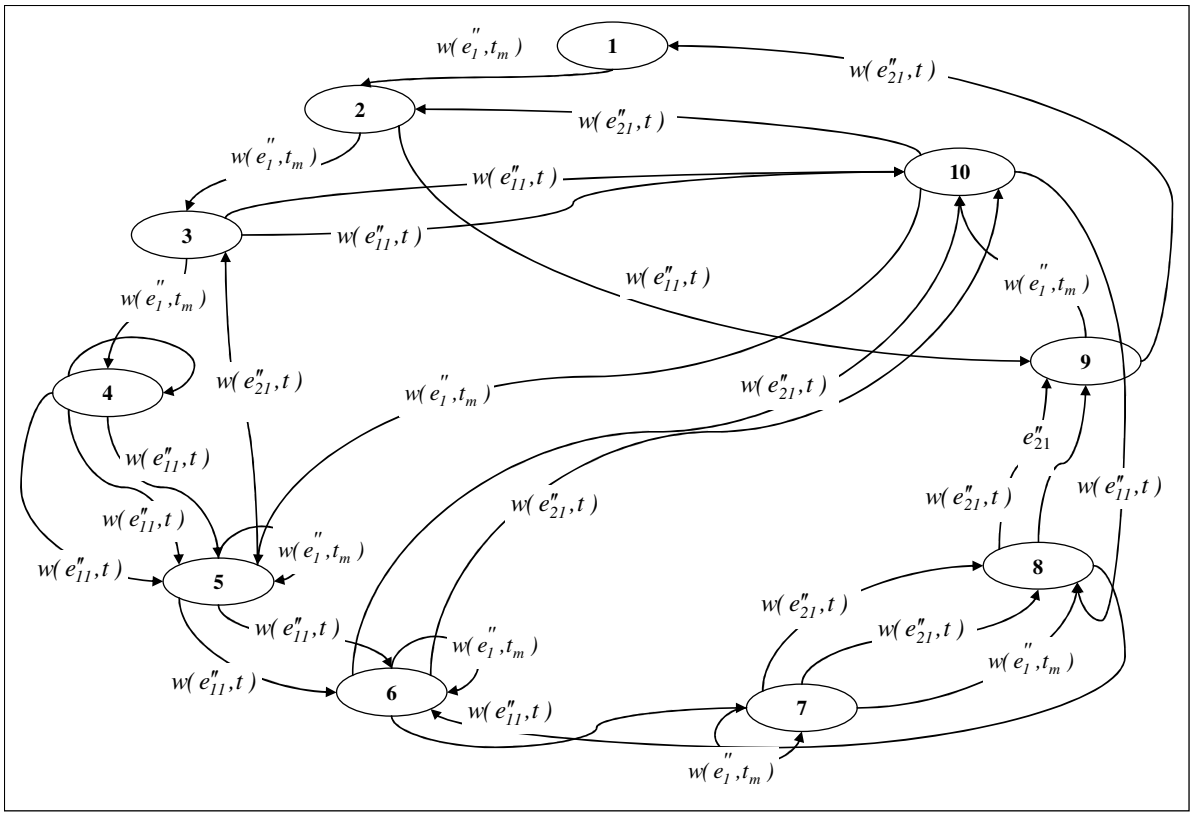
4.3 ANALIZĖS REZULTATAI

Aukščiau pateikto pavyzdžio specifikacija buvo išanalizuota.

Analizei buvo pasirinktas pasiekiamų būsenų metodas. Jis paremtas visų galimų būsenų generavimu. Specifikacijos analizė atlikta patikrinant bendrąsias savybes: 1) pilnumas, 2) aklaviečių nebuvimas, 3) apribojimų tenkinimas, 4) pertekliškumas.

Atlikus analizę paaiškėjo, kad aprašyta galutinė būsena buvo pasiekta ir atitiko tikrinamų savybių reikalavimus. Tai rodo, kad sudaryta specifikacija yra teisinga.

Pasiekiamų būsenų grafo fragmentas pateiktas 6 paveiksle. Be to, rodyklėms suteikti vardai atitinka įvykius, kurie inicijavo būsenų pasikeitimus.



30 pav. Pasiekiamų būsenų grafo fragmentas

5 IŠVADOS

Tarp REA modelio ir PLA agregatinio formalizmo yra sąryšis, aptartas šiame straipsnyje. Tokiu sąryšiu remiasi galimybė naudoti PLA formalizmą formalioms specifikacijoms ir REA modelio operacinei interpretacijai. Programinė įranga, sukurta pagal PLA formalizmą, užtikrina automatizuotą bendrųjų (aklaviečių nebuvimas, pilnumas, nepakankamumas) ir individualiųjų savybių analizę.

Straipsnyje aprašyta tik bendrųjų specifikacijos savybių analizė. Individualiųjų savybių analizė REA modelio gali būti atlikta taikant invariantus. Užrašant apskaitos kontrolės sąlygas invariantais, galima būtų sumažinti klaidų kiekį.

6 Literatūros sąrašas

[1] **Borch E. and Stefansen Ch.** *Evaluating the REA enterprise ontology from an operational perspective*. Submitted to INTEROP '04. March, 2004.

[2] **Dunn Ch. L. and McCarthy W. E**. *The REA Accounting Model: Intellectual Heritage And Prospects For Progress*. The Journal of Information Systems (Spring), pp. 31-51. 1997.

[3] **Daigle, R.J., Arnold, V**. *Analysis of the research productivity of AIS faculty.* International Journal of Accounting Information Systems. 1 106-122. 2000.

[4] **Geerts G. and McCarthy W.E**. *An ontological analysis of the economic primitives of the extended-REA enterprise information architecture*. International Journal of Accounting Information Systems, (3):1–16. 2002.

[5] **McCarthy W.E**. *The REA accounting model: A generalized framework for accounting systems in a shared data environment.* The Accounting Review, LVII(3):554–578. 1982.

[6] **Misevicius P.V., Misevicienė R**. *Model of accounting systems for enterprises* //International Conference „Modelling and simulation of business systems". Vilnius, Lithuania, May 13-14. P. 334-338. 2003.

[7] **Misevicius P. V., Misevicienė R.** *Application of aggregate method and logic programming for machining process planning validation and verification*. Mechanical engineering of the Baltic Region. Collection of research papers of the Baltic Association of Mechanical engineering experts No 3. Kaliningrad, Publishing house KSTU, P. 74-77. 2005.

[8] **Poels G., Maes A., Gailly F. Paemeleire R**. *The Pragmatic Quality of Resurces –Event-Agents diagrams: An Experimental Evaluation*. Ghent University. Working Paper. 2004.

[9] **Pranevicius H**. *Aggregate approach for specification, validation, simulation and implementation of computer network protocols*. Lecture notes in Computer Science No502, Springer–Verlag, pp.433–477. 1991.

[10] **Pranevicius, H**. *Formalization and simulation of business process. Modelling and simulation of business systems.* International conference. Vilnius, Lithuania. 2003.

[11] **Pranevicius, H., Misevičius P. V., Misevičienė R.** *Transformation of aggregate specifications to the predicate logic models* //The International Workshop on Harbour, Maritme and Multimodal Logistics Modelling and Simulation: HMS 2003, September 18-20, 2003, Riga, Latvia. P. 378-384.

[12] **Pranevicius H, Miseviciene R, Milciute V**. *Use of aggregate specification and logic programming for knowledge base validation and verification*. //International Conference „Modelling and simulation of business systems". Vilnius, Lithuania, May 13-14. P. 203-208. 2003.

[13] **Samuels, J.A., Steinbart, P.J**. *The Journal of Information systems: A review of the first 15 Years*. The Journal of Information systems 16(2) 97-116. 2002.

[14] **Smith J. D, Gerard G. J, McCarthy W.E**. *Design Science: Building the Future of AIS*. 2001. http://www.msu.edu/user/mccarth4.

[15] **UN/CEFACT**. *Modeling Methodology (UMM) User Guide 5 CEFACT/TMG/N093*. United Nations Centre for Trade Facilitation and Electronic Business. 2003.

[16] Živilė Janušauskaitė, Laisvūnas Muralis ir Regina Misevičienė. Įmonės verslo sistemos ontologija. „Informacinės technologijos 2005". Kaunas.

USING PIECE–LINEAR AGGREGATE MODEL FOR RESOURCE-EVENT-AGENT BUSINESS PROCESS ANALYSIS

In this paper we employ model-checking method a piece–linear aggregate (PLA) approach for analysis of business processes templates created on the ground of the REA model. The analysis contains translation of the business processes templates to aggregate approach and then analysis of general properties of the aggregate specification.