**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

**P175D003 INFORMATIKOS TEORIJA**

**E. VERSLO SISTEMŲ MODELIAVIMAS**

**Referatas**

Atliko: dokt. Aleksejus Michalkovič

Priemė: prof. H. Pranevičius

**KAUNAS, 2011**

Turinys

[1. E. verslas 3](#_Toc294026245)

[2. DEVS modelis 5](#_Toc294026246)

[2.1 Klasikinis DEVS modelis 5](#_Toc294026247)

[2.2 Dinaminis DEVS modelis 8](#_Toc294026248)

[2.3 Mobilieji agentai ir MDEVS modelis 10](#_Toc294026249)

[2.4 MDEVS modelio taikymo e. komercijoje pavyzdys 13](#_Toc294026250)

[3. PLA modelis 14](#_Toc294026251)

[3.1 Klasikinis PLA modelis 14](#_Toc294026252)

[3.2 PLA modelio imitavimo algoritmas 15](#_Toc294026253)

[3.3 Dinaminis PLA modelis 16](#_Toc294026254)

[3.4 dynPLA modelio taikymo e. komercijoje pavyzdys 19](#_Toc294026255)

[Apibendrinimai 20](#_Toc294026256)

[Literatūra 21](#_Toc294026257)

# E. verslas

Verslas jau nuo seniausių laikų supa žmones. Šiuolaikiniame gyvenime nėra tokio žmogaus, kuris nebūtų vienokiu ar kitokiu būdu susijęs su verslu. Visi mes esame bankų klientai, lankomes parduotuvėse, dirbame pelno siekiančiose organizacijose ir panašiai. Tačiau kitoje daugumai kasdieninių žmonių nežinomoje verslo pusėje vyksta ne tik sudėtingi matematiniai procesai, bet ir tokie verslo procesai kaip strateginis bei kooperacinis valdymas, marketingas, gamyba ir t.t. Žinoma, versle, kaip ir daugelyje kitų sryčių, yra susidūriama su ypač dideliu dokumentų kiekiu. Akivaizdu, kad jau praeito amžiaus viduryje buvo ieškomos galimybės, kurios leistų žymiai sumažinti popierių kiekį. Tokiu būdu galima pasiekti ne tik pigesnį, paprastesnį bei greitesnį duomenų apsikeitimo lygį, bet ir palengvinti bendravimą tarp verslo partnerių. Tai ir buvo elektroninio verslo (e. verslo) pradžia.

E. verslas – tai nėra verslas tiesiogine prasme. E. verslas yra suprantamas, kaip verslo modelis, kuris automatizuoja verslo procesus, duomenų apsikeitimą bei komecinę transakciją naudojant informacines sistemas. E. verslo modeliai leidžia suprasti atskirus modeliuojamos srities elementus, sąsają tarp jų, bei leidžia modifikuoti šiuos elementus. Dažnai verslo analitikai analizuoja formalius e. verslo modelius. Tai leidžia jiems pasigilinti į verslo sritį.

Tačiau kas gi iš tikrųjų yra verslo modelis? Otto Petrovic, Christian Kittl ir Ryan D. Teksten savo straipnyje pabrėžia, kad ši savoka yra viena iš labiausiai diskutuojamų, tačiau kartu ir mažiausiai suprantamų. Vieną iš geriausių šios savokos apibrėžimų pateikė P. Timmers‘as savo 1998 metų straipsnyje. Verslo modelis yra suprantamas kaip konceptualinis bei architektūrinis verslo strategijos taikymas. Verslo modelis yra verslo procesų taikymo pagrindas.

ICT įtampa

Planavimo lygis

Strategija

Archtektūrinis lygis

E. verslo galimybės

Verslo modelis

E. verslo procesų adaptacija

Verslo procesas

Taikymo lygis

**1 pav. Verslo logikos trikampis**

Verslo procesai yra naudojami ne tik e. versle, bet ir e. komercijoje. Verslo procesas – tai tarpusavyje surištų uždavinių sistema, kuri yra naudojama paslaugoms arba prekems sukurti. Dažnai atskiri verslo procesai yra jungiami į vieną bendrą procesą. Paprasčiausias verslo proceso pavyzdys yra šiuo metu ypač populiarios internetinės parduotuvės. 2 paveiksle pavaizduota Amazon.com internetinės parduotuvės verslo proceso schema:



**2 pav. Verslo procesas**

Reikia suprasti ir tai, kad e. verslas yra, kaip ir įprastas verslas, gana sudėtinga dinaminė sistema. Tai reiškia, kad ši sistema yra tokia, jog gali vienu metu teikti paslaugas kelioms e. verslo sudedomosioms dalims (pvz. verslo partneriams, klientams, darbotojams ir pan.). Todėl atliekant e. verslo modeliavimą būtina įvertinti ir sistemos dinamiškumą.

Dinaminė sistema yra tokia sistema, kurios būsiena kinta realiame laike priklausomai nuo ja veikiančių veiksnių. Tačiau pastaruoju metu ypač populiarėja tokios sistemos, kai veiksniai gali sukelti ne tik būsienos, bet ir pačios struktūros pakeitimus. Tai reiškia, kad sistemoje gali atsirasti papildomų komponentų, ryšių tarp jų, o kai kurie komponentai gali būti ir pašalinti iš sistemos. Tokio tipo sistemos dažnai yra vadinamos dinaminėmis su kintančia struktūra. Vienas iš tokių sistemų pavyzdžių yra mobiliųjų agentų sistemos. Modeliuojant šias sistemas reikia aprašyti kiekvienos struktūrą fiksuotu laiko momentų, kurios vėliau yra sujungiamos į bendrą modelį.

Pastaruoju metu yra siekiama formalizuoti dinaminių sistemų su kintančią struktūra modeliavimą. Dažnai tam yra naudojami automatų teorija pagrįsti DEVS ir jam artimas PLA matematiniai modeliai. Šių modelių modifikacijos taip pat įvertina sistemos dinamiškumą.

# DEVS modelis

DEVS (angl. Descrete Event System Specification) – tai formalus diskrečiais įvykiais valdomos sistemos modelis. Jis leidžia aprašyti taikomus praktikoje modulius naudojant blokais naudojant hierarchinį požiūrį. Pirmą kartą DEVS modelis buvo apibrėžtas aštuntame praeito amžiaus dešimtmetyje kaip priemonė, kuri buvo skirta diskretiems įvykiams modeliuoti. Realioji sistema, sumodeliuota taikant DEVS, gali būti aprašyta kaip submodulių (blokų) kompozicija. DEVS blokai gali būti dviejų tipų: atominiai ir jungtiniai. Šie blokai taip pat yra DEVS modeliai.

## Klasikinis DEVS modelis

Kiekvieną atominį DEVS modelį apibrėžia septynios komponentės:

čia X – įviesčių įvykių aibė, S – būsienų aibė, Y – išvesčių įvykių aibė, δint – vidinio perėjimo funkcija, δext – išorinio perėjimo funkcija, λ – išvesties funkcija, ta – laiko funkcija.

Kiekvienas DEVS modelis gali turėti ryšių su kitais modeliais. Šiuos ryšius įtakoja įvesties ir išvesties įvykiai. Kai modelis gauna įvesties duomenis, išorinio perėjimo funkcija nusako, koks bus sistemos elgesys esant tokiems duomenims. Vėliau, kai baigiasi būsienos gyvavimo trūkmė (tą nusako laiko funkcija), vidinio perėjimo funkcija sukūria vidinį įvykį ir sistema pereina į kitą būsieną. Pasiekti rezultatai yra persiunčiami kitiems modeliams išvesties funkcijos pagalba. Taip gaunami jungtiniai modeliai.

Jungtinis DEVS modelis gali apjungti kaip atominius, taip ir kitus jungtinius modelius. Formaliai šį modelį aprašo tokia struktūra:

čia X – įviesčių įvykių aibė, Y – išvesčių įvykių aibė, D – indeksų aibė. Kiekvienam Mi – yra atomonis arba jungtinis DEVS modelis, o Ii – i-tojo modelio poveikių aibė. Taip pat kiekvienam Zij – yra perėjimo nuo modelio Mi prie modelio Mj funkcija.

Matome, kad jungtinį modelį galima apibrėžti kaip bazinių komponenčių (atominių arba jungtinių DEVS modelių), kurios yra tarpusavyje surištos, aibę. Kiekvieno modelio Mi poveikis taip pat nusako, koks modelis Mj turi gauti rezultatą. Tam yra sukūriama i-tojo modelio poveikių aibė, kurią sudaro modelių, kuriems yra skirti atitinkamo modelio rezultatai, indeksai. Šie indeksai nusako kokiam modeliui bus persiunčiami kito modelio gauti išvesties duomenys. Perėjimo funkcija Zij modelio Mi išvesties duomenis paverčia modelio Mj įvesties duomenimis.

Taigi matome, kad jungtinis DEVS modelis turi hierarhinę struktūrą, kurią galima pavaizduoti taip:

Atominis modelis

Jungtinis modelis

Jungtinis modelis

Atominis modelis

Atominis modelis

Atominis modelis

Atominis modelis

Atominis modelis

Jungtinis modelis

Atominis modelis

Atominis modelis

Atominis modelis

Jungtinis modelis

Jungtinis modelis

Atominis modelis

Jungtinis modelis

Atominis modelis

Atominis modelis

Atominis modelis

Atominis modelis

Atominis modelis

Atominis modelis

Atominis modelis

Atominis modelis

Atominis modelis

Atominis modelis

**3 pav. Jungtinio DEVS modelio struktūros pavyzdys**

Matematine prasme jungtinio DEVS modelio struktūra yra panaši į grafą. Grafo viršūnės yra DEVS modeliai, o briaunos – yra ryšiai tarp šių modelių.

Kadangi sujungus daugybę išvien veikiančių komponenčių daugelis vidinių perėjimų vyksta tuo pačiu metu, tai yra įvedama papildoma funkcija *select* : 2D → D (čia D – unikalių komponenčių vardų aibė), kurios pagalba nuosekliose imitavimo sistemose yra sprendžiama kolizijų problema.

DEVS modelių imitavimas yra vykdomas pagal Zeigler pasiūlytą abstraktaus imitatoriaus koncepciją. Šis principas kiekvieną DEVS modelį susieja su virtualiu procesoriumi. Todėl, imitavimas yra vykdomas perduodant žinutes tarp abstrakčių imitatorių, o ne tarp pačių DEVS modelių. DEVS imitavimo algoritme egzistuoja trijų tipų procesoriai (4 pav.): imitatorius, koordinatorius ir pagrindinis koordinatorius (angl. *root coordinator*). Kiekvienas atominis modelis turi imitatorių, kuris saugo modelio artimiausio įvykio laiką *tnext*. Jungtinis modelis turi koordinatorių, kuris surenka visą informaciją iš jam pavaldžių imitatorių ir išrenka minimalų laiką *tmin*, kurį persiunčią pagrindiniam koordinatoriui. Pagrindinis koordinatorius yra atsakingas už minimumoišrinkimą iš visų atsiustų laikų. Minimalus laikas *Min* yra persiunčiamas visiems imitatoriams pranešimu (*\**). Imitatoriai, kurių artimiausias vidinis įvykis yra lygus *Min* inicijuoja vidinį įvykį, kurio metu yra generuojamas išvesties signalas *y* ir išsiunčiamas signalu (*done*). Šis signalas persiunčiamas tėviniam koordinatoriui (išorėje gaubiančio jungtinio modelio koordinatoriui), kuris pagal *Z* (išėjimo pavertimo į įėjimą) funkciją perduoda išorinį signalą atitinkamam modeliui pranešimu (*x*).



**4 pav.** DEVS abstraktus imitatorius

Tačiau, kai jau buvo minėta, e. verslo sistemoms modeliuoti reikia kitokių modelių. Aprašytas DEVS modelis netinka, nes turi fiksuotą struktūrą. Tai reiškia, kad šio modelio pokyčius sukeliantys įvykiai įtakoja tik modelio būsienas. Todėl buvo pasiūlyta DEVS modelio modifikacija – dinaminis DEVS modelis (dynDEVS) – kuri leidžia modeliuoti sistemas su kintančia struktūra.

## Dinaminis DEVS modelis

Žinoma, intuatyviai yra aišku, kad dinaminę sistemą galima aprašyti ir taikant formalųjį DEVS modelį. Tačiau tokiu atvėju reikėtų numatyti visus įmanomus sistemų pokyčius, padidinti modelio būsienų aibę bei įvesti painius ryšius tarp modelio sudedamųjų. Taigi teoriškai formalūs DEVS modeliai gali būti naudojami sistemoms su kintančia struktūra aprašyti. Dėja, praktiškas šių modelių taikymas sistemoms su kintančia struktūra modeliuoti yra neįmanomas arba neracionalus. Ypatingos problemos kyla norint modeliuoti sistemą, kurioje atsiranda arba yra naikinami kai kurie ryšiai. Tokių problemų neturi dinaminis DEVS modelis, kuris gali prisitaikyti prie sistemos evoliucijos bei yra save organizuojantis modelis.

dynDEVS modelis gali vykdyti vidinius struktūros pokyčius, tačiau lieka statinis išorinės aplinkos atžvilgiu. Modelis turi tokį apibrėžimą:

čia X ir Y – įvesties ir išvesties įvykių struktūrizuotos aibės, – modelis pradiniu laiko momentu, Minit – modelių aibė, kurios struktūra yra tokia:

čia S – būsienų aibė, – būsiena pradiniu laiko momentu, δint : Q×X→S – vidinio perėjimo funkcija (čia aibė ), δext : S→S – išorinio perėjimo funkcija, ρα : S→M(minit) – modelio struktūrinio perėjimo funkcija, λ : S→Y – išvesties funkcija, ta : – laiko funkcija. Taip pat dynDEVS modelis tenkina savybę:

.

Būtent modelio struktūrinio perėjimo funkcija ρα realizuoja atominių modelių struktūrinius pokyčius, t.y sukuria naujas arba naikina nereikalingas komponentes. Kiekvienas toks struktūrinis pokytis gali iššaukti naujo submodelio atsiradimą. Tam, kad būtų išlaikomas nuoseklumas tarp modelių funkcija išsaugo dviems sekantiems modeliams būdingas parametrų reikšmes m ir n, bei pagal nutilėjimą priskiria pradines reikšmes naujai susidarusiems modeliams. dynDEVS modelio perėjimus galima pavaizduoti tokia schema:



**5 pav. Struktūriniai ir nestruktūriniai perėjimai dynDEVS modelyje**

Bendruoju atvėju visi iš naujai pridėti modeliai yra saugomi aibėje M(minit). Taigi, kaip gaime matyti iš 5 paveiklslo, kiekvienas dynDEVS modelis vystosi kaip DEVS modelių seka. Tačiau, nors vidinė modelio struktūra gali pakisti, sąsaja tarp modelio ir vartotojo išlieka tokia pati. Taip pat išorinis stebėtojas gali matyti visus struktūrinius pokyčius net ir tuo atvėju, kai išvesties duomenys priklauso nuo vidinių sistemos pokyčių. Taigi sistemos sąsaja yra jos vientisumo dalis, kuri, nepriklausomai nuo galimų vidinių struktūros pokyčių, gali būti pasiūlyta vartotojui.

Realizuojant dynDEVS modelį naudojant programavimo kalbas tokias kaip C++ arba Java nauji modeliai yra prijungiami arba pašalinami naudojant metodus *addMethod( ), removeMethod( )*, kurie turi vieną parametrą – patį modelį. Ryšiams tarp modelių nustatyti arba pašalinti naudojami metodai *addCoupling(), removeCoupling()*, kurie turi keturius parametrus – patį modelį, šio modelio išvesties ryšius, modelį, kuriam skirti išvesties duomenys, bei jo įvesties ryšius. Toks parametrų skaičius yra naudojamas dėl to, kad norint pašalinti DEVS modelį reikia nutraukti ir jo ryšius, o norint prijungti naują modelį reikia nustatyti ir ryšius su kitais modeliais. Tarkime, turime tokią sistemą su kintančia strukturą:



**6 pav. Sistemos su kintančia struktūra pavyzdys**

Šią sistemą galime gauti naudojant tokį algoritmą:

*(1) addModel(C);*

*(2) addCoupling(C, COutputPort, B, BInputPort);*

*(3) removeModel(A);*

Nors teoriškai nėra jokių apribojimų kurios dynDEVS modelio komponentės gali sukelti struktūros pakyčius, praktikoje tam yra naudojami atominiai DEVS modeliai, nes jungtinių modeliu elgsena priklauso nuo atominių modelių. Dažniausiai struktūros pokytis pasideda nuo atominio modelio vidinių arba išorinių perėjimų. Tai yra patogu, nes struktūros pokyčius dažniausiai sukelia aplinkybių pokyčiai, kuriuos DEVS modelis supranta kaip tam tikrus įvykius ir apdoroja juos naudojant vidinio bei išorinio perėjimo funkcijas. Taigi tam tikra prasme atominiai DEVS modeliai stebi sistemą. Atsižvelgiant į 6 paveikslą matome, kad sistemos komponentė B gali atlikti priežiūros funkcijas ir inicijuoti sistemos pokyčius. Taip, pavyzdžiui, komponentė B gali stebėti komponentės A įvestį ir, jeigu yra tenkinama tam tikra sąlyga, gali pridėti naują komponentę C ir ryšį tarp C ir B. Toliau B stebi C įvestį ir tam tikros sąlygos tenkinimo atvėju pašalina iš sistemos komponentę A bei ryšį taip A ir B.

Jungtinio modelio analogas dinaminiu atvėju yra tinklinis DEVS modelis (dynNDEVS), kuris hierarhiniu būdu aprašo jį sudarančias kompomentes bei ryšius tarp jų. Šio modelio apibrėžimas yra panašus į klasikinio dynDEVS apibrėžimą, tik šį kartą vietoj minit naudojamas parametras ninit, kuris reiškia konfigūracijos pradžią bei priklauso N(ninit) modelių aibei. Šios aibės struktūra yra panaši į klasikinio jungtinio modelio struktūrą, tačiau vietoj statinių modelių čia yra naudojami dinaminiai, bei atsiranda papildoma tinklų perėjimo funkcija ρN. Naudojant šią funkciją yra apibrėžiami visi tinklo struktūros pokyčiai. Tinklo perėjimo funkcijos paskirtis yra reaguoti į atominių modelių būsienų pokyčius.

Imitavimo sistemose kolizijų problema yra sprendžiama naudojant funkciją *select* : 2D → D, kuri yra įterpiama į modelių aibės Minit (dynNDEVS atveju – į N(ninit)) struktūrą.

## Mobilieji agentai ir MDEVS modelis

E. versle kaip ir e. komercijoje dažnai yra naudojami mobilieji agentai. Mobilusis agentas yra toks programinis agentas, kuris gali pats migruoti kompiuteriniu tinklu iš vienos vietos į kitą ir joje atlikti užduotis (7 pav).



**7 pav. Mobiliojo agento migravimas**

 Pagrindinė idėja, pakeisti darbą tiesiogiai jungiantis prie nutolusio kompiuterio (virtualios darbo vietos, duomenų bazių ir pan.) mobiliuoju agentu. Paprastai vartotojas, norėdamas rasti tam tikrą informaciją, interneto pagalba turi prisijungti prie tam tikrų duomenų bazių. Neradęs reikiamos informacijos, jis jungiasi prie kito duomenų serverio. Atlikdamas paiešką, vartotojas visą laiką išlieka prisijungęs prie duomenų bazės, dėl to yra apkraunamas kompiuterinis tinklas. Be to, interneto ryšys gali nutrūkti ir užduotis negalės būti atlikta iki galo. Tuo tarpu, MA panaudojimas leidžia išvengti šių nepatogumų. Suformulavęs užklausą ir kitus MA reikiamus parametrus, vartotojas paleidžia agentą migruoti kompiuteriniu tinklu tam, kad surasti reikiamos informacijos ar atlikti tam tikrus skaičiavimus. Kol MA migruoja, vartotojui nereikia būti prisijungusiam prie tinklo. Tokiu būdu yra sumažinamas kompiuterinio tinklo apkrovimas bei ryšio kaštai.

E-komercijos erdvėje mobilieji agentai atlieka pirkėjo ir pardavėjo vaidmenis. Agentai pardavėjai atitinka prekybininkus, agentai pirkėjai atitinka vartotojus. Pardavėjų bei pirkėjų agentai veikia ir konkuruoja elektroninėje erdvėje (agentavietėje). Migruodami iš vienos agentavietės į kitą MA renka, analizuoja informaciją ir galiausiai ją pateikia vartotojui. Pagrindinė e-komercijos agentaviečių užduotis priimti atvykstančius pirkėjų ir pardavėjų agentus, juos aptarnauti, palengvinti komunikavimą, bei migravimą.

Mobiliesiems agentams modeliuoti reikia specialių priemonių. Taip yra todėl, kad šios sistemos skirtingai nuo kitų reikalauja mobilumo, ir dėl šios priežasties yra sudėtingesnės. Mobilieji agentai yra diskrečiųjų įvykių sistemos su kintančia struktūra žiūrint iš tos pusės, kad šios sistemos yra tinklai, kuriais juda mobilieji agentai. Mobiliojo agento modelio pavyzdys yra pademostruotas 8 paveiksle.



**8 pav. Mobiliojo agento modelio pavyzdys**

Mobiliųjų agentų modeliavimui buvo pasiūlytas DEVS modelio praplėtimas MDEVS. Šis modelis palaiko ne tik vidinius struktūros pokyčius, bet ir ryšių tarp modelių dinaminius pokyčius. Kaip ir klasikinio DEVS modelio atvėju, MDEVS formalizmą sudaro atominis ir jungtinis modeliai. Atominiai modeliai valdo nematomas sistemos komponentes. Šių modelių struktūra sutampa su klasikinio DEVS formalizmo atominiu modeliu. Jungtiniai modeliai aprašo vidinių modelių tarpusavio sąryšius. Naudojant šiuos sąryšius simuliacijos pranešimai yra persiūnčiami tam tikram modeliui. Skirtingai nuo klasikinio DEVS jungtinio modelio MDEVS jugtinis modelis sugeba atlikti vidinės struktūros pokyčius. Šis modelis yra apibrėžiamas kaip struktūra

Modelio kompomentės turi tokias reikšmes:

* , čia Xin – įvesčių įvykių aibė, Xch – struktūros pokyčių aibė
* Y – išvesčių įvykių aibė
* S – būsienų aibė
* – aktyviųjų modelių aibė (M\* – visų modelių aibė)
* – modelio aktyvacijos funkcija
* – struktūrinių perėjimų funkcija
* – ryšių aibė, čia - išorinių įvesties ryšių sąryšiai, - išorinių išvesties ryšių sąryšiai, - vidinių ryšių sąryšiai, - pokyčių ryšių sąryšiai
* - *select* funkcija

Jungtinis modelis keičia savo struktūrą gavęs įvesties pranešimą, kuris priklauso aibėi Xch. Struktūros būsiena duotoju laiko momentu nusako aktyviųjų modelių bei ryšių aibes. Gautas struktūros pokyčio įvykis pakeičia modelio būsieną naudojant struktūrinio perėjimo funkciją. Ryšių sąryšiai aktyvuojami priklausomai nuo modelio būsienos.

MDEVS modelio imitavimui naudojamas abstaktaus imitavimo koncepciją, kuri yra plačiai aprašyta šio darbo 2.1 skyrelyje. Imitavimo pavyzdys yra pavaizduotas 9 paveksle.



**9 pav. MDEVS modelio imitavimo pavyzdys**

## MDEVS modelio taikymo e. komercijoje pavyzdys

10 paveiksle pavaizduota e. komercijos proceso schema, kuri gali būti realizuota naudojant MDEVS modelį. Šioje schemoje mobilieji agentai bendrauja su paslaugų tiekėjais. 11 paveiksle yra pavaizduotas šios schemos modelis.



**10 pav. E. komercijos proceso schema**



**11 pav. E. komercijos proceso modelis**

Tarkime, vartotojas (Host A) nori pirkti knygą. *User Interface Agent* pagalba vartotojas sukuria mobilųjį agentą. Šis agentas gauna užklausą rasti gerą knygą naudojant internetines knygų parduotuves nuo vartotojo. Mobilusis agentas pirmiausia nueina į paslaugų direktoriją (angl. Service Directory), kurioje yra registruotų knygų parduotuvių sąrašas. Direktorijos agentas savo duomenų bazėje turi visą informaciją apie tiekėjus. Gavęs tiekėjų sarašą mobilusis agentas kreipiasi į kiekvieną tiekėją. Taip bendraujant su kiekvienu tiekėjo agentu vartotojo mobilusis agentas gauna knygų kurias, vartotojas nori pirkti, sąrašą.

# PLA modelis

## Klasikinis PLA modelis

Kitas matematimatinis modelis, skirtas dinaminėms sistemoms modeliuoti yra PLA (angl. Piece-Linear Aggregate – Atkarpomis tiesinis agregatas) modelis. Šis modelis priklauso laikinių automatų modelių klasei. Agregatas aprašomas įvesties bei išvesties aibėmis (atitinkamai X ir Y), ir būsena Z. Įvykiai šiuo atvėju yra suskirstomi į išorinius E‘ bei vidinius E‘‘. Reakcijos į įvykius (t.y būsenos pokyčiai) aprašomos operatoriais H, o išėjimo signalai generuojami operatorių G pagalba. Taigi, trumpai PLA modelį galima apibrėžti tokia struktūra:

Agregato būsiena Z yra aprašoma kaip dviejų komponenčių pora , čia ν(t) – diskrečioji dedamoji, o zν(t) – tolydžioji dedamoji. Tokiu būdu agregato būsena gali pakisti tik tuo atveju, jeigu agregatas gauna išorinį signalą arba kai kinta tolydžioji dedamoji (t.y. įvyksta vidinis įvykis). Agregato funkcionavimas yra stebimas diskrečiais laiko momentais, priklausančiais aibei , kuri įtraukia laiko momentus, kurie nusako, kada ateina išoriniai įvykiai bei įvyksta vidiniai įvykiai . Tokio agregato pavyzdys yra pavaizduotas 12 paveiksle:



**12 pav. PLA agregato pavyzdys**

Operatoriai *H* apibrėžia naują agregato būseną įvykus išoriniam ar vidiniam įvykiui:

Operatoriai *G*, priklausomai nuo būsenos, apibrėžia agregato generuojamus išvesties signalus:

Aibė susijungusių agregatų sudaro agregatinę sistemą. PLA agregatinė sistema yra aprašoma nurodant ją sudarančius agregatus *K* bei šių agregatų įvesties bei išvesties matricas. Šios sistemos pavyzdys yra pavaizduotas 13 paveiksle:



13 pav. Agregatinės sistemos schema

Agregatų įvesčių matrica turi tokį pavidalą:

čia - numeris agregato, priimančio įvesties signalą iš *i-*tojo ryšio kanalo. Komponentė įgyja diskrečiasias reikšmes iš aibės (1, 2, …, K);

- numeris -jo įėjimo, į kurį ateina signalas *i-*tuoju kanalu ().

Agregatų išvesčių matrica :

čia - numeris kanalo, į kurį ateina *i*-tojo agregato *j*-tasisišėjimo signalas, be to , o

Agregatinėje sistemoje vienu kanalu gali būti perduodamas tik vienas išvesties signalas. Sistema yra uždara, t.y į ją nepatenka ir iš jos neišeina jokie išoriniai signalai.

## PLA modelio imitavimo algoritmas

Tarkime, kad agregatinė sistema susideda iš K agregatų, kurie susijungę ryšiais. Aibę įvykių, kurie gali įvykti agregatinėje sistemoje, žymėkime E:

Čia E‘‘ – aibė įvykių, kurie gali įvykti *k-*jame agregate. Kitas aibės E įvykio laiko momentas tm+1 yra išrenkamas naudojantis formule:

čia .

Žemiau pateikti PLA modeliavimo algoritmo žingsniai:

1. Formuojamos agregatų pradinės būsenos;

2. Naudojantis tm+1 išraiška nustatomas kitas artimiausias vidinio įvykio įvykimo laiko momentas ir agregato numeris, kuriame tas įvykis turi įvykti.

3. Naudojantis operatorius H ir G apibrėžiama nauja *k-*tojo agregato būsena, formuojami išvesties signalai, priklausantys aibei  bei nustatoma nauja visų kitų agregatų būsena.

4. Vyksta išvesties signalų aibės patikrinimas. Jeigu šioje aibėje yra nors vienas elementas, tuomet naudojantis matrica H yra nustatomas kanalo numeris, kuriuo turi būti perduodamas išėjimo signalas, o naudojantis matrica R nustatomas ir agregato numeris , kuriam perduodamas signalas yi. Jeigu  aibė yra netuščia, pereinama prie 5 žingsnio, priešingu atveju grįžtama prie 2 žingsnio.

5. Apibrėžiama nauja agregato būsena gavus įėjimo signal xi ir formuojamas naujas išvesties signalas, kuris įtraukiamas į aibę . Apdorotas signalas yi pašalinamas iš aibės  ir grįžtama prie 4 žingsnio.

## Dinaminis PLA modelis

Klasikinis PLA modelis, panašiai kaip ir klasikinis DEVS modelis, yra statinis, todėl norint modeliuoti dinamines sistemas su kintančia struktūra reikia modifikuoti šį modelį. Tokiu atvėju galime kalbėti apie dinaminį PLA modelį (dynPLA).

dynPLA modelis praplėčia klasikinį PLA modelį naujomis operacijos, skirtomis modifikuoti agregato bei agregatinės sistemos aprašymus. Kiekvienas agregatas turi galimybę keisti savo bei visos sistemos struktūrą, manipuliuojant naujai įvestomis operacijomis, tokiomis kaip naujo agregato įvedimas arba agregato įėjimo signalo pašalinimas.

*H* ir *G* operatoriai dynPLA modelyje yra aprašomi kaip aibės, kurios gali pasipildyti naujais operatoriais arba pašalinti esamus, tačiau visi galimi agregato operatoriai turi būti aprašyti iš anksto. Papildomos išraiškų operacijos, kurios nusako struktūros pokyčius yra apibrėžiamos agregatų operatoriuose. dynPLA meta-modelis yra pavaizduotas 14 paveiksle:



**14 pav. dynPLA meta-modelis**

PLA modelyje agregato perėjimo operatoriai aprašo tolydinės bei diskretinės komponenčių pokyčius priklausomai nuo vidinių ir išorinių įvykių. dynPLA agregate tie patys įvykiai gali iššaukti ir struktūros pokyčius, aprašomus perėjimo operatoriuose. Todėl agregato būsena yra praplečiama įtraukiant visus likusius agregato atributus su vidinių agregatų ir ryšių aibėmis*.* Taigi dynPLA gali būti apibrėžta kaip struktūra:

čia - vidinių agregatų aibė laiko momentu t,

 - ryšių aibė laiko momentu t,

 - agregatas su struktūra **,

 čia - įvesties signalų aibė laiko momentu *t*,

  - išvesties signalų aibė laiko momentu *t*,

  - įvykių aibė laiko momentu *t*,

 - išorinių įvykių aibė laiko momentu *t*,

 - vidinių įvykių aibė laiko momentu *t*;

 - būseną keičiančių vidinių įvykių aibė laiko momentu *t*,

 - struktūrinių vidinių įvykių aibė laiko momentu *t*,

 - perėjimo operatorių aibė laiko momentu *t*,

 - išėjimo operatorių aibė laiko momentu *t*,

- vidinių agregatų aibė laiko momentu *t*,

- ryšių aibė laiko momentu *t*,

 -būsena laiko momentu *t*,

 - tolydinė komponentė laiko momentu *t*,

 - diskretinė komponentė laiko momentu *t*.

dynPLA modelyje tarp agregatiniams susijungimams aprašyti yra naudojama *M* matrica, kuri apibrėžia agregatų ryšius laiko momentu *t*. Kiekvienas ryšys iš sujungimų matricos yra aprašomas pagal žemiau pateiktą struktūrą.

 1 lentelė. Agregatų sujungimo matricos struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Kanalo numeris** | **Iš agregato** | **Išvestis** | **Į agregatą** | **Įvestis** |
| Skaičius | Agregato vardas | Išvesties signalo vardas | Agregato vardas | Įvesties signalo vardas |

dynPLA modelyje agregatinės sistemos bei ją sudarančių agregatų struktūros pokyčius gali iššaukti tik įvykiai. Kiekvienas agregato išorinis įvykis yra tam tikro vidinio įvykio pasekmė. Tai reiškia, kad egzistuoja vidiniai įvykiai, kurie inicijuoja struktūros pakitimus. Kiekvieno tokio įvykio metu agregatas turi galimybę atlikti struktūrines modifikacijas. Agregatas norėdamas atlikti struktūrinius pokyčius kituose agregatuose turi turėti visą informaciją apie jį supančią aplinką, kurios kiekvienos pokytis reiškia šios informacijos atnaujinimą. Toks būdas, kai agregatams leidžiama keisti ne vien savo, bet ir kitų agregatų struktūras, reikalauja daug papildomų veiksmų, kadangi visi agregatai turi būti nuolat informuojami apie kiekvieną išorės struktūrinį pakitimą. Visus dynPLA modelio vykdomus struktūros pokyčius galima suskirstyti į dvi grupes:

*Vidiniai agregato struktūros pokyčiai* – tai agregato struktūrą apibrėžiančių atributų aibių pokyčiai, kurie neįtakoja agregato išorinės sąsajos pokyčių, tokie kaip diskretaus kintamojo pašalinimas/pridėjimas ir t.t. Tam tikri agregatų atributai tarpusavyje yra susiję, todėl pokyčiai vienoje atributų aibėje iššaukia pokyčius kitose susijusiose atributų aibėse. Pavyzdžiui, norint pridėti naują vidinį įvykį, neužtenka vidinių įvykių aibę papildyti nauju elementu . Agregato vidinių įvykių skaičius yra lygus tolydinės dedamosios elementų skaičiui, be to, kiekvienam vidiniam ir išoriniam įvykiui yra priskiriami operatoriai. Todėl, atsiradus naujam vidiniam įvykiui, agregato tolydinė dedamoji turi pasipildyti nauja komponente , perėjimo operatorių aibė nauju operatoriumi skirtu šiam įvykiui apdoroti.

*Agregatinės sistemos pokyčiai* - tai struktūros pokyčiai, kurių metu laike pakinta agregatinę sistemą sudarančių agregatų bei ryšių aibės.

Visi agregatinės sistemos struktūros pokyčiai yra pastebimi išoriniam vartotojui. Tam tikros pokyčių operacijos keičia tik agregatinės sistemos struktūrą, kitos papildomai iššaukia struktūrines modifikacijas ir vidinių agregatų aprašymuose. Čia prie standartinių pokyčių, tokių kaip agregato prijungimas ir pašalinimas bei ryšio prijungimas ir pašalinimas, pridedamas agregato transportavimas iš vienos agregatinės sistemos į kitą, kuris iš esmės yra dviejų pokyčių (agregato pašalinimas iš vienos sistemos bei prijungimas prie kitos sistemos) kombinacija. Agregatas gali būti transportuojamas į kitą agregatinę sistemą vienu hierarchiniu lygiu žemiau arba aukščiau, o tap pat iš vienos agregatinės sistemos į kitą, kurias išorėje gaubia ta pati agregatinė sistema. Kiekvienas iš šių pokyčių iššaukia atitinkamus struktūrinius sistemos pokyčius. Pavyzdžiui šalinant agregatą iš sistemos taip pat turi pašalinami ir ryšiai taip šio ir kitų agregatų. Šie veiksmai yra būtini norint užtikrinti korektišką dynPLA modelio funkcionavimą. Priklausomai nuo modeliuojamo uždavinio specifikos, šie pokyčiai gali iššaukti ir papildomus struktūros pokyčius (pvz., atsirasti naujas diskretus kintamasis), kurių aprašymas yra paliekamas specifikuotojo nuožiūrai.

## dynPLA modelio taikymo e. komercijoje pavyzdys

15 paveiksle yra pavaizduota e. komercijos mobilios agentinės sistemos (e-KMAS) su maršrutiniu agento migravimu dynPLA modelis. Jame Vartotojų Sąsajos*,* Vartotojų Sąsajos Agentai,Paslaugų Katalogas*,* Tiekėjai yra vaizduojami atskiromis agregatinėmis sistemomis atitinkamai *VSi*, *VSAi*, *D*, *Ti*, kuriuos išorėje gaubia agregatinė sistema *Tinklas*, .



**15 pav.** e-KMAS su maršrutiniu agento migravimu agregatinė schema

Kiekviena *VSi* agregatinė sistema turi po vieną vartotojo sąsajos agento agregatą *VSAi* *,* Sistema *VSi* vidinio įvykio metu generuoja užduotis, kurios yra perduodamos *VSAi* agregatui. Šis agregatas, gavęs užduotį išsiunčia signalą *yadd* agregatui *VSi* irtaipinicijuoja mobilaus agento agregato *MAi* sukūrimą. *VSAi* agregatas sukurtam *MAi*  paveda užduoties atlikimą *ydo.* . *MAi* agregatas, gavęs užduotį, pradeda savo migravimą. Kai *MAi* turi tuščius tiekėjų bei prekių sąrašus išoriniu signalu inicijuoja migravimą iš *VSi* į paslaugų katalogą aprašančią agregatinę sistemą *D*. Visi *MAi* agregato išoriniai signalai migruoti yra perduodami į *Tinklas* agregatinę sistemą. Sistema *Tinklas* perkelia *MAi*  agregatą į nurodytą agregatinę sistemą bei prideda jam ryšius. Persikėlęs į *D*, agregatas *MAi* išoriniu signalu *ygetL* formuoja užklausą tiekėjų sąrašui gauti. Gavęs tiekėjų sąrašą *MAi* migruoja toliau, t.y į pirmą sąraše *list* esantį tiekėją. Pirmas sąraše nurodytas tiekėjas - agregatinė sistema *Ti*išimama iš eilės ir priskiriama kintamajam *VIET*. Kiekvieno tiekėjo agregate *MAi* agregatas generuoja išorinį signalą prekių sąrašui gauti. *MAi* agregato prekių sąrašas *goods(t)* yra papildomas gauta informacija. Kai visi tiekėjai iš sąrašo yra aplankyti, *MAi* grįžta į savo pradinę vietą − *VSi* agregatinę sistemą, kur perduoda visą surinktą informaciją. Kai *VSAi* gauna visą surinktą informaciją iš *MAi*, jis persiunčia duomenis *VSi* agregatinei sistemai bei išvesties signalu *ydel* inicijuoja *MAi* pašalinimą.

# Apibendrinimai

Žinoma, ekonomikos bei verslo srytys negali egzistuoti be matematinių modelių. Net e. verslo procesų atvėju jiems modeliuoti yra naudojami tokie statinių matematinių modelių DEVS ir PLA praplėtimai, kaip MDEVS ir dynPLA. Šie modeliai yra dinaminiai todėl gali pakankamai tiksliai aprašyti sudėtingas e. verslo sistemas su kintančia struktūra. Ypač sėkmingai šie modeliai gali būti pritaikyti e. prekybos srityje, naudojant specialius programinius agentus, kurie gali migruoti iš vienos sistemos į kitą bei atlikti jose įvairias užduotis – mobiliuosius agentus.

# Literatūra

1. Agnė Paulauskaitė – Tarasevičienė. Sistemų su kintančiomis struktūromis modeliavimas agregatiniu metodu
2. <http://www.sce.carleton.ca/faculty/wainer/papers/spe482.pdf>
3. L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds. A simulation algorithm for dynamic structure DEVS modeling
4. <http://sim.kaist.ac.kr/paper/JF/JF-34.pdf>
5. Alexander Osterwalder. An e-Business Model Ontology for Modeling e-Business
6. Xiaolin Hu, Bernard P. Zeigler, and Saurabh Mittal. Dynamic Reconfiguration in DEVS Component-based Modeling and Simulation
7. A. M. Uhrmacher. Dynamic structures in modeling and simulation: a reflective approach