KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKO KATEDRA

Kostas Mackevičius

Integruotų verslo sistemų formalizavimas, panaudojant komponentinę abstrakciją

Magistro darbas

Darbo vadovas:
prof., habil. dr. H.Pranevičius

Kaunas

2008
KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKO KATEDRA

Kostas Mackevičius

Integruotų verslo sistemų formalizavimas, panaudojant komponentinę abstrakciją

Magistro darbas

Recenzentas:
prof., habil. dr. V.Štuikys
2008.05.22

Darbo vadovas:
prof., habil. dr. H. Pranevičius
2008.05.19

Atliko:
IFM-2/1 gr. stud.
Kostas Mackevičius
2008.05.16

Kaunas

2008

TURINYS

ĮVADAS

Temos aktualumas. Darbe sprendžiama integruotų verslo sistemų (IVS) formalizavimo problema panaudojant komponentinę abstrakciją. Verslo sistemą sudaro šie lygmenys :

- ✓ Verslo lygmuo;
- ✓ Informacijos apdorojimo lygmuo;
- ✓ Programinės įrangos lygmuo.

Informacinės technologijos (IT) skverbiasi į visas žmogaus veiklos sritis, veikia žmonių gyvenimo būdą, kultūrą ir kt. XXI amžius yra informacijos amžius, kuriame informacija įgauna dominuojantį vaidmenį. Jau dabar egzistuoja ir ateityje stiprės naujų informacinių technologijų plėtros ir pasaulinių tendencijų ryšys.

Įmonės arba organizacijos integruota verslo sistema kuria ir palaiko visą įmonę ar organizaciją apimantį vieningos informacijos srautą, užtikrindama, kad įmonės darbuotojai bet kuriuo metu gaus visą jiems reikalingą kokybišką ir patikimą informaciją, taip pat tai užtikrins stabilų darbą ir veiklą. Gamybos tendencijos sudaro naujas sąlygas organizacijų veiklai ir jų pertvarkai. Norėdami sukurti arba formalizuoti puikią integruotą verslo sistemą, turime gerai žinoti, kokie reikalavimai bus keliami naujai sistemai, stengtis kuo labiau sumažinti projektavimo laiką, panaudoti jau turimus komponentus, sumažinti projekto kaštus ir maksimaliai padidinti IVS našumą ir žinomą - kokybę. Žinoma IVS turi būti lengvai tobulinama, kadangi technologijos sparčiai tobulėja, tai ir esamos sistemos turi būti lygiagrečiai tobulinamos.

Kuriant arba formalizuojant sudėtingas integruotas verslo sistemas galima aibė skirtingų realizacijos pasirinkimų. Kadangi šie pasirinkimai daromi pradinėse kūrimo fazėse, todėl jos vaidina lemiamą vaidmenį sistemos finaliniame darbo našume. Siekiant išvystyti, padidinti darbo našumą, priklausantį nuo pasirinktų kūrimo alternatyvų, prieš realizuojant sistemą, galima panaudoti priemones aprašančias sistemos formalią specifikaciją ir leidžiančias išanalizuoti sistemos našumo parametrus dar prieš pradedant sistemos realizaciją. Tokiomis priemonėmis yra „pereinamos“ pirmosios sistemos kūrimo proceso fazės ir suformuojamas abstraktus kuriamos sistemos modelis, kuris ir yra skirtas tos sistemos savybėms analizuoti. Tokie įrankiai turi dirbti su kokia nors modeliavimo kalba ir turi turėti darbo našumo analizės įrankius.

Viena iš tokių modeliavimo kalbų yra PLA (tiesinių agregatų modeliavimo metodika). Ši kalba turi aibę notacijų, leidžiančių specifikuoti sistemos funkcionalumą. Be to,

PLA dažnai naudojama sistemos koncepcijai vystyti ar imituoti ir dokumentacijai apipavidalinti.

Integruotos verslo sistemos formalizavimo tyrimo dalyje bus ištirtos formalizavimo ir analizės metodo galimybės, kurios remiasi trejomis sistemos formalizavimo (modeliavimo) kalbomis. Tai būtų dvi grafinio modeliavimo kalbos: UML (Unifikuota modeliavimo kalba) ir BPMN (verslo proceso valdymo žymėjimas), taip pat, atkarpomis tiesinių agregatų (PLA) formalizavimo metodas. Tyrimo eigoje neatliekamas detalus sistemos komponentų formalizavimas PLA metodu, bet pateikiamas vieno komponento formalizavimo pavyzdys (1 priedas). Šis metodas priklauso automatų modelių klasei, tačiau išskirtinis tuo, kad sistemos būsenai aprašyti naudojamos ir diskrečios ir tolydžios laiko koordinatės. Šis formalizavimo metodas yra išraiškingas, su griežtomis formaliomis semantikomis, kurios leidžia atlikti integruotų verslo sistemų analizę. Čia nagrinėjamos sistemos objektai arba komponentai aprašomi kaip atskiri agregatai. Čia įtraukta ir laiko sąvoka, todėl galima atlikti sistemų imitavimą tam, kad būtų išgautas sistemos darbo našumo parametras realiai dirbant (imituojant sistemos darbą) sistemai.

Komponentai – tai jau turimų (sukurtų) dalykų panaudojimas, kuriant ar specifikuojant integruotas verslo sistemas. Pasinaudojant komponentais, bandoma sukurti naujus metodus ir principus tam, kad geriau greičiau ir patikimiau pagaminti naują integruotą verslo sistemą ar patobulinti esamą.

Darbo tyrimo objektas yra integruotų verslo sistemų formalizavimo procesas, grindžiamas komponentine abstrakcija.

Darbo tikslas yra ištirti komponentinės paradigmos taikymo galimybes integruotoms verslo sistemoms kurti, suformuluoti svarbiausias problemas, kurias reikia spręsti, norint projektuoti ir realizuoti šias sistemas.

Darbe sprendžiami tokie uždaviniai:

1. Išanalizuoti egzistuojančią komponentinę formalizavimo paradigmą integruotoms verslo sistemoms;
2. Ištirti organizacijos sprendimo priėmimo sistemos pavyzdį remiantis komponentine paradigma.
3. Formaliai aprašyti sistemos lygmenyse naudojamus komponentus ir jų savybes.
4. Apibendrinti atlikto tyrimo rezultatus ir pateikti išvadas.

Mokslinės problemos esmė. Integruotos verslo sistemos į darnią visumą sujungia verslo informacijos apdorojimo ir techninės bei programinės įrangos komponentus. Tokių sistemų kūrimas, sudėtinės dalys, modulinė architektūra turi savo specifinių ypatumų.

Problemos esmė – ištirti komponentinės paradigmos taikymo galimybės tinkamumą, integruotoms verslo sistemoms kurti.

Dokumento paskirtis. Šis magistro baigiamasis darbas pateikia problemas, kurios egzistuoja formalizuojant integruotas verslo sistemas remiantis komponentine paradigma. Nagrinėjama komponentinė sistemų kūrimo paradigma, siekiant perkelti komponentinių programų sistemų kūrimo idėjas į integruotų verslo sistemų lygmenį. Pateikiama integruotos verslo sistemos samprata. Nagrinėjamos komponento ir jo pagrindinių dalių sampratos, įvairių abstrakcijos lygmenų komponentų ypatumai, jų jungimo būdai. Suformuluoti reikalavimai informacinės sistemos komponentui. Pateikiami egzistuojantys metodai toms problemoms spręsti.

Darbo struktūra. Darbą sudaro pagrindinės dvi dalys: teoriniai integruotos verslo sistemos aspektai (pirma dalis) ir integruotos verslo sistemos formalizavimo tyrimas (antra dalis).

Pirmoje dalyje aptariama komponentinės abstrakcijos samprata. Aptariamos jos svarba ne tik verslo sistemų kūrime, bet ir programinės įrangos projektavime. Pasirenkamas tinkamiausias IVS formalizavimo metodas.

Antrojoje dalyje tiriamos PLA, UML ir BPMN kalbos galimybės modeliuoti pasirinktą „Organizacijos sprendimo priėmimo sistemą“. Pritaikoma komponentinė kūrimo paradigma, kuri buvo analizuojama pirmoje dalyje. Darbo pabaigoje pateikiami tyrimo rezultatai.

Darbo apimtis. Magistrinį darbą sudaro įvadas, dvi dalys („teoriniai integruotos verslo sistemos aspektai“ ir „integruotos verslo sistemos formalizavimo tyrimas“), išvados, literatūros sąrašas, santrumpų ir terminų sąrašas, priedai. Darbo medžiaga išdėstyta 53 puslapiuose, kuriuose yra 26 paveikslai ir 4 lentelės. Panaudotos literatūros sąrašą sudaro 20 šaltinių.

1. TEORINIAI INTEGRUOTOS VERSLO SISTEMOS ASPEKTAI

1.1. Įžanga

Integruotą verslo sistemą (IVS) į darnią visumą sujungia verslo, informacijos apdorojimo ir programinės bei techninės įrangos dalis. Kitaip tariant, tokia informacinė sistema yra viena iš verslo sistemos posistemių, kurį sudaro verslui reikalingos informacijos apdorojimo procesai. Paprastai integruotos verslo sistemos yra kompiuterizuotos, vadinasi, jos turi būti kuriamos atsižvelgiant į trijų lygmenų - verslo, informacijos apdorojimo ir informacinės technologijos - tarpusavio ryšius. Tačiau IVS kūrimas turi savo specifinių ypatumų.

Pirma, integruotos verslo sistemos yra labai sudėtingi inžineriniai artefaktai. Vadinasi, šitokios sistemos turi būti projektuojamos, o tam turi būti naudojama inžinerinė disciplina, nusakanti kuo remiantis, kokia tvarka ir kokius sprendimus reikia priimti. Sistemos kūrimo procesas yra dekomponuojamas į stadijas, be to, turi būti taikomi integruotų sistemų kūrimo principai ir paradigmos.

Antra, kuriant IVS, jas reikia nagrinėti ne tik kaip sistemas, kurių elgsena gali būti išsamiai specifikuojama analizės metu, bet ir kaip į susiformuojančios elgsenos sistemas. Kitaip sakant, sistemos elgsena negali būti nusakyta iš anksto, ji atsiranda kaip dinaminės sąveikos tarp sistemos elementų rezultatas.

Trečia, IVS užtikrina visos organizacijos veiklai palaikyti reikiamą vieningos informacijos srautą. Tai įmanoma tik tada, jei verslo sistemos sudėtinės dalys yra kuriamos kaip tos sistemos komponentai, kas reiškia, kad jie turi standartizuotus kontraktiškai specifikuotus sąveikos interfeisus ir yra realizuojami vadovaujantis vieninga standartų sistema. Pastebėsiu, kad komponentinė inžinerija turi būti naudojama kuriant integruotas organizacijų informacines sistemas ir dėl daugelio kitų priežasčių. Nuolatos kintant aplinkos reikalavimams reikia turėti tokią IVS kūrimo infrastruktūrą ir metodus, kurie leistų nesunkiai modifikuoti jau egzistuojančias sistemas, pakartotinai panaudoti jau sukurtus artefaktus, užtikrintų sistemos vientisumą (naudojant komponento abstrakciją, galima vienodai traktuoti skirtingų rūšių IVS sudėtines dalis).

Ketvirta, kuriant integruotas verslo sistemas, kaip ir kitas sistemas, jas reikia nagrinėti skirtingais abstrakcijos lygmenimis. Taigi komponento samprata priklauso nuo pasirinkto abstrakcijos lygmens [3].

Ilgesnę istoriją turinčiose inžinerijos šakose, tokiose kaip statyba, transporto priemonių ir netgi elektroninės aparatūros kūrimas, komponentinės paradigmos prasmė

abejonių nekelia. Komponentinės technologijos, pavyzdžiui, CORBA, COM+/.NET, Enterprise Java Beans (EJB) ir kt., yra vis plačiau naudojamos projektuojant ir įgyvendinant sudėtingas programų sistemas. Pastebėsiu, kad komponentai faktiškai yra dabartinis programų sistemų (ypač išskirstytų) realizavimo standartas. Tačiau kuriant aukštesniojo abstrakcijos lygmens (informacinių, verslo) sistemas, komponentinis požiūris nėra plačiai naudojamas. Nors komponentinė inžinerija yra pakankamai brandi ir naudojama kaip teorinė disciplina sprendžiant praktines programų sistemų kūrimo problemas, šiuo metu nežinoma, kaip taikyti komponentinės paradigmos idėjas verslo sistemoms projektuoti. Nėra sukurtos darnios, nuo technologijos nepriklausančios komponentinės sistemos konceptų ir apibrėžčių visumos, nėra tinkamų notacijų ir modeliavimo kalbų. Kitaip tariant, komponentinių verslo sistemų teorija dar tik pradedama kurti.

1.2. Komponentinė sistemų kūrimo paradigma

Sistemų kūrimas iš komponentų, leidžia suprojektuoti nesunkiai modifikuojamas sistemas, pakartotinai naudojant jau turimus artefaktus. Kitaip tariant, tikslinga komponuoti sistemas iš maksimaliai nepriklausomų tarpusavyje sudėtinių dalių. Toks sistemų kūrimo būdas yra plačiai naudojamas įvairiose inžinerijos šakose.

Sukurti komponentinių kompiuterizuotų sistemų kūrimo metodai dažniausiai skirti programų sistemų lygmens artefaktams kurti arba naudojami programų sistemų kūrimo proceso konstravimo stadijoje. Pagrindinis komponentinės paradigmos reikalavimas - galimybė pakeisti komponentą nauja to paties komponento versija, arba kitu, turinčiu tą patį funkcionalumą, bet skirtingą realizaciją, komponentu.

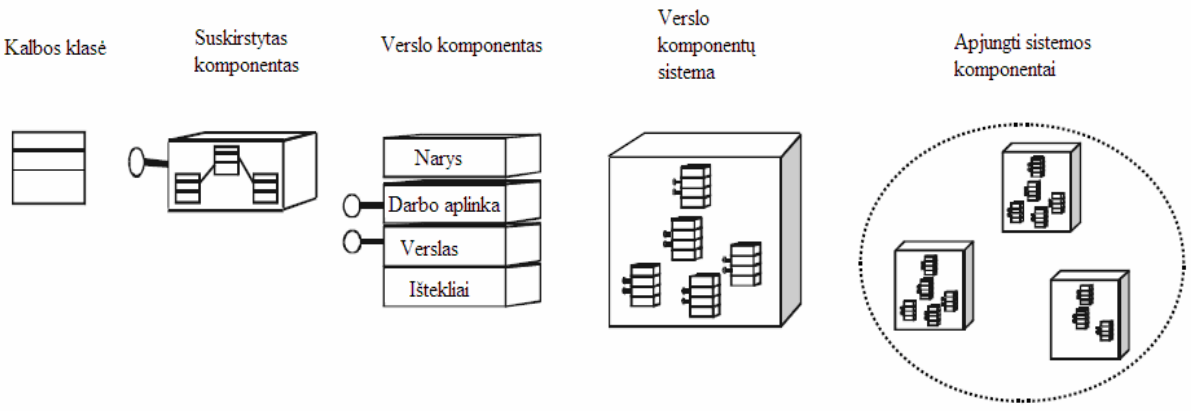
Kalbant apie programinį komponentą, dažniausiai sakoma (pvz., [1]), kad tai yra fizinė sistemos dalis, kuri atitinka ir realizuoja tam tikrą kontraktais specifikuotą interfeisų aibę. Kitais žodžiais tariant, komponentu laikomas tam tikrą funkcionalumą turintis binarinio kodo paketas. Tai buvo vyraujanti komponento samprata nuo komponentinės paradigmos atsiradimo iki UML 1.3 versijos sukūrimo. Be to, komponentinių programų sistemų kūrimo pradininkai labiausiai domėjosi, kaip komponentas atitinka naudojamos kūrimo metodikos ir infrastruktūros standartus.

UML 1.3 išplėtė komponento sampratą iki programų sistemos elemento ir posistemio. Realiai, komponentai naudojami tiek loginiu, tiek fiziniu lygmeniu, ir yra prasminės struktūrinės ir/arba funkcinės didesnės sistemos dalys. Nauja UML versija buvo papildyta galimybėmis apibrėžti komponentus ne tik kaip realizacinio, bet ir kaip projektinio

lygmens artefaktus, t. y. programų sistemos architektūrai kurti, UML 2.0 leidžia naudoti komponentinės paradigmos idėjas bei technikas ir realizacijai, ir projektavimui.

Kiti autoriai komponentą nagrinėja kaip programų sistemos elementą, apimantį sudėtingas saugomų duomenų struktūras. Priklausomybės tarp komponentų nusakomos kontraktais. Autoriai išskiria šias kontraktų rūšis: naudojimas ir realizacija. Kontraktai specifikuojami OCL sakiniais ir UML ansamblių (angl., assembly) diagramomis. Autoriai siūlo tokį UML komponento nusakymo būdą panaudoti verslo sistemos komponentui apibrėžti, specifikuoti ir realizuoti.

Darbe [8] pakylama į verslo sistemos lygmenį. Tačiau verslo sistemos kontekste nagrinėjami programiniai komponentai. Autorių nuomone, verslo sistemoje programiniu komponentu turi būti realizuojama kuri nors verslo sistemos esybė (pvz., daiktas, verslo procesas). Kiti autoriai skiria įvairaus granuliariškumo (1 pav.) komponentus - programinį komponentą, verslo komponentą, verslo komponentų sistemą. Verslo komponentų sistemos modelyje aprašomi šios sistemos funkcionalumą nusakantys verslo komponentai. Vadinasi, vis dėlto liekama programų sistemų lygmenyje. Informacinių sistemų programinė įranga gali būti sukomponuota iš rinkoje įsigytų pagamintų komponentų (angl. *commercial-off-the-shelf (COTS)).* COTS produktai yra projektuojami taip, kad juos būtų galima nesudėtingai integruoti į organizacijoje egzistuojančias programų sistemas.



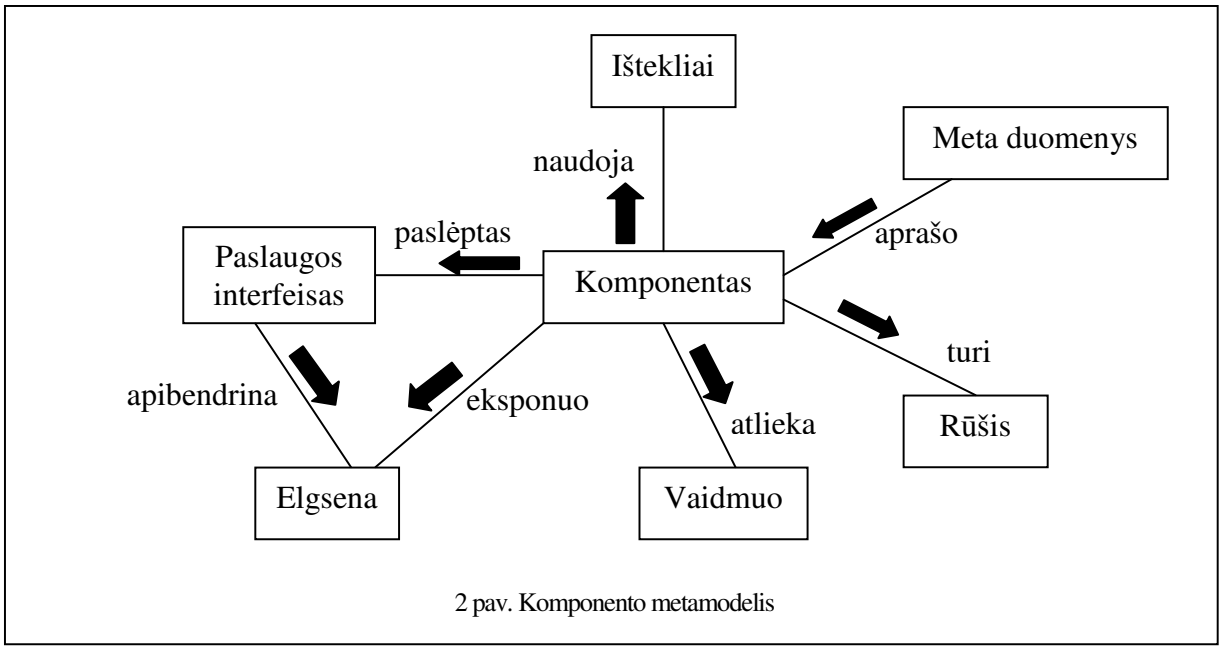
1 pav. Komponento granuliariškumo lygiai

Apibrėždami komponentus, darbų autoriai paprastai akcentuoja svarbiausius jų nagrinėjamoje srityje komponentų ypatumus, tokius kaip autonomiškumas, granuliariškumas, naudojimo kontekstas. Todėl terminas „komponentas“ įvairiuose

kontekstuose reiškia daug skirtingų dalykų. Atsižvelgiant į tai, reikia ne apibrėžti šį terminą visiems atvejams ir visiems laikams, o atsakyti į du esminius klausimus:

- ✓ nustatyti, kas skiria komponentą nuo bet kurios kitos sudėtinės dalies,
- ✓ sudaryti komponento metamodelio kuris nusakytų komponento struktūrą, įvairias leistinas savybes, galimą klasifikavimą ir pan.

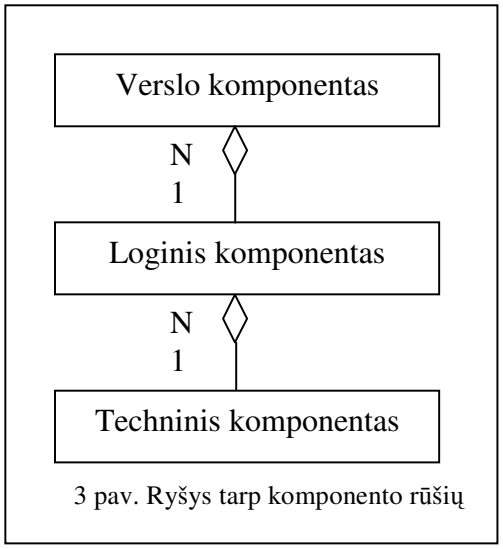
Panašus požiūris dėstomas ir M. Voelter darbe [19], kuriame autorius parodo termino nevienareikšmiškumą, naudoja komponentų aibės sutvarkymą ir aprašo leistinas jų savybes. Komponentui nusakyti vartojamos paslaugų interfeiso, meta duomenų, rūšies, išteklių ir vaidmens sąvokos (2 pav.). Reikia, kad būtų specifikuotos komponento teikiamos paslaugos, o tai paprastai specifikuojama operacijų ir jų parametrų terminais. Autorius nurodo komponentui būdingus vaidmenis: esybė, procesas, paslauga. Be to, specifikuojant komponentą, turi būti nusakyti jam reikalingi ištekliai.



2 pav. Komponento metamodelis

Darbo autorius taip pat siūlo skirti loginius ir techninius komponentus (3 pav.). Loginis komponentas suprantamas kaip paketas, turintis tam tikrą funkcionalumą ir sąlygojantis sistemos sudėtingumą. Pažymima, kad loginiai komponentai dar skirstomi į dalykinės srities, duomenų ir naudotojo komponentus - dalykinės srities komponentuose nusakoma verslo logika; duomenų komponentai užtikrina prieigą prie duomenų; naudotojo komponentai įgyvendina naudotojo interfeiso funkcionalumą. Techniniai komponentai yra programiniai elementai, iš kurių konstruojamos dalykinės programos, veikiančios tik tam tikroje aplinkoje (angl. *container).* Taikomas turinių atskyrimo principas - dalykinės

programos aspektai realizuojami skirtinguose techniniuose artefaktuose. Techniniai komponentai diegiami kliento programose arba, jei yra kelių lygmenų sistema, serveryje.



3 pav. Ryšys tarp komponento rūšių

Tarp techninių ir loginių komponentų yra agregavimo ryšys. Kitaip tariant, techniniai komponentai yra sudėtinės loginio komponento dalys. Darbe [14] vartojama verslo komponento sąvoka. Jis apibrėžiamas kaip loginių komponentų (duomenų, dalykinės srities ir naudotojo) agregatas. Taigi, sudėtingesni komponentai konstruojami iš paprastesnių. Be to, reikia išskirti skirtingus sistemos lygmenis - techninį, loginį ir verslo, kuriems įgyvendinti turi būti naudojami skirtingų rūšių komponentai.

Pastebėsiu, kad šiuo metu komponentinės paradigmos principai taikomi kuriant techninę įrangą ir programų sistemas, tai yra, naudojant funkcijas, procedūras, paketus, klases, objektus. Informacinių sistemų kontekste komponentinis požiūris dar tik pradedamas naudoti ir komponento samprata plačiai nenagrinėjama, todėl nėra konceptų ir apibrėžčių visumos, kuri leistų nusakyti, kaip ir iš kokių dalių sudaromi verslo sistemos komponentai.

Šiame darbe verslo sistemos komponentą apibrėšiu, kaip komponavimo agregatą, turintį sąveikos interfeisus. Dar viena savybė, skirianti verslo sistemos komponentą nuo bet kurios kitos sudėtinės dalies, yra galimybė jį realizuoti atskirai kaip agregatą, tai yra, nepriklausomai nuo kitų komponentų. Interfeisai nusako verslo sistemos komponento funkcionalumą ir teikiamas paslaugas. Verslo sistemos komponentų pavyzdžiais gali būti: organizacijos darbuotojų kompiuterių tinklas (angl. *intranet),* apsaugos modelis, techninė įranga, duomenų saugykla ir panašiai.

1.3. Verslo komponento samprata

Komponentinių sistemų kūrimo gyvavimo ciklo metu išskiriamos įvairios komponentų rūšis. Aukščiausiame lygmenyje yra įmonės (angl. enterprise) komponentas, galintis egzistuoti kaip savarankiška sistema. Įmonės komponentas teikia verslo procesų automatizavimo paslaugas arba gali būti naudojamas kelių įmonių verslo sistemų integravimui. Realizuojamas konstruojant iš žemesnio lygmens komponentų arba, kaip monolitinė struktūra.

Verslo komponentas teikia paslaugas, turinčias aiškiai nusakytą, išmatuojamą naudą verslo sistemai. Kitaip tariant, verslo paslaugos nusako funkcionalumą, reikalingą konkrečiam verslo tikslui įgyvendinti. Verslo komponentas teikiamas paslaugas susieja su verslo esybėmis, verslo taisyklių aibe. Apibrėždamas programinius komponentus, išskiriu du tipus: komponentus - esybes ir komponentus - procesus.

1.4. Informacinės sistemos komponento samprata

Darbų, nagrinėjančių informacinės sistemos komponentus, yra tik keletas. [14] autoriai aprašo duomenų bazes, kaip svarbiausią informacinių sistemų struktūrinį elementą.

Komponentas gali būti nagrinėjamas duomenų bazės, kaip svarbiausias informacinių sistemų struktūrinis elementas. Todėl informacinės sistemos komponentas apibrėžiamas, kaip trejetas $ISC =< SS, DS, IR >$. Gauname, kad informacinės sistemos komponentas yra koncepcinis objektas, kurį sudarant turi būti nusakyta:

- ✓ statinė sritis SS, aprašanti informacinės sistemos statines savybes, ir modeliuojama UML klasių diagrama;
- ✓ dinaminė sritis DS, t.y., modeliuojamos dinaminės informacinės sistemos savybės;
- ✓ darnos taisyklių sritis IR, t.y., nusakomi ribojimai, užtikrinantys korektišką sistemos veikimą.

Darbe [9] informacinės sistemos komponentas nagrinėjamas koordinuojančiųjų (angl., *Coordination - ready)* informacinių sistemų kūrimo kontekste. Autorius teigia, kad norint veiksmingai valdyti informacinių sistemų kūrimą, svarbu tinkamai moduliarizuoti informacinius išteklius koncepciniu lygmeniu, todėl informacinės sistemos komponentą apibrėžia, kaip pakartotinai panaudojamą, turintį tam tikrą funkcionalumą informacinės sistemos artefaktą. Informacinės sistemos komponentas turi visas savybes, kad galėtų

funkcionuoti, kaip atskira informacinė sistema. Kitaip tariant, informacinės sistemos komponentai yra savarankiški, nepriklausomi artefaktai. Tačiau verslo sistemą gali sudaryti ir keletas informacinės sistemos komponentų, t.y., komponentai bendradarbiauja įgyvendindami organizacijos tikslus. Specifikuojant informacinės sistemos komponentą, analizės metu aprašomos klasės, jų atributai, būsenos, metodai, įvykiai, darnos taisyklės.

1.5. Programinio komponento samprata

Kalbant apie programinį komponentą, dažniausiai sakoma, kad tai yra fizinė sistemos dalis, kuri atitinka ir realizuoja tam tikrą kontraktais specifikuotą interfeisų aibę. Kitais žodžiais tariant, komponentu laikomas tam tikrą funkcionalumą turintis, binarinio kodo paketas. Nuo komponentinės paradigmos atsiradimo iki UML 1.3 versijos sukūrimo tai buvo vyraujanti komponento samprata. Be to, komponentinių programų sistemų kūrimo pradininkai labiausiai domėjosi, kaip komponentas atitinka naudojamos kūrimo metodikos ir infrastruktūros standartus.

Nors UML 1.3 metamodelyje komponentai ir klasės yra panašūs (realizuoja interfeisų aibę, gali turėti egzempliorius, dalyvauti priklausomybės, apibendrinimo, asociacijos ryšiuose), tačiau reikia paminėti keletą esminių skirtumų. Klasės vaizduoja logines abstrakcijas, o komponentai yra fizinės sistemos dalys. Kitaip tariant, komponentus galima realizuoti sistemos mazguose. Komponentas yra fizinė loginio modelio elementų, tokių kaip klasės, realizacija. Klasė turi atributus ir operacijas, o komponentai – tik operacijas, pasiekiamas per interfeisus. Komponentų diagramoje pateikiamos priklausomybės tarp programinių komponentų, kartu ir išeities kodo, binarinio kodo ar vykdomųjų komponentų. Modeliuojant verslą, programiniai komponentai papildomi verslo procedūromis ir dokumentais. Programinės įrangos modulį siūloma vaizduoti, kaip komponento stereotipą.

Komponentas, kaip vykdomas sistemos elementas, vaizduojamas specializuota klase, turinčia išorinę specifikaciją, išreikštą per interfeisus ir vidinę realizaciją. Taip pat, yra apibrėžiami komponentų konektoriai (angl. *connectors*), kurie sujungia (anlg. *wire*) komponentus pagal suderinamumą. Komponentas yra keičiama sistemos dalis, kuri gali būti pakeista projektavimo arba vykdymo metu, kitu komponentu, siūlančiu lygiavertį funkcionalumą, remiantis jo interfeisų suderinamumu. Sistemą galima papildyti naujais komponentais, papildančiais jos funkcionalumą.

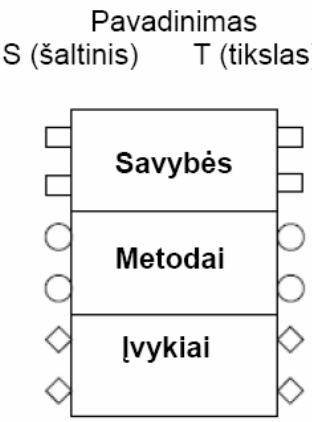
UML komponentas nagrinėjamas „juodosios dėžės“ ir „baltosios dėžės“ požiūriu. Komponento išorėje (komponentas kaip „juodoji dėžė“) matomos savybės ir

operacijos. Gali būti, kad pvz., būsenos mašinos protokolas prijungiamas prie interfeiso, porto ar paties komponento, išvardinant komponento išorėje esančių operacijų iškvietimų seką.

Komponento viduje („baltosios“ dėžės požiūriu) yra iš išorės nematomos (angl. *private*) savybės ir jas realizuojantys klasifikatoriai. Kitaip tariant, matoma išorėje stebimos elgsenos realizacija.

Komponento modelis apibrėžia, kas yra komponentas, kaip jį sukurti tam tikroje infrastruktūroje. Kiekviena komponento infrastruktūra turi pakartotinai panaudojamą komponentų biblioteką, sudarytą iš sudedamųjų dalių, atitinkančių komponento modelį.

[20] šaltinyje, komponentas yra apibrėžiamas kaip savybių, metodų ir įvykių aibė. Savybės inkapsuliuoja komponento būsenas ir atributus. Metodai nusako komponento elgseną ir paslaugas. Įvykiai aprašo veiksmus, kuriuos komponentas gali inicijuoti. Programinį komponentą vaizduoja grafiniu būdu (4 pav.).



4 pav. Komponento samprata pagal [20]

Komponentas teikia paslaugas ir reikalauja paslaugų iš kitų komponentų. [4] autoriai teigia, kad komponentas nėra paslauga, nors iš komponentų gali būti kuriamos paslaugų (angl.*, service - based*) architektūros. Išskiriami du kontraktų tipai:

- ✓ naudojimo (angl. *usage*) – tarp komponento objekto interfeiso ir jo klientų;
- ✓ realizacijos – tarp komponento specifikacijos ir realizacijos.

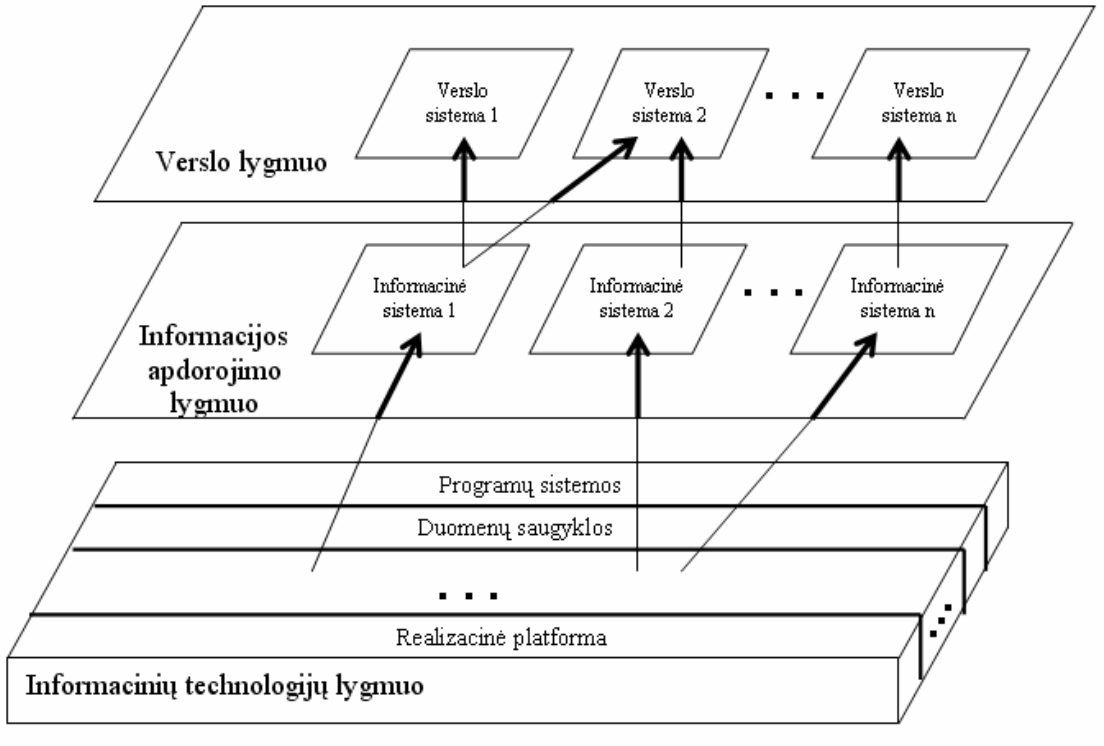
1.6. Integruotos verslo sistemos komponentinė samprata

Kiekvieną IVS sudaro trijuose lygmenyse veikiančios sistemos - verslo, informacinės ir programų sistemos (5 pav.). Verslo sistema yra suprantama kaip, „integruota

veiklų visuma, skirta tam tikriems socialiniams - ekonominiams (dažniausiai ilgalaikiams) tikslams siekti, kuriančią tam tikrą funkcionalumą, išplaukiantį iš siekiamų tikslų pobūdžio, ir tenkinančią tam tikrus nefunkcinio pobūdžio reikalavimus, užtikrinančius racionalų išteklių naudojimą, patikimumą bei kitas pageidautinas tos sistemos savybes" [6]. Organizacijai siekiant savo tikslų, svarbu, kad visų lygmenų sistemos būtų tinkamai integruotos. Organizacijos veiklos pokyčiai daro įtaką verslo sistemoms, kadangi informacinė sistema yra vienas iš verslo sistemos posistemių, sudarytų iš tarpusavyje susijusių informacijos apdorojimo procesų. Taigi, turi būti užtikrinamas, pokyčių verslo lygmeniu adekvatus atspindėjimas informacinės sistemos atžvilgiu.

Programų sistema suprantama, kaip integruota programų, failų, duomenų bei žinių bazių ir kitų komponentų visuma, skirta tam tikros klasės uždaviniams spręsti arba tam tikriems įrenginiams ar procesams valdyti. Taigi, jos yra vienos iš svarbiausių verslo sistemų sudėtinių dalių. Programų sistemos kompiuterizuoja informacijos apdorojimo procesus, todėl turi modeliuoti tuos pačius objektus, kuriais naudojasi informacinė sistema. Kitaip tariant, informacinės sistemos objektai perkeliami į programų sistemas, kur yra modeliuojami programų sistemos objektais. Kaip ir verslo sistemų atveju, šis "perkėlimas" nėra tiesioginis - vienas informacinės sistemos objektas gali būti modeliuojamas keliais programų sistemos objektais ir atvirkščiai, vienas programų sistemos objektas gali modeliuoti kelis informacinės sistemos objektus. Be to, programų sistemos turi savus, vidinius objektus, neturinčius atitikmenų informacinėse sistemose [6].

Apibendrindamas pabrėšiu, kad visų trijų lygmenų sistemos turi sudaryti vieningą visumą. Vadinasi, turi būti kuriamos integruotos verslo informacinės sistemos.



5 pav. Organizacijos kaip trijų lygmenų sistema

Integruotos verslo sistemos yra didelės ir sudėtingos. Pagrindinės jų kūrėjų problemos - efektyvus tvarkymasis su sudėtingumu ir užtikrinimas pakankamai greito bei paprasto sukurtų sistemų modifikavimu. Vienas galimų sprendimo būdų - komponentinės inžinerijos metodų taikymas. Pastebėsiu, kad naudojant komponentinį, požiūrį būtų garantuotas verslo, informacinių ir programų sistemų, kaip integruotos trijų lygmenų sistemos, unifikavimas. Komponentinę integruotą verslo sistemą apibrėšiu, kaip jos komponentų, kuriamų atsižvelgiant į verslo, informacijos apdorojimo ir informacinių technologijų lygmens tarpusavio ryšius ir turinčių standartizuotus kontraktiškai specifikuotus sąveikos interfeisus, bei realizuojamų nepriklausomai vienas nuo kito, visumą. Verslo sistemos komponentai turi tenkinti tam tikrus standartus, sudarančius galimybę juos jungti tarpusavyje, panaudojant leistinus komponentų jungimo būdus. Taigi, komponentinis požiūris turi būti užtikrinamas visais organizacijos lygmenimis (5 pav.), be to, visuose šių sistemų gyvavimo ciklo modelio stadijose.

Kuriant komponentinės verslo sistemas, projektavimo eigoje turi būti pasirenkama tinkama architektūra, aprašanti:

- ✓ verslo sistemos komponentus;
- ✓ leistinas verslo sistemų komponentų jungtis (t. y. kaip jie gali būti jungiami vienas su kitu);

- ✓ darnos reikalavimus, ribojančius IVS veikimą, kad būtų išvengta nepageidautinų efektų.

Viena vertus, šiuo metu naudojamos verslo sistemų kūrimo metodikos paprastai skirtos verslo sistemoms projektuoti siaurąja prasme, antra vertus, nėra tokių verslo sistemų kūrimo darnios ir išsamios teorijos. (IVS siaurąja prasme suprantama kaip programų, duomenų bazių ir aparatūros sistema).

1.7. Svarbiausios komponentinių IVS kūrimo problemos

Trumpai aptarsiu svarbiausias problemas, kurias reikia išspręsti, norint formalizuoti ir realizuoti komponentinės integruotas verslo sistemas.

Komponentinės IVS konceptų ir apibrėžčių visumos sudarymas. Nors komponentinė programų sistemų inžinerija yra pakankamai brandi, nėra darnios, nuo technologijos nepriklausančios, konceptų ir apibrėžčių visumos. Tas teiginys galioja ir komponentinėje IVS inžinerijoje. Kitaip tariant, komponentinės paradigmos kontekste reikia aiškiai apibrėžti, kas yra verslo sistemos komponentas, kokie galimi ryšiai tarp komponentų, kaip sudaromi sudėtiniai komponentai ir kt.

Elementarieji (primityvieji) IVS komponentai. Gerai suprojektuota sistema turi struktūrą, kuri dalo ją į tinkamus ir vienareikšmiškai apibrėžiamus sudėtinius elementus. Tai garantuoja sistemos darną, rekonfigūruojamumą, nesudėtingą realizaciją bei modifikavimą, ir panašiai. Komponentai - sudėtinės dalys, iš kurių konstruojama komponentinė IVS. Jei priimama tokia prielaida, reikia atsakyti į šiuos klausimus:

- ✓ kas yra elementarieji verslo sistemos komponentai (struktūriniai primityvai) ir, kuo jie skiriasi nuo kitokių elementarių sudedamųjų dalių, kurių nevadinsime komponentais;
- ✓ kokio granuliariškumo turi būti elementarieji verslo sistemos komponentai;
- ✓ kokių rūšių turi būti elementarieji verslo sistemos komponentai, nes akivaizdu, kad norint sukurti IVS, nepakanka vien tik programinių komponentų.

Sudėtinių IVS komponentų sudarymas. Elementarūs komponentai yra sudėtinių komponentų sudedamosios dalys. Kitaip tariant, komponavimas yra esminis sudėtingesnės sistemos ar jos dalies kūrimo metodas. Sudėtinių komponentų sudarymo problema turi būti sprendžiama trim aspektais. Pirma, turi būti žinoma, kokių rūšių elementarieji komponentai

kartu gali sudaryti vieną sudėtingesnį junginį. Antra, turi būti nustatyti leistini komponentų jungimo būdai. Trečia, turi būti nustatyti leistini sudėtinių dalių jungčių tipai.

IVS komponentų tipų aibės nustatymas. Tiek projektuojant IVS komponentus (kitaip dar šis procesas vadinamas verslo sistemos komponentų projektavimu pakartotiniam panaudojimui (angl. *for reuse),* tiek pasirenkant konstruojamos IVS komponentinę bazę (angl., *with reuse)* turi būti žinoma, kokių tipų komponentai yra prasmingi ir leistini. Vadinasi, turi būti apibrėžta, kaip gali būti klasifikuojami IVS sudarantys komponentai. Šiuo metu žinomo komponentų klasifikavimo nepakanka sistemos kūrimo atveju. Be to, reikia nustatyti, kokių tipų komponentų aibės pakanka bet kuriai sistemai sukurti.

Nauji komponentinės analizės ir projektavimo metodai. Komponentinė integruotų verslo sistemų inžinerija - nauja verslo sistemų inžinerijos šaka. Todėl akivaizdu, kad esamų metodų nepakanka. Be to, būtinas sisteminis požiūris, nes pavienių problemų sprendimo nepakanka rezultatams praktiškai taikyti. Dabar brandžiausi komponentinių programų sistemų kūrimo metodai gali būti taikomi tik verslo sistemos konstravimo stadijoje, be to, tik vieno tipo komponentams realizuoti. Kadangi komponentinis požiūris turi būti naudojamas ne tik realizuojant, bet ir analizuojant bei projektuojant, tai reikia tinkamų komponentinių sistemų analizės ir projektavimo metodų. Svarbiausieji yra šie:

- ✓ dekomponavimo metodai, naudojami IOIS projektavimo stadijoje;
- ✓ komponentų identifikavimas, apimantis nustatymą, kurių dalykinės srities fragmentų ar projektavimo metu gautų artefaktų pakartotinis panaudojimas yra prasmingas;
- ✓ komponavimo metodai.

Komponentinių informacinių sistemų architektūros. Informacinės sistemos architektūros stilius nusako šią sistemą sudarančius komponentus. Tačiau, kita vertus, komponentų ypatumai ir funkcionalumas lemia visos sistemos funkcionalumą ir elgseną. Todėl IOIS komponentai ir šių sistemų architektūra yra du neatskiriami IOIS kūrimo aspektai. Šiame kontekste iškyla dvi svarbiausios problemos:

- ✓ apibrėžti IVS architektūros stilius;
- ✓ nustatyti IVS architektūros projektavimo procesą.

*IVS komponentų sąveika ir jos specifikavimas***.** Kuriant komponentinės IVS komponentų sąveikos problema iškyla dviem aspektais. Pirma, turi būti nusakyta, kas yra leistina komponentų sąveika ir kokių metodų klasės ją gali realizuoti. Antra, reikia nustatyti, kurie architektūros elementų tipai gali būti aktyvūs, t. y. inicijuoti sąveiką. Be to, turi būti įvertinta, ar žinomų analizės ir projektavimo kalbų galimybių pakanka visiems sąveikos aspektams nusakyti.

1.8. Formalios specifikacijos apibrėžimas ir savybės

Apibendrinant literatūros šaltinius, formali specifikacija – tai sistemos savybių rinkinio išraiška kažkokioje formalioje kalboje, kažkokiame abstrakcijos lygyje. Tikslesnis apibrėžimas galimas tik tada, kai tiksliai žinome: kas slepiasi po žodžiu „sistema“, kokios savybės yra įdomios, koks abstrakcijos lygis bus naudojamas, ir galiausiai, kokia formali kalba bus naudojama.

Žodis „formalus“ dažnai yra maišomas su žodžiu „tikslus“. Aišku, kad pastarasis paveldi pirmąjį, bet tik ne atvirkščiai. Specifikacija yra formali, jei kalba, kuria ji parašyta yra sudaryta laikantis trijų taisyklių:

1. aiškios ir griežtos sintaksės taisyklės;
2. taisyklės aprašančios semantiką;
3. taisyklės, kuriomis galima įžvelgti naudingą informaciją specifikacijoje.

Norint parašyti teisingą specifikaciją, reikia įvertinti šiuos aspektus:

- ✓ Specifikacija turi būti adekvati – ji turi labai tiksliai nusakyti problemą.
- ✓ Specifikacija turi būti nuosekli – jei paimsime visas specifikuotas savybes į visumą, ji turi būti teisinga.
- ✓ Specifikacija turi būti nedviprasmiška – negali turėti dviprasmybių suvokiant kurį nors teiginį, kaip tiesą.
- ✓ Specifikacija turi būti pilna – žemesnio lygio savybių rinkinys turi būti pakankamas, kad būtų galima apibūdinti aukštesnio lygio teiginį.
- ✓ Specifikacija turi būti minimali – negali turėti perteklinės informacijos ar savybių, kurios nesusijusios su problema.

1.9.Sudėtingų sistemų formalizavimo metodai

Kompiuterių moksle formalizavimo metodai apibrėžiami kaip matematika pagrįsta metodika, skirta programinės ar techninės įrangos verslo sistemų specifikavimui, plėtojimui ir verifikavimui. Formalizavimas ypač reikšmingas didelės integracijos sistemose (kur svarbus saugumas ir patikimumas), kad būtų užtikrintas teisingas sistemos tobulinimo procesas ir išvengta klaidų atsiradimo. Formalizavimo metodai efektyviausi ankstyvosiose vystymo, reikalavimų ir specifikacijų dokumentų ruošimo stadijose, tačiau gali būti naudojamas ir jau įgyvendintos sistemos formalizavimui.

Formalizavimo metodai gali būti naudojami keliais būdais :

a) Sistema specifikuojama formaliai, bet tolesnėse stadijose tobulinama neformaliai. Dažniausiai tai mažiausiai resursų reikalaujantis ir duodantis pakankamai gerus rezultatus panaudojimo atvejis.

b) Formalus tobulinimas ir verifikavimas gali būti naudojamas bandant kurti sistemą formalesniais metodais. Pavyzdžiui, specifikacijoje pateiktos ir įrodytos savybės vystymo metu perkeliamos į sistemos realizaciją. Šis naudojimo atvejis tinkamiausias toms sistemoms, kuriose ypatingas dėmesys turi būti skiriamas saugumui ar patikimumui.

c) Teoremų įrodymo priemonės gali būti panaudotos tam, kad būtų garantuotos visiškai formalios automatiškai patikrinamos tiesos. Tai labai brangus panaudojimo atvejis, tačiau praktikoje dažnai atsiperkantis, ypač jei klaidų kainos sistemoje labai didelės (pvz. kritinių mikroprocesoriaus dalių projektavimas).

Formalizavimo metodus grubiai galime sukvalifikuoti į grupes pagal jų semantines savybes ir pateikimo stilių:

- ✓ *Žymėjimo semantikos* grupei priklauso tie metodai, kuriais sistemos prasmė išreiškiama matematinės teorijos erdvėje. Šie metodai remiasi tuo, kad aprašinėjama sistema gali būti nesunkiai suprasta ir aprašyta matematinėje erdvėje, tačiau toli gražu ne kiekviena sistema intuityviai ar iš prigimties gali būti išreiškiama kaip funkcija.
- ✓ *Veikimo semantikos* grupei priklauso metodai, kuriais sistemos prasmė išreiškiama kaip paprastesnio skaičiavimo modelio įvykių seka. Šios grupės metodų šalininkai pabrėžia jų modelių aprašų paprastumą ir aiškumą, tačiau kritikai teigia, kad neišspręsta pačios semantikos problema. Nors sudaromi ir modeliai, tačiau semantika nepakankamai apibrėžta ir naudojant sukelti problemų.
- ✓ *Aksiomatinės semantikos* grupei priskiriami metodai, kuriais sistemos prasmė išreiškiama išankstinėmis ir po to einančiomis sąlygomis, kurios yra atitinkamai teisingos prieš ir po tam tikros užduoties atlikimo. Gana ryškus šių metodų ryšys su klasikine logika. Tačiau šios grupės metodų kritikai pastebi, kad aksiomatinė semantika aprašytos sistemos niekada tiksliai neapibūdina kaip sistema iš tikrųjų „elgiasi“ (tik kokia jos būsena prieš ir po tam tikrų įvykių).

Kai kurie praktikai tvirtina, kad formalizavimo metodų visuma pernelyg viršija pilnai sistemos specifikacijai ar modeliui aprašyti reikalingą formalizmą. Jie tvirtina, kad aprašomosios kalbos išraiškingumas, taip pat modeliuojamų sistemų sudėtingumas padaro

visapusiško formalizavimo procesą labai sudėtingu ir sprendimui reikalaujančiu didelių sąnaudų uždaviniu. Kaip alternatyva, buvo pasiūlyta įvairių nesudėtingų formalizavimo metodų, kuriuose akcentuojamas dalinis formalizmas ir dėmesys nukreipiamas į jų pritaikomumą.

Pasaulyje žinoma daugybė formalizavimo metodų ir notacijų, tačiau dažniausiai randami šie:

- ✓ PLA (piece-linear aggregates)
- ✓ Agentų modeliai,
- ✓ Esterela,
- ✓ Petri tinklai,
- ✓ Z.

2. INTEGRUOTOS VERSLO SISTEMOS FORMALIZAVIMO TYRIMAS

2.1. Organizacijos sprendimų priėmimo sistemos formalizavimas

Integruota verslo sistema (IVS) – tai visa infrastruktūra, organizacija, personalas ir komponentai skirti surinkti, apdoroti, saugoti, perduoti, atvaizduoti, paskleisti ir sunaikinti duomenis. Jei nė viena sistemos dalis nėra automatizuota, integruotą sistemą sudaro tik du lygmenys: verslo ir informacinės sistemos.

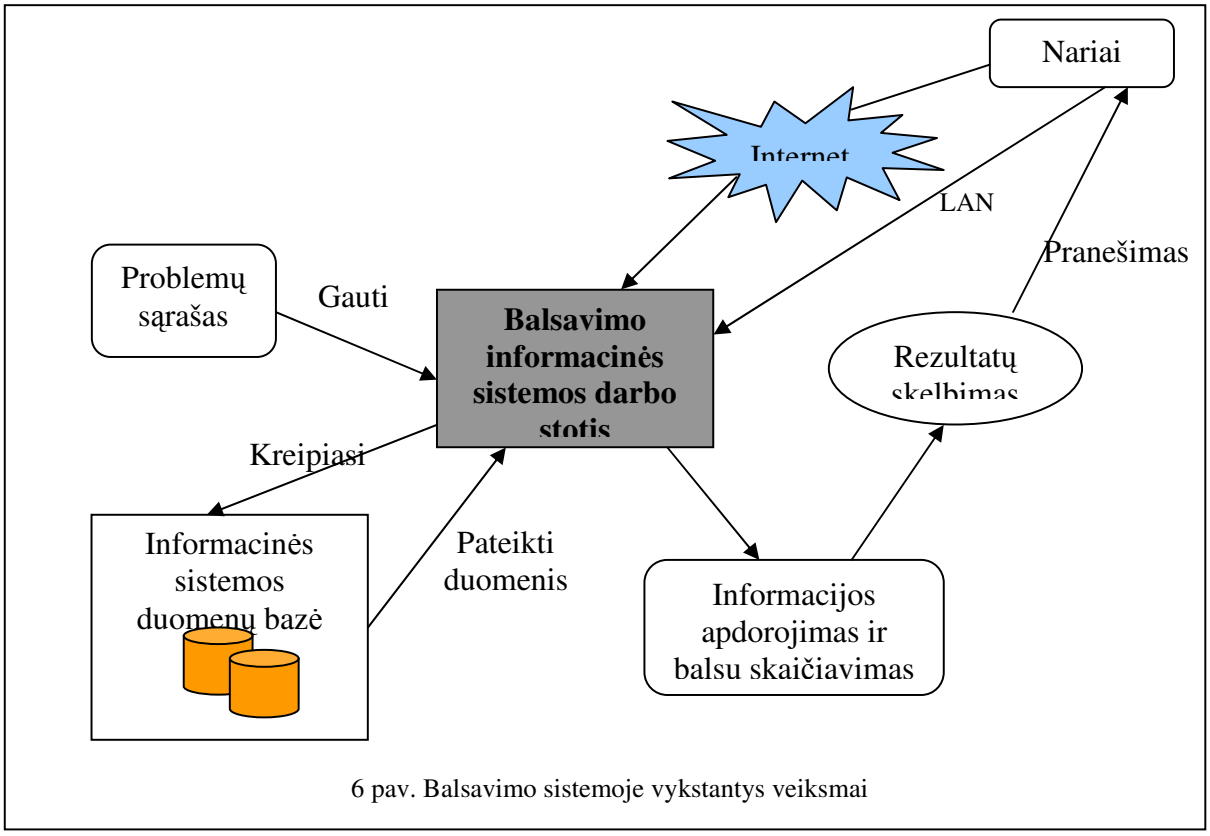
IVS sistemą sudaro veiklai vykdyti reikalingi informacijos apdorojimo procesai (planavimas, analizė, apskaita, auditas, sprendimų priėmimas ir pan.) ir jų realizavimo priemonės. Veiklos objektai (ištekliai, priemonės, rezultatai ir kt.) integruotoje verslo sistemoje modeliuojami (aprašomi) atitinkamais informaciniais objektais. Ji realizuoja dalį veiklos pagalbinių procesų ir teikia visas veiklai vykdyti reikalingas informacines ir skaičiuojamąsias paslaugas. Integruota verslo sistema sukuria veiklos informacinę infrastruktūrą ir yra prasminga tuomet, kuomet tiesiogiai ar netiesiogiai prisideda prie veiklos (jos bazinių procesų) rezultatų gerinimo.

Šiuolaikinėms IVS tenka visai kitoks vaidmuo. Jos yra strateginis įmonės elementas, nuo kurio esminiai priklauso verslo konkurencingumas. IVS programinė įranga ne tik palaiko įmonės verslo strategijas, bet ir daro tiesioginę įtaką tų strategijų parinkčiai ir formulavimui. Tai kelia naujus iššūkius ne tik IVS kuriantiems praktikams, bet ir informacinių bei programų sistemų inžinerijos teorines problemas nagrinėjantiems mokslininkams. Jie turi suvokti, kaip IVS tarpusavyje reikia sieti verslo, informacines ir programų sistemas, ir išsiaiškinti, dėl kokių priežasčių funkcinės programų sistemų galimybės taip dažnai atitrūksta nuo verslo tikslų ir realių verslo poreikių. Tokie atotrūkiai pažeidžia įmonės veiklos darną ir todėl į IVS įdėtos investicijos neduoda tokio rezultato, kokio buvo tikėtasi. Standish Group 2003 m. tyrimo duomenimis (Hartman, 2006) dėl to prarandama apie 25% investicijų. Taigi būtina atsakyti į klausimą, kokios yra programinės įrangos atotrūkio nuo verslo poreikių priežastys ir kaip galima tą atotrūkį pašalinti arba bent jau sumažinti.

Be abejo, problema labai sudėtinga ir kol kas ne iki galo išnagrinėta.

Integruotos verslo sistemos vaidina labai svarbų vaidmenį visuomenėje. Dažniausiai firmos veikla neįmanoma be tinkamos ir geros verslo sistemos. Dažnas atvejis kuomet save gerbianti firma tiesiog privalo turėti jai pritaikytą integruotą verslo sistemą, kuri būtų patogi naudoti, tiek klientams, tiek ir darbuotojams. Tobulėjant informacinėms technologijoms ir visuomenei, integruotos verslo sistemos įgyja vis didesnę reikšmę kiekvienos įmonės valdyme.

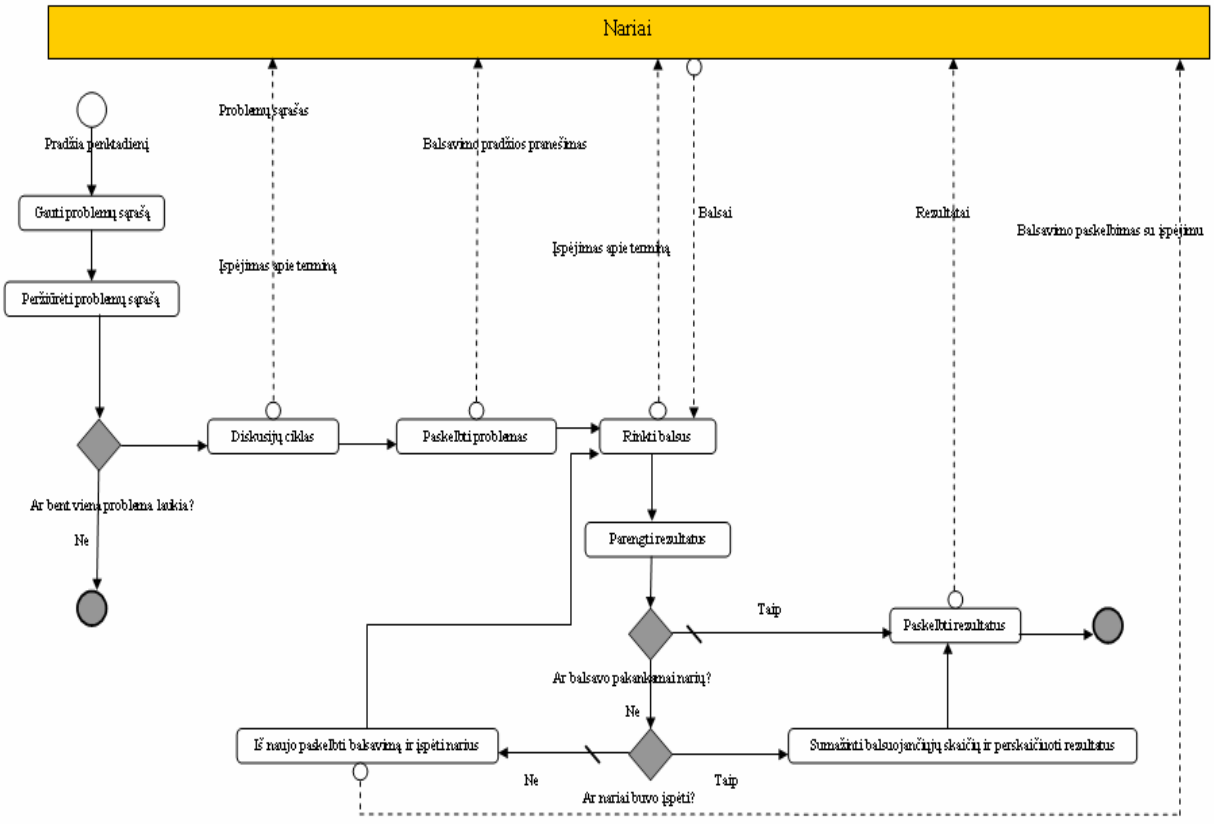
Tam, kad iliustruoti visus išanalizuotus teorijos šaltinius, reikia pasirinkti konkrečią verslo sistemą ir ją išanalizuoti konceptuliai. To pasekoje pasirinkau uždaros organizacijos vienijančios daug narių sprendimų (balsavimo) priėmimo verslo sistemą. Tam, kad geriau įsivaizduoti sistemos struktūrą ir vykstančius procesus ją pavaizduosiu paveikslėlyje (pav. 6):



6 pav. Balsavimo sistemoje vykstantys veiksmai

Pavyzdžio atveju, sprendimai priimami balsavimu. Tariamos įmonės (organizacijos) balsavimo procesas prasideda penktadienį – tada gaunamas problemų sąrašas. Sąrašas peržiūrimas ir, jei yra aktulių problemų, pradedamas diskusijų ciklas. Po diskusijų ciklo, paskelbiamas balsavimas ir savaitę vyksta balsavimas. Tuomet rengiami rezultatai. Po jų tikrinama, ar balsavo pakankamai balsavimo teisė turinčių narių. Jei ne, tikrinama, ar nariai buvo perspėti apie balsavimą. Jei ne – skelbiamas pakartotinis balsavimas perspėjant narius ir vėl renkami balsai. Jei balsavo pakankamai narių, skelbiami rezultatai. Jei balsavo nepakankamai narių ir visi buvo perspėti, sumažinamas balsuojančių skaičius, perskaičiuojami rezultatai ir jie skelbiami.

Organizacijos sprendimo priėmimo verslo procesas pavaizduotas paveikslėlyje (7 pav.). Kaip pastebėjote verslo procesas aprašytas BPMN (angl., *Business Process Modelling Notation*) modeliavimo kalba.

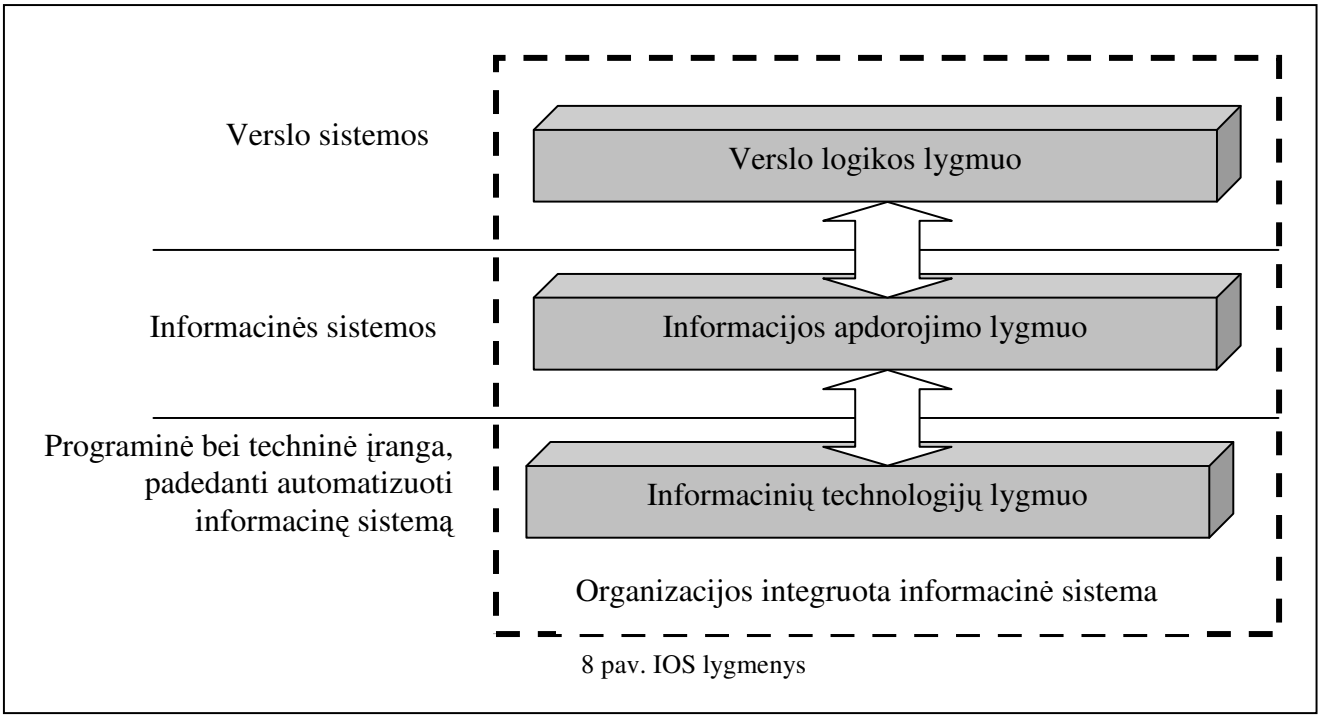


7 pav. Balsavimo proceso verslo proceso modelis

Ištyrus sprendimo priėmimo verslo procesą, reikia konceptualiai išskirti naudojamus komponentus. Ši sistema realizuojama apjungiant integruotos verslo sistemos lygmenis. Trys pagrindiniai diskusijų ciklo lygmenys yra šie:

- ✓ Verslo lygmuo;
- ✓ Informacijos apdorojimo lygmuo;
- ✓ Informacinių technologijų lygmuo.

Tarpusavyje, šie trys lygmenys yra glaudžiai susiję. Galima drąsiai teigti, kad tai pagrindiniai integruotos verslo sistemos lygmenys. Kartu šie lygmenys sudaro integruotą organizacijos sprendimų priėmimo sistemą. Ši sistema paprastai apibrėžiama kaip trijų lygių sistema, susidedanti iš verslo, informacijos ir jas palaikančių programinės bei techninės įrangos sistemų. „Balsavimo“ sistema yra sudedamoji integruotos verslo sistemos dalis, ir programinė sistema – būtina informacinės sistemos dalis. Integruota organizacijos balsavimo sistema vadinama integruota tada ir tik tada, kai žemesnio lygio sistemos „žino“ apie aukštesnio lygio sistemas ir žemesnio lygio sistemos yra sukurtos taip, kad veiktų pagal aukštesnio lygio sistemų diktuojamas taisykles. Balsavimo sistemos lygmenys ir jų sąsajos pavaizduotos 8 paveiksle.

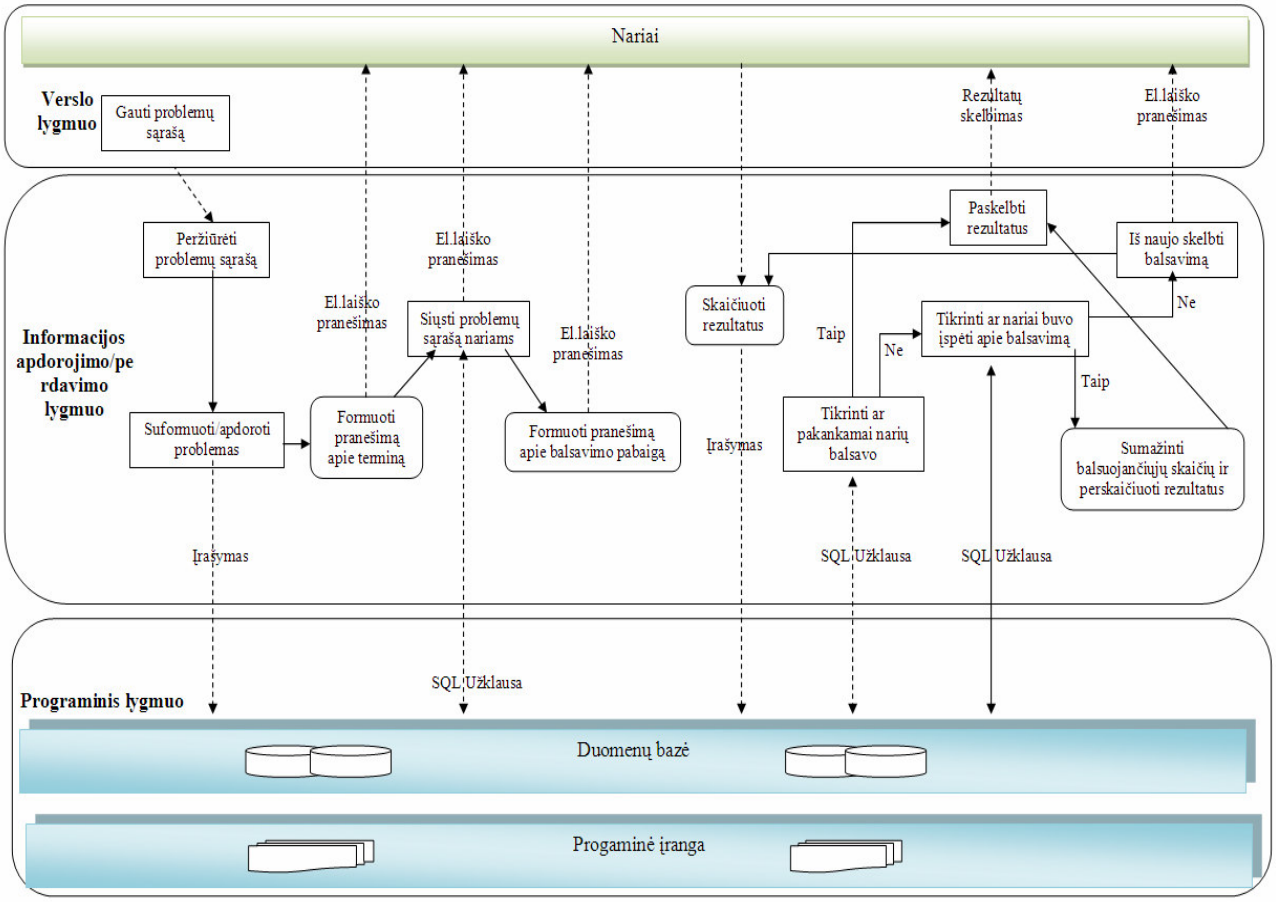


8 pav. IOS lygmenys

Iš integruotos verslo sistemos apibrėžimo, galima daryti išvadą, kad IVS turi būti kuriama vadovaujantis verslo modeliu. Mano manymu, verslo modelio esybės (objektai, procesai ir pan.) yra susietos su informacine sistema. Kadangi apibrėžimas teigia, kad IVS reikia kurti turint verslo modelį, tai šiuo atveju remsiuosi balsavimo proceso modeliu. Tai puikus pavyzdys norint išnagrinėti šią sistema visuose trijuose lygmenyse ir apibūdinti kiekvieną iš jų atskirai. Tam, kad gerai suprasti ir žinoti organizacijos integruotos verslo sistemos veikimo principą, reikia gerai žinoti, kas atliekama kiekviename sistemos lygmenyje: kokie procesai vykdomi, kokie komponentai naudojami ir t.t.

Integruota verslo sistema – tai pakankamai sudėtingas, didelės apimties ir svarbos darinys, kuriame derinami žmogiškieji ir technologiniai resursai. Kadangi organizacijos verslo logika gali kisti priklausomai nuo aplinkos poveikio, tai ir visą sistemą turi būti galimybė adaptuoti prie pakitusių reikalavimų ir sąlygų. Paprastai kinta tik tam tikra sistemos dalis. Taigi, perprojektuoti visą integruotą sistemą šiuo atveju, būtų nelogiška: sistema turėtų būti skaidoma į tam tikrais ryšiais susijusius, tačiau vienas nuo kito nepriklausančius modulius. Kas be ko, šiais laikais greitai kinta ne tik verslo aplinka, bet ir technologijos. Prie šių pokyčių sistema taip pat turėtų prisitaikyti. Reiktų paminėti, kad lygmenys tarpusavyje susiję ir naudojasi vienas kito paslaugomis. Tai puikiai iliustruoja 5 paveikslas.

Turint pavyzdį, kaip turėtų atrodyti integruota organizacijos verslo sistemos struktūra, galima nubraižyti visus tris lygmenis ir juose vykstančius procesus, funkcijas ir t.t. Tai iliustruoja sekantis paveikslas (9 pav.).



9 pav. Balsavimo ciklo sistemos lygmenys ir juose vykstantys procesai

Kaip matome paveikslėlyje (9 pav.), aiškiai išskirti trys balsavimo informacinės sistemos lygmenys: verslo lygmuo, informacijos apdorojimo/perdavimo lygmuo ir programinis lygmuo. Pats didžiausias ir stambiausias yra vidurinysis lygmuo. Žinoma, šiame sluoksnyje atliekama daugiausiai veiksmų, kadangi reikia apdoroti duomenis ir juos perduoti.

Tačiau šiandien, įmonės integruotos sistemos (ĮIS) dedamąsias projektuojantys ir įgyvendinantys specialistai (verslo konsultantai, informacinių sistemų inžinieriai, programų sistemų inžinieriai ir kt.) dažniausiai domisi tik savo veiklos baru ir tas priklausomybes ignoruoja. Dėl to, ĮIS programinė įranga atitrūksta nuo tikrųjų verslo poreikių. Taigi, viena iš pagrindinių atotrūkio priežasčių yra netinkama sistemos reikalavimų formulavimo metodika. Siekdami sukurti realią naudą teikiančią ĮIS programinę įrangą, programų sistemų inžinieriai privalo dirbti palaikydami glaudų ryšį su verslo konsultantais ir inžinieriais, išsiaiškinti ne tik tiesioginius kuriamos programinės įrangos reikalavimus, bet ir visus verslo procesų,

informacijos apdorojimo procesų, programinės įrangos ir organizacinės bei socialinės verslo aplinkos tarpusavio sąryšius. Analitikai turi perprasti, ne tik įmonėje vykstančius verslo procesus, bet ir jų tikslinę motyvaciją, taip pat organizacinę ir socialinę aplinką, kurioje tie procesai vyksta. Tai ne tik keičia analitikų darbo stilių, bet ir reikalauja naujo požiūrio į informatikos studijų programas. Analitikas privalo išmokti formuluoti, analizuoti ir vertinti reikalavimus, neatsiedamas jų nuo visos organizacijos konteksto. Be to, analitikui prireikia tokių modeliavimo priemonių, kurios leistų kurti modelius, aprašančius ne tik verslo procesų logiką, bet ir ryšius, siejančius tuos procesus su jų organizacine bei socialine aplinka, leistų atlikti formalius samprotavimus apie tų modelių savybes

Norint sukurti tokias modeliavimo kalbas, visų pirma reikalinga tokia sąvokų sistema, kurią vartojant būtų galima samprotauti tiek apie verslo, tiek apie informacines, tiek apie programų sistemas bei jų tarpusavio sąryšius. Bendra sąvokų sistema yra reikalinga ir tam, kad programų sistemų inžinieriai ir verslo konsultantai galėtų bendrauti tarpusavyje ir aptarti vieni su kitais skirtingų ĮIS lygmenų reikalavimus. Ar galima sukurti tokią sąvokų sistemą ir, jeigu tai pavyktų, ar nebus ji tokia bendra, kad ją vartojant atliekami samprotavimai taps tiek abstraktūs, jog bus praktiškai beverčiai? Mano nuomone, šis uždavinys yra išsprendžiamas.

Pagrindinė sąvoka, nuo kurios reikia pradėti formalizuoti sistemą, yra sistemos sąvoka. Tokią sąvoką pateikia bendroji sistemų teorija. Joje išsamiai išnagrinėtos formaliosios sistemų savybės ir sukurta visa sąvokų sistema, leidžianti formuluoti įvairius teiginius, apie bet kokios prigimties sistemas. Deja, mano tikslams ši sąvokų sistema yra mažai tinkama, nes ĮIS kontekste verslo, informacijos apdorojimo ir programų sistemos yra nagrinėjamos visai kitu, inžineriniu, aspektu.

Pasinaudojant sąsajomis sistemai galima duoti nurodymus, prašyti jos paslaugų ir galbūt stebėti, kaip sistemai veikiant kinta vieni ar kiti jos parametrai. Kai kurių sistemų sąsajos yra procedūrinės ir gali būti labai sudėtingos. Norint pasinaudoti tokia sąsaja reikia atlikti visą, kartais netgi labai sudėtingų veiksmų seką. Taip yra, pavyzdžiui, norint pasinaudoti kokios nors teisinės sistemos, tarkime, teismo, teikiamomis „paslaugomis“. Sistemos teikiamų paslaugų pobūdis yra nusakomas tos sistemos *funkciniais reikalavimais,* o kitos sistemų savybės (našumas, patikimumas, panaudojamumas ir kt.) - jos *nefunkciniais reikalavimais.*

Kuo gali būti naudinga tokių bendrų sąvokų sistema? Visų pirma, ji sudaro galimybes kalbėti apie visus ĮIS lygmenis, vartojant tuos pačius terminus. Pradėjus tai daryti, paaiškėja kai kurios verslo ir jį palaikančios programinės įrangos atotrūkio priežastys: pavyzdžiui, norint išvengti skirtingų lygmenų sąsajų atotrūkių, ĮIS programinės įrangos sąsajos privalo būti sudarytos iš atitinkamų verslo sistemos sąsajų. Vadinasi, dalykinę sritį analizuojantis analitikas turi gebėti į analizuojamą verslą pažvelgti per bendrosios sistemų inžinerijos prizmę,

išsiaiškinti analizuojamos sistemos sąsajas, jas specifikuoti ir iš tokių specifikacijų sudaryti integruotos sistemos ir kuriamos programinės įrangos sąsajų reikalavimus.

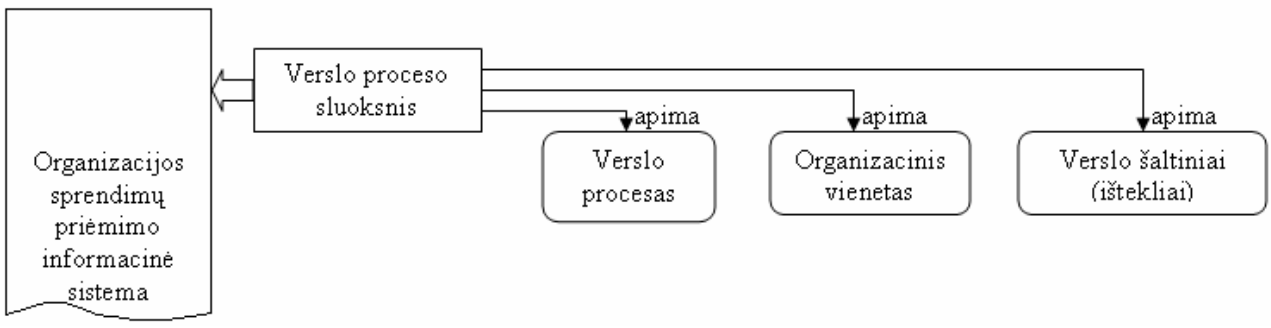
Panašiai yra ir su kitais funkciniais ir nefunkciniais reikalavimais. Analitikas privalo suformuluoti funkcinius ir nefunkcinius analizuojamos verslo sistemos reikalavimus ir iš jų išvesti atitinkamus integruotos sistemos ir kuriamos programinės įrangos reikalavimus. Pavyzdžiui, verslo trukmės reikalavimai lemia programinės įrangos našumo reikalavimus. Taip yra ir su patikimumo, apsaugos bei kitais nefunkciniais reikalavimais. Beje, pastaraisiais metais daug kalbama apie vadinamąsias verslo taisykles ir jų naudojimą kuriant integruotų verslo sistemų palaikymo programinę įrangą. Iš tiesų, verslo taisyklės yra dalis verslo sistemos reikalavimų. Traktuojant jas kaip specifinę informaciją, prisireikia savitų modeliavimo konceptų ir apskritai yra suardoma visos bendrų sąvokų sistemos darna. Pavyzdžiui, pasidaro nebeaišku, ar verslo taisykles reikia klasifikuoti pagal tą pačią schemą, pagal kurią klasifikuojami programų sistemų reikalavimai, ar kaip nors kitaip. Suvokus, jog verslo taisyklės yra tik darbinis terminas, tokių sunkumų galima lengvai išvengti.

Panašią naudą teikia ir kitų bendrosios sistemų inžinerijos sąvokų, pavyzdžiui, *komponentas, gyvavimo ciklas, technologinis procesas* ir kt., vartojimas, kalbant apie bet kurį IVS lygmenį. Tačiau būtų klaidinga manyti, jog bendrosios sistemų inžinerijos sąvokos tiesiogiai gali būti perkeltos į verslo, informacinių ir programų sistemų inžineriją. Be abejo, taip nėra. Nors ir vartojamas tas pats terminas, jo turinys keičiasi. Kitaip tariant, sąvokos turi būti specializuojamos atsižvelgiant į specifinius konkretaus pobūdžio sistemų ypatumus. Pavyzdžiui, galiu sakyti, kad bet kuri funkcinės architektūros sistema yra tam tikrų funkcinių vienetų agregatas. Funkcinis vienetas yra suprantamas kaip procesorius, gebantis realizuoti tam tikrą funkciją. Funkcinis vienetas gali būti realizuotas ne tik kaip programų sistema ar įrenginys, bet ir kaip pareigybė ar kaip įmonės padalinys. Pavyzdžiui, bilietus gali pardavinėti kasininkas, galima sukurti programinę įrangą, kompiuterizuojančią tam tikrą jo funkcijų dalį, bet įmanoma sukurti ir bilietų pardavimo automatą ir juo pakeisti kasininką. Taigi atrodytų, jog visus funkcinius vienetus galima nagrinėti kaip juodąsias dėžes ir, nesigilinant į jų realizavimo būdą, traktuoti kaip tam tikrus (programuojamus) procesorius. Iš tiesų taip nėra. Techniniai (aparatūra ar programomis realizuoti) procesoriai ir socialiniai (realizuojami pareigybių ar padalinių) procesoriai turi svarbių skirtumų. Techninio procesoriaus programa determinuoja visas jo elgsenos detales, o socialinio procesoriaus elgsena nėra iki galo kontroliuojama ir prognozuojama, toks procesorius turi tam tikrą veikimo laisvę, nors ir turinčią socialinių ir organizacinių ribojimų, kuriuos jis privalo tenkinti. Kitaip tariant, sąvokos *funkcinis vienetas* turinys visgi iš dalies priklauso nuo to, apie kokio pobūdžio sistemas kalbame.

2.2. Verslo logikos lygmuo

Kaip jau ir minėjau, balsavimo ciklo sistemą sudaro trys lygmenys. Pagrindinis lygmuo nuo kurio pradedama vystyti ir formalizuoti visa integruota verslo sistema yra „Verslo logikos lygmuo“. Iš esmės verslo logikos lygmuo apima visą verslo procesą ir galima sakyti atspindi visą verslo proceso esmę. Integruotos verslo sistemos kontekste ypač aktuali tampa programinės įrangos funkcinių ir nefunkcinių savybių atotrūkio nuo realių verslo poreikių problema. Darbe norima išdėstyti naują požiūrį į tai, kad, kuriant integruotas verslo sistemas, reikia verslo logikos reikalavimus nuleisti į programinės įrangos lygmenį.

Be kita ko, šia tematika vis labiau domimasi todėl, kad tradicinės verslo sistemos peraugo į integruotas verslo sistemas (IVS), jungiančias į visumą visas įmonės verslo, informacines ir programų sistemas. Tradicinėse verslo sistemose, programinė įranga paprastai kompiuterizuodavo tik apskaitos, planavimo, sprendimų priėmimo ir kitų administracinio pobūdžio užduočių vykdymą. Ji buvo traktuojama kaip savotiškas įmonių ar organizacijų „užnugaris", gaminantis reikalingus informacinius išteklius. 10 paveikslas iliustruoja tiriamos „Balsavimo sistemos“ verslo lygmens išskaidymą.



10 pav. Verslo lygmens išskaidymas

2.2.1. Verslo procesas

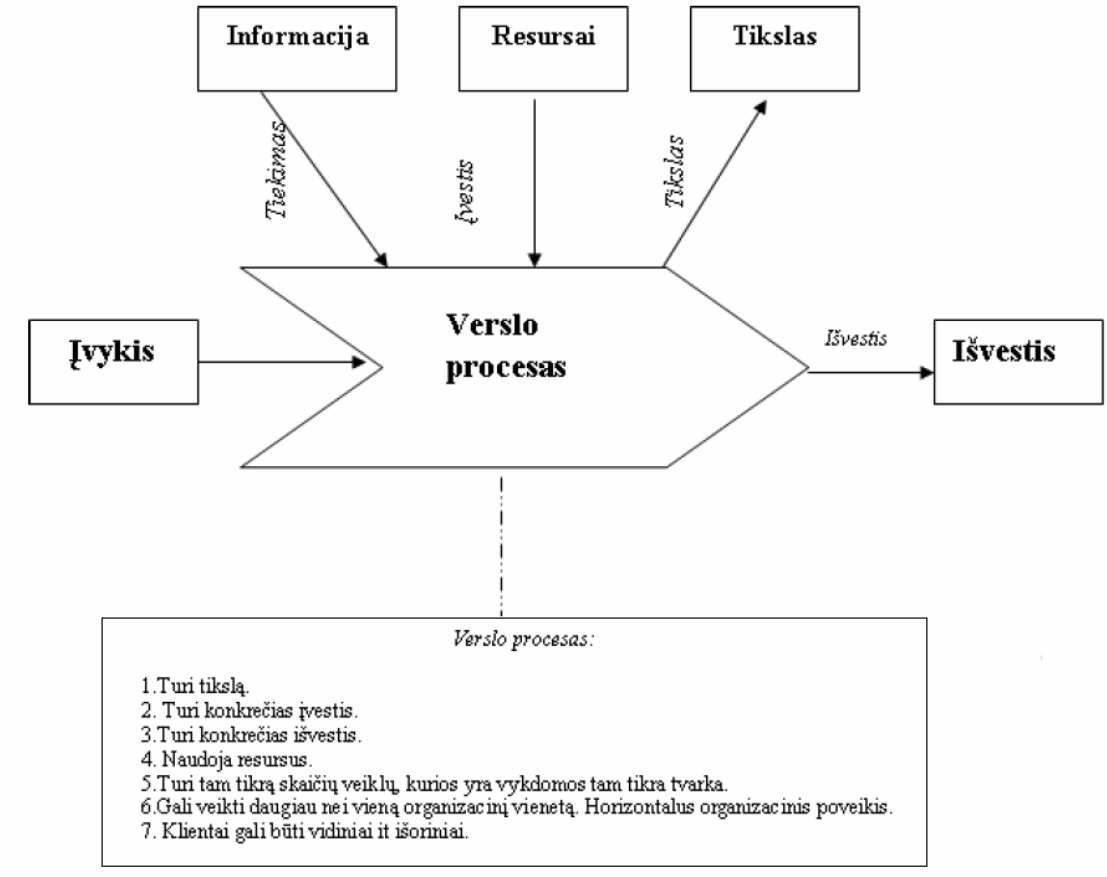
Organizacijos integruotos sistemos verslo lygmenį galima aprašyti tam tikrais verslo procesais, kurie vyksta visoje sistemoje. Bendriausiu atveju „Balsavimo sistemos“ modelis pavaizduotas (6 pav.) Taigi galima teigti, kad verslo lygmuo nusako, kokie bendrieji verslo procesai yra vykdomi. Pagrindiniai veikėjai šiame lygmenyje yra organizacijos nariai, kurie sistemos pagalba vykdo balsavimą. T.y valdo ir dirba su IVS.

Nario funkcijos yra šios:

- ✓ Narys gavęs pranešimą į elektroninio pašto dėžutę apie balsavimą, turi prisijungti prie sistemos;
- ✓ Sėkmingai prisijungęs turi išreikšti savo nuomonę organizacijos problemoms spręsti;
- ✓ Pasibaigus balsavimui narys informuojamas apie rezultatus.

Sprendimų priėmimo sistemą, įskaitant verslo, informacines ir programų sistemas, galima suvokti kaip tarpusavio sąryšiais susietų kokių nors, nebūtinai vienodų, elementų (sistemos komponentų) rinkinį, pasižymintį vientisumo bei tam tikromis funkcinėmis savybėmis. Sistemos komponentus siejantys sąryšiai yra vadinami *konstrukcijos sąryšiu.* Šio sąryšio pobūdis lemia sistemos vientisumo ir funkcines savybes. Kita vertus, funkcinės savybės priklauso nuo tikslinės sistemos paskirties arba, kitaip tariant, nuo to, kokias paslaugas sistema privalo teikti jos naudotojams. Taigi, visos sistemos, taip pat ir integruotos verslo bei informacinės sistemos, turi sąsajas jų teikiamoms paslaugoms gauti. Tiesa, kol kas daugelio abstrakčiųjų sistemų (verslo sistemų, informacinių sistemų, teisės sistemų ir pan.) sąsajos yra ištirtos gana menkai. Tokie tyrimai yra viena iš aktualiausių mokslinių problemų.

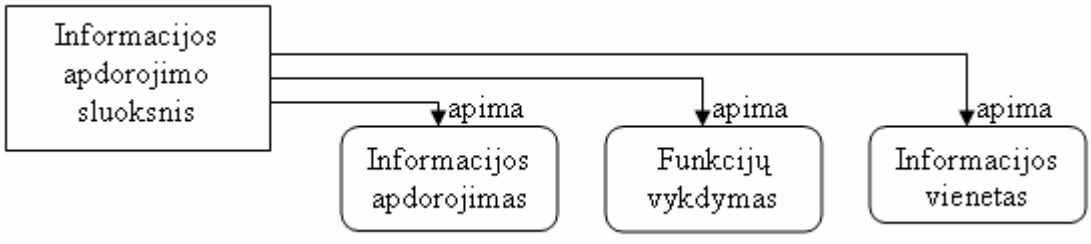
Verslo procesai vizualiai yra pateikiami diagramomis, kurias sudaro paprasta „dėžė“ su į ją įeinančiomis ir išeinančiomis rodyklėmis, prie kuriu pateikti trumpi apibūdinimai – informacija, susijusi su veikla. 11 paveiksle pateiktas verslo procesų grafinis vaizdas, kuris nusako verslo procesų modeliavimo sudedamąsias dalis.



11 pav. Verslo procesu modeliavimo apibudinimas

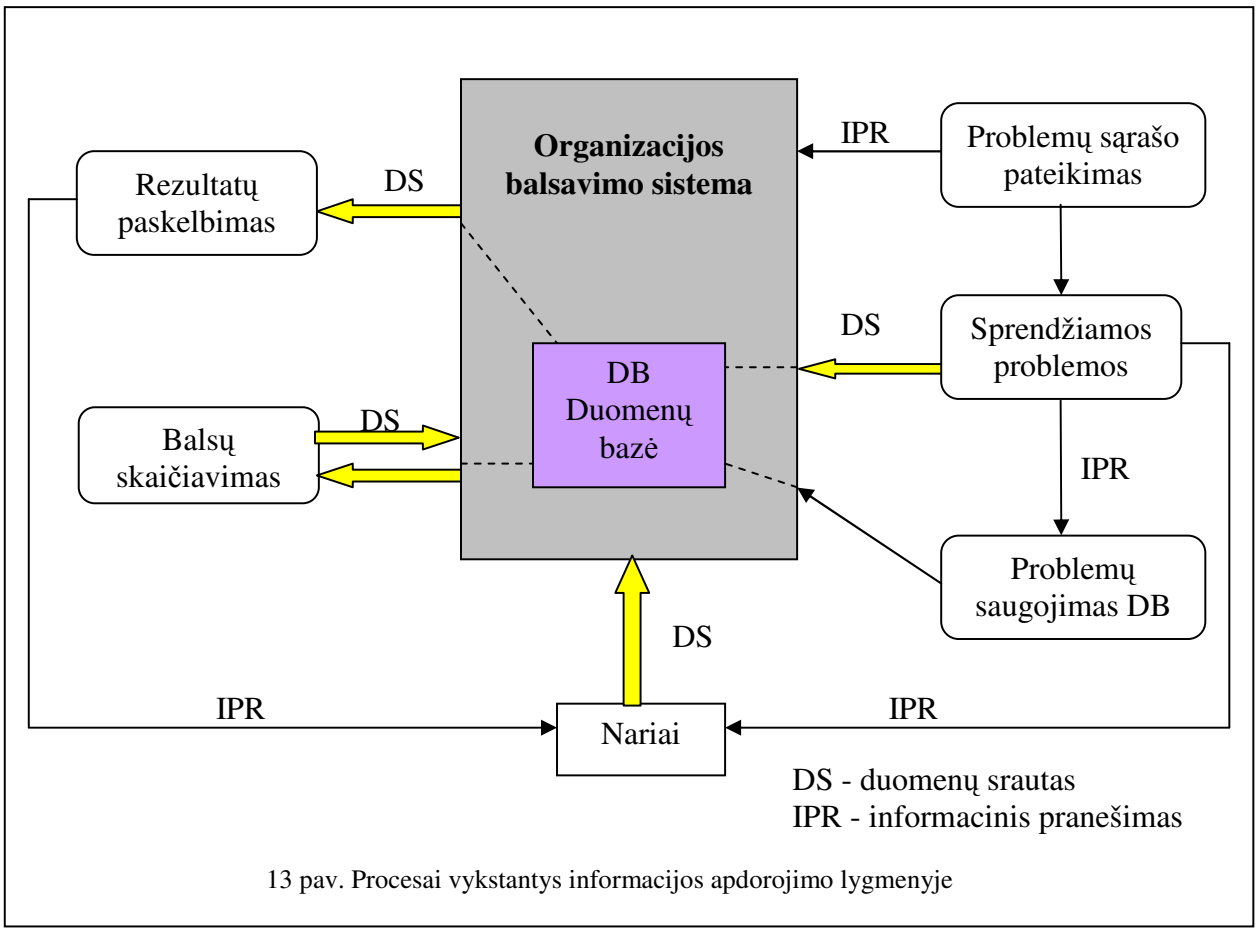
2.3.Informacijos apdorojimo lygmuo

Informacijos apdorojimo arba perdavimo lygmuo yra vienas iš svarbiausių balsavimo ciklo lygmuo. Apskritai galima teigti, kad tai vienas iš svarbiausių integruotos verslo sistemos lygmenų. Šis sluoksnis atlieka tris pagrindines funkcijas „Balsavimo sistemos“ funkcionavimo palaikymui. Tai gaunamos arba turimos informacijos apdorojimas arba tiesiog perdavimas. Vykdomos funkcijos, kurios reikalingos kito lygmens procesams.



12 pav. Informacijos apdorojimo lygmens išskaidymas

Pagrindiniai veiksmai ir procesai, kurie vyksta šiame lygmenyje iliustruojami 13 paveiksle. Žinoma šis iliustravimas nėra formalus, tačiau tai padeda geriau įsivaizduoti visą balsavimo eigą.



13 pav. Procesai vykstantys informacijos apdorojimo lygmenyje

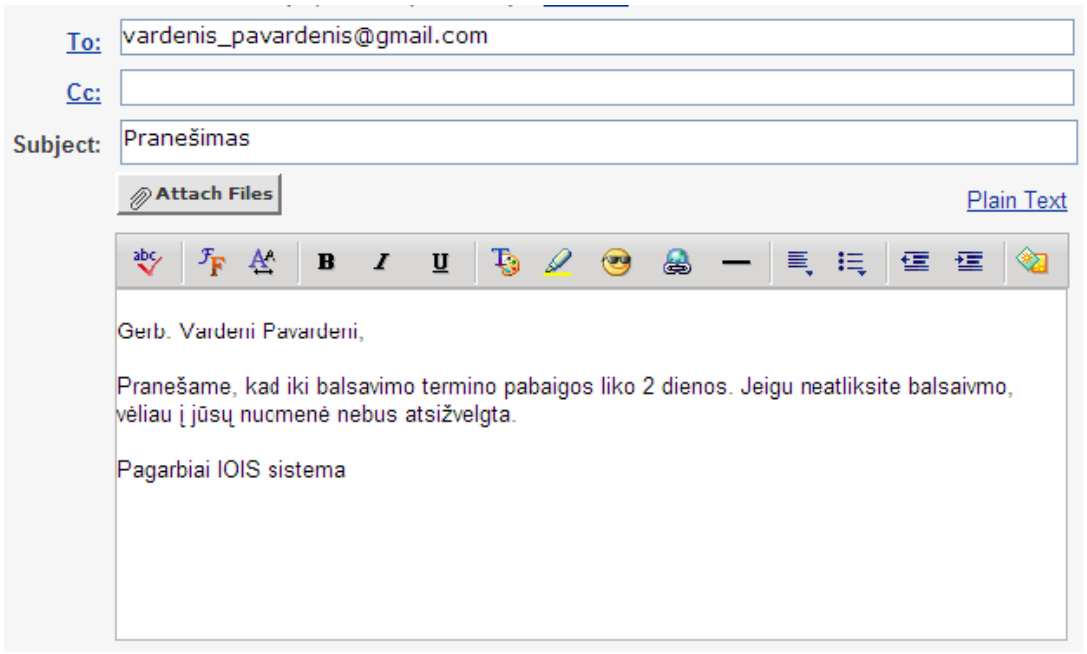
2.3.1. Informacijos apdorojimas

Balsavimo ciklo sistemos pagrindinės funkcijos yra: surinkti balsus iš narių ir juos apdoroti. Pradekime nuo pradžių. Šiame lygmenyje atliekama daug apdorojimo veiksmų tam, kad pagerinti ir padidinti sistemos našumą. Pirmiausiai kai sistema iš išorės gauna problemų sąrašą reikia jį apdoroti. Tiksliau tariant reikia peržvelgti ir patikrinti ar yra organizacijai aktualių problemų. Jei taip tai suformuluojamas konkretus problemos apibūdinimas. Visa tai gali padaryti sistemos administratorius ar aukštas pareigas užimantis darbuotojas. Čia jau pradeda veikti informacijos apdorojimo/perdavimo lygmens funkcijos. Suformuotos problemos, informacijos perdavimo kanalais siunčiamos į duomenų bazę, kad būtų išsaugotos. Vėliau nariai informuojami el.laiško pranešimais apie balsavimo ciklo pradžią arba pabaigą. Tai atlieka informacijos apdorojimo funkcijos. Jos aprašytos

sekančiame skyriuje. Tuomet, kai organizacijos nariai atlieka balsavimą, jų duomenys apdorojami. Toks procesas kiekvieno narių balsus sumuoja. Žinodami balsų skaičių, jį reikia saugiau išsaugoti duomenų bazėje (DB). SQL užklausų pagalba rezultatai saugomi duomenų bazėje. Vėliau norint atlikti patikrinimą, galima SQL užklausų pagalba šiuos rezultatus gauti ir juos patikrinti.

Funkcijų pagalba informacijos apdorojimo/perdavimo lygmenyje, formuojami informaciniai pranešimai nariams.

Informacinis pranešimas tai el.laiškas su tekstu. Šie el.laiškai siunčiami į organizacijos nario pašto dėžutę. Formuojamas el.laiško tekstas ir nurodytu adresu, kuris yra saugomas DB, siunčiamas informacinis laiškas. Vieno laiško pavyzdį pateikiu 14 paveiksle.

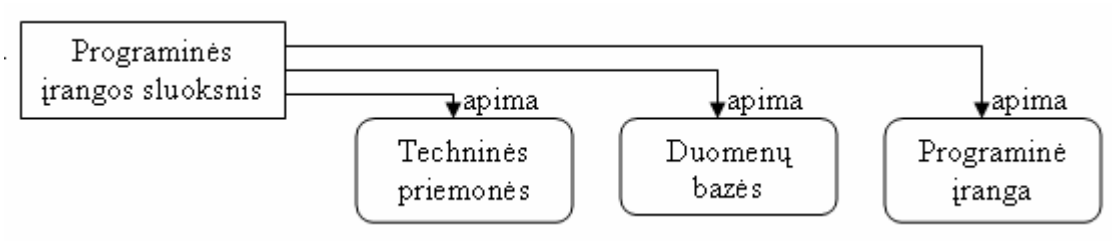


14 pav. El.laiško pavyzdys

2.4.Informacinių technologijų lygmuo

Tai programinės įrangos lygmuo. Galima drąsiai teigti, kad tai pats žemiausias ir techniškai naudingiausias sistemos lygmuo. Žodis programinės įrangos arba technologijų lygmuo duoda suprasti, kad šiame sluoksnyje talpinamos programų sistemos arba programiniai paketai, duomenų bazės, tam tikros platformos ir t.t. Visai tai susiję su techninėmis galimybėmis. Atsižvelgiant į organizacijos sprendimo priėmimo sistemos

pavyzdį, ištirsiu kokie komponentai reikalingi tam, kad sistema puikiai funkcionuotų. Paveikslas (15 pav.) vaizduoja kokias sudedamąsias dalis turi turėti paskutinysis verslo sistemos lygmuo.



15 pav. Programinės įrangos lygmuo

2.4.1. Techniniai reikalavimai

Techninės priemonės informacinių technologijų lygmenyje apima visą aparatūrinę dalį. Tai gali būti serverio aparatūrinė specifikacija, sudedamosios dalys, serverio darbo charakteristikos. Verslo sistema be programinės ir techninės įrangos būtų neautomatizuota. Tai reiškia, kad tokios verslo sistemos negalima vadinti „integruota“. Nustatysime minimalias ribas, kurios turi tenkinti minimalius balsavimo sistemos techninius reikalavimus. Žinoma, šie reikalavimai yra teorinio pobūdžio ir gali kisti priklausomai nuo integruotos verslo sistemos našumo.

Reikalavimai sistemai

Tam, kad būtų galima efektyviai dirbti su programine įranga, apibrėžiami minimalūs techniniai reikalavimai:

Minimalūs reikalavimai kliento sistemai:

- ✓ 486/66 Mhz procesorius
- ✓ 12 Mb RAM;
- ✓ 10 Mb laisvos vietos kietame diske;
- ✓ Turi būti įdiegta interneto naršyklė;

Minimalūs reikalavimai serverio sistemai:

- ✓ 1733 Mhz procesorius
- ✓ 1024 Mb RAM;
- ✓ 2 Gb laisvos vietos kietame diske;
- ✓ Turi būti įdiegta operacinė sistema Windows NT arba Windows Server ;

Nagrinėjama integruota verslo sistemos techninė įranga naudoja šiuos serverio resursus:

- ✓ vidinę atmintį;
- ✓ išorinę atmintį ;
- ✓ procesorių;

2.4.2. Duomenų bazės panaudojimas

Sprendimo priėmimo ciklo atveju, galima naudoti duomenų bazę (DB), kuri yra reikalingas techninis komponentas. DB reikalinga tam, kad joje saugotume duomenis apie organizacijos narius. DB naudojamos 3 lentelės. Žinoma tai konceptuali duomenų bazės schema. Jos tikslas – geriau iliustruoti visos integruotos verslo sistemos veikimą.

Lentelė pavadinimu *Nariai*. Šios lentelės laukus galėtų sudaryti *Nario_ID*, *vardas*, *pavardė*, *elektroninio pašto adresas*, *pareigos*. Elektroninio pašto adresas yra reikalingas atributas, kadangi el.laiškų pagalba organizacijos nariai yra informuojami apie balsavimą. Žinoma el.laišku siunčiami pranėšimai ir skelbiami rezultatai.

Lentelių pavyzdžiai pavaizduoti paveiksluose (16 pav., 17 pav., 18 pav.). Taip pat aprašomi laukelių specifikacijos ir apribojimai.

Nariai : Table

Field Name	Data Type	
Nario_ID	Number	Nario ID numeris
Vardas	Text	Organizacijos nario vardas
Pavarde	Text	Organizacijos nario pavardė
Pareigos	Text	Organizacijos nario užimamos pareigos
El_pastas	Text	El.pašto adresas

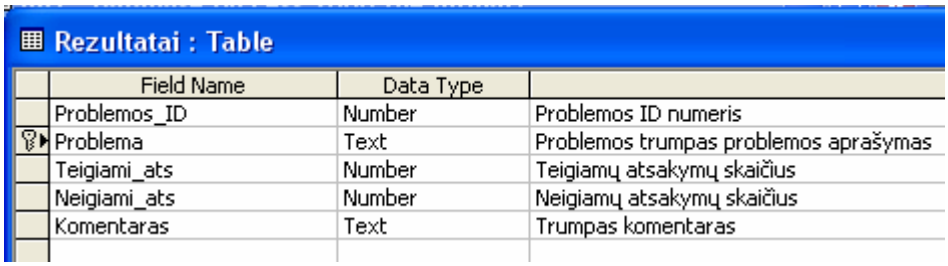
16 pav. Lentelė „Nariai“

Lentelės laukų specifikacija:

1 lentelė

Laukelio pavadinimas	Duomenų tipas	Laukelio dydis
Nario_ID	Skaičius	Sveikasis skaičius
Vardas	Tekstas	20 simbolių
Pavarde	Tekstas	30 simbolių
Pareigos	Tekstas	40 simbolių
El_pastas	Tekstas	40 simbolių su ženklu @

Sekanti lentelė turėtų būti pavadinta *Rezultatai*. Šio atributo laukų pavadinimai: *Problemos_ID*, *Teigiami_atsakymai*, *Neigiami_atsakymai*, *Komentaras*. Laukelis *Problemos_ID* reikalingas tam, kad galėtume identifikuoti problemą esančia kitoje lentelėje. Laukeliai pavadinimais *Teigiami_atsakymai* ir *Neigiami_atsakymai* savyje saugo balsavimo rezultatus. *Komentaro* laukelis saugos narių paliktus komentarus.



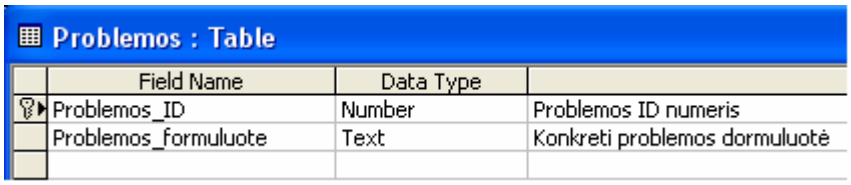
17 pav. Lentelė „Rezultatai“

Lentelės laukų specifikacija:

2 lentelė

Laukelio pavadinimas	Duomenų tipas	Laukelio dydis
Problemos_ID	Skaičius	Sveikasis skaičius
Problema	Tekstas	50 simbolių
Teigiami_ats	Skaičius	Sveikasis skaičius
Neigiami_ats	Skaičius	Sveikasis skaičius
Komentaras	Tekstas	100 simbolių

Kita lentelė galėtų būti diskusijų ciklo problemoms saugoti. Lentelės pavadinimas turėtų būti *Problemos*. Vieno iš atributo laukų pavadinimas *Problemos_formuluotė*. Šiame lauke būtų saugoma konkreti problemos formuluotė.



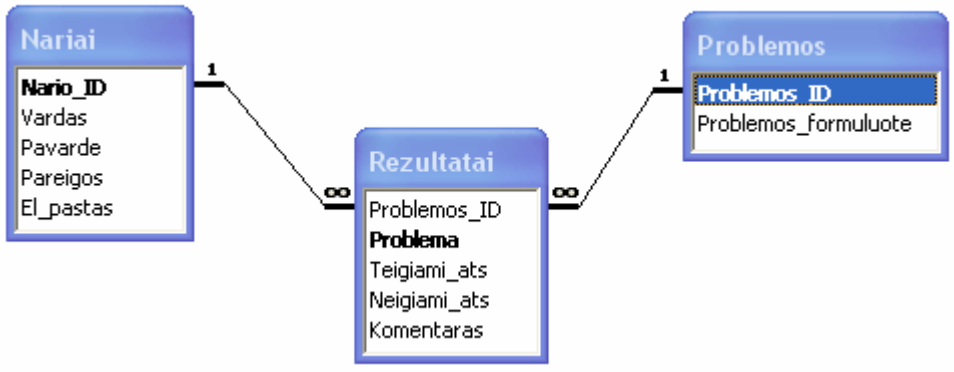
18 pav. Lentelė Problemos

Lentelės laukų specifikacija:

3 lentelė

Laukelio pavadinimas	**Duomenų tipas**	**Laukelio dydis**
Problemos_ID	Skaičius	Sveikasis skaičius
Problemos_formuluote	Tekstas	100 simbolių

Visą duomenų bazės schemą galima pavaizduoti taip (19 pav.). Kaip matome tris duomenų bazės lenteles sieja ryšiai. Lentelė *Nariai* susieta su lentele *Rezultatai* ryšiu „vienas-daug“. Žinoma, tokiu pačiu ryšiu susietos lentelės *Rezultatai* ir *Problemos*.

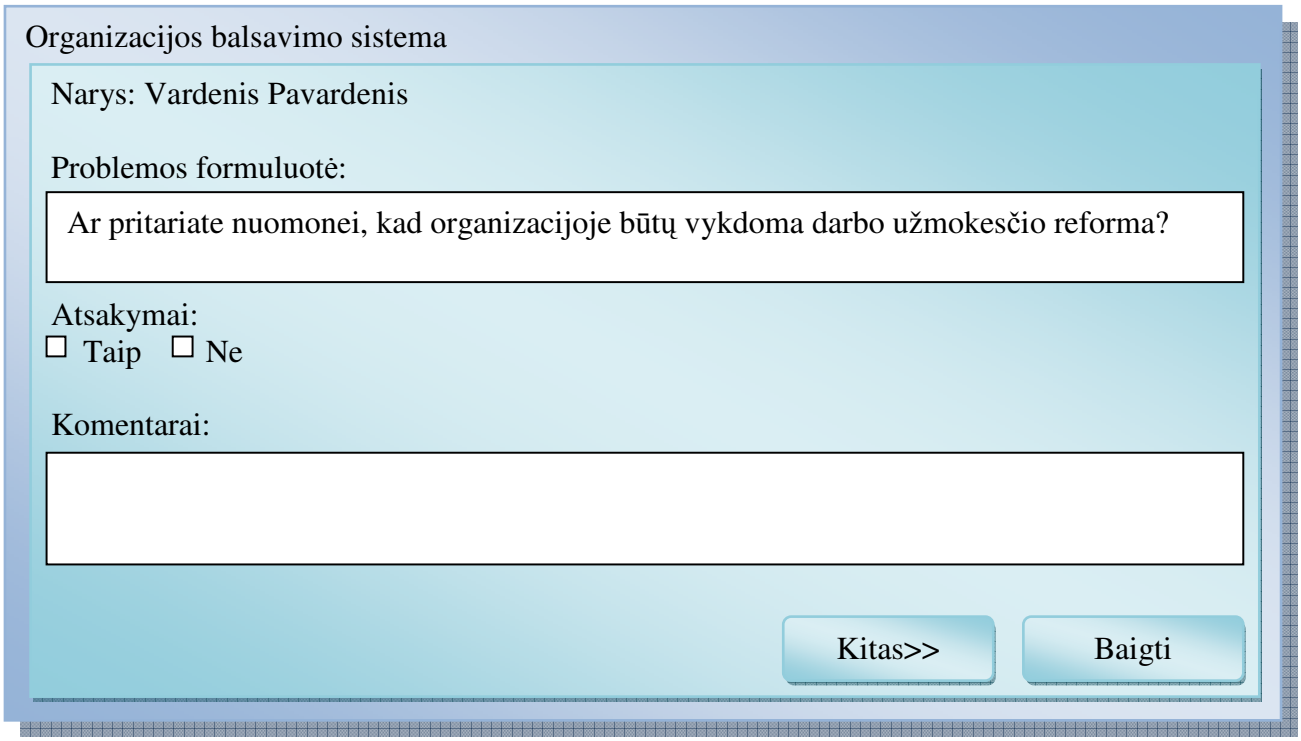


19 pav. Duomenų bazės schema

2.4.3. Programinė įranga

Sakykime organizacijos nariai interneto pagalba arba vietiniu kompiuteriniu tinklu prisijungti prie integruotos verslo sistemos. Tam, kad sėkmingai atlikti prisijungimą, vartotojui reikalinga patogi grafinė sąsaja, kad galėtų atlikti savo darbą. Sėkmingai atlikus veiksmus ir prisijungus, tariama vartotojo sąsaja turėtų atrodyti taip (20 pav.).

Programinės įrangos patikimumą galima nusakyti klaidų skaičiumi tam tikros apimties programoje, klaidingų programų elementų (komandų) dalimi joje, taip pat esamų klaidų įtaka gaunamiems rezultatams (įtakos jų panaudojimo rezultatams laipsniu). Tą patikimumą galima matuoti tikimybe, kad programa veiks norimą laiką be sutrikimų ir duos teisingų rezultatų, kad bus ne daugiau negu tam tikras skaičius (ar procentas) klaidų ir pan. Taigi programinės įrangos patikimumas - tai ne tik pačios programos, bet ir jos naudojimo pagal paskirtį bei naudojimo sąlygų savybė.



20 pav. Vartotojo darbo langas

2.5. Komponentinis sistemos specifikavimas

Komponentų diagrama – aprašo sistemoje naudojamus komponentus. Komponentas yra programinės įrangos modulis, kuris turis šias pagrindines savybes:

- ✓ Komponentas yra nepriklausoma ir pakeičiama sistemos dalis, atliekanti aiškiai apibrėžta funkciją;
- ✓ Komponentas bendrauja su kitais komponentais per apibrėžtas sąsajas;
- ✓ Komponentas veikia gerai apibrėžtos architektūros kontekste;

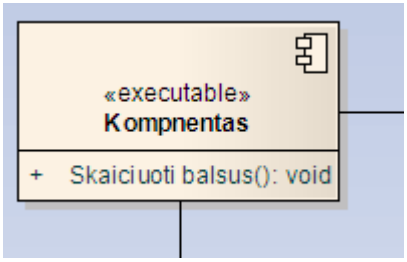
Aukščiausio lygio sistemos architektūra parodo:

- ✓ Kokie bus pagrindiniai sistemos komponentai;
- ✓ Kaip komponentai naudos vienas kitą;
- ✓ Kokiuose techninės įrangos elementuose komponentai bus įdiegti;
- ✓ Kokiais protokolais vyks bendravimą.

Sakykime, kad turėdami verslo proceso modelį, mes norime jį formalizuoti. Siekiant, kad verslo modeliavimo, projektavimo, analizavimo procesas būtų suprantamas tiek informacinių technologijų specialistams, tiek verslo dalyviams, vis dažniau naudojamos grafinio modeliavimo kalbos. ir t.t. Tačiau nėra, tokios vieningos grafinio modeliavimo kalbos, kurios pagalba galėtume specifikuoti visą sistemą, visuose trijuose lygmenyse. To pasekoje turime naudoti kelias modeliavimo kalbas. Kadangi atvaizduoti verslo procesą visuose jo kompiuterizavimo etapuose vienos kalbos nepakanka, buvo pasirinktos dvi veiklos procesų modeliavimo kalbos: UML 2.0 ir BPMN.

Norėdamas formalizuoti verslo proceso modelį pasirinkau BPMN (Business process modelling notation) grafinio modeliavimo kalbą. Šia formalizavimo kalba galima puikiai aprašyti verslo procesus. Tačiau turėtume nepamiršti, kad organizacijos diskusijų ciklo informacinę sistemą turime formalizuoti ir likusiuose dviejuose lygmenyse (informacijos apdorojimo/perdavimo, techninės įrangos). Kaip jau pastebėjote, integruotos verslo sistemos pirmajam lygmeniui modeliuoti geriausiai tinka BPMN modeliavimo kalba, likusiems dviems lygmenims geriausiai būtų naudoti UML 2.0.

Tačiau BPMN ir UML 2.0 buvo pasirinktos ne atsitiktinai. Kaip žinia UML 2.0 modeliavimo standartu galima aprašyti ir parodyti integruotos verslo sistemos komponentų architektūrą. Taip pat kaip šie komponentai bendrauja tarpusavyje ir kaip tai atspindi visuose trijuose IVS lygmenyse. Turėčiau pabrėžti, kad komponentą turėtume įsivaizduoti kaip tam tikrą „juodąją dėžę“, kurios vidus ir visi procesai vykstantys viduje, nėra matomi iš išorės. T.y. vartotojas nemato, kokios sudėtinės dalys sudaro komponentą. Trumpai tariant komponentą sudaro metodai, procesai ar funkcijos esančios jo viduje. Šie metodai, procesai ar funkcijos tarpusavyje yra glaudžiai susiję, todėl jie pajungiami į vieną grupę – komponentą. Žinodami komponento sąvoką galima apibūdinti bendrąsias jo funkcijas IVS sistemoje.

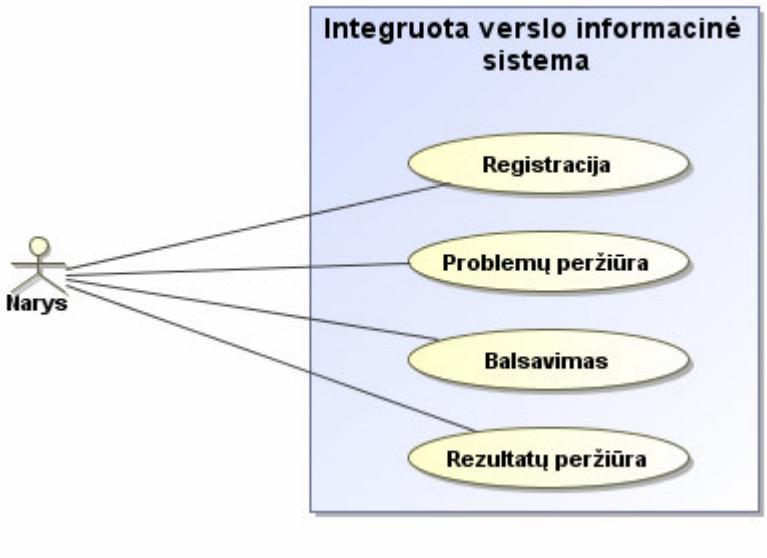


21 pav. Komponento modelis UML sistemoje

Kaip ir matome iš paveikslėlio, komponentas tarsi sisteminis vienetas, kuris turi vieną arba kelis įėjimus. Taip pat jis turi vieną ar net kelis išėjimus. Vadinasi komponentas gavęs įėjimo signalą arba informaciją, ją apdoroja ir pateikia galutinį rezultatą. Žinoma, šis rezultatas keliauja į kitą komponentą, kur bus apdorojamas arba pateikiamas galutinis rezultatas.

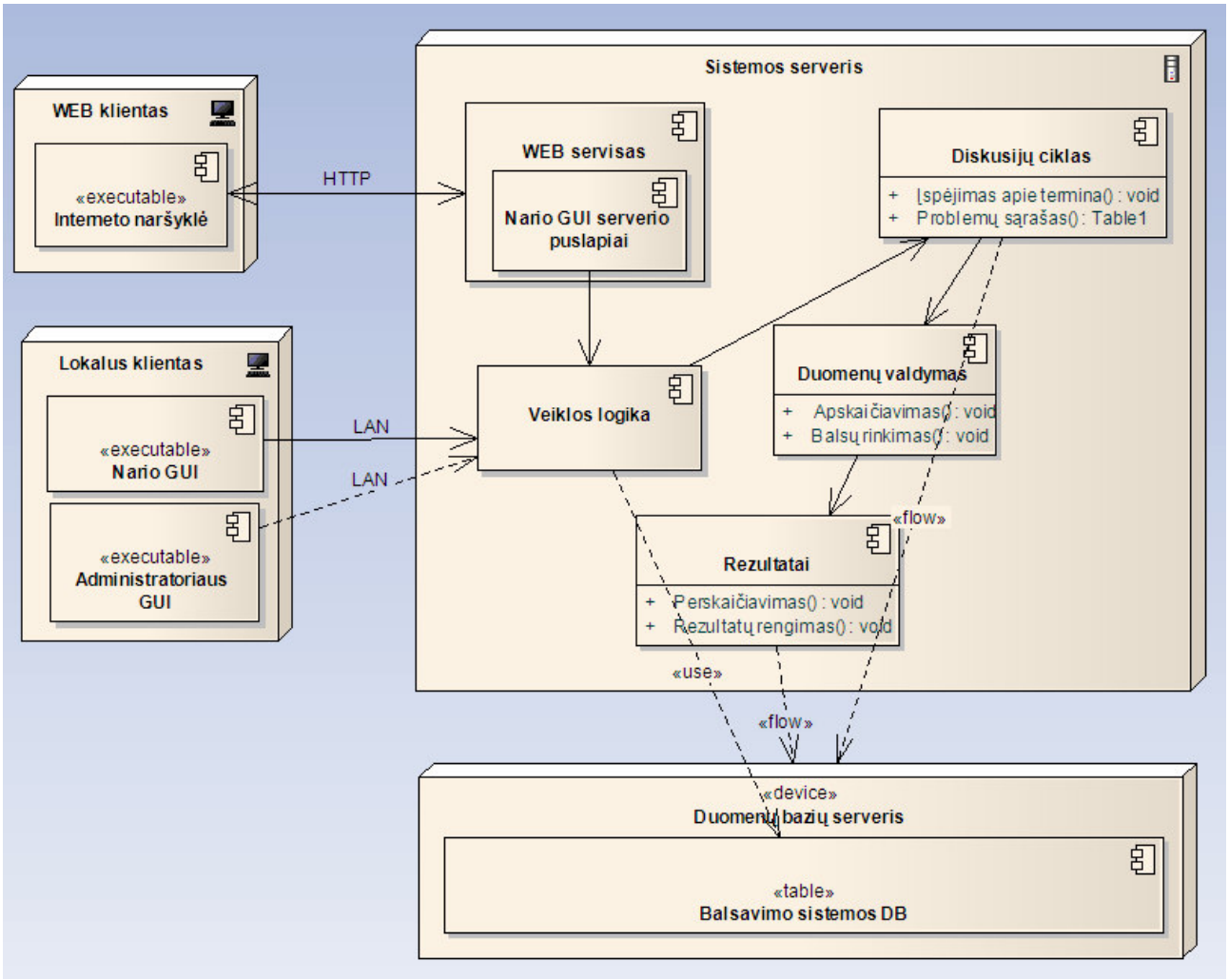
Komponentai dažniausiai skirstomi į dvi pagrindines grupes. Primityvūs ir sudėtiniai. Savo ruožtu, turėdami visą IVS struktūrą paremtą komponentine abstrakcija, galime komponente išskirti vidinius įvykius arba komponentą įsivaizduoti kaip agregatą turintį vidinius įvykius. Taigi žinodami kokie agregatai sudaro visą integruotą verslo sistemą, galima lengvai formalizuoti IVS, pasinaudojant PLA (Piece-Linear aggregates) specifikavimo kalba.

Pirmiausiai pateiksiu nario veiksmų diagramą.



22 pav. Nario pasirinkčių diagrama

Tam, kad geriau įsivaizduotume konceptualią verslo sistemos komponentinę architektūrą, ją pateiksiu 23 paveiksle. Ši komponentinė architektūra yra konceptualaus pobūdžio. Detali komponentinė išraiška pateikta 25 paveiksle.



23 pav. Sistemos konceptuali architektūra

Darbe nagrinėjama imitacinių modelių panaudojimo spendimams priimti integruotose verslo sistemose realiu laiku koncepcija, kuri apima tiek verslo procesų imitacinių modelių sudarymą, tiek jų integravimą į verslo sistemas. Verslo procesų imitacinių modelių sudarymas yra siejamas su verslo procesų modeliais, kurie yra sudaromi projektuojant integruotas verslo sistemas, bei su atkarpomis tiesinių agregatų metodo taikymu, šiems procesams formalizuoti. Nagrinėjama sprendimo priėmimo sistemos, veikiančios realiu laiku, architektūra. Darbe pateikiamas verslo procesams formalizuoti naudojamas atkarpomis tiesinių agregatų (PLA) formalizmas, kuris leidžia kurti šiuo metodu specifikuotų sistemų imitacinius modelius. Parodyta, kad tarp verslo procesų modeliavimo ir agregatinio formalizavimo sąvokų yra funkcinis sąryšis. Tai teoriškai pagrindžia PLA metodo taikymo verslo procesams formalizuoti galimybes.

Jau egzistuoja metodika, leidžianti verslo procesus, aprašytus BPMN notacijoje, transformuoti į verslo procesų formalias agregatines specifikacijas (PLA). Jau įmanoma BPMN modeliavimo kalba aprašytą verslo procesą transformuoti į PLA modelį. Tai atliekama pasitelkus papildomas priemones.

Kaip žinia komponentinė paradigma leidžia greičiau sukurti programų sistemas efektyviau pakartotinai naudojant gatavus programinius artefaktus – komponentus. Be to, komponentinės verslo sistemos yra ypač lengvai testuojamos ir modifikuojamos. Deja, praktiškai kuriant komponentines verslo sistemas daug laiko sugaištama komponentų paieškai ir verifikavimui. Vienas galimų šios problemos sprendimų – komponentinių programų specifikavimui, sintezei ir verifikavimui taikyti formaliuosius metodus.

Galimos komponentų abstrakcijos:

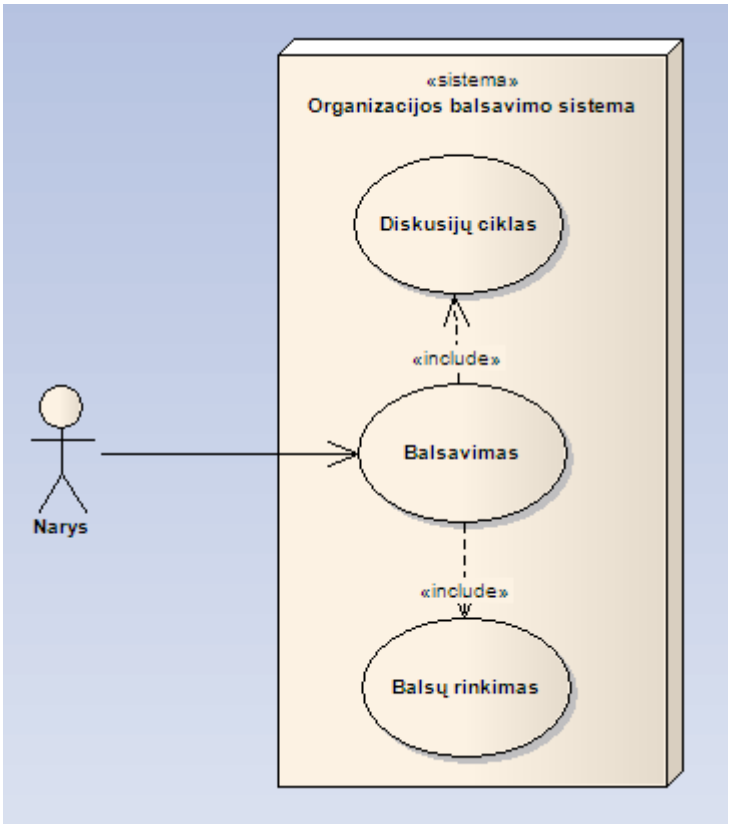
- ✓ *Funkcinės abstrakcijos*
  - o komponentas realizuoja vieną funkciją, sakykime matematinę funkciją
- ✓ *Atsitiktinis grupavimas*
  - o komponentas – laisvai susijusių esybių aibė, kurios gali būti duomenų paskelbimas, funkcijos ir pan.
- ✓ *Duomenų abstrakcijos*
  - o komponentas atvaizduoja duomenų abstrakciją arba klasę objektiniam programavime.
- ✓ *Grupės abstrakcijos*
  - o komponentas yra grupė susijusių klasių, kurios dirba kartu.
- ✓ *Sistemos abstrakcijos*
  - o komponentas yra visiškai nepriklausoma sistema

2.5.1 Komponentinio kūrimo procesas

Šiame skyriuje aprašomas organizacijos balsavimo sistemos tyrinėjimas komponentiniu metodu. Sekančiame paveikslėlyje (24 pav.) matome nario atliekamų veiksmų diagramą.

- ✓ Komponentinis kūrimas gali būti integruotas į standartinį programinės įrangos kūrimo procesą, įtraukiant pakartotinio panaudojimo veiklą į procesą;
- ✓ Visgi pakartotiniu panaudojimu pagrįstam kūrime, sistemos reikalavimai yra modifikuojami, kad atspindėtų prieinamus komponentus;

- ✓ Komponentinio kūrimo procesas paprastai naudoja prototipus arba palaipsninį kūrimą.



24 pav. Nario veiksmų diagrama

Atsiradus verslo modeliavimo standartizavimo poreikiui, sukurta daug įvairių modeliavimo standartų, kalbų, kurios akcentuoja skirtingas verslo modelių savybes ir nėra nė vienos, kuri tenkintų visus verslo modeliavimo poreikius. Be to, daugelis kalbų yra tekstinio pavidalo ir verslo ekspertams neprieinamos. Nuolat keičiantis reikalavimams, modeliavimo metodika turi prie jų prisitaikyti. Viena iš plačiausiai naudojamų grafinio modeliavimo kalbų yra UML (angl., *Unified Modelling Language*), kuri nuolat tobulinama. Naujausia UML versija yra 2.0, kuri turi išplėstas galimybės veiklai modeliuoti.

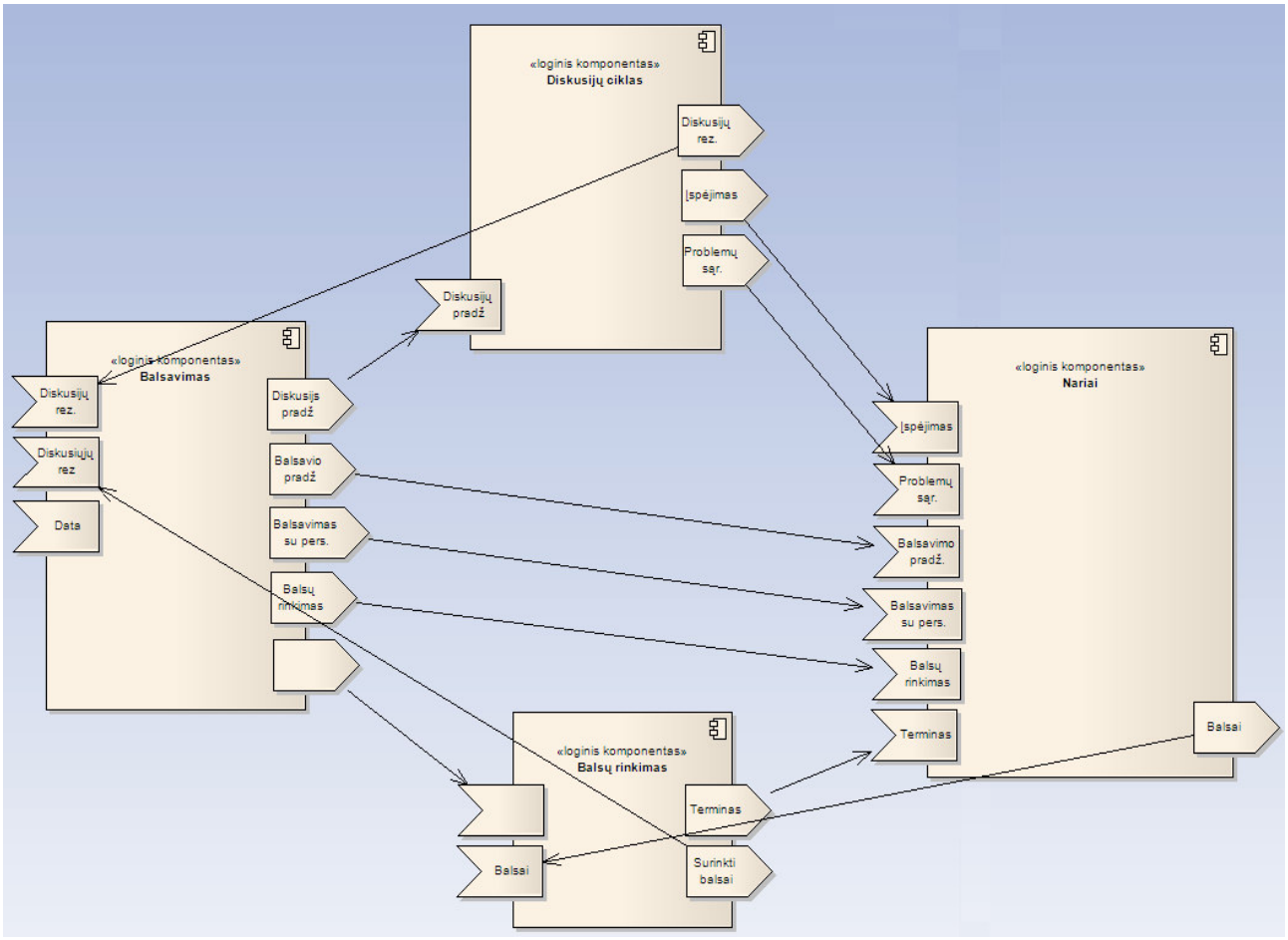
Plačias verslo modeliavimo galimybes palaiko naujas verslo modeliavimo standartas BPMN (angl., *Business process modelling notation*), kuris palengviną projektavimo kelią nuo projektavimo iki realizavimo, ir projektavimo procesą padaro prieinamą ir aiškų visiems verslo kūrimo dalyviams.

UML 2.0 versijos veiklos diagramos savo galimybėmis tolygios BPMN, tačiau neturi tokios išbaigtos stereotipų aibės. Tačiau UML turi tų privalumų, kad ji leidžia

modeliuoti ne tik verslo procesus, bet ir duomenis bei paslaugas, kurios reikalingos verslo procesams įgyvendinti. Šiame darbe yra siūloma naudoti UML 2.0.

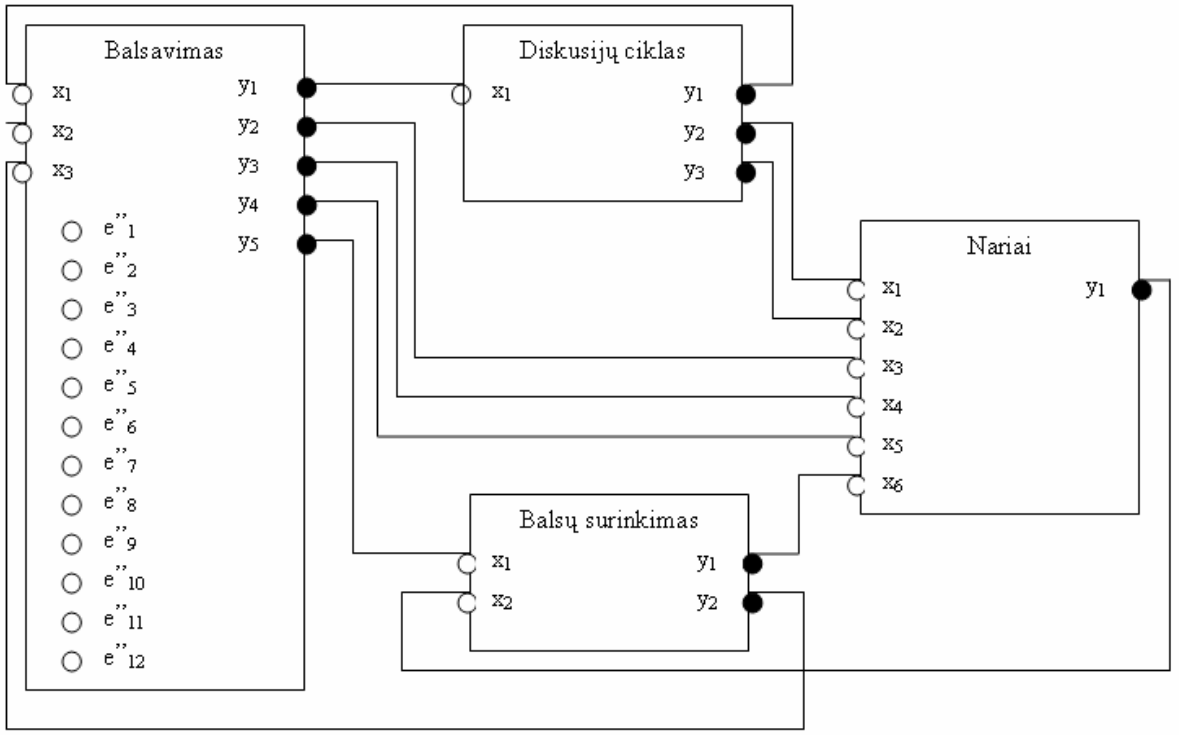
BPMN siūloma plati stereotipų aibė supaprastina modeliavimo procesą, padaro jį aiškesnį ir lengvesnį.

Darbe trumpai analizuojamos UML 2.0 ir BPMN. Pabrėšiu, kad visas konceptualus „Balsavimo sistemos“ modeliavimas atliktas įrankiu Magic Draw 15 ir Enterprise Architect 7.1. Šių programų pagalba galima lengvai pavaizduoti balsavimo sistemos atskirus komponentus ir kaip jie susiję vienas su kitu. Pateikiu detalią bendrą sistemos komponentinę architektūrą (25 pav.). Tačiau matome, kad vidiniai komponentų įvykiai nėra išskirti. Tai atliksime vėlesnėje tyrimo eigoje.



25 pav. Integruotos balsavimo sistemos komponentai

Remdamasis sudaryta „Balsavimo sistemos“ konceptualia komponentine architektūra, sudarysiu agregatinį balsavimo proceso modelį. Jį sudaro keturi agregatai, kurie ir vaizduoja visus komponentus: Balsavimas, Diskusijų ciklas, Balsų rinkimas, Nariai.



26 pav. Balsavimo proceso agregatinis modelis

Kaip matome iš 26 pav. Vidiniai įvykiai pavaizduoti tik agregatui „Balsavimas“, nes tik šis agregatas bus specifikuojamas. Formali agregato „Balsavimas“ specifikacija pateikta 1 priede.

2.5.1. Kalbos, naudojamos verslo procesams aprašyti

UML

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

Veiklos diagramos yra trijų lygių:

- ✓ Pirmo lygio diagrama abstrakčiai vaizduoja veiklą.
- ✓ Antro lygio diagramos gali būti dviejų tipų : tarpinės veiklos (angl., *Intermediate Activities*) diagramos aprašo veiklos grafus, veiklos valdymo bei duomenų srautus; struktūrinės veiklos (angl., *Structured Activities*) leidžia aprašyti darinius, panašius į naudojamus programose;
- ✓ Trečio lygio diagramose aprašomos tokios konstrukcijos kaip lanko svoris, srautų suskaidymas ir t.t

Veiklos procesų modeliavimo notacija BPMN

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

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

- ✓ *Vidinių verslo procesų diagramos* (angl., *Private business processes*). Tai vidinis įmonės darbų srautų modelis. Jeigu projektuojamos vidinės verslo diagramos naudojant juostas, tai vidinė diagrama negali išeiti už jos ribų. Jeigu dvi juostos sujungtos, tai reiškia, kad ne jų atskiri elementai, o vidinės diagramos siejasi viena su kita.
- ✓ *Sąveikų procesų diagramos* (angl., *Abstract processe*s).. Jos rodo, kaip sąveikauja skirtingi verslo procesai. Į diagramas įtraukiami tik tie procesai kurie veikia vidinių verslo procesų išorėje.
- ✓ *Bendradarbiavimo proceso diagramos* (angl., *Collaboration processes*). Šios diagramos parodo globalius procesus. Šios diagramos gali būti vaizduojamos be juostų. Veiksmai priklausantys tam pačiam vidiniam procesui šiose diagramose gali būti susieti.

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

- ✓ Aukšto lygio vidinių procesų veiklos.

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

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

2.6. Verslo procesų formalizavimas, naudojant PLA modelį

Atkarpomis tiesinių agregatų (PLA) formalizavimas, naudojamas kuriant dinaminius verslo procesų modelius. Šio formalaus metodo ypatumai:

- ✓ Galima paruošti analizuojamos sistemos formalius aprašymus turint tik viena reikšme.
- ✓ Modeliavimo ypatybės gali būti analizuojamos naudojant matematinės įrangos technika.
- ✓ Formalus aprašymo metodas yra teorinis pagrindas, kuriant programinės įrangos priemones formalių specifikacijų kompiuteriniam analizavimui (tikrinimui, kontrolei, modeliavimui).

Atkarpomis tiesiniai agregatai (PLA) yra automatinio modelio atskiras atvejis. Šiuo metodu sistemos funkcionavimas formalizuojamas taip:

1. Įėjimų signalų aibė X – čia kiekvienam agregato įėjimui yra priskiriama konkreti perduodamo signalo duomenų struktūra.
2. Išėjimų signalų aibė Y – čia kiekvienam agregato išėjimui yra priskiriama konkreti perduodamo signalo duomenų struktūra.

3. Išorinių įvykių aibė $E^{'}$ – įvykiai, įvykę agregato išorėje, bet veikiantys agregato būseną.
4. Vidinių įvykių aibė $E^{''}$ – įvykiai, įvykę agregato viduje ir keičiantys agregato būseną.
5. Valdymo seka $e_i^{''}\ \alpha\ \{\xi_j^i\}, j=\overline{1,\infty}.$
6. Diskrečioji būsenos dedamoji $v(t_m)$ – aibė agregato elementų būsenų reikšmių, galinčių įgyti diskretines reikšmes.
7. Tolydžioji būsenos dedamoji $z_v(t_m)$ – aibė agregato elementų, galinčių įgyti tolydines reikšmes.
8. Pradinė būsena $v(t_0), z_v(t_m)$ – diskretinių ir tolydinių būsenos vektoriaus komponenčių pradinės reikšmės.
9. Perėjimo ir išėjimo operatoriai: $H(e_i), G(e_i)$ – algoritmai (sakinių sekos), pagal kuriuos keičiama agregato būsena ir išduodami signalai į agregato išorę.

Agregato būsena gali pasikeisti tik dviem atvejais: kai įėjimo signalas x_i patenka į agregatą ar kai tolydusis komponentas įgyja nustatytą reikšmę. Agregato funkcionavimas nagrinėjamas diskrečiais laiko momentais, kurie priklauso aibei $T=\{t_0,t_1,...,t_m,...\}$. Tais laiko momentais gali įvykti vienas ar keli įvykiai, kurie sukelia agregato būsenos pasikeitimą. Agregato įvykių aibę $E=E^{'}\cup E^{''}$ sudaro įvykiai $E^{'}=\{e_1^{'},e_2^{'},...,e_N^{'}\}$, kurie įvyksta dėl įėjimo signalų atėjimo, ir įvykiai $E^{''}=\{e_1^{''},e_2^{''},...,e_N^{''}\}$, kurie fiksuoja operacijų pabaigą. Kiekvienam vidiniam įvykiui priskiriama valdymo seka: $e_i^{''}\ \alpha\ \{\xi_j^i\}, j=\overline{1,\infty}$.

Tolydžiosios dedamosios koordinatės $z_v(t_m)$ apibrėžia laiko momentus, kada agregate galimi įvykiai, čia $z_v(t_m)=\{w(e_1^{''},t_m),w(e_2^{''},t_m),...,w(e_f^{''},t_m)\}$. Be to, visada $w(e_i^{''},t_m)\ge t_m$, kur $w(e_i^{''},t_m)-$operacijos pabaigos momentas. Agregato būsena $z(t_m)$ gali pasikeisti tik diskrečiais laiko momentais $t_m, m=1,2,3,...,$ likdama pastovi laiko tarpuose $[t_m,t_{m-1}), m=0,1,2,...,$ čia t_0 - pradinis sistemos funkcionavimo momentas. Kai sistemos būsena yra žinoma $z(t_m), m=0,1,2,...,$ įėjimo signalo atvykimas į agregatą arba lygybė $t_{m+1}=\min\{w(e_i^{''},t_m)\}, 1\le i\le f$ nustato laiko momentą t_{m+1}, kai įvyksta kitas įvykis. Sekančio įvykio e_{m+1} rūšis yra nustatoma įėjimo signalo, jei įėjimo signalas patenka į agregatą

laiko momentu t_{m+1}, arba yra nustatomas valdymo koordinatės, turinčios minimalią reikšmę laiko momentu t_m, t. y. koordinatė $w(e_i^{"}, t_m)$ tampa minimalia, $e_{m+1} \in E^{"}$.

Operatorius H apibrėžia naują agregato būseną: $z(t_{m+1}) = H[z(t_m, e_i)], e_i \in E^{'} \cup E^{"}$.

Operatorius G apibrėžia išėjimo signalus: $y = G[z(t_m, e_i)], e_i \in E^{'} \cup E^{"}, y \in Y$.

Apžvelgiant agregatų sistemą, susidedančią iš K agregatų, įskaitant ir išorės agregatus.

Agregato galimų įvykių aibė sudaro vidinių įvykių poaibio sąjunga, t. y.: $E \in \bigcup_{k=1}^{K} E_k^{"}$, kur $E_k^{"}$ k-jo agregato įvykių aibė.

Sekantis laiko momentas t_{m+1}, kai įvyksta aibės E įvykis yra apibrėžiamas:

$t_{m+1} = \min_k \min_r \{w(e_i^{"}, t_m)\}, 1 \le k \le K, e_r^{"} \in E_k^{"}$.

Agregato sistemos charakteristikos gali būti nustatytos analizuojant seką:

$(z(t_0), e_1, z(t_1)), (z(t_2), e_2, z(t_2)), (z(t_2), e_3, z(t_3)), ..., (z(t_3), e_{m+1}, z(t_{m+1}))...$

kur $z(t_m)$ - yra agregato sistemos būsena po sistemos sudarymo ir $e_i \in E$.

Diskrečiais laiko momentais t_m, kai sistema juos pakeičia, susikuria nelygybių seka:

$t_0 \le t_1 \le t_2 \le ... \le t_m \le t_{m+1} \le ...$.

2.7. Tekstinio ir grafinio specifikacijų sudarymo būdų palyginimas

Sudarant sistemos specifikacijas, jas reikia aprašyti formaliai. Vienas iš aprašymo būdų yra aprašymas formaliomis išraiškomis tekstu. Tačiau toks sistemų specifikavimo būdas nėra patogus, nes nėra vaizdus, reikalauja didesnio įsigilinimo norint kažką suprasti. Daug laiko sugaištama bandant nustatyti sistemos dalis ir jų tarpusavio ryšius bei santykius.

Siekiant palengvinti formalios agregatinės specifikacijos sudarymą ir jos peržiūrėjimą, galima naudoti grafinį specifikacijos vaizdavimo, užrašymo būdą. Šiuo atveju agregatinės specifikacijos vaizduojamos diagramomis.

4 lentelė. Grafinio specifikacijų sudarymo privalumai ir trūkumai

Privalumai	Trūkumai
Vaizdžiai matoma specifikacija Lengvai skaitoma Lengvai sudaroma Realizuoti programiniai įrankiai gali nuimti pasikartojantį, bereikalingą darbą	Diagramose negalima atvaizduoti visos informacijos, nes tada jos būtų perkrautos (pvz., nesimato įėjimo ir išėjimo funkcijų). Diagramoje nesimato pradinių būsenų (jas reikia aprašyti atskirai).

2.8. Bendrieji reikalavimai skirti integruotos verslo sistemos lygmenims aprašyti

Integruotos verslo sistemos kontekste ypač aktuali tampa programinės įrangos funkcinių ir nefunkcinių savybių atotrūkio nuo realių verslo poreikių problema. Literatūros šaltiniuose dėstomas naujas požiūris į tai, kaip, kuriant integruotas verslo sistemas, verslo reikalavimus nuleisti į programinės įrangos lygmenį. Siūloma reikalavimus visuose sistemos lygmenyse, įskaitant verslo, informacijos apdorojimo ir programinės įrangos lygmenis, formuluoti vartojant tą pačią sąvokų sistemą (ontologiją) ir visas tos sistemos sąvokas apibrėžti vadovaujantis bendrojoje sistemų inžinerijoje priimta sistemos samprata. Programinės įrangos reikalavimams gauti iš verslo reikalavimų siūloma naudoti reikalavimų lokalizavimo ir pažingsninio patikslinimo technikas.

Kuo gali būti naudinga tokių bendrų sąvokų sistema? Visų pirma, ji sudaro galimybes kalbėti apie visus IVS lygmenis, vartojant tuos pačius terminus. Pradėjus tai daryti, paaiškėja kai kurios verslo ir jį palaikančios programinės įrangos atotrūkio priežastys: pavyzdžiui, norint išvengti skirtingų lygmenų sąsajų atotrūkių, IVS programinės įrangos sąsajos privalo būti sudarytos iš atitinkamų verslo sistemos sąsajų. Vadinasi, dalykinę sritį analizuojantis analitikas turi gebėti į analizuojamą verslą pažvelgti per bendrosios sistemų inžinerijos prizmę, išsiaiškinti analizuojamos sistemos sąsajas, jas specifikuoti ir iš tokių specifikacijų sudaryti integruotos verslo sistemos ir kuriamos programinės įrangos sąsajų reikalavimus.

Be abejo, problema labai sudėtinga ir kol kas ne iki galo išnagrinėta. Vis dėlto tam tikrą sprendimą galima rasti nagrinėjant vadinamąjį verslo priklausomybių modelį. Iš šio modelio matyti, kad integruotų verslo sistemų inžinieriaus požiūriu svarbiausiomis tarpusavio priklausomybėmis yra susieti:

- ✓ verslo vizija, įskaitant tos vizijos įgyvendinimo strategiją, su verslo procesais;

- ✓ verslo procesai su įmonės organizacine struktūra ir su įmonės technologine infrastruktūra, įskaitant naudojamas programų sistemas;
- ✓ įmonės technologinės infrastruktūros naudojimo būdai su jos darbuotojų gebėjimais.

Pavyzdžiui, vadinamasis vieno langelio principas, kuris paprastai yra siejamas su viešojo sektoriaus teikiamomis paslaugomis, iš tiesų yra tipinė konstrukcija, naudojama visose integruotose verslo sistemose. Ši konstrukcija įgyvendina reikalavimą, kad kiekviena sistemos vartotojų kategorija jai reikalingas paslaugas gautų per vieną sąsają, sukonstruotą atsižvelgiant į vartotojų savybes. Kitaip tariant, viskas, ko reikia, turi būti po ranka. Vadovaujantis šiuo principu, pavyzdžiui, yra išdėstyti stabdžių ir sankabos perjungimo pedalai, kiti automobilio „sąsajos" elementai, suprojektuota kompiuterio klaviatūra ir televizijos nuotolinio valdymo pultai, taip pat daugelis programų sistemų, įskaitant visiems gerai žinomą sistemą „Microsoft Word".

3. IŠVADOS

1. Išanalizavus egzistuojančią komponentinę formalizavimo paradigmą integruotoms verslo sistemoms, pastebėta, kad komponentinių sistemų kūrimo idėjos IOIS lygmenyje nėra plačiai naudojamos. Susidaro nuomonė, kad paradigma yra taikoma tik kuriant du IOIS struktūrinius elementus: techninę įrangą ir programų sistemas.
2. Tyrimo eigoje pastebėta, kad nėra vieningo komponento apibrėžimo. To pasakoje, reikėjo suformuluoti kriterijus, kuriais buvo remtasi konceptuliai aprašant integruotos „Balsavimo“ sistemos komponentus.
3. Ištyrus organizacijos sprendimo priėmimo sistemos pavyzdį, komponentų vaizdavimui siūloma pasitelkti UML grafinę notaciją.
4. Praktiškai parodyta, kad tiesinių agregatų metodas yra tinkamas, norint formaliai specifikuoti konceptualiai pateiktą komponentą.
5. Formaliai aprašant sistemos lygmenyse naudojamus komponentus ir jų savybes, pastebėta, kad komponento specifikacijos formalizavimo būdai nepakankami arba netinkami integruotos verslo sistemos komponentui formalizuoti. Norint kurti integruotas verslo sistemas, reikia aiškiai atskirti komponento specifikaciją nuo jo turinio, apibrėžti komponento sąvoką ir gebėti jį sukonstruoti. Atsižvelgiant į tai, komponento sąvoka darbe formalizuojama, kaip „juodosios dėžės" abstrakcija. Apibrėžtosios sąvokos ir jų formalizavimo būdas įgalina specifikuoti integruotos verslo sistemos komponentus.
6. Sudarinėjant integruotos verslo sistemos formalizavimą, siūloma naudoti PLA specifikacijas, kurios leidžia aprašyti kintančias laike sistemas, nes integruota verslo sistema turi greitai adaptuotis prie nuolat kintančių veiklos sąlygų.
7. Informacinės sistemos komponentas turi būti realizuojamas, kaip savarankiškas agregatas, tam, kad palengvintų integruotos verslo sistemos formalizavimą.

4. LITERATŪROS SĄRAŠAS

[1] Bachmann F., et al. *Volume II: Technical Concepts of Component-Based Software Engineering.* Technical Report CMU/SEI-2000-TR-008 ESC-TR-2000-007. Software Engineering Institute, 2000.

[2] Bagušytė L., Lupeikienė A., Organizacijų integruotų informacinių sistemų komponentinio kūrimo problemos. Informacinės technologijos 2006. Lietuvos mokslas ir pramonė. Konferencijos pranešimų medžiaga. I tomas, ISBN 9955-09-788-4, Kaunas, Technologija, 2006, p. 183-188.

[3] Bagušytė L., Lupeikienė A., Programinio komponento ontologija ir jos sudarymas. *Informacijos mokslai.* Vilniaus universitetas. 2005, t. 34. p.119-124.

[4] Cheesman J., Daniels J., UML Components. Addison-Wesley (2001)

[5] Cummins F. A., *Enterprise Integration. An architecture for Enterprise Application and Systems Integration.* John Wiley & Sons Inc. 2002, p.58 - 87.

[6] Čaplinskas A., Lupeikienė A., Sistemų inžinerijos intelektualizavimo problemos. *Informacinės technologijos 2001, Konferencijos pranešimų medžiaga.* Kaunas. Technologija. 2001, p. 200-205.

[7] Čaplinskas A., *Programų sistemų inžinerijos pagrindai, I dalis.* Matematikos ir informatikos institutas. Vilnius. 1996.

[8] Herzum P., Sims O., *Business Component Factory: a Comprehensive Overview of Business Component Development for the Enterprise.* John Wiley & Sons. 2000.

[9] Le Dinh T., *Information System Upon Information Systems: a Conceptual Framework.* Doctoral dissertation. University of Geneva, 2005.

[10] Lupeikienė A. Integrated Enterprise Information System Development through Component Abstraction. In *Proceedings of the Seventh International Baltic Conference on Databases and Information Systems*, Vasilecas O., Eder J., Čaplinskas A., (eds.), Institute of Electrical and Electronics Engineers, 2006, ISBN 1-4244-0345-6, p. 168-174.

[11] Čaplinskas A., Lupeikienė A., Programų, informacinių ir verslo sistemoms kurti naudojamų žinių formalizavimo problemos. Paskelbta: Lietuvos matematikos rinkinys, 41, spec. nr., ISSN 0132-2818. Vilnius: MII, 2001, p. 259-266.

[12] Lupeikienė A., Agentinių technologijų panaudojimo reikalavimai kuriant komponentines verslo, informacines ir programų sistemas. Paskelbta: Informacinės technologijos 2003, Konferencijos pranešimų medžiaga. ISBN 9955-09-335-8. Kaunas:

Kauno technologijos universitetas, 2003, p. VI-16 - VI-22. (Ir CD „XXI Amžiaus informacinės technologijos“. ISBN 9955-09-336-6. Kaunas: KTU, 2003).

[13] Lupeikienė A., Giedrimas V., Komponento specifikacijos formalizavimas. Lietuvos matematikos rinkinys, 44 (spec. nr.), 2004, ISSN 0132-2818, p. 276-280.

[14] Lupeikienė A., Integrated Enterprise Information Systems: Thinking in Component Concepts. Databases and Information Systems IV. Selected Papers from the Seventh International Baltic Conference DB&IS'2006, Vasilecas O., Eder J., Čaplinskas A., (eds.), IOS Press, 2007, ISSN 0922-6389, p. 203-215.

[15] Pranevičius H., Kompiuterių tinklų protokolų formalusis specifikavimas ir analizė: agregatinis metodas. Kaunas, 2005

[16] Pranevičius H., Vaškevičiūtė R., Organizacijos integruotos informacinės sistemos specifikavimas naudojant PLA. *Informacinės technologijos 2007, Konferencijos pranešimų medžiaga.* Kaunas. Technologija. 2007, p. 394-398.

[17] Stojanovic Z., *A Method for Component-Based and Service-Oriented Software Systems Engineering.* Doctoral Dissertation, Delft University of Technology. The Netherlands, 2005.

[18] Szyperski C., *Component Software: Beyond Object-Oriented Programming.* ACM Press. Addison -Wesley. 2002.

[19] Volter M., A taxonomy for components. *Journal of Object Technology,* vol. 2. no. 4. July - August 2003, p. 119-125.

[20] Wang J. A., Algebra for components. In: Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics, Vol. 5, Computer Science I , Callaos N., Leng T., and Sanchez B., (eds), International Institute of Informatics and Systemics, 2002, p. 213–218.

5. SANTRAUKA ANGLŲ KALBA (SUMMARY)

Formalization of Integrated Business Systems Used Component Abstraction

Summary

This final master degree paper analysis the integrated enterprise information system, its development and methodology using component based method. This method shows how the component - oriented paradigm can be used for developing system in all three layers of enterprise. The paper presents the main concept and its principles, context of paradigm through the enterprise development. The method offers the usage of component abstraction like the technique of realisation. The main principles can reduce the difficultness and maintenance of integrated enterprise information system.

The integrated enterprise information system connects enterprise information treatment, software and hardware components to the totality. The establishment of this kind of systems, its components, modular architecture has its specific peculiarities. The main objective of the paper – is to research a possibility of using the componentised paradigm in the establishment of the integrated enterprise information systems. Also, to analyse the main problems, which has to be solved if you want to engineer and implement this kind of systems.

The master degree paper shows the problems, which exists in formalization of integrated enterprise systems based on the componentised paradigm. It analysis the componentised system development paradigm, persuading the transferring of componentised system establishment ideas into the integrated information system layer. The paper analysis the main concepts of the component and its main parts, the singularity of different abstraction layer components and the ways of connection. It formulates the main requirements for the information system components and shows the existing methods for solving the problems.

6. SANTRUMPOS IR TERMINAI

IVS (angl., Integrated Enterprise System) - integruota verslo sistema

PLA (angl., Piece-Linear Aggregate) - tiesinių agregatų modeliavimo metodika

UML (angl., Unified Modelling Language) - unifikuota modeliavimo kalba (OMG grupės standartas)

BPMN (angl., Business Process Modelling Notation) - verslo proceso valdymo žymėjimas

IOIS (angl., Integrated Enterprise Information System) - integruota organizacijos informacinė sistema

OCL (angl., Object Constraint Language) - objektų apribojimo kalba

IS (angl., Information System) - informacinė sistema

COST (angl., Commercial-Off-The-Shelf) - rinkoje įsigyti pagaminti komponentai

CASE (angl., Computer-Aided Systems Engineering) - kompiuterizuotas informacijos sistemų kūrimas

DB (angl., Database) - duomenų bazė

DBVS (angl., Database Management System) - duomenų bazių valdymo sistema

GC (angl., Life Cycle) - gyvavimo ciklas (pvz., informacijos sistemos GC)

MDA (angl., Model-Driven Architecture) - OMG grupės pasiūlytas principas modeliais grindžiamam sistemų kūrimui

IT (angl., Information Technologies) - informacinės technologijos

7. PRIEDAI

1 priedas

Komponento „Balsavimas“ formali specifikacija PLA kalba

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

   x_1 - atėjo diskusijų ciklo rezultatai;

   x_2 - atėjo data;

   x_3 - atėjo surinkti balsai.

2. Išėjimo signalų aibė $Y = \{y_1, y_2, y_3, y_4, y_5\}$, čia:

   y_1 - prasideda diskusijų ciklas;

   y_2 - paskelbiamas balsavimas;

   y_3 - paskelbiamas balsavimas su perspėjimus;

   y_4 - prasideda balsų rinkimas.

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

   e'_1 - atėjo signalas x_1;

   e'_2 - atėjo signalas x_2;

4. Vidinių įvykių aibė $E'' = \{e''_i\}$, $i = \overline{1,15}$, čia:

   e''_1 - gauti problemų sąrašą;

   e''_2 - peržiūrėti problemų sąrašą;

   e''_3 - nustatyti, ar yra aktualių problemų;

   e''_4 - pradėti diskusijų ciklą;

   e''_5 - paskelbti problemas;

   e''_6 - pradėti balsų surinkimą;

   e''_7 - parengti balsavimo rezultatus;

   e''_8 - nustatyti, ar balsavo pakankamai narių;

   e''_9 - paskelbti rezultatus;

   e''_{10} - nustatyti, ar nariai buvo įspėti apie balsavimą;

   e''_{11} - iš naujo paskelbti balsavimą ir įspėti narius;

$e_{12}^{"}$ - sumažinti balsuojančių skaičių ir perskaičiuoti rezultatus;

5. Diskrečioji agregato būsenos dedamoji $v(t_m) = \{disk(t_m), bals(t_m)\}$, čia:

$disk(t_m) \in \{0,1\}$, 0 – diskusijų ciklas nepradėtas, 1 – diskusijų ciklas pradėtas;

$bals(t_m) \in \{0,1\}$, 0 – balsų surinkimas nepradėtas, 1 – balsų surinkimas pradėtas.

6. Tolydžioji agregato būsenos dedamoji $z_v(t_m) = \{w(e_i^{"}, t_m)\}, i = \overline{1,15}$, čia:

$w(e_1^{"}, t_m)$ - laiko momentas, kai gautas problemų sąrašas;

$w(e_2^{"}, t_m)$ - laiko momentas, kai peržiūrėtas problemų sąrašas;

$w(e_3^{"}, t_m)$ - laiko momentas, kai nustatyta, ar yra aktualių problemų;

$w(e_4^{"}, t_m)$ - laiko momentas, kai pradėtas diskusijų ciklas;

$w(e_5^{"}, t_m)$ - laiko momentas, kai paskelbtos problemos;

$w(e_6^{"}, t_m)$ - laiko momentas, kai pradėtas balsų surinkimas;

$w(e_7^{"}, t_m)$ - laiko momentas, kai pradėti parengti balsavimo rezultatai;

$w(e_8^{"}, t_m)$ - laiko momentas, kai nustatyta, ar balsavo pakankamai narių;

$w(e_9^{"}, t_m)$ - laiko momentas, kai pradėti skelbti balsavimo rezultatai;

$w(e_{10}^{"}, t_m)$ - laiko momentas, kai nustatyta, ar nariai buvo įspėti apie balsavimą;

$w(e_{11}^{"}, t_m)$ - laiko momentas, kai iš naujo paskelbtas balsavimas ir nariai įspėti;

$w(e_{12}^{"}, t_m)$ - laiko momentas, kai pradėti balsuojančiųjų skaičiaus sumažinimas ir rezultatų perskaičiavimas.

7. Valdymo sekos.

$e_1^{"} \to \{\zeta_1\}, i = 0, \infty$. ζ_1 - laiko tarpas iki gautas problemų sąrašas;

$e_2^{"} \to \{\zeta_{2i}\}, i = 0, \infty$. ζ_2 - laiko tarpas iki problemų sąrašo peržiūrėjimo;

$e_3^{"} \to \{\zeta_{3i}\}, i = 0, \infty$. ζ_3 - laiko tarpas iki nustatymo, ar yra aktualių problemų pradžios;

$e_4^{"} \to \{\zeta_{4i}\}, i = 0, \infty$. ζ_4 - laiko tarpas iki diskusijų ciklo pradžios;

$e_5^{"} \to \{\zeta_{5i}\}, i = 0, \infty$. ζ_5 - laiko tarpas iki problemų paskelbimo;

$e_6^{"} \to \{\zeta_{6i}\}, i = 0, \infty$. ζ_6 - laiko tarpas iki balsų surinkimo pradžios;

$e_7^{"} \to \{\zeta_{7i}\}, i = 0, \infty$. ζ_7 - laiko tarpas iki rezultatų parengimo pradžios;

$e_8^{"} \to \{\zeta_{8i}\}, i = 0, \infty$. ζ_8 - laiko tarpas iki nustatymo ar balsavo pakankamai narių, pradžios;

$e_9^{"} \to \{\zeta_{9i}\}, i = 0, \infty$. ζ_9 - laiko tarpas iki rezultatų paskelbimo;

$e_{10}^{"} \to \{\zeta_{10i}\}$, $i = 0, \infty$. ζ_{10} - laiko tarpas iki nustatymo, ar nariai buvo įspėti apie balsavimą pradžios;

$e_{11}^{"} \to \{\zeta_{11i}\}$, $i = 0, \infty$. ζ_{11} - laiko tarpas iki pakartotinio balsavimo paskelbimo ir narių įspėjimo pradžios;

$e_{12}^{"} \to \{\zeta_{12i}\}$, $i = 0, \infty$. ζ_{12} - laiko tarpas iki balsuojančiųjų skaičiaus sumažinimo ir rezultatų perskaičiavimo pradžios.

8. Konstanta Pr – balsavimo proceso pradžios parametras (šiuo atveju Pr = „penktadienis“).
9. Pradinė būsena. $z_v(t_m) = t_0; disk(t_m) = 0; bals(t_m) = 1.$
10. Perėjimo ir išėjimo operatoriai.

$H(e_1^{"})$: /gautas problemų sąrašas/ $w(e_1^{"}, t_{m+1}) = t_m + \zeta_{2m}$

$H(e_1^{'})$: /atėjo signalas x_1/ $w(e_1^{'}, t_{m+1}) = t_m + \zeta_{5m}$

$H(e_2^{"})$: /peržiūrėti problemų sąrašą/ $w(e_2^{"}, t_{m+1}) = t_m + \zeta_{3m}$

$H(e_2^{'})$: /atėjo signalas x_2/ $w(e_2^{'}, t_{m+1}) = \begin{cases} t_m + \zeta_{1m}, jei\_w(e_2^{'}, t_m) = \Pr \\ \infty, jei\_w(e_2^{"}, t_m) \neq \Pr \end{cases}$

$H(e_3^{"})$: /ar yra aktualių problemų/ $w(e_3^{"}, t_{m+1}) = \begin{cases} t_m + \zeta_{4m}, jei\_w(e_3^{"}, t_m) = 1 \\ \infty, jei\_w(e_3^{"}, t_m) = 0 \end{cases}$

$H(e_3^{'})$: /atėjo signalas x_3/ $w(e_3^{"}, t_{m+1}) = t_m + \zeta_{7m}$

$H(e_4^{"})$: /pradėti diskusijų ciklą/ $disk(t_{m+1}) = 1$

$H(e_5^{"})$: /paskelbti problemas/ $w(e_5^{"}, t_{m+1}) = \begin{cases} t_m + \zeta_{6m}, jei\_disk(t_m) = 1 \\ \infty, jei\_disk(t_m) = 0 \end{cases}; disk(t_{m+1}) = 0$

$H(e_6^{"})$: /pradėti balsų surinkimą/ $bals(t_{m+1}) = 1$

$H(e_7^{"})$: /parengti rezultatus/ $w(e_7^{"}, t_{m+1}) = \begin{cases} t_m + \zeta_{8m}, jei\_bals(t_m) = 1 \\ \infty, jei\_bals(t_m) = 0 \end{cases}; bals(t_{m+1}) = 0;$

$H(e_8^{"})$: /ar balsavo pakankamai narių/ $w(e_8^{"}, t_{m+1}) = \begin{cases} t_m + \zeta_{9m}, jei\_w(e_8^{"}, t_m) = 1 \\ t_m + \zeta_{10m}, jei\_w(e_8^{"}, t_m) = 0 \end{cases}$

$H(e_9^{"})$: /paskelbti rezultatus/ $w(e_9^{"}, t_{m+1}) = \infty$

$H(e_{10}^{"})$: /ar nariai buvo įspėti apie balsavimą/

$w(e_{10}^{"}, t_{m+1}) = \begin{cases} t_m + \zeta_{12m}, jei\_w(e_{10}^{"}, t_m) = 1 \\ t_m + \zeta_{11m}, jei\_w(e_{10}^{"}, t_m) = 0 \end{cases}$

$H(e_{11}^{"})$: /iš naujo paskelbti balsavimą ir įspėti narius/ $w(e_7^{"}, t_{m+1}) = t_m + \zeta_{6m}$

$H(e_{12}^{"})$: /sumažinti balsuojančiųjų skaičių ir perskaičiuoti rezultatus/

$w(e_7^{"}, t_{m+1}) = t_m + \zeta_{9m}$

$G(e_1^{"})$: y = 0;

$G(e_2^{"})$: y = 0;

$G(e_3^{"})$: y = 0;

$G(e_4^{"})$: y = y_1;

$G(e_5^{"})$: y = y_2;

$G(e_6^{"})$: y = y_5;

$G(e_7^{"})$: y = 0;

$G(e_8^{"})$: y = 0;

$G(e_9^{"})$: y = y_3;

$G(e_{10}^{"})$: y = 0;

$G(e_{11}^{"})$: y = y_4;

$G(e_{12}^{"})$: y = 0;