RĪGAS TEHNISKĀ UNIVERSITĀTE
Datorzinātnes un informācijas tehnoloģijas fakultāte
Lietišķo datorsistēmu institūts

Nataļja PAVLOVA
Datorsistēmas doktora programmas doktorante

PLATFORMNEATKARĪGĀ MODEĻA IZSTRĀDES PIEEJA MODEĻVADĀMAS ARHITEKTŪRAS IETVARĀ

Promocijas darba kopsavilkums

Zinātniskā vadītāja
Dr.sc.ing., asoc. profesore
O.ŅIKIFOROVA

Rīga 2007

UDK

Pavlova N. Platformneatkarīgā modeļa izstrādes pieeja modeļvadāmas arhitektūras ietvarā. Promocijas darba kopsavilkums. – R.:RTU, 2007.-25 lpp.

Iespiests saskaņā ar DITF LDI institūta 2007.gada 23.maija lēmumu, protokols Nr. 54.

Šis darbs izstrādāts ar Eiropas Sociālā fonda atbalstu Nacionālās programmas „Atbalsts doktorantūras programmu īstenošanai un pēcdoktorantūras pētījumiem" projekta „Atbalsts RTU doktorantūras attīstībai" ietvaros.

ISBN

PROMOCIJAS DARBS
IZVIRZĪTS RĪGAS TEHNISKAJĀ UNIVERSITĀTĒ
INŽENIERZINĀTŅU DOKTORA GRĀDA IEGŪŠANAI

Promocijas darbs inženierzinātņu doktora grāda iegūšanai tiek publiski aizstāvēts 2008.gada 4.februārī Rīgas Tehniskās universitātes Datorzinātnes un informācijas tehnoloģijas fakultātē, Meža ielā 1/3, 202 auditorijā.

OFICIĀLIE OPONENTI:
Profesors, Dr.habil.sc.ing. Jānis Grundspeņķis
Rīgas Tehniskā Universitāte, Latvija

Asoc. profesors, Dr.inž., Boriss Mišņevs
Transporta un Sakaru Institūts, Latvija

Profesore, Dr. Wita Wojtkowski,
Boise State University, ASV

APSTIPRINĀJUMS

Es apstiprinu, ka esmu izstrādājusi doto promocijas darbu, kas iesniegts izskatīšanai Rīgas Tehniskajā universitātē inženierzinātņu doktora grāda iegūšanai. Promocijas darbs nav iesniegts zinātniskā grāda iegūšanai nevienā citā universitātē.

Nataļja Pavlova _____(Paraksts)

Datums: _____

Promocijas darbs uzrakstīts angļu valodā, satur ievadu, 4 nodaļas, nobeigumu, literatūras sarakstu, 7 pielikumus, 87 attēlus, 10 tabulas, kopā 153 lappuses. Literatūras sarakstā ir 103 nosaukumi.

VISPĀRĪGS DARBA RAKSTUROJUMS

Pētījuma pamatojums

Viens no mūsdienu programmatūras inženierijas pētījumu mērķiem ir atrast programmatūras izstrādes procesu, kurš nodrošina ātru un kvalitatīvu programmatūras izstrādi. Šobrīd piedāvātās metodoloģijas un pieejas mēģina uzlabot programmatūras izstrādes procesu tā, lai procesā iekļautās aktivitātes varētu ātri realizēt un netiktu zaudēta kvalitāte. Lai sasniegtu šo mērķi, būtiska ir modeļa un modelēšanas loma. Viena no populārākajām pieejām programmatūras izstrādē ir modeļvadāma arhitektūra (angl. Model Driven Architecture – MDA) [48]. Objektu pārvaldības grupas (angl. Object Management Group – OMG) piedāvātajā pieejā modeļvadāma arhitektūra stratēģiski ir centrālā komponente, ko izmanto, lai samazinātu programmatūras izstrādes sarežģītību [44]. MDA piedāvā specificēt sistēmu, sadalot skatījumu uz sistēmu trīs dažādos abstrakcijas līmeņos:

1. Augstākais abstrakcijas līmenis, kurā ir specificēta sistēmas funkcionēšana, tiek saukts par "skaitļošanas" neatkarīgu modeli (angl. Computation Independent Model - CIM).
2. Sistēmas funkcionēšanas modelis neatkarīgi no izvēlētās realizācijas vides tiek saukta par platformneatkarīgu modeli (angl. Platform Independent Model - PIM).
3. Sistēmas funkcionēšanas modelis implementēšanas platformas jēdzienos tiek saukta par platformas specifisku modeli (angl. Platform Specific Model - PSM).

Modeļvadāmā arhitektūrā galvenie elementi programmatūras izstrādē ir modeļi. Visas aktivitātes ir definētas, kā pāreja no platformneatkarīga modeļa uz platformai specifisku modeli un no platformas specifiska modeļa uz programmatūras kodu. Īpaši svarīga loma programmatūras sistēmas izstrādē ir platformneatkarīgā modeļa kvalitātei, vai tā iespējai korekti un precīzi attēlot izstrādājamo sistēmu [8].

Sistēmas modelēšana, kuru atbalsta datorizētas programminženierijas (angl. Computer-Aided Software Engineering – CASE) rīki, nav nekas jauns programmatūras izstrādē. Jau 20.gs. astoņdesmitajos gados sāka plaši attīstīt programmatūras sistēmas, kas bija orientētas uz savstarpēji saistītu modeļu ģenerēšanu, dažādu diagrammu elementu savstarpēju saistību sistēmas modelī, un programmatūras produktivitātes un kvalitātes problēmu risināšanu [5], [24]. Pašreiz modeļvadāmas arhitektūras attīstība un atbalsts rīkos ir pamatots līdzīgi [48] un MDA atbalsta rīki piedāvā izmantot, integrēt un ģenerēt vienotas modelēšanas valodas (angl. Unified Modeling Language – UML) diagrammas dažādos abstrakcijas līmeņos [39]. Manipulācijas ar modeļiem padara iespējamāku programmatūras izstrādes automatizēšanu ar CASE rīkiem [12], [23], [29], [30]. Pagaidām nav iespējams apgalvot, ka rīkos ir pilnībā atbalstīti MDA pamatprincipi pilnībā, jo „pilna koda ģenerēšana", sākot no skaitļošanas neatkarīgā līmeņa vēl nav iekļauta MDA rīkos. Problemātiskākā vieta ir sākotnējā sistēmas modelēšana, kas tiek veikta, balstoties uz problēmvides zināšanām [24]. Parasti eksistē trīs sistēmas modelēšanas aspekti jeb skatījuma līmeņi: konceptuālais, loģiskais un fiziskais [54]. Mūsdienu tehnoloģijas vairumā gadījumu izmanto fiziskā līmeņa projektēšanu [30]. Skatījuma līmeņa „paaugstināšana" no fiziskā līmeņa uz konceptuālo līmeni ir viens no aktuālākajiem uzdevumiem programmatūras izstrādes tehnoloģijās, tāpēc UML modeļu savstarpējā pārveidošana tiek intensīvi pētīta [8]. Transformācijas valodu principi tiek piedāvāti [23]. Pastiprināta uzmanība tiek pievērsta UML klašu diagrammu izstrādes automatizēšanai, jo eksistējošajos CASE rīkos tās kalpo par programmatūras sistēmas izstrādes avotu, un ir pamats datu bāzu specifikācijai, grafiska lietotāja interfeisa un lietojuma sistēmas koda ģenerēšanai [51].

Promocijas darba mērķis

Promocijas darba mērķis ir piedāvāt formālu pieeju klašu diagrammas konstruēšanai, kas MDA ietvaros ir ļoti būtiska platformneatkarīga modeļa izstrādei no sistēmas sākotnējā modeļa. Sistēmas modelis tiek definēts biznesa procesu un datu plūsmu struktūras jēdzienos. Promocijas darbā ir jāizstrādā pieeja platformneatkarīga modeļa konstruēšanai, kas apmierina MDA pamatprincipu – programmatūras izstrādi balstīt uz formālām transformācijām starp modeļiem, kas definēti UML notācijā [39].

Darba uzdevumi

Lai sasniegtu darba mērķi ir svarīgi:

- analizēt pašreizējo situāciju sistēmu modelēšanas un analīzes jomā, kur sistēmu analīzes modeļi ir definēti UML notācijā;
- atrast pieejas vai pieeju soļus, kuri var tikt pielietoti klašu diagrammas konstruēšanai no sākotnējas dokumentācijas;
- piedāvāt klašu diagrammas konstruēšanas algoritmu formālā veidā un pamatot tā realizāciju ar formāliem transformācijas modeļiem;
- pārbaudīt piedāvātā algoritma darbību praktiskos piemēros.

Promocijas darba uzdevumi var tikt formulēti šādi:

- apskatīt eksistējošās pieejas klašu diagrammas konstruēšanai;
- izdalīt atsevišķus klašu diagrammas modelēšanas posmus, noteikt klašu diagrammas elementu identificēšanas un automatizēšanas iespējamās pakāpes;
- definēt avota elementus klašu diagrammas ģenerēšanai, kas paaugstinātu tās automatizēšanas pakāpi;
- pārbaudīt izstrādāto klašu diagrammas ģenerēšanas algoritmu praktiskos piemēros, realizējot to ar programmatūras nodrošinājumu.

Pētījumu veikšanai izmantotās metodes

Teorētiskie pētījumi ir pamatoti ar pieejamo literatūras avotu analīzē noskaidrotajām modeļvadāmās arhitektūras transformācijas virknes nepilnībām un trūkumiem. Teorētiskie rezultāti ir iegūti, izmantojot grafu teorijas un sistēmas analīzes metodes. Promocijas darbā izstrādāto sistēmu prototipu realizācijai ir izmantotas programmatūras inženierijas metodes. Darbā ir izmantoti arī eksperimentāli pētījumi, pamatojoties uz ieejošo un izejošo loku attiecību grafā, sistēmas modeļa diagrammām, kas iegūtas analizējot sistēmas funkcionēšanas un struktūras attēlošanu ar divpusložu pieeju [42].

Promocijas darba zinātniskais jauninājums un praktiskā vērtība

Mūsdienās pētnieki mēģina pilnveidot MDA principus efektīvai pielietošanai, pētot dažādus MDA aspektus, modeļus un modeļu transformācijas [1], [2], [7], [12], [26], [28]. Tomēr joprojām MDA ietvaros nav definēta pilnīga modeļu pārveidošanas secība no skaitļošanas neatkarīga modeļa līdz programmas kodam. Tāpēc darba autore ir izvēlējusies izpētīt platformneatkarīga modeļa konstruēšanas iespējas un izveidot formālu algoritmu platformneatkarīga modeļa izstrādei, kas tiek definēts klašu diagrammas veidā, un kuru var izmantot klašu diagrammas transformācijai par platformas specifisku modeli.

Klašu diagrammas veidošana ir svarīgs posms modelēšanā, jo tā definē sistēmas struktūru. Klašu diagrammas veidošanas algoritms piedāvā iespēju izvairīties no iespējamiem riskiem sistēmas izstrādes gaitā. Ja klašu diagramma ir veidota, pamatojoties uz sākotnējo sistēmas informāciju un izmantojot formālu pieeju, tad var cerēt, ka lietotāja prasības būs korekti un precīzi attēlotas. Pašlaik programmatūras objektu struktūras izstrāde, atbildības sadalīšana starp sistēmas objektiem un to apvienošana klasēs ir darbietilpīgs intelektuāls

sistēmanalītiķa darbs. Darba autore piedāvā klašu diagrammas veidošanu balstīt uz formālām transformācijām, pārejot no sākotnējā sistēmas atspoguļojuma biznesa procesu modeļa un konceptu modeļa (datu struktūras) veidā.

Izstrādātajam algoritmam ir jāatbalsta sistēmas analīzes posms UML orientētajā izstrādē un līdz ar modelēšanas uzdevuma izpildes automatizēšanu jāsamazina kopējie programmatūras izstrādes izdevumi. Formālais algoritms ļauj izvairīties no pārpratumiem un neuzmanības kļūdām, informācijas zaudējumiem programmatūras modelēšanas procesā, kā arī nodrošina pāreju starp modeļiem, kas ir orientēti uz programmatūras pasūtītāja sistēmas dokumentāciju un programmatūras izstrādātāja sistēmas arhitektūru.

Pētījuma galvenie rezultāti ir publicēti vai pieņemti publikācijai šādās starptautiskajās konferencēs un rakstu krājumos:

1. *Chukina H., **Pavlova N.,** Nikiforova O. Development of platform independent model in the framework of MDA// Scientific Proceedings of Riga Technical University. 5th Series, Computer Science, Applied Computer Science, Vol. 30 – Riga: RTU, 2007, pp. 18-28.*
2. *Nikiforova, O., Kirikova M., **Pavlova N**. Principles of Model Driven Architecture in Knowledge Modeling for the Task of Study Program Evaluation// Databases and Information Systems IV, by IOS Press in the series "Frontiers in Artificial Intelligence and Applications", Vasilecas O., Eder J., Caplinskas A. (eds), 2007, pp.291-304.*
3. ***Pavlova N**. Several Outlines of Graph Theory in the Framework of MDA// Proceeding of the 15th International Conference "Information Systems Development" (ISD'06).- Hungary, 2006 (in press).*
4. *Nikiforova O., Kirikova M., **Pavlova N**. Two-Hemisphere Driven Approach: Application for Knowledge Modeling// Proceedings of the Seventh IEEE International Baltic Conference on Databses and Information Systems (BalticDB&IS'2006), Vilnius, Lithuania. – Vilnius: Technika, 2006, pp. 244-250.*
5. ***Pavlova N**., Nikiforova O. Formalization of Two-Hemisphere Model Driven Approach in the Framework of MDA// Proceedings of the 9th Conference "Information Systems Implementation and Modelling" (ISIM'06), April 24-26, 2006, Prerov, Czech Republic. – Ostrava: Jan Štefan MARQ., 2006, pp. 105-112.*
6. *Nikiforova O., Kuzmina M., **Pavlova N**. Formal Development of Platform Independent Model in the Framework of MDA: Myth or Reality// Scientific Proceedings of Riga Technical University, 5th Series, Computer Science, Applied Computer Science Vol.22. – Riga: RTU, 2005, pp.42-53.*
7. ***Pavlova N**., Nikiforova O. An Overview of Advanced Approaches for Construction of Platform-Independent System Model// Scientific Proceedings of Riga Technical University, 5th Series, Computer Science, Applied Computer Science Vol.22. – Riga: RTU, 2005, pp. 156-168.*
8. ***Pavlova N**., Nikiforova O. Review of information flows during object oriented system modeling// Scientific Proceedings of Riga Technical University5th Series, Computer Science, Applied Computer Science,Vol.17. – Riga: RTU, 2003, pp. 109-115.*

Darba struktūra un galvenie rezultāti

Promocijas darbs satur ievadu, četras nodaļas, nobeigumu, 7 pielikumus un bibliogrāfisko sarakstu. Darbs sastāv no 153 lappusēm, 87 attēliem un 10 tabulām. Bibliogrāfiskais saraksts satur 103 literatūras avotus.

Ievadā ir sniegta darba motivācija. Darbā ir aprakstīta MDA arhitektūra, ar to saistītās problēmas, un ir noteikts pētījuma apgabals. Ir definēts arī darba mērķis, uzskaitīti uzdevumi, pamatota promocijas darba aktualitāte, praktiskā nozīme un atspoguļoti jauninājumi.

Pirmajā nodaļā ir apskatīta modeļvadāmā arhitektūra. Šajā nodaļā ir definēti MDA modeļi, iespējamās MDA transformācijas un detalizēti aprakstīts platformneatkarīgs modelis. Pirmajā nodaļā ir aprakstītas platformneatkarīga modeļa izstrādes metodoloģijas un izskatīta iespēja noteikt vienu pieeju, kuru var izmantot par pamatu formāla algoritma izveidei. Tālākai

izpētei, analīzei un pilnveidošanai šī pētījuma ietvaros ir izvēlēta divpusložu modeļvadāma (angl. Two Hemisphere Model Driven - 2HMD) pieeja [35], kā viena no biznesmodelēšanā sakņotām pieejām, kas ir atbalstīta ar CASE rīkiem.

Otrajā nodaļā ir apskatīta modeļvadāmajā arhitektūrā definētu transformāciju detalizētāka analīze un piedāvāts sadalīt platformneatkarīga modeļa konstruēšanu divu apakšmodeļu - $PIM_{initial}$ (sākotnējs platformneatkarīgs modelis) un $PIM_{refined}$ (pilnveidots platformneatkarīgs modelis) secīgā izstrādē. Ir atspoguļoti abu apakšmodeļu savstarpējo transformāciju teorētiskie aspekti un ir izskatītas pārveidošanas iespējas no $PIM_{initial}$ uz $PIM_{refined}$, pamatojoties uz divpusložu pieeju. Otrajā nodaļā ir apskatītas transformācijas avota un mērķa modeļu definīcijas un definēti šo modeļu metamodeļi. Ir detalizēti aprakstīta divpusložu pieeja, kuras izvēle ir pamatota pirmajā nodaļā, un ir analizēti pieejas trūkumi un nepilnības. Nodaļas noslēgumā ir sniegtas rekomendācijas pieejas pilnveidošanai un uzlabošanai.

Trešā nodaļa atspoguļo pārveidotās divpusložu pieejas būtību un pielietojuma iespējas klašu diagrammas ģenerēšanai. Visas iespējamās transformācijas starp $PIM_{initial}$ un $PIM_{refined}$ ir atspoguļotas, izmantojot reālus un abstraktus piemērus. Katram piemēram ir parādīti klašu diagrammas ģenerēšanas etapi un rezultāts, kā arī visu iespējamo elementu attēlošana starp $PIM_{initial}$ un $PIM_{refined}$ modeļiem ir apkopota tabulas veidā trešās nodaļās beigās.

Ceturtajā nodaļā ir apskatīts izstrādātā algoritma pielietojums. Rīka darbība, kurā ir realizēts izstrādātais algoritms, ir pārbaudīta, lietojot dažāda tipa piemērus. Algoritms ir aprobēts, izmantojot dažādas biznesa procesu un konceptu modeļa notācijas vienai un tai pašai problēmvidei, kā arī izmantojot vienu un to pašu biznesa procesu un konceptu modeļa notāciju dažādām problēmvidēm. Visos piemēros ģenerētās klašu diagrammas ir savstarpēji salīdzinātas un ir analizēta šo diagrammu struktūra.

Nobeigumā ir aprakstīti darba rezultāti, ir sniegti pētījuma secinājumi un iespējamie turpmākie pētījumu virzieni.

Promocijas darba kopsavilkuma saturs

Ievads. Ievadā ir aprakstīta pētījuma aktualitāte, zinātniskā novitāte, darba mērķis, uzdevumi, zinātniskā un praktiskā nozīme. Ir sniegts arī darba struktūras apraksts.

1. nodaļa. MDA principi un risinājumi

Pirmajā nodaļā ir aprakstīti modeļvadāmas arhitektūras galvenie principi. Pētījuma pamatjautājums ir formālās transformācijas iespējas MDA ietvaros, līmenī, kur notiek pāreja no skaitļošanas neatkarīga modeļa uz platformneatkarīgu modeli un iekšējā platformneatkarīga modeļa transformācija [27]. Pirmajā nodaļā ir dots platformneatkarīga modeļa konstruēšanas pieeju apraksts.

Apakšnodaļā 1.1. ir aprakstīti MDA pamatposmi un transformācijas starp tiem.

MDA nodala biznesa aspektus no sistēmas izstrādes aspektiem, kuri ir specifiski tehnoloģijas platformai [27]. MDA nosaka, ka sākotnēji sistēmas prasības ir jāspecificē neatkarīgi no platformas, un pēc tam, kad ir izvēlēta konkrēta sistēmas realizācijas vide, specifiski platformai. MDA ir definēta trīs modeļu līmeņos:

- skaitļošanas neatkarīgs modelis (angl. Computation Independent Model – CIM),
- platformneatkarīgs modelis (angl. Platform Independent Model – PIM),
- platformas specifisks modelis (angl. Platform Specific Model – PSM).

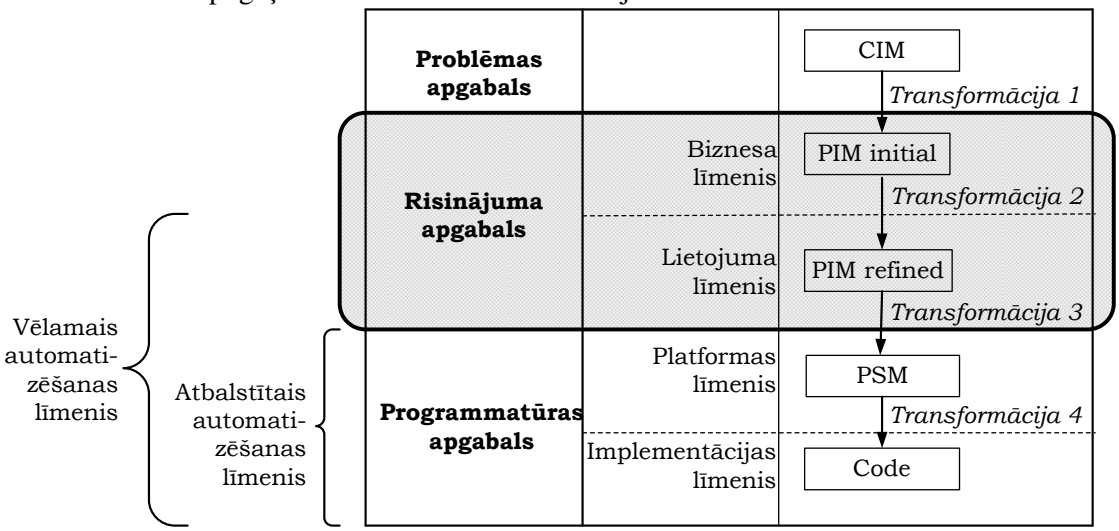
Platformas specifisku modeli tālāk ir iespējams pārveidot programmatūras kodā.

MDA piedāvā programmatūras izstrādes pieeju, kurā galvenā nozīme ir modeļiem un modeļu transformācijām [21]. Programmatūra tiek izstrādāta, konstruējot vienu vai vairākus modeļus un transformējot tos citos modeļos vai programmas kodā [22], [20]. Galvenais

mērķis šajā pieejā ir iegūt formālu sistēmas atspoguļojumu pēc iespējas augstākā abstrakcijas līmenī. Mūsdienās MDA principi tiek atbalstīti ar dažādiem rīkiem, kas pārsvarā nodrošina transformācijas starp platformneatkarīgu modeli un platformas specifisku modeli, un tālāk pārveidošanu uz kodu. Pētnieki cenšas paaugstināt transformācijas iespējas līdz platformneatkarīgam modelim, lai no to varētu formāli konstruēt vai ģenerēt no skaitļošanas neatkarīgu modeli, un pilnā mērā realizēt MDA pamatprincipus un prasības [37], [42].

Transformācijas secība MDA ietvaros ir atspoguļota 1. attēlā [38]. Pārveidošana ir sadalīta trīs apgabalos [42]. Pirmais apgabals ir problēmas apgabals, kurš iekļauj no skaitļošanas neatkarīgu modeli (CIM), kura būtība ir attēlot galvenās prasības un biznesa procesus [16] .

No skaitļošanas neatkarīgā modelī tiek apskatīta sistēmas specifikācija sākotnējiem elementiem problēmas līmenī, ko ir iespējams transformēt uz platformneatkarīgu modeli. Platformneatkarīgs modelis piedāvā sistēmas struktūras un funkcionēšanas formālo specifikāciju, norobežojoties no tehnisku detaļu izpētes, un apskata izstrādātas sistēmas risinājuma loģiskā aspektus. Risinājuma apgabala modeļu izstrāde balstās uz modelim nepieciešamo elementu iegūšanu no problēmas apgabalā realizētā apraksta. Apgabala pārveidošana ir atspoguļota 1. attēlā kā Transformācija 1.



1. att. MDA modeļu transformācijas [38]

Platformneatkarīgs modelis, kas iegūts, kā pirmās transformācijas rezultāts, tiešā veidā nav derīgs turpmākai pārveidošanai par platformas specifisku modeli. Iegūtais platformneatkarīgais modelis ir jāpilnveido par modeli, kuru ir iespējams izmantot platformas specifiska modeļa ģenerēšanai. Tas nozīmē, ka platformneatkarīgais modelis ir jāsadala divos apakšmodeļos, kurus darba autore ir nosaukusi par $PIM_{initial}$ un $PIM_{refined}$. Lai apmierinātu MDA formālās transformācijas prasības, no skaitļošanas neatkarīga modeļa iegūtais rezultāts, kas tiek saukts par $PIM_{initial}$, ir jāpārveido par modeli, kas tiek saukts par $PIM_{refined}$, Šīs aktivitātes tiek veiktas, lai būtu iespējams realizēt transformāciju uz platformas līmeni. PIM pārveidošana notiek risinājuma apgabala ietvaros, un 1. attēlā tā ir atspoguļota kā Transformācija 2, kas īsteno pāreju no biznesa līmeņa uz lietojuma līmeņa modeli. $PIM_{refined}$ tālāk var izmantot pārveidošanai par platformas specifisku modeli [38], kas 1. attēlā ir apzīmēta kā Transformācija 3. Par darba pētījuma sfēru ir izvēlēts transformācijas risinājuma apgabals, kurā ir definēta pāreja no biznesa līmeņa uz lietojuma līmeni. Skatīt 1. attēlu, kur pētījuma apgabals ir atspoguļots kā taisnstūris ar noapaļotiem stūriem un iezīmēts tumšākā krāsā.

Apakšnodaļā 1.2. ir detalizēti aprakstīta platformneatkarīga modeļa izstrāde atbilstoši pētījuma sfērā definētiem modeļa izstrādes etapiem.

Transformējot $PIM_{initial}$ uz detalizētāku modeli $PIM_{refined}$ [27], tiek veikta modeļa bagātināšana. Pilnveidošanas procesā sākotnējais modelis no augstāka abstrakcijas līmeņa tiek pārveidots un apskatīts zemākā abstrakcijas līmenī, tomēr pārveidošanas laikā tas joprojām ir neatkarīgs no platformas.

Platformneatkarīga modeļa izstrādei ir definēti četri soļi [43], [9] :

- platformneatkarīga modeļa konteksta definēšana;
- platformneatkarīga modeļa prasību specificēšana;
- platformneatkarīga modeļa analīze;
- platformneatkarīga modeļa projektēšana.

Apakšnodaļā 1.3. ir apskatītas objektorientētās modelēšanas metodes un stratēģijas, aprakstītas informācijas plūsmas starp modeļu diagrammām un izvērtētas automatizēšanas iespējas.

Objektu modelēšanas tehnikā (angl. Object Modeling Technique - OMT), kuru piedāvā J.Rambo (angl. J.Rumbaugh) [47], [46], sistēmas izstrāde tiek sākta ar lietotāja intervēšanu un prasību izgūšanu [40]. Pamatojoties uz iegūtajām prasībām, tiek konstruēts lietošanas gadījumu modelis, un transformācijas realizācijai Rambo sniedz stingrus ieteikumus [47], [46]. Izmantojot 1.1. apakšnodaļā apskatīto platformneatkarīga modeļa sadalīšanu divos apakšmodeļos, var teikt, ka apakšmodelis $PIM_{refined}$ OMT pieejā ir atspoguļots kā klašu un stāvokļu diagrammas. Rambo nav piedāvājis stingras rekomendācijas klašu diagrammu konstruēšanai [47], [46], bet ir definējis kritērijus klašu izgūšanai no sistēmas apraksta. Šos kritērijus un ieteikumus var izmantot, kā daļēji formālas vadlīnijas $PIM_{initial}$ modeļa konstruēšanai, veicot pāreju no sistēmas darbības scenārijiem. Ir iespējams definēt objektu apmaiņu ar ziņojumiem, kur ziņojumi turpmāk kalpo kā pamats klašu objektu attiecību un metožu definēšanai [46].

Vienotais „Rational" process (angl. Rational Unified Process – RUP) [17] ir balstīts uz lietošanas gadījumiem. Klašu diagrammas konstruēšanas etapus, ko ir iespējams identificēt RUP procesā, rekomendē balstīt uz sistēmas modelēšanu no lietotāja prasībām, kas tiek definētas biznesa lietošanas gadījumu un biznesa objektu modeļu veidā. Šie modeļi ir $PIM_{initial}$, lietošanas gadījumu un mijiedarbības diagrammu konstruēšanas pamatā. Diagrammu konstruēšana ir manuāls process, kuru veic sistēmanalītiķis. $PIM_{refined}$ RUP ir atspoguļots kā klašu diagramma, kura ir konstruēta, izmantojot lietošanas gadījumus un mijiedarbības diagrammas.

K.Larmans (angl. C.Larman) [25] apraksta vēl vienu uz lietošanas gadījumiem balstītu pieeju un piedāvā sistēmas modelēšanas gaitā konstruēt arī konceptuālo modeli. Modelis turpmāk kalpo par pamatu klašu diagrammas izveidei un tiek veidots balstoties uz sistēmas aprakstu dabīgajā valodā. K.Larmans nesniedz formālas rekomendācijas, kā definēt un konstruēt sistēmas modeli un diagrammas. Modelēšana ir manuāls process, kuru veic analītiķis, lietojot K. Larmana vadlīnijas konceptu izgūšanai, lietošanas gadījumu un to scenāriju izveidei. Vadlīnijas nav pietiekami formālas, lai varētu realizēt pāreju no $PIM_{initial}$ uz $PIM_{refined}$, taču ir iespējams veikt klašu diagrammu konstruēšanu, balstoties uz konceptu modeli un lietošanas scenārijiem.

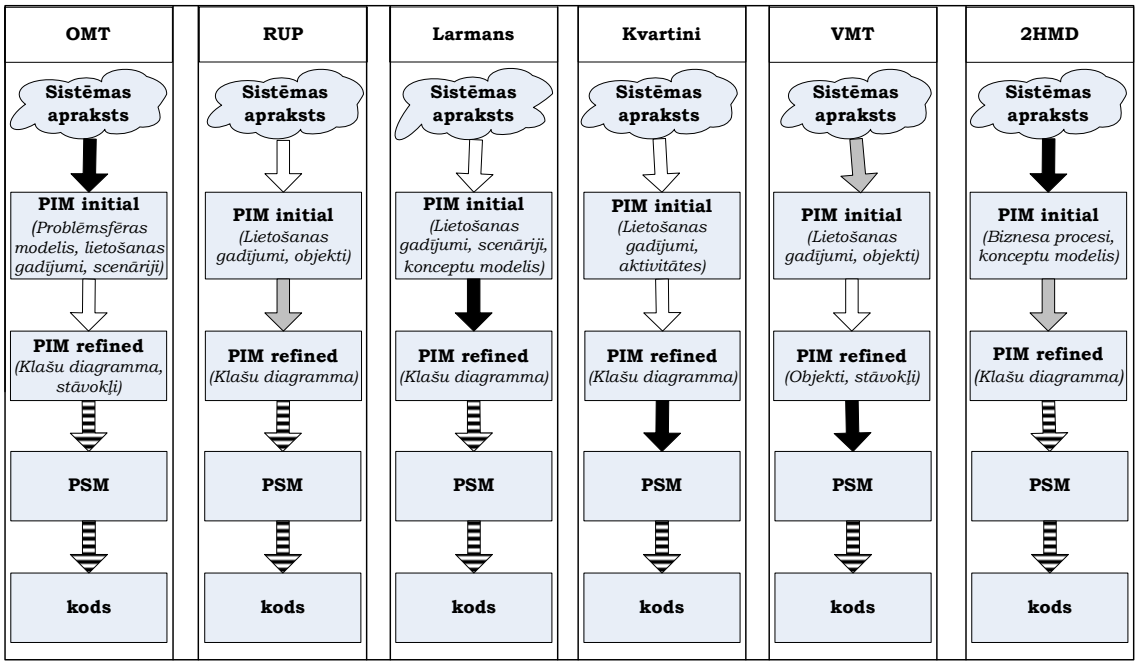
T.Kvatrani (angl. T.Quatrany) metodoloģijā [45] $PIM_{initial}$ ir atspoguļots ar lietošanas gadījumu un aktivitāšu diagrammām, un pāreja uz $PIM_{refined}$, kas ir līdzvērtīgs klašu diagrammai, ir manuāla, un to veic analītiķis.

Vizuālās modelēšanas tehnika (angl. Visual Modelling Technique - VMT) [53] sistēmas modelēšanai rekomendē izmantot aktivitāšu kopu, sākot no lietošanas gadījumiem un objektu mijiedarbībām līdz objektu un stāvokļu diagrammām. $PIM_{initial}$ modeļa konstruēšana ir

manuāls process, kas balstās uz sistēmanalītiķa intuīciju un intelektu. Nav piedāvāti arī ieteikumi diagrammu elementu identificēšanai, kas ļauj konstruēt $PIM_{refined}$.

Divpusložu (angl. Two Hemisphere Model Driven approach – 2HMD) pieeja [35] atbalsta pakāpenisku modeļu transformēšanu no problēmas uz programmatūras komponentēm. Ir rekomendēts programmatūras izstrādes procesu sākt no divpusložu problēmas apgabala modeļa, kur viena puslode atspoguļo funkcionālos aspektus, un otra atspoguļo atbilstošo datu struktūru. Divu modeļu sadarbība ļauj realizēt daļēji formālu zināšanu pāreju no viena modeļa uz citu [32]. Eksistē arī stingras vadlīnijas biznesa procesu un datu struktūras modeļu izveidei. $PIM_{initial}$ divpusložu pieejā ir atspoguļots ar konceptu un biznesa procesu modeli, bet $PIM_{refined}$ ar klašu diagrammu.

Apakšnodaļā 1.4. ir UML diagrammu modelēšanas metodes aprakstītas atbilstoši MDA galvenās idejas atbalstam. Promocijas darbā veiktās metožu analīzes un salīdzināšanas rezultāta grafisks atspoguļojums ir apskatāms 2. attēlā, kur ir attēlota CIM->$PIM_{initial}$->$PIM_{refined}$->PSM->kods transformācijas shēma. Pārejas starp modeļiem ir apzīmētas ar bultām.



2. att. Transformāciju analīze UML diagrammu modelēšanas metodēs

Bultu apzīmējumiem ir šāda nozīme:

- Baltas bultas ir manuālā transformācija un to var veikt tikai eksperts. Visi lēmumi tiek pieņemti balstoties uz pieredzi un intuīciju. Šādas transformācijas var pārbaudīt tikai subjektīvi.
- Pelēkas bultas ir modeļu transformācijas, balstoties uz rekomendācijām vai ieteicamām transformācijām. Nav precīzi definētu algoritmu, kā veikt šādas transformācijas, bet pastāv noteikumi un/ vai rekomendācijas, saskaņā ar kurām izstrādātājs var pieņemt lēmumus. Šīs transformācijas nevar būt atbalstītas ar rīkiem, jo rekomendācijas nav formalizētas.
- Melnās bultas ir daļēji formālas transformācijas, kas tiek veiktas, izmantojot speciālus algoritmus, precīzus norādījumus, noteikumus vai metodes, kuras apraksta, kā no viena modeļa elementiem var iegūt cita modeļa elementus. Algoritmi neaptver visu transformācijas procesu. Daži pārveidošanas likumi nav precīzi nodefinēti un joprojām ir nepieciešama cilvēka iejaukšanās.

- Svītrainās bultas ir automatizētas transformācijas, kur likumi ir pilnīgi un precīzi definēti. Šāda veida transformācija var tikt atbalstītas ar rīkiem un ir iespējams veikt pilnīgu viena modeļa pārveidošanu citā modelī.

Apakšnodaļā 1.5. ir apkopoti secinājumi par 1. nodaļā aprakstītā pētījuma rezultātiem. Ir analizētas transformācijas iespējas, pārejas starp modeļiem un apskatīto metodoloģiju soļu iespējamais formalitātes līmenis.

Veicot 2. attēla analīzi, var izdarīt šādus secinājumus:

- eksistē daudz metodes, kuras piedāvā paņēmienus, vadlīnijas, rekomendācijas sistēmas modeļa izveidei UML valodas notācijā;
- neviena no analizētajām metodēm neatbalsta pilnu programmatūras izstrādes procesu pat daļēji formālā līmenī, tāpēc pašlaik neeksistē rīki, kas ļauj automātiski veidot sistēmas modeli no sistēmas apraksta;
- viena no pamatproblēmām ir pāreja no biznesa līmeņa uz lietojuma līmeni, kas nav formalizēta nevienā no analizētajām metodēm, un tāpēc šī problēma ir risināta promocijas darbā;
- no apskatītajām metodēm turpmākai modeļa pilnveidošanai ir izvēlēta divpussložu pieeja, kas ļauj realizēt formālu klašu diagrammas izstrādi un kurā ir iespējams pielietot arī citu metožu formālos paņēmienus.

2. nodaļa. Transformācijas no $PIM_{initial}$ uz $PIM_{refined}$ divpussložu pieejā

Promocijas darba 2. nodaļa apraksta UML notācijas uzlabošanas iespējas sistēmas modelēšanai, vadoties pēc MDA principiem. Kā jau iepriekšējā nodaļā konstatēts galveno problēmu, kas neļauj nodrošināt MDA transformāciju virknes realizāciju, rada nepilnības modeļu transformācijās platformneatkarīgā modeļa līmenī. Darba 2. nodaļā uzmanība ir vērsta uz platformneatkarīga modeļa transformāciju izstrādi.

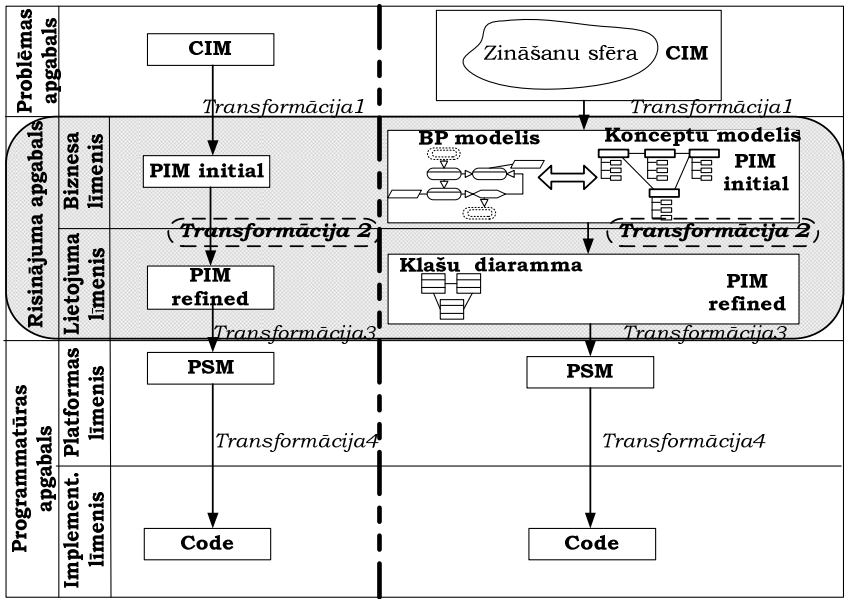
Apakšnodaļā 2.1. ir aplūkoti transformācijas pamati un detalizēti aprakstītas transformācijas iespējas divpussložu pieejā [35]. Divpussložu pieeja 3. attēlā ir attēlota atbilstoši etapiem, kas ir identificēti 1. nodaļā.

Platformneatkarīgs modelis arī 3. attēlā ir sadalīts divās daļās – $PIM_{initial}$ un $PIM_{refined.}$ Attēla kreisajā pusē ir parādīts MDA process kopumā, bet labajā pusē ir atspoguļots MDA process, izmantojot divpussložu pieejas elementus. Arī 3. attēlā pētījuma sfēra, kas iekļauj transformāciju starp $PIM_{initial}$ un $PIM_{refined}$, divpussložu pieejas terminos, ir iezīmēta pelēkā krāsā.

Divpussložu pieejā Transformācija 1 nozīmē biznesa procesu un konceptu modeļa konstruēšanu atbilstoši problēmvides situācijai. Šo transformāciju ir pētījuši vairāki autori [18], [19], [20], [32], [35], [36], un ir publicētas dažādas pieejas biznesa modelēšanai [4], [14]. Promocijas darbā autore pieņem, ka Transformācija 1 ir veikta formāli un ieejas informācija risinājuma apgabala biznesa līmenī, kas ir attēlota ar biznesa procesa diagrammām un konceptu modeļiem, ir pilnīga un atbilst problēmvides zināšanām [32], [35], [36].

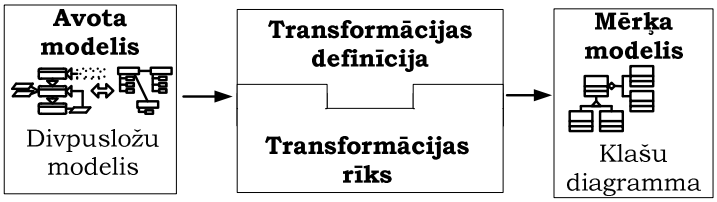
Transformācija 2 divpussložu pieejā ir realizēta kā pāreja no biznesa procesu diagrammas un sākotnējās datu struktūras uz klašu diagrammu (skatīt 3. attēlu). Oriģinālajā metodē transformācijai nav definēts formāls algoritms, un pārveidošanai ir nepieciešama cilvēka līdzdalība. Tas savukārt neatbilst MDA idejas prasībām, kas nosaka, ka visām transformācijām ir jābūt realizētām ar formāliem likumiem.

Formalitātes ieviešana pārejā Transformācija 2 ir apskatīta promocijas darba ietvaros. Attēlā 3. redzamās pārejas, Transformācija 3 un Transformācija 4, netiek apskatītas veiktā pētījuma ietvaros, jo tās jau ir automatizētas dažādos CASE rīkos.



3. att. MDA transformāciju ķēdes projekcija divpusložu pieejā

Formāli definētas pārejas starp modeļu elementiem ir iespējams implementēt modeļu transformācijas rīkos, kas, izmantojot avota modeli par ieejas informāciju, izejā automātiski izveido mērķa modeli [23]. Transformācijas rīka darbības princips un struktūra ir paradīta 4. attēlā. Visa informācija, kura ir iekļauta pārveidojamā modeļa aprakstā, tiek saukta par transformācijas definīciju [23] .



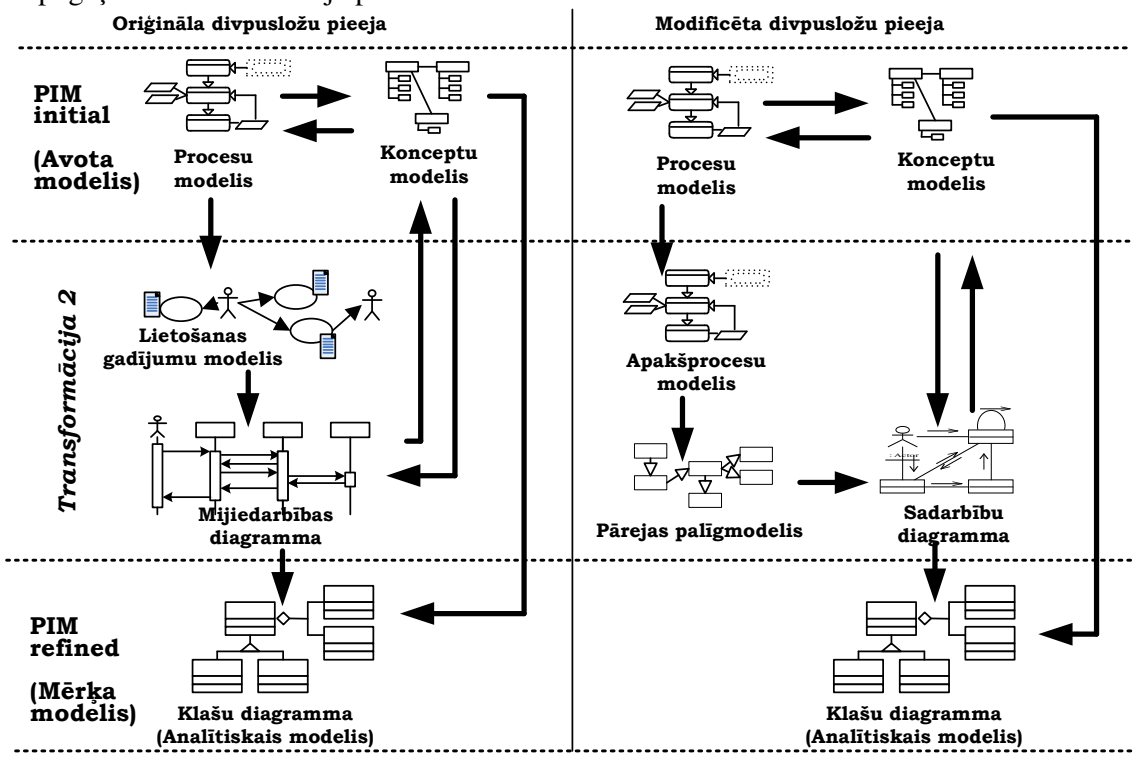
4. att. Modeļu transformēšanas rīka struktūra

Dažādiem modeļiem var būt vienādas transformācijas definīcijas. Katra transformācijas definīcija sastāv no transformācijas likumiem, kuri apraksta, kā no viena modeļa elementiem var iegūt cita modeļa elementus [23]. Transformācijas likumus veido attēlošanas likumi un iezīmēšana. 4. attēlā redzamajā shēmā ieejas modelis ir konstruēts ar biznesa procesu un konceptu modelēšanas notāciju. Izejas modelis ir atspoguļots kā klašu diagramma UML valodas notācijā, un transformācijas rīks izmanto transformācijas definīciju.

Apakšnodaļā 2.2. ir aprakstīts, kā definēt avota modeli transformācijai no $PIM_{initial}$ uz $PIM_{refined}$. Tā kā divpusložu pieeja piedāvā sākt sistēmas modelēšanu no biznesa procesu un konceptu modeļu konstruēšanas, divpusložu modelis ir definēts, kā avota modelis tālākai transformācijai. Avota modelis ir metamodelis, kas sastāv no šādiem elementiem: biznesa procesu diagrammas elementiem, kas atspoguļo procesu, notikumu un datu krātuvi, un konceptu modeļa elementiem, kas atspoguļo konceptu un tā īpašības. Visi elementi ir detalizēti aprakstīti promocijas darbā. Transformācijas likumus ir nepieciešams definēt, lai no avota modeļa elementiem būtu iespējams iegūt visus mērķa modeļa elementus. Promocijas darbā izvirzītā uzdevuma ietvaros tā ir transformācijas likumu definēšana, kas ļauj no divpusložu modeļa elementiem iegūt klašu diagrammas elementus.

Apakšnodaļā 2.3. ir aprakstīts, kā definē mērķa modeļa elementus transformācijai no $PIM_{initial}$ uz $PIM_{refined}$. Atbilstoši MDA modeļu transformācijas ķēdei, mērķa modelis ir $PIM_{refined}$, kas ir aprakstīts klašu diagrammas veidā UML valodas notācijā [39]. Mērķa modelis ir metamodelis, kas ir detalizēti aprakstīts promocijas darbā. Mērķa modeļa elementi ir klašu diagrammas pamatelementi: klase, atribūts, metode un asociācija starp klasēm.

Apakšnodaļā 2.4. aprakstītas iespējas transformēt avota modeli uz mērķa modeli, izmantojot oriģinālo divpusložu pieeju, kur $PIM_{initial}$ → $PIM_{refined}$ transformācija ir atspoguļota 5. attēla kreisajā pusē.



5. att. Pāreja Transformācija 2 divpusložu pieejā oriģinālajā un modificētajā versijā

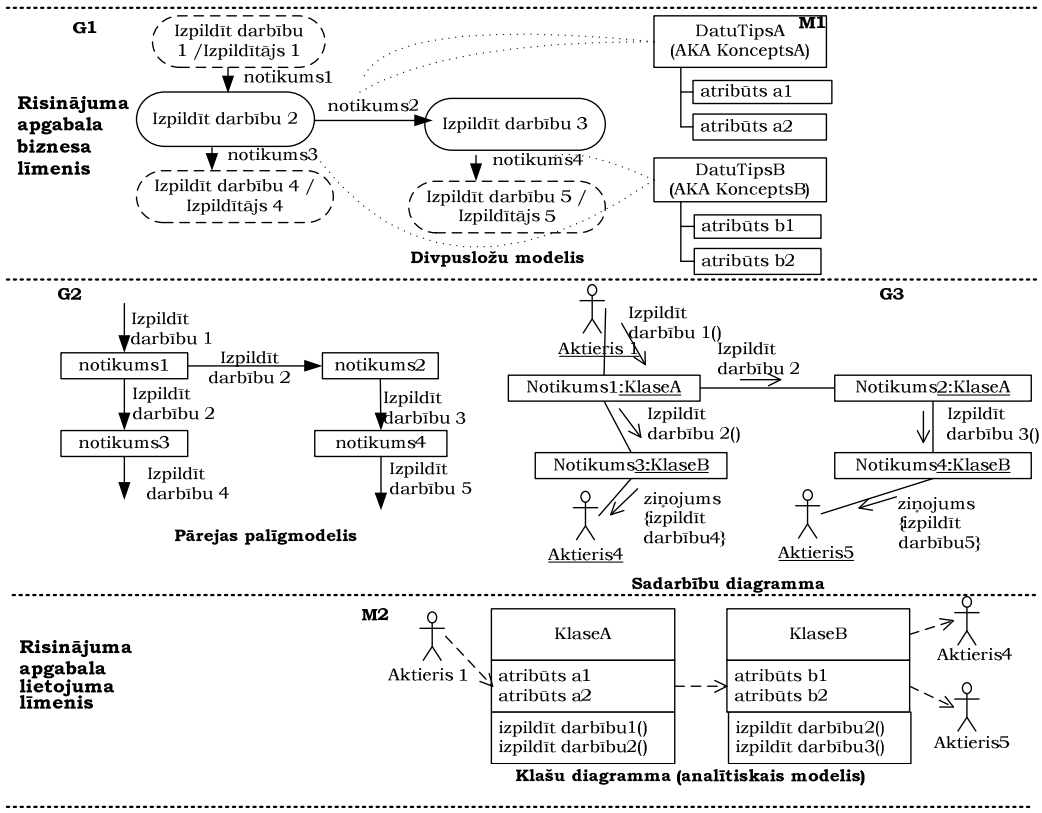
Divpusložu metodes pamatideja ir definēt, kā veikt sistēmas modelēšanu funkcionālajā (viena modeļa puslode) un konceptu modelī (otra modeļa puslode) [31] . Konceptu modelis atspoguļo biznesa sfēras jēdzienus. Biznesa procesu modelis attēlo procesus, kas tiek realizēti problēmas apgabalā [18]. Platformneatkarīga modeļa līmenī abi divpusložu modeļa skatījumi ir apvienoti vienā modelī, lai attēlotu gan sistēmas struktūras, gan dinamiskās īpašības. Pieeja nodrošina modeļu transformāciju no problēmas apgabala modeļiem uz platformas neatkarīgām programmatūras sastāvdaļām [35] .

Risinājumā, ko piedāvā divpusložu pieeja, lietošanas gadījumu diagramma ir līdzeklis, kas ļauj attēlot funkcionālās prasības, bet biznesa procesu diagrammas ļauj pārbaudīt atbilstošās lietošanas gadījumu diagrammas rezultātus [35]. Konceptu modelis ir paredzēts, lai attēlotu biznesa procesu notikumu datu struktūras. Mijiedarbības diagrammas tiek izstrādātas katram lietošanas gadījumam, un tās veido no realizācijas scenārijiem vai apakšprocesu aktivitāšu virknēm. Mijiedarbību atspoguļošanai nepieciešamie objekti tiek izgūti no konceptu modeļa. Klašu diagramma šādā gadījumā attēlo programmatūras struktūru, kurā ir apskatītas tikai tās klases, kuru objekti tiek izmantoti, lai realizētus lietošanas gadījumus. Shematiski tas ir atspoguļots 5. attēla kreisajā pusē.

Analizējot divpusložu pieejas transformācijas, kas realizē pārejas no viena modeļa diagrammas elementiem uz citiem, var secināt, ka lietošanas gadījumu diagramma nav pietiekoši formāla. Formalitātes problēma ir analizēta un izvērsta vairākos pētījumos [11],

[50], [52], [36], [41] [6], [49], [10]. Viena no modifikācijām, ko promocijas darba autore piedāvā ieviest divpusložu modelī klašu diagrammas ģenerēšanai, ir izvairīties no lietošanas gadījumu diagrammas izmantošanas, aizvietojot to ar citu modeli, kurš varētu kalpot par pamatu mijiedarbības diagrammām. 5. attēlā ir parādītas abas divpusložu modeļa lietošanas iespējas – oriģinālā (kreisajā pusē) un autores piedāvātā (labajā pusē). 5. attēla labajā pusē ir redzamas pārejas starp diagrammām atbilstoši ieviestām modifikācijām, neizmantojot lietošanas gadījumu diagrammas, bet izmantojot biznesa apakšprocesu modeli un pārejas palīgmodeli.

Ir veikta divpusložu modeļa lietošanas analīze, kurā ir apskatīts, kā pielietot divpusložu metodi zināšanu arhitektūras izstrādei [41], [42]. 6. attēls ilustrē autores spriedumus.



6. att. Divpusložu modeļa izmantošana klašu diagrammas ģenerēšanai

Autore ir izvirzījusi hipotēzi, ka sadarbību diagrammas (grafs G3 6. attēlā) objekti ir līdzīgi biznesa procesu un biznesa apakšprocesu diagrammas (grafs G1 6. attēlā) objektiem, kuru datu struktūras (modelis M1 6. attēlā) ir definētas, izmantojot notikumus starp biznesa procesiem [38], [42]. Tiek piedāvāts risinājums biznesa procesu diagrammu konstruēšanai, izmantot objektu sadarbību diagrammas. Lai atvieglotu pāreju no biznesa procesa objektiem uz sadarbību diagrammas objektiem, ir ieviests pārejas palīgmodelis (grafs G2 6. attēlā). Grafs G2 ir iegūts no biznesa procesa modeļa un pārveidots par sadarbību diagrammu. Izmantojot pārejas palīgmodeli un sadarbības diagrammu [42] ir iespējams pārveidot divpusložu modeļa elementus par klašu diagrammas elementiem (modelis M2 6. attēlā).

Pārejas palīgmodelis ir ģenerēts no biznesa apakšprocesu modeļa, izmantojot grafu transformācijas metodes. Gan apakšprocesu, gan pārejas palīgmodelis ir apskatīti kā grafi, un ir pielietota grafu transformācija [15], kas atspoguļota 6. attēlā, kas rāda klašu diagrammas elementu iegūšanas kopēju virzienu:

1) apakšprocesu modeļa virsotnes tiek transformētas par pārejas palīgmodeļa lokiem;
2) apakšprocesu modeļa loki tiek transformēti par pārejas palīgmodeļa virsotnēm.

Detalizētāki klašu diagrammas elementu iegūšana tiek aprakstīta un ilustrēta 3.nodaļā.

Apakšnodaļā 2.5. dots kopsavilkums par otrajā nodaļā iegūtajiem pētījuma rezultātiem. Divpusložu pieeja ir analizēta, ņemot vērā MDA prasības modeļu transformācijām. MDA pieejā ir uzsvērts, ka visas transformācijas starp avota un mērķa modeļiem ir jāveic, izmantojot formālus transformācijas likumus. Analizējot divpusložu pieejā realizētās transformācijas, ir konstatētas šādas nepilnības:

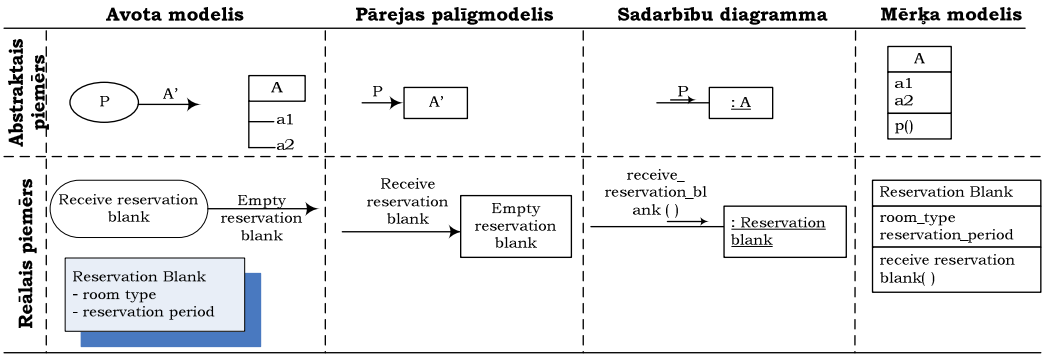
- transformācijas lietošanas gadījumu modelī nevar formalizēt, jo lietošanas gadījumu definēšana ir neformāla, kā arī neeksistē metodes lietošanas gadījumu atpazīšanai un pārbaudīšanai;
- transformācijai no lietošanas gadījumu diagrammas elementiem un scenārija soļiem uz secību diagrammu elementiem ir nepieciešama analītiķa piedalīšanās, jo scenāriju katram lietošanas gadījumam var uzrakstīt tikai balstoties uz sistēmas aprakstu. Arī informāciju par objektu savstarpējo saistību ir iespējams izgūt no scenārijiem ar cilvēka līdzdalību.

Promocijas darba autore piedāvā lietošanas gadījumu diagrammas konstruēšanu aizvietot ar apakšprocesu diagrammas transformāciju uz pārejas palīgmodeļiem. Pāreja no viena modeļa uz citu notiek bez papildus informācijas pievienošanas no ārējas vides un sistēmas analītiķa puses. Biznesa procesu diagrammas tiek piedāvāts transformēt par sadarbību diagrammām, kur, lai iegūtu objektus un klases, izmanto definēto datu struktūru. Ir izvirzīta hipotēze, ka procesu un notikumu secība ļauj sadalīt sistēmā esošos procesus starp objektiem kā metodes. Tas atrisina objektu savstarpējās saistības noteikšanas problēmu. Ieejošo un izejošo datu plūsmu skaits var tikt izmantots par parametru, definējot attiecības starp klasēm. Pārejas palīgmodelis darbā ir izmantots, lai izskaidrotu piedāvāto hipotēzi.

3. nodaļa. Piedāvātās metodes detalizēta izpēte

Darba 3. nodaļā ir izstrādāta un detalizēti aprakstīta divpusložu modeļa lietošanas modifikācija klašu diagrammas ģenerēšanai. Divpusložu modelis ir papildināts ar trīs jauniem elementiem: apakšprocesu modeli, pārejas palīgmodeli un sadarbību diagrammu. Izmantojot praktisku piemēru, ir apskatīts arī jaunā divpusložu modeļa pielietojums. Veicot pārbaudi, ir mēģināts definēt klašu diagrammas elementus no visiem iespējamiem biznesa apakšprocesu un datu plūsmu kombināciju variantiem.

Darbā izstrādātas vienkāršas transformācijas piemērs ir atspoguļots 7. attēlā.



7. att. Divpusložu modeļa elementu transformācijas piemērs

Ir attēlota pāreja no divpusložu modeļa fragmenta uz klašu diagrammas fragmentu, definējot iespējamos klašu diagrammas elementus. 7. attēls ir sadalīts četrās kolonās un divās rindās. Attēlā augšējā daļā ir atspoguļots abstrakts piemērs transformācijai no divpusložu modeļa fragmenta uz klašu diagrammas fragmentu. Attēlā tiek izmantoti šādi apzīmējumi:

- P atbilst biznesa procesam;
- A' ir no procesa izejošs notikums;

- A ir notikumam definētā datu struktūra;
- a1 un a2 ir datu struktūras īpašības;
- p() ir iegūtas klases operācija, saņemta no procesa P

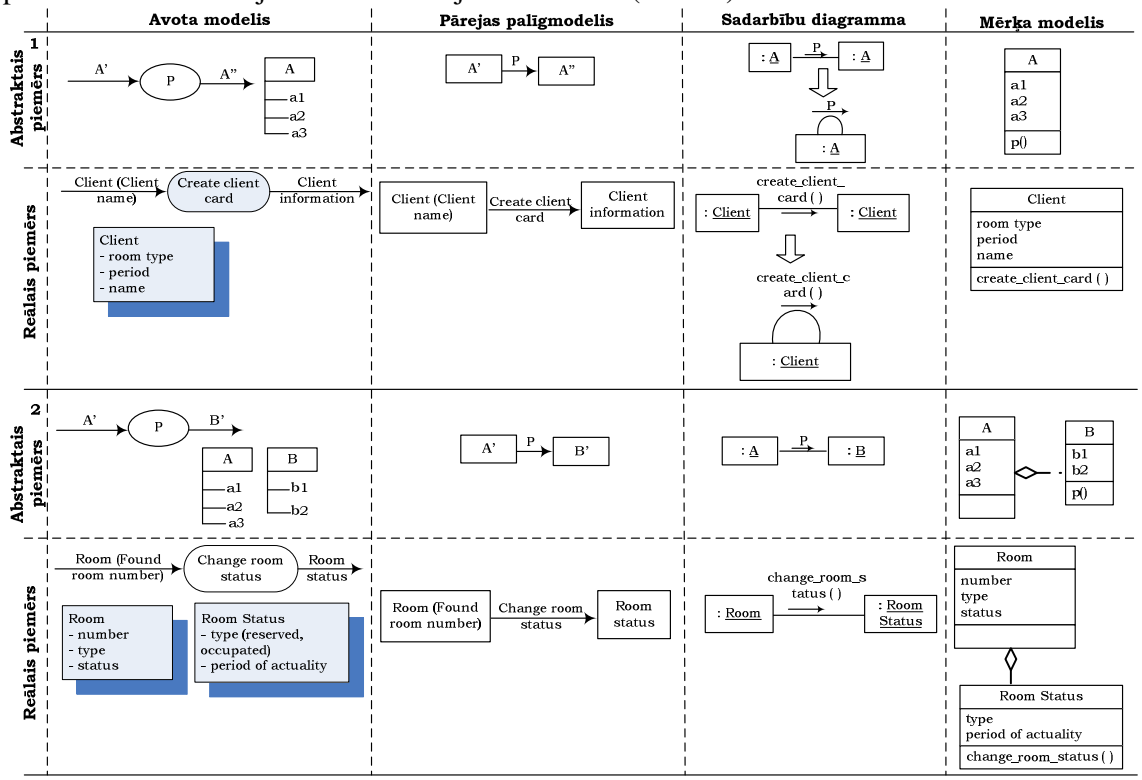
Otrajā rindā jeb apakšējā daļā ir atspoguļota abstraktajam piemēram atbilstošā diagramma un starpmodeļu fragments no praktiskās realizācijas piemēra. Attēla pirmajā kolonā ir apskatīts avota modelis, kas atbilst biznesa procesu un konceptu modeļa fragmentam. Otrajā kolonā ir attēlots pārejas palīgmodelis, kas atbilst pārejas palīgmodelim. Trešajā kolonā ir sadarbību diagramma, kas atbilst avota modelim, bet ceturtajā kolonā ir mērķa modelis, kas atbilst klašu diagrammas fragmentam.

Ņemot vērā ieejošo un izejošo biznesa procesu notikumu, ir definētas šādas notikumu kombinācijas:

- biznesa process ar vienu ieeju un vienu izeju;
- biznesa process ar vairākām ieejām un vienu izeju;
- biznesa process ar vienu ieeju un vairākām izejām;
- biznesa process ar vairākām ieejām un vairākām izejām.

Promocijas darba apakšnodaļas 3.2. - 3.5. ir veltītas iespējamo kombināciju izpētei. Atbilstoši divpusložu modelim, kas ir izveidots viesnīcas istabas rezervēšanas problēmas videi, ir ilustrēti gan abstrakti, gan reāli piemēri.

Apakšnodaļā 3.2. ir izstrādāti un aprakstīti iespējamie gadījumi situācijai, kad biznesa procesam ir viens ieejošs un viens izejošs notikums (8.attēls).

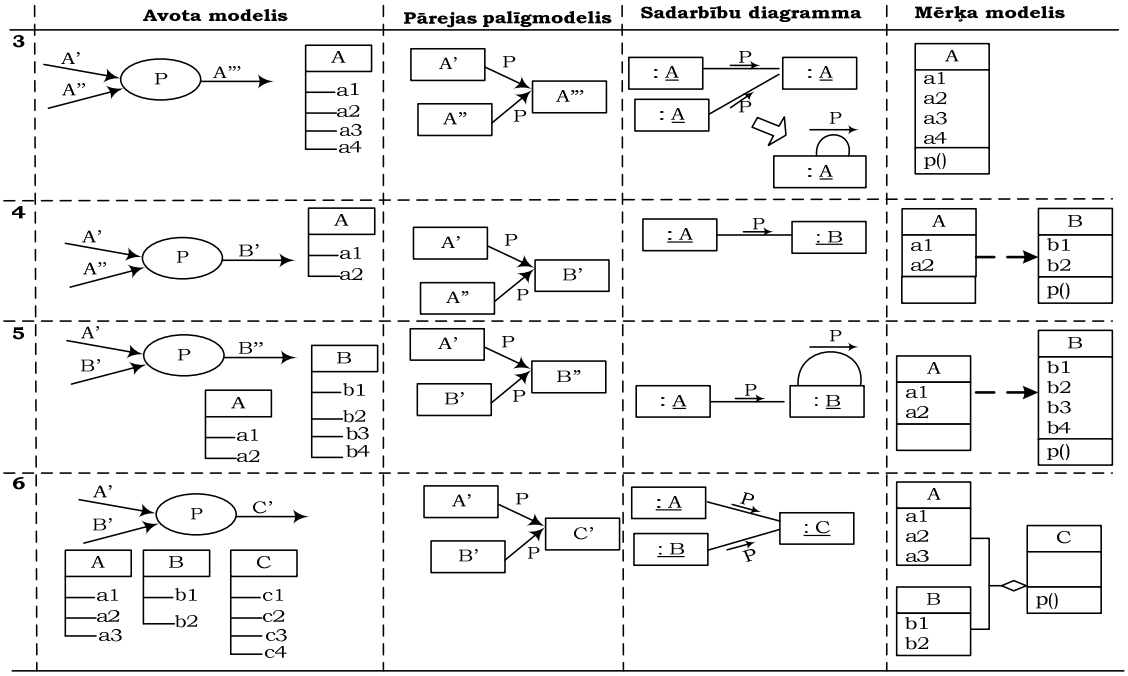


8. att. Biznesa procesu transformācijas ar vienu ieeju un vienu izeju

Šādā gadījumā ir iespējamas divas kombinācijas. Pirmajā kombinācijā ieejošās un izejošās datu plūsmas tipi ir vienādi (apzīmēti ar A' un A''), skatīt 8. attēla daļu, kas ir atzīmēta ar skaitli 1. Otrajā kombinācijā ieejošās un izejošās datu plūsmas tipi ir atšķirīgi, skatīt 8. attēla 2. transformācijas gadījumu.

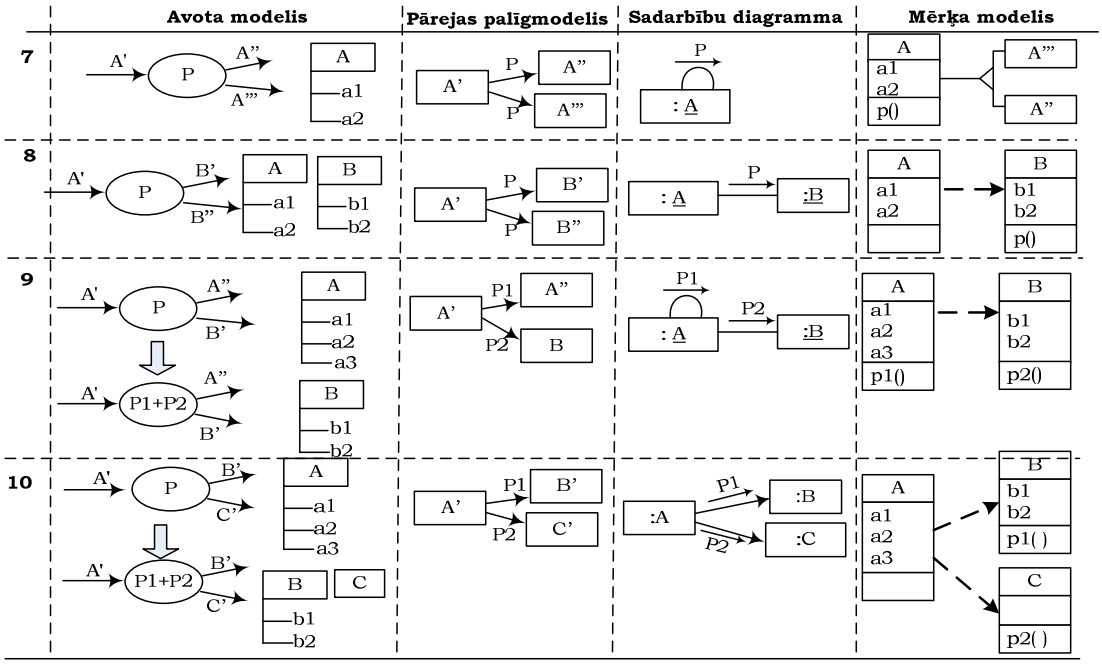
Apakšnodaļā 3.3. ir izstrādāti un aprakstīti biznesa procesi ar vairākām ieejām un vienu izeju. Šādā situācijā ir vairāk kombināciju, un tās ir atspoguļotas 9. attēlā, kur atšķirībā no 8.

attēla ir doti tikai abstrakti piemēri. Reālie piemēri, kas atbilst abstraktajiem, ir apskatīti promocijas darbā.



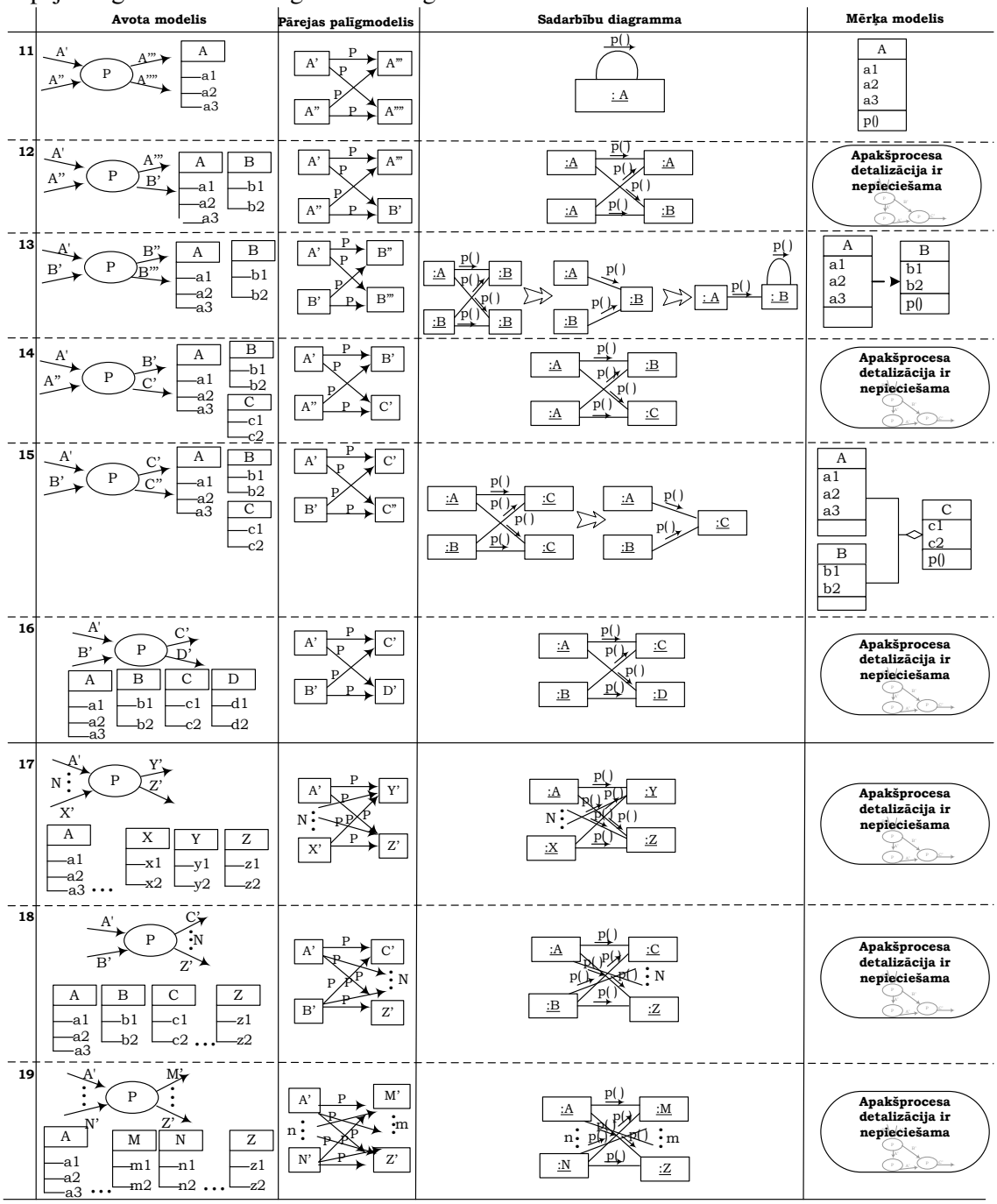
9. att. Biznesa procesu transformācijas ar vairākām ieejām un vienu izeju

Apakšnodaļā 3.4 ir izstrādātas un aprakstītas transformācijas biznesa procesiem ar vienu ieeju un vairākām izejām. Gadījumam, kad biznesa procesam ir viena ieeja un vairākas izejas, ir iespējamas daudz dažādas kombinācijas gan ar vienādiem, gan ar dažādiem datu tipiem ieejā un izejā. Arī šīs kombinācijas ir attēlotas tikai ar abstraktiem piemēriem (skatīt 10. attēlu), un reālie piemēri ir detalizēti apskatīti promocijas darbā.



10. att. Biznesa procesu transformācijas ar vienu ieeju un vairākām izejām

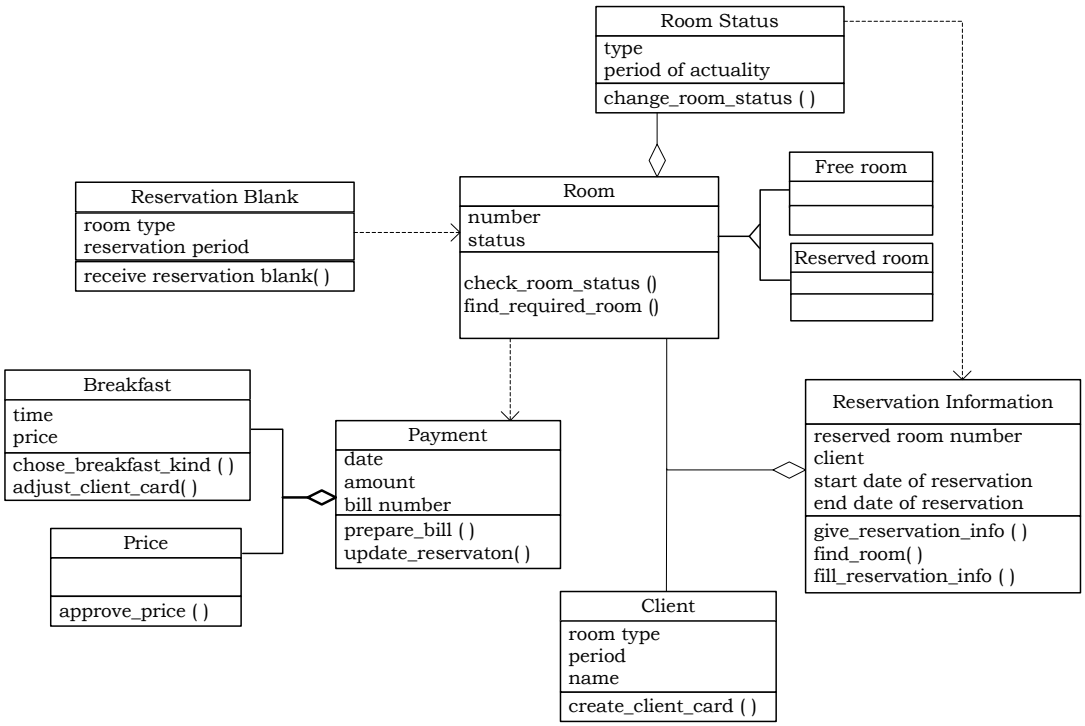
Apakšnodaļā 3.5. ir izstrādāti un aprakstīti biznesa procesi ar vairākām ieejām un vairākām izejām. Šādam gadījumam ir raksturīgs tas, ka dažas ieejošo un izejošo loku kombinācijas ar vienādiem vai dažādiem datu tipiem ir iespējams transformēt par darbā iepriekš definētajām kombinācijām. Ja nav iespējama transformācija, tad ir nepieciešama apakšprocesu sadalīšana, veicot dziļāku detalizāciju. Kombinācijas shematiskā veidā ir atspoguļotas 11. attēlā. Definētajām transformācijām dažos gadījumos ir šāds ierobežojums: „JA nav iespējas definēt klases, klašu saistību, vai operāciju piederību, TAD biznesa process ir jāsadala, konstruējot apakšprocesu diagrammas, līdz tiek iegūti gadījumi, kuriem ir iespējams ģenerēt klašu diagrammas fragmentus".



11. att. Biznesa procesu transformācijas ar vairākām ieejām un vairākām izejām

Apakšnodaļā 3.6. ir atspoguļots mērķa modelis (klašu diagramma), kas ir uzģenerēts viesnīcas istabas rezervēšanas problēmas videi, izmantojot definētās modeļu transformācijas no izveidotā divpusložu modeļa. Klašu diagrammā ir definētas klases, īpašības, operācijas un klašu saistības, kas ir iegūtas transformācijās no divpusložu modeļa elementiem.

Iegūtā klašu diagramma, kas aprakstīta promocijas darba 3. nodaļā un atbilst reālam piemēram ir atspoguļota 12. attēlā. Diagrammā ir attēlotas viesnīcas istabu rezervēšanas problēmas videi atbilstošās klases, attiecības starp tām, īpašības un operācijas.



12. att. Klašu diagramma viesnīcas istabu rezervēšanas sistēmai

Apakšnodaļā 3.7. ir aprakstīts, kā pārveidot avota modeļa elementus par mērķa modeļa elementiem, izmantojot pieejā definētās modeļu transformācijas. Promocijas darbā ir ievietota detalizēta pārveidošanas tabula, kurā ir parādīta divpusložu modeļa elementu atbilstība UML valodas notācijā veidotas klašu diagrammas elementiem. Apskatot elementu pārveidošanas tabulu var secināt, ka ne visus klašu diagrammas elementus var iegūt no divpusložu modeļa, un, lai definētu dažus elementus, ir nepieciešami papildus pētījumi. Piemēram, nav definēts, kā noteikt operāciju argumentus, klašu stereotipus, ierobežojumus. Ir realizēts promocijas darba galvenais uzdevums – izstrādātā pieeja ļauj no divpusložu modeļa, kas ir definēts biznesa procesu un datu struktūras terminos, ģenerēt klašu diagrammas kopējo struktūru, definēt klašu savstarpējo sasaisti un iekapsulēt klasēm atbilstošās metodes. Pirms autores ieteiktās metodes izstrādes minētās darbības bija jāveic sistēmanalītiķim.

Apakšnodaļā 3.8. ir apkopoti trešajā nodaļā ieteiktie risinājumi un to rezultāti. Viesnīcas istabu rezervēšanas sākotnējais biznesa procesu un konceptu modelis tiek lietots, lai veiktu pētījumu, kurā autore demonstrē modeļa transformācijas iespējas. Izmantojot pārejas palīgmodeli un sadarbību diagrammu un tiek ģenerēti klašu diagrammas pamata elementi. Praktiskais eksperiments, ar visām iespējamām modeļu kombinācijām, apstiprina izvirzīto hipotēzi, ka divpusložu modeli var izmantot klašu diagrammas ģenerēšanai piedāvātajā veidā. Divpusložu pieeju lietojot pārāk vispārīgi definētam biznesa procesu modelim, dažās kombinācijās ir konstatēti arī ierobežojumi, bet tas nenozīmē, ka nevar pielietot transformācijas. Šādā gadījumā ir nepieciešams detalizētāks avota modelis.

Izveidotā klašu diagramma nav pilnīga, jo nav nodefinēti operāciju argumenti, atribūtu tipi un klašu stereotipi, taču tā iekļauj lielāko daļu pamatelementu un tiek iegūta formālā veidā no divpusložu modeļa. Formālais diagrammas iegūšanas veids nodrošina, ka visa informācija, kura atradās sākotnējā divpusložu modelī, ir nodota klašu diagrammai. Ņemot vērā faktu, ka daudzas organizācijas lieto biznesa modelēšanas principus darbību analīzei un optimizācijai, var apgalvot, ka pieejai nepieciešamā sākotnējā informācija ir pieejama un nav vajadzīgi papildus resursi, kas var palielināt programmatūras sistēmas izstrādes izmaksas.

4. nodaļa. Metodes pielietojuma eksperimenti

Lai pārliecinātos par izstrādātās pieejas lietojamību un neatkarību no biznesa procesu un datu struktūras modelēšanas notācijas un to atbalstošiem rīkiem, tā ir jāpārbauda ar dažāda veida piemēriem. Darba 4. nodaļā ir aprakstīta pētījuma rezultātu pārbaude, kas tiek realizēta, lai iegūtu klašu diagrammu ar automatizēta rīka palīdzību.

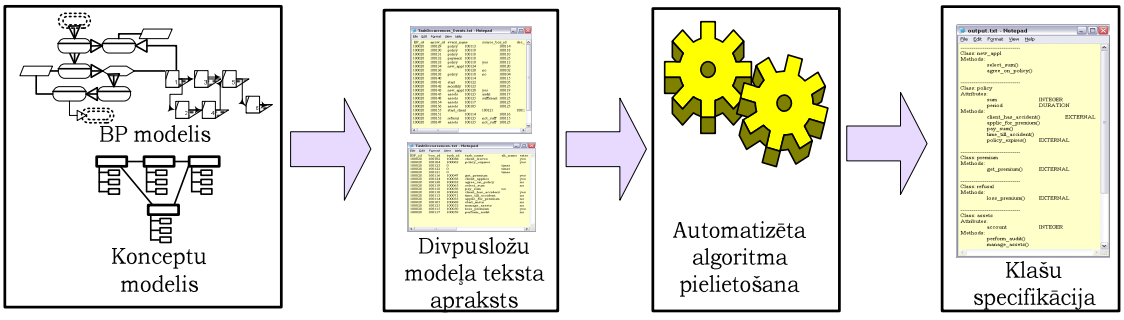
Apakšnodaļā 4.1. ir piedāvāta koncepcija rīkam, kas atbalsta izstrādāto pieeju. Lai būtu iespējama transformācija no $PIM_{initial}$ uz $PIM_{refined}$, rīkam ir jāveido klašu diagrammas elementi no biznesa procesu, apakšprocesu un konceptu modeļa. Lai divpusložu modelis būtu lietojams klašu diagrammas ģenerēšanai, modeļiem jābūt automātiskai lietošanai piemērotā veidā, piemēram, teksta faila veidā.

Klašu struktūras definēšana ir veikta vadoties pēc shēmām, kas ir aprakstītas 3. nodaļā. Biznesa procesu diagrammas fragmenti, kas ļauj ģenerēt klašu diagrammas elementus, izmantojot arī datu struktūras, tiek apstrādāti, lietojot 3. nodaļā definētas transformācijas. Gadījumā, ja kādam biznesa procesam nav bijis iespējams definēt klasi, vai kādai klasei nav bijis iespējams definēt operāciju vai sasaisti, tad tiek izvadīts paziņojums, kas pieprasa avota procesa papildus detalizāciju.

Rīka darbības ir iespējams iedalīt četrās daļās:

1. klašu meklēšana;
2. klašu īpašību definēšana;
3. klašu operāciju definēšana;
4. klašu savstarpējo attieksmju definēšana.

Piedāvātās metodes lietošanas darbība ir shematiski atspoguļota 13. attēlā. Pirmais solis ir divu modeļu konstruēšana (divpusložu modelis), lietojot tādu rīku, kurš ļauj ģenerēt šiem modeļiem automātiski lietojamas specifikācijas (piemēram, teksta faila veidā). Pēc tam ir jāģenerē sākotnējais modelis teksta veidā. Kad teksta faili ir gatavi, var pielietot izstrādāto rīku un iegūt klašu specifikāciju.



13. att. Metodes lietošanas darbības algoritms

Izejas (mērķa) klašu struktūra tiek definēta kā teksta faili ar šādu struktūru:

Klase: `class_name`

Īpašības:
```
attribute_name1 ( )
attribute_name2 ( )
attribute_name3 ( )
```

Operācijas:
```
method_name1 ( )
method_name2 ( )
method_name3 ( )
```

Attieksmes:
```
Associate (vai associated by): class_name
Aggregate (vai aggregated by): class_name
Generalize (vai generalized by): class_name
```

Apakšnodaļā 4.2. ir aprakstīts, kā pielietot automatizētu rīku viesnīcu istabas rezervēšanas problēmas videi, kura tika aprakstīta un izpētīta 3. nodaļā. GRADE [13] ir viens no CASE rīkiem, kas atbalsta biznesa procesa un konceptu modeļu (avota modeļa) konstruēšanu un ļauj ģenerēt šo modeļu teksta failus. GRADE spēj ģenerēt failu kopu, kura apraksta visus GRADE rīkā izveidotos modeļus. Lai veiktu operācijas, tiek lietoti šādi faili:

- 'TaskOccurrences.txt' apraksta visus biznesa procesu diagrammas procesus;
- 'Datatypes.txt' satur informāciju par datu struktūru: datu objektiem un to īpašībām;
- 'TaskOccurrences_Events.txt' nodrošina sasaisti starp datu struktūru un biznesa procesu modeļiem.

Biznesa procesu diagramma ir izveidota GRAPES [19] notācijā ar CASE rīku GRADE. Teksta faili ir ģenerēti turpmākai lietošanai derīgā formātā, un ir veidoti, pamatojoties uz ģenerēto klašu diagrammas struktūru, kas promocijas darbā ir apskatīta grafiskā veidā.

Apakšnodaļā 4.3. ir aprakstīts, kā pielietot izstrādāto metodi viesnīcas istabu rezervēšanas problēmas videi, bet modelējot IDEF0 [19] notācijā, lai pierādītu pieejas neatkarību no biznesa procesu notācijas. Šo notāciju atbalsta rīks BPWin, un biznesa procesu diagramma tiek konstruēta. Tā kā BPWin neatbalsta konceptu un/vai objektu modeļa konstruēšanu, 4.3. apakšnodaļā iegūtajai klašu diagrammai nav īpašību, bet arī automātiskā metožu sasaistes sadalīšana starp sistēmas klasēm ir būtisks ieguvums programmatūras izstrādē.

Apakšnodaļā 4.4. ir aprakstīts, kā pielietot pieeju citai problēmas videi, lai pierādītu pieejas neatkarību no problēmvides. Sākotnējā modeļa konstruēšanai tiek lietots rīks GRADE. Lai aprobētu piedāvāto pieeju, neizmantojot autores izstrādāto piemēru, bet neatkarīgi izveidotu divpusložu modeli, tiek lietots GRADE rīkā iebūvētais piemērs – apdrošināšanas sistēma. Biznesa procesu diagramma un datu struktūra tiek iegūta no GRADE teksta failu veidā un tālāk no informācijas, kas atrodas failos tiek ģenerēta klašu diagramma.

Apakšnodaļā 4.5. ir apkopoti iegūtie secinājumi par 4. nodaļā veiktajiem eksperimentiem. Veiktie pētījumi pierādīja izstrādātās pieejas neatkarību no divpusložu modeļa notācijas un to atbalstoša rīka. Vienīgais papildus nosacījums ir, lai notācija vai rīks ļauj modelēt elementus, kas ir nodefinēti, kā obligāti avota modelim.

Izstrādātā pieeja, kas izmanto divpusložu modeli klašu diagrammas ģenerēšanai, var tikt pielietota biznesa procesu un konceptu modeļiem, kas konstruēti jebkurā notācijā un jebkurā rīkā. Metodes ierobežojums, kas ir saistīts ar procesu papildus detalizācijas nepieciešamību,

rīkā ir atrisināts šādi: tiek definētas transformācijas visiem modeļa fragmentiem un par tiem procesiem, kam nav iespējams uzreiz noteikt atbilstošās klases, tiek izvadīts paziņojums ar procesa nosaukumu un prasību sadalīt šo procesu sīkāk.

Darba rezultāti un secinājumi

Promocijas darbā tika izvirzīts **mērķis**, pamatojoties uz esošo sistēmas modeli, piedāvāt formālu metodi klašu diagrammas konstruēšanai. Darbā tika izstrādāta pieeja platformneatkarīga modeļa konstruēšanai, kas apmierina MDA pamatprincipu – programmatūras izstrādi balstīt uz formālām transformācijām starp modeļiem, kas definēti UML notācijā [39] .

Izvirzītā mērķa sasniegšanai tika realizēti šādi **uzdevumi**:

- aprakstīti modeļvadāmas arhitektūras pamatprincipi un identificēti problemātiskās vietas platformneatkarīga modeļa konstruēšanā;
- analizēti eksistējošās platformneatkarīga modeļa konstruēšanas metodes;
- izvēlēta viena metode, kuru izmantot tālāk, lai realizētu klašu diagrammas ģenerēšanu;
- izstrādāta metode platformneatkarīga modeļa konstruēšanai, kas ir pietiekami atbilstošs platformspecifiska modeļa ģenerēšanai;
- piedāvātā metode tika izskaidrota ar abstraktiem un reāliem piemēriem;
- izstrādātās metodes iespējas dažādās problēmas vidēs un dažādos sākotnējā modeļa konstruēšanas rīkos ir pārbaudītas.

Promocijas darba galvenais rezultāts *ir izstrādātā metode klašu diagrammas ģenerēšanai no sākotnējā sistēmas attēlojuma divpusložu modelī, kur sistēmas aspekti ir atspoguļoti biznesa procesu un konceptu modeļa veidā. Izstrādātā metode ļauj atrisināt šobrīd aktuālo problēmu, kas ir saistīta ar neiespējamību precīzi un nepārprotami attēlot problēmvides zināšanas klašu diagrammas veidā, lietojot objektorientētu programmatūras izstrādes tehnoloģiju. Klašu diagramma programmatūras izstrādes procesā kalpo par pamatu sistēmas objektorientētai realizācijai.*

Promocijas darba ***svarīgākie rezultāti*** ir:

1. Izstrādātā metode izmanto sākotnējo biznesa procesu un konceptu modeli platformneaktarīga modeļa izveidei, tāpēc ir iespējams izmantot organizācijā esošo biznesa procesu modeli programmatūras sistēmas izstrādē.
2. Pētījuma gaitā ir atklāti lietošanas gadījumu diagrammas ierobežojumi un trūkumi, un atrasts paņēmiens, kā aizvietot lietošanas gadījumu diagrammu ar formālām modeļu transformācijām lietderīgu analogu.
3. Izstrādātā metode ļauj formāli iegūt sistēmas statisko modeli UML klašu diagrammas veidā, kas tiek izmantots tālākai sistēmas modelēšanai un realizēšanai.
4. Ir izstrādāta metode klašu diagrammas ģenerēšanai no sākotnējā biznesa procesu un konceptu modeļiem. Šāda klašu diagrammas veidošana ļauj izvairīties no informācijas zudumiem manuālās transformācijas laikā un kļūdām analītiķu darbā, ko izraisa cilvēciskais faktors.
5. Klašu diagrammas formālā ģenerēšana ļauj paaugstināt abstrakcijas līmeni, lai būtu iespējams pāriet no platformneatkarīga uz platformspecifisku modeli. Iegūtais rezultāts ir svarīgs, jo abstrakcijas līmeņa paaugstināšana, ir viena no MDA pamatidejām.

Iegūtie teorētiskie rezultāti ir realizēti praktiski trīs piemēros:

1. Viesnīcu istabu rezervēšanas problēmas videi, kur biznesa procesu diagramma konstruēta, izmantojot GRAPES notāciju un rīku GRADE.
2. Viesnīcu istabu rezervēšanas problēmas videi, kur biznesa procesu diagramma konstruēta, izmantojot IDEF0 notāciju un rīku BPWin.
3. Apdrošināšanas sistēmas problēmas videi, kur biznesa procesu diagramma konstruēta, izmantojot GRAPES notāciju un GRADE rīku. Apdrošināšanas sistēmas divpusložu modelis ir iegūts no GRADE rīkā iebūvētā piemēra.

Analizējot rezultātus, kas radušies veicot pieejas pārbaudi ar praktiskiem piemēriem, ir iegūti šādi secinājumi:

- automatizēta rīka darbības rezultātā ir iegūta klašu struktūras specifikācija, ko ir iespējams izmantot klašu diagrammas veidošanai rīkā ar UML vai MDA atbalstu. Klašu diagrammu tālāk izmanto MDA modeļu transformācijās;
- visos realizētajos piemēros automātiski iegūtās klašu diagrammas ir salīdzinātas ar manuāli izstrādātām klašu diagrammām, un iegūtie rezultāti ir apstiprinājums tam, ka ģenerētās diagrammas var tikt lietotas sistēmas realizācijai;
- no eksperimentiem, kas veikti ar metodi dažādās problēmvidēs, var secināt, ka metode ir neatkarīga no lietojumsistēmas, kam ir jāveido klašu diagramma. Savukārt, eksperimenti, kas realizēti izmantojot metodi dažādās notācijās, liecina par metodes neatkarību no izvēlētās modelēšanas valodas un rīkiem, kas atbalsta šo valodu. Ir jāņem vērā, ka valodā un rīkā ir nodrošināta iespēja attēlot visus divpusložu modeļa elementus.

Tālākie pētījumu virzieni var būt saistīti ar:

- klašu metožu argumentu definēšanu;
- klašu stereotipu un interfeisu definēšanu;
- sistēmas uzvedības definēšanas aspektiem, kas var tikt izteikti ar aktivitāšu un stāvokļu diagrammu konstruēšanu;
- izstrādātās metodes vizualizēšanu.

Izstrādāto metodi ir rekomendēts lietot jaunu programmatūras sistēmu izstrādē un eksistējošo programmatūras sistēmu modificēšanā neatkarīgi no sistēmas sarežģītības. Klašu diagramma ir sistēmas struktūras atspoguļojums, un to var izmantot sistēmas arhitektūras projektēšanai, programmatūras koda ģenerēšanai, testēšanas gadījumu projektēšanai.

IZMANTOTĀ LITERATŪRA

[1] Alhir S, Understanding the Model Driven Architecture, Methods & Tools: An international software engineering digital newsletter published by Martinig & Associates, / Internet. - *home.earthlink.net/~salhir/UnderstandingTheMDA.PDF*

[2] Ambler S.W., Approaches To Agile Model Driven Development (AMDD) / Internet. - *http://www.agilemodeling.com/essays/amddapproaches.htm#manual*

[3] Ambler, S.W. The Elements of UML Style// Cambridge University Press.- 2003

[4] ARIS Toolset / Internet : *http://www.ids-scheer.com/*

[5] Balzer, R. A 15 Year Perspective on Automatic Programming// IEEE Transactions on Software Engineering Vol 11, No 11.- 1985.

[6] Berard E. V., Be Careful With "Use Cases", The Object Agency, Inc. August 23, 1998.

[7] Brown A. W. Model driven architecture: Principles and practice// Software and System Modeling 3(4), 2004.- pp. 314-327.

[8] Ceponiene, L., Nemuraite, L. Transformations of UML Diagrams for Reconciliation of Requirements// Proceedings of 13th International Conference on Information Systems Development – Advances in Theory, Practice and Education, 2005. – pp .289-302.

[9] Exertier D., Langlois B., Normand V. MASTER: WP2 MDE Foundation// UML Specialization Approach, 2003/ Internet.- *http://modeldrivenarchitecture.esi.es/mda_publicDocuments.html*

[10] Ferg S., What's Wrong with Use Cases? / Internet.- *http://www.ferg.org/papers/ferg--whats_wrong_with_use_cases.html*
[11] Firesmith D. G. Modeling the Dynamic Behavior of Systems, Mechanisms, and Classes with Scenarios// Report on Object-Oriented Analysis and Design (ROAD), SIGS Publications, Vol. 1, No. 2.- New York: New York.- 1994, pp. 32-36.
[12] Frankel D. Model Driven Architecture: Applying MDA to Enterprise Computing, John Wiley & Sons, 2003.
[13] GRADE tools, GRADE Development Group / Internet.- *http://www.gradetools.com/*
[14] GRADE Business Modeling, Language Guide// INFOLOGISTIK GmbH.- 1998.
[15] Grundspenkis J. Causal Domain Model Driven Knowledge Acquisition for Expert Diagnosis System Development // Kaunas: Kaunas University of Technology Press.- 1997.
[16] Hendrix S. Integrating Computation Independent Business Modeling Languages into the MDA with UML2.- 2003. / Internet. - *http://www.omg.org/docs/ad/03-01-32.pdf*
[17] Jacobson I., Booch G., Rumbaugh J., The Unified Software Development Process// Addison-Wesley.- 2002.
[18] Kalnins A., Barzdins J., Podnieks K. Modeling languages and tools: state of the art. – Proceedings of Second International Conference on Simulation, Gaming, Training and Business Process Reengineering, Riga, 2000, pp.211-214.
[19] Kalnins. A, Vitolins. V Modeling business. - Proceedings of International Conference on Modelling and Simulation of Business systems, Vilnius, 2003, pp. 215-219.
[20] Kalnins A., Kalnina D., Kalis. A, Comparison of Tools and Languages for Business Process Reengeneering. - Proceedings of the Third International Baltic Workshop on Databases and Information Systems, Riga, 1998, pp. 24-38.
[21] Kent S. Model driven engineering. In Proceedings of IFM2002, volume 2335 of LNCS. Springer-Verlag, 2002.
[22] Kleppe A. MCC: A model transformation environment. In: Proceedings of the ECMDA-FA 2006, LNCS 4066, A. Rensink and J.Warmer (Eds.), Springer-Verlag, 2006, pp. 173-187.
[23] Kleppe A., Warmer J., Bast W. MDA Explained: The Model Driven Architecture – Practise and Promise, Addison Wesley, 2003, p. 192.
[24] Krogstie, J. Integrating Enterprise and IS Development Using a Model Driven Approach. Proceedings of 13th International Conference on Information Systems Development – Advances in Theory, Practice and Education (ISD 2005). Springer Science+Business media, Inc., 2005, pp.43-53.
[25] Larman C. Applying UML And Patterns: An Introduction To Object-Oriented Analysis And Design, Prentice Hall, New Jersey, 2000.
[26] Mcneile A., MDA: The Vision With The Hole?, 2003 / Internet. - *www.metamaxim.com*
[27] MDA Guide Version 1.0.1 / Internet. - *http://www.omg.org/docs/omg/03-05-01.pdf*
[28] Mellor S. J., Kendall Scott, Axel Uhl and Dirk Weise. "MDA Distilled, Principles of Model_driven Architecture. Addison-Wesley, 2004.
[29] Mellor, S.J., Balcer, M.J. Executable UML. A Foundation for Model-Driven Architecture, Addison-Wesley, Boston, 2002, p.368.
[30] Nemuraite, L., Paradauskas B. From Use-Cases to Well Structured Conceptual Schemas Proceedings of 13th International Conference on Information Systems Development – Advances in Theory, Practice and Education (ISD 2005). Vasilecas O., Caplinskas A., Wojtkowski W., Wojtkowski W., G., Zupancic J., Wrycza S. (Eds). Springer Science+Business media, Inc. 2005, pp. 303-313.
[31] Nikiforova O. General Framework For Object-Oriented Software Development Process, // Scientific Proceedings Of Riga Technical University, Computer Science, Applied Computer Systems, 3rd Thematic Issue, Riga, Latvia, 2002, pp. 132-144.
[32] Nikiforova O., Kirikova M. Enabling Problem Domain Knowledge Transformation During Object Oriented Software Development, // Conference Of Information System Development, ISD'2003, Melbourne, Australia, 25-27 August 2003, pp. 12.
[33] Nikiforova O., Kirikova M., Pavlova N. Two-hemisphere driven approach: Application for Knowledge Modeling. In: Proceedings of the Seventh IEEE International Baltic Conference on DB and IS (BalticDB&IS'2006), O. Vasilecas, J. Eder, A. Caplinskas (Eds.), Vilnius, Lithuania, 2006, pp. 244-250.
[34] Nikiforova O., Kirikova M., Sukovskis U. Two Hemisphere Model Driven Architecture for Knowledge Map Development in the Task of Study Program Analysis, The 46th Scientific Conference of Riga Technical University, Computer Science, Applied Computer Systems, October 13-14, Riga, Latvia, 2005, published in the 5th Series "Computer Science. Applied Computer Systems, Vol. 26, 2006, pp. 112-123.
[35] Nikiforova O., Kirikova M., Two-Hemisphere Model Driven Approach: Engineering Based Software Development, // Proceeding Of The 16th International Conference Advanced Information Systems Engineering Caise'2004, Persson A.And Stirna J. (Eds.), Lncs 3084, Springer – Verlag Berlin

Heidelberg, 2004, pp. 219 – 233.
[36] Nikiforova O., Kirikova M., Wojtkowski W., Role Of Models In Knowledge Transfer During OO Software Development, // The 15th European – Japanese Conference On Information Modeling And Knowledge Bases, Tallinn,,May 16-19, 2005, pp. 305-320.
[37] Nikiforova O., Kuzmina M., Pavlova N. Formal Development of Platform Independent Model in the Framework of MDA: Myth or Reality// Scientific Proceedings of Riga Technical University, 5th Series, Computer Science, Applied Computer Science Vol.22. – Riga: RTU, 2005, pp.42-53.
[38] Nikiforova, O., Kirikova M., Pavlova N. Principles of Model Driven Architecture in Knowledge Modeling for the Task of Study Program Evaluation// Databases and Information Systems IV, by IOS Press in the series "Frontiers in Artificial Intelligence and Applications", Vasilecas O., Eder J., Caplinskas A. (eds), 2007, pp.291-304.
[39] OMG Unified Modeling Language Specification / Internet.- *http://www.tu-chemnitz.de/wirtschaft/wi2/lehre/2001_ws/pris_1/uml_1.4.pdf*
[40] Pavlova N., Nikiforova O. An Overview of Advanced Approaches for Construction of Platform-Independent System Model// Scientific Proceedings of Riga Technical University, 5th Series, Computer Science, Applied Computer Science Vol.22. – Riga: RTU, 2005, pp. 156-168.
[41] Pavlova N., Nikiforova O. Formalization of Two-Hemisphere Model Driven Approach in the Framework of MDA// Proceedings of the 9th Conference "Information Systems Implementation and Modelling" (ISIM'06), April 24-26, 2006, Prerov, Czech Republic. – Ostrava: Jan Štefan MARQ., 2006, pp. 105-112.
[42] Pavlova N. Several Outlines of Graph Theory in the Framework of MDA// Proceeding of the 15th International Conference "Information Systems Development" (ISD'06).- Hungary, 2006 (in press).
[43] PIM to PSM mapping techniques, MASTER-2003-D5.2-V1.0-PUBLIC, 2003 / Internet. - *http://modeldrivenarchitecture.esi.es/mda_publicDocuments.html*
[44] Pokorny, T. The Model Driven Architecture: No Easy Answers. Proceedings of Simulation Conference (SimTecT), 2005. / Internet.- *http://www.consec.com.au/simtect2005/abstracts.html?confid=STT2005*
[45] Quatrany T. Visual Modeling With Rational Rose 2000 And UML. Second Edition, Addison-Wesley, 2000.
[46] Rumbaugh J. Models Through The Development Process, // Journal Of Object Oriented Programming, May 1997.
[47] Rumbaugh J. OMT: The Developing Process, // Journal Of Object Oriented Programming, No.8, 1995, pp. 14-18.
[48] Siegel, J. Developing in OMG's Model-Driven Architecture. OMG document omg/01-12-01, 2001 available at *http://www.omg.org/mda/papers.htm*
[49] Simons A., Graham I. 37 Things That Don't Work in Object-Oriented Modelling with UML// ECOOP 98 Workshop on Behavioural Semantics, Technical Report TUM-I9813, Technische Universitat Muchen, / Internet .- 1998 / Internet.- *http://citeseer.ist.psu.edu/simons98things.html*
[50] Sindre, G., Opdahl, A.LEliciting Security Requirements by Misuse Cases Technology of Object-Oriented Languages and Systems. Proceedings of the 37th International Conference on TOOLS-Pacific. Volume, Issue , 2000, pp. 120 – 131.
[51] Skersys, T., Gudas S. Class Model Development Using Business Rules, Advances in Information Systems Development. Bridging the Gap between Academia and Industry, Nilsson A.G., Gustas R., Wojtkowski W., Wojtkowski W., G., Zupancic J., Wrycza S. (eds), Volume 1, Springer Science+Business media, Inc. 2006, pp. 203-216.
[52] Syversen, E., Anda, B., Sjeberg, D.I.K. An Evaluation of Applying Use Cases to Construct Design Versus Validate Design. Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 6-9 Jan., 2003.
[53] Tkach D., Fang W., So A. Visual Modeling Technique : Object Technology Using Visual Programmimg-Addison Wesley, 1996.
[54] Ullman, J., Widom J. A First Course in Database Systems, 2nd ed., Prentice-Hall, Upper Saddle River, 2002, pp. 511.