RĪGAS TEHNISKĀ UNIVERSITĀTE
Datorzinātnes un informācijas tehnoloģijas fakultāte
Lietišķo datorsistēmu institūts

Armands ŠLIHTE
Doktora studiju programmas «Datorsistēmas» doktorants

INTEGRĒTĀ PRIEKŠMETISKĀS VIDES MODELĒŠANA: PIEEJA UN RĪKU KOPA TOPOLOĢISKĀ FUNKCIONĒŠANAS MODEĻA IEGŪŠANAI

Promocijas darba kopsavilkums

Zinātniskais vadītājs
profesors *Dr. habil. sc. ing.*
J. OSIS

RTU Izdevniecība
Rīga 2015

Šlihte A. Integrētā priekšmetiskās vides modelēšana: pieeja un rīku kopa topoloģiskā funkcionēšanas modeļa iegūšanai. Promocijas darba kopsavilkums. – R.: RTU Izdevniecība, 2015, 39 lpp.

Iespiests saskaņā ar DITF LDI padomes 2014. gada 30. jūnija lēmumu Nr. 12300-4.1/2.

Šis darbs izstrādāts ar Eiropas Sociālā fonda atbalstu projektā «Atbalsts RTU doktora studiju īstenošanai».

Darbs izstrādāts ar daļēju Valsts pētījumu programmas *SOPHIS* atbalstu līgumā Nr.10-4/VPP-4/11.

ISBN 978-9934-8260-4-7

PROMOCIJAS DARBS
IZVIRZĪTS INŽENIERZINĀTŅU
DOKTORA GRĀDA IEGŪŠANAI RĪGAS TEHNISKAJĀ
UNIVERSITĀTĒ

Promocijas darbs inženierzinātņu doktora grāda iegūšanai tiek publiski aizstāvēts 2015. gada 8. jūnijā Rīgas Tehniskās universitātes Datorzinātnes un informācijas tehnoloģijas fakultātē Rīgā, Meža ielā 1/3, 202. auditorijā.

OFICIĀLIE RECENZENTI

Profesore *Dr. sc. ing.* Mārīte Kirikova
Rīgas Tehniskā universitāte, Latvija

Profesors *Dr. sc. ing.* Artis Teilāns
Rēzeknes Augstskola, Latvija

Asociētais profesors Visente Garsija Diazs (*Vicente Garsia Diaz*)
Oviedo Universitāte, Spānija

APSTIPRINĀJUMS

Apstiprinu, ka esmu izstrādājis šo promocijas darbu, kas iesniegts izskatīšanai Rīgas Tehniskajā universitātē inženierzinātņu (vai cita) doktora grāda iegūšanai. Promocijas darbs zinātniskā grāda iegūšanai nav iesniegts nevienā citā universitātē.

Armands Šlihte …….(Paraksts)

Datums: ……

Promocijas darbs ir uzrakstīts angļu valodā, tajā ir ievads, 4 nodaļas, secinājumi, literatūras saraksts, 10 pielikumu, 52 zīmējumi un ilustrācijas, kopā 216 lappuses. Literatūras sarakstā ir 100 nosaukumi.

SATURS

IEVADS

Programmatūras inženierijas kontekstā ar terminu «priekšmetiskā vide» visbiežāk tiek apzīmēta lietojuma sfēra jeb vide, kurai tiek izstrādāta programmatūra [74]. Priekšmetiskās vides modeli izmanto par pamatu programmatūras izstrādei kādai konkrētai priekšmetiskajai videi. Šā modeļa elementi var tikt izmantoti par pamatu pirmkodam, manuāli tos transformējot vai arī izmantojot automatizētu koda ģenerāciju, kā tas tiek piedāvāts modeļu vadāmajā arhitektūrā (MDA) [41]. Priekšmetiskās vides modeli var saukt par integrētu modeļu sistēmu, kas atspoguļo uzņēmumu jeb iestādi, kurā paredzēts lietot programmatūru [89]. Turpretī priekšmetiskās vides modelēšana ir cilvēku aktivitāte, kas noved pie dažāda veida priekšmetiskās vides atspoguļojumiem, kas var būt nenoteikti (tikai cilvēku prātā) vai noteikti (izveidoti uz papīra vai kādā rīkā) [36]. Kā norādīts [74], priekšmetiskās vides analīze ir process, kurā programmatūras izstrādei nepieciešamā informācija tiek identificēta, strukturēta, pierakstīta un sakārtota turpmākai izmantošanai. Atbilstoši [58] pastāv daudz dažādu pieeju priekšmetiskās vides modelēšanai, taču tās parasti balstās uz standartu apvienošanu nevis uz matemātiski formāliem modeļiem; vienīgie izņēmumi ir Petri tīkli un topoloģiskais funkcionēšanas modelis (TFM). Formalitātes trūkums ir būtiska problēma programmatūras inženierijā, jo tas traucē izveidot formālu sasaisti starp priekšmetiskās vides modeli un programmatūras risinājumu.

Pētījuma motivācija un tēmas aktualitāte

Savos pētījumos Keipers Džons [30] analizē pozitīvās un negatīvās programmatūras inženierijas inovācijas un secina, ka veids, kādā programmatūra tiek izstrādāta, ir pārsteidzoši primitīvs. Pētījumos [30], [96] tiek izklāstīts, ka vairums programmatūras izstrādes projektu izgāžas pārtērētu budžetu un grafika dēļ vai arī bīstami zemas izstrādātās programmatūras kvalitātes dēļ. Dažas no pētījumos minētājam problēmām (kas nav mainījušās 30 gadu garumā) ir šādas: 1) sākotnējās prasības reti kad ir vairāk kā 50 % pilnīgas; 2) prasībās un projektējumā ir vairāk kļūdu kā pirmkodā; 3) kļūdu atrašana un labošana ir dārgākā programmatūras izstrādes aktivitāte. Viens no galvenajiem iemesliem ir tas, ka programmatūras izstrāde nav labi strukturēta disciplīna, tā prasa daudz cilvēka manuālu darbu un procesi nav nekādā veidā automatizēti. Disertācijas autors uzskata, ka formāls priekšmetiskās vides modelis ir atslēga programmatūras izstrādes automatizācijai un *MDA* ir viena no programmatūras inženierijas pozitīvajām inovācijām. Taču nespēja izstrādāt formālu priekšmetiskās vides modeli programmatūras inženierijā noved pie šādām problēmām. Priekšmetiskās

vides modeļa apgabalu un saturu nav iespējams pārbaudīt automatizētā veidā (ir iespējama tikai manuāla, subjektīva pārbaude). Nav iespējams automātiski transformēt priekšmetiskās vides modeli uz risinājuma modeli (ir iespējama tikai manuāla, subjektīva transformācija). Tātad nepastāv formālas transformācijas, kā arī ir apgrūtināta formāla trasēšana starp priekšmetiskās vides un risinājuma modeļu elementiem (pēc vairākām manuālas transformācijas iterācijām daži elementi var tikt aizmirsti vai kļūdaini transformēti). Šīs problēmas noved pie zemas programmatūras kvalitātes un neefektīvas programmatūras izstrādes. Pētījuma ilgtermiņa mērķis ir mazināt programmatūras izstrādes izmaksas un celt izstrādājamās programmatūras kvalitāti, piedāvājot metodoloģiju un rīku kopu, kas dotu iespēju visaptverošai priekšmetiskās vides analīzei, balstoties uz formālu priekšmetiskās vides modeli; tādējādi samazinot kļūdu un izmaiņu pieprasījumu skaitu nesaprastas priekšmetiskās vides dēļ.

Pētījuma lauks

MDA piedāvā programmatūras izstrādē abstrahēties no pirmkoda kā galvenā funkcionalitātes noteicēja uz informācijas sistēmas modeli [23]. *MDA* ir programmatūras izstrādes ietvars, kas sastāv no trīs abstrakcijas slāņiem sistēmu analīzei: no skaitļošanas neatkarīgais modelis (*CIM*), no platformas neatkarīgais modelis (*PIM*) un platformas specifiskais modelis (*PSM*). *CIM* apraksta sistēmas prasības un veidu, kādā sistēma strādā tās apkārtējā vidē, taču tās programmatūras struktūras un realizācijas detaļas ir slēptas vai vēl nav noteiktas. *CIM* sauc arī par priekšmetiskās vides modeli vai biznesa modeli. Kā biznesa modelim *CIM* vajadzētu precīzi aprakstīt biznesu tā apkārtējā vidē biznesam saprotamā valodā pēc biznesa cilvēku iniciatīvas un biznesam paredzētiem mērķiem [5]. Taču vienīgais formālais veids *CIM* definēšanai ir Petri tīkli, kas ir pārāk sarežģīti, lai tos varētu saprast biznesa cilvēki, jo tie balstās uz «smago» matemātiku [5]. Vēl viens formāls modelis, kas varētu tikt izmantots kā *CIM*, ir TFM.

Šis darbs ir daļa no topoloģiskā funkcionēšanas modeļa programmatūras inženierijas (*TFM4SE*) pētījumiem. TFM ir priekšmetiskās vides modelis, kas dod iespēju definēt sistēmu formālā veidā, aprakstot sistēmas funkcionālās un topoloģiskās īpašības [4]. TFM atspoguļo sistēmu tās biznesa vidē un parāda, kā sistēma funkcionē, neparādot sistēmas uzbūves detaļas. Šis pētījums piedāvā izmantot TFM par *CIM* tāpat kā *TFM4MDA* metode [53], [64], [4], [55] un [61], iegūstot matemātiski formālu un tādējādi transformējamu CIM. Saistītajā pētījumā [12] TopUML pieeja tiek piedāvāta

programmatūras izstrādei ar uzsvaru uz topoloģiju, kurā *PIM*/*PSM* ir papildināti ar topoloģiju. TopUML ir *UML* profils un pieeja cēloņu un seku attiecību iekļaušanai *UML*, balstoties uz TFM topoloģiju. Lai arī TFM, *TFM4MDA* un TopUML dod labus pamatus *CIM* veidošanai *MDA* kontekstā un iespēju veikt transformāciju uz *PIM*/*PSM*, līdz šim TFM veidošana ir atkarīga no smagnēja manuāla procesa bez rīku atbalsts un ar vāju integrāciju citā informācijas tehnoloģijas (IT) praksēs. Esošie *TFM4MDA* procesi ir aprakstīti [56], [58], [60] un TopUML [13].

Pētījuma mērķis

Disertācijas mērķis ir priekšmetiskās vides analīzes procesa uzlabošana, piedāvājot jaunu pieeju un atbalstošo rīku kopu formāla priekšmetiskā vides modeļa iegūšanai. Tas būtu transformējams un varētu tikt izmantots kā *CIM MDA* kontekstā, tādējādi mazinot programmatūras izstrādes projektu izmaksas, ceļot izstrādātās programmatūras kvalitāti un uzlabojot projektu veiksmīgumu, pateicoties labākai priekšmetiskās vides izpratnei.

Disertācijas uzdevumi mērķu sasniegšanai ir šādi: 1) veikt analīzi par esošajām priekšmetiskās vides modelēšanas pieejām, nosakot to priekšrocības un trūkumus, kā arī atbilstību *CIM MDA* kontekstā; 2) novērtēt priekšmetiskās vides modelēšanas pieejas atbilstoši to formalitātei, atbilstībai *MDA* un praktiskajai lietojamībai; 3) veikt TFM pieejas analīzi, nosakot tās priekšrocības, trūkumus un iespējamos uzlabojumus; 4) izstrādāt integrēto priekšmetiskās vides modelēšanas (*IDM*) pieeju formāla priekšmetiskās vides modeļa iegūšanai TFM formā, balstoties uz formālām zināšanām par priekšmetisko vidi un izmantojot esošo IT praksi; 5) atbilstoši *MDA* standartiem izstrādāt *IDM* atbalstošo rīku kopu, kas sastāvētu no lietošanas gadījumu rīka, TFM rīka un lietošanas gadījumu uz TFM transformācijas rīka; 6) veikt gadījuma izpēti, izmantojot *IDM* pieeju un rīku kopu reālam programmatūras izstrādes projektam.

Pētījuma tēma ir priekšmetiskās vides modelēšana ar uzsvaru uz priekšmetiskās vides modeli programmatūras inženierijas kontekstā.

Pētījuma objekti ir *MDA* un TFM ar uzsvaru uz to kā formālā veidā iegūt *CIM* atbilstoši *MDA* standartiem.

Pētījuma metodes, kas tika izmantotas, ir šādas – salīdzinājuma analīze, metamodelēšanas metode un modeļu transformācija, projektēšana un gadījuma izpēte.

Disertācijas tēzes ir šādas: 1) ja kādai konkrētai biznesa videi visas atbilstošās priekšmetiskās vides zināšanas ir formāli definētas, tad jābūt iespējai tās automātiski transformēt uz priekšmetiskās vides modeli; 2) TFM lietojamību ir iespējams būtiski uzlabot novēršot neformālā apraksta un rīku trūkuma problēmas ar jaunu pieeju, kas

balstītos uz vispārpieņemtiem standartiem; 3) deklaratīvās un procedurālās zināšanas papildina viena otru un dod iespēju pilnvērtīgai priekšmetiskās vides analīzei.

Zinātniskais pienesums un praktiskā vērtība

Šā pētījuma **zinātniskais pienesums** ir jauna pieeja – integrētā priekšmetiskās vides modelēšana (*IDM*) priekšmetiskās vides modelēšanai, kas dod iespēju iegūt matemātiski formālu priekšmetiskās vides modeli TFM formā, kā arī izmanto vispārpieņemtus standartus par pamatu priekšmetiskās vides modelēšanas procesā un piedāvā modeļu transformāciju. *IDM* balstās uz deklaratīvajiem un procedurālajiem priekšmetiskās vides zināšanu aspektiem, lietojot ontoloģiju un lietošanas gadījumu kā ieeju, kā arī modeļu transformāciju uz TFM, kas izmanto dabīgās valodas apstrādes metodes (*Natural Language Processing – NLP*).

Pētījuma **praktiskā vērtība** ir *IDM* atbalstošā rīku kopa, kas sastāv no lietošanas gadījumu rīka, TFM rīka un lietošanas gadījumu uz TFM transformācijas rīka; kā arī gadījuma izpēte, kurā *IDM* tiek lietots e-komercijas programmatūras izstrādes projektam. Pirms šī pētījuma TFM atbalstam nav bijis nekādu rīku, taču, pateicoties izstrādātajai *IDM* pieejai un rīku kopai, ir iespēja iegūt TFM automātiskā veidā, izmantojot modeļu transformāciju.

Rezultātu aprobācija

Galvenie pētījuma rezultāti ir prezentēti 8 starptautiskās zinātniskās konferencēs (4 no tām organizētas Latvijā, 4 ārzemēs): 1) *11th International Baltic Conference on DB and IS* (*DBIS 2014*), Igaunija, Tallina, 8.–11. jūnijs, 2014; 2) RTU 54. starptautiskā zinātniskā konference, Rīga, Latvija, oktobris, 2013; 3) *7th International Conference on Evaluation of Novel Approaches to Software Engineering* (*ENASE 2012*), Polija, Vroclava, 29.–30. jūnijs, 2012; 4) *3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development* (*MDA & MDSD 2011*), Ķīna, Pekina, 8.–11. jūnijs, 2011; 5) *9th International Baltic Conference on DB and IS* (*DBIS 2010*), Latvija, Rīga, 5.–7. jūlijs, 2010; 6) *2nd International Workshop on Model-Driven Architecture and Modeling Theory-Driven Development* (*MDA & MTDD 2010*), Grieķija, Atēnas, 2010; 7) RTU 51. starptautiskā zinātniskā konference, Rīga, Latvija, oktobris, 2010; 8) *13th East-European Conference* (*ADBIS 2009*), Latvija, Rīga, 7.–10. septembris, 2009.

Pētījuma rezultāti ir publicēti 11 zinātniskajos rakstos:

1. Šlihte A. Introduction to Integrated Domain Modeling Toolset// Scientific Journal of RTU. Computer Science. – 2014. (to be published)
2. Šlihte A. The Integrated Domain Modeling: A Case Study// In Proceedings of the 11th International Baltic Conference, Baltic DB&IS 2014. TUT Press, 2014. – pp. 465–470. [ISBN 978-9949-23-633-6]
3. Osis J., Šlihte A., Jansone A. Using Use Cases for Domain Modeling// Proceedings of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012). Poland, Wrocław, June 29–30, 2012. Lisbon: SciTePress, 2012. – pp. 224–231. [ISBN 9789898565136, Indexed by Thomson Reuters, Inspec, EI, DBLP]
4. Doniņš U., Osis J., Šlihte A., Asņina Ē., Gulbis B. Towards the Refinement of Topological Class Diagram as a Platform Independent Model// Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development (MDA & MDSD 2011). China, Beijing, June 8–11, 2011. Lisbon: SciTePress, 2011. – pp. 79–88. [ISBN 9789898425591, Indexed by Thomson Reuters, Inspec, EI, DBLP, ISTP]
5. Šlihte A., Osis J., Doniņš U. Knowledge Integration for Domain Modeling// Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development (MDA & MDSD 2011). China, Beijing, June 8-11, 2011. Lisbon: SciTePress, 2011. – pp. 46–56. [ISBN 9789898425591, Indexed by Thomson Reuters, Inspec, EI, DBLP, ISTP]
6. Šlihte A., Osis J., Doniņš U., Asņina Ē., Gulbis B. Advancements of the Topological Functioning Model for Model Driven Architecture Approach// Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development (MDA & MDSD 2011). China, Beijing, June 8–11, 2011. Lisbon: SciTePress, 2011. – pp. 91–100. [ISBN 9789898425591, Indexed by Thomson Reuters, Inspec, EI, DBLP, ISTP]
7. Asņina Ē., Gulbis B., Osis J., Alksnis G., Doniņš U., Šlihte A. Backward Requirements Traceability within the Topology-based Model Driven Software Development// Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development (MDA & MDSD 2011). China, Beijing, June 8–11, 2011. Lisbon: SciTePress, 2011. – pp. 36–45. [ISBN 9789898425591, Indexed by Thomson Reuters, Inspec, EI, DBLP, ISTP]
8. Šlihte A. The Concept of a Topological Functioning Model Construction Tool// Advances in Databases and Information Systems: 13th East-European Conference,

ADBIS 2009. Associated Workshops and Doctoral Consortium, Local Proceedings, Latvia, Riga, September 7–10, 2009. – pp. 476–484.
9. Šlihte A. The Specific Text Analysis Tasks at the Beginning of MDA Life Cycle// Databases and Information Systems Doctoral Consortium, Latvia, Riga, July 5–7, 2010. – pp. 11–22.
10. Osis, J., Šlihte, A. Transforming Textual Use Cases to a Computation Independent Model// Model-Driven Architecture and Modeling Theory-Driven Development: Proceedings of the 2nd International Workshop on Model-Driven Architecture and Modeling Theory-Driven Development (MDA & MTDD 2010). Greece, Athens, July 22–24., 2010. Lisbon: SciTePress, 2010. – pp. 33–42. [ISBN 9789898425164, Indexed by SCOPUS, Thomson Reuters, Inspec, DBLP]
11. Šlihte A. Implementing a Topological Functioning Model Tool// Scientific Journal of RTU. 5. series., Computer Science. – 43. vol. – 2010. – pp. 68–75.

Promocijas darba saturs

Disertācijā ir ievads, 4 nodaļas, secinājumi, 10 pielikumu un literatūras saraksts. Kopā – 216 lapaspuses, 52 attēli un 4 tabulas. Literatūras sarakstā ir 100 avotu.

Ievadā tiek izskaidrota disertācijas motivācija, definēts pētījuma mērķis un uzdevumi, pētījuma novitāte, praktiskā vērtība un rezultātu aprobācija. *1. nodaļa* definē priekšmetiskās vides modeli, priekšmetiskās vides modelēšanu un modeļu vadāmo arhitektūru (*MDA*), analizē dažas no priekšmetiskās vides modelēšanas pieejām. *2. nodaļa* vispirms izskaidro integrētās priekšmetiskās vides modelēšanas (*IDM*) pamatojumu, tad definē un demonstrē pieejas pamatprincipus. *3. nodaļa* izskaidro *IDM* atbalstošās rīku kopas izstrādi un demonstrē tās lietošanu. *4. nodaļa* apraksta gadījuma izpēti, kurā *IDM* pieeja un atbalstošā rīku kopa tiek lietota reālam e-komercijas programmatūras izstrādes projektam. *Secinājumos* tiek apkopoti disertācijas rezultāti un norādīti iespējamie turpmākie pētījuma virzieni. Disertācijai ir desmit *pielikumu*: 1) attēlu saraksts; 2) tabulu saraksts; 3) *IDM* lietošanas gadījumu metamodelis atbilstoši *Ecore*; 4) *IDM TFM* metamodelis atbilstoši *Ecore*; 5) *IDM* rīku kopas lietošanas gadījumu rīka artefakti; 6) *IDM* rīku kopas TFM rīka artefakti; 7) *IDM* rīku kopas TFM diagrammas rīka artefakti; 8) *IDM* rīku kopas lietošanas gadījumu uz TFM transformācijas rīka artefakti; 9) *IDM* rīku kopas lietotāja pamācība; 10) *IDM* rīku kopas testētāju anketas rezultāti.

1. PRIEKŠMETISKĀS VIDES MODELĒŠANA UN MODEĻU VADĀMĀ ARHITEKTŪRA

Šī nodaļa piedāvā pārskatu par priekšmetiskās vides modelēšanas tēmu un analizē dažas no esošajām pieejām, kā arī apskata modeļu vadāmo arhitektūru (*MDA*). Autors vispirms definē priekšmetiskās vides modelēšanu un tad apskata literatūru par priekšmetiskās vides modelēšanas pieejām, analizējot to priekšrocības un trūkumus.

1.1. Priekšmetiskās vides modelēšana

Priekšmetiskās vides modelis un priekšmetiskās vides modelēšana ir ļoti nozīmīgas programmatūras inženierijas sastāvdaļas. Šajā pētījumā autors lietos šādu priekšmetiskās vides modeļa definīciju: *priekšmetiskās vides modelis ir kādas priekšmetiskās vides jeb biznesa vides atspoguļojums no skaitļošanas neatkarīgā veidā, kas eksistē vai varētu eksistēt reālajā dzīvē, tas sastāv no terminoloģijas, konceptiem, attiecībām, likumiem un biznesa procesiem. MDA* pamācībā [43] Objektu vadības grupa (*OMG*) uzsver to, ka, ja ēkas tiktu būvētas tādā veidā, kā šobrīd tiek izstrādāta programmatūra, tad nebūtu iespējams tās savienot, vienkāršā veidā nomainīt apdari vai pielāgot tās citām vajadzībām; un pat ļaunāk, šīs ēkas pastāvīgi sabruktu. Ir nepieciešama atbilstoša projektēšanas fāze un priekšmetiskās vides modelis, lai būtu iespējam ietaupīt pūles un būtu mazāk kļūdu vai neatbilstību programmatūra mērķim. Tas nepieciešams arī izstrādātās programmatūras integrēšanas un uzturēšanas vajadzībām, kam nepieciešama atbilstoša dokumentācija. *OMG* min vēl vienu ieguvumu no formāla priekšmetiskās vides modeļa – iespēja automatizēt vismaz daļu no programmatūras risinājuma izstrādes [43]. Atbilstoši Keiperam Džonsam [30] vairums programmatūras izstrādes projektu nav veiksmīgi. Pētījuma [56] autori secina, ka priekšmetiskās vides modelim vajadzētu būt programmatūras izstrādes stūrakmenim, tādā veidā būtu iespējams mazināt neatbilstības un ierobežot izdevumus, kā arī iegūt kvalitatīvu projektējumu un dokumentāciju. Vēl viens ieguvums ir tāds, ka priekšmetiskā vides modelēšana piedāvā iespēju analizēt priekšmetisko vidi, pat ja nav paredzēts izstrādāt programmatūru. Piemēram, tas var būt nepieciešamas kāda biznesa procesa optimizēšanai, lai samazinātu izmaksas vai uzlabotu kvalitāti.

1.2. Modeļu vadāmā arhitektūra

MDA ir *OMG* pieeja modeļu lietošanai programmatūras izstrādē [86]. *OMG* uzskata *MDA* par vienu mazu soli garam ceļam uz programmatūras izstrādes pārveidi no amatniecības uz inženierijas disciplīnu [43]. *OMG* apraksta dažādos abstrakcijas

slāņus un to attiecības, taču tas nedefinē veidu, kādā modeļi ir jāizstrādā un kādai notācijai būtu jāseko to atspoguļošanai. Pastāv vairākas dažādu pētnieku rekomendācijas, kas var būt ļoti atšķirīgas [33]. *MDA* definē 3 skatupunktus un atbilstošos modeļus – no skaitļošanas neatkarīgais modelis (*CIM*), no platformas neatkarīgais modelis (*PIM*), platformas specifiskais modelis (*PSM*). Konceptuālā līmenī *MDA* ir holistiska pieeja IT dzīves cikla uzlabošanai – specifikācijas, arhitektūras, izstrādes, uzstādīšanas, uzturēšanas un integrācijas – balstoties uz formālu modelēšanu [58]. Jānis Osis uzskata, ka *MDA* spēks ir modeļu transformācijā, jo tā pieprasa formālas valodas lietošanu modeļu aprakstam, un tas varētu novest pie matemātikas lomas pieauguma programmatūras izstrādē [54]. *OMG* piedāvā arī citus svarīgu standartus, kas papildina *MDA*. Vienotā modelēšanas valoda (*UML*) dod iespēju veidot, apskatīt, labot modeļus standartizētā veidā analīzes un projektēšanas laikā. Meta-objektu iekārta (*MOF*) standartizē veidu, kā modeļi tiek glabāti un pārvaldīti [42]. *XML* meta-datu apmaiņa (*XMI*) ir modeļu apmaiņas formāts, kas balstās uz *MOF* un dod iespēju glabāt modeļus vienotā formātā. Vaicājums/skats/transformācija (*QVT*) ir *OMG* standarts modeļu transformācijām, kas sastāv no vairākām modeļu transformāciju valodām – deklaratīvā, imperatīvā, operacionālā kārtošana (*QVTo*) un melnās kastes [42].

1.3. Priekšmetiskās vides modelēšanas pieejas un to novērtējums

Šī disertācija apskata dažas no šobrīd izmantotajām priekšmetiskās vides modelēšanas pieejām. Tiek apskatītas šādas pieejas – biznesa procesu modelēšana un notācija (*BPMN*), notikumu vadītās procesu ķēdes (*EPC*), topoloģiskais funkcionēšanas modelis (TFM), ontoloģija un lietošanas gadījumi. Petri tīkli ir matemātiski formāls modelis, taču tas parasti netiek izmantots vai tiek slēpts no biznesa lietotājiem, jo tā atspoguļojums nav intuitīvs un pārāk sarežģīts [5]. Tādēļ Petri tīkli netiks apskatīti. Taču tendenču novērtēšanai tiks apskatītas dažas inovatīvas pieejas, kas izmanto mākslīgā intelekta metodes – lingvistiskais asistents priekšmetiskās vides modelēšanai (*LIDA*) [76] un dabīgās valodas prasību analīze (*NIBA*) [72].

Protams, pastāv ļoti daudz priekšmetiskās vides modelēšanas pieeju, un autors ir apskatījis tikai dažas. Taču šīs pieejas dod ieskatu priekšmetiskās vides modelēšanā un parāda, kādos veidos priekšmetisko modeli šobrīd iespējams modelēt. Priekšmetiskās vides modelēšana tiks novērtēta, balstoties uz 3 kritēriju grupām, kas apkopotas tabulā 1.1: 1) kritēriji formālam priekšmetiskās vides modelim; 2) kritēriji atbilstībai *MDA*; 3) kritēriji praktiskai lietojamībai. Formālisms ir teorija vai matemātisks filozofijas skatupunkts atbilstoši Nikolasam D. Gudmenam [46], kas apgalvo, ka matemātika ir

likumu pārvaldīta simbolu manipulācija. Tātad šī manipulācija ir formāla. Kādā veidā novērtēt, vai priekšmetiskās vides modelis ir formāls? Metamodelis sniedz zināmu formalitātes pakāpi, jo tas definē likumus, kādā veidā var tikt veidoti priekšmetiskās vides modeļa elementi. Pie tam, ja metamodelis atbilst meta-metamodelim, tas piedāvā vēl lielāku formalitātes pakāpi. Lai novērtētu priekšmetiskās vides modeļa formalitātes pakāpi, autors piedāvā 3 kritērijus: matemātiski formāls modelis, formāla apgabala definēšana, formāla modeļa pārbaude. Lai novērtētu atbilstību *MDA*, autors piedāvā šādus 3 kritērijus: no skaitļošanas neatkarīgais skatupunkts, atbilstība *MOF* un modeļu transformācija uz *PIM*/*PSM*. Tā kā diskusija ir par priekšmetiskās vides modeli, tam jāatbilst *MDA CIM* abstrakcijas līmenim. Papildus formālam modelim un atbilstībai *MDA*, arī praktiskā lietojamība tiek novērtēta atbilstoši šādiem 4 kritērijiem: atbalsts deklaratīvajām zināšanām, atbalsts procedurālajām zināšanām, rīku atbalsts un popularitāte/atpazīstamība. Popularitāte tiks novērtēta, balstoties uz publikāciju skaitu *Google Scholar* [54] datubāzē.

1.1. tabula

Priekšmetiskās vides modelēšanas pieeju novērtējums

Kritērijs	**Vērtību apgabals**	**BPMN**	**EPC**	**TFM**	**Ontoloģija**	**Liet. gad.**	**LIDA**	**NIBA**
Matemātiski formāls modelis	Jā/Nē	Nē	Nē	Jā	Jā	Nē	Nē	Nē
Formāla apgabala definēšana	Jā/Nē	Nē	Nē	Jā	Nē	Nē	Nē	Nē
Formāla modeļa validēšana	Jā/Nē	Nē	Nē	Jā	Jā	Nē	Nē	Nē
No skaitļošanas neatkarīgs skatupunkts	Jā/Nē	Jā	Jā	Jā	Jā	Jā	Jā	Jā
Atbilstība MOF	Jā/Nē	Jā	Nē	Jā	Jā	Nē	Nē	Jā
Modeļa transformācija uz PIM/PSM	Jā/Nē	Jā	Jā	Jā	Nē	Nē	Jā	Jā
Deklaratīvo zināšanu atbalsts	Jā/Nē	Nē	Jā	Nē	Jā	Nē	Nē	Nē
Procedurālo zināšanu atbalsts	Jā/Nē	Jā	Jā	Jā	Nē	Jā	Jā	Jā

Rīku atbalsts	Jā/Nē	Jā	Jā	Nē	Jā	Jā	Jā	Jā
Popularitāte	Jā/Nē	Jā	Jā	Nē	Jā	Jā	Nē	Nē
Kopējais punktu skaits	Punkti no 0 līdz 10	**6**	**6**	**7**	**5**	**4**	**4**	**5**

Kaut arī *BPMN*, *EPC* un *NIBA* ir savi meta-modeļi, taču to modeļi nav matemātiksi formāli, jo likumi, ar kuriem tos veido, nav definēti, balstoties uz matemātiku. Savukārt TFM balstās uz matemātiku un tādēļ tas tiek uzskatīts par matemātiski formālu modeli. Ontoloģija balstās uz predikātu loģiku, tādējādi autors pieskaita ontoloģiju pie matemātiski formāliem modeļiem. Vēl viens piemērs matemātiski formālam modelim ir Petri tīkli [27]. Turklāt TFM apgabals tiek formāli definēts ar noslēguma procedūru un balstās uz ieejām un izejām. TFM pārbaude tiek veikta, izmantojot ciklu analīzi, un galvenais likums paredz, ka jābūt vismaz vienam galvenajam ciklam, lai sistēma varētu funkcionēt. Tā kā ontoloģija balstās uz predikātu loģiku, ir iespējams to pārbaudīt, veicot vaicājumus ar semantisko spriedēju (no angļu valodas *reasoner*). Savukārt *BPMN* pilnībā atbilst *MDA*. Pētījuma [52] autori apraksta transformāciju no *BPM* uz *UML* aktivitāšu diagrammu. Biznesa procesu izpildīšanas valoda tīmekļa servisiem (*WS-BPEL*) ir standarts modeļu transformācijai no *BPMN* uz *BPEL*, tādējādi iespējams iegūt strādājošu programmatūru ar modelēšanas palīdzību [67]. No apskatītajām pieejām ontoloģija un *EPC* atbalsta deklaratīvās zināšanas. Rīku atbalsts eksistē visām pieejām, izņemot TFM, kas ir liels trūkums TFM pieejai. Popularitātes novērtēšanai tika izmantots 1000 publikāciju slieksnis, pēc kura autors noteica, vai pieeja ir populāra vai nē. Publikāciju skaits pieejām ir šāds: 1) ontoloģija – 1250000; 2) lietošanas gadījumi – 98500; 3) *BPMN* – 7690; 4) *EPC* – 2970; 5) *NIBA* – 101; 6) *LIDA* – 60; 7) TFM – 57.

No formalitātes viedokļa stiprākās pieejas ir TFM un ontoloģija. 3 pieejas ieguva augstāko vērtējumu atbilstībā *MDA – BPMN*, TFM un *NIBA*. No praktiskās lietojamības viedokļa *EPC* sasniedza vislabāko rezultātu, atstājot *BPMN*, ontoloģiju un lietošanas gadījumus aiz sevis. Tomēr neviena no apskatītajām pieejām nesasniedza maksimālo punktu skaitu 10. Vistuvāk maksimālajam punktu skaitam ir TFM, bet šai pieejai galvenie trūkumi ir praktiskā lietojamība, un tas ir galvenokārt tādēļ, ka tai nav atbalstoša rīka, kā arī tā neatbalsta deklaratīvās zināšanas un nav atpazīstama. Taču šī pieeja ir visstiprākā no formālisma viedokļa, un tā ir TFM galvenā priekšrocība. TFM praktisko lietojamību ir jāuzlabo, un tas ir arī viens no šā pētījuma mērķiem

1.4. Kopsavilkums

Galvenie esošo priekšmetiskās vides modelēšanas pieeju trūkumi ir šādi. TFM ir nepopulāra (maz atpazīstama) pieeja, un tai ir sarežģīts izstrādes process. *TFM4MDA* nav atbalstošo rīku un galvenokārt tas ir tādēļ, ka TFM izstrāde balstās uz neformālu priekšmetiskās vides aprakstu dabīgajā valodā. Šis apraksts programmatūras izstrādes (vai priekšmetiskās vides analīzes) projekta sākumā varētu nemaz neeksistēt. Protams, sistēmanalītiķis vienmēr varētu šo neformālo aprakstu izstrādāt, taču vai tam ir jēga, ja tajā pat laikā varētu izmantot kādu labāk strukturētu formātu zināšanu pierakstīšanai (ja analītiķim ir izvēle)? Autors apšauba šāda teksta izstrādes efektivitāti, ja mērķis ir izstrādāt priekšmetiskās vides modeli.

2. INTEGRĒTĀ PRIEKŠMETISKĀS VIDES MODELĒŠANAS PIEEJA

Šī nodaļa apraksta integrēto priekšmetiskās vides modelēšanas (*IDM*) pieeju, kuru izstrādājis autors. Lietošanas gadījumi un ontoloģija tiek lietotas kā priekšmetiskās vides zināšanas, un topoloģiskais funkcionēšanas modelis (TFM) tiek lietots kā priekšmetiskās vides modelis. IDM pieeja piedāvā formālu veidu, kā izmantot ontoloģiju programmatūras inženierijā, dodot iespēju iegūt no skaitļošanas neatkarīgo modeli (*CIM*) modeļu vadāmās arhitektūras (*MDA*) kontekstā.

2.1. IDM pieejas pamatojums

IDM pieeja tiek pamatota ar TFM, *TFM4MDA* un TopUML pieeju pētījumiem. Atbilstoši [5] vienīgie formālie veidi priekšmetiskās vides modeļa veidošanai ir Petri tīkli [46] un TFM. TFM matemātiskais pamats ir tā galvenā priekšrocība. Funkcionālās īpašības balstās uz savienojamību, slēgumu, apkārtni un nepārtrauktu kartēšanu. Otrā TFM priekšrocība ir ieejas un izejas, kas atbilst sistēmu teorijai [65]. Ieejas un izejas ir svarīgas priekšmetiskās vides modelim un dod iespēju noteikt tā apgabalu. Ieeju un izeju analīze dod iespēju arī pētīt priekšmetiskās vides modeļa atkarības. Ciklu struktūra TFM un tā galvenais funkcionēšanas cikls ir trešā TFM priekšrocība. No funkcionēšanas perspektīvas visām iespējamām sistēmām (tehniskajām, biznesa vai bioloģiskajām) vajadzētu būt vismaz vienam orientētam ciklam [60]. Katrā TFM jābūt vismaz vienam noslēgtam ciklam, taču pārsvarā to ir daudz, un veidojas ciklu hierarhija. Šī TFM īpašība dod iespēju analizēt dažādu funkcionējošu sistēmu līdzības un atšķirības[53]. Ceturtā TFM priekšrocība ir modeļu transformācija, kas definēta kā daļa no *TFM4MDA*, kuru uzlaboja Uldis Doniņš, izstrādājot TopUML [13].

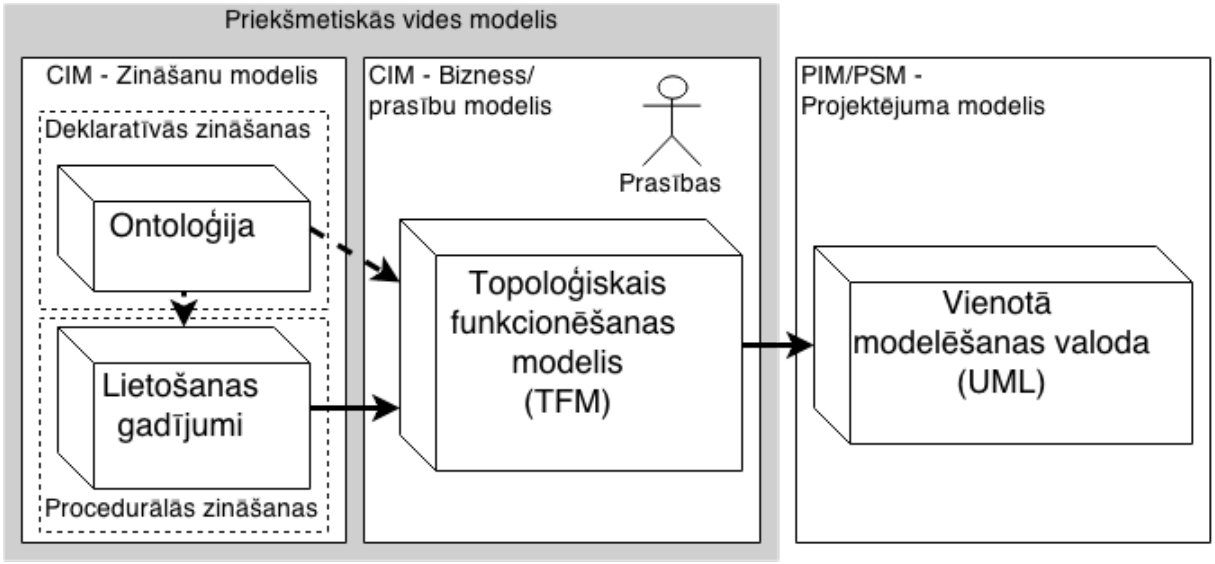
Pētnieki ārpus TFM grupas ir pamanījuši TFM pieeju un uzskata to par ceļu uz CIM definēšanu *MDA* kontekstā. Taču viņi secina, ka daži svarīgi CIM un projektēšanas priekšnosacījumi nav ņemti vērā [21]. Pirmais trūkums ir tas, ka TFM nepiedāvā visus priekšmetiskās vides modeļa aspektus. No priekšmetiskās vides viedokļa TFM apraksta tikai procedurālās zināšanas, taču deklaratīvās netiek ņemtas vērā. Otrais TFM trūkums ir *TFM4MDA* pieejas sākums – neformālais apraksts dabīgajā valodā. Šāds apraksts var būt pārāk nestrukturēts, lai kalpotu par labu pamatu labam modeli, un pēc modeļa uzbūvēšanas to elementus nav iespējams atsekot. Trešais trūkums ir tas, ka TFM, *TFM4MDA* un TopUML nav rīku atbalsta. Ceturtais trūkums ir tāds, ka, neskatoties uz to, ka pastāv pamācības un piemēri TFM konstruēšanai, šis process ir smags un attāls no pieņemtajiem programmatūras izstrādes standartiem.

Balstoties uz TFM pieejas priekšrocībām, ir skaidrs, ka tā ir spēcīga pieeja priekšmetiskās vides modelēšanai. Taču trūkumi ir diezgan nopietni, un tie kavē šīs pieejas reālu lietošanu praksē. Pirmkārt, rīku atbalsts ir vitāli svarīgs jebkurai priekšmetiskās vides modelēšanas pieejai. Lai būtu iespēja šādu rīku atbalstu izstrādāt *TFM4MDA*, kā tas aprakstīts līdz šim, būtu jārisina sarežģītais dabīgās valodas apraksta analīzes uzdevums. Autors ir apskatījis dažas no pieejām, kas izmanto dabīgās valodas apstrādes metodes (*NLP*) priekšmetiskās vides modelēšanai (*LIDA* un *NIBA*), taču šīs pieejas piedāvā ierobežotus rezultātus, un tie vienalga balstās uz kāda veida struktūru neformālajā aprakstā. Tādēļ autors atsakās no šāda neformāla apraksta un piedāvā izvēlēties strukturētāku (formālāku) veidu, kā definēt šīs zināšanas. Pētījumā [5] autori apgalvo, ka transformāciju no *CIM* zināšanu modeļa uz *CIM* biznesa modeli (šāds *CIM* sadalījums tiek piedāvāts [5]) nav iespējams veikt automātiski, jo zināšanu modelis ir veidots neformāli, tādēļ transformācijai ir nepieciešama cilvēka līdzdalība. Tātad vēl viens nepieciešamais uzlabojums TFM pieejai ir šīs transformācijas izstrāde formālā veidā, lai to varētu veikt automātiski no formāli definēta zināšanu modeļa.

2.2. Deklaratīvo un procedurālo zināšanu integrēšana

Zināšanas nozīmē saprast kādu tēmu, tai skaitā – tās konceptus un faktus, kā arī attiecības starp tiem un mehānismu, kā tos apvienot, lai risinātu šīs tēmas problēmas [25]. Priekšmetiskās vides zināšanas var tikt uzskatītas par daļu no *CIM*, t. i., par *CIM* zināšanu modeli [5]. Termins «zināšanas» var tikt izmantots, lai apgalvotu faktu, metožu vai principu zināšanu. Šis izplatītais lietojums atbilst zināšanu veidam – «zināt par». Otrs šā termina lietojums attiecas uz faktu, metožu un principu zināšanu tik padziļināti, lai spētu tos lietot kāda mērķa sasniegšanai (tātad, lai varētu kaut ko izdarīt). Šī definīcija

atbilst zināšanu veidam – «zināt, kā». Kognitīvajā psiholoģijā un biznesa analītikā zināšanas tiek iedalītas deklaratīvajās un procedurālajās zināšanās [73], [34]. No studenta viedokļa, kurš apgūst zināšanas: 1) deklaratīvās zināšanas ir tad, ja students zina vai saprot, piemēram, «Rīga ir Latvijas galvaspilsēta» vai «Grāmatu katalogs sastāv no grāmatām»; 2) procedurālās zināšanas ir tad, ja students spēj kaut ko izdarīt. Piemēram, viņš zina, kā nopirkt lidojuma biļeti uz Rīgu vai paņemt grāmatu bibliotēkā.

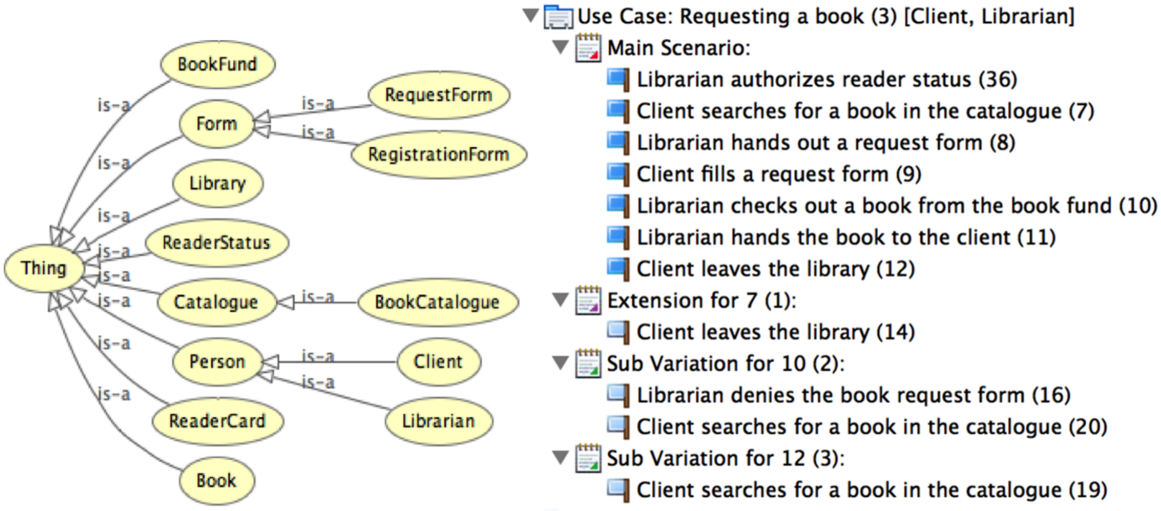


2.1. att. IDM pieejas uzbūve

Priekšmetiskās vides zināšanas par biznesa sistēmu un tās apkārtējo vidi parasti eksistē dažādos dokumentos dabīgās valodas formā vai arī tikai cilvēku galvās. Ir nepieciešams šīs zināšanas notvert un glabāt tādā veidā, lai tā varētu saprast dators (ir nepieciešams formālāks veids par dabīgo valodu). Lai pietiekami aprakstītu biznesa sistēmu un tās apkārtējo vidi, ir nepieciešamas abas – deklaratīvās un procedurālās zināšanas. Attēlā 2.1 autors parāda, kā deklaratīvās un procedurālās zināšanas var tikt integrētas, lai iegūtu *CIM MDA* kontekstā. Tas sākas ar priekšmetiskās vides zināšanu definēšanu formālā *CIM* zināšanu modelī, kas ietver lietošanas gadījumus un ontoloģiju. Pēc tam tiek piedāvāta formāla transformācija no zināšanu modeļa uz *CIM* biznesa/prasību modeli, kas tiek definēts ar TFM palīdzību. Tad atbilstoši *MDA CIM* var tikt transformēts uz *PIM/PSM*. Detalizēta transformācija uz *UML* no TFM ir aprakstīta Ulda Doniņa doktora disertācijā [13].

IDM pieejā tiek izmantots ontoloģijas standarts – *OWL* [70]. Ontoloģija ir labi piemērota deklaratīvo zināšanu atspoguļošanai. Attēla 2.2 kreisajā pusē autors ir devis ontoloģijas piemēru bibliotēkas priekšmetiskajai videi – tiek parādīta klašu hierarhija. Ontoloģijas izstrādei autors izmanto *Protégé* [75] rīku. Papildus klašu hierarhijai arī

klašu īpašības tiek definētas ontoloģijā. Piemēram, objektam «*Librarian*» (latviski – bibliotekārs) tiek definēta īpašība «*checkOut*» (latviski – izņemt), kas nosaka, ka bibliotekārs var izņemt grāmatu (klase «*Book*») nevis kādu citu klasi.



2.2. att. Ontoloģijas klašu hierarhija un lietošanas gadījums bibliotēkas piemēram

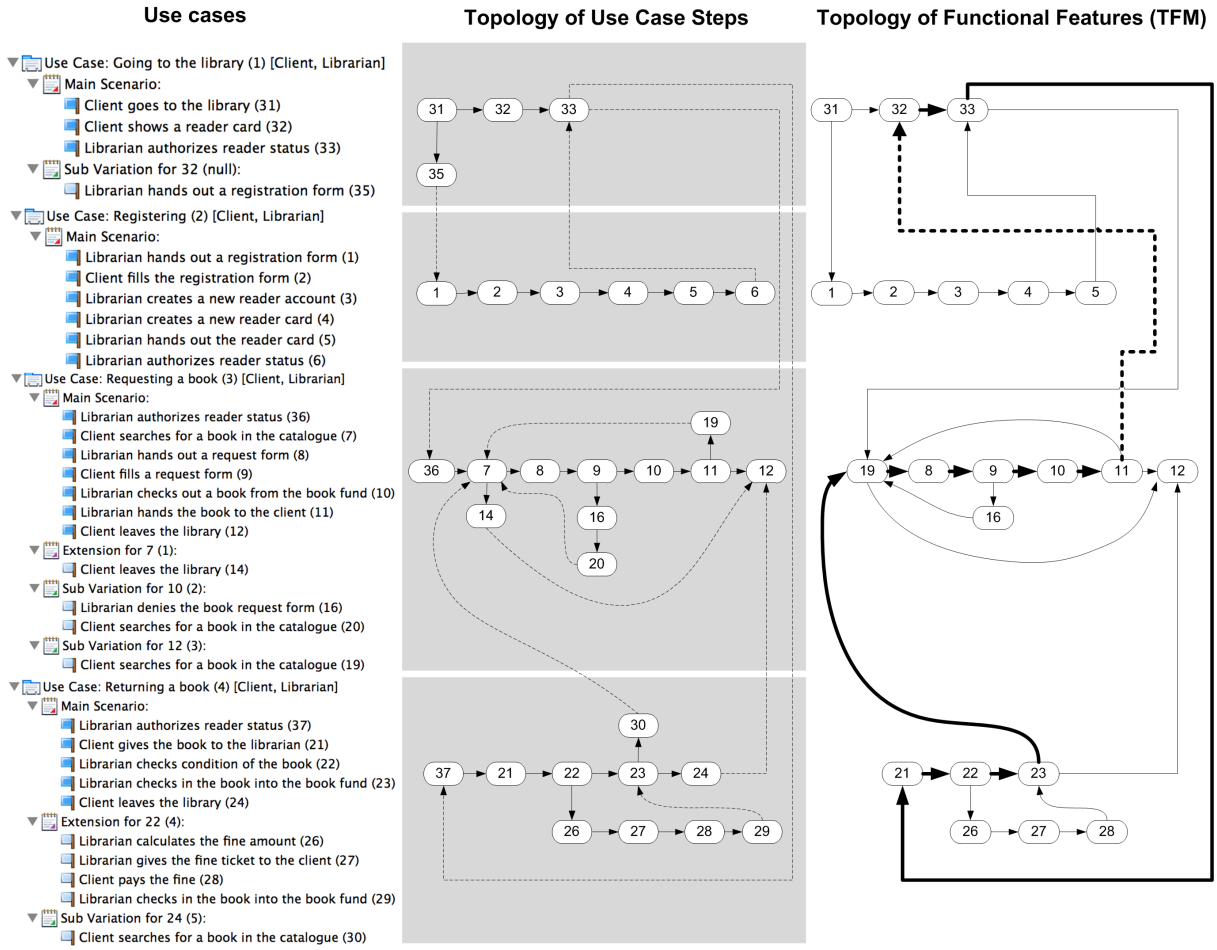
Lietošanas gadījumus nevajadzētu jaukt ar *UML* lietošanas gadījumu diagrammu, kā arī pastāv daudz dažādas sagataves lietošanas gadījumiem, pie tam to struktūru drīkst mainīt atbilstoši situācijai un izstrādes komandai [39]. *IDM* pieejai autors izstrādāja īpašu lietošanas gadījumu struktūru un atbilstošu metamodeli. Attēla 2.2 labajā pusē ir parādīts lietošanas gadījuma piemērs atbilstoši šim īpašajam formātam – lietošanas gadījums ir grāmatas pieprasīšana bibliotēkā. Šāda struktūra tiek izmantota lietošanas gadījumiem: 1) lietošanas gadījuma nosaukums; 2) aktieru saraksts; 3) galvenais scenārijs; 4) paplašinājumi; 5) apakšvariācijas.

Dabīgās valodas apstrādes metodes (*NLP*) tiek lietotas *IDM*, lai dators spētu saprast šīs zināšanas un būtu iespējams veikt modeļu validāciju un transformāciju. *IDM* lietošanas gadījumu modelis tiek validēts atbilstoši ontoloģijai, lai novērstu divdomības lietošanas gadījumu soļos un neatbilstību priekšmetiskajai videi. Statistiskais parsētājs [88] tiek lietots lietošanas gadījuma soļu teikumu analīzei, tādējādi tiek izgūti dati TFM funkcionālo īpašību veidošanai. Tas tiek darīts, lai palīdzētu sistēmanalītiķim novērst nepilnības, divdomības vai neatbilstības lietošanas gadījumu teikumos atbilstoši ontoloģijai.

2.3. Topoloģiskā funkcionēšanas modeļa izgūšanas

Šī apakšnodaļa definē lietošanas gadījumu uz TFM modeļu transformācijas principus, kā arī skaidro *NLP* pielietošanu *IDM* pieejā. *NLP* tiek lietots aktieru (atbildīgo personu) noteikšanai, kā arī funkcionālo īpašību apraksta noteikšanai. Tiek demonstrēts modeļu transformācijas piemērs no lietošanas gadījumiem uz TFM.

Šajā IDM dzīves cikla posmā konkrētajai priekšmetiskajai videi jau ir definēta ontoloģija un lietošanas gadījumi. Nākamais mērķis ir iegūt TFM. Tam nepieciešami 2 soļi: 1) funkcionālo īpašību izgūšana; 2) topoloģijas izgūšana. Pirmkārt, katrs lietošanas gadījuma solis ir sakārtots funkcionālo īpašību rinda.



2.3. att. Topoloģisko attiecību izgūšana no lietošanas gadījumiem

Papildus analizējot paplašinājumus un apakšvariācijas, ir iespējams noteikt zarošanos TFM modelī. Paplašinājumi pievieno papildu sekas atspoguļotajai funkcionālajai īpašībai, uz kuru paplašinājums atsaucas. Taču apakšvariācija pievieno papildu sekas iepriekšējai funkcionālajai īpašībai (kuru atspoguļo iepriekšējais solis). Tādējādi tiek definētas cēloņu un seku attiecības starp funkcionālajām īpašībām, kuras atspoguļo konkrēts lietošanas gadījums. Kā parādīts attēlā 2.3., 4 galvenās funkcionālo

īpašību rindas nāk no lietošanas gadījumu galvenajiem scenārijiem. Lietošanas gadījumu paplašinājumu un apakšvariāciju piemēram jāpievērš uzmanība funkcionālajām īpašībām ar numuru 1 un 11. Funkcionālajai īpašībai nr. 1 ir vienas papildu sekas apakšvariācijas nr. 2a dēļ, bet funkcionālajai īpašībai nr. 11 ir papildu sekas paplašinājuma nr. 4a dēļ.

Attēlā 2.3. labajā pusē ir parādīta lietošanas gadījumu soļu topoloģija atbilstoši cēloņu-seku attiecībām starp funkcionālajām īpašībām (lietošanas gadījumu soļu numuri tiek izmantoti arī funkcionālajām īpašībām, lai atvieglotu uzskatāmību). IDM iesaka vairākas iterācijas starp lietošanas gadījumiem un TFM, līdz sistēmanalītiķis var verificēt TFM atbilstību priekšmetiskajai videi. Ar treknajām bultām tiek apzīmēts TFM galvenais funkcionēšanas cikls. Galveno funkcionēšanas ciklu nosaka sistēmanalītiķis. Attēlā 2.3. parādītajā piemērā galvenais funkcionēšanas cikls sastāv no šādām funkcionālajām īpašībām: 32 – klients parāda lasītāja kartiņu; 33 – bibliotekārs apstiprina lasītāja statusu; 21 – klients iedod grāmatu bibliotekāram; 22 – bibliotekārs pārbauda grāmatas stāvokli; 19 – klients meklē grāmatu katalogā; 8 – bibliotekārs izsniedz pieprasījuma formu; 9 – klients aizpilda pieprasījuma formu; 10 – bibliotekārs izņem grāmatu no grāmatu noliktavas; 11 – bibliotekārs izsniedz grāmatu klientam. Galvenais funkcionēšanas cikls ietver visus šīs bibliotēkas galvenos procesus, sākot no klienta atnākšanas uz bibliotēku, grāmatas atdošanu un jaunas grāmatas pieprasīšanu. Šajā TFM ir arī citi cikli, ieskaitot reģistrāciju vai grāmatas pieprasīšanu bez atdošanas. Galveno funkcionēšanas ciklu uzstāda sistēmanalītiķis, balstoties uz priekšmetiskās vides ekspertiem. Šajā bibliotēkas piemērā sistēmanalītiķis galvenā funkcionēšanas cikla noteikšanā manuāli izveidoja cēloņu-seku attiecības starp funkcionālajām īpašībām nr. 11 un 32 (kas atzīmētas attēlā 2.3. ar raustītu bultu). Šo cēloņu-seku attiecību noteikt automātiski nav iespējams, jo tā nav norādīta lietošanas gadījumos.

Vēl viens sistēmu analītiķa uzdevums ir noteikt cēloņu-seku attiecības starp funkcionālajām īpašībām, kas nāk no dažādiem lietošanas gadījumiem. Tam tiek izmantoti lietošanas gadījumu priekšnosacījumi un pēcnosacījumi. Lietošanas gadījuma priekšnosacījumi tiek lietoti, lai sasaistītu konkrēto soli ar soļiem no citiem lietošanas gadījumiem (taču nav aizliegts izmantot arī soli no tā paša lietošanas gadījuma). Piemēram, lietošanas gadījuma «Pieprasīt grāmatu» (angļu valodā – «*requesting a book*») priekšnosacījums ir «Bibliotekārs apstiprina lasītāja statusu» (angļu valodā – «*librarian authorizes reader status*»), kas ir solis nr. 3 no lietošanas gadījuma «Ierašanās» (angļu valodā – «*arriving*»). Ja lietošanas gadījuma soļi atspoguļo vienu un

to pašu funkcionālo īpašību, tas jāņem vērā visās attiecībās, kas tiek izveidotas atbilstoši šiem soļiem.

2.4. Kopsavilkums

IDM pieeja tiek piedāvāta kā solis pretī *MDA* dzīves cikla uzlabošanai (ar uzsvaru uz tā sākumu – *CIM*), tādējādi paverot iespējas automatizēt sistēmu analīzi un programmatūras izstrādi. *IDM* pieeja dod iespēju lietot ontoloģiju programmatūras izstrādei, izmantojot esošās ontoloģijas izstrādes metodes. Šī pieeja paredz ontoloģijas izmantošanu priekšmetiskās vides modelēšanai, veicot lietošanas gadījumu validāciju ar dabīgās valodas apstrādes metodēm (*NLP*). Tādējādi tiek apvienotas deklaratīvās un procedurālās zināšanas priekšmetiskās vides modelēšanai. *IDM* pieeja integrē biznesa vidē vispārpieņemtus standartus – lietošanas gadījumus un ontoloģiju, un tad piedāvā iespēju ģenerēt priekšmetiskās vides modeli automātiski. Galvenā *IDM* pieejas inovācija ir tieši šis automatizācijas līmenis un IT standartu atkalizmantošana. Tas ir iespējams, pateicoties topoloģiskā funkcionēšanas modeļa (TFM) izmantošanai par priekšmetiskās vides modeli.

Taču mūsdienās nav jēgas pieejai, kurai nav rīku atbalsta. Tādēļ šī pieeja un modeļu transformācija priekšmetiskās vides modelēšanai tika veidota tā, lai būtu iespējamo tai izstrādāt atbalstošo rīku kopu. Šajā nodaļā tika apspriesti un atrisināti visi nepieciešami priekšnosacījumi un uzdevumi IDM rīku kopas izstrādei.

3. ATBILSTOŠĀ RĪKU KOPA UN LIETOJUMS

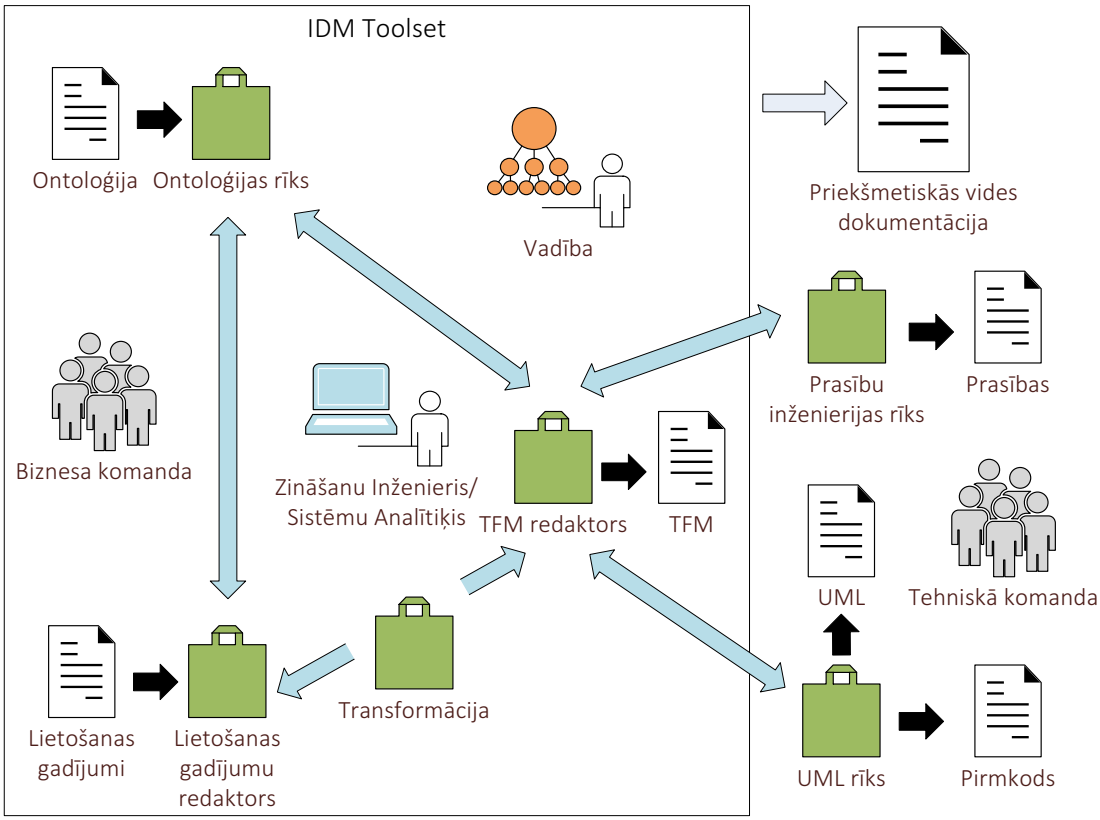
Integrētā priekšmetiskās vides modelēšanas (*IDM*) pieeja piedāvā elegantu risinājumu sarežģītajam priekšmetiskās vides modelēšanas procesam. Taču ir nepieciešams izstrādāt pieejas atbalstošu rīku kopu, lai šo pieeju varētu lietot programmatūras izstrādē reālā biznesa vidē. Šiem rīkiem jāatbilst *MDA* standartiem un jābalstās uz *MDA* ietvariem, lai tos būtu iespējams attīstīt un integrēt ar citiem *MDA* rīkiem.

3.1. IDM rīku kopas apgabals un arhitektūra

IDM rīku kopai jāatbalsta lietotājs šādu uzdevumu veikšanai: 1) izstrādāt ontoloģiju vai izmantot eksistējošu ontoloģiju; 2) izstrādāt lietošanas gadījumus; 3) veikt lietošanas gadījumu validāciju, izmantojot dabīgās valodas apstrādes metodes un ontoloģiju; 4) uzģenerēt TFM atbilstoši lietošanas gadījumiem; 5) veikt turpmāku

priekšmetiskās vides modelēšanu, piemēram, nosakot TFM galveno funkcionēšanas ciklu; 6) veikt visu modeļu validāciju atbilstoši to meta-modeļiem.

Šīs rīku kopas lietotāji ir zināšanu inženieris un/vai sistēmanalītiķis (tā var būt arī viena un tā pati persona). Viņu uzdevums ir strādāt kopā ar biznesa cilvēkiem, lai apkopotu un pārbaudītu priekšmetiskas vides zināšanas, kā arī saskaņotu iegūto priekšmetiskās vides modeli. Attēlā 3.1. autors ir parādījis rīkus, artefaktus un lomas, kas piedalās *IDM* rīku kopas procesā. Dokumenta ikona apzīmē artefaktus – ontoloģija, lietošanas gadījumi, *TFM*, *UML*, pirmkods, prasības un dokumentācija. Portfeļa ikona apzīmē rīkus – ontoloģijas rīks, lietošanas gadījumu rīks, transformācijas rīks, *TFM* rīks, prasību inženierijas rīks, *UML* rīks.

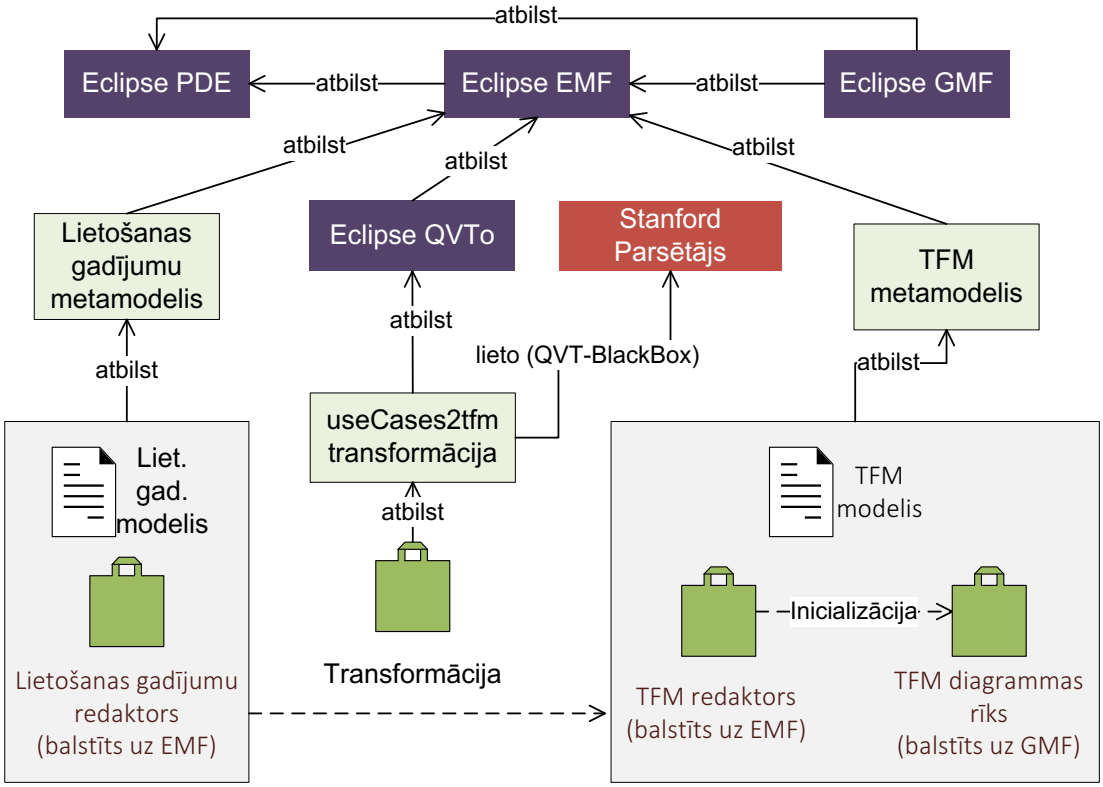


3.1. att. *IDM* rīku kopas apgabals

Melnās bultas norāda uz rīku artefaktiem. Zilās bultas norāda uz rīku sadarbību. Iesaistītās lomas tiek apzīmētas ar personu ikonām – zināšanu inženieris, sistēmanalītiķis, biznesa komanda, uzņēmuma vadība, tehniskā komanda. Pelēkā bulta apzīmē attiecības starp *IDM* rīku kopu un dokumentāciju.

Autors ir izstrādājis speciālus meta-modeļus lietošanas gadījumiem un TFM atbilstoši *MOF* standartam *IDM* rīku kopām. Attēlā 3.2. autors skaidro *IDM* rīku kopas arhitektūru. Portfeļa ikona apzīmē rīkus – lietošanas gadījumu rīks, transformācijas rīks,

TFM diagrammas rīks. Bultas norāda atbilstību meta-modeļiem, Eclipse ietvariem vai papildu Java bibliotēkām. Raustītās bultas norāda transformācijas virzienu, kā arī diagrammas inicializāciju. Arhitektūrā nav iekļauts ontoloģijas rīks, jo ontoloģijai tiek izmantots eksistējošs rīks *Protégé* [75]. IDM rīku kopai tiek izmantoti šādi *Eclipse* ietvari – *Eclipse* spraudņu izstrādes vide (*PDE*), *Eclipse* modelēšanas ietvars (*EMF*), grafiskais modelēšanas ietvars (*GMF*) un *QVTo*. *EMF* un *GMF* ietvari dod iespēju ģenerēt *Eclpise* spraudņu sagataves, tāpēc uz tām balstītie rīki atbilst *PDE*, kas vēlāk var tikt izmantoti funkcionalitātes papildināšanai. Papildus tiek izmantota Stenfordas statistiskā parsētāja [88] Java bibliotēka dabīgās valodas apstrādei. *QVT*/BlackBox mehānisms tiek izmantots citu bibliotēku integrēšanai (ne *QVT*). Autors izstrādāja šādus rīkus, kas kopā veido *IDM* rīku kopu – lietošanas gadījumu rīks, *TFM* rīks, *TFM* diagrammas rīks, transformācijas rīks.

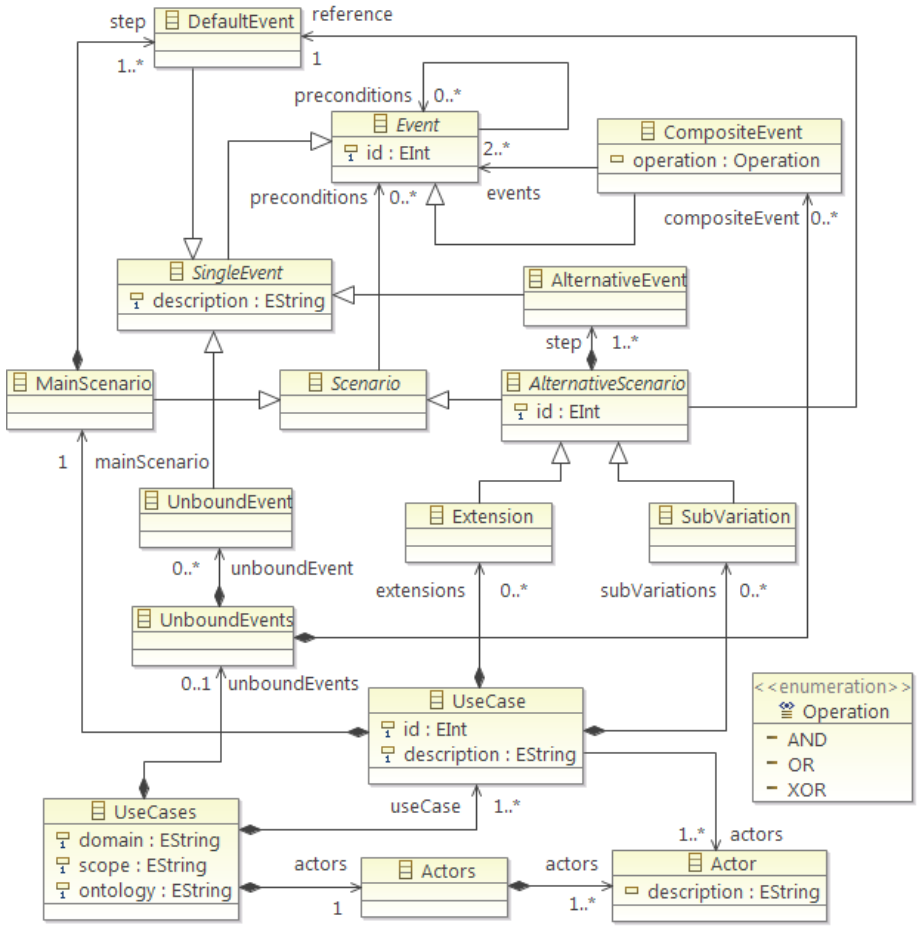


3.2. att. *IDM* rīku kopas arhitektūra

3.2. Lietošanas gadījumu rīks

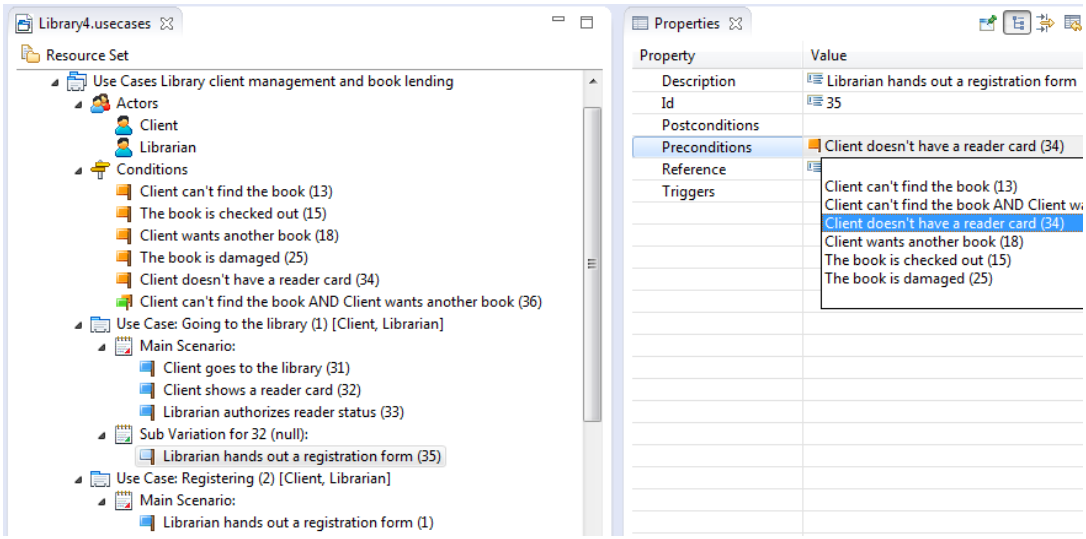
Lietošanas gadījumu rīku autors izstrādāja kā *Eclipse* spraudni, kas balstās uz *EMF*. Pirmais svarīgais solis lietošanas gadījumu rīka izstrādē ir lietošanas gadījumu meta-modeļa izstrāde. Attēlā 3.3. ir parādīts lietošanas gadījumu metamodelis atbisltoši *MOF* standartam, kuru izstrādājis autors. Lietošanas gadījumu kopa sastāv no viena vai

vairākiem lietošanas gadījumiem, kam ir galvenais scenārijs, paplašinājumi un apakšvariācijas, kas sastāv no lietošanas gadījumu soļiem.



3.3. att. Lietošanas gadījumu meta modelis

Katram lietošanas gadījumam ir nosaukums, aktieru saraksts (vismaz viens aktieris) un var būt priekšnosacījumi. Katram lietošanas gadījuma solim ir tā numurs (identifikators), un tam arī var būt priekšnosacījumi. Paplašinājumiem un apakšvariācijām ir paredzēts atsauces atribūts, lai norādītu, uz kuru soli galvenajā scenārija tas atsaucas.

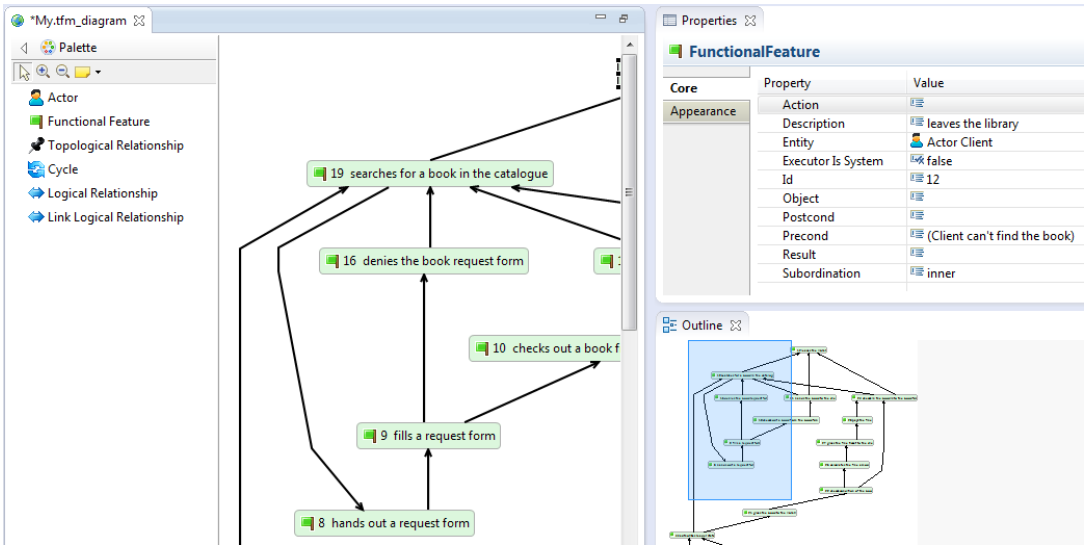


3.4. att. Lietošanas gadījumu rīks

Attēlā 3.4 parādīts lietošanas gadījumu rīks. Kreisajā pusē ir lietošanas gadījumu modelis, kas izstrādāts bibliotēkas biznesa sistēmai. Labajā pusē ir viena lietošanas gadījuma soļa rediģēšanas skats, kurā iespējams iestatīt, piemēram, priekšnosacījumus. Katram lietošanas gadījumu meta-modeļa elementam atbilst sava ikona, lai lietotājam būtu ērtāk šo rīku izmantot. Šis rīks dod iespēju lietotājam definēt aktierus, nosacījumus, saliktus nosacījumus, lietošanas gadījumus, galvenos scenārijus, alternatīvos scenārijus (paplašinājumus un apakšvariācijas), kā arī to soļus. Izveidotā lietošanas gadījumu modeļa datnes paplašinājums ir «*.usecases*». Šo datni ir iespējams atvērt arī ar parastu teksta redaktoru un tādā veidā redzēt šo modeli *XMI* formātā.

3.3. TFM rediģēšanas un diagrammas rīki

TFM rīks no lietotāja saskarnes viedokļa izskatās līdzīgi lietošanas gadījumu rīkam, jo tas arī balstās uz *Eclipse* EMF ietvaru. Turpretī TFM diagrammas rīks papildus balstās arī uz *Eclipse* GMF ietvaru, un tā saskarne dod lietotājam iespēju apskatīt un veidot TFM diagrammu. Attēlā 3.5. ir parādīts izstrādātais TFM diagrammas rīks, kuram sagatave tika ģenerēta atbilstoši *GMF* darba plūsmai un TFM metamodelim. Šī sagatave (kas būtībā ir strādājoša rīka pirmkods) pēc tam tiek papildināta ar nepieciešamo funkcionalitāti. Šī rīka funkcionalitāte nodrošina šādu elementu veidošanu – aktieri, funkcionālās īpašības, topoloģiskās īpašības, cikli un loģiskās operācijas. Iegūtajam TFM modelim datnes paplašinājums ir «*.tfm*», bet TFM diagrammai «*.tfm_diagram*». Abas šīs datnes iespējams atvērt ar teksta redaktoru un redzēt šo modeļu atbilstošo *XMI*.



3.5. att. TFM diagrammas rīks

3.4. Transformācija no lietošanas gadījumiem uz TFM

Modeļu transformācija no lietošanas gadījumiem uz TFM ir izstrādāta kā *Eclipse* spraudne, balstoties uz *Eclipse QVTo* un Stenfordas parsētāju [75]. Lai palaistu transformāciju, lietotājam uz izstrādātā lietošanas gadījuma modeļa (datne ar paplašinājumu «*.usecases*») ir jāuzklikšķina ar peles labo taustiņu un jāizvēlas komanda «*Transform to TFM*» (no angļu valodas – transformēt uz TFM). Pēc tam konsolē tiek izvadīts modeļu transformācijas žurnāls, kas parāda identificētās lietošanas gadījumu soļu atsauces (kad vairāki soļi atbilst vienai funkcionālajai īpašībai) ar atbilstošajiem identifikatoriem, izveidotos aktierus, izveidotās funkcionālās īpašības un izveidotās topoloģiskās īpašības. Pēc transformācijas pabeigšanas tajā pašā *Eclipse* projektā parādīsies jauna datne ar paplašinājumu «*.tfm*», tātad TFM modelis. Izmantojot šo TFM modeli, iespējams arī inicializēt tam atbilstošo diagrammu un iegūt datni ar paplašinājumu «*.tfm_diagram*».

3.5. Kopsavilkums

Šajā nodaļā tika aprakstīta *IDM* pieejas atbalstošā rīku kopa, parādot tās arhitektūru, lietotās tehnoloģijas, izstrādi, modeļu transformāciju un lietošanu. *IDM* rīku kopa dod iespēju definēt biznesa procesus, izmantojot lietošanas gadījumus, un tad ģenerēt tiem atbilstošo priekšmetiskās vides modeli TFM formā. Šī rīku kopa nodrošina iespēju sistēmanalītiķim iegūt formālu priekšmetiskās vides modeli. Tādējādi programmatūras izstrādes sākumā ir iespējam validēt biznesa procesu, nosakot tā atbilstību ontoloģijai, nosakot tā apgabalu atbilstoši ieejām un izejām, kā arī validējot to atbilstoši funkcionēšanas cikliem.

4. INTEGRĒTĀS PRIEKŠMETISKĀS VIDES MODELĒŠANAS PIEEJAS APROBĀCIJA

Šī nodaļa apraksta *IDM* pieejas un rīku kopas gadījuma izpēti, kas tik veikta pierakstīšanas komercijas biznesam. Šī gadījuma izpēte balstās uz *Pearl Consulting AS* [71] projektu klientam, kas nodarbojas ar pierakstīšanās komerciju Norvēģijā, Zviedrijā, Dānijā, Somijā, kā arī Apvienotajā Karalistē. Konkrēts kompānijas vārds netiks minēts, taču tā vietā tiks izmantots saīsinājums *SCB* (angļu valodā – *subscription commerce business*, latviski – pierakstīšanās komercijas bizness). Integrētais *SCB* risinājums sastāv no šādām komponentēm: *ERP*, *CRM*, *BI* un e-komercijas. Šīs komponentes balstās uz SAP [77] programmatūras produktiem: SAP ERP, SAP CRM, SAP BW, SAP PI un Hybris. Atbilstoši *IDM* pieejai vispirms tika izstrādāta ontoloģija, tad lietošanas gadījumi, pēc tam lietošanas gadījumi tika manuāli validēti atbilstoši ontoloģijai, un, visbeidzot, tika uzģenerēts sākotnējais TFM, kas vēlāk tika analizēts un uzlabots.

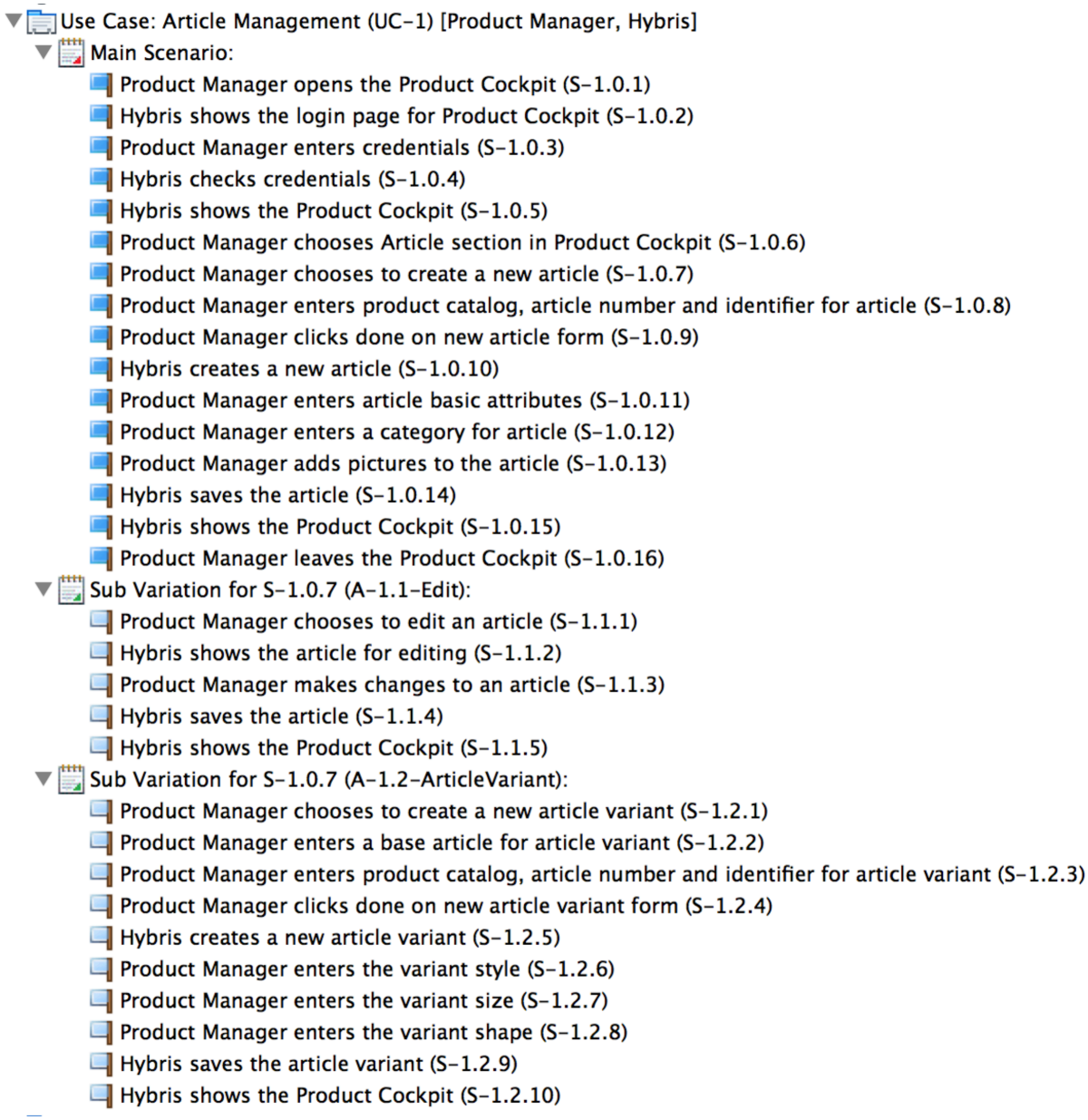
4.1. Biznesa priekšmetiskā vide

SCB piedāvā saviem klientiem pierakstīties uz dažāda veida produktiem, kas pēc pierakstīšanās tiek nosūtīti klientiem uz mājām pa pastu noteiktos laika intervālos. Svarīga *SCB* biznesa sastāvdaļa ir produktu vadība, kas sastāv no preču vienību vadības, paciņu vadības, produktu vadības un pierakstīšanās likumu vadības. Lai izskaidrotu *SCB* produktu uzbūvi, autors piedāvā šādu piemēru. *SCB* klientiem ir iespēja pierakstīties uz šādu produktu – «*Wilkinson Hydro 5*», kas ir skūšanās piederumu produkts. Tas sastāv no vairākām paciņām – parauga paciņas, pierakstīšanās paciņas un periodiskās paciņas. Parauga paciņa sastāv no 2 preču vienībām – *Wilkinson Hydro 5* skuveklis un *Wilkinson* skūšanās putas. Šā parauga paciņa tiks nosūtīta klientam bez maksas pēc pierakstīšanās (vai pēc pieraksta atjaunošanas). Pierakstīšanās paciņa savukārt sastāv no 1 preču vienības – *Wilkinson* asmeņiem, kas tiks sūtīti klientam par maksu ik pēc 3 mēnešiem. Klientu pierakstīšanās produktiem notiek *SCB* e-komercijas mājaslapā.

4.2. Lietošanas gadījumu izstrāde

SCB projektējamajam biznesa procesam tika identificēti šādi 9 lietošanas gadījumi: 1) preču vienību vadība; 2) paciņu vadība; 3) produktu vadība; 4) klientu pierakstīšanās; 5) paciņu piegāde; 6) iepirkums; 7) mājaslapas satura vadība; 8) kampaņu vadība; 9) klientu segmentācija. Attēlā 4.1. ir parādīts ekrānuzņēmums no IDM lietošanas gadījumu rīka, kurā ir izveidots preču vienību vadības lietošanas gadījums «UC-1». Šajā lietošanas gadījumā ir 2 aktieri – produktu vadītājs un *Hybris*.

Šim lietošanas gadījumam ir viens galvenais scenārijs un 2 apakšvariācijas – «*A-1.1-Edit*» un «*A-1.2-ArticleVariant*».



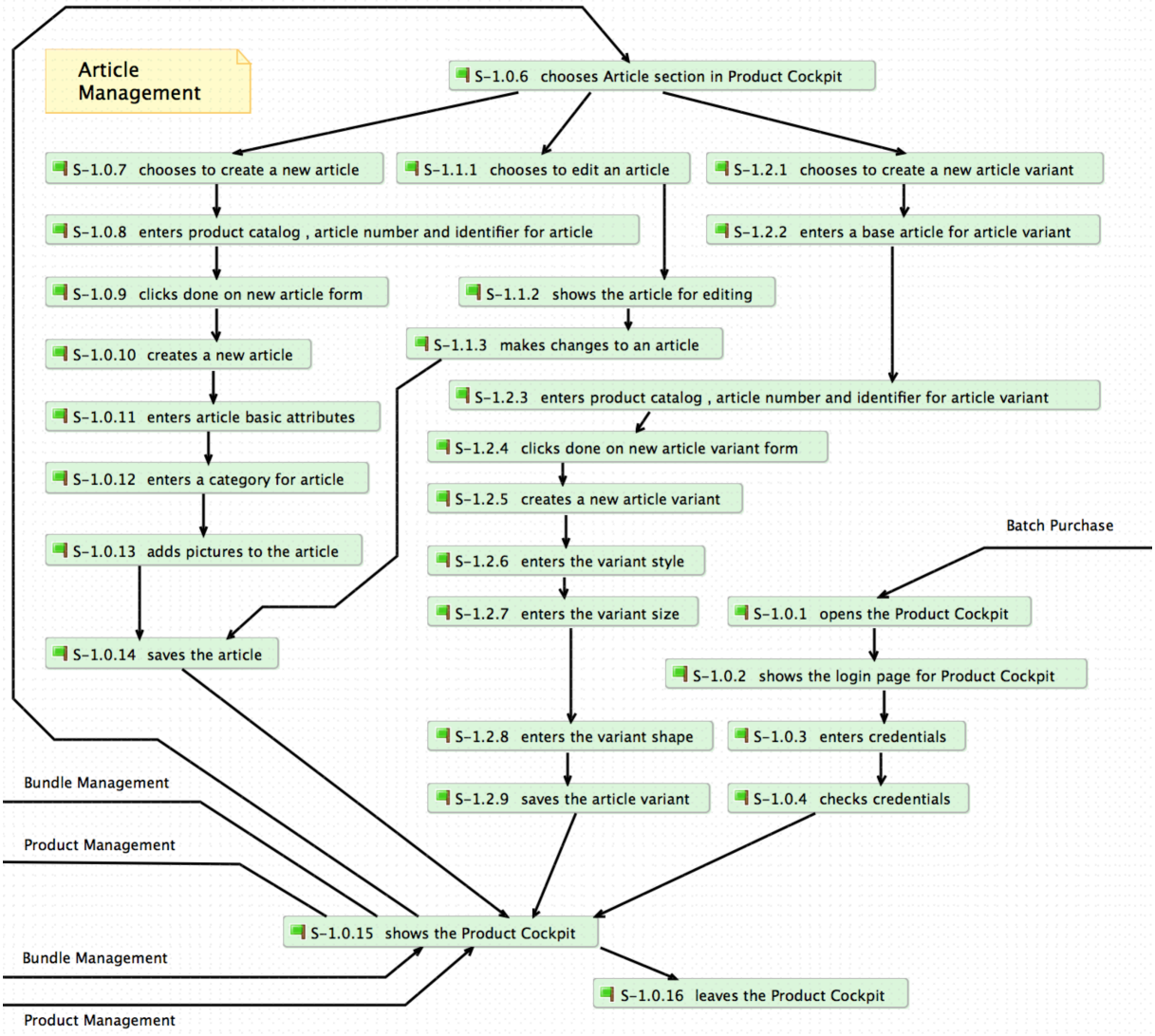
4.1. att. Preču vienību vadības lietošanas gadījums (izstrādāts ar *IDM* rīku kopu)

Galvenais scenārijs apskata, kā produktu vadītājs ieiet *Hybris* produktu vadības vietnē un izveido jaunas preču vienības ar atbilstošajiem atribūtiem. Pirmā apakšvariācija apraksta alternatīvu scenāriju, kurā produktu vadītājs nevis taisa jaunu preču vienību, bet rediģē esošo. Otrā apakšvariācija apraksta scenāriju, kurā produktu vadītājs veido preču vienības variantu, balstoties uz esošu preču vienību. Alternatīvie scenāriji norāda uz alternatīvu notikumu virkni. Piemēram, apakšvariācija «*A-1.1-Edit*» atsaucas uz soli «*S-1.0.7*», kas nosaka: «Produktu vadītājs izvēlas veidot jaunu preču

vienību» (attēlā teksts angļu valodā). Šis alternatīvais scenārijs ir apakšvariācija (alternatīvā scenārija veids), tāpēc jaunā notikumu virkne būs paralēla solim, uz kuru norādīta atsauce (to varēs redzēt attēlā 4.2.). Alternatīvais scenārijs beidzas ar soli «*S-1.1.3*», kas nosaka «*Hybris* parāda produktu vadības vietni».

4.3. TFM izgūšana

Kad lietošanas gadījumu izstrāde (vai vismaz pirmā iterācija) ir pabeigta un tie ir veiksmīgi novalidēti atbilstoši ontoloģijai, ir iespējams veikt modeļu transformāciju, izmantojot IDM rīku kopu. Šī transformācija automātiski ģenerē TFM modeli, kas atbilst definētajiem lietošanas gadījumiem. Attēlā 4.2. ir parādīts TFM *SCB* preču vienību vadības procesam, kas atbilst preču vienību vadības lietošanas gadījumam (no attēlā 4.1.).



4.2. att. Preču vienību vadība TFM formā (*IDM* rīku kopa)

Pēc TFM iegūšanas sistēmanalītiķim papildus uzģenerētajām topoloģiskajām attiecībām jāsavieno procesi, kas atbilst atsevišķiem lietošanas gadījumiem. Tādējādi tiek veidoti funkcionēšanas cikli. Vēl sistēmanalītiķim ir jādefinē galvenais funkcionēšanas cikls. *SCB* biznesa procesam galvenais funkcionēšanas cikls iet cauri šādiem procesiem – iepirkums, klienta pierakstīšanās un paciņas piegāde. Šie 3 procesi atspoguļo *SCB* biznesa ikdienas darbību. Lai šos procesus savienotu, sistēmanalītiķim bija nepieciešams izveidot 3 papildu topoloģiskās attiecības, savienojot atbilstošās funkcionālās īpašības. Papildus galvenajam funkcionēšanas ciklam pastāv arī cikls produktu veidošanai un publicēšanai, kas aptver šos procesus – preču vienību vadība (kas parādīts attēlā 4.2.), paciņu vadība un produktu vadība. Lai savienotu šo ciklu ar galveno funkcionēšanas ciklu, sistēmanalītiķim nepieciešams izveidot 2 papildu topoloģiskās attiecības, savienojot iepirkumu lietošanas gadījumu ar preču vienību vadības lietošanas gadījumu, kā arī produktu vadības lietošanas gadījumu ar klienta pierakstīšanās lietošanas gadījumu (savienojot atbilstošās funkcionālās īpašības). Tātad, lai pabeigtu TFM modeli, sistēmanalītiķim ir manuāli jāizveido tikai 5 topoloģiskās attiecības (pārējās IDM rīku kopa ir uzģenerējusi automātiski).

4.4. Kopsavilkums

Sākotnējais TFM priekšmetiskajai videi tika iegūts automātiski, lietojot *IDM* rīku kopu, balstoties uz lietošanas gadījumiem. Izmaiņas TFM modelī arī tika veiktas, izmantojot *IDM* rīku kopu. Ontoloģija, lietošanas gadījumi un TFM tika iekļauti projekta dokumentācijā, un, pēc tās apstiprināšanas, šī dokumentācija tika izmantota projekta izstrādes posmā. *IDM* pieeja un rīku kopa kalpoja par pamatu priekšmetiskās vides analīzei un modelēšanai.

Salīdzinot *IDM* pieeju ar *TFM4MDA*, sākumu abām pieejām varētu uzskatīt par līdzīgu, taču nepieciešamie artefakti ir atšķirīgi. *IDM* pieeja sākas ar ontoloģiju un lietošanas gadījumiem, savukārt *TFM4MDA* – ar neformālu priekšmetiskās vides aprakstu dabīgajā valodā. No patērētā laika viedokļa *TFM4MDA* prasa mazāk pūļu sākumā, taču *IDM* piedāvā labāku kvalitāti, jo tai ir labāka struktūra. Taču īstais *IDM* pieejas spēks ir TFM iegūšanā, kas notiek automātiski ar modeļu transformāciju. *IDM* ieved zināmu automatizācijas līmeni priekšmetiskās vides izstrādes procesā un ietaupa nepieciešamo izstrādes laiku, vienlaikus uzlabojot modeļu kvalitāti.

SECINĀJUMI

Šīs disertācijas mērķis ir uzlabot priekšmetiskās vides analīzes procesu, piedāvājot jaunu pieeju un atbalstošo rīku kopu formāla priekšmetiskās vides modeļa iegūšanai, kas būtu transformējams un varētu tikt izmantots kā *CIM MDA* kontekstā. Galvenais šī darba rezultāts ir *IDM* pieejas un atbalstošās rīku kopas izstrāde, kas dod iespēju definēt biznesa procesus, izmantojot lietošanas gadījumus, validēt tos atbilstoši ontoloģijai, kā arī automātiski ģenerēt formālu priekšmetiskās vides modeli TFM formā. Visi izvirzītie uzdevumi mērķa sasniegšanai ir veiksmīgi atrisināti un ir iegūti šādi rezultāti, kā arī izdarīti šādi secinājumi.

1. Esošo priekšmetiskās vides modelēšanas pieeju analīzes rezultāti ir šādi:
    a. priekšmetiskās vides modelis ir paredzēts biznesa cilvēkiem, tāpēc tam būtu jābūt viegli uztveramam bez papildu apmācības, taču esošās pieejas ir sarežģītas, kas var novest pie kļūdām priekšmetiskās vides modelī;
    b. dažas no esošajām pieejam ir stipras deklaratīvajās zināšanās, bet citām stiprā puse ir procedurālās zināšanas, taču no apskatītajām pieejām neviena nav stipra abos zināšanu veidos;
    c. lietošanas gadījumi ir vienkārša pieeja procedurālo zināšanu definēšanai, kā arī to viegli uztver biznesa cilvēki, taču tie tiek definēti dabīgajā valodā, un tādēļ var būt neatbilstoši, divdomīgi, grūti pārvaldāmi, un grūti transformējami;
    d. pastāv dažas inovatīvas pieejas, kas lieto dabīgās valodas apstrādes metodes (*NLP*), lai analizētu tekstus priekšmetiskās vides modelēšanai, taču šīs pieejas balstās uz neformāliem aprakstiem, kas ir pārāk nestrukturēti un grūti pārvaldāmi, lai tos varētu izmantot par pamatu priekšmetiskās vides modelim;
    e. rīku atbalsts ir ļoti svarīgs priekšmetiskās vides modelēšanai, lai iegūtu priekšmetiskās vides modeli, kuru tālāk būtu iespējams izmanto programmatūras izstrādē;
    f. priekšmetiskās vides modelēšanas pieejām ir salīdzinoši zems formalitātes līmenis, piedāvājot tikai modeļa metamodeli, taču TFM un ontoloģija balstās uz matemātiski formāliem modeļiem un piedāvā formālu modeļu validāciju.
2. TFM pieejas priekšrocību, trūkumu un potenciālo uzlabojumu analīzes rezultāti ir šādi:
    a. galvenā TFM priekšrocība ir tā matemātiskie pamati, jo tādējādi tas dod iespēju formālā veidā aprakstīt priekšmetisko vidi no procedurālā viedokļa;

b. vēl viena TFM priekšrocība ir tā ieejas un izejas, kad dod iespēju identificēt TFM apgabalu, nosakot robežu starp sistēmu un tās vidi;
c. trešā TFM priekšrocība ir tā ciklu struktūra un galvenais funkcionēšanas cikls, kas dod iespēju priekšmetiskās vides validācijai;
d. ceturtā TFM priekšrocība ir modeļu transformācija, kas vispirms definēta *TFM4MDA* pieejā un tad pilnveidota TopUML;
e. viens no TFM trūkumiem ir tas, ka tas apraksta tikai procedurālās zināšanas, bet deklaratīvās zināšanas tas neatbalsta;
f. vēl viens TFM trūkums (konkrēti *TFM4MDA*) ir tāds, ka šī pieeja balstās uz neformālu aprakstu dabīgajā valodā, kas tomēr ir pārāk grūti pārvaldāms un analizējams;
g. trešais trūkums ir tāds, ka TFM pieejām nav rīku atbalsta, un tādēļ vienīgais veids, kā izveidot TFM, ir smags manuāls process;
h. daži no iespējamajiem uzlabojumiem TFM pieejām ir neformālā apraksta aizstāšana ar formālām priekšmetiskās vides zināšanām, integrācijas realizācija ar deklaratīvajām zināšanām, modeļu transformācijas realizēšana ar atbilstošu rīku.

3. Autors ir izstrādājis integrēto priekšmetiskās vides modelēšanas (*IDM*) pieeju formāla priekšmetiskās vides modeļa iegūšanai TFM formā, balstoties uz formāli definētām zināšanām par priekšmetisko vidi, izmantojot lietošanas gadījumus un ontoloģiju. *IDM* izstrādē tika izmantota analīze par TFM pieeju un veikti ieteiktie uzlabojumi.
4. Autors ir izstrādājis *IDM* rīku kopu *IDM* pieejas atbalstam, balstoties uz *MDA* standartiem un *Eclipse EMF, GMF, M2M* ietvariem, kas sastāv no lietošanas gadījumu rīka, TFM rīka un transformācijas rīka. Izmantojot priekšmetiskās vides modeli, kas iegūts ar *IDM* rīku kopu, sistēmanalītiķis kopā ar biznesa cilvēkiem var validēt biznesa procesus pirms programmatūras izstrādes procesa sākuma. Tādējādi ir iespējams pārliecināties, ka biznesa procesi atbilst priekšmetiskajai videi un ontoloģijai. Atkļūdošana var tikt veikta vēl pirms programmatūras koda rakstīšanas.
5. Praktiskais lietojums e-komercijas programmatūras izstrādes projekta parādīja, kā *IDM* pieeja un rīku kopa var tikt veiksmīgi izmantota priekšmetiskās vides zināšanu apkopošanai, kā arī TFM iegūšanai; tajā pašā laikā iegūstot grafiski biznesa procesu atspoguļojumu automātiskā veidā. Secinājumi pēc veiktās gadījuma izpētes ir šādi:
    a. TFM rīku kopa ir intuitīva un tās apgūšana nepatērē vairāk par 1 dienu, jo vienīgais obligātās priekšzināšanas ir lietošanas gadījumi;

b. *IDM* lietošanas gadījumi ir ērts veids, kāda dokumentēt biznesa procesu programmatūras izstrādes projekta sākumā, jo tos saprot gan tehniskā komanda, gan biznesa komanda bez apmācības;
c. vispārīga ontoloģija palīdz priekšmetiskās vides analīzes procesam, taču, iedziļinoties detaļās (piemēram, objektu īpašību noteikšana), prasa daudz laika, taču tas *IDM* pieejā nav obligāts solis (piemēram, *SCB* projektā būtu pieticis ar klašu hierarhiju, taču tas ir atkarīgs no projekta un komandas pieredzes);
d. grafiska biznesa procesa atspoguļojuma iegūšana ir ātra un vienkārša, tiklīdz lietošanas gadījumi ir izstrādāti (TFM tiek iegūts automātiski, izmantojot modeļu transformāciju atbilstoši *MDA* standartiem);
e. TFM grafisko atspoguļojumu varētu uzlabot, nodalot procesus pēc to atbilstošajiem lietošanas gadījumiem un ieviešot aktieru celiņus.

Nākotnes pētījuma virzieni

- Esošajā *IDM* rīku kopā lietošanas gadījumu rīks atbalsta lietošanas gadījumu izstrādi, taču neatbalsta ontoloģijas augšupielādi un lietošanas gadījumu automātisku validāciju atbilstoši šai ontoloģijai. To ir iespējams izstrādāt, izmantojot Stenfordas statistisko parsētāju (tāpat kā modeļu transformācijā) lietošanas gadījumu analīzei, *OWL API* ontoloģijas augšupielādei un *Eclipse EMF* validācijas ietvaru lietošanas gadījumu modeļa validācijai.
- Būtu nepieciešams integrēt *IDM* rīku kopu un kādu no *UML* rīkiem, realizējot TopUML pieejas aprakstītās modeļu transformācijas no TFM uz *UML*.

LITERATŪRAS SARAKSTS

[1] Alphabetical list of part-of-speech tags used in the Penn Treebank Project / Internet. - https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html [Accessed: June 14, 2014]
[2] ARIS / Internet. – http://www.ariscommunity.com/ [Accessed: June 14, 2014]
[3] Arlow J., Neustadt I. Introduction to BPMN 2. Mountain Way Limited, 2012. – 182 p.
[4] Asnina E. The Formal Approach to Problem Domain Modeling within Model Driven Architecture// In 9th International Conference on Information Systems Implementation and Modelling. – Prerov, Czech Republic, Ostrava, 2006. – pp. 97–104.

[5] Asnina E., Osis J. Topological Functioning Model as a CIM-Business Model// Model-Driven Domain Analysis and Software Development: Architectures and Functions. – IGI Global, 2011. – pp. 40–64.
[6] Baclawski K., Kokar M. K., Kogut P., Hart L., Smith J. E., Letkowski J., Emery P. Extending the Unified Modeling Language for Ontology Development// Int. Journal Software and Systems Modeling (SoSyM). – 2002. – Vol. 1, No. 2. – pp. 142–156.
[7] Broy M. Domain Modeling and Domain Engineering: Key Tasks in Requirements Engineering// Perspectives on the Future of Software Engineering. – Springer Berlin Heidelberg, 2013. – pp. 15–30.
[8] Business Process Model and Notation (BPMN) version 2.0.2 / Internet. – omg.org/spec/BPMN/2.0.2/PDF [Accessed: June 14, 2014]
[9] Caliusco M. L., Galli M. R., Ruidías H. J. Towards the integration of ontologies in the context of MDA at CIM level// XVIII Congreso Argentino de Ciencias de la Computación. – 2012.
[10] Coleman D. A. Use Case Template: draft for discussion// Fusion Newsletter. – April 1998. Available. – http://www.engr.sjsu.edu/~fayad/current.courses/cmpe202-Fall2009/docs/lecture3/CmpE202-22-UC-Template-Ex-L3-3g.pdf
[11] Cranefield S. Networked knowledge representation and exchange using UML and RDF// Journal of Digital Information. – 2001. – Vol. 1, No. 8. Available. – https://journals.tdl.org/jodi/article/viewArticle/30/31 [Accessed: June 14, 2014]
[12] Donins U. Software Development with the Emphasis on Topology// In Proceeding of 13th East-European Conference on Advances in Databases and Information Systems (ADBIS 2009). – Volume 5968 of LNCS. Springer, 2010. – pp. 220–228.
[13] Doniņš U. Topological Unified Modeling Language: Development and Application. Doctoral thesis, Riga Technical University, 2012. – 224 p.
[14] Eclipse GMF: Graphical Modeling Framework / Internet. – http://wiki.eclipse.org/Graphical_Modeling_Framework [Accessed: June 14, 2014]
[15] Eclipse Papyrus / Internet. – http://www.eclipse.org/papyrus/ [Accessed: June 14, 2014]
[16] Eclipse Requirements Management Framework / Internet. – http://www.eclipse.org/rmf/ [Accessed: June 14, 2014]
[17] EMF Developer Guide: The Eclipse Modeling Framework (EMF) Overview. – 2005. / Internet. - http://help.eclipse.org/ganymede/index.jsp?topic=/org.eclipse.emf.doc/references/overview/EMF.html [Accessed: June 14, 2014]
[18] Evans E. Domain-driven Design: Tackling Complexity in the Heart of Software. – Addison-Wesley Professional, 2004. – 359 p.
[19] Falkovych K., Sabou M., Stuckenschmidt H. UML for the Semantic Web: Transfor-mation-Based Approaches// In Knowledge Transfor-mation for the Semantic Web. – IOS Press, 2003. – pp. 92–106. Available. – http://www.cwi.nl/~media/publications/UML_for_SW.pdf [Accessed: June 14, 2014]
[20] Fliedl G., Kop C., Mayr H. C., Salbrechter A., Vohringer J., Weber G., Winkler C. Deriving static and dynamic concepts from software requirements using

sophisticated tagging// Data & Knowledge Engineering. – 2007. – Vol. 61, Iss. 3. – pp. 433–448.
[21] Fouad A. Embedding requirements within model-driven architecture// Software Quality Journal 19.2. – 2011. – pp. 411–430.
[22] Francu J., Hnetynka P. Automated Generation of Implementation from Textual System Requirements// In Proceedings of the 3rd IFIP TC 2 CEE-SET. – Brno, Czech Republic, Wroclawskiej. – 2008. – pp. 15–28.
[23] Frankel D. S. Model Driven Architecture: Applying MDA to Enterprise Computing. Wiley, Indianapolis, 2003. – 352 p.
[24] Fuchs N. E., Kaljurand K., Kuhn T. Attempto Controlled English for Knowledge Repre-sentation// In Reasoning Web, Fourth International Summer School 2008, Lecture Notes in Computer Science 5224. – Springer, 2008. – pp. 104–124.
[25] Gasevic D., Djuric D., Devedzic V. Model Driven Architecture and Ontology Development. Springer, Heidelberg, 2006. – 311 p.
[26] Goodman N. D. Mathematics as an objective science// American mathematical monthly. – 1979. pp. 540–551.
[27] Google Scholar / Internet. – http://scholar.google.com [Accessed: June 14, 2014]
[28] Guarino N. Formal Ontology and Information Systems// In Proceedings of Formal Ontology and Information Systems. – Trento, Italy, IOS Press, Amsterdam, 1998. pp. 3–15.
[29] Jacobson I., Spence I. Use case 2.0: Scaling up, scaling out, scaling in for agile projects. Ivar Jacobson International, 2011. – 54 p.
[30] Jones C. Positive and negative innovations in software engineering. International Journal of Software Science and Computational Intelligence (IJSSCI) 1.2. – 2009. – pp. 20–30.
[31] Jurafsky D., Martin J. H. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. - Pearson Education, 2000. – 934 p.
[32] Kaindl H. Structural Requirements Language Definition, Defining the ReDSeeDS Languages / Internet. – http://publik.tuwien.ac.at/files/pub-et_13406.pdf
[33] Kardoš M., Drozdová M. Analytical method of CIM to PIM transformation in Model Driven Architecture (MDA)// Journal of Information and Organizational Sciences 34.1. – 2010. – pp. 89–99.
[34] Kaur A., Mansaf A. Role of Knowledge Engineering in the Development of a Hybrid Knowledge Based Medical Information System for Atrial Fibrillation// American Journal of Industrial and Business Management. Vol. 3., No 1. – 2013. – pp. 36–41.
[35] Kirikova M, Finke A., Grundspenkis J. What is CIM: an information system perspective// Advances in Databases and Information Systems. Vol. 5968. – Springer Berlin Heidelberg, 2010. – pp. 169–176.
[36] Kirikova M. Domain Modeling Approaches in IS Engineering. Model-Driven Domain Analysis and Software Development: Architectures and Functions. – IGI Global, 2011, pp. 388–406.
[37] Kolmogorov A. N., Fomin S. V. Introductory Real Analysis (Silverman, R. A., Ed.). – Mineola, NY: Courier Dover Publications, 1975. – 416 p.
[38] Kontio M. Architectural manifesto: Choosing MDA tools / Internet. – http://www.ibm.com/developerworks/library/wi-arch18.html [Accessed: June 14, 2014]

[39] Malan R., Bredemeyer D. Functional Requirements and Use Cases / Internet. – http://www.bredemeyer.com/pdf_files/functreq.pdf [Accessed: June 14, 2014]
[40] Mayr H. C., Kop C. A User Centered Approach to Requirements Modeling// In Modellierung. – 2002. – pp. 75–86.
[41] MDA: Model Driven Architecture / Internet. – http://www.omg.org/mda/ [Accessed: June 14, 2014]
[42] Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification / internet. – http://www.omg.org/cgi-bin/doc?ptc/07-07-07.pdf [Accessed: June 14, 2014]
[43] Miller J., Jishnu M. MDA Guide Version 1.0. 1. Object Management Group, 2003. – 51 p.
[44] Miller, Joaquin, and Jishnu Mukerji. «Model driven architecture (mda).» Object Management Group, Draft Specification ormsc/2001-07-01 (2001).
[45] Moore R., Lopes J. Paper templates// In TEMPLATE'06 1st International Conference on Template Production. – SciTePress, 1999.
[46] Murata T. Petri nets: Properties, analysis and applications// In Proceedings of the IEEE 77.4. – 1989. pp. 541–580.
[47] Nickols F. The Knowledge in Knowledge Management. The Knowledge Management Yearbook 2001–2002. – Butterworth-Heinemann, Boston, 2000. – pp. 12–21.
[48] Nikiforova, O., et al. «Development of the tool for transformation of the two-Hemisphere model to the UML class diagram: technical solutions and lessons learned.» Proceedings of the 5th International Scientific Conference «Applied Information and Communication Technology». 2012.
[49] Noy N. F., McGuinness D. L. Ontology development 101: A guide to creating your first ontology. Technical Report SMI-2001-0880, Stanford Medical Informatics. – 2001.
[50] OMG Meta Object Facility (MOF) Core Specification Version 2.4.2 OMG April 2014 / Internet. – http://www.omg.org/spec/MOF/ [Accessed: June 14, 2014
[51] OMG: Object Management Group / Internet. – omg.org [Accessed: June 14, 2014]
[52] Ondrej M, Richta K. The BPM to UML activity diagram transformation using XSLT// Dateso. – Vol. 9. – 2009. – pp. 119–129.
[53] Osis J. Investigating Troubles of Complex System Functioning and Category Theory // Cybernetic and Diagnosis, Volume. 4. – Riga: Zinatne, 1970. – pp. 15–20 (in Russian)
[54] Osis J. Software Development with Topological Model in the Framework of MDA// In Proceedings of the 9th CAiSE International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'2004) in connection with the CAiSE'2004. – Vol. 1. – RTU, Riga, 2004. pp. 211–220.
[55] Osis J., Asnina E. A Business Model to Make Software Development Less Intuitive// In Proceedings of the 2008 International Conference on Innovation in Software Engineering. – Vienna, Austria, IEEE Computer Society CPS, Los Alamitos, USA, 2008. – pp. 1240–1246.
[56] Osis J., Asnina E. Derivation of Use Cases from the Topological Computation Independent Business Model. Model-Driven Domain Analysis and Software Development: Architectures and Functions. – IGI Global, Hershey, New York, 2011. – pp. 65–89.

[57] Osis J., Asnina E. Enterprise Modeling for Information System Development within MDA// In 41th Annual Hawaii International Conference on System Sciences. – HICSS, USA, 2008. – pp. 490.
[58] Osis J., Asnina E. Is Modeling a Treatment for the Weakness of Software Engineering? Model-Driven Domain Analysis and Software Development: Architectures and Functions. – IGI Global, Hershey, New York, 2011. – pp. 1–14.
[59] Osis J., Asnina E. Model-Driven Domain Analysis and Software Development: Architectures and Functions. – IGI Global, Hershey, New York, 2011. – 487 p.
[60] Osis J., Asnina E. Topological Modeling for Model-Driven Domain Analysis and Software Development: Functions and Architectures. Model-Driven Domain Analysis and Software Development: Architectures and Functions. – IGI Global, Hershey, New York, 2011. – pp. 15–39.
[61] Osis J., Asnina E., Grave A. Formal Problem Domain Modeling within MDA// Communications in Computer and Information Science (CCIS). Software and Data Technologies. – Vol. 22. Springer-Verlag Berlin Heidelberg, 2008. – pp. 387–398.
[62] Osis J., Asnina E., Grave A. Computation Independent Modeling within the MDA// In Proceedings of the IEEE International Conference on Software Science, Technology and Engineering. – Herzlia, Israel, IEEE Computer Society, 2007. – pp. 22–34.
[63] Osis J., Asnina E., Grave A. Computation Independent Representation of the Problem Domain in MDA// J. Software Eng. – Vol. 2, iss. 1. – pp. 19–46. Available. – http://www.e-informatyka.pl/e-Informatica/Wiki.jsp?page=Volume2Issue1 [Accessed: June 14, 2014]
[64] Osis J., Asnina E., Grave A. Formal Computation Independent Model of the Problem Domain within the MDA. Information Systems and Formal Models// In Proceedings of the 10th International Conference ISIM'07. – Silesian University in Opava, Czech Republic, 2007. – pp. 47–54.
[65] Osis J., Asnina E., Grave A. MDA Oriented Computation Independent Modeling of the Problem Domain// In Proceedings of the 2nd International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2007). – Barcelona, Spain, 2007. – pp. 66–71.
[66] Osis J., Donins U. Formalization of the UML Class Diagrams// Evaluation of Novel Approaches to Software Engineering. – Springer-Verlag, Berlin Heidelberg, New York, 2010. – pp. 180–192.
[67] Ouvans C. From BPMN process models to BPEL web services// Web Services, 2006. ICWS'06. International Conference on. IEEE, 2006. – pp. 285–292.
[68] Overmyer S. Lavoie B. Rambow O. Conceptual Modeling through Linguistic Analysis Using LIDA// In Proceedings of 23rd International Conference on Software Engineering (ICSE 2001). – Toronto, Canada, 2001.
[69] Overmyer S., Lavoie B., Rambow O. Conceptual Modeling through Linguistic Analysis Using LIDA// In Proceedings of the 23rd International Conference on Software Engineering. – Toronto, Ontario, Canada, 2001. – pp. 401–410.
[70] OWL: Web Ontology Language / Internet. – http://www.w3.org/TR/owl2-quick-reference/ [Accessed: June 14, 2014]
[71] Pearl Consulting / Internet. – http://www.pearlconsulting.no

[72] Plante F. Introducing the GMF Runtime, 2006. / Internet. – http://www.eclipse.org/articles/Article-Introducing-GMF/article.html [Accessed: June 14, 2014]
[73] Poesio M. Domain modelling and NLP: Formal ontologies? Lexica? Or a bit of both?// Applied Ontology, Vol. 1, No. 1. IOS Press, 2005. – pp. 27–33.
[74] Prieto-Díaz R. Domain Analysis: An Introduction// ACM SIGSOFT Software Engineering Notes 15.2. – 1990. – pp. 47–54.
[75] Protege / Internet. – http://protege.stanford.edu [Accessed: June 14, 2014]
[76] Santorini B. Part-Of-Speech Tagging Guidelines for the Penn Treebank Project// Technical report MS-CIS-90-47, Department of Computer and Information Science, University of Penllsylvania, 1990.
[77] SAP AG / Internet. – http://www.sap.com
[78] Scheer A. W., Oliver T., Otmar A. Process modeling using event-driven process chains// Process-Aware Information Systems. – 2005. – pp. 119–146.
[79] Šlihte A. Implementing a Topological Functioning Model Tool// In Scientific Journal of Riga Technical University, 5. series., Computer Science. – Vol. 43. – Riga, 2010. – pp. 68–75.
[80] Šlihte A. Introduction to Integrated Domain Modeling Toolset // Scientific Journal of RTU. Computer Science. – 2014. (to be published)
[81] Šlihte A. The Concept of a Topological Functioning Model Construction Tool// In 13th East-European Conference, ADBIS 2009, Associated Workshops and Doctoral Consortium, Local Proceedings. JUMI, Riga, Latvia, 2009. – pp. 476–484.
[82] Šlihte A. The Specific Text Analysis Tasks at the Beginning of MDA Life Cycle// In Data-bases and Information Systems Doctoral Consortium. Latvia, Riga, July 5–7, 2010. – pp. 11–22.
[83] Šlihte A. Transforming Textual Use Cases to a Computation Independent Model// MDA & MTDD 2010. Greece, Athens, July 22–24, 2010. – pp. 33–42.
[84] Šlihte A. Using Use Cases for Domain Modeling// In Proceedings of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012) – 2012. – pp. 224–231.
[85] Šlihte A., Osis J., Doniņš U. Knowledge Integration for Domain Modeling// In Proceedings of the 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development. China, Bejing, June 8–11, 2011. – pp. 46–56.
[86] Soley R. Model driven architecture. OMG white paper, 2000. Available. – http://www.geocities.ws/pravin_suman/Resources/00-11-05.pdf [Accessed: June 14, 2014]
[87] Subramaniam K., Liu D., Far B., Eberlein A. UCDA: Use Case Driven Development Assistant Tool for Class Model Generation// In Proceedings of the 16th SEKE. Banff, Canada, 2004. Available. - http://enel.ucalgary.ca/People/eberlein/publications/SEKE-Kalaivani.pdf [Accessed: June 14, 2014]
[88] The Stanford Parser: A statistical parser. The Stanford Natural Language Processing Group, 2010 / Internet. – http://nlp.stanford.edu/software/lex-parser.shtml [Accessed: June 14, 2014]

[89] Van Lamsweerde A. Requirements engineering: from craft to discipline// In Proceedings of the 13th international Workshop on Early Aspects. – New York: Association for Computing Machinery, Inc, 2008. – pp. 238–249.
[90] W3C, OWL Web Ontology Language Overview, W3C Recommendation February 10 2004 / Internet. – http://www.w3.org/TR/owl-features/ [Accessed: June 14, 2014]
[91] White S. A., Introduction to BPMN. IBM Cooperation, 2004. Available. – http://yoann.nogues.free.fr/IMG/pdf/07-04_WP_Intro_to_BPMN_-_White-2.pdf [Accessed: June 14, 2014]
[92] XML Metadata Interchange (XMI) Specification, v2. 3.2, 2014 /Internet. – http://www.omg.org/spec/XMI/ [Accessed: June 14, 2014]
[93] Tsai A., Wang J., Tepfenhart W., Rosca, D: EPC workflow model to WIFA model conversion// In Proceedings of Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference, Vol. 4, pp. 2758–2763
[94] Suchanek F. M., Kasneci G., Weikum G. Yago: A large Ontology from wikipedia and wordnet// Web Semantics: Science, Services and Agents on the World Wide Web 6.3, 2008, pp. 203–217.
[95] Ferrario R., Oltramari A.: A first-order cutting process ontology for sheet metal parts// Formal Ontologies Meet Industry 198, 2009, pp. 22.
[96] Jones C.: Software project management practices: Failure versus success// CrossTalk: The Journal of Defense Software Engineering 17, 2004.
[97] Kruczynski K.: Business process modelling in the context of SOA–an empirical study of the acceptance between EPC and BPMN// World Review of Science, Technology and Sustainable Development 7.1, 2010, pp. 161–168.
[98] Nüttgens M., Feld T., Zimmermann V.: Business Process Modeling with EPC and UML: transformation or integration?// The Unified Modeling Language. Physica-Verlag HD, 1998, pp. 250–261.
[99] Strommer M, Murzek M., Wimmer M.: Applying model transformation by-example on business process modeling languages// Advances in Conceptual Modeling–Foundations and Applications, Springer Berlin Heidelberg, 2007, pp. 116–125.
[100] Bodrow W.: The dynamic of professional knowledge utilized in software applications for process controlling// Advances in Manufacturing, 2014, pp. 1–6.