RĪGAS TEHNISKĀ UNIVERSITĀTE
Datorzinātnes un informācijas tehnoloģijas fakultāte
Lietišķo datorsistēmu institūts

Ērika ASŅINA

PROBLĒMVIDES MODELĒŠANAS FORMALIZĀCIJA MODEĻU VADĀMĀ ARHITEKTŪRĀ

Promocijas darba kopsavilkums

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

RTU Izdevniecība
Rīga 2006

VISPĀRĪGS DARBA RAKSTUROJUMS

Problēmvides analīze ir viena no pirmajām svarīgajām aktivitātēm programmatūras izstrādē. Programmatūras izstrādātāju kopiena izmanto dažādas metodes problēmvides raksturojumu un prasību identificēšanai un specificēšanai. Jāatzīmē, ka problēmvides analīzes metodes galvenokārt ir orientētas uz lietojuma vides analīzi. Savukārt problēmvide bieži tiek apskatīta kā „melnā kaste", kura apraksta tikai dažus sistēmas aspektus. Tādās pieejās joprojām trūkst problēmvides zināšanu formāla attēlojuma lietojumprogrammās.

Aprakstītā situācija ierosina dažas problēmas. Pirmkārt, vissvarīgākais ir jautājums par nesaskaņotību starp lietojumprogrammu un reālo pasauli - kāda problēmvide jāmodelē vispirms: pētīšanas laikā eksistējošā realitāte vai realitāte, kuru grib redzēt pasūtītājs? Otra problēma ir aspektu atdalīšana un viennozīmīgs problēmvides apraksts specifikācijas. Galvenais, kas jāsaprot, ir tas, ka sistēmas prasības ir reālās pasaules parādību ierobežojumi, nevis otrādi. Tas nozīmē, ka, ja programmatūra tiek izstrādāta kādu mērķu sasniegšanai reālajā pasaulē, tad tās izstrādātājiem jāzina, kā tā ietekmēs šo pasauli. Tas ir būtiski svarīgi ne tikai mehatroniskām un iegultām sistēmām, kuru kļūdaina darbība var novest pie cilvēku bojāejas, bet arī biznesa sistēmām, kur kļūdas var novest pie milzīgiem finanšu zaudējumiem.

Formālas matemātiskas metodes varētu samazināt specifikāciju neprecizitāti un divdomību. Bet, atšķirībā no neformālām metodēm, tās parasti prasa papildus piepūli gan apmācībā, gan lietošanā. Tādēļ modelēšanas valodu izstrādātāji bieži vien mēģina izvairīties no vienīgi šādu metožu izmantošanas un piedāvā tā saucamās pusformālās modelēšanas valodas. Tās ir precīzi aprakstītas, bet neizmanto formālo matemātisko pamatu (kā piemēru var pieminēt vienoto modelēšanas valodu UML).

Temata aktualitāte ir saistīta ar programmatūras izstrādes kvalitāti un izstrādātās programmatūras atbilstību tās darba videi. Darba videi neatbilstoši izstrādāta programmatūra prasa lielus izdevumus neatbilstības izlabošanai. Pie tam modeļu vadīta programmatūras izstrāde dos ievērojamu labumu tikai tad, kad spēs nodrošināt lietojuma modeļu atbilstību problēmvidei, citādi tā būs tāda pati kā jau eksistējošās pieejas.

Promocijas darba **galvenais mērķis** ir ieviest vairāk formālisma problēmvides modelēšanā - Objektu pārvaldības grupas *(Object Management Group, OMG)* modeļu vadāmas arhitektūras *(Modei Driven Architecture, MDA)* ietvaros. Galvenā ideja ir formalizēt atbilstību starp problēmvidi (reālajā pasaulē) un tās attēlošanu lietojuma vidē (specifikācijas). Tas nozīmē ieviest formālu definīciju saskaņai starp reālās pasaules parādībām un darbam tajā paredzēto lietojumu, neapgrūtinot to ar sarežģītu, grūti saprotamu matemātiku.

Saskaņā ar mērķi tika izvirzīti šādi **uzdevumi:**

- Novērtēt objektorientētajā programmatūras izstrādē eksistējošos risinājumus problēmvides modelēšanai;
- Izpētīt formālas problēmvides analīzes metodes un to iespējas formalizācijai atbilstoši promocijas darba mērķim;
- Izstrādāt formālu pieeju, kura ļautu: a) izveidot formālu specifikāciju problēmvides raksturošanai, kuru iespējams ierobežot ar lietojuma sistēmas prasībām; b) izveidot formālu lietojuma specifikāciju, kurā augstā abstrakcijas līmenī būtu attēloti lietojumā īstenotie biznesa procesi, kuri būtu saskaņoti ar eksistējošās sistēmas biznesa procesiem;
- Salīdzināt izstrādātās pieejas galvenās īpašības ar vienotu procesu *(Unified Process),* biznesa objektorientēto modelēšanu *(Business Object Oriented Modeling)* un Alistāra Kokburna pieeju (lietošanas gadījumu strukturēšanu pēc mērķiem) problēmvides modelēšanas aspektā;

- Demonstrēt lietošanas piemērā, ka izstrādātā pieeja sasniedz promocijas darba mērķi.

Pētījuma objekts ir reālās pasaules parādību un darbam tajā paredzētā lietojuma atbilstības modelēšanas formalizācija.

Pētījuma priekšmets ir lietošanas gadījumus izmantojošas problēmvides modelēšanas pieejas, kuras tiek lietotas programmatūras modelēšanai objektorientētā programmatūras izstrādē saskaņā ar MDA.

Darba mērķa sasniegšanai veiktie pētījumi balstās uz vārdu analīzi, grafu teoriju, topoloģiskās funkcionēšanas modelēšanas un kategoriju loģikas iespējām. Minētās metodes tiek izmantotas šādi:

1) Vārdu analīze tiek izmantota sistēmas neformālā apraksta analīzei;
2) Grafu teorijas likumi tiek izmantoti transformācijās starp grafiem, kuri apraksta problēmu no dažādiem viedokļiem, kā arī grafu precizēšanā;
3) Topoloģiskā funkcionēšanas modelēšana tiek izmantota problēmvides funkcionēšanas un statisko attiecību analīzei;
4) Kategoriju loģika tiek izmantota sarežģītu attiecību specifikācijā.

Darba rezultātā **jaunieguvums** ir *eksistējošai problēmvidei* ***nepretrunīgas*** *funkcionēšanas un struktūras definēšana lietojuma specifikācijā.* Tas ir realizēts, lietojot formālas matemātiskas konstrukcijas un likumus, kurus atbalsta topoloģiskā funkcionēšanas modelēšana un universālā kategoriju (bultu diagrammu) loģika. Šie formālie pamati veicina mazāk intuitīvu eksistējošās problēmvides modeļu konstruēšanu, kā ari padara atbilstības informācijas, kura ir aprakstīta lietošanas gadījumos, pārvaldību formālāku. Izstrādātās pieejas, kura ir dēvēta par *topoloģisko funkcionēšanas modelēšanu modeļu vadāmai arhitektūrai (TFMfMDA),* īpašības ir salīdzinātas ar plaši izmantoto modelēšanas pieeju īpašībām. Pie tam šis pētījums ietver formalizācijas iespēju, ko sniedz *Topoloģiskās Funkcionēšanas Modelēšana (TFM),* salīdzinājumu ar iespējām, ko sniedz Sovas konceptuālie grafi un universālā kategoriju loģika. Šis salīdzinājums paskaidro TFMfMDA formālo pamatu stiprās un vājās puses.

Darba praktiskais lietojums. *Izstrādātā modelēšanas pieeja TFMfMDA* ļauj lietot formālās pieejas problēmvides analīzei, kā ari ļauj pārliecināties, ka plānotā programmatūra atbilst reālajā pasaulē eksistējošai sistēmai. TFMfMDA ir ieteikta lietot daudzfunkcionālo sistēmu modelēšanai projektos, kuros tiek izmantotas lietošanas gadījumu vadītas *(use case driven)* pieejas. TFMfMDA lietojums ir detalizēti izskaidrots un demonstrēts realizētajā lietošanas piemērā (bibliotēkas funkcionēšanas sistēma). Promocijas darbā ir izstrādāts pieejā izmantoto jēdzienu un to attiecību metamodelis un UML profils, kas saskaņo TFMfMDA ar MDA idejām, kā arī vienkāršo TFMfMDA saskaņošanu ar citām MDA modelēšanas valodām.

Darba gaitā tika veikti **pētījumi** un **izstrādātas metodes:**

- Izstrādāta un lietošanas piemērā pārbaudīta topoloģiskā funkcionēšanas modelēšana modeļu vadāmai arhitektūrai (TFMfMDA), kura ir paredzēta atbilstības definēšanai starp lietojuma modeli un problēmvides modeli; izstrādāts pieejā izmantojamo koncepciju metamodelis, kurš balstīts uz metaobjektu vides (Meta Object Facility, MOF) konstrukcijām, un UML profils;
- Novērtēta TFMfMDA pieeja salīdzinājumā ar UP, B.O.O.M. un Alistāra Kokburna pieejas īpašībām problēmvides modelēšanas aspektā; salīdzināts lietošanas gadījumu identificēšanas mehānisms šajās pieejās; piemērā lietojot TFMfMDA,

iegūtais lietošanas gadījumu modelis, ir novērtēts, izmantojot metrikās balstīto apskates metodi.

- Izstrādāta problēmvides modeļa konstruēšanas metode, kura balstīta uz TFM;
- Izpētītas MDA galvenās koncepcijas, labumi un kritika, UML formalizācijas vēsture un pašreizējais stāvoklis, problēmvides modelēšana, izmantojot lietošanas gadījumu *(use case)* īpatnības un atbilstošas tā saucamās lietošanas gadījumu vadītās pieejas *(use case driven approaches)*, tādas kā vienots process *(UP)* ar UML, biznesa objektorientēta modelēšana *(B.O.O.M.)* ar UML un pieeja lietošanas gadījumu strukturēšanai pēc mērķiem, kuru piedāvāja Alistārs Kokburns;
- Izpētīts iespējamais formalizācijas līdzeklis problēmvides modelēšanai: novērtētas un ilustrētas ar piemēriem Sovas konceptuālo grafu, topoloģiskās funkcionēšanas modelēšanas un universālās kategoriju teorijas formalizācijas iespējas;

Par darba **galvenajiem rezultātiem** tika ziņots (vai ir pieņemts publicēšanai) šādās starptautiskajās zinātniskajās konferencēs:

1. *Asniņa E. Formalization Aspects of Problem Domain Modeling within Model Driven Architecture// Databases and Information Systems. Seventh International Baltic Conferenee on Databases and Information Systems. Communications, Materials of Doctoral Consortium, July 3-6, 2006, Vilnius, Lithuania. - Vilnius: Technika, 2006 (ISBN 9955-28-013-1). -93-104 p.*
2. *Asniņa E., Osis J, The Computation Independent Viewpoint: a Formal Method of Topological Functioning Model Constructing// Seientific Proceedings of Riga Technical University, Series — Computer Science (5), Applied Computer Systems. - Rīga: RTU, 2006. - pieņemts publicēšanai.*
3. *Asniņa E. The Formal Approach to Problem Domain Modelling Within Model Driven Architecture// Proceedings of the 9' International Conference "Information Systems Implementation and Modelling" (ISIM'06), April 25-26, 2006, Prerov, Czech Republic. - Ostrava: Jan Stefan MARQ., 2006. - 97-104 p.*
4. *Alksnis G., Asniņa E., Osis J., Silins J. Formalization of Software Development: Problems and Solutions// Seientific Proceedings of Riga Technical University, Series — Computer Science (5), Applied Computer Systems, Volume 22. - Riga; RTU, 2005. - 204 -216 p.*
5. *Asniņa E. Formatisation Aspects of Problem Domain Analysis// Proceedings of the 8th International Conferenee "Information Systems Implementalion and Modelling " (ISIM '05). April 19-21, Hradec nad Moravici, Czech Republic- Ostrava: Jan Stefan MARQ., 2005.- 195-202 p.*
6. *Asniņa E. Topological Modeling and Arrow Diagram Logic Formalism Application for Software Development// Scientific Papers of University of Latvia, Volume 673, Databases and Information Systems, Doetoral Consortium, Sixth International Baltic Conference BalticDB&IS 2004, Riga, Latvia, June 6-9, 2004. -Riga: Latvijas Universitāte, 2004.- 220 - 231 p.*
7. *Asniņa E. Formai Integration Perspective in the Software Development// Seientific Proceedings of Riga Technical University, Series — Computer Science (5), Applied Computer Systems, Volume 17, - Riga: RTU, 2003. -145 - 154 p.*
8. *Asniņa E., Osis J. Formalization Problems and Perspectives of the Software Development// Scientific Proceedings of Riga Technical University, Computer Science, Applied Computer Systems, 3rd thematic issue.- Riga: RTU, 2002. - 145-156 p.*

Promocijas darba **struktūra** - darbs sastāv no ievada, četrām nodaļām, secinājumiem, pieciem pielikumiem un izmantotās literatūras saraksta. Darba pamatteksts izklāstīts 162 lappusēs un paskaidrots ar 49 attēliem un 31 tabulu. Literatūras sarakstā iekļauti 104 informācijas avotu nosaukumi.

Ievadā ir pamatota veikto pētījumu aktualitāte, formulēts galvenais pētījumu mērķis un uzdevumi, kā arī sniegts īss pētījuma pamatvirzienu raksturojums.

Pirmajā nodaļā ir aprakstītas problēmas un realizētie risinājumi šodienas problēmvides modelēšanā.

Otrajā nodaļā ir aprakstīta formālo modelēšanas metožu daba. Novērtētas Sovas konceptuālo grafu, topoloģiskās funkcionēšanas modelēšanas un universālās kategoriju loģikas formalizācijas iespējas pētījuma mērķa sasniegšanai.

Trešajā nodaļā ir aprakstīta izstrādātā pieeja, kura formalizē problēmvides modelēšanu - topoloģiskā funkcionēšanas modelēšana modeļu vadāmai arhitektūrai TFMfMDA, kura ietver topoloģiskā funkcionēšanas modeļa konstruēšanas metodi, sistēmas funkcionālo prasību attēlošanu problēmvides modelī, lietošanas gadījumu definēšanas metodi un koncepciju definēšanu no problēmvides modeļa. Pie tam šajā nodaļā ir aprakstīts TFMfMDA metamodelis un deklarēts UML profils.

Ceturtajā nodaļā ir aprakstīts TFMfMDA pieejas lietojums, tās īpašības salīdzinātas ar UP, B.O.O.M. un Alistāra Kokburna pieeju īpašībām. Turklāt rezultātā iegūtais lietošanas gadījumu modelis ir apskatīts, lietojot metrikās bāzēto metodi.

Secinājumos ir dots paveiktā darba kopsavilkums un formulēti tālāko pētījumu virzieni.

1.pielikumā ir piedāvāts darbā izmantoto saīsinājumu un to atšifrējumu saraksts.

2.pielikumā ir bibliotēkas funkcionēšanas neformāls apraksts, kas izmantots ceturtajā nodaļā izskatītā lietošanas piemēra analīzē.

3.pielikumā ir aprakstītas ceturtajā nodaļā izmantotās sistēmas funkcionālās prasības bibliotēkas lietojumprogrammai.

4. pielikumā ir detalizēti aprakstīta noslēgšanas operācija bibliotēkas topoloģiskā funckionēšanas modeļa atdalīšanai.

5. pielikuma ir aprakstīta ceturtajā nodaļā iegūto lietošanas gadījumu specifikācija teksta formā.

DARBA SATURA IZKLĀSTS

Ievadā ir pamatota veikto pētījumu aktualitāte, formulēts galvenais pētījumu mērķis un sniegts īss pētījumu pamatvirzienu raksturojums, kā arī ieskats promocijas darba nodaļās.

Pirmajā nodaļā (Mūsdienu risinājumi objektorientētā programmatūras izstrādē) ir aplūkotas problēmvides modelēšanas problēmas un to mūsdienu risinājumi objektorientētajā programmatūras izstrādē.

Pirmajā apakšnodaļā ir aprakstītas problēmas, kuras ir saistītas ar semantikas specifikāciju. Vizuālajam modelim, kurš ir aprakstīts kādā modelēšanas valodā, jāattēlo problēmvide skaidri un atbilstoši (nepretrunīgi). Taču formālo pamatu trūkums modelēšanas valodās un problēmvides daudzveidība izraisa šādas problēmas [33]:

- Modelēšanas valodu sintaktiskā pārslodze;
- Semantikas apraksts nevis specifikācijā, bet realizācijas konstrukcijās; arī aspektu sajaukums specifikācijā;
- Specificēšanas loģikas neformāla vai pusformāla daba.

Otrajā apakšnodaļā ir aprakstīti modeļu vadāmas arhitektūras MDA pamatprincipi, pamatmodelis, lietošana un kritika. Kopš 2001. gada eksistējošās modeļu vadāmās arhitektūras galvenais mērķis ir atdalīt aspektus specifikācijas un palielināt analīzes un projektēšanas lomu programmatūras izstrādē.

MDA paredz trīs specificēšanas veidus un attiecīgus modeļus [60]:

- No skaitļošanas neatkarīgs modelis *(Computation Independent Model, CIM),* kurš apraksta sistēmas prasības un veidu, kā sistēma mijiedarbosies ar apkārtējo vidi. Sistēmas struktūra un lietojumprogrammas realizācija ir nedeterminēta;

- Platformneatkarīgs modelis *(Platform Independent Model, PIM),* kurš fokusējas uz sistēmas darbu. Šis modelis paslēpj platformai, uz kuras lietojums strādās, nepieciešamās detaļas.
- Platformai specifisks modelis *(Platform Specific Model, PSM),* kurš parāda visas detaļas, kas nepieciešamas konkrētai lietojuma platformai.

MDA koncepcija atbalsta abstrahēšanu un detalizēšanu modeļos.

Kodola standarti, kurus atbalsta MDA, ir vienotā modelēšanas valoda *(Unified Modeling Language, UML),* metaobjektu vide *(Meta Object Facility, MOF)* un vispārējais noliktavu metamodelis *(Common Warehouse Metamodel, CWM).*

2005.gadā formulētais MDA pamatmodelis [65] apraksta MDA lietotās koncepcijas un to attiecības, kā arī noteic citu (ne tikai UML) modelēšanas valodu definēšanu un izmantošanu saskaņā ar modeļu vadāmās arhitektūras principiem.

MDA paredz transformācijas starp modeļiem, kuri apraksta sistēmu no dažādiem viedokļiem, bet pagaidām deklarē tikai četrus iespējamus transformācijas tipus [61]:

- No PIM uz PIM - modeļa precizēšana bez platformspecifiskās informācijas pievienošanas;
- No PIM uz PSM - modeļa precizēšana ar platformspecifiskās informācijas pievienošanu;
- No PSM uz PSM - modeļa precizēšana ar paplašinošo informāciju par komponenšu realizāciju un izvietošanu;
- No PSM uz PIM - modeļa abstrahēšana no platformspecifiskās informācijas; ideālā gadījumā tam jāsakrīt ar ieejas modeli attiecīgajā transformācijā no PIM uz PSM.

Pie tam, katrai transformācijai ir jābūt pierakstītai *transformācijas ierakstā (record of transformation).* MDA definē sešas transformācijas pieejas: marķēšanu, metamodeļa transformāciju, modeļa transformāciju, šablona lietošanu, modeļu saplūšanu, kā arī papildus informācijas pievienošanu [60].

Neskatoties uz MDA ieguvumiem, daži zinātnieki un praktiķi atzīmē iespējamos šķēršļus tās lietošanā [58], [88], [100]:

- MDA nevar būt par universālu līdzekļi, kamēr tā nav pietiekoši labi izpētīta un pārbaudīta industriālā mērogā;
- Rīku izstrādātāju inertums un nevēlēšanās veikt izmaiņas bez skaidri redzamiem ieguvumiem var kavēt MDA principu automatizāciju;
- Visu transformāciju tipu starp modeļiem realizācija, piemēram, pilnā koda ģenerēšana no modeļiem var kļūt par murgu programmētājiem;
- MDA idejas var ieviest vēl neizpētītas izmaiņas ražošanas procesa organizēšanā.

Taču neskatoties uz visu augšminēto, joprojām vairākums modelēšanas pieeju izstrādātāju, tai skaitā MDA izstrādātāji, nepievērš uzmanību tam, ka nav paredzēta formāla trasējamība starp no skaitļošanas neatkarīgiem un platformneatkarīgiem modeļiem. Tas nozīmē, ka atbilstības noteikšanas problēma lietojuma problēmvidei joprojām atrodas ārpus vairākuma MDA izstrādātāju redzesloka [15], [73], [79], [98], Tādēļ šis pētījums ir solis MDA brieduma virzienā.

Trešajā apakšnodaļā ir aprakstīta UML valodas formālisma attīstība. UML valodas 1.1. versija tika atzīta par standarta valodu 1997.gadā. 2005. gadā OMG pieņēma par spēkā esošu jau UML 2.0. versiju. Šajā apakšnodaļā ir aplūkots formālisms 1.4. un 2.0. versijās.

UML valodas modelis ir aprakstīts precīzās konstrukcijās, kaut gan to nedrīkst uzskatīt par formālu valodu dabiskās valodas izmantošanas dēļ. Abās versijās UML valodas konstrukcijas ir aprakstītas metamodelī un attēlotas trīs aspektos [66]:

- Abstraktā sintakse apraksta UML valodas metaklases UML klašu diagrammu veidā, kā arī nosaka izteiksmju veidošanas likumus un dod īsus metaklašu un metaatribūtu aprakstus dabiskajā valodā;
- Izteiksmju veidošanas likumi ir aprakstīti dabiskajā valodā un formālajā objektu ierobežojumu valodā *(Object Constraint Language, OCL).* Šie likumi specificē metamodelī definēto atribūtu un asociāciju ierobežojumus;
- Semantika jeb konstrukciju jēga ir definēta dabiskajā valodā.

Galvenās izmaiņas formālajā pamatā UML valodas 2.0. versijā ir pamata konstrukciju precizēšana un atdalīšana. Pie tam šajā versijā vienas no diagrammām, proti, aktivitāšu diagrammas, pamatā, ir ieviests Petrī tīklu formālisms. Kaut gan valodas izstrādātāji uzsver, ka formālismu var ieviest tikai gadījumā, kad tas dod acīmredzamu labumu. [68].

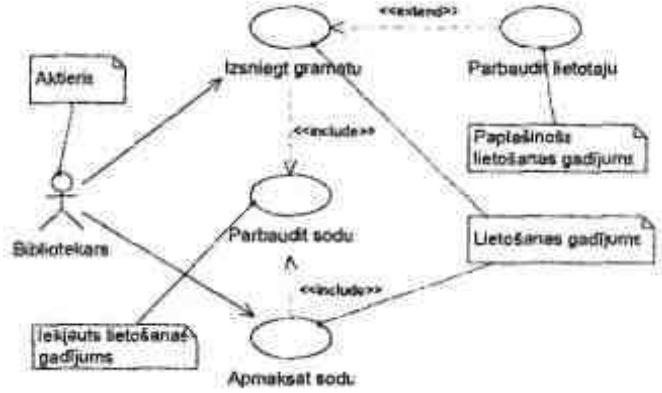
UML valodas relatīvi vājās vietas ir liels notācijas izmērs, neskaidra saskaņa starp diagrammām, konstrukciju dažādu interpretāciju iespēja, savas UML valodas apakškopas definēšanas iespēja [1], 12], [3], [46], [68], [70].

Ceturtajā apakšnodaļā ir aplūkota problēmvides modelēšana, izmantojot lietošanas gadījumu vadītas pieejas.

Sistēmu modelēšanā ir jāizdala divi svarīgi aspekti: analīze, kura definē *ko* sistēma darīs ar vidi, lai izpildītu klienta prasības, un projektēšana, kura definē *kā* sistēma tiks konstruēta. Objektorientētajā programmatūras izstrādē robeža starp analīzi un projektēšanu ir diezgan izplūdusi [79]. Pie tam objektorientētā sistēmas analīze paredz, ka programmatūras izstrādātājam vajadzētu analizēt klienta esošo situāciju pēc iespējas precīzāk un vākt plānotās sistēmas prasības, jo objektorientētajā programmatūras izstrādē nav izdalīta atsevišķa objektorientētā prasību vākšana. Bieži vien prasību analīzi kļūdaini uzskata par klienta *vēlmju* noteikšanu. īstenībā prasību analīzei jādefinē klienta *vajadzības* un vide, kurā plānotā programmatūra strādās [15], [86], [98].

Objektorientētajā sistēmas analīzē prasību definēšanai plaši lieto tā saucamos *lietošanas gadījumus (use cases),* kurus ieviesa Ivars Džekobsons 1987.gadā. Galvenais lietošanas gadījumu ieviešanas mērķis bija uzlabot trasējamību starp tekstuāli attēlotām sistēmas prasībām un analīzes modeļiem. Tādēļ I. Džekobsons u.c. izdalīja trīs lietojumorientētas prasību modeļa daļas, kur var izmantot lietošanas gadījumus [62]:

1. Lietošanas gadījumu modelis, kurš definē, kas eksistē ārpus sistēmas un kam jābūt izpildāmam sistēmā;
2. Saskarnes apraksts,
3. Problēmvides objektu modelis.



1. att. Lietošanas gadījumu modelis

Lietošanas gadījumus var aprakstīt dažādos detalizācijas līmeņos [87], Lietošanas gadījumi ir „...ar uzvedību saistīto transakciju īpašā secība, kura tiek izpildīta aktieru un

sistēmas dialogā... Šī transakcija sastāv no dažādām aktivitātēm, kuras jāizpilda... Visu lietošanas gadījumu aprakstu kopa specificē pilnu sistēmas funkcionalitāti..." [62]. Lietošanas gadījumi var aprakstīt kopējo funkcionalitāti (iekļauti *(included)* lietošanas gadījumi) un kādam nosacījumam pakļauto funkcionalitāti (paplašinoši *(extending)* lietošanas gadījumi). Lietošanas gadījumu un aktieru notācija ir ilustrēta 1. attēlā.

Lietošanas gadījumiem ir šādas priekšrocības [47], [87]:

- Tie ir orientēti uz lietotājiem;
- Labs līdzeklis saziņai starp izstrādātājiem un pasūtītājiem (lietotājiem);
- Sarežģītu un lielu sistēmu funkcionalitāti var sadalīt galvenajās funkcijās;
- Tie izmanto zema līmeņa scenārijus;
- Tos var izmantot kā labu bāzi funkcionālo prasību validācijai un augsta līmeņa modeļu verificēšanai; lietošanas gadījumi ir trasējami;
- Tos var izmantot kā pārvaldības vienumu projekta izstrādes procesu plānošanā.

Lietošanas gadījumiem ir šādi ierobežojumi [28], [47]:

- Informācijas tveršana - tos var aprakstīt dažādos detalizācijas līmeņos, kā arī tie ļauj iekļaut informāciju par lietotāja saskarni;
- Domāšanas ierobežojums - to koncepcijas vienkāršība mudina izstrādātājus, prasības vācot, palikt tikai pie lietošanas gadījumu saraksta definēšanas, izslēdz citas prasību vākšanas pieejas un rūpīgu problēmvides analīzi;
- Pilnīguma pārbaude - lietošanas gadījumi ir notācija nevis pieeja, tādēļ, salīdzinājumā ar citām prasību vākšanas metodēm, to lietošanā nav sistēmas; tie nedod atbildes uz šādiem jautājumiem: 1) Vai visi lietošanas gadījumi ir identificēti? 2) Kādi ir konflikti starp lietošanas gadījumiem? 3) Vai ir palikušas nedefinētas sistēmas prasības? 4) Kā izmaiņas prasībās var ietekmēt lietošanas gadījumus?

Lai atrisinātu minētās problēmas, pieejas, kuras izmanto lietošanas gadījumus, joprojām tiek precizētas. Piemēram, interesanta ir to lietošana vienotā procesa UP *(Unified Process)* prasību darbplūsmā [7]. Tā kā lietošanas gadījumi ir paredzēti pamatprasību glabāšanai, UP izmanto tos kā bāzi dažādu sistēmas skatu konstruēšanai. Kritiskās vietas šajā pieejā ir:

- UP un, tādējādi, tajā izmantojamos lietošanas gadījumus, pārvalda atkarībā no prasībām. UP nenoliedz arī kāda biznesa modeļa izmantošanu, taču nedefinē, kā formāli pāriet no biznesa modeļa uz lietojumprogrammas modeli. Biznesa modelis šajā gadījumā ir tikai palīglīdzeklis lietošanas gadījumu un aktieru identificēšanai;
- Lietošanas gadījumi un aktieri tiek definēti, izmantojot pagaidu ideju par sistēmas robežām;
- Prasību un lietošanas gadījumu trasējamība tiek definēta patvaļīgi, izmantojot tikai izstrādātāja intuīciju un pieredzi šajā jomā;
- Iekļaušanas un paplašināšanas lietošanas gadījumu identificēšana ir monotons darbs, kurš prasa funkcionēšanas rūpīgu pētīšanu visos lietošanas gadījumu aprakstos. Kaut gan šādas attiecības UP pieejā tiek uzskatītas par papildus un tās drīkst neizmantot.

Objektorientētu biznesa modelēšanu ar lietošanas gadījumiem (B.O.O.M.) izstrādāja Hovards Podesva [84]. Šī pieeja ir paredzēta biznesa analītiķu darba rezultātu saskaņošanai ar objektorientēto izstrādi. Šī pieeja izmanto *biznesa lietošanas gadījumus (business use cases)* problēmvides specificēšanai un *sistēmas lietošanas gadījumus (system use cases)* sistēmas prasību specificēšanai. Apkopojot visas B.O.O.M. aktivitātes iniciēšanas fāzē, tika atklātas šādas vājās vietas:

- Šī pieeja izslēdz citas prasību vākšanas metodes;

- Biznesa līmeņa lietošanas gadījumu definēšana ir orientēta uz projekta mērķiem nevis uz problēmvidi, kaut gan tiek pārbaudīta šo lietošanas gadījumu atbilstība esošajiem biznesa procesiem;
- Lietošanas gadījumu pakotņu *(package)* veidošana saskaņā ar loģiskajām attiecībām starp sistēmas lietošanas gadījumiem ir intuitīva;
- Ja sistēmas lietošanas gadījumu pakotnes tiek veidotas atdalīti no biznesa lietošanas gadījumiem, tad izmaiņas biznesa procesos ir grūti attēlot visos ar tām saistīto sistēmas lietošanas gadījumu aprakstos.

Alistāra Kokburna pieejā, kura tiek dēvēta par lietošanas gadījumu strukturēšanu pēc mērķiem, ir izdalītas četras lietošanas gadījumu dimensijas: mērķis (stāsti, prasības), saturs (pretrunīgs, saskanīgā proza, formālais saturs), skaits (viens, daudzi) un struktūra (nestrukturētā, pusformālā, formālā struktūra) [42], Šajā pieejā lietošanas gadījumi tiek definēti pēc mērķiem, pie kam tiek izvietoti vai nu sistēmas, vai stratēģiskajā līmenī. Kokburns piedāvāja izvietot lietošanas gadījumus hierarhiski, kur visaugstākajā līmenī ir stratēģiskie lietošanas gadījumi un kopsavilkuma mērķi, kuri tiek sadalīti stratēģiskā līmeņa lietotāju mērķos un sistēmas līmeņa kopsavilkumu mērķos, kas savukārt norāda uz sistēmas līmeņa lietotāju mērķiem un tālāk uz sistēmas līmeņa apakšfunkcijām.

Šīs pieejas priekšrocības ir šādas: a) iespēja piesaistīt mērķiem nefunkcionālās prasības, b) pārvaldīt projektus pēc mērķiem, c) agrāk atpazīt vājas prasības, tām nesasniedzot mērķi, d) šī pieeja atvieglo darbu ar prasību specifikācijām.

Tomēr dažas problēmas pastāv arī šajā pieejā, piemēram, līmeņu, mērķu un datu variāciju sajaukšanas iespēja.

Piektajā apakšnodaļā ir apkopotas problēmvides modelēšanas problēmas [28], [70], [73], [79] un izklāstīta pētījuma motivācijas būtība:

- MDA piedāvā aprakstīt sistēmas prasības, sistēmu un vidi, kurā sistēma strādās, no skaitļošanas neatkarīgajā modelī (CIM). Pēc MDA idejas, šim modelim ir jābūt par ieeju transformācijā no CIM uz PIM. Taču CIM attēlošana platformneatkarīgajā modelī tiek noteikta patvaļīgi.
- Viens no plaši lietotiem paņēmieniem problēmvides un prasību specificēšanai - lietošanas gadījumi apraksta problēmvidi kā „melno kasti", pie tam problēmvides modelēšanai ir diezgan zema prioritāte. Tādēļ lietojumprogrammas struktūra un funkcionēšana tiek pamatota uz *intuitīvu* problēmvides saprašanu.
- Viss minētais var kļūt par iemeslu sliktai sistēmas kvalitātei un neatbilstībai *vajadzībām,* jo lietošanas gadījumi (to pašreizējā stāvoklī) ir fokusēti uz šauru apgabalu, kur *lietojums* tieši mijiedarbojas ar reālo pasauli. Tādējādi analītiķa uzmanība ir pievērsta tikai notikumiem *lietojuma* robežās un apkārtējas vides īpašības netiek adekvāti novērtētas. Tieši tādēļ pirmajai aktivitātei ir jābūt *problēmvides* analīzei un saprašanai.

Tādēļ lietošanas gadījumi jāizmanto kā metodes daļa, kuras pirmā aktivitāte ir labi definēta *problēmvides modeļa* veidošana.

Sestajā apakšnodaļā ir dots īss pirmās nodaļas kopsavilkums un secinājumi.

Otrajā nodaļā (Formālas metodes problēmvides modelēšanai) ir apskatītas formālās problēmvides modelēšanas pieejas un izpētītas to formalizācijas spējas promocijas darba mērķa sasniegšanai.

Brans Seliks (Bran Selic) ir uzskaitījis piecas galvenās efektīva modeļa īpašības [88]: abstrahēšanas spēja, saprotamība, precizitāte, prognozēšanas spēja un nelielas izmantošanas izmaksas.

Šajā nodaļā ir apskatīta Topoloģiskā Funkcionēšanas Modelēšana (TFM), Sovas konceptuālie grafi un universālā kategoriju loģika. Visas šīs pieejas ļauj konstruēt minētajām īpašībām atbilstošu modeli. Tika nolemts neaplūkot Petrī tīklus, jo tie nav pietiekoši saprotami, attēlojot struktūru , pie tam šis pētījums nav saistīts ar simulācijas jautājumiem.

Pirmajā apakšnodaļā ir dots īss ieskats formālās loģikas izmantošanā zināšanu attēlošanai, jo jautājums par semantikas formālo aprakstu joprojām pastāv, it īpaši saistībā ar vizuālām shēmām, pie tam ir jāatšķir simbolu un grafu loģika [25].

Tātad, ja eksistē kāda diagramma *D,* tad tai ir kāda jēga *M(D).* Šī jēga jāapraksta precīzās izteiksmēs. Šīs precīzās izteiksmes formē precīzu specifikāciju S_D ar precīzu semantiku $M(S_D)$. Tātad $M(S_D)$ ir formālā intuitīvas jēgas *M(D)* abstrakcija, un S_D var uzskatīt par diagrammā *D* paslēptas (iekšējas) loģikas specifikāciju. Tad tas nozīmē, ka, ja kāds sāk domāt par formālu jēgas *M* attēlošanu, viņš domā par tādām šaurām vidēm kā kopu teorija, predikātu rēķini, tipu teorija utt. Tas nozīmē, ka jebkuru formālo jēgu var specificēt jebkurā no tām (formālajā valodā) [25].

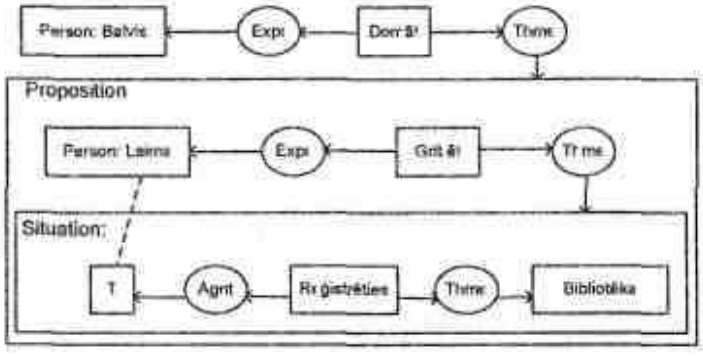
Patlaban pirmās kārtas un augstākās kārtas predikātu rēķinus [101] aktīvi izmanto datorzinātnē. Piemēram, pirmās kārtas rēķini kopā ar kopu teoriju veido formālās specificēšanas valodas Z pamatu [18], [50], [56]. Taču šādām valodām trūkst attēlošanas spēju. Izmantojot vienīgi matemātiskas konstrukcijas, pastāv problēma, ka lielas specifikācijas ātri kļūst nepārvaldāmas un nelasāmas.

Semantiskie tīkli un kadri lika rādīti kā šīs problēmas risinājums [89]. Šiem attēlošanas veidiem ir savas priekšrocības ua trūkumi. Galvenās priekšrocības ir tieša zināšanu par problēmvidi atspoguļošana modelī, secināšanas pēc noklusēšanas atbalsts (t.i. secināšana, izmantojot „is-a" un „a-kind-of" saites), produktivitāte un procedurālu zināšanu atbalsts. Galvenie trūkumi ir semantikas attēlošanas nepietiekamība (piemēram, saitei „is-a" var būt dažādas interpretācijas) un ierobežojumi semantiskās izteiksmēs [21], [55], [85], [89].

Grafu teoriju plaši izmanto gan matemātikā, gan datorzinātnē. Galvenais pētījumu priekšmets ir grafu īpašības. To galvenokārt lieto tīklu analīzē, kā arī topoloģijas problēmu pētīšanā [104].

Otrajā apakšnodaļā ir aplūkoti Sovas konceptuālie grafi (Sovas CGs), kurus Džons Sova (John Sowa) atvasināja no Pirsa (Pierce) eksistenciālajiem grafiem 1984. gadā. Pašlaik notiek šo grafu standartizācija zināšanu attēlošanai [93], [94], [95], [96],

Konceptuālie grafi ir loģikas vizualizēšana, lietojot abstrakto sintaksi. Konceptuālais grafs grafiskā formā ir ilustrēts 2. attēlā. Tas apraksta teikumu „Balvis domā, ka Laima grib reģistrēties bibliotēkā".



2. att. Sovas konceptuālais grafs teikumam „Balvis domā, ka Laima grib reģistrēties bibliotēkā"

Šā grafa simbolu attēlošanas veids ir šāds:

[Person: Balvis]←(Expr)←[Domāt]→(Thme)-

[Proposition: [Person: Laima *x]←(Expr)← [Gribēt]→(Thme)-
[Situation: [?x]←(Agnt)←[Reģistrēties]→(Thme)→[Bibliotēka]]].

Darbā ir aprakstīta Sovas konceptuālo grafu abstraktā sintakse, kuras galvenie elementi ir šādi:

- Koncepcijām, kuras ir attēlotas taisnstūrī, ir tips un referents. Referents ļauj definēt atsevišķu entītijas eksemplāru;
- Konceptuālām attiecībām, kuras ir attēlotas ovālos, ir tips un valence;
- Konteksts ir koncepcija ar ielikto *(neted)* konceptuālo grafu, kurš apraksta tās referentu; piemēram, 2.attēlā ir parādīti divi konteksti - *Proposition* un *Situation.*

Koncepcijas un konceptuālās attiecības ir hierarhiski organizētas starp divām primitīvām iezīmēm *Entity* (universāls tips) un *Absurdity* (aplamības tips).

Konceptuālie grafi atbalsta kanona formēšanas likumus: specializēšanas, vispārināšanas un ekvivalences likumus, kā arī secināšanas likumus.

Sovas konceptuālo grafu galvenā priekšrocība ir iespēja izmantot tos kā starpslāni starp cilvēku valodām un formālajām specifikācijām. Tie ir piemēroti zināšanu glabāšanai, spriešanai un skaitļošanai gan konceptuālo grafu, gan predikātu rēķinu formā.

Trešajā apakšnodaļā ir apskatīta Topoloģiskā Funkcionēšanas Modelēšana (TFM), kuru izstrādāja Jānis Osis Rīgas Tehniskajā universitātē 1969. gadā [72], [81]. Jānis Osis, Zigurds Markovičs, Jānis Grundspeņķis u.c studēja šīs metodes lietošanu medicīniskos uzdevumos, zināšanu ieguvē, iegultās sistēmās, modeļu vadāmā arhitektūrā un citus jautājumus [5], [6], [8], [9], [10], [11], [12], [13], [14], [24], [36], [37], [38], [39], [40], [51], [52], [53], [54], [59], [73], [74], [75], [77], [78], [79], [80], [82], [83].

Šai pieejai ir stingra matemātiska bāze. Galvenā TFM ideja ir lielas vai sarežģītas sistēmas funkcionēšanas attēlošana topoloģiskās telpas *(X, Θ)* formā, kur *X* ir galīgā sistēmas funkcionālo īpašību kopa, bet 6>ir topoloģija, kura ir attēlota orientēta grafa formā [81].

Abstrakts sistēmas topoloģisks funkcionēšanas modelis var būt attēlots kā orientēts grafs *G(X, U),* kur *X* ir galīgā noslēgtā elementu kopa ar starp tiem noteikto topoloģiju *Θ*, un *U* ir loku kopa, kura ilustrē šo topoloģiju. Topoloģija kopā *X* ir kopas X atvērto kopu A sistēma *Θ*. Sistēmai *Θ*jāapmierina divas Kolmogorova aksiomas (1) un (2).

$$X \in \Theta; \quad \emptyset \in \Theta \tag{1}$$

$$\forall \eta \left(\bigcup_{\eta} A_{\eta} \in \Theta \right); \forall \varphi \bigcup_{\varphi=1}^{\kappa} A_{\varphi} \in \Theta \tag{2}$$

Galvenais nosacījums topoloģiskā modeļa konstruēšanai ir pilns, verbāls, nozīmīgs, neformāls sistēmas funkcionēšanas apraksts. Lai no tā konstruētu topoloģisko funkcionēšanas modeli :

- Jānosaka sistēmas aprakstā funkcionālās īpašības, kuras veido sistēmu Z kā izteiksmē (3). Šīm īpašībām jābūt svarīgām sistēmas funkcionēšanai.

$$Z = N \cup M \tag{3}$$

Kur, *Z*- visu definētu funkcionālu īpašību kopa,

N - sistēmas iekšējo funkcionālo īpašību kopa,

M - ārējo sistēmu funkcionālo īpašību un sistēmas īpašību, kuri ietekmē ārējas sistēmas, kopa.

- Jānosaka topoloģija *Θ,* kura norāda *cēloņu un seku attiecības* starp sistēmas fiziskajām vai bioloģiskajām īpašībām grafa vai matrices formā. Ir pieņemts, ka cēloņu un seku attiecība eksistē starp divām īpašībām, ja vienas īpašības parādīšana

izraisa otras īpašības parādīšanu bez kādas trešās īpašības piedalīšanās. Abstrakta topoloģiskā funkcionēšanas modeļa atbilstība problēmai tiek sasniegta ar būtiskas pētāmās sistēmas jēgas piešķiršanu šim abstraktajam matemātiskajam objektam.

- Topoloģiskais funkcionēšanas modelis tiek atdalīts no topoloģiskās telpas, izmantojot noslēgšanas operāciju, kura ir parādīta izteiksmē (4).

$$X = [N] = \bigcup_{\eta=1}^{n} X_\eta \qquad (4)$$

Kur, X_η - kopas N piederības punkts,

$[N]$ - kopas N noslēgšana,

$[\eta]$ - visu kopas N piederības punktu skaits (kopas N jauda).

Topoloģiskais funkcionēšanas modelis atbalsta topoloģiskās īpašības (saistība, noslēgšana, apkārtne, nepārtrauktā attēlošana) un funkcionālās īpašības (cēloņu un seku attiecības, ciklu struktūra, ieejas un izejas) [79]. Šajā apakšnodaļā ir aprakstīti arī apgalvojumi, kuri formalizē šīs īpašības, un no tiem izrietošie secinājumi. Bez tam šajā apakšnodaļā ir apskatīts topoloģiskā funkcionēšanas modeļa piemērs.

Ceturtajā apakšnodaļā ir aprakstītas universālās kategoriju loģikas īpašības. To pamatā ir kategoriju teorijas un bultu domāšanas principi [6], [16], [22], [43]. Šis apraksts pamatojas galvenokārt uz Zinovija Diskina un Borisa Kadiša darbiem šajā jomā [25], [29], [30], [31], [32], [33].

Kategoriju loģikā semantikas specifikācija ir attēlota grafa veidā - tā saucamajā *skicē*. Galvenā ideja ir tāda, ka jebkuru problēmvidi var aprakstīt kā objektu (virsotņu) un to morfismu (bultu) kolekciju kompozīciju. Iekšējā objektu struktūra un attiecības ar citiem objektiem ir attēlotas vienīgi ar bultu diagrammu palīdzību.

Universālā kategoriju loģika ir elastīga - atkarībā no konteksta objekti un bultas var būt attiecīgi:

- kopas un funkcijas (datu modelēšanā),
- objektu klases un asociācijas (objektorientētajā analīzē un projektēšanā),
- datu tipi un procedūras (funkcionālajā programmēšanā),
- teorēmas un pierādījumi (loģikā/loģiskajā programmēšanā),
- saskarnes un procesi (procesu modelēšanā),
- stāvokļi un pārejas (transakciju modelēšanā),
- specifikācijas un to attēlojumi (metamodelēšanā).

Pēc definīcijas kategoriju loģika ir grafu loģika. Citiem vārdiem sakot, predikāta arguments sastāv no virsotņu vietu turētājiem un bultu vietu turētājiem, kuri ir organizēti orientētā grafa formā. Daži no promocijas darbā minētajiem predikātiem darbam ar kopu un funkciju definēšanas sistēmām ir parādīti 1. tabulā [25]. Darbā aprakstītie predikāti ir paskaidroti ar piemēriem.

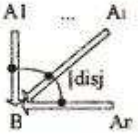
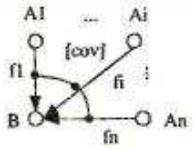
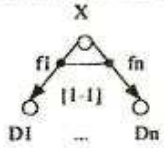
Galvenā doma ir tāda, ka jebkura diagrammā aprakstītā īpašība, kurai ir jēga, var būt piesaistīta kādai iepriekš definētai virsotņu un bultu konfigurācijai - bultu predikātam. Bultu predikāts ir pāris (<predikāta marķieris>, < ar nosaukumiem iezīmētā predikāta struktūra >).

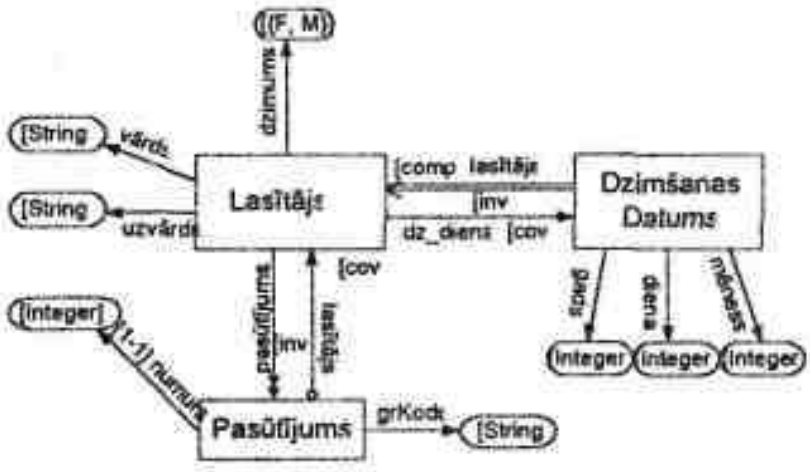
Bultu predikātu kompozīcija veido skici, kuras piemērs ir ilustrēts 3. attēlā. Tā attēlo trīs objektu klases *Lasītājs*, *Dzimšanas datums* un *Pasūtījums* ar atribūtiem bultu veidā un atribūtu tipiem *[String]*, *[Integer]* un *[{fF.M}]*.

Universālā kategoriju loģika atbalsta modularizācijas pamatprincipus, kā arī skiču abstrahēšanu un detalizēšanu, lietojot funktoru naturālo transformāciju.

1. tabula

Bultu diagrammu predikāti kopu un funkciju sistēmai

Predikāta nosaukums	Vizuālais attēlojums	Apzīmējuma semantika
Kopas iekļaušana – avota kopa ir mērķa kopas apakškopa un attēlošana *f* ir to iekļaušana.		$A \subset B$ un $f(a) = a$ visiem $a \in A$
Atdalīšana – par mērķa kopas elementu var būt *tikai vienas apakškopas* elements.		$\bigcup_{i=1}^{n} Ai \subset B$ un $\bigcap_{i=1}^{n} Ai = \varnothing$
Pārklāšana – katrs elements mērķa kopā ir *vismaz vienas* attēlošanas f_i vērtība.		$(\forall b \in B)(\exists i \le n)b \in f_i(A_i)$
Funkciju saimes atdalīšana – kopas X elements ir viennozīmīgi definēts ar D_i kopu kortežu struktūrām.		jebkuram x, $x' \in X$, $x \neq x'$ paredz, ka $f_i(x) \neq f_i(x')$ kādam i. Taču ir jāatzīmē, ka šajā gadījumā funkciju kortežs $f = \langle f_1 \ldots f_n \rangle$ apgabalu Dekarta reizinājumā $f = (f_1 \ldots f_n): X \to D_1 \times \ldots \times D_n$, $fx = (f_1 x, \ldots, f_n x)$ ir iekļaujošs (viens pret vienu) tādā veidā, ka elementi no X var būt apskatīti kā unikālie nosaukumi kortežiem no noteiktas $D_1 \times \ldots \times D_n$ apakškopas, t.i. f attēls..



3. att. Skices piemērs

Piektajā apakšnodaļā ir izpētītas iepriekšējajās apakšnodaļās apskatīto pieeju formalizācijas spējas.

Topoloģiskajai funkcionēšanas modelēšanai un universālajai bultu loģikai ir daudz kopīgā:

- Abām pieejām ir uz grafiem bāzēta struktūra, kur virsotnes var apzīmēt jebkurus objektus atkarībā no konteksta;
- Abas pieejas apmierina saistības nosacījumu;
- Abas pieejas atbalsta detalizēšanu un abstrahēšanu saskaņā ar nepārtrauktās attēlošanas likumiem;

- Pateicoties nepārtrauktajai attēlošanai, datu trūkumu var aizpildīt, attēlojot zināmās tāda paša tipa sistēmas modeli pētāmās sistēmas modelī;
- Skices un topoloģiskā funkcionēšanas modēja atbilstība problēmvides semantikai tiek sasniegta, piešķirot atbilstošo jēgu definētajām matemātiskajām konstrukcijām;
- Abas metodes ļauj pētīt viena tipa sistēmu atšķirības un līdzības.

Bet ir arī nopietnas atšķirības:

- Problēmvides īpašību modelēšanai kā minimālo loģisko vienumu TFM izmanto virsotnes un lokus, universālā kategoriju teorija, savukārt, izmanto bultu diagrammas;
- Universālajā kategoriju loģikā nav mehānisma sistēmas modēja atdalīšanai no problēmvides un apakšsistēmu noteikšanas mehānisma; bet tie eksistē un ir matemātiski pamatoti topoloģiskajā funkcionēšanas modelēšanā;
- Universālajā kategoriju loģikā funkcionalitāte tiek attēlota, izmantojot sarežģītas, grūti saprotamas n-bultu (n>l) grafa struktūras.

Galvenais secinājums no šī salīdzinājuma ir tas, ka analīzes pašā sākumā piemērotāka ir TFM lietošana, jo tā ne tikai piedāvā formālās apakšsistēmas un sistēmas definīcijas, bet arī ļauj matemātiski precīzi, saprotami, bez liekas sarežģītības attēlot un pētīt problēmvidi.

Sovas konceptuālie grafi ir salīdzināti ar TFM:

- Atšķirībā no TFM Sovas konceptuālie grafi pilnībā neatbalsta grafu loģiku, šīs grafiem līdzīgās konstrukcijas ir drīzāk predikātu loģikas grafiskais atspoguļojums;
- Konceptuālie grafi neatbalsta nepārtrauktās attēlošanas likumus. Tie realizē tā saucamā ,.ieliktā grafa" koncepciju. Ielikts grafs ir atkarīgs no konteksta un var būt patiess vai aplams;
- Konceptuālajiem grafiem ir lielāks notāciju izmērs, līdz ar to zināšanas attēlojošās shēmas ir lielāka izmēra;
- Konceptuālie grafi ne tikai attēlo attiecības un koncepcijas, bet arī to, cik šis apgalvojums ir uzticības cienīgs;
- Ciklu struktūra tiek attēlota nedabiskā veidā atšķirībā no attēlošanas topoloģiskajā funkcionēšanas modelī, jo, lai parādītu sistēmas atgriešanos jau apmeklētā stāvoklī, konceptuālajos grafos ir jāattēlo jauns grafs;
- Loģiskās attiecības tiek attēlotas ieliktos kontekstos, kas palielina specifikāciju un apgrūtina tās saprašanu.

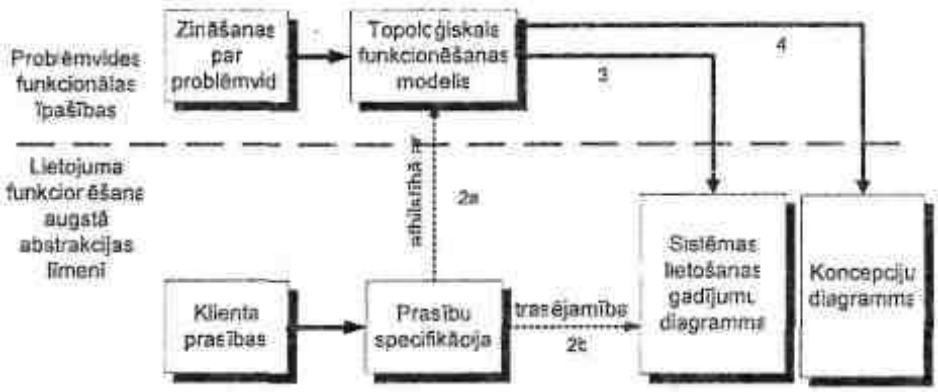
Galvenais secinājums ir, ka Sovas konceptuālie grafi un universālā kategoriju loģika promocijas darba mērķa sasniegšanai piedāvā šajā posmā nevajadzīgu detalizāciju un sarežģītību, bet nepiedāvā visai svarīgu koncepciju ,.sistēma" un ,.apakšsistēma" formālo matemātisko definīciju.

Sestajā apakšnodaļā ir dots īss otrās nodaļas rezultātu kopsavilkums un TFM izstrādes pamatojums modeļu vadāmai arhitektūrai *(TFM forMDA*, vai *TFMfMDA).*

Trešajā nodala (Risinot grūtības problēmvides modelēšanā) ir piedāvāta izstrādātā pieeja - TFMfMDA un tās lietošanas piemērs.

Galvenā izstrādātās pieejas ideja ir šāda (4. attēls). Savāktās zināšanas par problēmvidi var aprakstīt topoloģiskajā funkcionēšanas modelī, lietojot topoloģiskā funkcionēšanas modeļa konstruēšanas metodi (bulta i). Tad klienta funkcionālās prasības var saskaņot ar šo modeli, t.i. var attēlot topoloģiskajā modelī. Rezultātā funkcionālās prasības ir precizētas un validētas atbilstībā pret pastāvošo problēmvides funkcionalitāti. Modeli var arī papildināt ar jaunu prasībās definētu funkcionalitāti (bulta 2a). Pēc tam, lietojot uz mērķiem bāzēto metodi atbilstoši sistēmas prasībām, modelī var definēt sistēmas lietošanas gadījumus (bulta 3), bet lietojot transformāciju - koncepciju diagrammu (bulta 4). Turklāt, funkcionālās prasības būs

trasējamas no lietošanas gadījumiem caur formālo bāzi - topoloģiskā modeļa funkcionālajām īpašībām (bulta 2b) [11].



4. att. TFM vieta problēmvides modelēšanā

Pirmajā apakšnodaļā ir aprakstīta topoloģiskā funkcionēšanas modeļa konstruēšanas metode (4. att. bulta 1). Tās etapi ir ilustrēti 5. attēlā.

5. att. Formālā metode topoloģiskā funkcionēšanas modeļa konstruēšanai

Metodes etapos ir definētas šāda darbības [10], [11], [14]:

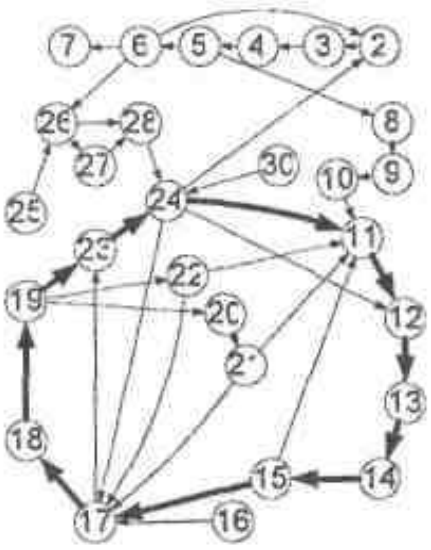
- *Fizisko vai biznesa funkcionālo raksturojumu definēšanā:*
  - o Objektu un to īpašību definēšana no problēmvides apraksta. Definēšana notiek, lietojot lietvārdu analīzi - izdalot jēgpilnus lietvārdus un tiešus papildinātājus, apstrādājot sinonīmus un homonīmus;
  - o Ārējo sistēmu (t.i. objektu, kuri nepakļaujas sistēmas likumiem) un daļēji atkarīgo sistēmu (t.i. objektu, kuri daļēji pakļaujas sistēmas likumiem, piemēram, sistēmas darbinieku lomu) izdalīšana starp definētajiem objektiem;
  - o Funkcionālo īpašību definēšana, lietojot darbības vārdu analīzi problēmvides aprakstā - izdalot jēgpilnas aktivitātes. Katra funkcionālā īpašība apraksta objekta darbību, darbības rezultātu un objektu, kurš saņem darbības rezultātu vai tiek izmantots šajā darbībā (piemēram, loma, laika periods, katalogi utt.), kā arī priekšnosacījumu jeb atomāro biznesa likumu un atbildīgo par izpildi entītiju. Katrs priekšnosacījums un atomārs biznesa likums vai nu jādefinē kā funkcionāla īpašība, vai nu jāpiesaista kādai iepriekš definētai funkcionālai

īpašībai. Promocijas darbā ir definēti divi pieraksta veidi (šeit tie ir doti angļu un latviešu valodā):

- Detalizēts pieraksts
  <action>-ing the < result> [to, into, in, by, of, from] a(n) < object>
  <rezultāts> <darbība>-šana [uz, no, pēc] <objekts>
- Abstrakts pieraksts
  <action>-ing a(n) <object>
  <objekts> <darbība>-šana

- *Topoloģijas Θ ieviešana* ir cēloņu un seku attiecību definēšana starp funkcionālām īpašībām. Cēloņu un seku attiecības tiek attēlotas kā orientētā grafa loki, kuru orientācija ir no cēloņa virsotnes uz efekta virsotni. Cēloņu un seku attiecību galvenās īpašības ir šādas: a) cēlonis hronoloģiski notiek pirms sekām; b) cēlonis var būt pietiekams vai nepieciešams (vai pilnīgs jeb daļējs), pie kam topoloģiskā funkcionēšanas modeļa konstruēšanā ir pieņemts, ka visi cēloņi ir nepieciešami, jo sistēmas funkcionēšanas riski analīzes laikā var būt nezināmi; c) cēlonis ne tikai notiek pirms sekām un aiz tā vienmēr notiek sekas, tas ir seku ģenerēšanas nosacījums un ģenerētājs; d) cēloņu un seku attiecības ir universālas - tās pastāv jebkurā problēmvidē, pat ja tas nav redzams cilvēkam. Cēloņi un sekas var veidot ķēdi, kurā visas attiecības ir svarīgas. Promocijas darbā ir aprakstīti daži padomi cēloņu un seku attiecību noteikšanai.
- *Topoloģiskā funkcionēšanas modeļa atdalīšanas* etapā darbības ir tādas pašas kā TFM pieejā - sistēmas iekšējo funkcionālo īpašību kopas noslēgšanas operācijas lietošana [81].

Lietojuma piemērā konstruētais bibliotēkas topoloģiskais funkcionēšanas modelis ir ilustrēts 6. attēlā un attiecīgas funkcionālās īpašības ir definētas 2. tabulā. Galvenais funkcionēšanas cikls **„11-12-13-14-15-17-18-19-23-24-11"** ir attēlots ar treknām bultām.



6. att. Bibliotēkas topoloģiskais funkcionēšanas modelis

Otrajā apakšnodaļā ir aprakstīti funkcionālo prasību attēlojumi topoloģiskajā funkcionēšanas modelī TFMfMDA pieejā (4. att. bulta 2a). Funkcionālās *īpašības* apraksta funkcionalitāti, kura *pastāv problēmvidē*. Savukārt, funkcionālās *prasības* apraksta funkcionalitāti, kurai *jāpastāv lietojumā.* Tādēļ šo kopu attēlojumi ir iespējami.

2. tabula

Lietojuma piemēram definētās funkcionālās īpašības un priekšnosacījumi

Nr.	Funkcionālā īpašība	Priekšnosacījums	Atbildīgais	Ārējais
2	Vārda, uzvārda, vecuma meklēšana pēc personas		Reģistrētājs	nav
3	Lasītāja kartes veidošana		Reģistrētājs	nav
4	Lasītāja kartes aizpildīšana		Reģistrētājs	nav
5	Koda piesaistīšana lasītāja kartei		Reģistrētājs	nav
6	Bibliotēkas biļetes aizpildīšana		Reģistrētājs	nav
7	Bibliotēkas biļetes izsniegšana		Reģistrētājs	nav
8	Kataloga pārmeklēšana		Lasītājs	nav
9	Pieprasījuma formas aizpildīšana		Lasītājs	nav
10	Pieprasījuma formas iesniegšana bibliotekāram		Lasītājs	nav
11	Pieprasījuma formas pieņemšana no lasītāja		Bibliotekārs	nav
12	Pieejamības pārbaudīšana kopijai		Bibliotekārs	nav
13	Grāmatu skaita aprēķināšana lasītājam	[ja kopija ir pieejama]	Bibliotekārs	nav
14	Grāmatu limita pārbaude lasītājam		Bibliotekārs	nav
15	Kopijas izsniegšana lasītājam	[ja grāmatu skaits ir mazāks nekā grāmatu limits]	Bibliotekārs	nav
16	Kopijas atdošana		Lasītājs	nav
17	Kopijas stāvokļa pārbaudīšana		Bibliotekārs	nav
18	Soda aprēķināšana lasītājam		Bibliotekārs	nav
19	Soda piespriešana lasītājam		Bibliotekārs	nav
20	Restaurācijas formas aizpildīšana kopijai	[ja kopijai ir nepieciešama restaurācija]	Bibliotekārs	nav
21	Kopijas sūtīšana restauratoram		Bibliotekārs	nav
22	Kopijas aizvākšana	[ja kopiju nevar restaurēt]	Bibliotekārs	nav
23	Kopijas pārbaudīšana	[ja kopija ir labā stāvoklī]	Bibliotekārs	nav
24	Pieejamības nodrošināšana kopijai		Bibliotekārs	nav
25	Kopijas saņemšana		Reģistrētājs	nav
26	ieraksta pārbaudīšana katalogā		Reģistrētājs	nav
27	Ieraksta pievienošana katalogam	[ja tāda ieraksta nav]	Reģistrētājs	nav
28	Koda piešķiršana kopijai		Reģistrētājs	nav
30	Grāmatas saņemšana atpakaļ no restauratora		Restaurators	nav

TFMfMDA pieejā ir definētas piecas attiecības un attiecīgie bultu predikāti (3. tabulā) [11], [10]. Šīs prasību un īpašību attiecības ir šādas:

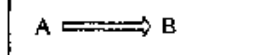
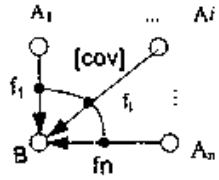
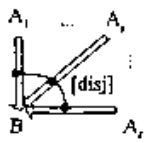
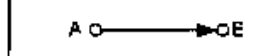
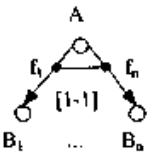
- *Viens uz. vienu.* Viena funkcionāla prasība piinīgi specificē vienu funkcionālu īpašību;
- *Daudzi uz vienu.* Daudzas funkcionālas prasības specificē vienu funkcionālu īpašību, tādēļ ka: 1) tās ir pārklājošās prasības, vai 2) tā ir abstraktāka funkcionālā īpašība;
- *Viens uz daudziem.* Viena funkcionāla prasība specificē daudzas funkcionālas īpašības, tādēļ ka: 1) funkcionālā prasība aptver vairākas prasības un var tikt sadalīta, vai 2) funkcionālās īpašības ir detalizētākas nekā prasība;
- *Viens uz nulli.* Viena funkcionāla prasība specificē kādu jaunu vai nedefinētu funkcionalitāti. Šajā gadījumā ir nepieciešams izpētīt iespējamās izmaiņas problēmvides funkcionalitātē.
- *Nulle uz vienu.* Prasību specifikācija nesatur prasību, kura specificētu doto funkcionālo īpašību. Tas nozīmē, ka šī (problēmvides) funkcionalitāte netiks realizēta lietojumā.

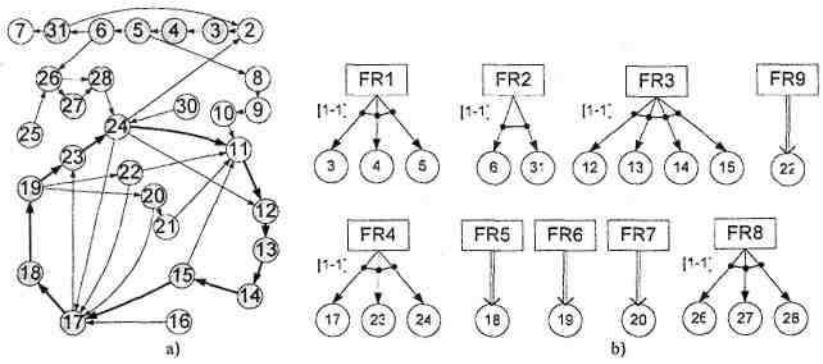
Papildus jāpiemin, ka attēlojums var būt pierakstīts arī matricas veidā, bet šādā gadījumā attēlošanas īpatnības netiks specificētas.

Lietojuma piemērā atbilstība starp topoloģisko funkcionēšanas modeli un funkcionālām prasībām (4. tabulā) ir definēta tā, kā ir ilustrēts 7(b). attēlā. Iegūtais topoloģiskais funkcionēšanas modelis ir parādīts 7(a). attēlā.

3. tabula

Bultu diagrammu predikāti

Attēlošana	Bultu diagrammu predikāti	Apraksts
Viena uz vienu		*Iekļaušanas predikāts* tiek izmantots, ja funkcionāla prasība *A* pilnīgi specificē to, kam jābūt implementētam atbilstībā ar funkcionālu īpašību *B*
Daudzas uz vienu		*Pārklāšanas predikāts* tiek izmantots, ja funkcionālas prasības $A_1, A_2, \ldots, A_n$ pārklāj tās funkcionalitātes specifikāciju, kurai jābūt realizētai atbilstībā ar funkcionālu īpašību *B*
		*Atdalīšanas (komponenšu) predikāts* tiek izmantots, ja funkcionālas prasības $A_1, A_2, \ldots, A_n$ kopā pilnīgi specificē funkcionālu īpašību *B* un nepārklājas
Viena uz daudzām		*Projekcija* tiek izmantota, ja kāda funkcionālas prasības *A* daļa nepilnīgi specificē kādu funkcionālu īpašību B_j
		*Funkciju saimes atdalīšanas predikāts* tiek izmantots, ja viena funkcionāla prasība *A* pilnīgi specificē funkcionālas īpašības $B_1,\ldots,B_n$



7. att. Galīgais topoloģiskais funkcionēšanas modelis (a) un funkcionālo prasību attēlojumi (b)

Trešajā apakšnodaļā ir aprakstīta pāreja no sākotnējā no skaitļošanas neatkarīgā modeļa uz izejas modeli (4. att. bulta 3). Sākotnējais C1M ir topoloģiskais funkcionēšanas modelis. Izejas modelis ir lietošanas gadījumu modelis. Citiem vārdiem sakot, TFMfMDA pieejā ir izstrādāta lietošanas gadījumu modeļa iegūšanas metode.

4. tabula

Funkcionālās prasības lietojuma piemēram

Iezīme	Funkcionālā prasība
FR1	Sistēmai jāizpilda jauna lasītāja reģistrācija, kura iekļauj lasītāja kartes
FR2	Sistēmai jāizpilda bibliotēkas biļetes drukāšana uz speciālas veidlapas.
FR3	Sistēmai jāizpilda grāmatas kopijas izsniegšana lasītājam.
FR4	Sistēmai jāizpilda grāmatas kopijas pieņemšana, iekļaujot grāmatas stāvokļa
FR5	Sistēmai jāaprēķina naudassods.
FR6	Sistēmai jāpiespriež naudassods lasītājam.
FR7	Sistēmai jāpieraksta grāmatas nosūtīšana uz restaurāciju.
FR8	Sistēmai jāizpilda ieraksta pievienošana kopiju katalogam.
FR9	Sistēmai jāizpilda kopijas aizvākšana no bibliotēkas.

Šī metode paredz šādus soļus f 10], [11]:

- *Biznesa sistēmas lietotāju un to mērķu identificēšana.* Par biznesa sistēmas lietotājiem var būt aktieri un darbinieki:
    - Biznesa sistēmas aktieri ir *ārējas* entītijas, kuras tieši mijiedarbojas ar biznesa sistēmu un izvirza *biznesa mērķus.* Topoloģiskajā funkcionēšanas modelī tie tiek attēloti kā ārējo sistēmu funkcionēšana vai kā pētāmās sistēmas funkcionālas īpašības, kuras mijiedarbojas ar ārējām sistēmām. Pēdējā gadījumā ir nepieciešama šo sistēmu identificēšana. Aktieri var būt ārējas kompānijas, klienti u.c.
    - Biznesa sistēmas darbinieki ir sistēmas *iekšējas* entītijas, kuras tieši mijiedarbojas ar biznesa sistēmu un izvirza *sistēmas mērķus* (vai realizē biznesa mērķus). Darbinieki var būt cilvēki, lomas u.c.
    - Biznesa sistēmas lietotāju tiešu mērķu identificēšana ir saistīta ar attiecīgas funkcionālo īpašību kopas identificēšanu. Šī kopa ir nepieciešama mērķa sasniegšanai. Katram mērķim šajā kopā var identificēt ieejas funkcionālo īpašību (ieejas transakciju), izejas funkcionālo īpašību (izejas transakciju) un funkcionālo īpašību ķēdi starp tām.

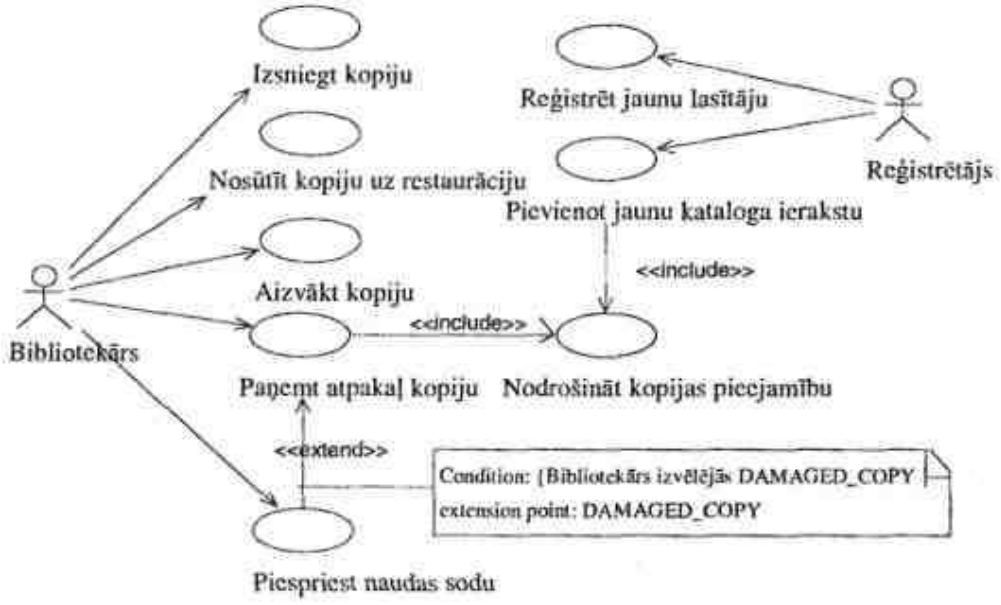
  Gan aktieri, gan darbinieki var būt lietojumprogrammas lietotāji. Sistēmas (lietojumprogrammas) mērķu identificēšana palīdz papildus prasību validēšanai, tas ir, aizmirstu prasību atklāšanai. Šajā metodē mērķis ir lietošanas gadījumu identificēšanas līdzeklis. Tas tika izvēlēts tāpēc, ka mērķi var sasniegt, izpildot kādu procesu, kurš var ilgt vairākas dienas un nedēļas. Laika princips šajā gadījumā ir bezjēdzīgs.
- ***Sistēmas lietošanas gadījumu identificēšana un precizēšana***. Funkcionālās īpašības, kuras ir specificētas ar funkcionālajām prasībām un tiek izpildītas biznesa mērķa sasniegšanai, apraksta *sistēmas lietošanas gadījumu.* Biznesa sistēmas lietotājs, kurš izvirzīja šo mērķi, ir aktieris, kurš piedalās dialogā ar šo lietošanas gadījumu. Šis princips ļauj formāli identificēt lietošanas gadījumu modeli topoloģiskajā funkcionēšanas modelī. Bet no šā principa izriet arī papildus iespējas sistēmas lietošanas gadījumu precizēšanai:
    - *Iekļaušanas lietošanas gadījums* ir kāda kopēja darbību secība, kuru izmanto vairāki lietošanas gadījumi [87]. Topoloģiskajā funkcionēšanas modelī tas ir funkcionālo īpašību, kuras ir definētas mērķa sasniegšanai un specificētas ar funkcionālām prasībām, kopu šķērsojums. Bet šī kopējā funkcionalitāte

jāizpēta, jo var pastāvēt situācija, kad kopēja funkcionāla īpašība atrodas lietošanas gadījuma alternatīvā darbību plūsmā. Tas nozīmē, ka šī īpašība atrodas topoloģiskā modeļa zarā jeb apakšciklā;

- *Paplašināšanas lietošanas gadījums* [87] var būt topoloģiskā funkcionēšanas modeļa zars jeb apakšcikls. Funkcionāla īpašība, no kuras sākas zars, ir paplašinošs punkts *(extending point)* lietošanas gadījumu modelī.
- Funkcionalitāti, kuru apraksta identificētie lietošanas gadījumi, var attēlot UML aktivitāšu diagrammās, attiecīgas funkcionālas īpašības transformējot diagrammas aktivitātēs un cēloņu un seku attiecības - kontrolplūsmās.

- *Lietošanas gadījumu prioritāšu noteikšana* var būt a) saistīta ar klienta prasībām, b) definēta, lietojot kādas prasību atribūtu sistēmas, piemēram, MoSCoW jeb GRASP [7]. Izstrādātajā pieejā lietošanas gadījumu realizēšanas prioritātes tiek definētas šādi (atbilstoši Rational Unified Process):
  - *Kritiska prioritāte* (jārealizē, citādi sistēma būs nepieņemama), ja lietošanas gadījums realizē galvenā cikla funkcionālu īpašību;
  - *Svarīga prioritāte* (var ievērojami ietekmēt sistēmas lietošanu), ja lietošanas gadījums implementēs funkcionālu īpašību, kura ir kādas galvenā cikla īpašības cēlonis vai sekas;
  - *Derīga prioritāte* (nav ievērojamas ietekmes uz sistēmas funkcionēšanu), ja lietošanas gadījums neimplementēs nevienu galvenā cikla īpašību vai īpašību, kura ietekmē vai to ietekmē kāda galvenā cikla funkcionāla īpašība.

Lietojuma piemērā ir apskatīta lietošanas gadījumu modeļa iegūšana, lietošanas gadījumos aprakstītās funkcionalitātes specificēšana UMI. aktivitāšu diagrammā un funkcionālu prasību trasējamība no lietošanas gadījumiem (4. att. bulta 2b). Rezultātā iegūtais lietošanas gadījumu modelis ir parādīts 8. attēlā.

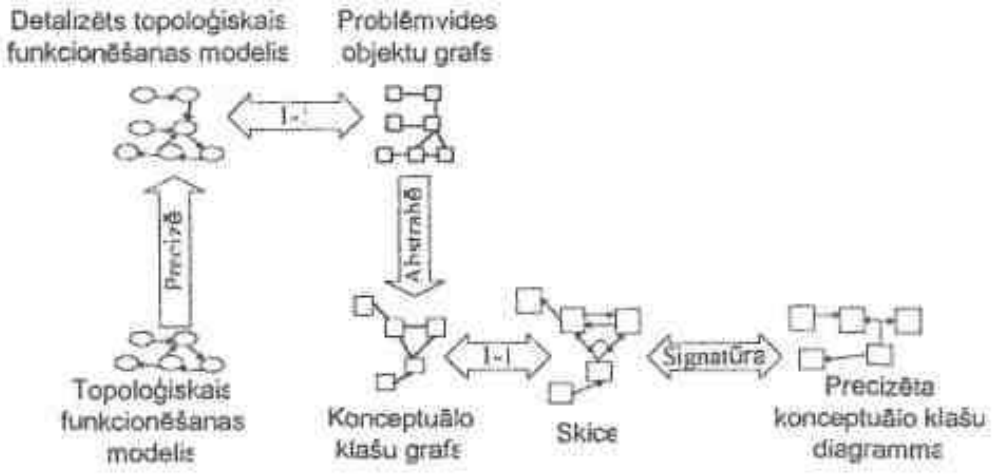


8. att. Lietojuma piemērā iegūtais lietošanas gadījumu modelis

Ceturtajā apakšnodaļā ir apskatīta koncepciju diagrammas un kontroles klašu identificēšana TFMfMDA pieejā (4. att. bulta 4). Koncepcija jeb konceptuāla klase ir ideja, lieta vai objekts [48]. Tā kā pēc funkcionālo prasību attēlošanas topoloģiskais funkcionēšanas

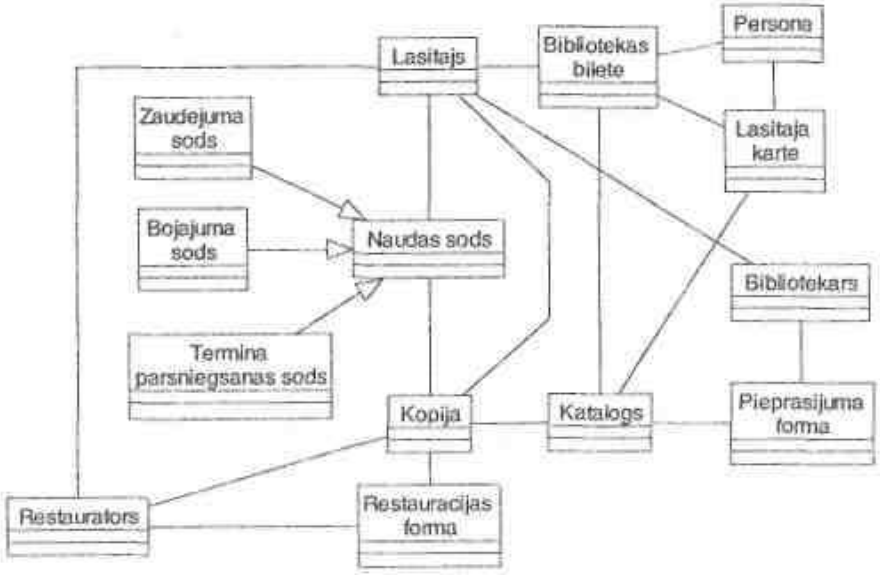
modelis apraksta arī funkcionēšanu, kura jārealizē lietojumprogrammā, tad tam būtu jāsatur visas koncepcijas, kuras vajadzīgas funkcionēšanai.

Konceptuālo klašu diagrammas iegūšana ir ilustrēta 9. attēlā. Lai iegūtu konceptuālo klašu diagrammu, jāprecizē katra topoloģiskā funkcionēšanas modeļa funkcionālā īpašība līdz stāvoklim, kad katrā no specializētajām īpašībām izmantoti tikai viena tipa objekti. Pēc tam precizētais modelis jātransformē problēmvides objektu grafā viens pret vienu un jāsapludina virsotnes ar vienādiem objektu tipiem, saglabājot visas attiecības ar citām grafa virsotnēm. Rezultātā tiks iegūts konceptuālo klašu grafs ar nenoteiktām asociācijām [9], [12], [73]. Lai precizētu šīs attiecības, grafu var pārveidot skicē, precizēt un attēlot jau precizētā konceptuālo klašu diagrammā.



9. att. Konceptuālo klašu diagrammas iegūšana

Papildus šajā apakšnodaļā ir definēti divi secinājumi. Viens no tiem norāda uz iespējamām mantošanas attiecībām starp tipiem. Otrs - uz kopējām operācijām, kuras vēlāk var izdalīt lietošanas gadījumu interfeisos.



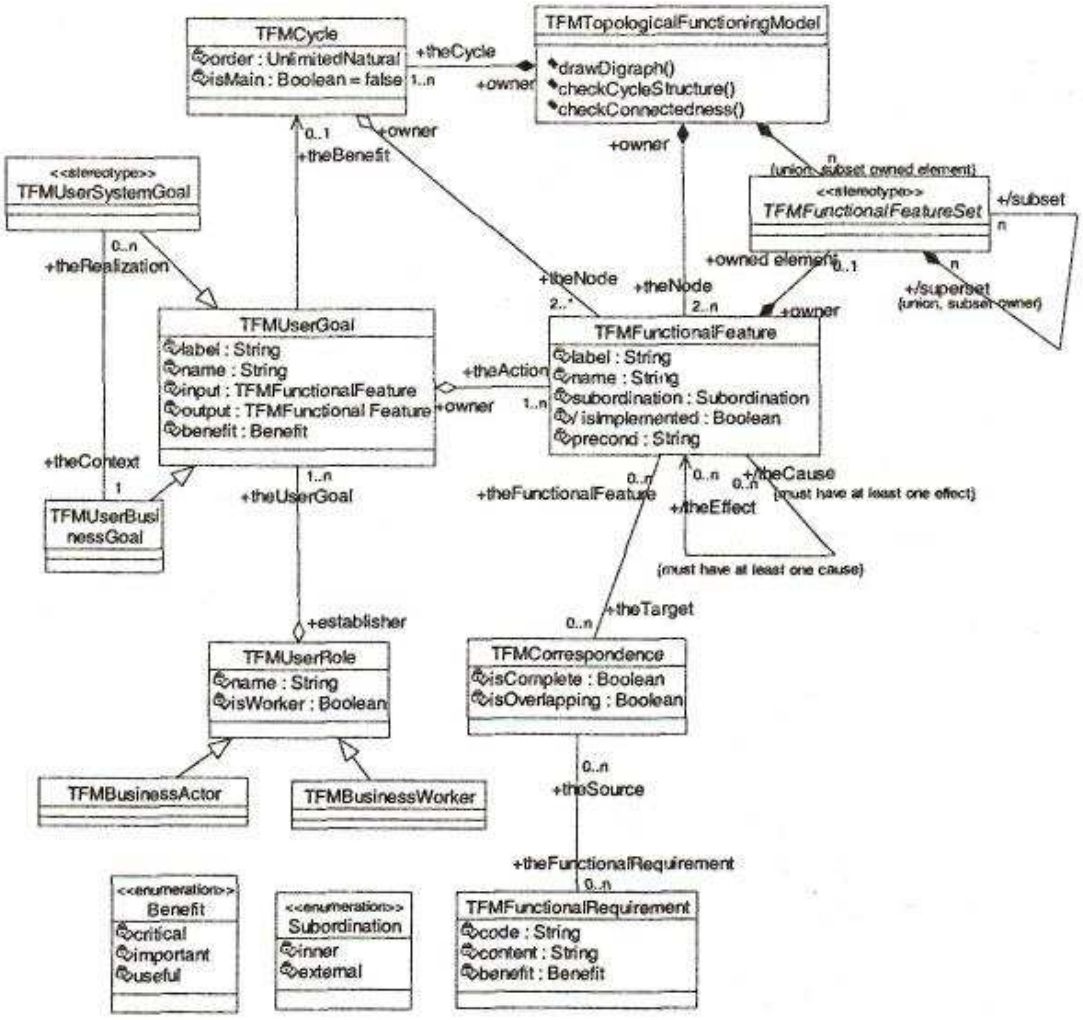
10. att. Konceptuālo klašu diagramma

Bez tam var identificēt kontroles objektus. Kontroles objekti attēlo starpslāni starp lietotāja saskarni un klases objektiem [7], [48], Ja lietotājs iniciē kādas funkcionālās īpašības izpildi, tad šīs īpašības darbību var izdalīt lietošanas gadījuma kontroles objektā.

Lietojuma piemērā ir aplūkota konceptuālo klašu diagrammas iegūšana, transformējot topoloģisko funkcionēšanas modeli, un identificēti lietošanas gadījumu kontroles objekti. Konceptuālo klašu diagramma ir attēlota 10. attēlā.

Piektajā apakšnodaļā ir aprakstīts universālās bultu loģikas lietošanas piemērs shēmu integrācijas uzdevumam [8], [25], [26], [31], [32].

Sestajā apakšnodaļā ir aprakstīta TFMfMDA pieejas atbilstība MDA pamatmodelim. TFMfMDA koncepcijas ir aprakstītas metamodelī, izmantojot MOF elementus, kā arī ir izstrādāts UML profils šai pieejai. TFMfMDA pieejas metamodelis ir parādīts 11. attēlā.



11. att. MOF konstrukcijas definētais TFMfMDA pieejas metamodelis

Metamodelis ir aprakstīts MOF līmenī M2. Tas attēlo topoloģisko funkcionēšanas modeli kā *TFMTopologicalFunctioningModel* tipa eksemplāru, kurš sastāv no vismaz divām *TFMFunctionalFeature* tipa funkcionālām īpašībām. Tās var būt iekļautas funkcionālu īpašību kopās (tips *TFMFunctionalFeatureSet).* Viena funkcionāla īpašība var ietvert tikai vienu kopu, un funkcionāla īpašība var piederēt tikai vienai kopai. Funkcionāla īpašība var būt pakļauta vai nu pētāmai sistēmai, vai nu kādai ārējai sistēmai (tips *Subordination).*

Funkcionālas īpašības var sastādīt dažādas kārtas funkcionēšanas ciklus (tips *TFMCycle).* Funkcionālas īpašības saista cēloņu un seku attiecības - cēloņu īpašībai jābūt vismaz vienam efektam, efekta īpašībai jābūt vismaz vienam cēlonim. Funkcionālas īpašības ir saistītas ar funkcionālām prasībām (tips *TFMFunctionalRequirement),* izmantojot atbilstību (tips *TFMCorrespondence).* Atbilstība var būt pilnīga vai nepilnīga, pārklājoša vai atdalīta. Funkcionālas īpašības var būt asociētas ar vairākiem mērķiem (tips *TFMUserGoal). Mērķus* izvirza tiešie lietotāji (tips *TFMBusinessUser):* biznesa sistēmas *TFMBusinessActor* tipa aktieri un *TFMBusinessWorker* tipa darbinieki . Lietotāja mērķis var būt specializēts kā biznesa mērķis (tips *TFMUserBusinessGoal)* vai sistēmas mērķis (tips *TFMUserSystemGoal).* Lietotāja mērķis (un, tādējādi, funkcionālās prasības) ir asociēts ar funkcionēšanas ciklu, kura kārta norāda prasību un mērķa ieguvuma vērtību (tips *Benefit).*

Septītajā apakšnodaļā ir dots īss trešās noda]as kopsavilkums un secinājumi par izstrādātās TFMfMDA pieejas lietošanu.

Ceturtajā nodaļā (TFMfMDA pieejas **novērtēšana)** ir apskatītas topoloģiskās funkcionēšanas modelēšanas modeļu vadāmai arhitektūrai *(TFMfMDA)* īpašības problēmvides modelēšanas un lietošanas gadījumu formalizācijas aspektā.

Pirmajā apakšnodaļā ir lietota neempīriskā pētīšanas metode - TFMfMDA pieejas, vienota procesa (UP) prasību darbplūsmas, B.O.O.M. iniciēšanas fāzes un A. Kokburna pieejas īpašību apskats. Ir secināts, ka tikai TFMfMDA un B.O.O.M. prasa obligātu esošās problēmvides analīzi, bet tikai TFMfMDA nodrošina formālo bāzi šai analīzei. Kokburna pieejā nav definēta lietojuma un problēmvides sākotnējas statiskās struktūras modelēšana.

Visās pieejās, izņemot promocijas darbā izstrādāto, problēmvides robežas sākotnēji tiek definētas intuitīvi vai - vadoties no eksperta lēmumiem. Izstrādātajā pieejā problēmvides robežas tiek definētas, izmantojot matemātisku mehānismu - noslēgšanas operāciju.

Izstrādātajā pieejā formāli tiek identificēta arī kopēja funkcionēšana, bet pārējās pieejās tas tiek darīts manuāli, apskatot lietošanas gadījumu aprakstus vai, kā A. Kokburna pieejā, definējot kopējas apakšfunkcijas. Vienīgi TFMfMDA nodrošina teksta formā aprakstīto funkcionālo prasību trasējamību no lietošanas gadījumiem caur formālo bāzi -topoloģisko funkcionēšanas modeli.

5. tabula

Lietošanas gadījumu ierobežojumi apskatītajās pieejās

Ierobežojums	**TFMfMDA pieeja**	**UP ar UML**	**B.O.O.M.**	**A. Kokburna pieeja**
Informācijas tveršana				
Informācija par lidotāja saskarni	nav	ir	nav	nav
Dažāds detalizācijas līmenis	nav	ir	nav	ir
Domāšanas ierobežošana				
Prasību vākšana aprobežojas ar lietošanas gadījumu saraksta veidošanu	nav	nav	ir	nav
Izslēdz citas prasību vākšanas metodes	nav	nav	ir	nav
Izslēdz rūpīgu problēmvides analīzi	nav	nav	nav	ir
Pilnīguma pārbaude				
Netiek identificēti visi sistēmas lietošanas gadījumi	nav	ir	ir	ir
Konflikti starp lietošanas gadījumiem	nav	ir	nav	nav
Plaisas, kuras var būt atstātas sistēmas prasībās	nav	ir	nav	ir
Nav skaidrs, kā izmaiņas ietekmē uzvedību, kura ir aprakstīta citos lietošanas gadījumos	nav	ir	ir	ir

5. tabulā ir parādīts, kādi lietošanas gadījumu ierobežojumi šajās pieejas ir atrisināti vai joprojām pastāv.

Otrajā apakšnodaļā ir lietota gan empīriska metode - lietošanas piemēra pētījums, gan neempīriska - metriskā pieeja.

Lietošanas gadījumu skaitu nevar izmantot par novērtēšanas kritēriju [84]. Tādēļ tika izstrādāts piemērs, kurā ir aprakstīta problēmvide ar iespējami daudzveidīgu funkcionalitāti un lielu lomu skaitu. Lietojot TFMfMDA pieeju no problēmvides apraksta tika identificēti 4 aktieri un 18 lietošanas gadījumi (no 82 funkcionālām īpašībām).

Pēc tam, vadoties pēc topoloģiskā funkcionēšanas modeļa funkcionālajām īpašībām, tika izstrādāti lietošanas gadījumu apraksti saskanīgajā prozā. Lietošanas gadījumi tika novērtēti, lietojot metrisko lasīšanas pieeju (Metric Based Reading technique) [17]. Metriskā novērtēšana palīdzēja izdarīt šādus secinājumus:

- TFMfMDA pieeja novērš paslēpto nepilnīgo lietošanas gadījumu veidošanu, jo visi nepilnīgie lietošanas gadījumi ir definēti vai nu kā iekļaušanas, vai nu kā paplašināšanas lietošanas gadījumi atklātā veidā;
- Tas, ka tekstuālais apraksts tiek veidots manuāli, pieļauj defektu ieviešanos lietošanas gadījumu aprakstos;

Metriskā novērtēšana pievēra uzmanību arī tādiem jautājumiem, kuri pētījuma gaitā netika apskatīti. Tās ir problēmas, kas saistītas ar līdzsvaru starp aktiera un sistēmas darbībām un ar lietošanas gadījumu aprakstu saprotamību, ja lietošanas gadījumā ir specificēta daudznosacījumu secība. Bet jāatzīmē, ka šīs problēmas ir visai atkarīgas no problēmvides funkcionēšanas īpatnībām un ir grūti atrast universālo atbildi uz tām.

Trešajā apakšnodaļā ir dots īss ceturtajā nodaļā veikto darbību un rezultātu kopsavilkums.

DARBA GALVENIE REZULTĀTI

Promocijas darba galvenais mērķis bija izpētīt iespēju formalizēt problēmvides modelēšanu un izstrādāt pieeju, kura realizētu šī pētījuma rezultātus saskaņā ar modeļu vadāmās arhitektūras idejām.

Promocijas darba galvenais rezultāts ir *izstrādāta jauna problēmvides modelēšanas pieeja TFMfMDA (topoloģiskā funkcionēšanas modelēšana modeļu vadāmai arhitektūrai),* kas *demonstrēta ar diviem lietojuma piemēriem.*

Promocijas darba *svarīgākie rezultāti* ir:

1. Izstrādāta metodika, kura formāli apraksta un analizē problēmvidi, izejot no tās neformāla apraksta.

   Metodikas pamatā ir formālā loģika - topoloģiskās funkcionēšanas modelēšanas principi. Problēmvides funkcionēšana ir formāli nedalīti saprotami aprakstīta topoloģiskajā funkcionēšanas modelī. Modeļa funkcionālās un topoloģiskās īpašības formāli pamato konstruētā modeļa atbilstību problēmvidei.

2. Izstrādāta metodika, kura palīdz noteikt sistēmas funkcionālo prasību atbilstību problēmvides noteiktajai funkcionalitātei.

   Metodika definē iespējamos sistēmas funkcionālo prasību (teksta formā) attēlojumus topoloģiskajā modelī aprakstītājā funkcionalitātē. Attēlojumu formālais pieraksts balstās universālajā kategoriju loģikā. Sistēmas funkcionālās prasības tiek validētas un precizētas.

3. Izstrādāta metodika lietošanas gadījumu noteikšanai no problēmvides apraksta topoloģiskajā funkcionēšanas modelī.

   Problēmvides funkcionalitāte ir precīzi transformēta lietošanas gadījumu modelī pēc lietotāju mērķiem atbilstoši klienta prasībām un funkcionēšanas ierobežojumiem.

4. Izstrādāta metodika konceptuālo klašu un to attiecību, kā arī kontroles klašu noteikšanai no problēmvides apraksta topoloģiskajā funkcionēšanas modelī.

   Metodikā lieto grafu transformācijas likumus. Sistēmas struktūru pamato sistēmas funkcionēšana nevis otrādi.

5. TFMfMDA ir saskaņota ar MDA idejām.

   Visi pieejā izmantotie jēdzieni ir aprakstīti gan *ar MOF saskaņotā metamodelī,* gan darba gaitā izstrādātā *UML profila* pirmajā versijā.

6. TFMfMDA pieeja tika aprobēta divos dažāda izmēra lietošanas piemēros (bibliotēkas funkcionēšana) un ir iegūts pozitīvs rezultāts - sastādītais lietošanas gadījumu modelis pilnībā atbilst prasībām un pētāmajai problēmvidei.

   TFMfMDA praktiskā lietojuma priekšrocības (sakārtotas svarīguma secībā) :

- Problēmvides apraksts lietošanas gadījumos ir izmainīts no „melnās kastes" uz „balto kasti";
- Rūpīga ciklu struktūru analīze var palīdzēt identificēt visas funkcionālās un kauzālās attiecības sarežģītās biznesa sistēmās;
- Tā ļauj pašā analīzes sākumā noteikt, ko pasūtītājam patiešām vajag. Iespēju automatizēt funkcionalitāti, kura ir nepieciešama pasūtītāja biznesa mērķu sasniegšanai, var noteikt pašā sākumā. Tas nozīmē, ka, redzot sistēmas biznesa loģiku kopumā, analītiķis un pasūtītājs var pieņemt lēmumu par izmaiņu pieejamību tajā pirms izmaiņu realizācijas produktā;
- TFMfMDA palīdz validēt funkcionālo prasību pilnīgumu; atsekot tās lietošanas gadījumos, izmantojot formālo bāzi - topoloģiskā funkcionēšanas modeļa funkcionālās īpašības; kā arī palīdz pārbaudīt, kā funkcionālo prasību izmaiņas ietekmēs esošo problēmvides funkcionalitāti;
- TFMfMDA atrisina dažus lietošanas gadījumu ierobežojumus informācijas tveršanā, domāšanas ierobežojumā un pilnīguma pārbaudē;
- Izstrādātajai pieejai ir vienkārša, saprotama notācija. Tas var stimulēt lietotājus aktīvāk piedalīties problēmu risināšanā.

  TFMfMDA pieejas *praktiskā lietojuma vājā vieta* ir rīku atbalsta trūkums. Bet jāatzīmē, ka metodikas, kuras ir īstenotas TFMfMDA pieejā, var daļēji automatizēt.

TFMfMDA pieeja **tiek rekomendēta** lietošanai sarežģītu biznesa sistēmu izstrādē, kurām piemīt izteikta funkcionalitāte un daudz lietotāju lomu.

Promocijas darba izstrādes rezultātā tika apstiprināta pārliecība, ka problēmvides modelēšanu saskaņā ar MDA idejām var formalizēt bez ievērojamas sarežģīšanas.

Tālāko pētījumu virziens var būt saistīts ar sekojošiem priekšmetiem:

- nefunkcionālo prasību pārvaldības aspekts;
- rīku atbalsta izstrāde TFMfMDA pieejai ar iespēju integrēties ar UML un MDA atbalstošiem izstrādes rīkiem;
- topoloģiskā funkcionēšanas modeļa lietošana modeļu trasējamībai.

IZMANTOTĀ LITERATŪRA

[I] Alhir, S. S. UML Extension Mechanisms//Distributed Computing. - 1998. - December, - 29-32 p. (Internet. -http://home-comcast.net/-salhir/UMLExtensionMechanisms.PDF)

[2] Alhir, S. S. Understanding Use Case Modeling// Methods & Tools: Newsletters. - 2000. - Spring (Volume 8 - number 1). - 10-16 p. (Internet.- http://home.comcast.net/-salhir/UnderstandingUseCaseModeling.PDF. http://www.methodsandtools.com/PDF/DMT0100.pdf) [3] Alhir, S. S. Understanding Structural Modeling// Methods & Tools: Newsletters. - 2001. - Winter (Volume 9 - number 4). - 2-13 p. (Internet. - http://home.comcast.net/~salhir/UnderstandingStructuralModeling.PDF. http://www.methodsandtools.com/PDF/DMT0401.pdf)

(4] Alhir, S. S. Understanding the Model Driven Architecture (MDA)// Methods & Tools: Newsletters. - 2003- - Winter (Volume 11 - number 3). - 17-24 p. (internet. - http://home.comcast.net/~salhir/UnderstandingTheMDA.PDF, http://www.methodsandtools.com/PDF/dmt0303.pdf)

[5] Alksnis G., Asnina E., Osis J., Silins J. Formalization of Software Development: Problems and Solutions// Scientific Proceedings of Riga Technical University, Series — Computer Science (5), Applied Computer Systems, Volume 22. - Riga: RTU, 2005. - 204 - 216 p.

[6] Alksnis G., Osis J. Formalization of Software Engineering by Means of the Theory of Categories// Scientific Proceedings of Riga Technical University, Computer Science, Applied Computer Systems, 3rd thematic issue.- Riga: RTU, 2002. - 157-163 p.

[7] Arlow J., Neustadt I. UML2 and the Unified Process: Practical Object-Oriented Analysis and Design. - Addison-Wesley, Pearson Education, 2005. - 592 p.

[8] Asnina E. Formal Integration Perspective in the Software Development// Scientific Proceedings of Riga Technical University, Series — Computer Science (5), Applied Computer Systems, Volume 17. - Riga: RTU, 2003.-145-154 p.

[9] Asnina E. Formalisation Aspects of Problem Domain Analysis// Proceedings of the 8th International Conference "Information System Implementation and Modelling" (ISIM'05), April 19-21, Hradec nad Moravicf, Czech Republic- Ostrava: Jan Stefan MARQ., 2005.- 195-202 p.

[10] Asnina E. Formalization Aspects of Problem Domain Modeling within Model Driven Architecture// Databases and Information Systems. Seventh International Baltic Conference on Databases and Information Systems. Communications, Materials of Doctoral Consortium, July 3-6, 2006, Vilnius, Lithuania. - Vilnius: Tcchnika, 2006 (ISBN 9955-28-013-1). - 93-104 p

[II] Asnina E. The Formal Approach to Problem Domain Modelling Within Model Driven Architecture// Proceedings of the 9th International Conference "Information Systems Implementation and Modelling" (ISIM'06), April 25-26, 2006, Prerov, Czech Republic, l[st]edn. (ISSN 1613-0073).- Ostrava: Jan Stefan MARQ., 2006.-97-104 p.

[12] Asnina E. Topological Modeling and Arrow Diagram Logic Formalism Application for Software Development// Scientific Papers of University of Latvia, Volume 673, Databases and Information Systems, Doctoral Consortium, Sixth International Baltic Conference BalticDB&IS 2004, Riga, Latvia, June 6-9, 2004. -Riga: Latvijas Universitāte, 2004.- 220 - 231 p.

[13] Asnina E., Osis J. Formalization Problems and Perspectives of the Software Development// Scientific Proceedings of Riga Technical University, Computer Science, Applied Computer Systems, 3rd thematic issue- Riga: RTU, 2002. - 145-156 p.

[14] Asnina E., Osis J. The Computation Independent Viewpoint: a Formal Method of Topological Functioning Model Constructing// Scientific Proceedings of Riga Technical University, Series — Computer Science (5), Applied Computer Systems.- Riga: RTU, 2006. - in press.

[15] AT&T Research: The Real World (by Jackson M., 18/07/2003) / Internet. - http://www.ferg.org/papers/jackson--the real world.pdf

[16] Barr M., Wells C. Category theory for computer science, 2nd ed. - London: Prentice Hall International, 1995.-344 p.

[17] Bernardez B., Genero M., Duran A., Toro M. A Controlled Experiment for Evaluating a Metric-Based Reading Technique for Requirements Inspection// Proceedings of the 10th IEEE International Symposium on Software Metrics (METRICS'04).- IEEE. September 2004.- 257-268 p.

[41] Holt, Rinchart, Winston: Elements of Language: Examining Causes and Effects, 4th Course (2005) 92-103 p. / Internet. - http://go.hrw.com/elotM/003052667 l/student/ch03/ lg1403092 103.pdf
[42] Human and Technology: Structuring Use Cases with Goals (by Alistair Cockburn) / Internet. - http://alistair.cockburn.us/crystal/articles/sucwg/structuringucswithgoals.htm
[43] Informatique Thcorique ct Applications: A Category Theory Approach to Conceptual Data Modeling (by Lippe E., tcr Hofstde A.H.M., 1996) / Internet, -http://citeseer.nj.nec.com/lippe96category.html
[44] Institute for Automation and Control Processes, Far East Division, Russian Academy of Science: Experimental Version of the Petri Net Translator into the Executable Code: Petri Nets in Brief, 2000 / Internet. - http://www.iacp.dvo.ru/lab 1 l/otchet/ot2000/pn3.html (in Russian)
[45] Jensen K. An Introduction to the Theoretical Aspects of Coloured Petri Nets// A Decade of Concurrency, Lecture Notes in Computer Science vol. 803. - Springer-Verlag, 1994. - 230-272 p.
[46] Kent S. The Unified Modeling Language// Formal Methods for Distributed Processing: An 00 Approach. - 2000. - December. - 1-31 p.
[47] Knowledge Systems Corporation: Use Cases: the Pros and Cons / Internet. - http://www.ksc.com/article7.htm
148] Larman Cr. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd ed.- Prentice Hall PTR, 2005. - 703 p.
[49] Leffingwell D., Widrig D. Managing Software Requirements: a use case approach. 2nd ed. - Addison-Wesley, 2003.- 502 p.
[50] Logica UK Ltd.: Object Orientation in Z: Why an Object Oriented Z? (by cds. Stepney S., Barden R. and Cooper D., 1992) / Internet. - http://www-users.cs.york.ac.uk/~susan/bib/ss/ooz/cl .htm
[51] Markovitch Z., Markovitcha I. Modelling as a Tool for Therapy Selection// Proc. Of the 14th European Simulation Multiconference "Simulation and Modelling". - Ghent, Belgium, May 23-26, 2000.- 621-623 p.
[52] Markovitch Z., Rckners Ya. Synthesis of Systems Model on Basis of Topological Minimodels// Automatic Control and Computer Sciences, Vol. 32 (3). - New-York: Allerton Press Incorp., 1998. - 59-66 p.
[53] Markovitch Z., Stalidzans E. Expert Based Model Building Using Incidence Matrix and Topological Models// Proc. Of the 12th European Simulation Symposium "Simulation in Industry 2000". - Hamburg, Germany,, Sept. 28-30, 2000. - 328-332 p.
[54] Markovitcha I., Markovitch Z. Mathematical Model of Pathogenesis of Hard Differentiable Diseases (Matematicheskaja model' patogeneza trudno differenciirujemyh boleznej)// Cybernetics and Diagnostics (Kibenetika I Diagnostika), vol. 4. - Riga: Zinatne, 1970. — 21-28 p. (in Russian)
[55] Marshall D. Artificial Intelligence II. Courseware / Internet. http://www.cs.ef.ac.uk/Dave/AI2/AI notes.html
[56] Mathworld: First-Order Logic / Internet. - http://mathworld.wolfram.com/First-OrderLogic.html
[57] Mellor S.J., Clark A.N., Futagami T. Model-Driven Development// IEEE Software.- 2003. - September/October. - 14-18 p. (Internet.-http://www.cs.umb.edu/~ixs/courses/2005/681/rcadings/mda-mcllor.pdf)
[58] Meservy Th. O., Fenstermacher K. D. Transforming Software Development: An MDA Road Map// Computer. - 2005. - September (Vol. 38, Nr. 9). - 52-58 p. (Internet.- http://www.cs.umb.edu/~jxs/courses/2005/681/readings/mda-meservv.pdn
[59] Method of Dynamic Parameter Selection for Efficiency Check-up of Continuous Objects that arc Represented by Topological Models Taking into Account Non-Equivalence of Checks/ J. Osis, Z. Marckovich, J. Grundspenkis, J. Salenieks, V. Shaitsane. - All-Union Scientific Research Institute for Normalization in Engineering (VNIINMASH), Gorky Branch, Gorky, 1980. - 39 p. (in Russian)
[60] Miller J., Mukerji J. (cds.). OMG: MDA Guide Version 1.0.1, 2003 / Internet. - http://www.omg.org/docs/omg/03-06-01.pd
[61] Miller J., Mukerji J. (eds.): Model Driven Architecture (MDA). Architecture Board ORMSC, ormsc/2001-07-01 / Internet.- http://www.omg.org/docs/ormsc/01-07-01 .pdf
[62] Object-Oriented Software Engineering: A Use Case Driven Approach./ I.Jacobson, M.Christerson, P.Jonsson *et al.* - Addison-Wesley, 1992. - 528 p.
[63] OMG official web-page on MDA / Internet. - http://www.omg.org/mda
[64] OMG official web-page on UML / Internet. - http://www.omg.org/uml
[65] OMG: A Proposal for an MDA Foundation Model. ORMSC While Paper, V00-02, ormsc/05-04-01 / Internet. - www.omg.org/docs/ormsc/05-04-01.pdf
[66] OMG: OMG Unified Modeling Language Specification. Version 1.4. September 2001 / Intcrnct.- http://www.omg.org/docs/formal/0 l-09-67.pdf.

[67] OMG: UML Extension for Business Modeling, version 1.1 / Internet. - http://urnlcenter.visual-paradigm.com/umlresources/exte 11.pdf
[68] OMG: Unified Modeling Language (UML) Specification: Infrastructure. Version 2.0. ptc/2004-10-14. October 2004. / Intenet.- http://www.omg.org/cgi-bin/doc?ptc/2004-10-14
[69] OMG: Unified Modeling Language: Superstructure. Version 2.0. August 2005 / Internet, - http://www.omg.org/docs/formal/05-04-07.pdf
[70] Osis J. Brief Survey of Object-Oriented Approach// Scientific Proceedings of Riga Technical University, Computer Science, Applied Computer Systems, Vol. 3. - Riga, 2001. - 23-32 p.
[71] Osis J. Development of Object-Oriented Methods for Hybrid System Analysis and Design // Proceedings of the 23rd ASU Conference. - Stara Lesna, Slovakia, 1997. -162-170 p.
[72] Osis J. Diagnostics of Complex Systems (Dissertation of Dr. Habil. Sc. Eng.). - Riga: Latvian Academy of Sciences, 1972.
[73] Osis J. Extension of Software Development Process for Mechatronic and Embedded Systems// Proceeding of the 32nd International Conference on Computer and Industrial Engineering. - Limerick, Ireland, University of Limerick, August 11-13, 2003. - 305-310 p.
[74] Osis J. Investigating troubles of complex system functioning and category theory (Issledovanie narushenij funkcionirovanija slozhnyh system i teorija kategorij).// Cybernetic and Diagnosis (Kibernetika i Diagnostika), vol. 4. - Riga: Zinātne, 1970. - p. 15-20. (in Russian)
[75] Osis J. Mathematical description of complex system functioning (Matematicheskoe opisanie funkcionirovanija slozhnyh sistem).// Cybernetic and Diagnosis (Kibernetika i Diagnostika), vol. 4. - Riga: Zinātne, 1970. - p. 7- 14. (in Russian)
[76] Osis J. Programming Language ADA 1st part. Lections. - Riga: Riga Technical University, 1993. - 59 p. (in Latvian)
[77] Osis J. Software Development with Topological Model in the Framework of MDA// Proceedings of the 9th CaiSE/IFlP8.1/EUNO International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'2004) in connection with the CaiSE'2004, Vol. 1. - Riga: RTU, 2004. -211- 220 p.
[78] Osis J. Some questions of microprogramming optimization using topological model properties (Nekotorye voprosy optimizacii mikroprogramm s ispol'zovaniem svoistv topologicheskih modelei).// Cybernetic methods in the diagnosis (Kiberneticheskie melody v diagnostike).- Riga: Zinātne, 1973.- p. 30-34 (in Russian)
[79] Osis J. Topological functioning model support for software engineering// Scientific Proceedings of Riga Technical University, Computer Science, Applied Computer Systems, 4th thematic issue. - Riga, 2003. - 31-42 p.
[80] Osis J. Topological Functioning Model Within the MDA Life Cycle// Accepted to publishing in Scientific Proceedings of Riga Technical University, Computer Science, Applied Computer Systems. - 2006.
[81] Osis J. Topological Model of System Functioning// Automatics and Computer Science. - J. of Acad. Of Sc, Riga, Latvia #6, 1969. - 44-50 p. (in Russian)
[82] Osis J., Beghi L. Topological Modelling of Biological Systems// Proceedings of the third IFAC Symposium on Modelling and Control in Biomedical Systems (Including Biological Systems), D.A. Linkens, E.R. Carson (eds). - Oxford, UK: Elsevier Science Publishing, 1997 - 337-342 p.
[83] Osis J., Sukovskis U., Teilans A. Business Process Modeling and Simulation Based on Topological Approach// Proceedings of the 9th European Simulation Symposium and Exhibition. - Passau, Germany, 1997. - 496-501 p.
[84] Podcswa H. UML for the IT Business Analyst: A Practical Guide to Object-Oriented Requirements Gathering/ - Boston: Thomson Course Technology PTR, 2005. – 378 p.
[85] Robinson G.P. Model-Based Recognition of Anatomical Objects from Medical Images: Knowledge Representation / Internet. - http://noodle.med.yale.edu/alums/robinson/ivc/section3 2.html
[86] Schach St. R. Classical and Object-Oriented Software Engineering with UML and Java. International edition. 4th edn., - WCB/McGraw-Hill, 1999.
[87] Schneider G., Winters J.P. Applying Use Cases, 2nd cd. A Practical Guide. - The Addison-Wesley, 2001. - 245 p.
[88] Selic B. The Pragmatics of Model-Driven Development// IEEE Software. - 2003. - September/October. -19 - 25 p.

[89] Sharpies M., Hogg D., Hutchison Chr., Torrance S., Young D. Computers and Thought: A Practical Introduction to Artificial Intelligence. Chapter 6 / Internet. - http://www.informatics.susx.ac.uk/books/computers-and-thought/index.html
[90] Siau K., Rossi M. Evaluation of Information Modeling Methods - A Review// Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences-Volume 5.- IEEE, January 1998'- 314-322 p.
[91] Sindre G., Opdahl A.L. Eliciting Security Requirements with Misuse Cases// Requirements Engineering. - 2005. - October. - 34-44 p.
[92] Softeam: Making a Success of Preliminary Analysis using UML (by Ph. Desfray) / Internet. - www.obiecteering.com/pdf/whitepapers/us/modelisation des phases amont.pdf
[93] Sowa J.F. Conceptual Graphs, working draft IS0/JTC1/SC 32/WG2 N 000 (2001)/ Internet. - http://www.jfsowa.com/cg/cgstand.htm
[94] Sowa J.F. Conceptual Structures: Information Processing in Mind and Machine. - Addison-Wesley Publishing Company, 1984.-481 p.
[95] Sowa J.F. Knowledge Representation: Logical, Philosophical, and Computational Foundations.- Brooks Cole Publishing Co., Pacific Grove, CA, 2000. - 594 p.
[96] Sowa J.F. Towards the Expressive Power of Natural Language// Principles of Semantic Networks: Explorations in the Representation of Knowledge. - Morgan Kaufmann Publishers, Inc., 1991. - 157-190 p.
[97] The Object Agency: Be Careful With "Use Cases" (by Edward V. Berard, 1998)/ Internet. - www.toa.com/pub/use-cases.htm
[98] The Open University: Problem Frames and Software Engineering (by Jackson M., 09/12/2004) / Internet. - http://mcs.open.ac.uk/mi665/PFrame7.pdf
[99] The Unified Modeling Language Reference Manual/ J. Rumbaugh, I. Jacobson, and G. Booch. - Addison-Wesley, 1999.-568 p.
[100] Thomas D. MDA: Revenge of the Modelers or UML Utopia? // IEEE Software. - 2004. - May-June. - 22-24 p. (Internet. - http://www.martinfowler.com/ieeeSoftware/mda-thomas.pdf)
[101] University of Latvia: Introduction to Mathematical Logic. Hyper-textbook for students by Detlovs V., Podnieks K. - 2000-2004. / Internet.- http://www.ltn.lv/~podnieks/mlog/ml.htm
[102] University of Tasmania: Object Oriented Modelling with Object Petri Nets (by Lakos Ch., 1997) / Internet. - http://citeseer.nj.nec.com/lakos97object.html
[103] Vaidyanathan S. The Role of Model-Driven Architecture in Business Integration // Business Integration Journal. - 2004. - September. - 14-16 p. (Internet. - http://bijonline.com/PDF/vaidyanathan%20sept.pdf)
[104] Wikipedia, the free encyclopedia: Graph Theory / Internet. - http://en.wikipcdia.org/wiki/Graph theory