RĪGAS TEHNISKĀ UNIVERSITĀTE
Datorzinātnes un informācijas tehnoloģijas fakultāte
Lietišķo datorsistēmu institūts
Lietišķo datorzinātņu katedra

Jurijs GRIGORJEVS
Doktora studiju programmas „Datorsistēmas" doktorants

IEGULTO SISTĒMU NEFUNKCIONĀLO ĪPAŠĪBU UZ MODEĻIEM BALSTĪTA TESTĒŠANAS METODE

Promocijas darba kopsavilkums

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

Rīga 2012

UDK 004.031.6+004.2](043.2)
Gr 543 i

Grigorjevs J. Iegulto sistēmu nefunkcionālo īpašību uz modeļiem balstīta testēšanas metode. Promocijas darba kopsavilkums.-R.:RTU, 2012.-21 lpp.

Iespiests saskaņā ar DITF LD institūta 2011.gada 10. novembra lēmumu, protokols Nr.73.

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

ISBN 978-9934-10-280-6

PROMOCIJAS DARBS
IZVIRZĪTS INŽENIERZINĀTŅUDOKTORA GRĀDA IEGŪŠANAI RĪGAS TEHNISKAJĀ UNIVERSITĀTĒ

Promocijas darbs inženierzinātņu doktora grāda iegūšanai tiek publiski aizstāvēts 2012. gada 11. jūnijā Rīgas Tehniskās universitātes Datorzinātnes un informācijas tehnoloģijas fakultātē, Meža ielā 1/3, 202 auditorijā.

OFICIĀLIE RECENZENTI

Profesors, Dr.habil.sc.ing. Leonīds Novickis
Rīgas Tehniskā universitāte

Vecākais IS auditors, Dr.sc.comp. Mārtiņš Gills
Norvik Banka

Associate professor, doc. Ing., CSc. Jiří Kunovský
Brno University of Technology

APSTIPRINĀJUMS

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

Jurijs Grigorjevs …….(Paraksts)

Datums: 04.01.2012

Promocijas darbs ir uzrakstīts latviešu valodā, satur ievadu, 4 nodaļas, nobeigumu, literatūras sarakstu, 5 pielikumus, 37 attēlus, 5 tabulas, kopā 147 lapaspuses. Literatūras sarakstā ir 100 nosaukumi.

VISPĀRĪGS DARBA RAKSTUROJUMS

Ikdienā mēs saskaramies ar daudziem un dažādiem datoru vadītiem aparatūras līdzekļiem un tie veic arvien vairāk sadzīviskus un citus pakalpojumus un to klāsts katru dienu pieaug. Dažas vienkāršākās un populārākās no tām ir elektroniskā cepeškrāsns, mikroviļņu krāsns, ledusskapis, veļas mazgājamā mašīna, visāda veida signalizācijas un trauksmes sistēmas, liftu sistēma, dzelzceļa pārbrauktuves sistēma, bankomāts, biļešu un daudzas citas sistēmas. Minētās sistēmas pārsvarā var attiecināt pie nelielām un vidēja izmēra iegultām sistēmām. Savukārt, par lielo iegulto sistēmu piemēriem var minēt šādas: kravas un vieglās mašīnas, lidmašīnas, helikopteri, kuģi, dažādas militārās sistēmas, kosmosa kuģi, atomstacijas, dažādas automatizētās ražošanas un apkalpošanas industrijas roboti un citas sistēmas. Eksistē arī iegultās sistēmas, kas pēc savas būtības sastāv no minimālās programmatūras un dažiem aparatūras līdzekļiem. Pie šādas kategorijas parasti tiek attiecinātas vienkārša risinājuma apgaismojuma sistēmas, automātiski atveramo durvju sistēmas un citas. Galvenā atšķirība no iepriekšējo kategoriju sistēmām ir programmatūras un saslēgto aparatūras līdzekļu sarežģītība, kā arī izpildāmo funkciju nosacījumi un to kritiskums. Apskatot iegultās sistēmās, promocijas darbā autors fokusējas uz vidējā un lielā izmēra sistēmām.

Vidējā un lielā izmēra iegultās sistēmas apvieno to specifika un funkcionēšanas īpašības. Tās parādās tāpēc, ka rodas nepieciešamība vienlaikus vadīt vairākās ārējās ierīces, vākt un apstrādāt informāciju no ārējas vides, patstāvīgi darboties cilvēkam neiejaucoties un iespēju robežās turpināt funkcionēt nenozīmīgu kļūmju gadījumos. Lai nodrošinātu korektu sistēmas funkcionēšanu, iegulto sistēmu specifikai arī ir jābūt pārbaudītai visos iespējamos gadījumos. Par dotā pētījuma objektu ir izvēlētās iegulto sistēmu nefunkcionālās īpašības.

Aktualitāte

Iegultajām sistēmām piemīt vairāk nefunkcionālo īpašību, sarežģītākā programmatūras struktūra un tā izstrāde salīdzinājumā ar parastajām sistēmām. Pēdējo gadu laikā ir zināmi vairāki fakti par iegulto sistēmu programmatūras un aparatūras kļūmēm, kas noved pie cilvēku upuriem un milzīgiem zaudējumiem. Kaut arī visu lidmašīnu avāriju tehniskais iemesls ir ap 20% [ACRO 2011], kopējais skaits tomēr ir liels un tas ir atkarīgs no lidmašīnas programmatūras un aparatūras. Tā, piemēram, lielākās pēdēja laika lidmašīnu avārijas tehnisko iemeslu dēļ ir [LEY 2010] [BEA 2011] [BBC 2009]. Līdzīgi lidmašīnu problēmām šādas eksistē un regulāri parādās arī kosmiskajos aparātos [SVO 2011], kur atšķirībā no lidmašīnām tehnisko kļūmju statistika ir tuva 100%, jo tie tiek izlaisti orbītā balstoties uz iepriekš ieprogrammētiem parametriem cilvēkam neiejaucoties. Iegulto sistēmu kļūmes tiek atklātas arī mazākās sistēmās, piemēram, automašīnās [VWREC] , kafijas automātos, mobilajos telefonos un citur. Īpaši aktuāls iegulto sistēmu izstrādē kļūst sistēmu testēšanas uzdevums.

Lai padarītu par drošu iegulto sistēmu izstrādi, tiek ieviestas jaunās metodoloģijas, programmēšanas un specificēšanas valodas un tiek ražoti atbalsta rīki. Modernajā programmatūras izstrādē automatizācija kļūst arvien populārāka. Pirmkoda ģenerācija ir zināma un vairākus gadus tiek pielietota, bet joprojām pilna izstrādes cikla automatizācija tiek pētīta un ir virzībā uz manuālās darbības aizvietošanu. Modeļvadāmās arhitektūras (angl. *Model-Driven Architecture* turpmāk MDA) [MDA] standartizētie principi, pieejamas izstrādes vides un rīki stimulē pilna programmatūras izstrādes dzīves cikla automatizāciju, tai skaitā arī testēšanas uzdevuma izpildē, kas parasti aizņem ap 50% no visa izstrādes laika. Viens no MDA izmantotiem līdzekļiem ir vienota modelēšanas valoda (angl. *Unified Modeling Language* turpmāk UML) [UML], kas testēšanas procesa atbalstam piedāvā testēšanas profilu [UTP], kas savukārt nodrošina testēšanas procesa artefaktu specificēšanu standartizētā veidā. Kaut arī kopš 2005. gada, kad testēšanas profils tika standartizēts, joprojām neeksistē vispārīgi pieņemto metožu testēšanas procesa automatizācijai un testpiemēru ģenerēšanai no sistēmas modeļa.

Esošo risinājumu pārskatīšana

Eksistējošas iegulto sistēmu testēšanas metodes nav pilnīgas un nenodrošina testēšanas procesa automatizēšanu un atbilstību modernajām programmatūras izstrādes tendencēm. Metodes pārsvarā balstās uz vispārīgiem testēšanas standartiem un paņēmieniem. Ir zināmās specifiskās metodes, piemēram, nodrošināt pēc iespējas lielāko mezglu un ceļu pārklāšanu, automātiski ģenerējot testpiemērus iegūstot maksimālo pārklāšanu [STH 2001] [WU 2007]. Metodei ir vairākas priekšrocības, ieskaitot metodes autoru uzstādījumu par automātiskās testpiemēru ģenerēšanas nepieciešamību. Tomēr šāda pārklāšana nevar nodrošināt iegulto sistēmu nefunkcionālo īpašību verificēšanu veiksmīgos un neveiksmīgos gadījumos.

Eksistējošo testēšanas metožu formalizācijas mēģinājumi [AGR 2001] [MAT 1995] [KRS 2002] [ZHI 1999] [BRO 2003] [STI 2008], neatbalsta automātisko testpiemēru ģenerēšanu no sistēmas modeļa, kas ir viens no uzdevumiem risināms promocijas darba ietvaros.

Pētījuma mērķis

Promocijas **darba mērķis** ir piedāvāt uz MDA principiem un standartiem balstītu testēšanas metodi, kas dod iespēju automātiski ģenerēt testpiemērus iegulto sistēmu nefunkcionālo īpašību testēšanai. Piedāvātai metodei jānodrošina iegulto sistēmu nefunkcionālo īpašību verificēšana, balstoties uz vispārpieņemtiem sistēmu modelēšanas standartiem.

Pētījuma uzdevumi

Darba mērķa sasniegšanai autors izvirza **šādus uzdevumus**:

1. analizēt eksistējošās testēšanas metodes un klasificēt tās;
2. izpētīt iegulto sistēmu nefunkcionālās īpašības un to testēšanas metodes un paņēmienus;
3. veikt MDA principu analīzi un izvērtēt iespēju pielietot tos testpiemēru ģenerēšanā;
4. definēt testēšanas metodi, kas nodrošinātu testpiemēru ģenerēšanu no UML modeļiem;
5. pielietot piedāvāto metodi iegultās sistēmas nefunkcionālo īpašību testēšanai un secināt par metodes pielietošanas iespējām.

Pētījuma metodes

Balstoties uz uzdevumu rezultātiem, tiek iegūts priekšstats par testēšanas specifiku, par nepieciešamo testēšanas veidu automatizāciju un modernām automatizācijas metodēm, kā arī tiek analizētas un definētas iegulto sistēmu nefunkcionālās prasībās un to eksistējošas modelēšanas un testēšanas metodes. Darbā tiek izdalītās šādas iegulto sistēmu nefunkcionālās īpašības: laicīgums, asinhronā darbība, sinhronizācija, uzdevumu plānošana un drošums. Balstoties uz MDA principiem un aprakstītām īpašībām, tiek izvirzīta hipotēze par testēšanas metodi automātisko testpiemēru ģenerēšanai. Hipotēze sniedz vīziju par testēšanas metodes eksistenci, tās pielietojumu un tajā pašā laikā hipotēze nodrošina eksistējošo MDA standartu pielietošanu modeļu prezentēšanā, apstrādē un transformēšanā testēšanas modelī. UML un XML metadatu apmaiņas (angl. *XML Metadata Interchange* – XMI) standartu, kā arī vispārīgo MDA principu pielietošana veicina hipotētiskās metodes efektivitāti un universālo integrēšanu dažādos izstrādes procesos. Piedāvātā metode netiek saistīta ar kādu konkrētu izstrādes rīku un atbalsta vispārpieņemtus standartus. Metodes galvenās universālās īpašības ir sistēmas modeļa atbalsts XMI 2.1 versijas formāta un rezultātā iegūto testpiemēru saglabāšana UML definētā testēšanas profilā. Pirmā īpašība nodrošina neatkarību no sistēmas modeļu izstrādes rīkiem, jo eksistējošie modelēšanas un MDA atbalsta rīki, piemēram [EA] vai [EMF], nodrošina modeļu eksportēšanu XMI formātā. Savukārt, UML testēšanas profila izmantošana ļauj strukturēt transformācijas rezultātu un glabāt to standartizētā veidā. Šāds princips atbalsta uzģenerēto rezultātu turpmāko pielietošanu dažādos testēšanas vadības rīkos, kas nodrošina UML standartizēto testa datu importu no ārējiem avotiem.

Hipotēzes pārbaude tiek veikta uz reālas maksājuma karšu sistēmas, lietojot reālus sistēmas modeļus ar laika ierobežojumiem. Maksājuma karšu sistēma Card Suite [TIE] atbilst reālā laika sistēmām ar „mīkstiem" laika ierobežojumiem (angl. *soft time constraints*), kas pieļauj laika ierobežojumu pārsniegšanu, bet tajā pašā laika pārsniegšanu skaits ir stingri ierobežots ar sistēmai

izvirzītām prasībām. Laika ierobežojumu pārsniegšana sistēmas darbībā var vest pie finansiālām sekām un tāpēc ir svarīgi ievērot un verificēt tos. Balstoties uz Card Suite sistēmas specifiku, tā var tikt izmantota par hipotēzes pārbaudes platformu. Piedāvātas testēšanas metodes validācija notiek salīdzinot automātiski ģenerēto rezultātu ar manuāli iegūtiem testpiemēriem, veicot modeļa analīzi pēc klasiskās testēšanas paņēmieniem.

Promocijas darba novitāte

Automātiskā testpiemēru kopas ģenerēšanas metode iegulto sistēmu nefunkcionālām prasībām, kas balstās uz MDA principiem un eksistējošiem standartiem, ir dotā pētījuma novitāte. MDA principi galvenokārt tiek pielietoti pirmkoda ģenerācijai un reti testpiemēru izveidei, pārsvarā funkcionālo sistēmas īpašību testēšanai. Promocijas darba autors piedāvā metodi, kas spēj nodrošināt tieši nefunkcionālo prasību verificēšanu balstoties uz modeļu transformācijas principiem MDA kontekstā.

Darbā metode tiek realizēta un aprobēta laicīguma īpašībām ar automātisko testpiemēru ģenerēšanu. Metodes aprobācijā aprakstītais piemērs nodrošina testpiemēru ģenerēšanu laicīguma verificēšanai vienībtestēšanas līmenī.

Pētījuma ietvaros piedāvātā metode balstās uz izstrādāto rīku pielietošanu, kas atbalsta modeļu transformāciju un testpiemēru ģenerēšanu, kam autors ir definējis specifisku rīku lietošanas secību, t.s. rīku ķēdi (angl. *tool chain*). Rīku izstrādē apzināti tika pielietots komponenšu bāzēts projektējums, sadalot modeļu ielasīšanu XMI formātā datu bāzē, modeļu vienkāršotu reprezentēšanu un pašu transformācijas daļu. Modeļu vienkāršošanas posms ir nepieciešams, lai nodrošinātu kvalitatīvu un saturisku transformācijas likumu definēšanu. Sadalīšana komponentēs ļauj nodrošināt katras komponentes savstarpējo neatkarību un fokusēšanos uz konkrētām operācijām. Šāds princips ļauj pielāgot un papildināt eksistējošus rīkus tā, lai piedāvāta metode varētu tikt pielietota arī pārējo nefunkcionālo īpašību verificēšanai un atbilstu UML un XMI standartu attīstībai nākotnē.

Promocijas darba praktiskā nozīme

Vairāk kā 10 gadu strādājot Tieto Latvia uzņēmumā par reālā laika maksājuma karšu sistēmas testētāju, ieviešanas, testēšanas un integrācijas testēšanas nodaļu vadītāju, darba autors novēroja kā nefunkcionālo īpašību nepietiekama testēšana var novest pie veselas sistēmas dīkstāves un citām negatīvām sekām. Dotā pētījuma rezultāti ir izstrādāti, balstoties uz iegūto pieredzi reālā laika sistēmu testēšanā un modernām tendencēm programmatūras izstrādē, kā arī piedāvātā metode tika veiksmīgi pielietota finansu ziņojumu apstrādes scenārija verificēšanā no laika ierobežojuma viedokļa. Praktiskā pielietošana ļauj spriest par metodes funkcionēšanu un nepieciešamo testpiemēru ģenerēšanu, balstoties uz standarta sistēmas modeļiem.

Daži no promocijas darba rezultātiem tika izstrādāti vai lietoti šādos zinātniskajos un mācību metodiskajos projektos, kuros autors piedalījās kā izpildītājs:

- LZP lietišķo pētījumu projekta Nr. 09.1269 “Uz izkliedēta mākslīgā intelekta un tīmekļa tehnoloģijām balstītas metodes un modeļi intelektuālas lietišķās programmatūras un datorsistēmu arhitektūras izstrādei”, virziena “Programmatūras izstrāde MDA ietvarā” (2009.-pašlaik)
- Līdzdalība ESF līdzfinansētā projektā „Studiju moduļa izstrāde modeļvadāmai programmatūras attīstības tehnoloģijai datorsistēmas programmā” (Līgums Nr. 2007/0080/VPD1/ESF/PIAA/06/APK/3.2.3.2./0008/0007) (2006.-2008.)
- IZM/RTU zinātniski pētnieciskais projekts R7389 "Programmatūras sistēmas klašu struktūras ģenerēšanas rīka prototipa izstrāde, pamatojoties uz divpusložu modeli" (2008)

Iegūto darba rezultātu publikācijas un to prezentācija konferencēs

Promocijas darba rezultāti ir publicēti 10 rakstos starptautiski recenzējamos krājumos un 6 publikācijās vietējos krājumos. Rezultāti ir prezentēti 12 starptautiskajās konferencēs, ka arī 4 vietējas testēšanas konferencēs un vienā RTU studentu zinātniskajā un tehniskajā konferencē. Autora publikāciju saraksts ir pievienots izmantotās literatūras sarakstam kopsavilkuma beigās.

Promocijas darba publikācijas ir prezentētas šādās starptautiskās konferencēs:

1. Grigorjevs J. „Model-Driven Testing Approach for Embedded Systems Specifics Verification Based on UML Model Transformation", 3rd International Workshop on Model-Driven Architecture and Modeling-Driven Software Development June 8-11, 2011, Beijing, China
2. Grigorjevs J. „Card Suite Testing Toolbox – product launch case study", 12th Annual Software Testing Conference TAPOST2011, May 26, 2011, Riga, Latvia
3. Grigorjevs J. „Model-Driven Testing Approach Based on UML Sequence Diagram", RTU 51st International Scientific Conference, October, 2010, Riga, Latvia
4. Grigorjevs J. „Several practical approaches in testing automation", 11th Annual Software Testing Conference TAPOST2010, July6, 2010, Riga, Latvia
5. Grigorjevs J., Nikiforova O. „Several Outlines on Model-Driven Approach for Testing of Embedded Systems", RTU 50th International Scientific Conference, October, 2009, Riga, Latvia
6. Nikiforova O., Pavlova N., Grigorjevs J. „Several Facilities of Class Diagram Generation from Two-Hemisphere Model in the Framework of MDA", 23rd International Symposium on Computer and Information Sciences, ISCIS 2008, 27-29 October, 2008, Istanbul, Turkey
7. Grigorjevs J., Nikiforova O. "Complience of Popular Modelling Notations to Non-functional Requirements of Embedded Systems", International Scientific Conference Informatics in the Scientific Knowledge, June, 2008, Varna, Bulgaria
8. Grigorjevs J. "Testing of Embedded System's Non-functional Requirements", International Baltic Conference Baltic DB&IS 2008, June 2-5, 2008, Tallinn, Estonia
9. Grigorjevs J., Nikiforova O. "Modelling of Non-Functional Requirements of Embedded Systems", 42nd Spring International Conference MOSIS2008, April 22-24, 2008, Hradec nad Moravici, Czech Republic
10. Grigorjevs J., Nikiforova O. „Features of embedded systems that require specific testing approaches", RTU 48th International Scientific Conference, October, 2007, Riga, Latvia
11. Grigorjevs J., Nikiforova O. „Testing Process Adjustment for Real Time Systems", RTU 47th International Scientific Conference, October, 2006, Riga, Latvia
12. Grigorjevs J., Nikiforova O. „Unit Testing for Real Time Systems", RTU 46th International Scientific Conference, October, 2005, Riga, Latvia

Promocijas darba struktūra

Darbs ir strukturēti sadalīts 4 nodaļās. Pirmās divas nodaļas ir veltītas pētījuma tēmas apskatam, analīzei un autora hipotēzes izvirzīšanai. Nākamajās nodaļās detalizēti tiek aprakstīta piedāvātā metode un tās aprobācija uz reālā laika sistēmas piemēra, kā arī metodes novērtēšana.

Pirmajā nodaļā ir sniegts ieskats iegulto sistēmu specifikā, tiek apskatītas dažāda veida iegultās sistēmas un to kopīgās īpašības. Turpat tiek apskatīti vispārīgie testēšanas principi, atsevišķi izdalot iegulto sistēmu testēšanas specifiku. Nodaļas turpinājumā autors pievēršas iegulto sistēmu specifikai un to nefunkcionālām īpašībām. Detalizēti tiek definēta katra no tām, apskatot īpašības būtību, specificēšanas, modelēšanas un testēšanas paņēmienus.

Modernās tendences programmatūras izstrādē un modeļvadāmās izstrādes principi ir aprakstīti darba 2. nodaļā. Nodaļā tiek apskatīts testēšanas process un programmatūras izveidošanas process ar MDA rīkiem un tiek izvirzīta hipotēze par jaunu testēšanas metodi, balstītu uz MDA transformācijas principiem, kas būtu pielietojama iegulto sistēmu nefunkcionālo īpašību testēšanai. 3.nodaļa ir veltīta detalizētai metodes definēšanai un aprakstam. Metodes piedāvātā realizācija tiek fokusēta uz laicīguma īpašību verificēšanu un pēc savas struktūras pārredz tās pielietošanu arī pārējo nefunkcionālo īpašību testēšanai. Metodes demonstrācijas piemērs sniedz iespēju iepazīties ar tās darbības principiem un veikt sākotnējo izvērtēšanu.

4.nodaļa ir veltīta metodes aprobācijai uz reālā laika maksājuma karšu sistēmas. Nodaļā tiek sniegts verificējamās sistēmas vispārīgs apraksts ar tās funkcionēšanas un realizācijas īpašībām. Balstoties uz aprobācijas rezultātiem, autors veic metodes izvērtēšanu, aprakstot tās trūkumus, priekšrocības un pielietošanas iespējas.

1. IEGULTO SISTĒMU TESTĒŠANAS PAMATI

Iegultās sistēmās ir tādas sistēmas, kuru sastāvdaļās ir aparatūra un programmatūra, un kas paredzētas specifiska lietojuma funkcionēšanai, cilvēkam neiejaucoties [JEN 2011].

Iegultās sistēmas ir specifiskās ar patstāvīgo funkcionēšanu un pieslēgto aparatūras līdzekļu vadību. Šāda specifika ienes papildprasības testēšanas procesam un pieprasa iegulto sistēmu īpašību speciālu verificēšanu. Iegulto sistēmu verificēšanā un validācijā pārsvarā pielieto klasiskās testēšanas metodes un tikai daļa no pieejām apskata šādu sistēmu specifikas papildus verificēšanu. 1. nodaļā autors sniedz eksistējošo testēšanas metožu apskatu un klasificēšanu, definē iegulto sistēmu nefunkcionālās īpašības un apskata to modelēšanas un testēšanas paņēmienus.

1.1. Iegulto sistēmu īpašības

Iegulto sistēmu dažādība ir plaša, taču to vieno neatņemama saskarne ar ārējo vidi un pieslēgto ierīču vadība. Šī specifika arī definē papildus prasības iegulto sistēmu uzbūvei un funkcionēšanas principiem. Iegultām sistēmām ir jānodrošina šādas iespējas [ZIM 2009]:

- Fiziskā sasaiste (angl. *physical coupling*) – jāmāk transformēt fiziskos fenomenus analoga datos (to nodrošina visu veidu sensori).
- Saskarnes (angl. *interfaces*) – jāmāk no analoga datiem dabūt programmatūrai saprotamus datus.
- Laicīgums (angl. *timing*) – jānodrošina laika ierobežojumu specificēšana un to sasniegšana [GRI 2005] [GRI 2006].
- Resursu kontrole (angl. *resource control*) – jāmāk pārvaldīt resursi.
  - Asinhronā darbība (angl. *asynchronisms*) – parasti nodrošina ar pārtraukumu apstrādes mehānismu, kas ļautu paralēli darboties vairākām saskarnēm un programmatūras daļām.
  - Datu sinhronizācija (angl. *synchronization*) – līdzi ar pārtraukumu pārķeršanu ir jānodrošina arī citi sinhronizācijas mehānismi, piemēram, semafori.
  - Uzdevumu izpildes plānošana (angl. *scheduling*) – multiapstrādes sistēmās vienmēr parādās konkurējošie procesi, tāpēc ir jāmāk atbilstoši izstrādātai stratēģijai vienlaicīgi apstrādāt vairākus uzdevumus.
- Ticamība, drošums (angl. *reliability*) – programmatūrai jābūt bojājumpiecietīgai (angl. *fault tolerant*) un programmas kompilēšanas laikā ir jāmāk pārķert noteiktā tipa kļūdas (piemēram, tipizēšanas un sintakses kļūdas).

Visas šīs iespējas nav unikālās tieši iegultām sistēmām un dažas no tām piemīt arī citām sistēmām. Tā, piemērām, laicīgums ir reāla laika sistēmu neatņemama sastāvdaļa, savukārt, datu sinhronizācija un uzdevumu plānošana ir obligāta prasība daudz procesu sistēmām, bet fiziskās sasaistes līdzekļi ir sastopami, ikdienā strādājot ar peli vai tastatūru. Tajā paša laikā, visām iegultām sistēmām visas šīs prasības ir obligātās.

1.2 Iegulto sistēmu testēšanas pamatprincipi un metodes

Testēšana ir sistēmas darbināšanas un analīzes process, ar mērķi gūt pārliecību, ka tā spēs ilglaicīgi funkcionēt noteiktajos apstākļos un izpildīt iepriekš noteiktās darbības. Pēc autora viedokļa izsmeļošākās testēšanas definīcijas ir šādas:

Testēšana ir programmas izpildīšanas un apskatīšanas process ar mērķi atrast kļūdas [MYE 2004] – vispārīgs testēšanas definējums.

Sistēmas/komponentes darbināšanas process ar noteiktiem nosacījumiem, pārbaudot sistēmas/komponentes kādu aspektu. Darbināšanas procesa rezultāti tiek pierakstīti un analizēti [COP 2004] – vairāk formāls definējums, ko piedāvā IEEE Standard 610.12-1990.

Iegulto sistēmu testēšana konceptuāli atbilst vispārīgām un vispārpieņemtām testēšanas procesam. Tas sastāv no sekojošiem posmiem [SPI 2007]: plānošana, specificēšana, darbināšana, pierakstīšana (piemēram, rezultātu dokumentēšana), pārbaude par plānoto darbu izpildi un slēgšanas aktivitātes (piemēram, gala rezultātu izvērtēšana).

Promocijas darba 1.2. sadaļā ir aprakstītas un klasificētas vairākas testēšanas metodes, kas var tikt pielietotas iegulto sistēmu testēšanā. Īpaša uzmanība ir veltīta uz modeļu transformāciju balstītām metodēm, tomēr kaut arī aprakstītas metodes var tikt lietotas iegulto sistēmu testēšanai, tās pārsvarā koncentrējas tieši uz praktisko funkcionālo prasību testēšanas procesu un tā realizāciju.

Iegulto sistēmu īpašības, tās modelēšanas un reprezentēšanas iespējas, kā arī nefunkcionālo prasību testpiemēru kopas ģenerēšana ir joprojām neatrisināts uzdevums pētnieku interešu lokā.

Apakšnodaļā uzskaitītie pētījumi iegulto sistēmu testēšanas jomā pārsvarā fokusējās uz sistēmu funkcionālo īpašību verificēšanu, nodrošinot ātrāku un plašāku testpiemēru ģenerēšanu un izpildi. Savukārt, metodes, kas paredz nefunkcionālo īpašību verificēšanu, balstās uz vecām modelēšanas notācijām, tādejādi fokusējoties uz konkrētas īpašības pārbaudi, nelietojot modernās universālās modelēšanas iespējas un rīkus. Šādu apgalvojumu apstiprina arī citi pētnieki, kas analizēja dažādas metodes un pieejas gan funkcionālo, gan nefunkcionālo īpašību verificēšanai un to sarežģītību iegultajās sistēmās. Par tādiem var minēt iegulto sistēmu projektēšanas procesa pētniekus [ACK 2008], iegulto sistēmu kompilatoru realizācijas metodes autorus [KUG 2008], kā arī MARTE ietvara autorus [PER 2010] u.c. Šī promocijas darba autora pētījums fokusējās tieši uz sistēmas nefunkcionālo īpašību testēšanu.

1.3 Iegulto sistēmu īpašības un to modelēšanas un testēšanas paņēmieni

Iegulto sistēmu nefunkcionālās īpašības ietver asinhrono darbību, sinhronizāciju, laicīgumu, uzdevumu plānošanu un drošumu [GRI 2007]. Asinhronā darbības, sinhronizācijas un laicīguma īpašības tiek implementētas programmatūrās abos gadījumos, kad tiek un netiek lietota operētājsistēma. Savukārt, uzdevumu plānošana pārsvarā tiek attiecināta uz operētājsistēmas funkcionalitāti, kā viena no īpašībām, kas ir jānodrošina. Tieši tāpēc mūsdienās arvien biežāk iegulto sistēmu programmatūra neimplementē šo īpašību un atstāj to operētājsistēmas pārziņā. Drošuma īpašību nodrošināšana ir komplicēts un joprojām pētīts jautājums. Kaut arī pastāv konkrētas metodes drošuma izskaitļošanai un modelēšanai, no implementācijas viedokļa šī īpašība joprojām ir atsevišķs pētniecības objekts. Lai pietiekami detalizēti izklāstītu iegulto sistēmu nefunkcionālo īpašību verificēšanu un tajā pašā laikā fokusējoties uz aktuālākām nefunkcionālām īpašībām, darba autors promocijas darba ietvaros apskata asinhrono darbību, sinhronizāciju un laicīgumu, kas detalizēti ir aprakstītas 1.3. sadaļā.

2. MODEĻVADĀMĀS ARHITEKTŪRAS BĀZES ELEMENTU ATTĒLOŠANA TESTĒŠANAS ARTEFAKTOS

Iegulto sistēmu specifika uzstāda paaugstinātas prasības to nefunkcionālo īpašību testēšanai. UML dod iespēju testēšanas uzdevuma izpildīšanai veidot sistēmas modeļus, kas detalizēti atspoguļo katras nefunkcionālās īpašības būtību. 2. nodaļā autors izvirza hipotēzi par to, ka ir iespējams modeļvadāmās programmatūras izstrādes principus izmantot iegulto sistēmu verificēšanas automatizācijai.

2.1 Modeļvadāmās arhitektūras pamata principi

Ir zināmas vairākas MDA definīcijas un pēc autora domām īsi un konstruktīvi MDA būtību atspoguļo šādas definīcijas:

„MDA ir informāciju sistēmu specificēšanas pieeja, kas atdala funkcionalitātes specificēšanu no šīs funkcionalitātes implementēšanas uz specifiskās tehnoloģijas platformas specificēšanās” [MDA].

„MDA būtība ir dažādu modeļu veidošana dažādos abstrakcijas līmeņos un tālāka šo modeļu saistīšana ar mērķi implementēt sistēmu. Daži no modeļiem eksistē neatkarīgi no programmatūras platformas, tajā pašā laikā citi modeļi ir specifiski konkrētai platformai. Katrs modelis tiek veidots lietojot tekstu un vairākās papildinošās un savā starpā saistītas diagrammas” [MEL 2004].

Modeļvadāmās arhitektūras princips balstās uz modeļa būtību, kas atspoguļo un specificē sistēmas funkcionēšanu un modeļu transformāciju [KLE 2003], [NIK 2007]. MDA pamatjēdzieni un principi ir izskaidroti promocijas darba 2.1. nodaļā.

2.2 Modeļvadāmās arhitektūras principu pielietošana testēšanā

Modeļvadāmajā arhitektūrā galvenie artefakti ir modeli, jo tie apraksta gan sistēmas biznesa funkcionalitāti, gan iekšējās tehniskās īpašības un kalpo par avotiem koda ģenerācijas procesā. Klasiskajā testēšanas procesā testu gadījumu veidošanā tiek lietoti sistēmas funkcionālie apraksti, kas modeļvadāmajā arhitektūrā var tikt realizēti modeļu veidā. Šis fakts norāda uz to, ka testpiemēri var tikt veidoti no modeļiem. Šāds process tiek saukts par uz modeļiem balstītu testēšanu (angl.

model-based testing vai *model-driven testing*) – programmatūras testēšana, kur testpiemēri tiek pilnīgi vai daļēji iegūti no modeļa, kas apraksta kādus (vai visus) testējamās sistēmas aspektus [ENG 2006]. Modeļu abstrakcijas līmeņa dažādība nodrošina un atbalsta atbilstošu testēšanas paveidu. Aplūkojot MDA bāzēto un klasisko programmatūras izstrādes un testēšanas secību, var pamanīt, ka abstrakciju līmeņu artefakti paliek nemainīgie. Promocijas darbā ir secināts par testēšanas procesa V modeļa līmeņu analoģiju ar modeļvadāmās izstrādes ķēdi. Šī analoģija ir viens no aspektiem, kas ir ņemts vēra autora piedāvātas testēšanas metodes izstrādē.

Lai nodrošinātu vienotu pieeju testēšanas artefaktu vadībai un strukturēšanai OMG [OMG] ierosināja izstrādāt testēšanas profilu, kas atbalstītu modeļvadāmo testēšanas procesu. 2007. gadā tika publicēta UML testēšanas profila (angl. *UML testing profile* – UTP) versija 1.0, kas ir bāzēta uz vispārīgo UML 2.0 versiju un kas joprojām ir pēdējā un aktuālā versija. UTP definē valodu testējamo sistēmu artefaktu projektēšanai, vizualizēšanai, specificēšanai, analīzei, konstruēšanai un dokumentēšanai [UTP]. UTP var tikt lietots atsevišķi vai arī var tikt integrēts ar sistēmas UML aprakstu, lai aprakstīt sistēmas un testēšanas artefaktus kopā. UTP paplašina vispārīgo UML ar tādiem testēšanas specifiskiem konceptiem kā testu konteksts (angl. *test context*), spriedums (angl. *verdict*), testpiemērs (angl. *test case*), testu uzvedība (angl. *behavior*) un citiem. UTP ir vēl viens aspekts, uz ko balstās autora piedāvāta testēšanas metode.

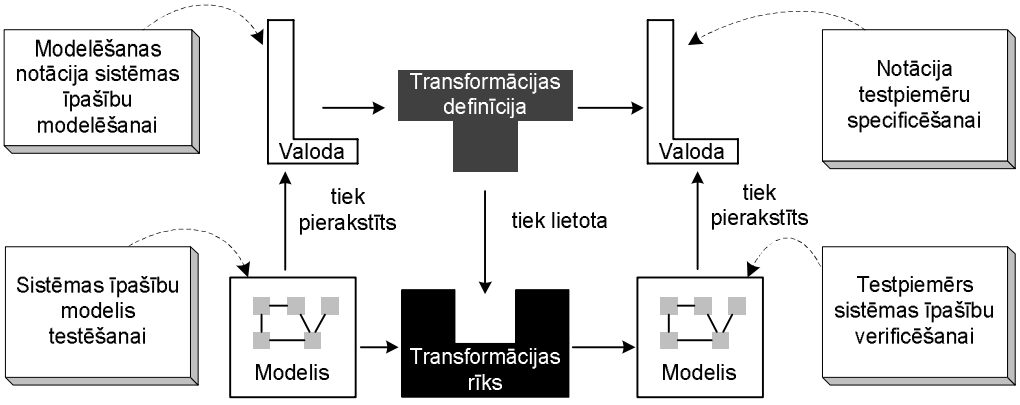
2.3 Hipotēze par iespēju iegulto sistēmu nefunkcionālo īpašību testēšanu balstīt uz MDA principiem

UTP standarta līdzekļi piedāvā funkcionālo īpašību testpiemēru ģenerēšanu no sistēmas modeļiem [ZAN 2009]. Savukārt, nefunkcionālo īpašību testēšana ar MDA principiem joprojām tiek pētīta. Promocijas darba 2.3. sadaļā autors izvirza **hipotēzi**:

Sistēmas atbilstības testēšanu nefunkcionālajām prasībām var balstīt uz vispārīgiem MDA modeļu transformācijas principiem, kas, savukārt, var tikt lietoti automātiskai testpiemēru ģenerācijai iegulto sistēmu nefunkcionālo īpašību verificēšanai.

Modeļvadāmās programmatūras izstrādes principi ir balstīti uz modeļu transformāciju, kur jaunie modeli tiek ģenerēti no eksistējošiem modeļiem. Transformācija starp modeļiem tiek nodrošināta ar transformācijas definīciju – transformācijas likumu kopa, kas ir pierakstīta nepārprotamās specifikācijas formā, kur daļējais vai vesels modelis tiek lietots cita vesela vai daļējā modeļa izveidei [KLE 2003]. Transformācijas likumi tiek pierakstīti noteiktajā transformācijas definīcijas valodā. Lai nodrošinātu testēšanas modeļa ģenerēšanu, darbā tiek piedāvāta transformācijas valodas definīcija, kas ir fokusēta uz testēšanas artefaktu ģenerēšanu un būtu viegli lietojama transformācijas likumu izstrādes laikā.

Automātiskā modeļu transformācija var tikt realizēta ar speciālu rīku, kas ir spējīgs atpazīt un strādāt ar avota un mērķa modeļa objektiem, pielietojot definētus transformācijas likumus avota modelim. Transformācijas rezultātā tiek iegūts mērķa notācijai atbilstošs modelis ar atbilstošiem uzģenerētiem objektiem. Vispārīgs modeļvadāmās programmatūras izstrādes princips definē vispārīgo shēmu modeļu transformācijai atbilstoši attiecīgām transformācijas shematiskām attēlojumam no [KLE 2003]. 1. attēlā ir parādīts izvirzītas hipotēzes koncepts, kas ir projicēts vispārīgajā modeļu transformācijas shēmā.



1. att. Autora piedāvātā hipotētiskā risinājuma attēlojums transformācijas koncepcijas shēmā [GRI 2009]

Par avota modeli autors piedāvā uzskatīt attiecīgās nefunkcionālās īpašības modeli, kas ir veidots UML secību/stāvokļu diagrammas notācijā. Tādējādi UML valoda ir uzskatīta par modelēšanas notāciju, kas specificē avota modeli. Par mērķa modeli autors piedāvā uzskatīt testēšanas piemērus, kas ir izteikti UML testēšanas profilam atbilstošā pieraksta formā. Tādējādi UML testēšanas profila notācija kalpo par mērķa modeļa modelēšanas valodu. Avota modeļa transformācija mērķa modelī tiek veidota izmantojot transformācijas definīciju, kuras pamatā ir abu modelēšanas valodu sintakse un semantika. Attēlā paradītie principi atspoguļo darbā izvirzīto hipotēzi un reprezentē testēšanas metodes pamata artefaktus ar saitēm starp tiem.

3. IEGULTO SISTĒMU ĪPAŠĪBU TESTĒŠANAS METODE

3. nodaļā autors detalizēti apraksta piedāvāto metodi iegulto sistēmu testēšanai, kas balstās uz 2. nodaļā izvirzītās hipotēzes apgalvojumiem un demonstrē metodes lietošanas principus uz abstrakta iegultās sistēmas fragmenta testēšanas piemēra. Kā arī autors analizē piedāvātās metodes priekšrocības un trūkumus un demonstrē efektu, ko dod metodes pielietojums testēšanas uzdevuma izpildē. Tas balstās uz identificēto testpiemēru komplektu analīzi, kas tiek iegūts ar metodes pielietošanu.

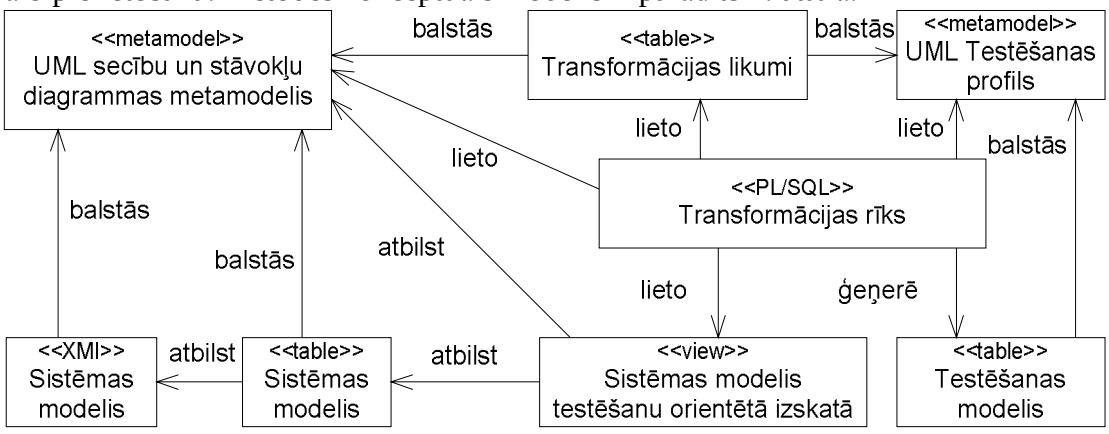
3.1 Avota un mērķa modeļu analīze

Iegulto sistēmu sinhronizācijas, asinhronās darbības un laicīguma īpašības var tikt modelētas ar UML standartizētām diagrammām [GRI 2008b] [GRI 2008c]. UML secību un stāvokļu diagrammas tika izvēlētas par avota modeļiem. UML secību un stāvokļu diagrammas ir pietiekama modelēšanas notācija, lai attēlotu sinhronizācijas, asinhronās darbības un laicīguma īpašības [GRI 2008a] [PIL 2005] [UML]. Darba 3.1. sadaļā ir definēts avota modeļa metamodelis, kas apraksta secību diagrammas elementus un savstarpējas attiecības starp tiem.

Par mērķa notāciju tika izvēlēts testēšanas profils, ko piedāvā UML. UTP paredz melnās kastes testēšanas principu [COP 2004] [UTP], kad par testējamās programmatūras uzbūvi nav informācijas un testēšana notiek ārēji, izsaucot testējamo objektu dažādos nosacījumos. Neskatoties uz to, ka šīs pētījums balstās uz sistēmu īpašību verificēšanu, kas pēc būtības ir iekšējo procesu analīze un atbilstošu testpiemēru ģenerēšana, UTP tiek izvēlēts par piemēroto metamodeli testēšanas datu vadībai. Tas ir aprakstīts 3.1. sadaļā.

3.2 Metodes detalizēts apraksts

Testpiemēru ģenerēšanas metode balstās uz izvirzīto hipotēzi par šādas metodes eksistēšanu un potenciālo pielietošanu. Metodes konceptuāls modelis ir parādīts 2. attēlā.



2. att. Piedāvātās testēšanas metodes konceptuāls modelis [GRI 2011b]

2. attēlā prezentētais modelis parāda piedāvātās metodes svarīgākus konceptus un atkarības starp tiem. Izvirzītā metode balstās uz UML modelēšanas notācijas secību un stāvokļu diagrammām un UML testēšanas profilu. Diagrammā abas notācijas ir atzīmētas ar <<*metamodel*>> stereotipu, lai parādītu šo konceptu būtību un atšķirību no pārējiem konceptiem. Abi metamodeli balstās uz MOF principiem un ir detalizēti aprakstīti 3.1 sadaļā.

Sistēmas modeļi, kas kalpo par metodes sākotnējiem modeļiem, atbilst UML secību un stāvokļu diagrammu modelēšanas principiem. Metode paredz sistēmas modeļu prezentēšanu XMI formātā [XMI], detalizētāk par formātu ir aprakstīts sadaļā 3.3), kas tiek parādīts diagrammā ar atbilstošu <<*XMI*>> stereotipu. XMI formāts plaši tiek pielietots kā koplietošanas standarts UML modeļu apmaiņai starp vairākiem izstrādes procesa atbalsta rīkiem.

Implementējot secību diagrammu kartēšanu no XMI formāta uz tabulām, tiek nodrošināta sistēmas modeļu jaunā tehniskā prezentācija. Tas nozīme, ka sistēmas modeli tabulās pēc būtības un satura ir tie paši modeli, kas ir pieejami XMI formātā. Tajā pašā laikā, tas arī nozīme to, ka tie atbilst arī jau minētājam UML secību un stāvokļu diagrammu metamodelim. Līdzīgi ar <<*XMI*>> stereotipu, <<*table*>> stereotips diagrammā parāda modeļu prezentācijas formātu.

Uz testēšanu orientēta sistēmas modeļu reprezentācija ir metodes koncepts, kas nodrošina apstrādājamo modeļu sagatavošanu transformācijas procesam. UML modeļu transformācija uz testpiemēriem ir specifiska ar to, ka lai uzģenerēt vienu testpiemēru, ir nepieciešams analizēt datus par vairākiem saistītiem objektiem. Metodes realizācijā paredz, ka uz datu bāzes tabulām pārnestie UML modeļi tiek attēloti skatu veidā (angl. *view*), parādot to diagramma ar atbilstošu <<*view*>> stereotipu. Pie šādas pieejas skati ir vienkāršotais kopsavilkums no komplicētā sistēmas modeļa, prezentējot testēšanai nepieciešamus aspektus. Skati tiek konstruēti balstoties uz sistēmas modeļiem, iekapsulējot nepieciešamos elementus un atribūtus pieprasītājā formātā. Skati neievieš jaunus datus un satur tikai un vienīgi oriģinālo modeļu datus. Balstoties uz šiem apgalvojumiem, var spriest par skatu atbilstību sākotnējo sistēmas modeļu metamodelim.

Transformācijas rīks nodrošina mērķa modeļa ģenerēšanu, t.i. transformācijas likumu nolasīšanu un apstrādi attiecībā pret vienkāršoto sistēmas modeli un testēšanas modeļa ģenerēšanu, kas atbilst UML testēšanas profilam. Rīka implementācija realizē noteiktās transformācijas likumu notācijas apstrādi, kas detalizēti ir apskatīta 3.2.1 nodaļā. Pats rīks ir realizēts PL/SQL programmēšanas valodā ar dinamisko SQL [DSQL].

Transformācijas likumu kopa, ir koncepts, kas apraksta nosacījumus sistēmas modeļu transformācijai uz testēšanas modeli. Likumi tiek saglabāti atsevišķajā tabulā, nodrošinot pieeju to modificēšanai un nolasīšanai transformācijas procesa laikā. Transformācijas likumi atbilst minētai iepriekš definētai notācijai, kas savukārt balstās uz UML secību un stāvokļu diagrammu un UML testēšanas profila metamodeļiem.

Mērķa modelis apraksta testēšanai nepieciešamus aspektus un atbilst UML testēšanas profilam. Līdzīgi sistēmas modeļu tabulu prezentācijai, testēšanas modelis sastāv no savā starpā saistītu tabulu kopas, attēlojot testēšanas profilā aprakstītus klasifikatorus. Katram klasifikatoram tiek sagatavota tabula, kurā transformācijas laikā tiek ierakstīti atbilstošie ģenerētie dati, balstoties uz transformācijas likumiem un sistēmas modeli.

Lai nodrošinātu transformāciju likumu pielietošanu sistēmas modelim un testpiemēru ģenerēšanu, tam ir jāsatur verificējamo elementu izvēles kritēriji un testējamās uzvedības kritēriji. Autors piedāvā notāciju transformācijas likumu definēšanai, kas ir apskatīta darba 3.2.1 nodaļā.

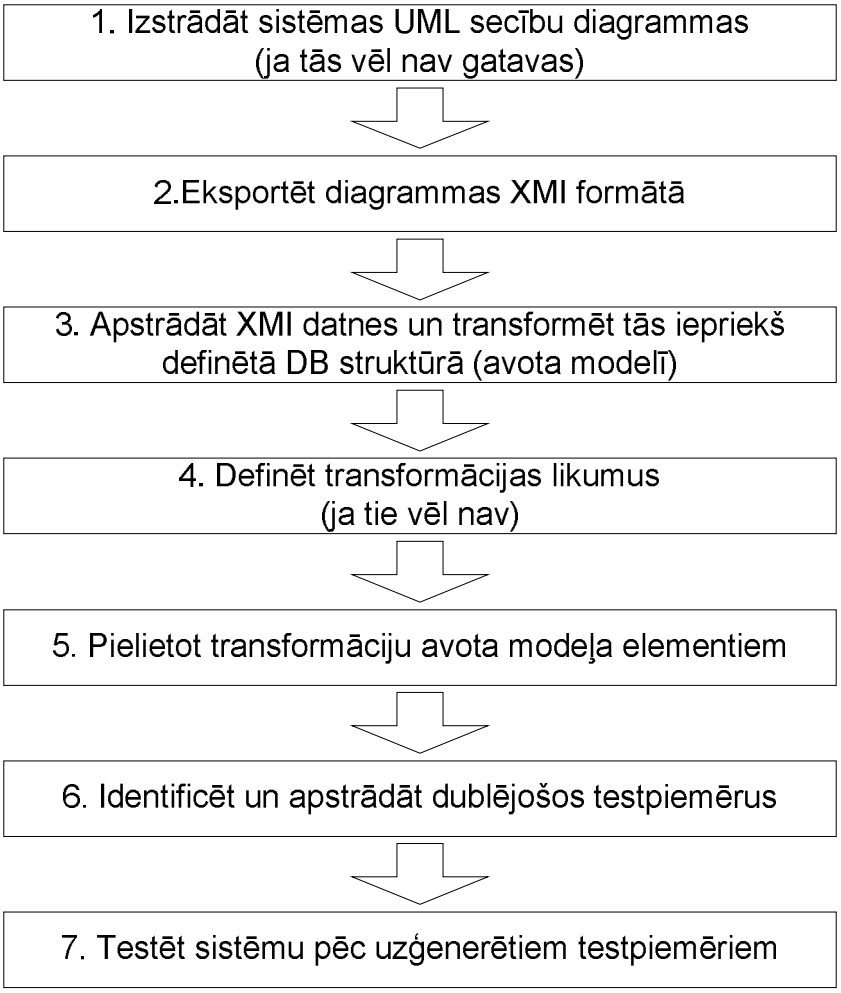
3.3 Metodes realizācijas principi

Metode paredz transformācijas implementēšanu divos soļos. Pirmajā solī notiek sākotnējo UML modeļu pārveidošana no XMI standarta datnes uz datu bāzes tabulām ar speciāli izstrādāto kartēšanas rīku. Ar iepriekš definētiem skatiem uz UML modeļiem, tiek iegūta vienkāršotā sistēmas modeļu prezentācija, kas tālāk tiek lietota transformācijās. Otrajā solī notiek pati transformācija, lasot un apstrādājot vienkāršotus UML modeļus ar definētiem transformācijas likumiem. Transformācijas rezultātā tiek iegūts testēšanas modelis ar uzģenerētiem testpiemēriem.

3.4 Metodes pielietošanas soļu secība

Metodes pielietošana sastāv no vairākiem soļiem, daļa no kuriem var tikt izlaista atkarībā no projekta specifikas. 3. attēlā ir parādīta vispārīgā metodes pielietošanas soļu secība, kas definē iepriekš aprakstītas testēšanas metodes nepieciešamās aktivitātes. Attēlā parādītā shēma prezentē nepieciešamus soļus, ieskaitot manuāli darbināmus un automatizētus. Ar automatizētiem soļiem tiek saprastas darbību, kuru rezultātu sagatavošanu nodrošina atbilstošs atbalsta rīks. Pie manuāliem

soļiem var attiecināt 1., 4. un 7. darbības, savukārt visas parejas aktivitātes tiek nodrošinātas ar standarta vai speciāliem, ka arī pētījuma ietvaros papildus izstrādātiem rīkiem.



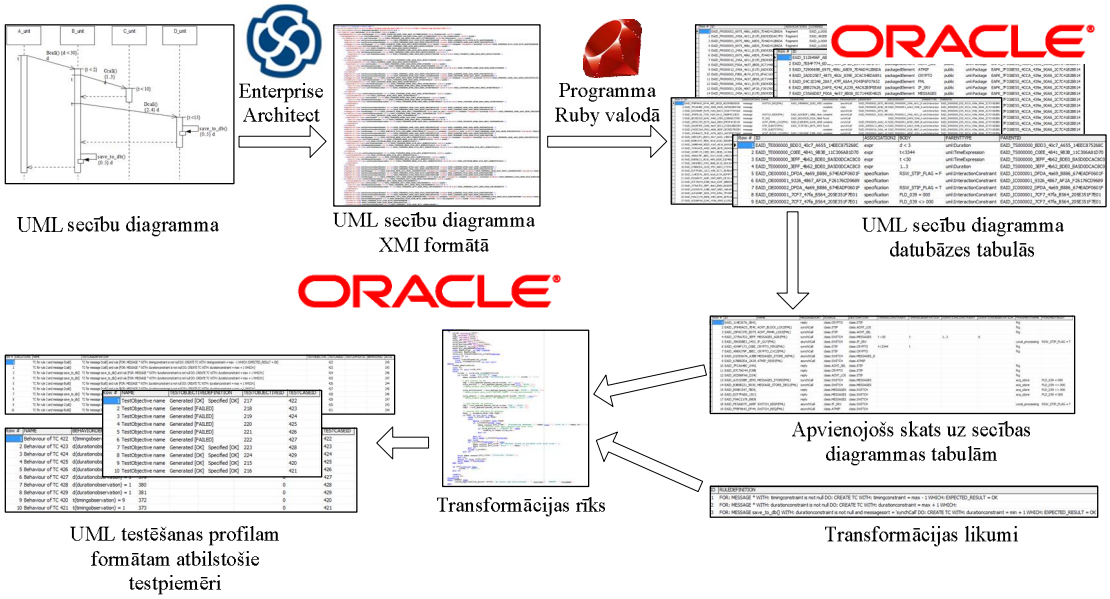
3. att. Piedāvātās metodes soļu secība

3. attēlā parādīta shēma apraksta vispārīgu darbību secību metodes pielietošanai. Metodes detalizēts pielietošanas apraksts ar tehniskam detaļām ir sniegts nākamajā sadaļā.

3.5 Metodes pielietošanas piemērs

Aprakstītas metodes demonstrācijai tiek piedāvāts tās pielietošanas piemērs testpiemēru ģenerēšanai abstraktai programmai ar sinhroniem izsaukumiem ar laika ierobežojumiem. Promocijas darbā piedāvātais risinājums paredz vairāku soļu realizāciju, kur katrā solī ir nepieciešams izmantot attiecīgas aktivitātes atbalsta rīku. Tādējādi izstrādātā algoritma pielietošanai ir definēta rīku ķēde (angl. *tool chain*), kuras vispārīga struktūra ir parādīta 4. attēlā.

Lai nodrošinātu sistēmas modeli transformācijai, metodes demonstrācija paredz programmas specificēšanu UML secību diagrammas veidā ar Enterprise Architect rīku. Šis komerciāls rīks nodrošina sistēmu modelēšanu, modeļu transformāciju uz kodu, testēšanas skriptu veidošanu un izstrādāto modeļu eksportēšanu XMI formātā. Pielietojot šo funkciju, rīkā izveidotā UML secību diagramma tiek eksportēta XMI formāta, nodrošinot sākotnējus datus aprakstītai testēšanas metodei. Atbilstoši 3.3.1 nodaļā aprakstītam XMI datņu apstrādes rīkam, tiek iegūta datne ar INSERT vaicājumiem datu ielādēšanai datubāzes tabulās. Pēc datu ielādes tabulās, tās satur pilnīgu sistēmas modeli, kas saturiski atbilst apstrādātai XMI datnei. Tālāk ielādētais modelis tiek reprezentēts skata veidā, attēlojot transformācijai nepieciešamus datus vienkāršotā un viegli apstrādājamā formātā.



4. att. Piedāvātās testēšanas metodes rīku ķēde [GRI 2011a] [NIK 2008]

Aktivizējot transformācijas rīku ar definētiem un ielādētiem datiem, tiek ģenerēts testēšanas modelis atbilstoši UML testēšanas profilam. Sistēmas modeļa apstrāde un testpiemēru ģenerēšana atbilst 3. nodaļā aprakstītiem principiem.

4. PIEDĀVĀTĀS TESTĒŠANAS METODES PIELIETOJUMS REĀLĀ LAIKA SISTĒMAS VERIFICĒŠANĀ

4. nodaļā autors veic piedāvātās metodes un izstrādāto rīku aprobāciju uz reālā laika maksājuma karšu sistēmas Card Suite [TIE] laicīguma īpašības testēšanā. Par testējamo objektu ir paņemts minētas sistēmas pamata datu plūsmas apstrādes scenārijs, kas iekļauj vairāku laika ierobežojumu veidu nosacījumus. Balstoties uz iegūtiem aprobācijas rezultātiem, autors analīze metodes priekšrocības un ierobežojumus.

4.1 Projekta apraksts

Piedāvātā metode tika pielietota reālā laika maksājuma karšu sistēmas Card Suite RTPS (angl. *Real-Time Processing System*) versijas 8.0 testēšanā uzņēmumā SIA Tieto Latvia (izziņa par metodes pielietošanu ir atrodama darbā 1. pielikumā). Produkta versijas izstrāde un pārbaude tika pabeigta 2010 gada 4. ceturksnī un pašlaik tā tiek ekspluatēta pie vairākiem esošiem klientiem. Šīs versijas galvenie uzlabojumi ir vērsti uz sistēmas drošuma īpašību uzlabošanu un atbilstību maksājuma aplikāciju datu drošības standarta prasībām (angl. *Payment Application Data Security Standard, PA DSS*) [PADSS].

Šī darba autors bija aktīvi iesaistīts aprakstītājā projektā un bija atbildīgs par vairākām testēšanas aktivitātēm. Darba autors izstrādāja testēšanas plānu visām RTPS 8.0 projekta testēšanas aktivitātēm un veica visu testēšanas darbu, t.sk. funkcionālu, sistēmas, integrācijas, platformu un ātrdarbības testu uzraudzību un atsekošanu. Tajā pašā laikā nodrošināja arī specifisko gadījumu verificēšanu, izņemot lietojamības testus. Projekta ietvaros notika reālā laika sistēmas laicīguma īpašību testēšana, balstoties uz šī darba 3. nodaļā aprakstītu metodi. Tā kā modeļvadāmās programmatūras izstrādes process tikai daļēji ir ieviests uzņēmumā un ar modeļiem tiek specificēta neliela daļa no sistēmas funkcionēšanas, laicīguma īpašību testēšanas posmā darba autors izstrādāja UML secību diagrammas sistēmas komponentēm Sparx Enterprise Architect lietojuma programmā un turpmāk tās pielietoja testēšanas procesā. Izstrādātas UML secību diagrammas specificē laicīguma ierobežojumus. Gadījumā, ja sistēmai būtu pieejamas šādas diagrammas, papildus diagrammu izstrāde nebūtu nepieciešama un tiktu lietotas eksistējošās UML secību diagrammas ar laicīguma ierobežojumiem. Pēc modeļu sagatavošanas notika šajā darbā piedāvātas metodes pielietošana, kuras rezultātā tika uzģenerēti testpiemēri. Ar uzģenerētiem testpiemēriem darba autors veica RTPS sistēmas laicīguma ierobežojumu testēšanu. To darbināšanai bija nepieciešams

simulēt kļūdainas situācijas ar sistēmas komponenšu izslēgšanu, Oracle tabulu bloķēšanu un komunikāciju pārtraukšanu. Darba autors pilnībā veica šos testus atbilstoši automātiski izveidotiem testpiemēriem. To darbināšanas rezultātā detalizēti verificēti sistēmas laicīguma ierobežojumi un atrasti nepareizas konfigurācijas parametri, kas tiek uzstādīti sākotnējās instalācijas laikā, tādējādi varēja tikt piegādāti kopā ar maksājuma karšu sistēmu.

4.2 Testējamās sistēmas apraksts

Metodes aprobācija laika ierobežojumu testēšanai tiek veikta uz reālā laika maksājuma karšu sistēmas Card Suite. Card Suite ir produktu kopa, kas nodrošina maksājuma karšu izdošanu (angl. *issuing*), pieņemšanu (angl. *acquiring*) ar visām populārākām metodēm (piemēram, tīmeklī, bankomātos (angl. *automated teller machine* – ATM), pārdošanas vietu terminālos (angl. *point of sale* – POS) un citās ierīcēs), autorizāciju apstrādi reālā laikā, kā arī daudzas citas atbalsta operācijas.

4.3 Reālā laika autorizācijas apstrādes scenārijs

Autorizācijas apstrādē atkarībā no sistēmas konfigurācijas un ziņojuma datiem parasti piedalās no 10 līdz 20 un vairāk atsevišķi servisi. RTPS sistēmā iekšējā struktūra atbilst komponēšu bāzētai arhitektūrai, turklāt visas infrastruktūras pārvaldībai un veseluma nodrošināšanai tiek lietots Oracle Tuxedo transakciju monitors [TUX]. Sistēmas darbībā tiek lietoti četru veidu laika ierobežojumi:

- Globālās Tuxedo transakcijas laika ierobežojums – maksimāli pieļaujamais laiks kopš transakcijas iniciēšanas kādā no servisiem līdz visu operāciju pabeigšanas, kas notiek pirms atbildes ziņojuma pārsūtīšanas kādam citam servisam. Transakciju var uzsākt viens serviss un pabeigt to var cits. Globālās transakcijas iniciēšanas servisi tiek definēti sistēmas konfigurācijā. Viena ziņojuma apstrādē var tikt iniciētas vairākas globālās transakcijas.
- Oracle sesijas laika ierobežojums – maksimāli pieļaujamais laiks, kas tiek izdalīts uz visām sesijas ietvaros iekļautām operācijām, ieskaitot visu izmaiņu saglabāšanu. Oracle sesija sākās ar globālās transakcijas iniciēšanu un beidzās ar *commit* operāciju.
- Ilguma ierobežojums uz Tuxedo servisa izsaukšanu – laika intervāls starp izsaukuma iniciēšanu un vadības nodošanu saucamajām servisam.
- Laika ierobežojums uz ārējo sistēmu izsaukumiem – maksimāli pieļaujamais laika intervāls starp pieprasījuma aizsūtīšanu uz ārējo sistēmu un atbildes saņemšanu no tās.

Darba 4.3 nodaļā parādītā diagramma apraksta finansu autorizācijas apstrādi RTPS sistēmas servisos sākot ar sistēmas komponentēm, kas atbild par komunikāciju nodrošināšanu un datu pārsūtīšanu uz ārējiem avotiem – starptautiskiem tīkliem un dažāda veida ierīcēm. Diagrammā ir attēlo visi četri minētie laika ierobežojumu veidi. Darbā prezentētas UML secību diagrammas apraksta autorizācijas ziņojuma apstrādes scenārijus būtiskākajos RTPS sistēmas servisos. Sistēmas sarežģītības dēļ, nav iespējams darbā aprakstīt pilnu apstrādes scenāriju ar visiem iespējamiem servisiem un apstrādes alternatīvām. Prezentētais modelis atbilst sistēmas darbības scenārijam un prezentē sistēma implementētus laika ierobežojums, kas ir atslēgas moments dotajā pētījumā

4.4 Transformācijas likumu definēšana

Transformācijas likumi apraksta testpiemēru ģenerēšanas principus. Manuālajā testēšanā šādi principi balstās uz plaši pielietojamām metodēm, kas kļuva par testēšanas standarta metodēm.

Aprakstītam finansu autorizācijas apstrādes procesam tiek definēti sekojošie testēšanas nosacījumi laika ierobežojumu verificēšanai:

1. pārbaudīt visu servisu *commit* operācijas darbību laika ierobežojumu pārsniegšanas un nepārsniegšanas gadījumos;
2. pārbaudīt laika ierobežojumu pārsniegšanu un nepārsniegšanu konkrētajā *local* fragmentā;
3. pārbaudīt gadījumus, kas tiek un netiek pārsniegti laika ierobežojumi *local_processing* fragmenta izsaukumiem pie nosacījuma, ka *RSW_STIP_FLAG = F*;
4. pārbaudīt laika ierobežojumus visiem izsaukumiem no konkrētā *STIP* servera;
5. pārbaudīt laika ierobežojumus visiem sinhroniem izsaukumiem;
6. pārbaudīt laika ierobežojumus visiem asinhroniem izsaukumiem.

Minētie testēšanas nosacījumi apraksta iespējamos verificēšanas aspektus, kas var tikt pielietoti kopā vai jebkādā citā kombinācijā. Pielietojot šos nosacījumus kopā, acīmredzami, ka tiks

iegūti arī dublējošie testpiemēri, jo 6. un 7. nosacījumi pārklājās ar iepriekšējiem nosacījumiem. Šāds nosacījumu klāsts tiek izvēlēts lai parādītu iespējamos un dzīvē pielietojamus verificējamus gadījumus.

Darba pielikumā 2 ir parādīti transformācijas likumi, kas atbilst iepriekš minētiem testēšanas nosacījumiem (60 transformācijas likumi atbilst sešiem nosacījumiem). Definētie transformācijas likumi paredz laika un ilguma ierobežojumu analīzi un ir fokusēti uz testpiemēru ģenerēšanu atbilstoši aprakstītām testēšanas metodēm – sadalīšana ekvivalences klasēs un robežvērtību analīze.

4.5 Transformācijas process un iegūtais modelis

Transformācijas rīks tiek darbināts ar definētiem transformācijas likumiem un ar sistēmas modeli. Transformācijas rezultātā tiek iegūti 180 testpiemēri, kas ir aprakstīti 3. pielikumā. Iepriekšējā nodaļā minētie testēšanas nosacījumi paredz dublējošo testpiemēru ģenerēšanu. Testpiemēru unikalitāti nodrošina konkrētā verificējamā gadījuma sistēmas uzvedība – komplekts no *behavior* un *sut* laukiem. Pielikumā 4 ir parādīti dublējošie uzģenerētie testpiemēri un to atkārtošanas skaits. Tajā pašā laika tabulā nav parādīts 21 unikāls testpiemērs. Neņemot vērā dublējošus testpiemērus, transformācijas laikā tika uzģenerēti 72 unikālie testpiemēri.

4.6 Metodes novērtēšana

Darbā aprakstīta metode ļauj definēt testpiemēru ģenerēšanas likumus un ar transformācijas rīku veidot testēšanas datus, nepieciešamus laika ierobežojumu verificēšanai. Tekošā rīka realizācija apzināti neparedz dublējošo testpiemēru dzēšanu vai neģenerēšanu. Tas ir darīts ar mērķi veidot visus gadījumus lai, balstoties uz tiem, varētu spriest par rīka funkcionēšanu un analizēt tā apstrādes rezultātu.

Viens no būtiskākajiem pozitīviem metodes aspektiem ir metodes automātiskā testpiemēru ģenerēšana. Tas nozīme, ka tiek samazināts cilvēciskais faktors kļūdu pieļaušanai testpiemēru ģenerēšanas ķēdē. Pateicoties iepriekš nodefinētiem transformācijas likumiem, sistēmas modelis tiek analizēts un transformēts pa tiešo uz testēšanas modeli cilvēkam neiejaucoties. Automātiska testpiemēru ģenerēšana nodrošina ātrāku testa datu iegūšanu, netērējot laiku uz manuālo testu veidošanu. UML testēšanas profila lietošana par metamodeli testēšanas datiem ļauj pielietot tos arī citās sistēmās. Atbilstība OMG standartam ļauj izvairīties no potenciāliem posmiem, kas varētu būt nepieciešami testēšanas datu pārnešanai kādā no testēšanas vadības rīkiem, kas atbalsta šo standartu.

Piedāvātai transformācijas valodai ir divas būtiskās priekšrocības. No vienas puses valoda ir vienkārša un ļauj fokusēties uz sistēmas funkcionalitāti un testēšanas artefaktiem, bet no otrās puses – piedāvā iespējas rakstīt triviālus un komplicētus transformācijas likumus. Tā, piemēram, var uzrakstīt transformācijas likumu, kurš analīzēs ne tikai vienu konkrētu ziņojumu, bet arī meklēs citus ziņojumus, kuriem avota serveris ir tekošā ziņojuma galamērķis.

Vēl viena piedāvātās metodes priekšrocība ir UML un XMI standartu pielietošana avota modeļu definēšanai, kas ļauj viegli integrēt metodi dažādās izstrādes vidēs. UML modelēšanas valoda ir atzīta visā pasaulē un tiek plaši pielietota. Savukārt, XMI standartu nodrošina lielāka daļa modelēšanas rīku un tas ļauj sagatavot XMI datni ar sistēmas modeli turpmākai transformācijai uz testēšanas modeli.

Metode ir realizēta ar vairākiem rīkiem, ievērojot komponenšu bāzēto projektēšanu. Šāda pieeja ļauj uzlabot atsevišķas komponentes pēc atbilstošās nepieciešamības. Tekošā rīku realizācija atbalsta XMI 2.1 versiju un gadījumā, kad būs nepieciešams pāriet uz jaunāko XMI versiju ieviesto izmaiņu dēļ, izmaiņas būs nepieciešamas tikai Ruby programmā, kas nodrošina datu translēšanu uz datubāzes tabulām. Šajā gadījumā, transformācijas rīka izmaiņas nav nepieciešamās.

Piedāvātai metodei ir arī daži ierobežojumi. Pirmkārt, metode neparedz sistēmas automātisko darbināšanu. Lai nodrošinātu universālo metodi un tās pielietošanu dažādām izstrādes vidēm un sistēmām, automātiskā sistēmas darbināšana apzināti netiek veidota promocijas darba ietvaros veikta pētījuma gaitā.

Vēl viens metodes ierobežojums ir transformācijas rīka implementēšana Oracle DBVS vidē, kas ir komerciāls produkts, un līdz ar to metodes pielietošana var prasīt noteiktās izmaksas Oracle licencēm. Oracle tika izvēlēta kā populārākā DBVS [GAR 2011], kas ļauj nodrošināt metodei

nepieciešamās funkcijas. Transformācijas rīka izstrādē tiek lietoti PL/SQL bloki, kas tiek uzturēti tikai Oracle vidē. Savukārt, šo trūkumu nevar attiecināt pie kritiskajiem, jo programmatūras izstrādes uzņēmumos un mācību iestādēs DBVS pielietošana neprasa papildus licences un līdz ar to neierobežo metodes pielietošanu. Tajā pašā laikā, komponenšu bāzētā projektēšana, nepieciešamības gadījumā, pieļauj transformācijas komponentes pārrakstīšanu citā tehnoloģijā.

DARBA REZULTĀTI UN SECINĀJUMI

Dotais pētījums ir vērsts uz modeļvadāmās programmatūras principu pielietošanu iegulto sistēmu nefunkcionālo īpašību testēšanai. Pētījuma rezultātā ir izstrādāta metode testpiemēru ģenerēšanai no iepriekš sagatavota sistēmas modeļa, kas ir veidots ar UML modelēšanas valodu. Piedāvātās metodes realizēšanai darba ietvaros, balstoties uz eksistējošiem OMG standartiem, autors nodefinēja rīku ķēdi un to komponenšu saskarnes realizāciju, kas nodrošina sistēmas modeļu apstrādi un testpiemēru ģenerēšanu. Lai pārbaudītu metodes pielietošanu, tā tika aprobēta laika ierobežojumu verificēšanā reālā laika maksājuma karšu sistēmai, kurai šīs nefunkcionālās īpašības ir kritiskas standarta darbību veikšanai. Izvirzītā mērķa sasniegšanai tika izpildīti šādi **uzdevumi**:

- apskatītas un izanalizētas eksistējošās testēšanas metodes, kā arī tās tika izskatītas attiecībā uz pielietošanu iegulto sistēmu testēšanai;
- izpētītas iegulto sistēmu nefunkcionālās īpašības un to eksistējošās testēšanas metodes un modelēšanas paņēmieni;
- veikta MDA principu analīze un ir izvērtētas iespējas pielietot tos testpiemēru ģenerēšanā;
- izvirzīta hipotēze par iespēju testēšanas metodi balstīt uz MDA principiem, kas nodrošinātu testpiemēru ģenerēšanu no UML modeļiem;
- veikta hipotēzes pārbaude uz reālā laika maksājuma sistēmas nefunkcionālo īpašību piemēra un ir secināts par metodes pielietošanas iespējām.

*Promocijas darba **galvenais rezultāts** ir piedāvātā testēšanas metode iegulto sistēmu nefunkcionālo īpašību testēšanai un izstrādāto rīku kopa, kas nodrošina UML modeļu transformāciju un testpiemēru ģenerēšanu. Piedāvātā metode balstās uz modeļvadāmās programmatūras izstrādes pamatiem un vispārīgiem modeļu transformācijas principiem. Izstrādātais rīks tika pielietots reālās maksājuma karšu sistēmas laicīguma īpašības testēšanā.*

Promocijas darba **svarīgākie rezultāti** ir:

1. Sistematizēta informācija par iegulto sistēmu nefunkcionālām īpašībām, kā arī ir aprakstīti analizēto īpašību modelēšanas un testēšanas paņēmieni.
2. Balstoties uz to, ka iegulto sistēmu testēšanai izmanto arī klasiskās testēšanas metodes, ir aprakstīts vispārīgais testēšanas process un tā metodes.
3. Definēti modeļvadāmās programmatūras izstrādes procesa principi, kā arī detalizēti ir aprakstīti tā artefakti.
4. Piedāvāts atbilstības modelis starp modeļvadāmās arhitektūras principiem un vispārīgo testēšanas V modeli.
5. Definēta uz modeļiem balstīta testēšanas metode, kas ir pielietojamā nefunkcionālo īpašību testēšanai.
6. Balstoties uz testēšanas specifiku, ir izstrādātā transformācijas notācija, kas ir pielietota likumu definēšanai, lai transformētu sistēmas modeļus uz testēšanas modeli.
7. Balstoties uz piedāvāto testēšanas metodi, ir izstrādāti rīki, tajā skaitā transformācijas rīks, kas nodrošina UML modeļu transformāciju uz testpiemēriem pierakstītiem UML testēšanas profilam atbilstošajā formā.
8. Piedāvātā metode un izstrādātie rīki ir aprobēti uz reālā laika maksājuma karšu sistēmas laicīguma īpašību verificēšanā.

Balstoties uz promocijas darba ietvaros veiktiem pētījumiem un iegūtiem rezultātiem, ir veikti **sekojošie secinājumi**:

1. Iegulto sistēmu funkcionālajā testēšanā pielieto klasiskās testēšanas metodes, verificējot pēc iespējas vairāk gadījumus un sasniedzot augstāku pārklājumu pirmkodam.
2. Eksistējošās iegulto sistēmu nefunkcionālo īpašību testēšanas metodes balstās uz iepriekšējo paaudžu modelēšanas notācijām un manuālo testpiemēru ģenerēšanu.

3. Neeksistē vispāratzītās modeļvadāmās testēšanas pieejas funkcionālo un nefunkcionālo īpašību verificēšanai.
4. Testpiemēri var tikt prezentēti noteiktajā notācijā un tie var tikt automātiski iegūti no sistēmas modeļiem, pielietojot modeļvadāmās programmatūras izstrādes principus.
5. Lai nodrošinātu transformācijas likumu fokusēšanos uz loģiskajām darbībām un izvairīties no komplicētās struktūras aprakstiem, testēšanas metodē ir paredzēta sākotnējā sistēmas modeļa vienkāršota prezentācija, kas tiek apstrādāta ar transformācijas likumiem.
6. Balstoties uz to, ka piedāvātā metode tika veiksmīgi pielietota reālās sistēmas laicīguma īpašības verificēšanai, metode var tikt pielietota arī pārējām nefunkcionālām īpašībām.

Turpmākais virziens jaunajam pētījumam var būt saistīts ar metodes attīstīšanu lai nodrošinātu pilnīgi automatizētu sistēmas testēšanu, ieskaitot testpiemēru ģenerēšanu, to darbināšanu un rezultāta pārbaudi.

Izstrādāto metodi un rīkus ir ieteikts pielietot laicīguma, sinhronizācijas un asinhronās darbības īpašību testēšanai. Metode var tikt pielietota ne tikai iegultām sistēmām, bet arī citu veidu sistēmām, kurās ir implementētas minētās īpašības. Metode var tikt viegli integrēta izstrādes procesā, kas jau paredz sistēmas darbības specificēšanu ar UML modeļiem, jo tā neparedz papildus soļus sākotnējo datu sagatavošanai un apstrādā UML modeļus standartā XMI formātā.

Izstrādāto metodi un rīkus autors iesaka pielietot arī citu nefunkcionālu īpašību testēšanā, jo rīku implementēšanā ir pielietots komponenšu bāzēts projektējums, kas nodrošina to neatkarību savā starpā un nepieciešamības gadījumā tie var tikt elastīgi uzlaboti. Piedāvāto testēšanas metodi, kas balstās uz vienkāršotu sistēmas modeļa prezentāciju, darba autors iesaka pielietot arī funkcionālo īpašību testēšanā.

IZMANTOTĀ LITERATŪRA

Autora publikācijas starptautiski recenzējamos zinātniskajos krājumos:

[GRI 2005] Grigorjevs J., Nikiforova O. Testing Process Adjustment for Real Time Systems // In Proceedings of the 46th Scientific Conference of Riga Technical University, 5th Series, Vol. 22, Computer Science, Applied Computer Systems, Riga, Latvia: RTU Publishing. – 2005. – pp. 229-241

[GRI 2006] Grigorjevs J., Nikiforova O. Unit Testing for Real Time Systems // Scientific Journal of Riga Technical University, Computer Science, Applied Computer Systems, 5th series, Vol. 26, Riga, Latvia: RTU Publishing. – 2006. – pp. 67-77

[GRI 2007] Grigorjevs J., Nikiforova O. Features of embedded systems that require specific testing approaches // In Proceedings of the 48th Scientific Conference of Riga Technical University, 5th Series, Vol. 30, Computer Science, Applied Computer Systems, Vol. 26, Riga, Latvia. – 2007. – pp. 47-56

[GRI 2008a] Grigorjevs J., Nikiforova O. Modeling of Non-Functional Requirements of Embedded Systems // In Scientific Proceedings of 42nd Spring International Conference MOSIS2008, Ostrava, Czech Republic. – 2008. – pp. 13-20

[GRI 2008b] Grigorjevs J., Nikiforova O. Compliance of Popular Modeling Notations to Non-functional Requirements of Embedded Systems // In Proceedings of the International Scientific Conference Informatics in the Scientific Knowledge 2008, Varna, Bulgaria. – 2008. – pp. 139-149

[GRI 2008c] Grigorjevs J. Testing of Embedded System's Non-functional Requirements // In Scientific Proceedings of the 8th International Baltic Conference Baltic DB&IS 2008, Tallinn, Estonia. – 2008.

[GRI 2009] Grigorjevs J., Nikiforova O. Several Outlines on Model-Driven Approach for Testing of Embedded Systems // Scientific Journal of RTU, 5th series, Computer Science, 38. vol. – 2009. – pp. 96-107

[GRI 2011a] Grigorjevs J. Model-Driven Testing Approach for Embedded Systems Specifics Verification Based on UML Model Transformation // In Proceedings of 3rd International Workshop

on Model-Driven Architecture and Modeling-Driven Software Development, Beijing, China. – 2011. – pp. 26-35

[GRI 2011b] Grigorjevs J. Model-Driven Testing Approach Based on UML Sequence Diagram // In Scientific Journal of Riga Technical University, 5th series, Computer Science, Applied Computer Systems, Vol. 47, Riga, Latvia – 2011 – pp. 85-90

[NIK 2008] Nikiforova O., Pavlova N., Grigorjevs J. „Several Facilities of Class Diagram Generation from Two-Hemisphere Model in the Framework of MDA". 23rd International Symposium on Computer and Information Sciences, ISCIS 2008 : Proceedings. - Piscataway, NJ : IEEE, 2008. 6 p.

Citi kopsavilkumā izmantotie avoti (visi tīmekļa avoti ir pārbaudīti 19. martā 2012. gadā):

[ACK 2008] Ackermann C., Cleaveland R., Ray A., Shelton C., Martin C. Integrated Functional and Non-Functional Design Verification for Embedded Software Systems // In proceedings of SAE World Congress & Exhibition, Paper Number 09AE-0202, USA – 2008. / Internets: http://www.cs.umd.edu/Grad/scholarlypapers/papers/Ackermann.pdf

[ACRO 2011] Aircraft Crashes Record Office. Statistics – 2011. / Internets: http://www.baaa-acro.com/Statistiques%20diverses.htm

[AGR 2001] Agrawal S., Bhatt P. Real-time Embedded Software Systems – 2001. / Internets: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.202.2819&rep=rep1&type=pdf

[BBC 2009] BBC. Yemen jet crashes in Indian Ocean – 2009. / Internets: http://news.bbc.co.uk/2/hi/8125664.stm

[BEA 2011] Bureau d'Enquêtes et d'Analyses. Interim report on the accident on 1st June 2009 – 2011. / Internets: http://www.bea.aero/docspa/2009/f-cp090601e1.en/pdf/f-cp090601e1.en.pdf

[BRO 2003] Broekman B., Notenboom E. Testing embedded software. Pearson Education – 2003. – 348 pages.

[COP 2004] Copeland L. A Practitioner's Guide to Software Test Design. Artech House Publishers, London, England. – 2004. – 294 p.

[DSQL] Oracle. Coding Dynamic SQL Statements. / Internets: http://download.oracle.com/docs/cd/B10500_01/appdev.920/a96590/adg09dyn.htm

[EA] Sparx Systems. Enterprise Architect./ Internets: http://www.sparxsystems.com/products/ea/index.html

[EMF] Eclipse Foundation. Eclipse Modeling Framework Project (EMF). / Internets: http://www.eclipse.org/modeling/emf/

[ENG 2006] Engels G., Guldali B., Lohmann M. Towards Model-Driven Unit Testing // In Proceedings of the 9th International Conference on Models in Software Engineering MoDELS'06, Genova, Italy. – 2006. – pp. 182-192. / Internets: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.62.200

[GAR 2011] Gartner. Magic Quadrant for Data Warehouse Database Management Systems. / Internets: http://www.sybase.com/files/White_Papers/Gartner_MagicQuad_forDataWarehouseDMS.pdf

[JEN 2011] Jensen E. D. Real-Time Overview – 2011. / Internets: http://www.real-time.org/realtimeoverview.htm

[JOU 2006] Jouault F., Kurtev I. Transforming Models with ATL // In Proceedings of the MoDELS'06 Conference, Montego Bay, Jamaica. – 2006. – pp. 128-138. / Internets: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.99.6117

[KLE 2003] Kleppe A., Warmer J., Bast W. MDA Explained: The Model Driven Architecture: Practice and Promise. Addison Wesley – 2003. – 167 p.

[KRS 2002] Krstic A., Lai W.-C., Chen L. u.c. Embedded Software-Based Self-Testing For SoC Design // 39th Design Automation Conference, DAC 2002, New Orleans, USA. – 2002. – pp. 355-360. / Internets: http://esdat.ucsd.edu/~lichen/esdat/paper/dac02_krstic.pdf

[KUG 2008] Kugele S., Haberl W., Tautschnig M., Wechs M. Optimizing Automatic Deployment Using Non-Functional Requirement Annotations // In Proceedings of the 3rd International Symphosium ISOLA 2008, Communications in Computer and Information Science, Leveraging Applications of Formal Methods, Verification and Validation, Margaria T., Steffen B. (Eds.), Springer, Greece – 2008. – pp. 400-414

[LEY 2010] Leyden J. Trojan-ridden warning system implicated in Spanair crash – 2010. / Internets: http://www.theregister.co.uk/2010/08/20/spanair_malware/

[MAR 2003] Marschall, F., Braun, P. Model Transformations for the MDA with BOTL // In Proceedings of the Workshop on Model Driven Architecture: Foundations and Applications, Enschede, The Netherlands. – 2003. – pp. 25-36. / Internets: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.7991

[MAT 1995] Mattai J. Real-Time Systems: Specification, Verification, and Analysis. Prentice Hall, PTR Upper Saddle River, New Jersey, USA. – 1995. – 278 p.

[MDA] Object Management Group (OMG). Model Driven Architecture. / Internets: http://www.omg.org/mda

[MEL 2004] Mellor S. J., Scott K., Uhl A. u.c. MDA Distilled: Principles of Model-Driven Architecture, Addison Wesley Professional – 2004. – 176 p.

[MYE 2004] Myers G. J. The art of software testing, Second Edition. New Jersey: John Wiley & Sons, Inc. – 2004. – 255 pages.

[NIK 2007] Ņikiforova O., Ņikuļšins V., Buzdins D. u.c. Modeļvadamas programmatūras izstrādes pamati (II daļa), Projektā Studiju moduļa izstrāde modeļvadāmai programmatūras attīstības tehnoloģijai datorsistēmas programmā ietvaros izstrādāts mācību metodiskais materiāls mācību kursam „Programmatūras attīstības tehnoloģijas" datorzinātnes un informācijas tehnoloģijas akadēmiskās bakalaurantūras 3. kursa studentiem , RTU – 2007. – 24 lpp.

[OMG] Object Management Group (OMG). About OMG®. / Internets: http://www.omg.org/gettingstarted/gettingstartedindex.htm

[PADSS] PCI Security Standards Council. Documents Library. PCI Standards Documents. / Internets: https://www.pcisecuritystandards.org/security_standards/documents.php?association=PA-DSS

[PER 2010] Peraldi-Frati M., Mallet F., Deantoni J. MARTE for time modeling and verification of real-time embedded system // In proceedings of 1st Inter. Colloque. RUNSUD, Sophia Antipolis, France – 2010. – pp. 480-489, Internets: http://www.i3s.unice.fr/~map/map_fichiers_publi/RUNSUD2010.pdf

[PIL 2005] Pilone D., Pitman N. UML 2.0 in a Nutshell. First Edition. O'Reilly Media Inc., Sebastopol, USA. – 2005. – 240 p.

[SPI 2007] Spillner A., Linz T., Schaefer H. Software Testing Foundations: A Study Guide for the Certified Tester Exam, 2nd Edition, Rocky Nook. – 2007. – 288 p.

[STH 2001] Sthamer H., Baresel A., Wegener J. Evolutionary Testing of Embedded Systems // 14th International Internet & Software Quality Week, San Francisco, USA. – 2001. / Internets: http://citeseerx.ist.psu.edu/viewdoc/download? doi=10.1.1.86.1798&rep=rep1&type=pdf

[STI 2008] Stiles G. S., Rice D. D., Doupnik J. R. Design, Verification, and Testing of Synchronization and Communication Protocols with Java – 2008. / Internets: http://www.neng.usu.edu/ece/research/rtpc/projects/JavaCSP/Synch_Comms.pdf

[SVO 2011] Свободная Пресса. Неудачные запуски спутников в 2010 году – 2011. / Internets: http://svpressa.ru/world/photo/36421/

[TIE] Tieto. Card Suite. / Internets: http://www.tieto.com/industries/financial-services/transaction-banking/card-suite

[TUX] Oracle. Tuxedo. / Internets: http://www.oracle.com/technetwork/middleware/tuxedo/overview/index.html

[UML] Object Management Group (OMG). Unified Modeling Language™ (UML®). / Internets: http://www.omg.org/spec/UML/

[UTP] Object Management Group (OMG). UML Testing Profile. / Internets: http://www.omg.org/utp

[VWREC] Volkswagen. Recall Compaigns./ Internets: http://www.volkswagen.co.nz/nz/en/service_and_parts/maintenance_care/mcare_recalls.html

[WU 2007] Wu X., Li J.J., Weiss D.M., Lee Y. Coverage-Based Testing on Embedded Systems // In Proceedings of AST, Minneapolis, USA. – 2007. – pp. 31-36. / Internets: http://arnetminer.org/viewpub.do?pid=2798644

[XMI] Object Management Group (OMG). XML Metadata Interchange (XMI®). / Internets: http://www.omg.org/spec/XMI/

[ZAN 2009] Zander-Nowicka J. Model-based Testing of Real-Time Embedded Systems in the Automotive Domain, doctoral thesis at Technical University Berlin – 2009. / Internets: http://opus.kobv.de/tuberlin/volltexte/2009/2186/pdf/zandernowicka_justyna.pdf

[ZHI 1999] Zhiming L., Mathai J. Specification and Verification of Fault-tolerance, Timing and Scheduling – 1999. / Internets: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.54.2264

[ZIM 2009] Zimmer U. R. Real-Time and Embedded Systems – 2009. / Internets: http://cs.anu.edu.au/student/comp4330/Level-0/Contents.html