LATVIJAS UNIVERSITĀTE

VALDIS VĪTOLIŅŠ

BIZNESA PROCESU MODELĒŠANA, IZMANTOJOT METAMODELĒŠANAS PAŅĒMIENUS

Promocijas darba kopsavilkums
datorzinātņu doktora (Dr. sc. comp.) zinātniskā grāda iegūšanai

Nozare: datorzinātnes
Apakšnozare: programmēšanas valodas un sistēmas

Zinātniskais vadītājs:
profesors, Dr. habil. sc. comp.
AUDRIS KALNIŅŠ

R ī g a - 2007

Darbs ir izstrādāts ar Eiropas Sociālā fonda atbalstu. Projekts „Doktorantu un jauno zinātnieku pētniecības darba atbalsts Latvijas Universitātē”

Darba vadītājs:

Profesors, Dr. habil. sc. comp. Audris Kalniņš
Latvijas Universitāte

Oponenti:

Profesors, Dr. habil. sc. comp. Jānis Bārzdiņš
Latvijas Universitāte

Asoc. Profesore, Dr. sc. ing. Mārīte Kirikova
Rīgas Tehniskā universitāte

Profesors, Ph. D. Olegas Vasilecas
Viļņas Gedimina Tehnskā Universitāte

Darba aizstāvēšana notiks Latvijas Universitātes Datorzinātnes nozares promocijas padomes atklātā sēdē 11.09.2007 LU Matemātikas un informātikas institūtā (Raiņa bulvārī 29), 413. auditorijā.

Ar darbu un tā kopsavilkumu var iepazīties LU bibliotēkā (Kalpaka bulvārī 4).

Padomes priekšsēdētājs Jānis Bārzdiņš

Saturs

Promocijas darba tēmas aktualitāte un iegūtie rezultāti

Tēmas aktualitāte. Darba tēma ir saistīta ar biznesa procesu vadības sistēmu izstrādi. Biznesa procesu vadības sistēmas ir attīstījušās no darba plūsmas pārvaldības sistēmām, tajās pakāpeniski iekļaujot arvien vairāk standarta informācijas sistēmu funkciju, piemēram, resursu un klientu pārvaldību.

Lai pārvaldītu biznesa procesu vadības sistēmu sarežģītību un kāpinātu izstrādes efektivitāti, ir nepieciešams pielietot uz modeļiem balstītu sistēmu izstrādi. Atšķirībā no klasiskām informācijas sistēmām, biznesa vadības sistēmu darbību augstā mērā nosaka to izpildītie procesu modeļi. Līdz ar to arī biznesa procesu modeļus var izmantot biznesa vadības sistēmu projektēšanā un izstrādē, ar secīgām modeļu transformācijām tos pielāgojot konkrētai izpildes platformai.

Biznesa vadības sistēmu funkciju paplašināšanās un nepieciešamība izmantot biznesa procesu modeļus programmatūras izstrādē ir izvirzījušas jaunas prasības gan biznesa modeļiem, gan procesu modelēšanas valodām. Šobrīd nav vienas labākās biznesa procesu modelēšanas valodas, tāpēc dažādus biznesa procesu vadības aspektus noteiktos izstrādes posmos apraksta ar dažādām modelēšanas valodām.

Piemēram, programmatūras izstrādes *de-facto* standartam – UML valodai darbību semantika nav pietiekami detalizēta, lai to lietotu darba plūsmas aprakstam, savukārt biznesa procesu modelēšanas valoda – BPMN nav pietiekami precīza programmatūras izstrādes vajadzībām. Esošo valodu procesu izpildes semantika nav pietiekami precīzi definēta procesu izpildei sadalītās biznesa vadības sistēmās. Tas apgrūtina modeļu izmantošanu biznesa procesu vadības sistēmu izstrādē, un šobrīd modeļu bāzēta izstrāde ir pielietojama tikai specifiskās biznesa vadības jomās un atsevišķos izstrādes posmos.

Tāpēc ir nepieciešams izstrādāt paņēmienu, kas ļautu aprakstīt dažādus biznesa procesu vadības apgabalus dažādās modelēšanas valodās, kā arī nodrošināt pāreju no vienas valodas citā, gan virzoties no biznesa modeļa uz konkrētu izpildes platformu, gan pārejot uz alternatīvu valodu. Šāds paņēmiens ļautu izveidot vienotu rīku platformu, kurā iespējams aptvert visus nepieciešamos biznesa procesu vadības sistēmas darbības aspektus un efektīvi izmantot dažādos sistēmas izstrādes posmos iegūto informāciju.

Šajā darbā biznesa procesu modelēšanas problēmas tiek piedāvāts risināt, izmantojot metamodelēšanas paņēmienus. Metamodelēšana ļauj dažādus biznesa modelēšanas aspektus aplūkot vienotā un vispārīgā formā, tai pat laikā nezaudējot precīzu jēdzienu nozīmi. Izmantojot metamodelēšanas principus, darbā ir analizētas dažādas modelēšanas valodas, precīzi salīdzinot to jēdzienus, kā arī izstrādāti modeļu izpildes un izpildes laika mērījumu paņēmieni. Izmantojot izstrādātos paņēmienus, darbā ir izstrādāts ietvars ar dažādu valodu redaktoriem un implementētas modeļu transformācijas, ar kurām konkrētus biznesa modeļus var transformēt konkrētā izpildes platformā.

Darba galvenie rezultāti.

- Izstrādāts "notācijas neatkarīgais" biznesa procesu metamodelis, kas parāda biznesa modelēšanas jēdzienus un to saistību. Šis metamodelis visos turpmākajos pētījumos ir izmantots kā dažādu biznesa modelēšanas valodu "kanoniskā forma".
- Izstrādāta biznesa modeļu jēdzienu kartēšanas metode jēdzienu kartēšanai no viena domēna uz vairākām modelēšanas valodām, kas izmanto līdzīgus jēdzienus (uz "semantiski līdzīgām" valodām). Šis paņēmiens ir izmantots par pamatu modeļu transformāciju būvei.

❑Izstrādāta precīza UML (*Unified Modeling Language*) aktivitāšu diagrammas (AD) izpildes semantika, izmantojot virtuālo mašīnu. Izstrādāti uz metamodeli bāzēti paņēmieni modeļu raksturlielumu mērījumu definēšanai un mērījumu apstrādei modeļa izpildes laikā. Šāda mašīna der par pamatu procesu imitācijas vai biznesa procesu vadības sistēmas būvei.
❑Izstrādāti funkcionāli ekvivalenti populārāko biznesa modelēšanas valodu – UML AD apakškopas profila un BPMN (*Business Process Modeling Notation*) apakškopu metamodeļi, un uz to bāzes izveidoti šo valodu redaktori. Izstrādātas transformācijas valodā MOLA (*MOdeling LAnguage*), kas veic modeļu transformāciju no AD uz BPMN. Šāda transformācija ir viens no biznesa vadības sistēmu modeļu bāzētas izstrādes soļiem. Ar līdzīgu paņēmienu modeļus tālāk transformējot uz BPEL (*Business Process Execution Language*) valodu, tos var izpildīt reālās biznesa vadības sistēmās.

Promocijas darba vispārējais raksturojums

Promocijas darbs "**Biznesa procesu modelēšana, izmantojot metamodelēšanas paņēmienus**" ir izstrādāts no 2002. gada līdz 2007. gadam Latvijas Universitātes Fizikas un matemātikas fakultātē un Matemātikas un informātikas institūtā (MII) profesora Audra Kalniņa vadībā. Šis darbs turpina MII biznesa modelēšanas tradīcijas, kas aizsāktas jau 1986. gadā.
Promocijas darba pētījumu galvenie rezultāti ir atspoguļoti četrās publikācijās [1-4] un referēti četrās starptautiskās konferencēs. **Promocijas darbs veidots kā** šo četru **publikāciju krājums**, apkopojot autora pētījumu rezultātus dažādos biznesa procesu modelēšanas aspektos.

Pētījuma priekšmets – biznesa modeļi, dažādu procesu modelēšanas metodiku un modeļu salīdzināšana, modeļu izpildes analīze, izmantojot metamodelēšanas pieeju.

Pētījuma mērķis – izstrādāt vienotu un precīzu pieeju dažādu biznesa procesu aspektu modelēšanai, procesu izpildei, mērīšanai un pārveidošanai, izmantojot metamodelēšanas paņēmienus.

Pētījuma posmi – promocijas darba pētījumi ir veikti vairākos posmos, sākot ar vispārīgu procesu modelēšanas aspektu analīzi, turpinot ar precīzu valodu jēdzienu semantikas un izpildes analīzi. Tālāk, izmantojot valodu jēdzienu precīzo semantiku, ir izstrādātas transformācijas, ar kurām biznesa procesa modeļus var pārveidot uz biznesa vadības sistēmās izpildāmu valodu. Kopsavilkuma tālākajās nodaļās ir īsi izklāstītas pētījumu galvenās problēmu nostādnes un iegūtie rezultāti, kas publikācijās ir aprakstīti detalizēti.

- Pirmajā nodaļā "**Biznesa modelēšana**" tiek piedāvāts vispārīgs biznesa procesu metamodelis, kas tiek izmantots jēdzienu kartēšanai valodās ar līdzīgiem jēdzieniem. Darbā tiek veikta esošo biznesa procesu modelēšanas valodu un biznesa modelēšanas ietvaru analīze. Izpētē ir ņemta vērā gan biznesa procesa darbināšanai nepieciešamā informācija, gan biznesa procesu apkārtne, kas nepieciešama biznesa procesa skaidrošanai un mērījumiem. Esošo metodiku biznesa procesu jēdzieni un to savstarpējā saistība ir analizēti, kā jaunu paņēmienu izmantojot metamodelēšanu. Izmantojot biznesa procesu metamodeli, tiek piedāvāta metode, kurā biznesa modelēšanas valodas, kas izmanto līdzīgus jēdzienus, attēlo kā "notācijas neatkarīgā" metamodeļa skatus. Izstrādātais vispārīgais biznesa procesu metamodelis visos turpmākajos pētījumos ir izmantots kā biznesa procesu definīcijas "kanoniskā forma".
- Otrajā nodaļā "**Biznesa procesu mēri**" tiek piedāvāts jauns biznesa procesu mērījumu definēšanas paņēmiens un ir izstrādāts jauns, uz metamodeli balstīts, biznesa procesu mērījumu ietvars. Izstrādātais paņēmiens ļauj biznesa procesu mērus definēt integrēti kopā ar biznesa procesa definīciju un nosaka procesa elementu iespējamos mērus. Ir dots īss esošo biznesa procesu mērīšanas paņēmienu apraksts, un ir secināts, ka esošie mērīšanas paņēmieni nav universāli un ka tie ļauj lietotājam definēt mērus bez praktiskas nozīmes. Tālāk tiek piedāvāti paņēmieni gan biznesa procesa elementu mērījumu definēšanai, gan iespējamo mēru ierobežojumiem, gan arī šo mērījumu automatizētai apstrādei (agregēšanai) procesa izpildes laikā. Mērījumu definēšanai ir izstrādāts UML AD profils, mērāmo lielumu apstrādes principi ir nodrošināti, papildinot UML metametamodeli.

- Trešajā nodaļā "**UML 2.0 aktivitāšu diagrammas biznesa modelēšanas semantika kā virtuālā mašīna**" tiek piedāvāta UML standarta aktivitāšu diagrammas izpildes semantikas analīze, kā jaunu paņēmienu izmantojot virtuālo mašīnu. Tiek noteikti biznesa procesu definēšanai nepieciešamie AD elementi un ir precizēta diagrammu izpilde biznesa procesos. Ir izstrādāts jauns, vienkāršāks par oriģinālo, AD izpildes algoritms, kas tomēr nezaudē oriģinālo aktivitāšu diagrammas izpildes nozīmi. Tas ir panākts, izmantojot "grūdošos" un "velkošos" dzinējus, kas pārvieto marķierus pa aktivitāšu izpildes plūsmas ceļiem. Izstrādātais algoritms ir izmantojams aktivitāšu diagrammas imitācijas, kā arī biznesa procesu vadības sistēmas dzinēja izstrādei.
- Ceturtajā nodaļā "**UML un modeļu transformācijas biznesa procesu definēšanā**" tiek piedāvāta uz modeļu transformācijām balstīta biznesa vadības sistēmu izstrādes tehnika. Ir aplūkotas divas pasaulē populārākās biznesa modelēšanas valodas – UML aktivitāšu diagramma un biznesa procesu modelēšanas notācija BPMN (*Business Process Modeling Notation*). Apskatot esošās biznesa vadības sistēmas, ir noteikti svarīgākie biznesa procesu modelēšanas aspekti. Uz to bāzes ir izstrādāti UML AD apakškopas profils un funkcionāli ekvivalenta BPMN apakškopa, kā arī izstrādāti šo valodu redaktori. Precīza valodu izpildes semantika ir noteikta, izmantojot jēdzienu pārveidi uz izpildāmu biznesa procesu valodu BPEL. Salīdzinot AD un BPMN valodu metamodeļus, ir pierādīts, ka modeļu pārveide no vienas valodas otrā nav triviāla un ka modeļus visefektīvāk var salīdzināt (un pārveidot), izmantojot modeļu transformācijas. Modeļu transformācijas ir izstrādātas MOLA transformāciju valodā un realizētas ar MOLA transformāciju rīka palīdzību.
- Nobeigumā ir dots pētījumu rezultātu novērtējums jaunāko notikumu kontekstā, kā arī ir aprakstīta ideju praktiskā realizācija modelēšanas rīkos.

Pētījumu teorētiskā un praktiskā nozīme.

Darbā ir izstrādāti teorētiskie principi, kā, izmantojot procesu modelēšanas valodu metamodeļus, šīs valodas var aprakstīt un salīdzināt un kā šo valodu modeļus var pārveidot un izpildīt. Izstrādātais biznesa procesu metamodelis un modeļu kartēšanas paņēmiens ir izmantojami precīzai "semantiski līdzīgu" modelēšanas valodu modeļu pārveidei, bet modeļu transformācijas ļauj iegūt modeļa ekvivalentu arī ļoti sarežģītu atbilstību gadījumā. Izstrādātais uz metamodeli balstītās virtuālās mašīnas modeļu izpildes paņēmiens ir izmantojams dažādu procesu modelēšanas valodu imitācijas vai izpildes sistēmas izstrādei, ļaujot veikt procesu raksturlielumu mērīšanu izpildes laikā. Darbā izstrādātie metamodelēšanas paņēmieni un atziņas ir izmantoti modelēšanas rīku ietvara un uz tā balstītu modelēšanas redaktoru izstrādē, apliecinot metamodelēšanas paņēmienu efektivitāti.

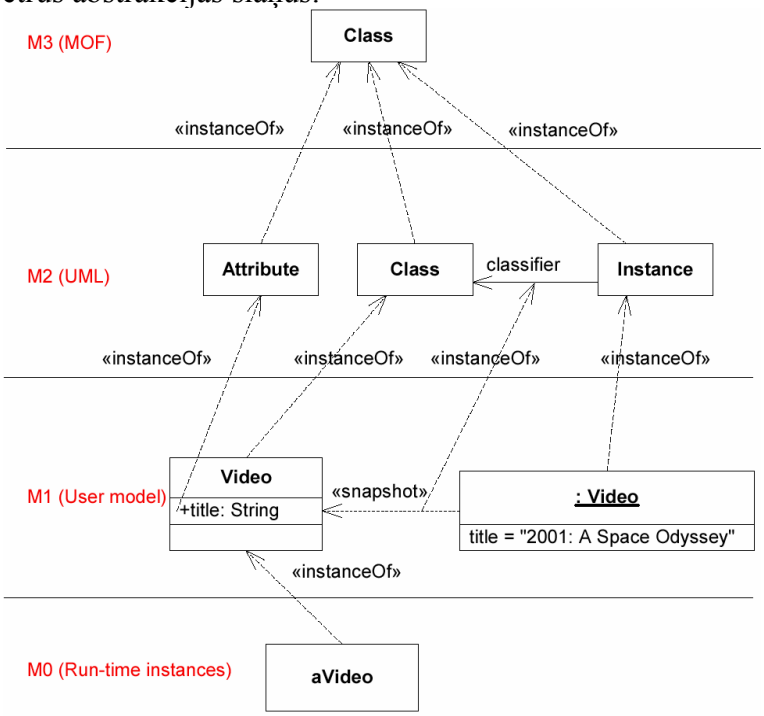
Darba pētījumu rezultāti izmantoti GMF (*Generic Modeling Framework*, [7.3]) rīka biznesa modeļu redaktoru izstrādē. Izmantojot GMF rīku, ir izveidoti jauni UML AD un BPMN modelēšanas valodu redaktori. Šajos redaktoros izveidotos modeļus var pārveidot no vienas valodas uz otru ar MOLA transformācijas rīka palīdzību.

1 Modeļi, metamodeļi un metamodelēšana

Modelis ir kāda reāla pasaules priekšmeta vai parādības abstrakts atspoguļojums. Tas var būt veidots ar dažādiem paņēmieniem, kā formula, zīmējums, u.tml. Daudzas reālās pasaules lietas un parādības var modelēt diskrēti – norobežojot atsevišķus objektus, starp kuriem parāda to mijiedarbību. Parasti šādi modeļi tiek attēloti grafiski, objektus attēlojot ar kastītēm, bet to mijiedarbību – ar līnijām starp kastītēm. Grafisks attēlojums modeļiem nav obligāts, bet ilgstoša diskrētās modelēšanas prakse ir pierādījusi, ka grafiska prezentācija šādiem modeļiem ir uzskatāmāka un saprotamāka par tekstuālu prezentāciju.

Viens no diskrēto modeļu izveides paņēmieniem ir metamodelēšanas paņēmiens, kurā diskrētos modeļus izstrādā saskaņā ar precīzi definētu likumu kopu jeb metamodeli, kas pats par sevi arī ir diskrēts modelis. Modeli aprakstot ar metamodeli, pētāmā lieta vai parādība tiek aplūkota augstākā abstrakcijas pakāpē jeb slānī. Katrs modelis, kas ir veidots saskaņā ar metamodeli, ir konkrēts metamodeļa pielietojums jeb metamodeļa instance.

Vispārīgā gadījumā modeļa metamodeļa izstrādi, vai pretēji – metamodeļa instances radīšanu var veikt dažādi. Šī darba pētījumos tiek izmantots plašāk izmantotais OMG grupas izstrādātais metamodelēšanas standarts MOF (*Meta-Object Facility*) [7.3], kurš nosaka četrus abstrakcijas slāņus:

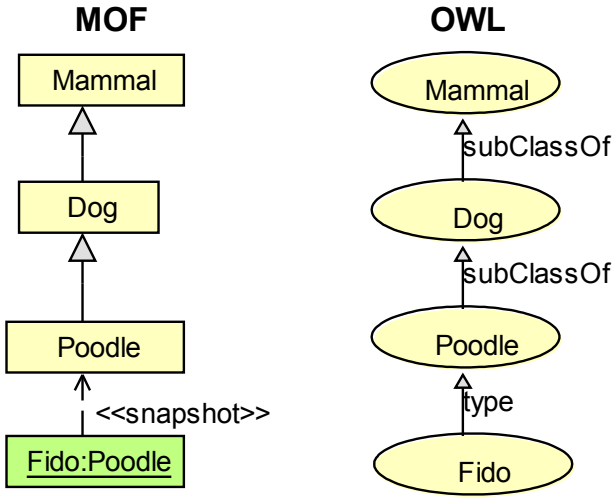


Att. 1 OMG slāņu hierarhijas piemērs

Šajā piemērā saskaņā ar MOF ir aprakstīta video kasete ar filmu "2001: A Space Odissey". Modelī (M1 slānī) visas videokasetes apraksta klase *Video*, kurai ir atribūts *title*. Faktu, ka pastāv konkrēta videokasete, modelī apraksta ar instanci, kas ir klases Video "momentuzņēmums" un šai konkrētajai instancei atribūtam *title* ir konkrēta vērtība "2001: A Space Odissey". Formāli šāds modelis lasāms tā, ka ir *Video* kolekcija, kurā vienmēr ir "2001: A Space Odissey" kasete. Programmēšanā modeļu instances ir statiskie mainīgie jeb konstantes.

Konkrētais lietotāja modelis, kas apraksta videokasetes, ir noteikta metamodeļa (M2 slānis) instance. Klase *Video* ir metaklases *Class* instance, atribūts *title* ir metaklases *Attribute* instance, bet instance *Video* ir metaklases *Instance* instance. Konkrētās

metamodeļa metaklases (*Class, Attribute, Instance*) ir noteikta metametamodeļa (M3 slānis) metametaklases *Class* instances.
Lasot zīmējumu no augšas, var teikt, ka M3 slānis definē, kā, veidojot M3 slāņa instances, radīt pareizus M2 slāņa modeļus, bet M2 slāņa modeļi definē, kā, veidojot M2 slāņa instances, veidot pareizus M1 slāņa modeļus. Metaklases *Class* un metametaklases *Class* nosaukumi sakrīt, jo MOF līdzīgi kā citās "pašpietiekamās" valodās (piem., BNF [7.3]) valodas sintakse ir aprakstīta ar tās pašas valodas līdzekļiem. (Precīzāk – MOF izmanto UML klašu diagrammas līdzekļu apakškopu [7.3]). Valodas atsaukšanās pašai uz sevi (rekursija) principā nodrošina bezgalīgi daudzu slāņu ieviešanu, tomēr programmatūras izstrādē šādu iespēju parasti neizmanto.
Aprakstot dabīgās valodas saskaņā ar MOF, katrs konkrēts raksts (piemēram, šis) ir reālās pasaules modelis, metamodelis apraksta valodas (piemēram, latviešu valodas), konteksta brīvo sintaksi (t.i., valodu kopīgos jēdzienus bez gramatikas), bet metametamodelis apraksta valodu abstraktās sintakses līdzekļus.
Att. 2 ir parādīts, kā pielietojot citus metamodelēšanas likumus, to pašu reālās pasaules faktu var attēlot savādāk. Zīmējumā pa kreisi ir parādīts MOF formālisms, bet pa labi – ontologu skatījums valodā OWL [7.3]:



Att. 2 UML un ontoloģiskās metamodelēšanas salīdzinājums

Pieņemsim, ka ir kāds reāls zīdītāju dzimtas suņu ģints pūdeļu sugas eksemplārs vārdā *Fido*. Saskaņā ar MOF to var attēlot kā konkrētas pūdeļu klases *Poodle* "momentuzņēmumu", kas ir suņu ģints *Dog* specializācija, kas savukārt ir zīdītāju dzimtas *Mammal* specializācija. Visus šos reālās pasaules faktus attēlo modelī jeb M1 slānī. <<*snapshot*>> asociācija uzsver to, ka *Poodle* klases instance *Fido* nav vis M0 slāņa instance (reāls objekts), bet ir tās attēlojums M1 metaslānī.
Ontologi šo pašu pasaules faktu OWL valodā attēlo, ka *Fido* tips ir *Poodle*, kas ir *Dog* un tālāk *Mammal* apakšklase. OWL *type* asociācijas analogs UML ir <<*instance of*>> nevis <<*snapshot*>> analogs, jo OWL parasti vienā diagrammā (modelī) attēlo gan klases, gan to eksemplārus (t.i., vairākus abstrakcijas slāņus). Izmantojot *type* asociāciju principā OWL var veidot patvaļīgi daudz abstrakcijas slāņus, bet parasti to ierobežo līdzīgi kā MOF līdz četriem abstrakcijas slāņiem.
Šajā darbā metamodelēšanā tiek izmantots MOF formālisms. Modeļi tātad attēlo reālās pasaules faktus, piemēram, uzņēmuma struktūru un resursus, biznesa procesus un to parametrus. Metamodeļi apraksta biznesa procesu modelēšanas valodas (biznesa modeļu sintaksi). Metametamodeļi apraksta dažādu metamodeļu vispārīgās īpašības (biznesa modelēšanas valodu sintaksi).

Darba pētījumos tiek izmantota svarīga metamodeļu īpašība – vienu un to pašu reālās pasaules modeli var uzbūvēt, veidojot dažādu metamodeļu instances [7.3]. Atkarībā no metamodeļu līdzības pakāpes var izdalīt sekojošus atbilstības veidus:

- Metamodeļi ir sintaktiski ekvivalenti, ja jebkura instanču kopa, kas atbilst vienam metamodelim, precīzi atbilst arī otram metamodelim, ieskaitot klašu, asociāciju un atribūtu nosaukumus. Sintaktiski ekvivalentiem metamodeļiem vienādām konkrētām (neabstraktām) klasēm atribūti un asociācijas ir vienādi (ieskaitot mantotos). Šo paņēmienu izmanto modeļu praktiskajā realizācijā "izmetot" abstraktās virsklases (veicot t.s. "metamodeļa plakanošanu") konkrētas metamodeļa apakškopas realizācijai, un tas ir izmantots arī šajā darbā.
- Metamodeļi ir semantiski ekvivalenti, ja jebkura instanču kopa atbilst gan vienam, gan otram metamodelim. Semantiski ekvivalentiem modeļiem var atšķirties tikai asociāciju kardinalitātes (modeļu ierobežojumi) vai arī klašu un asociāciju lomu nosaukumi. Praktiski tas nozīmē, ka šādi metamodeļi ir ar atšķirīgu modeļu ierobežojumu pakāpi un vienas un tās pašas lietas sauc dažādos vārdos.
- Ja divu metamodeļu instanču kopas neatbilst, tomēr tās vienādi precīzi apraksta to pašu reālās pasaules faktu, tad tādi metamodeļi šajā darbā tiks saukti par "semantiski līdzīgiem". Semantiski līdzīgas ir dažādas objektorientētās programmēšanas valodas, piemēram, Java un C#. Arī latviešu un angļu valoda ir "semantiski līdzīgas", jo konkrēti šo valodu teikumi neatbilst kopīgai abstraktajai sintaksei, bet tie līdzvērtīgi apraksta tos pašus reālās pasaules faktus. Tā kā apgalvojums par "semantiski līdzīgiem" metamodeļiem nav formāli pārbaudāms, to var noteikt tikai konkrētiem piemēriem.

Izmantojot metamodeļu semantisko ekvivalenci vai "līdzību", tos ir iespējams salīdzināt, aizstāt, veidot to apvienojumus, kā veidot to modeļus jeb metametamodeļus. Sintaktiski vai semantiski ekvivalentus modeļus ērtāk salīdzināt ar kartēšanas (*mapping*) paņēmienu, jo atbilstības ir pietiekami triviālas. Semantiski līdzīgu modeļu atbilstība bieži vien ir netriviāla, un tos ērtāk ir salīdzināt ar modeļu transformācijas paņēmienu.

2 Biznesa modelēšana

Biznesa procesu modelēšana (saīsināti – BPM) aizsākās kā daļa no biznesa procesu reinženierijas pagājušā gadsimta 90-tajos gados. Sākotnēji biznesa procesu modeļi bija konceptuāli, un tie tika izmantoti programmatūras apraksta vietās, kur diagramma bija uzskatāmāka par tekstu. Lai nodrošinātu viennozīmīgu modeļu izpratni, tika izstrādātas pirmās modelēšanas valodas, modeļu veidošanai tika izstrādāti atbilstoši modelēšanas rīki.

Viena no pirmajām modelēšanas valodām tika definēta t.s. Zahmana ietvarā [7.3], kas tika ieviests Popkin Software System Architect rīkā. Rīkos tika implementētas arī citas modelēšanas valodas, piem., GRAPES BM – GRADE rīkā [7.3], EPC diagrammas ARIS rīkā [7.3, 7.3], kā arī tādas valodas kā IDEF 3 [7.3] un UML 1.0 aktivitāšu grafi [7.3]. Izstrādājot precīzas modelēšanas valodas, parādījās arī modeļu validācijas un imitācijas iespējas (piem., ARIS, System Architect [7.3], GRADE), un tika izstrādātas pirmās darba plūsmas pārvaldības sistēmas (FileNet, MQ Workflow). Līdz ar to biznesa modelis vairs nebija tikai programmatūras specifikācija, bet gan izpildāma programma, un modelēšanas valodu precīza izpildes semantika kļuva par aktuālu problēmu.

2002. gadā, kad tika uzsākts šis pētījums, biznesa procesu modelēšana bija kļuvusi par plašu nozari un ar BPM sāka apzīmēt biznesa procesu pārvaldību (*business process management*), tādējādi uzsverot gan procesu analīzes, gan izpildes aspektus. Procesu modeļu izpildes nepieciešamība savukārt uzlika stingras prasības modelēšanas valodu semantikai. Daudzās procesu modelēšanas valodas un modelēšanas ietvari biznesa modelēšanu skaidroja katrs savā veidā, tāpēc integrētai uzņēmuma biznesa procesu apstrādei vienota biznesa modelēšanas pieeja bija absolūti nepieciešama.

Pētījuma mērķis bija izstrādāt vienotu biznesa procesu modelēšanas metodiku, kā jaunu paņēmienu izmantojot metamodelēšanu, ko izmantoja modelēšanas valodā UML [7.3]. Tā laika izplatītākie modelēšanas ietvari bija izstrādāti pirms UML ieviešanas (piem., Zachmana satvars, ARIS) un tāpēc nebija izstrādāti kā metamodeļi. Līdz ar to bija nepieciešams analizēt esošos ietvarus un izstrādāt vispārīgu un saskaņotu biznesa procesu metamodeli.

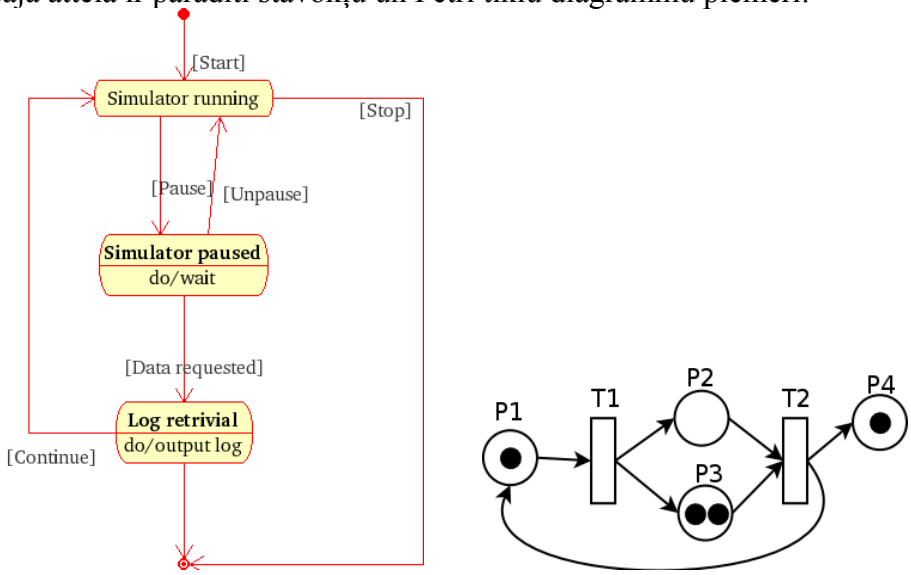
Izmantojot vienotu metamodeli, tiek piedāvāta metode, kurā valodas, kas izmanto līdzīgus jēdzienus ("jēdzieniski līdzīgas" valodas), ir iespējams attēlot kā harmonizēta metamodeļa skatus.

2.1 Biznesa procesu modeļi dažādās modelēšanas valodās

Biznesa procesu modelēšanas valodas ir veidotas uz sistēmu dinamiku aprakstošu valodu bāzes. Pasaulē ir izplatītas divas šādas valodas – stāvokļu diagramma [7.3] un Petri tīkli [7.3].

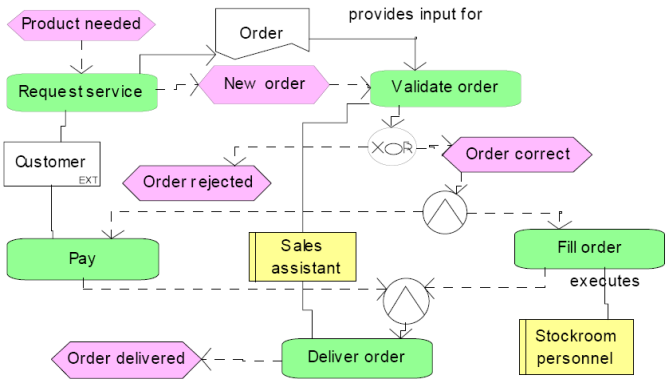
Stāvokļu diagramma tika izstrādāta uz galīgo automātu teorijas (*Finite State Machine – FSM*) bāzes 1960. gados. Tajā galīga stāvokļu automāta pārejas tiek aprakstītas ar orientētiem grafiem. Grafa virsotnes ir automāta stāvokļi, bet šķautnes norāda stāvokļu pārejas automātam reaģējot uz attiecīgiem ieejas simboliem (notikumiem), kas var izraisīt arī noteiktas izejas simbolus (darbības). Notikumi un darbības (ja tādas ir) parasti tiek atdalītas ar slīpsvītru. Vienkāršu stāvokļu automātu diagrammu ir iespējams aprakstīt algebriski (ar regulāru valodu). 1987. gadā Deivids Harels (*David Harel*) to papildināja ar jaunām iespējām sarežģītu gadījumu aprakstīšanai, ieviešot papildus pseidostāvokļus (piemēram, izvēles un saliktos stāvokļus, paralēlismu) un pāreju nosacījumus (*guard conditions*). Diagramma ar pāreju nosacījumiem vairs nav algebriski aprakstāma, bet tā ir funkcionāli pilna dinamiskas sistēmas programmēšanas valoda, kas tika ieviesta vienā no pirmajiem

grafiskās modelēšanas rīkiem Statemate [7.3]. Harela diagramma tika izmantota UML 1.x aktivitāšu diagrammas darbības precīzai aprakstīšanai.
Petri tīklus 1962. gadā izstrādāja Karls Petri (*Carl Adam Petri*). Petri tīkli ar orientētiem grafiem apraksta sadalītu sistēmu darbību, kas apmainās ar datiem. Petri tīkli sastāv no virsotnēm – pozīcijas, kur marķieri var atrasties, un pārejām, kurām "atveroties" (izpildoties nosacījumam vai notikumam), kurām marķieris drīkst iziet cauri. Virsotnes tiek savienotas ar šķautnēm, kas savieno tikai pozīcijas ar pārejām un otrādi. Pārejām "atveroties" marķieri maina savu atrašanās vietu uz nākamo pozīciju. Vienā Petri tīklu mašīnā vienlaicīgi var atrasties vairāki marķieri. Vienkrāsainus Petri tīklus (t.i., marķieri attēlo tikai vadības plūsmu) ar galīgu marķieru skaitu principā var aprakstīt ar sadarbojošos automātu kopu, tātad var aprakstīt ar regulāru valodu. Vairumā gadījumu to nevar izdarīt praktiski, jo kopējais stāvokļu skaits pieaug eksponenciāli no marķieru skaita.
Paplašinātajos Petri tīklos (t.s. krāsainie Petri tīkli) marķieri var būt dažādi (tie var saturēt dažāda tipa datu laukus). Ar dažādu "krāsu" marķieriem var modelēt dažādu datu plūsmu.
Sekojošajā attēlā ir parādīti stāvokļu un Petri tīklu diagrammu piemēri:



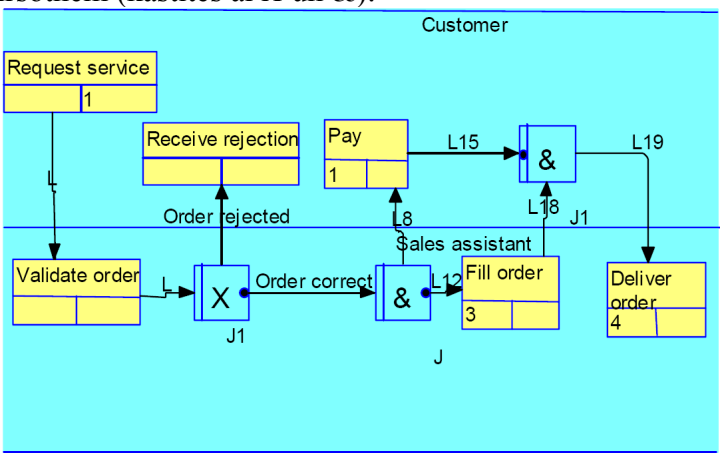
Att. 3 Stāvokļu diagrammas un Petri tīkla piemērs

Procesu modelēšanas valodas tika izstrādātas uz stāvokļa diagrammas un Petri tīklu bāzes, jo šīm valodām ir precīza matemātiskā semantika, un tās var izmantot citu valodu semantikas aprakstīšanai. Tomēr šo notāciju izmantošana biznesa procesu aprakstīšanai tiešā veidā ir praktiski neērta. Tāpēc biznesa modelēšanas valodās tika ieviesti papildus līdzekļi, lai aprakstītu kādas darbības, kādos apstākļos un kādā kārtībā procesu izpilda dažādi veicēji, kā arī informācijas plūsmu, ko viens veicējs var nodot nākamajam. Lai gūtu vispārīgu priekšstatu par biznesa procesu modelēšanas valodām, ir apskatīti dažu izplatītāko biznesa procesu modelēšanas valodu piemēri [7.3] (t.i., konkrēti modeļi).
Tipiska ARIS EPC (*Event-driven Process Chain*) diagramma [7.3] sastāv no funkcijām (kastītes ar noapaļotiem stūriem, piem., *Request service*) un notikumiem (sešstūri), kurus secīgi savieno ar raustītām bultām. Notikums pirms funkcijas (*Product needed*) izraisa funkcijas uzsākšanu, bet notikums pēc funkcijas tiek radīts, kad funkcija ir pabeigta (*New order*). EPC funkcijai (*Request service*), var būt ieejas un izejas dati (*Order*). Konkrētu funkciju pilda noteiktas organizatoriskās vienības pārstāvis (*Sales assistant*), un tai var būt nepieciešami resursi (*Customer*). Funkciju izpildes secības kontrolei iespējams izmantot izvēles un apvienošanas nosacījumus (aplīši):



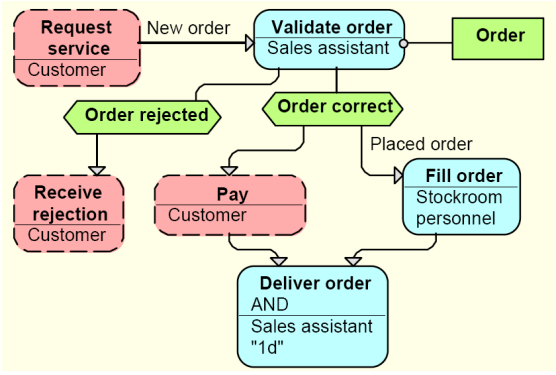
Att. 4 ARIS EPC diagrammas paraugs

Savukārt tipiska IDEF3 [7.3] diagramma sastāv no uzvedības vienībām (*UOB – Unit of behavior*, *Request service*) kas izkārtotas joslās (*Customer* un *Sales Assistant*), kas norāda, kurš doto uzvedības vienību spēj veikt. Uzvedības secību norāda ar nepārtrauktām bultām, bet darbības loģikas nosacījumus attēlo ar sadalīšanās un apvienojuma virsotnēm (kastītes ar *X* un *&*):



Att. 5 IDEF3 diagrammas paraugs

GRAPES BM biznesa procesu diagramma galvenie elementi ir uzdevumi (kastīte *Validate order*). Uzdevumu izpildes secību norāda ar šķautnēm – notikumiem (nenosauktas bultas) vai ziņojumiem (nosaukta bulta *New order*). Diagrammā ir iespējams attēlot arī datus (kastīte *Order*) un zarošanās nosacījumus (kastīte *Order correct*). Uzdevumu veicēji tiek norādīti uzdevuma kastītē apakšējā daļā (*Sales assistant*).

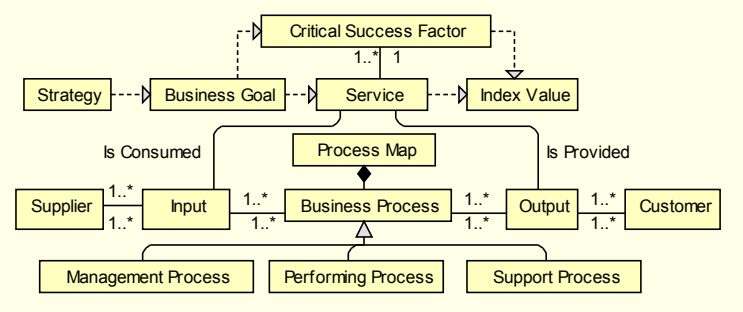


Att. 6 GRADE biznesa procesa diagrammas paraugs

Visas apskatītās biznesu procesu modelēšanas valodas sastāv no darbībām (funkcijas, UOB, uzdevumi), kas ir savienotas ar vadības šķautnēm. Diagrammām ir arī kontroles elementi – izvēles, sadalīšanās, apvienošanās un saplūšanas simboli, kuri var būt vai nebūt parādīti kā atsevišķas virsotnes. Darbības veicēju norāda darbības virsotnē, vai arī darbības virsotni ievieto veicēju joslā. Darbību var detalizēt ar citu diagrammu. Darbības tiek izpildītas sākot ar īpašu sākuma simbolu (vai netieši sākot ar to, kurai nav ieejošās vadības virsotnes), iespējamās paralēlās darbības tiek parādītas tieši ar paralēliem vadības plūsmas zariem.
IDEF3 diagrammas ir semantiski līdzīga ar vienkāršu stāvokļu diagrammu (kur neizmanto datus), savukārt ARIS EPC ir GRAPES BM semantiski līdzīgas krāsainajiem Petri tīkliem. Krāsaino Petri tīkla uzvedības semantiku izmanto arī UML 2.0 aktivitāšu diagramma, kuras darbība detalizēti tiek apskatīta darba turpmākajos posmos (nodaļās 4 un 5).

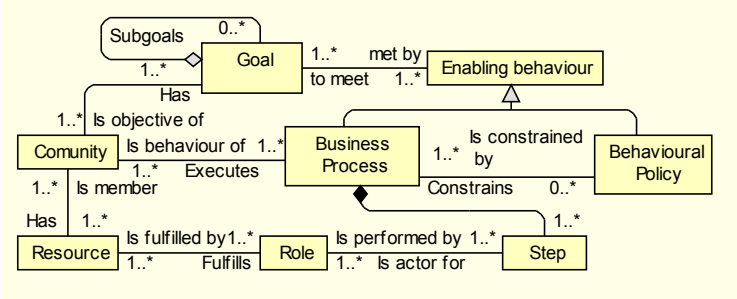
2.2 Biznesa procesa apkārtnes metamodeļi

Lai paskaidrotu biznesa procesa nozīmi, jēgu un kvalitātes kritērijus, ir svarīgi aprakstīt arī biznesa procesa apkārtni un tā "metadatus" – procesa ieejas (piegādātājus) un izejas (pasūtītājus), procesa saistību ar citiem procesiem, tā mērķus, nozīmi un kvalitātes mērus. Katrs konkrēts biznesa procesa apkārtnes modelis saskaņā ar MOF atbilst M1 slānim. Aplūkojot šo modeļu jēdzienus, mēs paceļamies nākamajā abstrakcijas slānī, tātad aplūkojam valodu jēdzienus jeb metamodeli (M2). Lai arī daži aplūkotie modeļi nav veidoti stingri saskaņā ar MOF, tie tiek aplūkoti no šāda formāli "pastiprināta" viedokļa.
Viens no piedāvātajiem biznesa procesa metamodeļiem ir izstrādāts saistība ar e-biznesu. E-biznesa procesu apkārtnes jēdzienus kā klašu diagrammu ir izstrādājis Andreass Diečs [7.3]. Šis metamodelis parāda biznesa mērķus un stratēģiju, kas sasaucas ar Zahmana ietvaru, un ievieš arī uzņēmuma ieejas un izejas jēdzienu. Daži jēdzieni (piegādātājs, ieeja, izeja, biznesa process, pasūtītājs) labi saskan ar ISO/DIS standartu [7.3]. Šis metamodelis ievieš arī vairākus uzņēmuma procesus (pārvaldības, uzturēšanas, veikšanas).



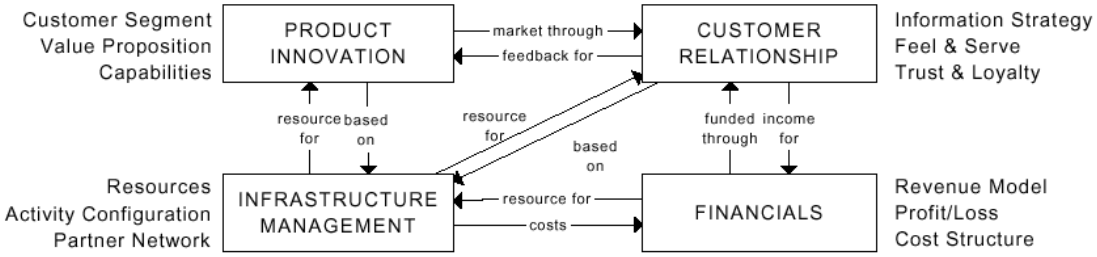
Att. 7 A. Dieča biznesa procesu apkārtnes metamodelis

Cits skats uz biznesa procesu apkārti ir izstrādāts *COMBINE* projektā [7.3]. Šajā projektā galvenie jēdzieni ir izstrādāti saistībā ar sadalītajām sistēmām:



Att. 8 Combine projekta biznesa procesu apkārtnes metamodelis

Šajā metamodelī ir labi parādīta biznesa procesa un resursu saistība. Tas, kas šajā modelī ir komūna (*Community*) citos metamodeļos tiek saukts par uzņēmumu (*Enterprise*).
Kā redzams, šie metamodeļi ir pietiekami dažādi, un vēl savādāks metamodelis tiek piedāvāts [7.3], kurā izstrādāta e-biznesa ontoloģija:



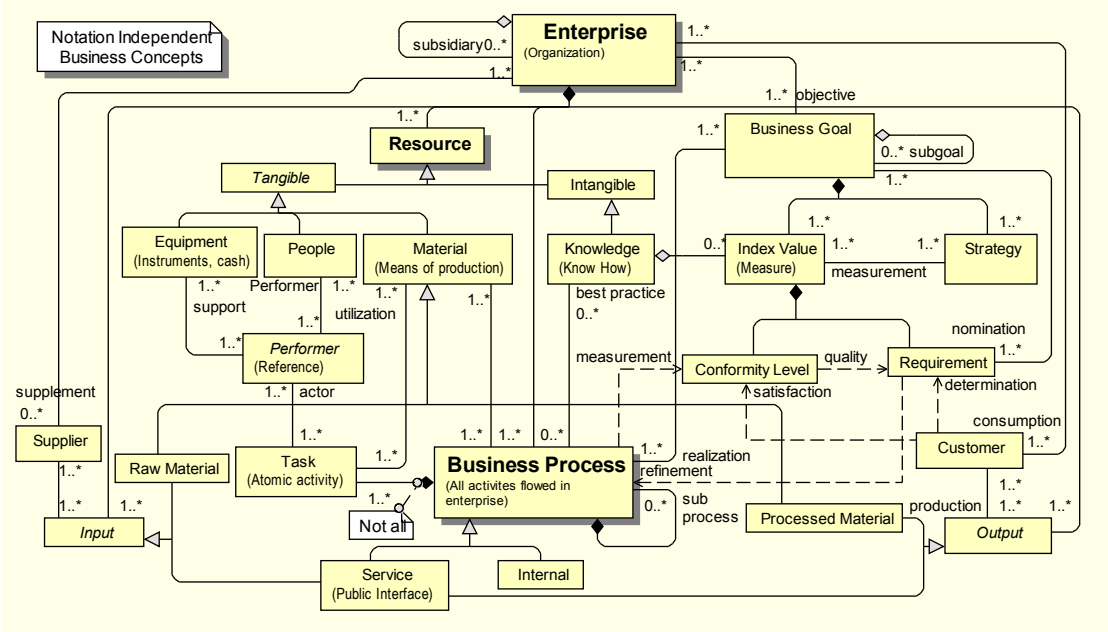
Att. 9 E-biznesa ontoloģija

Nedaudz biznesa procesu apkārtnes metamodelis ir aplūkots arī BPMN [7.3], kur biznesa apkārtne tiek skaidrota ar procesu saskarni un sadarbību.
Ir redzams, ka, līdzīgi kā paši biznesa procesi, arī to apkārtne tiek skaidrota ļoti dažādi un ļoti dažādi tiek paradīta arī to savstarpējā saistība. Tāpēc, lai izstrādātu vienotu metamodeli, ir nepieciešams pieņemt pietiekami vispārīgus un vienotus principus, saskaņā ar kuriem tiek apskatīti dažādie metamodeļos ieviestie jēdzieni. Apskatot vadošās biznesa procesu pārvaldības metodikas, kā pietiekami vispārīgas tika atzītas E. Deminga kvalitātes pārvaldības cikls, kas atspoguļojas arī ISO standartos [7.3], un M. Portera ieviestais pievienotās vērtības ķēdes process [7.3].

2.3 Galvenie biznesa jēdzieni un to saistība

Izpētot esošos, gan uz metamodeli bāzētos, gan citus modelēšanas ietvarus [7.3,7.3, 7.3,7.3], tika konstatēts, ka tie ir fragmentāri un ka neviens no tiem nav pietiekami aptverošs un vispārīgs. Tāpēc ir izstrādāts jauns biznesa procesu metamodelis, kurā ir apvienotas izplatītākās ietvaros sastaptās biznesa procesu pārvaldības metodes. Sastaptie jēdzieni ir īpaši harmonizēti, izmantojot metamodeļu "semantiskās līdzības" principu. Turklāt, apvienotajā metamodelī ir parādīti tikai svarīgākie jēdzieni, kas ir kopīgi (pēc modeļa harmonizācijas) visām apskatītajām biznesa procesu pārvaldības metodēm..
Att. 10 ir parādīts izstrādātais biznesa procesu apkārtnes metamodelis. Tas atbilst vairākiem pasaules vadošajiem biznesa pārvaldības standartiem (piem., pievienotās vērtības ķēdei, ISO kvalitātes standartam [7.3]). Metamodelis ir līdzīgs OMG izstrādātajam Biznesa motivācijas modelim [7.3]. Šī standarta izstrādi Biznesa likumu grupa (*Business Rules Group*) uzsāka 2000. gadā, bet metamodelis tika izstrādāts 2005. gadā, kad standarta izstrādi pārņēma OMG, un tas vēl joprojām ir projekta stadijā. Jāpiezīmē, ka procesa resursi, ieejas un izejas autora pētījumā ir aprakstīti

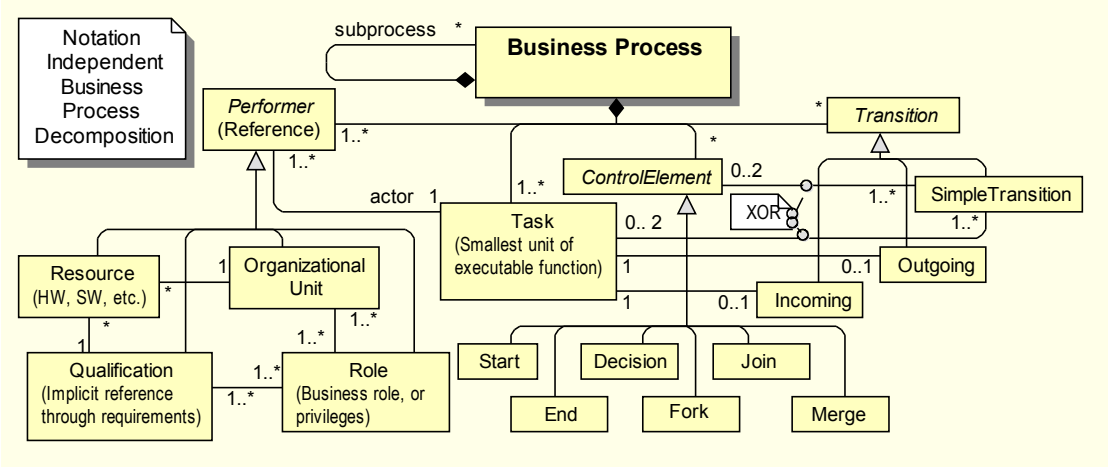
detalizētāk kā topošajā OMG standartā. Kā ontoloģija semantiski līdzīgs biznesa modelis ir izstrādāts 2006. gadā Lozannas universitātē [7.3].



Att. 10 Biznesa procesa apkārtnes metamodelis

Att. 11 ir parādīts izstrādātais biznesa procesa sastāvdaļu metamodelis, kas nepieciešams procesa definēšanai un izpildei. Procesa galvenā sastāvdaļa ir uzdevums, kas attēlo vienu nedalāmu darbību. Dažādās biznesa modelēšanas valodās to sauc dažādi: GRAPES BM un BPMN valodās to sauc par uzdevumu (*Task*), UML 1 – par darbības stāvokli (*Action State*), UML 2 – par darbību (*Action*), IDEF 3 – par uzvedības vienību (*Unit of Behavior*). Uzdevumu izpilda veicējs (*Performer*). Pārejas (*Transition*) nosaka, kādā kārtībā uzdevumi tiek izpildīti. Uzdevumi un pārejas ir galvenie biznesa procesa jēdzieni, kas parādās visās procesu modelēšanas valodās.

Vairumā modelēšanas valodu ir arī vadības elementi (*ControlElements*), kas nosaka procesa zarošanos. Izvēle (*Decision*) norāda zarošanās sākumu, sadalīšanās (*Fork*) – paralēlu plūsmu sākumu, savienošanās (*Merge*) – zarojumu apvienošanos un saplūšana (*Join*) – paralēlu zaru apvienošanu. Var būt arī tieši norādīti procesa sākuma (*Start*) un beigu (*End*) punkti. Tomēr vadības elementi, kā arī pāreju savienojumi ar vadības elementiem dažādās valodās atšķiras diezgan krasi. Tāpēc, lai viennozīmīgi attēlotu noteiktu vadības elementu atbilstību analizētajām UML 2.0 AD un GRAPES BM valodām, ir ieviestas papildus pāreju apakšklases – vienkāršā (*SimpleTransition*), ieejošā (*Incoming*) un izejošā (*Outgoing*) pāreja. Šo apakšklašu izmantošana jēdzienu kartēšanai (*mapping*) ir parādīta 2.4. nodaļā.

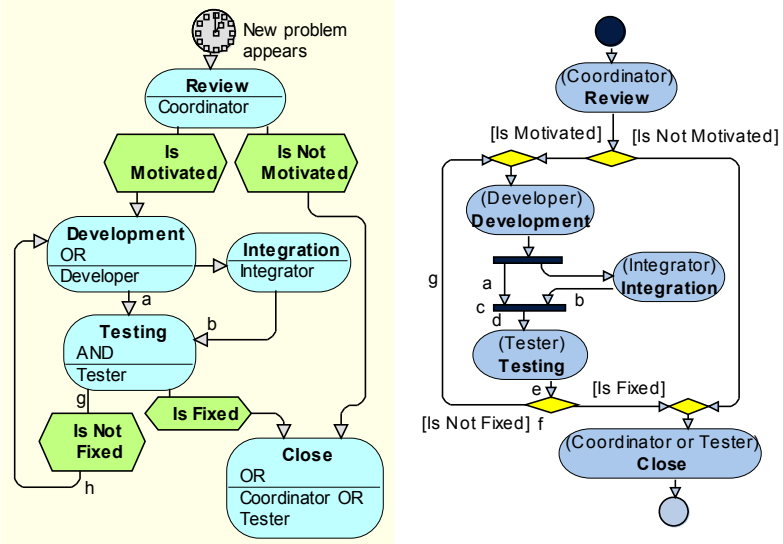


Att. 11 Biznesa procesa sastāvdaļu metamodelis

Šis biznesa procesa sastāvdaļu metamodelis ir līdzīgs vēlāk izstrādātajam OMG Procesu definīcijas standarta projektam [7.3]. Jāpiezīmē, ka uzdevuma veicēju modeļa daļa autora izstrādātajā variantā ir detalizētāka kā OMG piedāvātajā projektā.
Izstrādātais harmonizētais jeb "notācijas neatkarīgais" biznesa procesa apkārtnes un sastāvdaļu metamodelis (Att. 10, Att. 11) visos turpmākajos pētījumos ir izmantots kā biznesa procesu metamodeļa "kanoniskā forma" (3., 4. un 5. nodaļā). Lai arī izstrādātais metamodelis ir samērā neliels, tas ir pietiekami vispārīgs, jo parāda ikviena uzņēmuma biznesa procesu galvenos jēdzienus. Tāpēc tas labi saskan ar vēlāk izstrādātajiem starptautiskajiem standartiem.

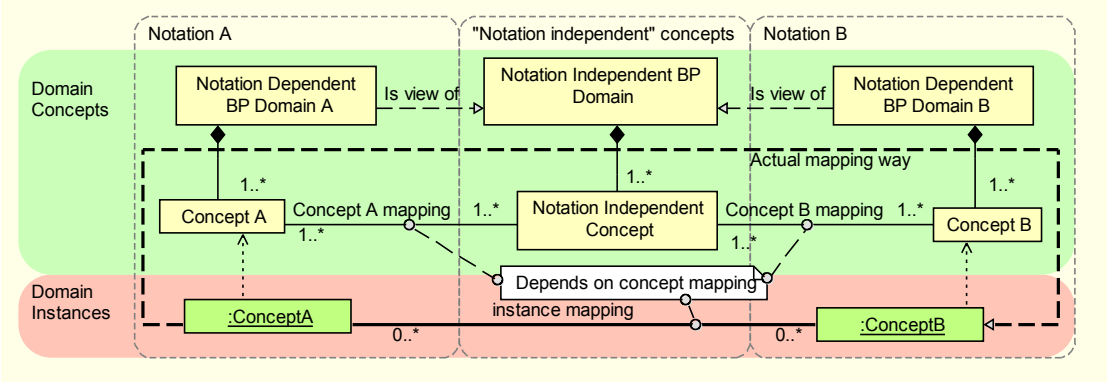
2.4 Jēdzienu kartēšana dažādās notācijās

Izmantojot "notācijas neatkarīgo" metamodeli, ir izstrādāta metode, kurā "semantiski līdzīgas" biznesa modelēšanas valodas var definēt kā specifiskus vispārīgā metamodeļa skatus.
Par piemēriem jēdzienu kartēšanas idejas izklāstam ir izmantotas UML 2.0 aktivitāšu diagramma (tajā laikā kā projekts) un GRAPES BM valoda. Att. 12 ir parādīts viens un tas pats biznesa process abās valodās, parādot šo valodu atšķirības – piem., nosacījumu virsotnes GRAPES BM [7.3] valodā attēlojas kā nosacījumi pie šķautnēm UML AD [7.3] valodā:



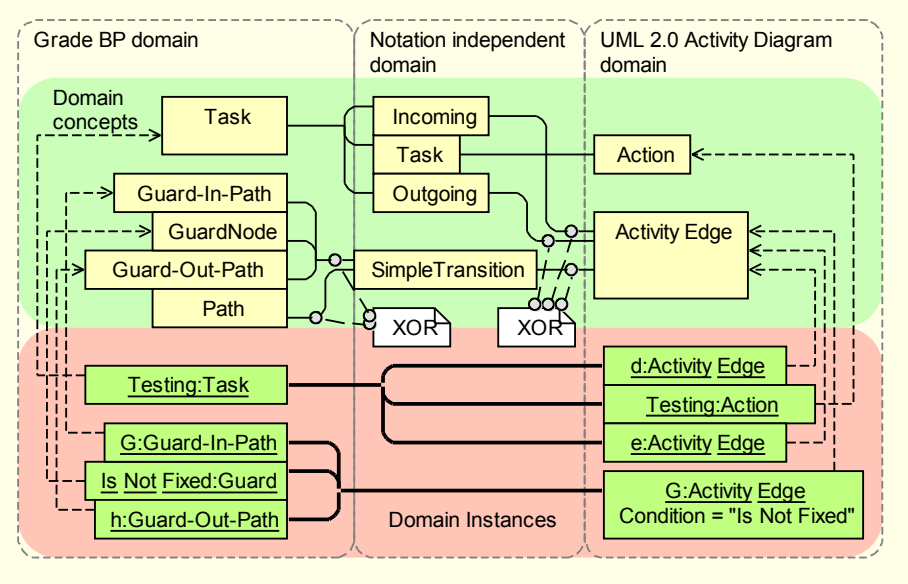
Att. 12 Biznesa process GRAPES BM un UML AD valodās

Jēdzienu kartēšanas vispārīgā shēma ir parādīta Att. 13. Attēla augšējā daļa parāda biznesa procesa aprakstīšanai izmantotos jēdzienus (*Domain concepts*). Jēdzieni valodā A (*Notation A*) tiek kartēti uz jēdzieniem valodā B (*Notation B*), kā starpniekus izmantojot "notācijas neatkarīgos" (*Notation independent*) jēdzienus. Daudzos gadījumos saistība starp dažādu valodu jēdzieniem ir daudzi-pret-daudziem, un tā nav pietiekami noteikta un izsekojama. Lai šādu atbilstību "normalizētu", tiek ieviesti papildus starpposma jēdzieni, kas daudzi-pret-daudziem atbilstību starp valodu jēdzieniem parāda kā viens-pret-daudziem atbilstību pāri starp abām valodām un starpposma jēdzieniem.
Kad kartējums, izmantojot starpposma jēdzienus, ir viennozīmīgi definēts, jēdzienu instances vienā notācijā tiek pārvērstas par jēdzienu instancēm otrā notācijā, kā tas parādīts attēla apakšējā daļā. Atkarību līnijas parāda <<instance of>> saistību starp jēdzienu klasēm un to instancēm. Lai arī vispārīgā gadījumā instanču atbilstība ir daudzi-pret-daudziem, konkrētām instancēm šī atbilstība parasti ir viens-pret-daudziem, kā tas parādīts Att. 14.



Att. 13 Metamodeļu kartēšanas vispārīgā shēma

Att. 14 ir parādīts konkrēts kartējuma fragments starp GRAPES BM un UML AD valodām. Attēla augšējā daļa parāda domēna jēdzienus ar GRADE metamodeļa klasēm kreisajā pusē, UML AD metamodeļa klasēm labajā pusē un notācijas neatkarīgā domēna klasēm pa vidu. Vispārīgā daudzi-pret-daudziem atbilstība starp GRAPES BM un UML AD nosacījumu virsotnēm un šķautnēm tiek novērsta, izmantojot vairākas viens-pret-daudziem attiecības starp valodu jēdzieniem un starpposma jēdzieniem (*Transition* apakšklases – *Incoming*, *Outgoing* un *SimpleTransition*). Att. 14 apakšējā daļa parāda konkrētas domēna jēdzienu instances atbilstoši Att. 12 parādītajai procesa diagrammai. Kā tas redzams Att. 14, konkrētām instancēm abās valodās atbilstība ir viens-pret-daudziem, tātad, vienkārši izsekojama.



Att. 14 Kartējuma definīcijas un instanču fragments

Lietotāju patiesībā interesē abu modelēšanas valodu grafiskā prezentācija, tāpēc pēdējais solis ir atbilstošo grafisko elementu (šķautņu un virsotņu) izveide diagrammā. Tā kā domēna elementi ar grafisko prezentāciju parasti ir saistīti kā viens-pret-vienu, tas ir salīdzinoši vienkāršs uzdevums. Kā tas redzams Att. 14 parādītajā kartējuma fragmentā, kaut arī jēdzienu kartēšana tiek veikta starp "semantiski līdzīgām" valodām, to tehniskie elementi atšķiras pietiekami, lai atbilstošo jēdzienu kartējums būtu diezgan sarežģīts.

Jēdzienu kartēšanas paņēmienu šobrīd izmanto OMG Domēna darba grupa (*Domain task force*), apvienojot UML 2.0 AD un BPMN 1.0 valodas [7.3]. Tā kā UML AD un BPMN ir vēl lielākas atšķirības, kā UML un GRAPES BM (piem., AD nav ekvivalenta BPMN biznesa procesam, kas satur vairākus procesus, savukārt BPMN nav precīza ekvivalenta AD objektu plūsmai), jēdzienu atbilstību ar kartēšanas paņēmienu vai nu nevar parādīt viennozīmīgi, vai arī ir nepieciešams ieviest daudz starpposma jēdzienu. Tomēr gadījumos, kad "semantiski līdzīgas" valodas izmanto arī līdzīgus tehniskos elementus, jēdzienu kartēšana ir ērts vienkāršojums. Pēdējā laikā šādā veidā ir gūti zināmi panākumi, kartējot BPMN, BPEL, XLANG un WSFL valodas [7.3].

Vēlākā pētījumā (aprakstīts 5. nodaļā) ir pierādīts, ka sarežģītu jēdzienu atbilstību var efektīvi risināt, izmantojot modeļu transformācijas. Tādā gadījumā kartēšanas asociācijas ir noderīgas divējādi – definīcijā tās uzskatāmi parāda vispārīgo kartēšanas shēmu (kas gan nav viennozīmīga), bet konkrētām instancēm, katrai mērķa instancei parādot avota instanci, tās parāda transformācijas rezultātu, kas ir nepieciešams transformāciju trasējamībai.

3 Biznesa procesu mēri

Biznesa pasaulei globalizējoties, konkurence kļūst arvien asāka, bet biznesa procesi ģeogrāfiski kļūst arvien sadalītāki. Lai noteiktu kopējo procesa efektivitāti, ļoti svarīgi ir noteikt katra atsevišķā biznesa procesa soļa izmaksas, laiku u.c. parametrus. Biznesa procesu efektivitātes mērījumi ir svarīgi uzņēmuma kopējo izmaksu un investīciju atpelnīšanās noteikšanā. Kā to parāda pētījumā veiktā esošo biznesa vadības sistēmu analīze, procesu mērīšanas iespēja ir to neiztrūkstoša sastāvdaļa [7.3].
Daudzas kvalitātes un biznesa procesu pārvaldības metodikas biznesa procesu stipro un vājo pušu noteikšanai izmanto skaitliskas metodes [7.3,7.3,7.3,7.3,7.3]. Tās atbalsta dažādi rīki [7.3,7.3,7.3,7.3], tomēr katrs no tiem piedāvā tikai "savā klasē labāko" metodi kādam noteiktam biznesa procesa aspektam, nenodrošinot vienlaicīgu dažādu paņēmienu pielietošanu. Mēru definēšana un to vērtības aprēķināšana, it sevišķi vērtību agregēšana (piem., summēšana, vidējās vērtības, minimuma un maksimuma noteikšana) esošajās darba plūsmas pārvaldības sistēmās un imitācijas rīkos ir sarežģīts uzdevums, kas prasa pamatīgas tehniskās zināšanas.
Pētījuma mērķis bija izstrādāt mēru definēšanas un apstrādes ietvaru, kas noteiktu algoritmus un ierobežojumus, kas biznesa procesa elementiem piedāvātu tikai tādus mēru veidus un to agregēšanu, kuriem ir praktiska nozīme. Šādā ietvarā mērus varētu norādīt pašā biznesa procesa modelī. Biznesa procesu analītiķiem tas nodrošinātu ērtu procesa mēru definēšanu un ļautu domāt biznesa jēdzienos, izvairoties no procesa izpildes īpatnībām.
Darbā ir praktiski pierādīts, kā šādu ietvaru iespējams izveidot, izmantojot metamodelēšanas paņēmienus. Mēru definēšanas valoda ir izstrādāta ar UML [7.3] aktivitāšu diagrammas profila palīdzību, bet mēru apstrādes ietvars ir izveidots, ar "smagsvara paplašināšanu" paplašinot UML metametamodeli [7.3].
Izstrādātā pieeja ir aprakstīta, sākot ar konkrētu procesa un mēru definīcijas piemēru un beidzot ar vispārīgu mēru definēšanas ietvaru.

3.1 Biznesa procesu mērīšanas metodikas

Līdz ar biznesa procesa jēdzienu ieviešanu tika izstrādātas kvantitatīvas metodes, kā noteikt biznesa procesu efektivitāti, kvalitāti izmaksas, u.c. parametrus. Laika gaitā ir izstrādātas daudzas ļoti atšķirīgas biznesa procesu mērīšanas un kvalitātes kontroles metodikas. Lai gūtu vispārīgu priekšstatu, ir aplūkotas dažas pasaulē izplatītākās.
Darbību izmaksu (*Activity based costing*) [7.3] metodika tika izstrādāta ASV pagājušā gadsimta 70.tajos – 80.tajos gados. Šī metode ir procesu orientēta. Tās pamatideja ir tāda, ka piegādājot kādu produktu vai pakalpojumu, izmaksas tiek rēķinātas, nevis pēc grāmatvedības pozīcijām (algas, nodokļi, materiāli, pamatlīdzekļu nolietojums, u.tml.), bet gan katram produkta izstrādes vai pakalpojums sniegšanas solim jeb elementārai darbībai. Tādējādi iespējams analizēt un optimizēt katru atsevišķu biznesa procesa darbību. Metodikā tiek izšķirti vairāki darbības izmaksu veidi, fiksētās izmaksas un atkarīgās izmaksas. Izmaksu atkarība (*cost drivers*) var būt atkarīga no laika, no aktivitātes izpildes biežuma u.c. parametriem. Šīs metodes princips, ka katrai aktivitātei var piemērot izmaksas, ir ieviests daudzos biznesa procesu un resursu pārvaldības sistēmās, kā arī modelēšanas rīkos [7.3,7.3,7.3,7.3]. Vairums rīku gan operē ar fiksētām izmaksām, un to atkarību no dažādiem parametriem (izmaksas uzdot kā formulu) nav iespējams.
CMM (*Capability Maturity Model*) [7.3] ieviesa Vets Humprejs (*Watts Humphrey*) 1986. gadā. Metodes jaunākā versija CMMI (*Capability Maturity Model Integration*) tika izstrādāta 2003. gadā. Tā ir uz procesu orientēta kvalitātes pārvaldības metodika, kas analizē veidus, kā uzlabot liela mēroga programmatūras izstrādes projektus.

CMMI ievieš vairākus pārvaldības līmeņus. Ja organizācija atrodas zemākajā līmenī, tajā nepastāv nekādas noteiktas pārvaldības metodikas (projektu pārvaldība notiek ekspromtā, balstoties uz darbinieku pieredzi), bet augstākajā līmenī ne tikai tiek veikti daudzi un dažādi procesu kvalitātes mērījumi, bet, balstoties uz mērījumu rezultātiem, organizācija pastāvīgi uzlabo savus biznesa procesus. Tā kā metodika vairāk ir vērsta uz organizācijas kopējās kvalitātes uzlabošanu, procesu mērījumi pamatā tiek veikti ar statistiskām metodēm. Kā mērījumu piemērus var minēt:

- problēmu/defektu attiecība uz visu saražoto/piegādāto,
- prasību izmaiņu attiecība no visām prasībām,
- izmaksas, laiks, resursi viena uzdevuma veikšanai,
- faktiski iztērēto izmaksas/laika/resursu attiecība pret plānoto,
- problēmu/defektu novēršanas izmaksas no kopējā,
- izmaksu/resursu/problēmu/defektu izmaiņas laikā.

Tā kā CMM galvenokārt operē ar agregētiem datiem, biznesa vadības sistēmās šī metode tieši nav integrēta.

Līdzsvaroto mērījumu (*Balanced Scorecard*) [7.3] metodi 1992. gadā ieviesa Roberts Kaplans (*Robert S. Kaplan*) un Deivids Nortons (*David P. Norton*). Šī metode sākotnēji tika izstrādāta ražojošo uzņēmumu pārvaldībai, bet vēlāk tika pielāgota arī pakalpojumu sniegšanas uzņēmumiem. Bez biznesa procesu pārvaldības šī metode pievēršas arī citu uzņēmumu raksturojošu parametru kontrolei. Tās pamatideja ir tā, ka uzņēmuma vienmērīgai attīstībai ir nepieciešams kontrolēt ne tikai finansiālos rādītājus. Šie rādītāji tiek grupēti tādās grupās kā Finanses, Pasūtītājs, Iekšēji biznesa procesi, Mācīšanās un augšana. Katrai grupai tiek noteikti 5-6 kvantitatīvi izmērāmi rādītāji. Mērījumu tiek veikti salīdzinot konkrēto izorientēto uzdevumu plānotās un faktiskās vērtības. Kā tipiskus šīs metodes mērus var minēt:

- Finansiālie
  - Naudas plūsma
  - Investīciju atgriešanās (*ROI*)
  - Finansiālie rādītāji
- Pasūtītājs
  - Piegādes datums
  - Piegādes daudzums
- Iekšējie biznesa procesi
  - Veikto darbību skaits
  - Darbības izmaksas
- Mācīšanās un augšana
  - Investīciju apjoms
  - Darbinieku nestrādāto dienu skaits

Izmantojot šo metodi, bieži tiek izmantots t.s. kontrolpanelis (*dashboard*), kas uzlabo situācijas pārskatāmību. Šajā kontrolpanelī, ja faktiskie mērījumu rezultāti iekļaujas plānotajā diapazonā, to attēlo zaļu, bet, ja ir novirzes, atkarībā no to lieluma, rādījumu attēlo dzeltenu vai sarkanu.

6σ (*six sigma*) metodi izstrādāja Motorola 1986. gadā [7.3]. Tā ir statistiska metode, kurā galvenā uzmanība tiek pievērsta procesa noviržu noteikšanai, analīzei un noviržu samazināšanai. 6σ metodes nosaukums cēlies no statistiskās sakarības, ka, ja kādā mērījuma vidējā kvadrātiskā novirze ir σ, tad pārskatāmā mērījumu laikā praktiski nav iespējams, ka kādam konkrētam mērījumam novirze varētu pārsniegt 6σ. Šīs metodes balstās uz to, ka:

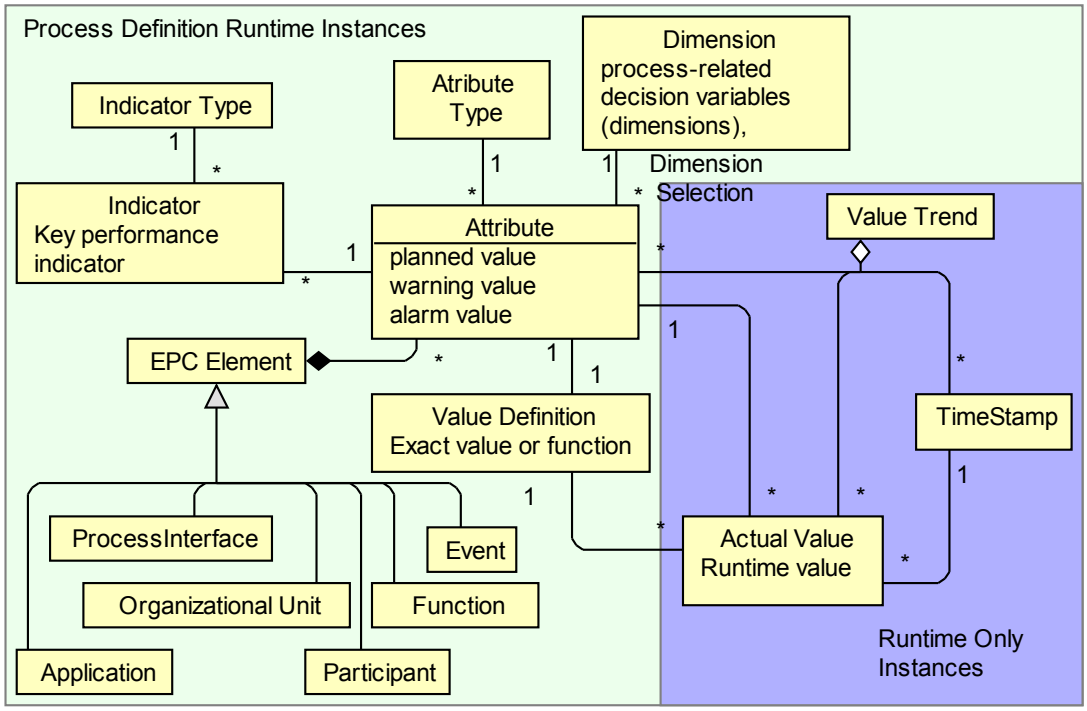
- Nepārtraukta procesa izejas (piegādes pasūtītājam) noviržu novēršana ir galvenais biznesa veiksmes faktors.
- Biznesa procesu un ražošanu var mērīt, analizēt, kontrolēt un arī uzlabot.

- Lai sasniegtu nepārtrauktu kvalitātes uzlabojumu, nepieciešams, lai tajā iesaistās visa organizācija, tajā skaitā arī augstākā līmeņa vadība.

Balstoties uz mērījumu datiem, 6σ metode piedāvā soļus problēmu risināšanai, fokusējoties galvenokārt uz procesa uzlabošanu un to, kā piegādāt pasūtītājam to, kas viņam nepieciešams. Metode ir implementēta dažādos analīzes rīkos, piemēram [7.3].
Nemateriālo resursu pārraudzības (*Intangilbe Assets Monitoring*) metodi ieviesa Kārlis Eriks Sveibijs (*Karl Erik Sveiby*) 1987. gadā [7.3]. Šī metode tehniski sasaucas ar līdzsvaroto mērījumu metodi, tomēr tās galvenā atšķirība ir tā, ka tajā par galveno vērtību tiek uzskatīti uzņēmumā strādājošie cilvēki un to zināšanas. Šī metode sākotnēji tika piemērota augsto tehnoloģiju uzņēmumu pārvaldībai, kuros izpēte un izstrāde (*R&D*) sastāda nozīmīgu uzņēmuma darbības daļu. Tā kā mūsdienās izpēte, izstrāde un darbinieku zināšanas ieņem nozīmīgu lomu arvien plašākās biznesa jomās, šī metode kļūst piemērota arvien plašākam uzņēmumu lokam.
Šajā metodē bez uzņēmuma materiālajiem resursiem nemateriālos resursu sadala trīs grupās – iekšējos, ārējos un kompetencē. Uzņēmuma ārējo resursu grupā ieskaita klientus, to apmierinātības novērtējumu, klientu stabilitāti, apgrozījumu un peļņu uz vienu klientu. Iekšējo resursu grupā apskata tādus rādītājus kā investīcijas attīstība, vecāko un jaunāko darbinieku attiecību, procesu uzlabojumu un kvalitāti. Kompetences grupā apskata tādus rādītājus kā darbinieku kompetences rādītājus, kadru mainību jaunāko un vecāko darbinieku vidū, jaunāko un vecāko darbinieku attiecību, apgrozījuma un pelņas rādītājus uz vienu darbinieku.
Šajā metodē uzņēmuma apgrozījumu un peļņu apskata no vairākiem aspektiem – gan attiecībā uz klientu, gan attiecībā uz uzņēmuma procesiem un darbiniekiem. Tādējādi iespējams noteikt, kas ir labākā uzņēmuma pievienotā vērtība, un ļauj fokusēties uz noteiktu attīstības virzienu un uzlabojumiem.

3.2 Procesu izpildes mērīšanas rīki

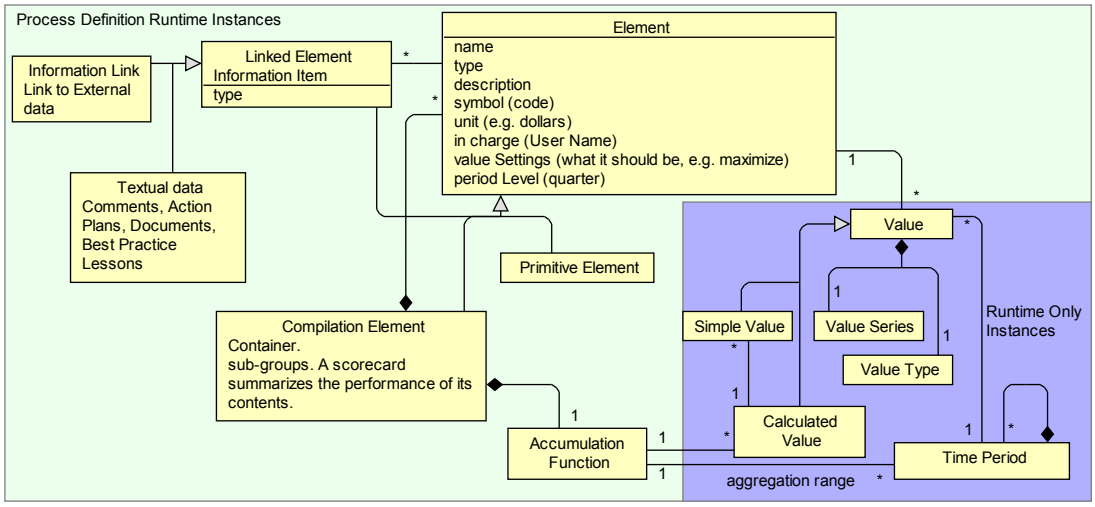
Procesu imitācijas un izpildes mērīšana dažādos rīkos tiek veikta ļoti dažādi. Ir apskatīti ARIS [7.3], QPR [7.3] un GRADE [7.3,7.3] rīki, kuru izpildes laika (*runtime*) datiem ir izstrādāti metamodeļi. Lai arī izstrādātie metamodeļi nav pilnīgi, tie labi ilustrē procesu izpildes un mērīšanas pieeju dažādību šajos rīkos.
ARIS procesu imitācija tiek veikta, izpildot EPC procesu definīcijas valodu. Rīka izpildes metamodelis ir veidots, analizējot rīka aprakstu [7.3]. Modeļa objektiem (EPC diagrammas elementiem) var piekārtot dažādus atribūtus, kuru vērtību (*Value Definition*) definē tieši vai ar funkciju uzreiz modelī, vai arī nosaka tikai procesa izpildes laikā. Modeļa izpildes laikā atribūtu aktuālās vērtības tiek saglabātas, piesaistot attiecīgam laika momentam. Procesa elementam iespējams noteikt arī agregētas vērtības, kas tiek uzdotas kā noteikta funkcija no citām elementārām vērtībām.
Izmantojot atribūta vērtības dažādos laika momentos, tās var tālāk apstrādāt izmantojot laika funkcijas, kā arī iespējams analizēt vērtības izmaiņas laikā. Atribūtiem var piekārtot īpašus indikatorus, kas veic izpildes vērtību pārraudzību. Ja izpildes vērtība sasniedz definēto vērtības slieksni, indikators var veikt kādu darbību (iezīmēt attiecīgo elementu sarkanu, u.tml.).
Izpildot procesa diagrammu, procesa definīcijas instancēm tiek radītas atbilstošas izpildes laika instances, kā arī papildus elementi, kas nepieciešamas procesa izpildei (piem., dzinēji un marķieri). Lai vienkāršotu metamodeli, šeit ir parādīti tikai galvenie procesa izpildes laika dati. (Detalizēti procesa izpilde ir skaidrota 4.2.3 nodaļā). Instances, kuras tiek radītas atbilstoši procesa definīcijai, ir attēlotas ar klasēm uz gaiša fona. Savukārt, tie dati, kas rodas tikai procesa izpildes laikā, ir attēloti ar klasēm uz tumša fona. Vispārīgs ARIS procesu imitācijas un mērīšanas rīka metamodelis ir dots Att. 15.



Att. 15 ARIS imitācijas rīka metamodelis

QPR rīkam procesu definēšanai ir sava procesu modelēšanas valoda. Arī šī rīka procesu imitācijas un mērīšanas dzinēja metamodelis ir veidots, analizējot rīka aprakstu [7.3]. QPR rīkā procesa diagrammas elementi var būt elementāri, vai arī kalpot kā norādes uz citiem elementiem, vai arī būt agregēti (atvasināti) no citiem elementāriem elementiem. Elementiem, atkarībā no to tipa, procesa izpildes laikā var būt dažādas vērtības, kas var būt elementāras, vai arī iegūtas atvasināti no kādas funkcijas. Atšķirībā no ARIS, QPR rīkā izpildes laiks netiek uzskaitīts ar konkrētiem laika momentiem, bet gan katra vērtība tiek piekārtota kādam laika diapazonam (periodam). Laika periodu lielums un to kompozīcija tiek definēta procesa modelī. Lai arī šāds paņēmiens neļauj izmantot patvaļīgu laika periodu izmantošanu procesa analīzē, tas būtiski paātrina imitācijas ātrumu, jo rīks var izlaist lielus laika periodus, kuros procesā nenotiek nekādas izmaiņas. Tā kā rīks implementē līdzsvaroto mērījumu metodiku, tajā ir predefinēti šīs metodikas elementu tipi, piemēram, darbības izmaksas, darbības ilgums, darbības skaits laika vienībā.

QPR diagrammu izpildes laika datu metamodelis ir attēlots Att. 16. Arī šeit instances, kas tiek radītas atbilstoši procesa definīcijai ir attēlotas ar klasēm uz gaiša fona, bet tās datu instances, kas rodas tikai izpildes laikā – ar klasēm uz tumša fona.



Att. 16 QPR rīka procesu mērīšanas metamodelis

GRADE BM imitācijas rīkā tiek izpildīti GRAPES BM valodas modeļi. Imitācijas rīka metamodelis ir izveidots balstoties uz [7.3,7.3] aprakstu.

Modelī norādītajiem elementiem – uzdevumiem (*Task*), resursiem (*Resource*) un veicējiem (*Performer*) var definēt vairāku parametru vērtības. Procesa imitācijas laikā modeļa elementiem tiek noteikta parametru vērtību bāzes statistika (minimālā, maksimālā un vidējā vērtība). Izmantojot elementu bāzes statistiku, var definēt un imitācijas laikā mērīt arī atvasinātas parametru vērtības.

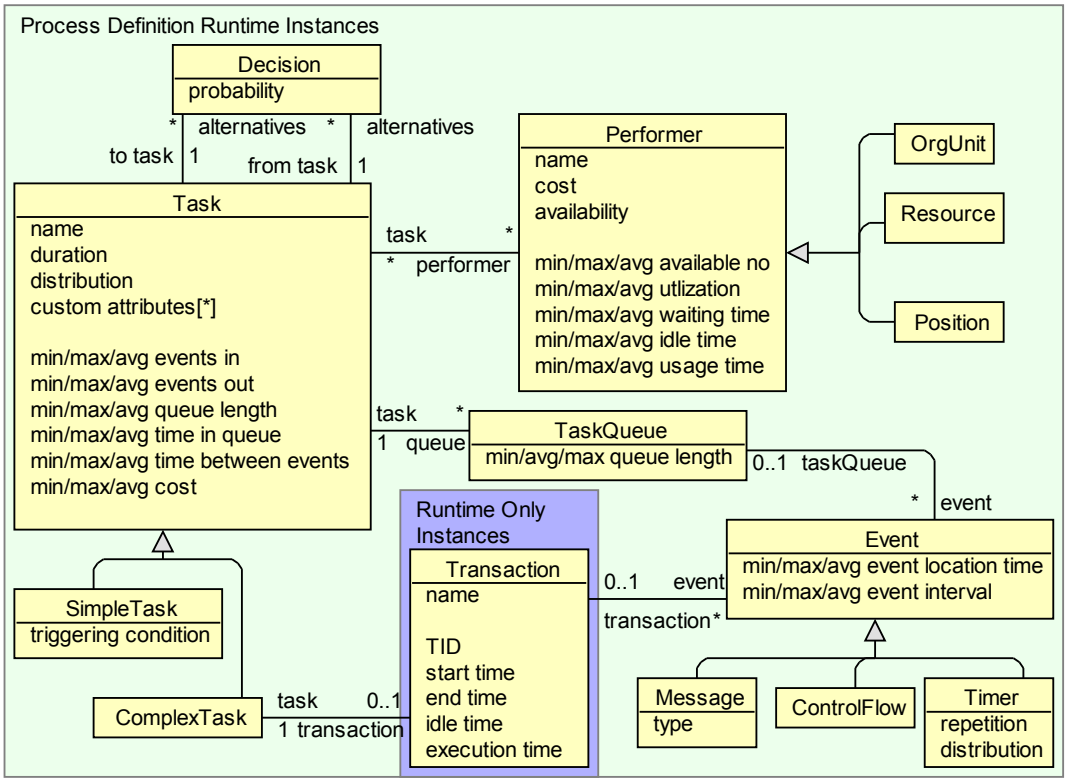
Procesa definīcijas elementiem tiek izveidotas atbilstošas izpildes laika instances, un imitācijas process tiek uzsākts, kad rodas ziņojuma (*Message*) instance, vai procesa taimeris (*Timer*) sasniedz norādīto laika vērtību. Konkrēts uzdevums imitācijas laikā tiek izpildīts, ja ir pieejami uzdevuma resursi un/vai veicēji, un izpildās norādītie elementārā uzdevuma (*SimpleTask*) izpildes nosacījumi (*triggering condition*). Izpildot uzdevumu, notikumi tiek savākti no uzdevuma ieejas rindām un tiek nodoti uzdevuma apstrādei. Apstrādājot uzdevumu, tiek atjaunoti uzdevuma, resursu un veicēju parametru vērtības. Uzdevuma izpildes beigas nosaka, izmantojot uzdevuma izpildes laika nosacījumus (*duration*). Kad uzdevums ir izpildīts, tiek ģenerēti jauni notikumi, kas tiek nodoti sekojošo uzdevumu ieejas rindās, atbilstoši izvēlēs (*Decision*) norādītajiem sadalījuma nosacījumiem (*probability*).

Ja process ir strukturēts vairākos līmeņos, uzsākot salikta uzdevuma izpildi, tiek izveidota jauna transakcija (*Transaction*). Ja notikums ietilpst transakcijā, apstrādājot notikumu, tiek atjaunota arī transakcijas statistika. Kad notikums atstāj pēdējo apakšprocesa uzdevumu, transakcija tiek noslēgta, attiecīgi fiksējot tās beigu laiku, un kopējo izpildes laiku.

Tā kā procesa imitācijā tiek cikliski apstrādāti notikumu marķieri, tā notiek salīdzinoši ātri, jo dzinējs izlaiž laika momentus, kuros nav notikumu. To, cik lielus laika posmus dzinējs var izlaist, nosaka diagrammā izmantoto taimeru un uzdevumu izpildes ilguma nosacījumi.

GRADE imitācijas rīka izpildes laika datu metamodelis ir parādīts Att. 17. Šajā metamodelī atšķirībā no iepriekšējiem diviem (Att. 15, Att. 16) izpildes laika atribūti nav savākti vienkopus klasē, bet ir parādīti detalizētāk, piekārtojot tos attiecīgajām izpildes laika klasēm. Atribūti, kuru vērtības tiek mantotas no procesa definīcijas, ir parādīti klašu augšējā daļā, bet atribūti, kuru vērtības tiek iegūtas tikai izpildes laikā, ir parādīti klašu apakšējā daļā. Tā kā GRAPES BM valodā (atšķirībā, piemēram, no UML AD) kontroles plūsmas (*ControlFlow*) ir definētas kā beztipa notikumi, visi diagrammas darbināšanai nepieciešamie marķieri ir diagrammā definēto notikumu

izpildes laika instances. Līdz ar to, kā jauna izpildes elementu klase parādās tikai transakcijas (*Transaction*), kas tiek radītas uzsākot salikta uzdevuma izpildi.



Att. 17 GRADE procesu imitācijas elementu metamodelis

3.3 Mērīšanas metodiku vienotas apstrādes principi

Apskatot vadošās procesu un kvalitātes pārvaldības metodikas, var redzēt, ka, lai arī to pamatideja ir līdzīga – mērīt un uzlabot uzņēmuma biznesa procesus, izmantotie paņēmieni ir ļoti atšķirīgi. Tāpēc arī līdz šim dažādu metodiku ieviešana vienā rīkā vēl nav veikta. No vienas puses – tam būtu jābūt pietiekami elastīgam un vispārīgam, lai tajā varētu implementēt arī principiāli atšķirīgas metodikas. Bet no otras puses – rīkā ir nepieciešami ierobežojumi (ietvars), kas rīka lietotāju "vada" pareizajā virzienā, un ļauj iegūt tikai tādus agregētus un apstrādātus datus, kam ir praktiska jēga.

Lai arī rīkam izvirzītās prasības šķiet pretrunīgas, šī problēma tomēr ir risināma, izmantojot metamodelēšanas pieeju. Izmantojot vairākus abstrakcijas slāņus, iespējams izstrādāt stingru ietvaru, kas "vada" lietotāju pareizajā virzienā, bet tajā pat laikā šis ietvars nav stingri fiksēts, un to iespējams modificēt un paplašināt. Principiālais rīka slāņu apraksts ir sekojošs:

- Konkrētas biznesa procesu mēru definīcijas (reālās pasaules modeļi) tiek veidoti M1 slānī, veidojot M2 slāņa instances.
- Mērījumu metodes un likumi tiek definēti kā "procesu mērīšanas modeļu sintakse" M2 slānī, veidojot M3 slāņa instances.
- Mērīšanas valodu sintakse tiek definēta fiksētā metametamodelī (M3 abstrakcijas slānī).

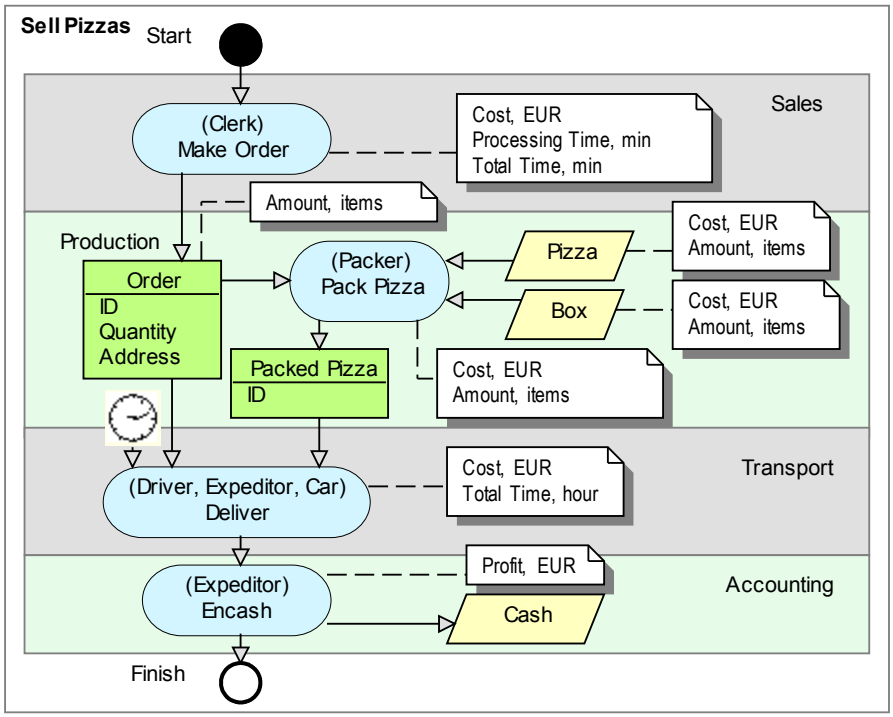
Apskatīto mērījumu metodiku jeb "procesu mērīšanas valodu" kopīgās īpašības var novērtēt, analizējot izveidotos procesu analīzes rīku metamodeļus (Att. 15, Att. 16, Att. 17). Var redzēt, ka kopīgie (ekvivalentie), jēdzieni ir ļoti vispārīgi, un tos var raksturot sekojoši:

- Procesu mērīšanas rīkā ir procesa definīcijas daļa, kurā var definēt biznesa procesa modeli un uzdot mērus; un procesa izpildes daļa, kurā process tiek izpildīts saskaņā ar uzdoto modeli, un tiek noteiktas izpildes laika vērtības.
- Jebkurš process sastāv no elementiem, kuriem var piekārtot īpašības. Tajā skaitā īpašība var būt arī kāda mēra definīcija.
- Mēra definīcija sastāv no mērvienības un tieši noteiktas vērtības vai funkcijas, kas apstrādā citu mēru vērtības.
- Funkciju argumentu vērtības var būt definētas tieši, vai arī tikt iegūtas atvasināti kā kādu citu funkciju vērtības.

Šie kopīgie vispārīgie jēdzieni tiek nemaināmi "iešūti" (*hardcoded*) rīka metametamodelī, tomēr, pateicoties tā vispārīgumam, metametamodelis ir pietiekami elastīgs dažādu mērījumu metožu implementēšanai. Mērījumu metodikas tiek implementētas kā konkrētas metametamodeļa instances. Savukārt veidojot konkrētus mērījumus kā metamodeļa instances, rīks piedāvā tikai metamodelī definētās iespējas, tādējādi būtiski atvieglojot tā lietošanu.
Piedāvātais risinājums ir parādīts kā lietojuma piemērs, vispirms to aprakstot no lietotāja viedokļa kā tam būtu jāizskatās, (kas faktiski ir specifikācija), un tālāk parādot, kas ir nepieciešams lai šādas prasības būtu iespējams implementēt.

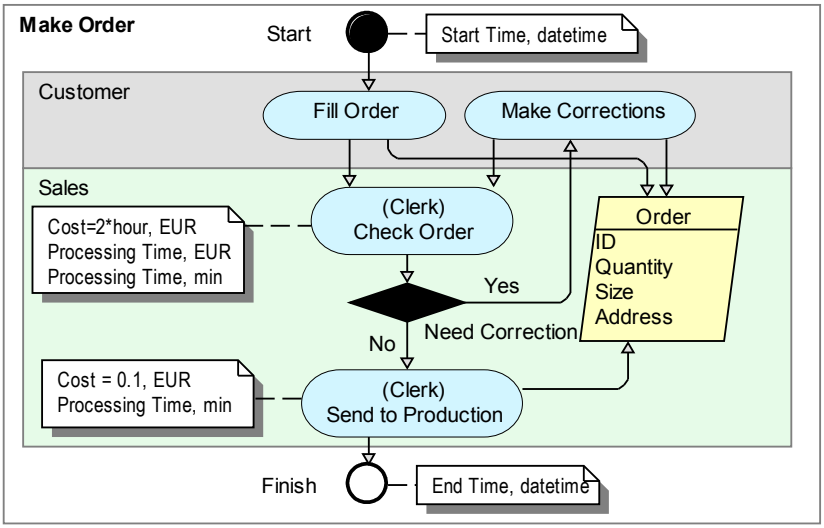
3.4 Biznesa procesa modelis

Procesa mēru apstrādes principu demonstrācijai tiek izmantots vienkāršs biznesa process, kas aprakstīts, izmatojot UML aktivitāšu diagrammu (AD) [7.3]. Att. 18 ir redzams biznesa process veikalam, kas piegādā picas uz pasūtītāju mājām. Picu pārdošana (*Sell Pizzas*) ir biznesa process (aktivitāte AD valodā), kas sastāv no vairākām darbībām (noapaļoti taisnstūri). Organizatoriskā struktūra nav atsevišķi parādīta, to var saprast no procesa apraksta. Organizatoriskās struktūrvienības ir attēlotas ar procesa joslām, un tajās esošie procesu veicēji (amati/lomas un resursi) ir norādīti darbības nodalījumā iekavās. Ar amatiem/lomām norādītie veicēji (*People*) ir aktīvi – tie veic manuālas darbības, resursi ir pasīvi, tie ir nepieciešami uzdevuma izpildei un tiek aizņemti vai iztērēti izpildes laikā. Objektu plūsma ir attēlota ar objektu virsotnēm (taisnstūri), krājumi ir attēloti kā paralelogrami. Procesu mēri ir parādīti ar piezīmēm (taisnstūri ar nolocītu stūri), kas ar raustītu līniju pievienotas mērāmajam objektam.
Saskaņā ar MOF [7.3] šāds biznesa procesa modelis ir visu faktisko biznesa procesu instanču abstrakcija, tātad atbilst M1 abstrakcijas slānim.



Att. 18 Picu pārdošanas biznesa process attēlots ar UML AD profilu (*M1*)

Make Order darbība izsauc biznesa apakšprocesu (aktivitāti), kas ir attēlota Att. 19.



Att. 19 Pasūtījuma izveidošanas apakšprocess

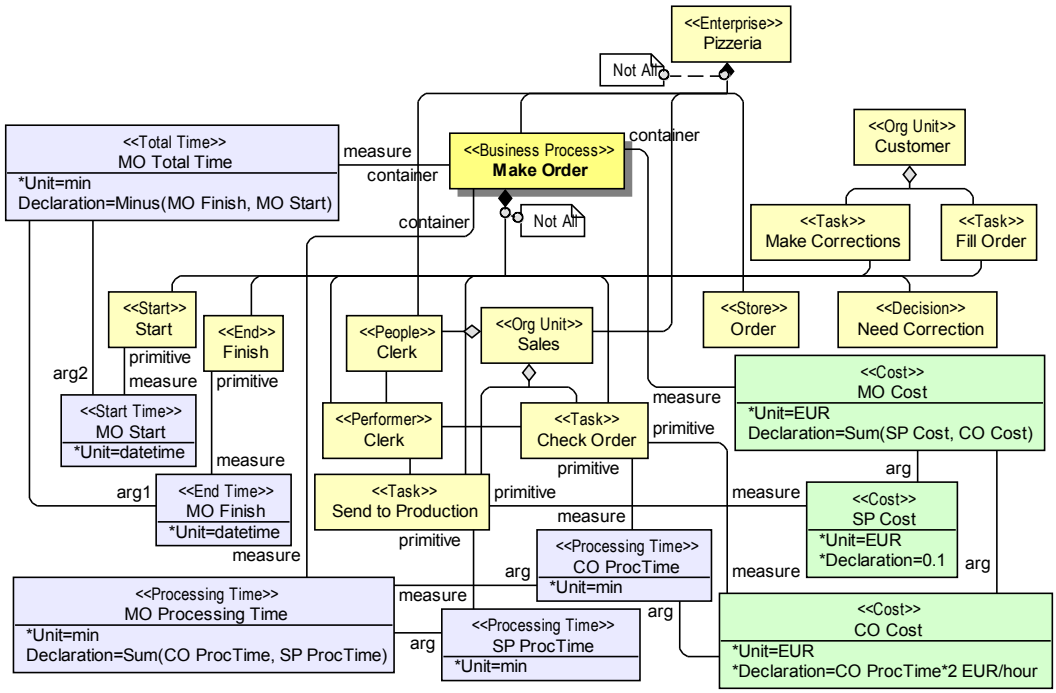
Lai AD nodrošinātu mēru definēšanas iespēju, tā saskaņā ar UML standartu ir paplašināta ar procesa mēru profilu. Procesa mēri faktiski ir metaklases stereotipa atribūti, kurus saskaņā ar UML standartu var attēlot piezīmēs. Attēlojuma uzskatāmībai vairāki viena objekta mēri ir apvienoti vienā piezīmē. Katra mēra deklarācija ir veidota ar sekojošu sintaksi: Veids[=deklarācija], mērvienība (piem., *Cost=2*hour, EUR*). Objektam pievienotais mērs nozīmē, ka procesa izpildes vai imitācijas laikā ir jānosaka dotā objekta parametra skaitliskā vērtība.

Šis piemērs parāda, ka procesu mēru definēšanai (mērīšanas specifikācijai) ir iespējams izmantot ar klašu stereotipiem paplašinātu UML aktivitāšu diagrammu, līdzīgi kā to dara esošajās imitācijas sistēmās (piem., ARIS [7.3]).

3.5 Mēru agregācijas modeļa piemērs

Ar UML standartam atbilstošu aktivitāšu diagrammas profilu var definēt biznesa procesa mērus, tomēr mēru apstrādei ar to vien nepietiek. Lai parādītu, kā tieši procesa izpildes laikā tiek apstrādātas mēru skaitliskās vērtības un kā tiek veiktas noklusētās darbības, ir nepieciešamas papildus klases un asociācijas. Tāpēc faktiski UML AD ir tikai ietvara ārējā specifikācija jeb saskarne, bet tā implementācija ir krietni sarežģītāka. Ietvara iekšienē aktivitāšu diagramma jātransformē uz īpašu "iekšējo modeli", kurā UML AD bez mēru profila tiek paplašināta ar jaunām klasēm un asociācijām (t.s. "smagsvara" paplašināšana). Tā kā MOLA [7.3] transformāciju valoda tajā laikā vēl nebija izstrādāta, šādu aktivitāšu diagrammas transformācija ir parādīta ar UML klašu diagrammas līdzekļiem, izmantojot īpašus sintakses paņēmienus.

Att. 20 diagrammā ir attēlota daļa no Att. 18 un Att. 19 definētā biznesa procesa kā ietvara mēru aprēķina modelis. Šis modelis ir "izpildes neatkarīgs", jo neatkarīgi no implementācijas parāda, kādi elementi, asociācijas un darbības ir nepieciešami, lai veiktu uzdoto mēru vērtību apstrādi. Formāli šāds modelis būtu jāattēlo ar divām diagrammām – ar klašu diagrammu, kura apraksta visas noteiktā procesa izpildes vai imitācijas laikā radušās instances (M0), un ar instanču diagrammu tālāk aprakstītajam ietvara metamodelim (M2, Att. 21). Tā kā abas diagrammas parādītu viena un tā paša objekta dažādas īpašības, attēlojuma ekonomijas dēļ tās ir apvienotas vienā klašu diagrammā, kur šādas duālas "instanču klases" ir parādītas ar īpašu sintaksi. Lai parādītu, ka kāda klase (M1) ir vispārīgākas klases (M2) instance, "instanču klases" ir attēlotas, izmantojot stereotipus. Vispārīgākās klases (M2) vārds ir attēlots kā konkrētās zemākās abstrakcijas slāņa (M1) "instances klases" stereotips.



Att. 20 Mēru agregācijas modeļa piemērs (*M1*)

Gaišās klases ir UML AD profila stereotipi, kam no metamodeļa "smagsvara" paplašinājuma parādās papildus agregācijas asociācijas (piem., *Sales* agregācija). Procesa mēri (dažāds tonējums attēlo dažādus mēru tipus), kas ar asociācijām *primitive/container-measure* ir pievienoti pie procesu elementiem, ir AD stereotipu atribūti, kas attēloti kā "instanču klase". Uzskatāmībai mēriem ir piešķirti vārdi, kas sastāv no saistītā elementa nosaukuma vārdu pirmajiem burtiem un mēra tipa

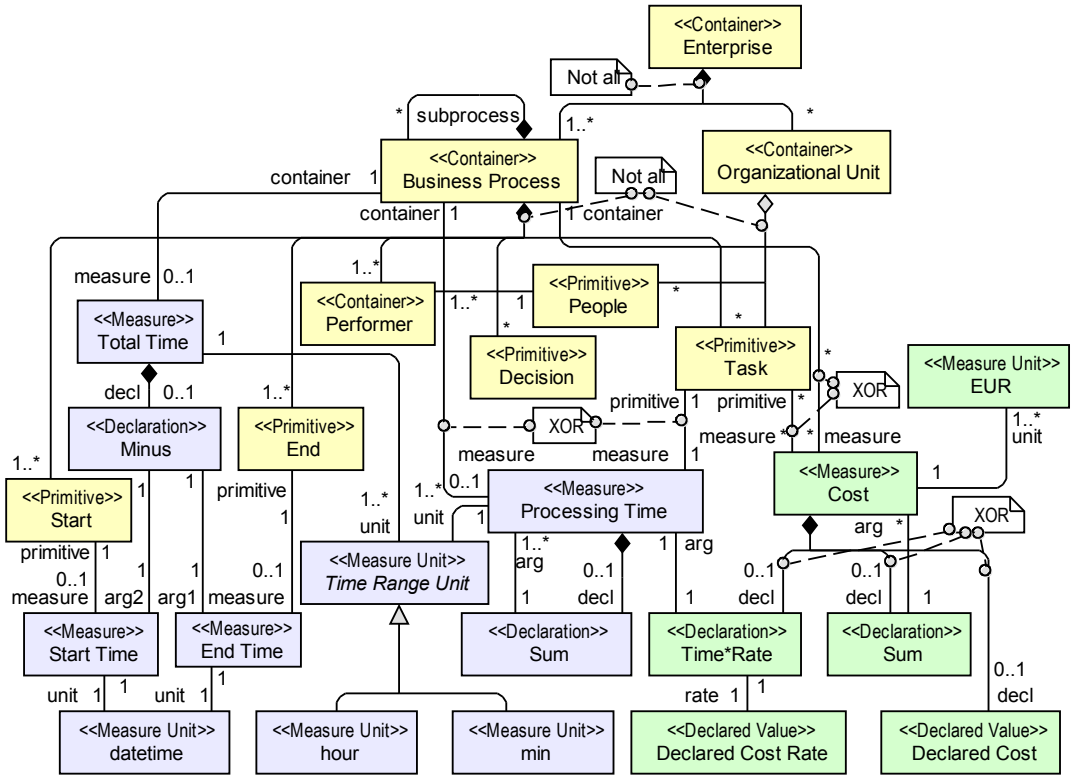
(apstrādei tos nevajag). Ja kāds mērs izmanto citu mēru, atbilstošā asociācija ir ar lomas vārdu *arg*.
No M2 slāņa mantotā procesa mēru kompozīcija M1 slānī ir parādīta saskaņā ar MOF tradīcijām, izmantojot klašu nodalījumus (*compartments*, piem., *unit=EUR*, *declaration= Minus*(*MO_Finish, _MO_Start*)). Tie nodalījumi, kas ir skaidri definēti biznesa procesa modelī (Att. 18), ir atzīmēti ar zvaigznīti. Atbilstošās mēru asociācijas un ar zvaigznīti neatzīmētie nodalījumi tiek mantoti netieši, veidojot mēru definīcijas metamodeļa instances. Piem., *MakeOrder* biznesa procesa *MOTotalTime* mēra deklarācija automātiski tiek noteikta kā starpība starp procesa *MOStart* un *MOEnd* mēru vērtībām ar atbilstošām asociācijām.
Lai modelis būtu lasāmāks, pārējām M2 slāņa klasēm kompozīcija arī M1 slānī ir attēlota kā kompozīcija. Piem., *Business Process* un *Enterprise* M2 kompozīcijas instances M1 ir parādītas kā kompozīcija ar atbilstošām klasēm un to stereotipiem.

3.6 Biznesa procesa metamodelis

Lai definētu procesa elementu mēru definēšanas un agregēšanas noteikumus, ir izstrādāts īpašs metamodelis (M2), kas kalpo kā ietvars visām metamodeļa instancēm jeb procesu definēšanas modeļiem (M1). Šis metamodelis ir attēlots Att. 21. Metamodelī ir definēta mēru savstarpēja saistība, un kā tie ir saistīti ar biznesa procesa elementiem. Arī šajā attēlā ir sapludinātas divas diagrammas – visu iespējamo modeļu (M1, Att. 20) klašu diagramma un konkrēta metametamodeļa (M3, Att. 22) instanču diagramma.
Šis biznesa procesu mēru metamodelis atbilst iepriekš izstrādātajam biznesa procesu metamodelim (. nodaļa), tomēr ir arī atšķirības:

- Visas klases ir papildinātā metametamodeļa (M3) instances.
- Lai arī gaišās klases atbilst UML AD profila stereotipiem (piem., *BusinessProces* kā *Activity* stereotips), bez UML definētajām asociācijām šīm klasēm ar papildināto metametamodeli (M3) ir iespējamas papildus agregācijas asociācijas.

UML definētās un papildus izveidotās agregācijas asociācijas tiek izmantotas, lai noteiktu, kuriem elementiem ir iespējama mēru agregācija. Tam varētu izmantot pilnīgi jaunas binārās asociācijas, bet, izmantojot esošās UML asociācijas, tiek samazināts papildus vajadzīgo asociāciju skaits, un metamodelis ir pārskatāmāks. Agregācijas (kompozīcijas) asociācija norāda, ka atbilstošai "saimnieka" elementa instancei iespējams veikt viena veida mēru agregāciju (piem., summa, vidējais, minimums, maksimums) pa visām tās "bērnu" instancēm.



Att. 21 Mēru definīcijas metamodeļa fragments (*M2*)

<<*Measure*>> klases (dažādi toņi norāda dažādus mēru tipus) ir klašu metaatribūti (formāli UML 2.0 īpašības – *properties*), kas iznesti kā atsevišķas klases ar asociācijām. Arī šī diagramma ir gan M1 slāni aprakstoša klašu diagramma, gan M3 slāņa instanču diagramma, tāpēc metaatribūti saskaņā ar ieviesto duālo notāciju ir attēloti kā klases. Šie metaatribūti parāda, kādi mēri var tikt definēti katram procesa elementam (gaišā klase).

Asociācijas starp stereotipiem un metaatribūtiem no papildinātā metametamodeļa manto īpašas *primitive/container-measure* lomas. Ja mēra deklarācija ir izteiksme (netieši mantota no metamodeļa vai arī konkrēti norādīta modelī), tā ir parādīta kā funkcijas argumentu kopa ar <<*Declaration*>> stereotipu.

Att. 21 ir parādīti daži iespējamie biznesa procesa elementu mēri. Piem., biznesa procesam (*Business Process*) vai uzdevumam (*Task*) izmaksas var noteikt kā konkrētu vērtību (*Declared Cost*), kā apstrādes laika un izmaksu koeficienta reizinājumu (*Processing Time, Declared Cost Rate*) vai arī kā summu vairākām izmaksām.

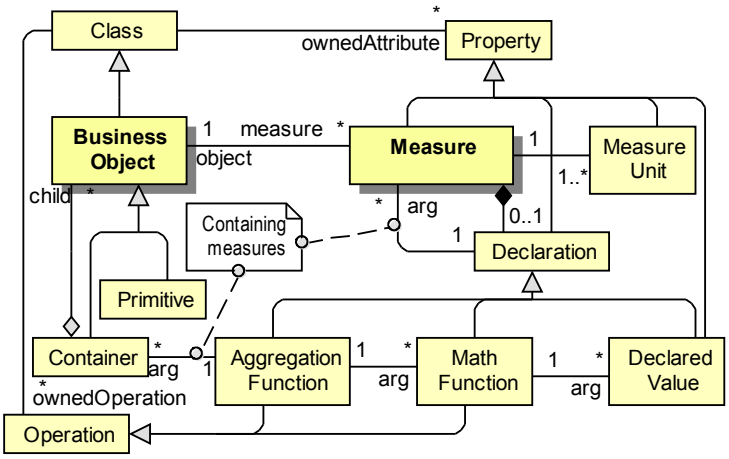
Apskatītajā piemērā mēru definēšanai ir izmantotas elementāras matemātikas darbības – summēšana, atņemšana, reizināšana. Lai ar šādu rīku efektīvi nodrošinātu arī statiskajā analīzē izmantojamo mēru definēšanu, ir nepieciešams atbalstīt arī statistiskajā analīzē izmantojamās funkcijas.

Izstrādātais metamodelis uzskatāmi ar klasēm un asociācijām parāda biznesa procesa modeļu elementu iespējamos mērus, kā arī to, kā mērus iespējams definēt un agregēt, lai mērījumu vērtībai būtu praktiska jēga. Uz šī metamodeļa bāzes veidojot konkrētus modeļus, definētās klases nosaka, kādas mēru instances var definēt procesa elementu instancēm, bet asociācijas nosaka mēru instanču atļautās saites un automātiskās apstrādes veidus.

3.7 Biznesa mēru metametamodelis

Saskaņā ar MOF tradīcijām metametamodelis (M3) tiek veidots pēc iespējas vienkāršāks, un visa konkrētās jomas specifika tiek atspoguļota metamodelī (M2).

Tomēr, lai ar vispārīgu metametamodeli (M3) paradītu konkrētā domēna metamodeļa (M2) nozīmi un specifiku, nepieciešams izmantot daudzus OCL ierobežojumus. Tāpēc tiek izmantota cita (pilnīgi legālu saskaņā ar MOF standartu) pieeja – UML metametamodeļa paplašināšana ar specializētām klasēm un asociācijām, kas saprotami parāda biznesa procesu mērus un to saistību.
Izstrādātais metametamodelis ir redzams Att. 22. Tā galvenā īpatnība ir jaunas biznesa objektu (*BusinessObject*) un mēru (*Measure*) metaklases, kas specializē *InfrastructureLibrary::Constructs* pakotnes *Class* metaklasi. Šāds specializēts metametamodelis atvieglo biznesa procesu mēra metamodeļa izveidošanu, jo, kā redzams Att. 21, pat neliels M2 fragments arī ar specializētu M3 ir pietiekami sarežģīts. Ar standarta UML MOF un OCL [7.3] ierobežojumiem M2 izveidot būtu daudz grūtāk, un tas būtu daudz nepārskatāmāks.



Att. 22 Biznesa procesu mēru metametamodelis (*M3*)

Izstrādātajā procesa mērīšanas paņēmienā ir parādīts, kā iespējams efektīvi izmantot divus UML paplašināšanas veidus. Biznesa procesu mēru definēšanai ir izmantota UML aktivitāšu diagramma, papildināta ar biznesa procesa mēru profilu, attiecīgos mērus parādot kā stereotipu atribūtus. Šādā veidā ir nodrošināta procesa mēru definīcijas savietojamība ar UML. Savukārt mēru apstrādes ietvars pārskatāmi un efektīvi ir izveidots, paplašinot UML metametamodeli. Tādējādi ir panākts, ka mēru ierobežojumus un to agregēšanas loģiku var definēt kā pārskatāmas klases un asociācijas starp mēriem un mērāmajiem elementiem procesa metamodelī.
Lai arī piedāvātais mērījumu definēšanas un apstrādes paņēmiens ir aprakstīts uz viena piemēra bāzes, tas ir pietiekami reprezentatīvs vispārīgās pieejas aprakstīšanai. Tas izriet no sekojošiem faktiem:

- Tā kā paņēmiena izstrādē tika izmantots vienots biznesa procesu metamodelis (skat. nodaļu 2.3), tas ir pietiekami pilnīgs, lai atbalstītu arī citas biznesa procesu modelēšanas valodas. Līdzīgi arī procesu mērīšanas metametamodelis ir izveidots pietiekami pilnīgs, lai nodrošinātu dažādas mērīšanas metodikas.
- Kā jau tas tika norādīts nodaļā 2.1, ja vien konkrētā metamodeļa "izteiksmes līdzekļi" ir pietiekami plaši, tad izmantojot metamodeļu "semantisko līdzību", uz viena metamodeļa bāzes var radīt dažādu modelēšanas valodu modeļus (lai arī ne vienlīdz ērti).

Tātad, lai arī izstrādātie metamodeļi ir parādīti kā konkrēti piemēri, tie ir pietiekami atbilstoši dažādām procesu modelēšanas valodām (EPC, QPR un GRAPES BM) un procesu mērīšanas metodikām (ARIS, QPR un GRADE). Papildus iespējas nodrošina tas, ka izstrādātajā paņēmienā piedāvātais metamodelis nav fiksēts un to ir iespējams papildināt, veidojot jaunas metametamodeļa instances.

Paši par sevi biznesa procesu mēri ir tikai definīcijas, kas nosaka, kā konkrēts biznesa procesa elements tiks mērīts. Faktiskās mēru vērtības ir zināmas tikai procesa izpildes laikā. Lai procesa definīciju izpildītu, ir nepieciešams noteikt precīzu mēru nozīmi neatkarīgi no procesa izpildes veida. Procesu vadības sistēmu izpilde ir detalizēti analizēta nākošajā pētījumu posmā, kas aprakstīts 4. nodaļā (gan nedetalizējot procesu mērīšanas aspektu).

4 UML 2.0 aktivitāšu diagrammas biznesa modelēšanas semantika kā virtuālā mašīna

2004. gadā UML 2.0 standarts [7.3] bija nobeiguma apstiprināšanas stadijā un tā aktivitāšu diagramma (AD) tika ieteikta arī biznesa procesu modelēšanai. Tāpēc precīza AD izpildes semantika biznesa procesos bija aktuāls temats.
Pētījuma mērķis bija padarīt AD lietojamu precīzu (t.i., izpildāmu) biznesa procesu modelēšanai. Lai to veiktu, bija nepieciešams noteikt precīzu AD izpildes semantiku un izvēlēties atbilstošu izmantojamo elementu apakškopu, kas būtu pietiekama biznesa procesu modelēšanai, modeļu validācijai, imitācijai un izpildei.
Pētījumā ir parādīts, ka UML standartā definētā oriģinālā AD semantika izpildāma biznesa modeļa definēšanai nav aprakstīta pietiekami detalizēti un ir ļoti sarežģīta. Tāpēc ir izstrādāta jauna pieeja, skaidrojot esošo AD semantiku ar aktivitāšu diagrammas virtuālo mašīnu (ADVM). Semantikas skaidrojumam ir izvēlēta minimāla AD elementu apakškopa, kas ir nepieciešama biznesa procesu definēšanai. UML AD formālais definīcijas modelis ir pārveidots uz vienkāršāku un daudz ērtāku izpildes modeli, kas tai pat laikā atbilst oriģinālajai AD izpildes semantikai izvēlētajā elementu apakškopā.
Šāda maksimāli pietuvināta pieeja oriģinālajai AD semantikai, lai arī nenodrošina formālu matemātiskās analīzes iespēju kā pilnīgi formālas algebriskās metodes (piem., Petri tīkli [7.3]), tomēr dod iespēju analizēt modeļa uzvedību. Papildus izveidotā virtuālā mašīna var kalpot par pamatu praktiskai imitācijas rīka vai darba plūsmas pārvaldības sistēmas būvei.

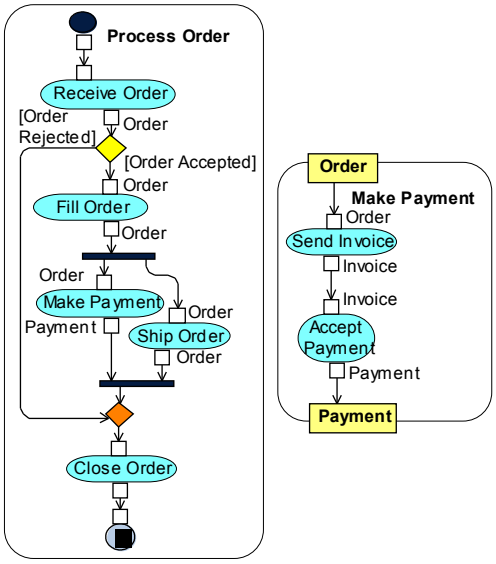
4.1 UML 2.0 aktivitāšu diagrammas apakškopa un ierobežojumi

Tā kā faktiski UML 2.0 aktivitāšu diagrammā ir ļoti daudz jēdzienu, kuru izpildes semantika ir optimizēta iegulto sistēmu projektēšanai, ir izvēlēti tikai tie, kas nepieciešami AD darbības skaidrošanai tieši biznesa modelēšanā. Biznesa procesu modelēšanai nepieciešamā minimālā jēdzienu kopa ir analizēta nodaļā 2.3, kā arī [7.3, 7.3, 7.3], un tā ir sekojoša:

- Iespēja aprakstīt procesa vadības secību – zarošanos, izvēles, dalīšanos, sapludināšanu un cilpas.
- Iespēja modelēt datu plūsmas un uz konkrētā procesa instancei atbilstošajiem datiem balstītas izvēles, cilpas un zarošanos.
- Iespēja definēt dažādus uzdevumu veidus:
  - biznesa procesa apakšprocesu izsaukumus;
  - automatizētus uzdevumus, kurus izpilda pati procesa vadības sistēma, vai izsaucot kādu ārēju automātisku servisu
  - manuālus uzdevums, kurus izpilda cilvēks, izmantojot kādu atsevišķu programmatūru (piem., MS Word), kam sistēma nodrošina vajadzīgo datu sagatavošanu šai programmatūrai; vai arī pilnībā manuālus uzdevumus (piem., preces iepakošana), tādā gadījumā nodrošinot, ka cilvēks sistēmai var paziņot, kad šis darbs ir izpildīts.
- Resursu pārvaldība – galvenokārt manuāli veicamo uzdevumu veicēju pārvaldībai.

Līdzīgi kā skaitļošanas teorijā pastāv "Tjūringa pilnā" mašīna, ar kuru var izskaitļot katru algoritmisku uzdevumu, šeit tiek izmantota "darba plūsmas pilnā" mašīna, ar kuru var aprakstīt katru biznesa procesu. Lai arī šādā mašīnā neparādās tādi augstāka līmeņa jēdzieni kā transakcijas un kļūdu, izņēmumu vai kompensācijas plūsmas, tas netraucē šādu procesu modelēt ar pieejamiem "izteiksmes līdzekļiem" un to izpildīt. Līdzīgi kā ar "Tjūringa pilno" mašīnu var izpildīt jebkuru algoritmisku procesu,

izvēlētajā "darba plūsmas pilnajā" mašīnā ir iespējams aprakstīt un izpildīt jebkuru biznesa procesu. Tomēr, netiek apgalvots, ka jebkura procesa gadījumā to var izdarīt eleganti, efektīvi vai vienkārši.
Kā jau tika aprakstīts nodaļā 2.1, AD uzvedība tiek aprakstīta, izmantojot Petri tīklu semantiku. Tas ir, AD modeļa darbību nosaka marķieri, kas plūst no virsotnes uz virsotni caur šķautnēm no aktivitātes sākuma uz beigu punktu. Dažādi marķieri tiek izmantoti arī datu pārsūtīšanai no vienas aktivitātes darbības uz citu. Tāpēc, nosakot nepieciešamos elementus, ir izvēlēti tikai tie, kas tieši ietekmē marķieru kustību. Praktiski tās ir darbības un kontroles virsotnes, kā arī šķautnes ar nosacījumiem, kas tās savieno.
Ir vairāki darbi, kuros ir izvērtēti tikai AD izpildes vadības plūsmas (*control flow*) aspekti [7.3,7.3,7.3]. Apskatot AD vadības plūsmu atsevišķi no datu plūsmas [7.3], ir secināts, ka AD nav "darba plūsmas pilna". Tomēr autora pētījumā ir praktiski pierādīts, ka AD datu un vadības plūsmu apskatot integrēti, AD ir pielietojama precīzai un formālai biznesa procesu modelēšanai.
Att. 23 ir parādītas divas aktivitāšu diagrammas, kas ilustrē visus izvēlētās apakškopas diagrammas elementus. Galvenais AD process (*Process Order*) izsauc apakšprocesu (*Make Payment*). Galvenais process sākas ar sākuma virsotni, tālāk process iet caur izvēles, sadalīšanās, darbības, apvienojuma un saplūšanas virsotnēm un beidzas beigu punktā. *MakePayment* darbība izsauc apakšaktivitāti, kas sākas un beidzas ar aktivitātes parametru virsotnēm. Visas šķautnes (vadības un objektu plūsmās), kas iet no darbības uz darbību, no sākuma uz beigu virsotnēm ir pievienotas piespraudēm (*pins*) – kā tas ir pieprasīts izvēlētajā apakškopā. Ieguvums, izmantojot piespraudes, ir tāds, ka šajā gadījumā ir labi zināma marķieru atrašanās vieta, kā tas ir arī Petri tīklos.



Att. 23 Aktivitāšu diagrammas piemērs, kur "Process order" aktivitāte izsauc "Make Payment" aktivitāti

Tā kā zarošanās diagrammās tiek parādīta atklāti ar izvēles un sadalīšanās virsotnēm, no piespraudēm drīkst iziet un tajās drīkst ieiet tikai viena šķautne. Izvēles virsotņu izejošo šķautņu nosacījumi ir savstarpēji izslēdzoši un laikā nemainās. Bez tam tiek aizliegti sekojoši (nestandarta) slēgumi:

- ❑Vadības virsotnes izejošā šķautne nevar būt tās pašas virsotnes ieejošā šķautne. Tas novērš vadības virsotņu strupsaķeres (*deadlocks*), kur vadības virsotne ieejā gaida marķieri, kas tai pašai ir jārada izejā.
- ❑Starp darbību izsaukuma, sākuma, beigu un aktivitāšu parametru virsotnēm nedrīkst būt ceļi, kas satur gan sadalīšanās, gan saplūšanas virsotnes. Šis

ierobežojums ir dabīgs no praktiskā viedokļa, jo nav vajadzības veidot paralēlus zarus, ja tie tiek vienkārši sapludināti bez jebkādām darbībām šajos zaros (t.i., zaros nav darbību izsaukuma virsotņu).

Šie ierobežojumi principā neierobežo dabīgu biznesa procesu veidošanu, bet to rezultātā starp dažādām darbībām nenotiek "cīņa par marķieriem" un ir izslēgta nedeterminēta diagrammas izpilde. Līdz ar to AD marķieru kustības likumi tiek būtiski vienkāršoti.

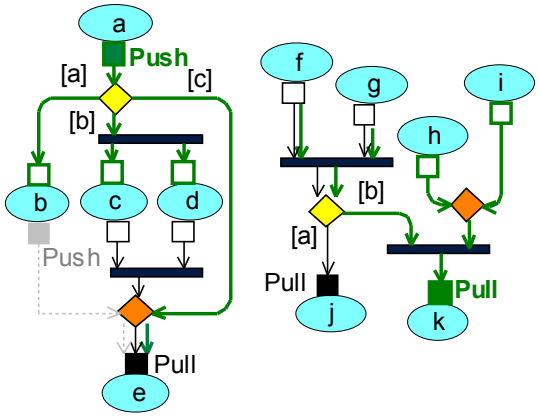
4.2 Vispārīgs UML 2.0 AD un izstrādātās VM apraksts

4.2.1 Aktivitāšu diagrammas standarta semantika

UML standartā aktivitāšu diagrammas izpildes semantika ir skaidrota ļoti sadalītā veidā, kur katrs AD elements pilda savu lomu [7.3,7.3,7.3]. Katra sadalīšanās, izvēles, apvienošanas un saplūšanas virsotne kopā ar šķautņu nosacījumiem nosaka, kuri no marķieriem tiek "piedāvāti" darbībām. Apkopojot atsevišķo kontroles elementu darbības aprakstu, marķieru "piedāvāšana" ir skaidrojama tā, ka vadības virsotnes marķierus padara darbībām "redzamus" un darbība tiek izpildīta tikai tad, "kad visām tā ieejas piespraudēm tiek piedāvāti marķieri, darbība tos saņem visus vienlaicīgi, neļaujot tos izmantot kādai citai darbībai" [7.3]. Praktiski tas nozīmē, ka darbības marķierus apstrādā ar vilkšanas paņēmienu un patiesi aktīvie diagrammas elementi ir "darbību dzinēji", kas cenšas savākt "redzamos" marķierus savās piespraudēs un izmantot tos definētajās darbībās. Skaidrs, ka ir iespējams uzbūvēt virtuālo mašīnu, kas precīzi atbilst UML AD standartam (ADVM), ar darbību dzinējiem kā aktīvajiem elementiem, kas pārvieto marķierus, un vadības virsotnēm, kas strādā kā "marķieru redzamības slēdži". Tomēr tas būtu daudz sarežģītāk, jo šajā gadījumā marķieru redzamības uzstādīšana būtu acīmredzami nelokāla (sadalīta) darbība ar daudziem atkarīgiem iesaistītajiem elementiem.

4.2.2 Izstrādātās virtuālās mašīnas apraksts

Pētījumā ir izstrādāts atšķirīgs AD virtuālās mašīnas variants, kurā vadības virsotnes ("nestabilās vietas", kurās marķieri nevar atrasties) un šķautnes tiek apvienotas ceļos. Ceļi tātad savieno virsotnes, kurās marķieri var atrasties ("stabilās vietas") – parasti ieejas un izejas piespraudes (darbības, sākuma un beigu virsotnēm), kā arī aktivitāšu parametru virsotnes (aktivitātēm). Katram ceļam ir nosacījums, kas tiek veidots no apvienotiem šķautņu ierobežojumiem šajā ceļā. Augšminētie diagrammas veidošanas ierobežojumi nodrošina to, ka ceļi ir savstarpēji izslēdzoši. Pat tad, ja kāda piespraude ir sākums vairākiem ceļiem, katram konkrētam marķierim ir atļauts tikai viens ceļš.

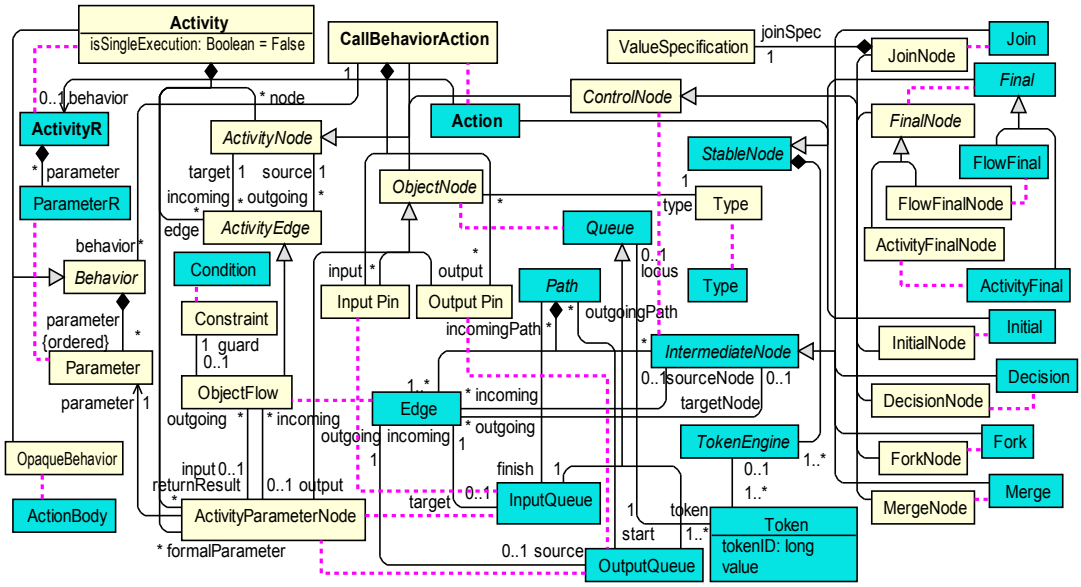


Att. 24 Velkošo un grūdošo ceļu un dzinēju izveide dažādiem slēgumiem

"Stabilās vietas" apkalpo aktīvie elementi – marķieru dzinēji. Ir ieviesti divi dzinēju veidi – velkošie (*pull*) un grūdošie (*push*) marķieru dzinēji, kā arī grūdošie un velkošie ceļi. Grūdošie ceļi satur tikai izvēles, apvienošanas un saplūšanas virsotnes vai vispār nesatur vadības virsotnes. Grūdošos ceļus apkalpo grūdošie marķieru dzinēji, kas tiek piesaistīti ceļa sākumam – izejas piespraudei vai aktivitātes ieejas parametram. Izvēlētajā apakškopā marķieri no izejošajām piespraudēm var tikt pārvietoti pa grūdošajiem ceļiem pilnīgi neatkarīgi viens no otra līdz pat mērķim, ja vien ceļa nosacījumi to pieļauj. Līdz ar to grūdošajos ceļos marķieru kustība ir vienkārša un acīmredzama.
Velkošie ceļi ir tādi, kuros ir kaut viena saplūšanas (*join*) virsotne, kā arī izvēles un apvienošanas virsotnes. Velkošos ceļus apstrādā velkošie dzinēji, kas tiek piesaistīti ceļa mērķim – ieejas piespraudei (vai aktivitātes izejas parametram). Saskaņā ar AD semantiku, marķieri ar kopīgu mērķi pa velkošajiem ceļiem ir jāsaskaņo – saplūšanas virsotni var šķērsot tikai saskaņota marķieru grupa.
Darbības dzinējs ir daudz vienkāršāks par tā oriģinālās semantikas līdzinieku. Tā uzdevums ir savākt visus marķierus no visām ieejas piespraudēm vai vienu marķieru grupu no "velkošās piespraudes", tos "izmantot" un piedāvāt izejas piespraudēs.
Galvenā izstrādātās semantikas atšķirība no oriģinālās ir tā, ka marķieri vai to grupas tiek pārvietoti pa ceļiem uz ieejas piespraudi neatkarīgi, kamēr oriģinālajā semantikā darbības dzinējs savāc visus marķierus no izejas piespraudēm "visus vienlaicīgi". Tomēr tas nerada būtiskas procesa izpildes izmaiņas, jo darbību izvēlētajā apakškopā darbību dzinēju "cīņa par marķieriem" nav iespējama.

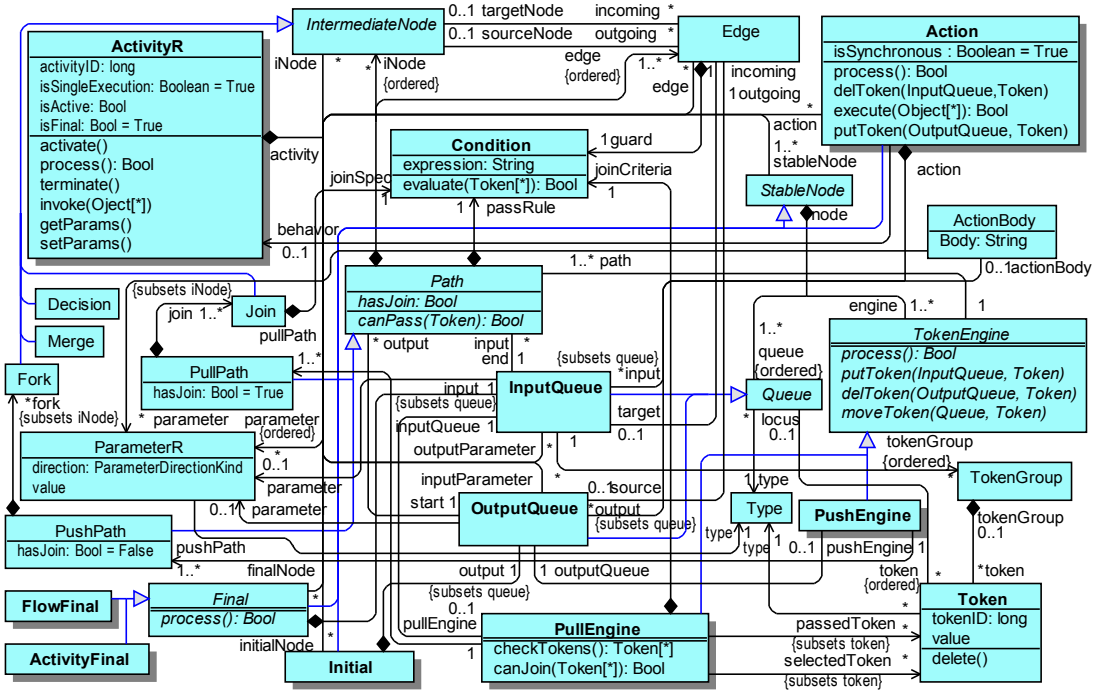
4.2.3 Metamodeļa paplašināšana un modeļu kartēšana

Lai formāli definētu ADVM, ir izveidots AD izpildes metamodelis, kurā ir visas nepieciešamās klases un operācijas. Katrai konkrētai aktivitāšu diagrammai izpildes modelis tiek radīts, veicot modeļa transformāciju.
Att. 25 parāda vienā diagrammā gan AD oriģinālās metamodeļa klases, gan izpildes metamodeļa (virtuālās mašīnas) klases – oriģinālās klases ir gaišas, izpildes klases ir tumšas. Kur vien iespējams, atbilstošās klases abos metamodeļos ir savienotas ar īpašām divvirzienu asociācijām (2.4. nodaļā minētās kartēšanas asociācijas, raustītās līnijas).



Att. 25 UML aktivitāšu diagrammas apakškopa un tās saistība ar izpildes klasēm

Brīdī, kad aktivitāte tiek izsaukta, tās instancei ar visām sastāvdaļām tiek izveidotas atbilstošas izpildes klašu instances. Respektīvi, izpildes instances darbojas kā virtuālā mašīna, kas izpilda doto aktivitāti. Att. 25 attēlo "vispārīgo transformācijas shēmu", kur kartēšanas asociācijām ir formāla transformācijas jēga. Virzienā no definīcijas klases uz izpildes klasi asociācija norāda, ka transformācijas procesā katrai definīcijas klases instancei ir jāizveido atbilstoša izpildes klases instance. Pretējā virzienā asociācija norāda, no kuras definīcijas klases instances izpildes klases instance ir izveidota. Šī informācija ir nepieciešama, lai jaunizveidotajām instancēm iestatītu specifiskas īpašības, kuras iespējams iegūt tikai no avota (definīcijas) modeļa.
Att. 26 ir parādīta izstrādātā virtuālā mašīna. Šī diagramma attēlo citu skatu uz to pašu metamodeli, kas attēlots Att. 25, tikai šajā diagrammā nav avota (definīcijas) klašu, savukārt izpildes klases parādītas detalizētāk. Diagrammā ir parādītas pilnīgi visas izpildes klases, to asociācijas un operācijas, kas ir nepieciešamas virtuālās mašīnas darbināšanai.



Att. 26. Aktivitāšu diagrammas virtuālās mašīnas metamodelis

Katra izpildes elementa izveides apraksts (faktiski transformācija) un tā darbības detalizēti ir parādīti publikācijā. Java pseidokods ir izmantots tur, kur ērtāka procedurāla pieeja, bet OCL nosacījumi – kur ērtāka deklaratīva pieeja. Kā tas ir aprakstīts 5. nodaļā, šādu pārveidi var definēt ar specializētu transformācijas valodu, piem., MOLA [7.3].
Tā kā izstrādātā virtuālā mašīna marķieru kustības līmenī darbojas citādāk kā oriģinālā, ir jāpierāda, ka oriģinālā un izstrādātā AD darbojas vienādi darbību izpildes secības līmenī. Darbā ir pierādīts, ka izstrādātajā virtuālajā mašīnā, lai arī atsevišķu marķieru kustības laiki var atšķirties, marķieru ceļi un marķieru izsaukto darbību sākums un beigas izvēlētajā elementu apakškopā pilnībā sakrīt. Pierādījums balstās uz sekojošo:

- ❑Ar izvēlēto elementu kopu un diagrammas izveides ierobežojumiem gan izstrādātā, gan oriģinālā AD mašīna strādā determinēti.
- ❑Lai arī izstrādātajā mašīnā atsevišķi marķieri var nonākt mērķa "stabilajā vietā" ātrāk kā oriģinālajā, tomēr pēdējais izstrādātās mašīnas marķieris vienmēr tiek

saņemts tajā pašā brīdi, kad oriģinālā AD mašīna savāc visus nepieciešamos marķierus.

Tātad aktivitātes darbību izpildes līmenī abas mašīnas darbojas vienādi, bet izstrādātā mašīna ir daudz vienkāršāka. (Detalizēts pierādījums dots publikācijā, kur tas aizņem veselu nodaļu.)

Ar piedāvāto virtuālo mašīnu iespējams implementēt arī plašāku aktivitāšu diagrammas elementu kopu. Tajā var izmantot visu pilnīgo (*Complete*) un daudzu pilnīgo strukturēto (*Complete structured*) aktivitāšu diagrammu elementus. Piemēram, tieši tādā pašā veidā iespējams darbināt diagrammas ar notikumiem un pārtraukuma apgabaliem. Nedaudz grūtāk ir realizēt konteksta atribūtu vērtības šķautņu pārejas nosacījumus (*Guards*). Lai tos realizētu precīzi kā standartā [7.3], nepieciešama nepārtraukta nosacījumu pārbaude, jo nosacījumos uzdotā konteksta vērtība laika gaitā var mainīties. Ja aktivitātes konteksts laika gaitā nemainās, un tā maiņa tiek modelēta norādot tieši kā notikumu, tad pārejas nosacījumi vienmēr izpildās vienādi, un tos var implementēt ar piedāvāto mašīnu. Datu masīvu (*CentralBuffer*) ar vienu izejošo šķautni var ieviest kā vēl vienu stabilo virsotni. Ja tiek pieļautas vairākas izejošās šķautnes, nepieciešama sarežģītāka pieeja, jo tad ir iespējama procesu cīņa par marķieriem, kas gan biznesa procesu diagrammām nav raksturīgi.

Esošās imitācijas vai biznesa procesu vadības sistēmās marķieru kustības apstrāde tiek veikta kopīgā virtuālajā mašīnā. Tām ir nepieciešams vienots dzinējs, kas veic konkrēta procesa izpildes uzskaiti un izsauc neatkarīgos procesa izpildes aģentus. Tas var kļūt par limitējošo faktoru reālu biznesa vadības sistēmu izstrādē, jo šāds kopīgs dzinējs var kļūt par sistēmas "šauro vietu".

Izstrādātās mašīnas galvenā priekšrocība ir tā, ka tā darbojas sadalīti, lai arī elementu skaits ir mazāks kā oriģinālajā standarta mašīnā. Katrs atsevišķais uzdevuma veikšanas solis ir pilnīgi neatkarīgs no pārējiem, un definēto biznesa procesu var izpildīt vairākās neatkarīgi sadarbojošās sistēmās, bez vienotas procesa izpildes uzskaites. Šāda procesu izpilde būtiski atvieglo procesu izpildes mērogojamību, jo sistēmas darbību nodrošina vairāki neatkarīgi aģenti.

Pseidokods un OCL nosacījumi uzskatāmi un precīzi parāda arī aktivitāšu diagrammas virtuālās mašīnas darbību (marķieru apstrādi). Šādā veidā uzbūvētā mašīna faktiski ir modeļu validācijas vai imitācijas sistēmas dizains. Darbā izmantotā pieeja var kalpot par pamatu arī darba plūsmas pārvaldības sistēmas izveidei.

5 UML un modeļu transformācijas biznesa procesu definēšanā

Modeļu bāzēta arhitektūra (*model driven architecture* – MDA), izmantojot modeļu bāzētu izstrādi (*model driven development* – MDD), ieņem arvien nozīmīgāku lomu arī biznesa vadības programmatūras izstrādē. Savukārt automātiska modeļu transformācija būtiski atvieglo sistēmu izstrādi ar MDD paņēmienu dažādos sistēmas izstrādes dzīves cikla posmos. Modeļu bāzētai biznesa vadības sistēmu izstrādei ir nepieciešama precīza biznesa modelēšanas valoda. Tā kā pasaulē nav "vienīgās labākās" biznesa modelēšanas valodas, modeļu bāzētā biznesa vadības programmatūras izstrādē bieži ir nepieciešama ātra modeļu transformācija no vienas valodas citā.

Pētījuma mērķis bija pierādīt, ka iespējams definēt precīzas un automātiski izpildāmas modeļu transformācijas, kas ļauj biznesa procesa modeli pārveidot no vienas modelēšanas valodas citā, nezaudējot modeļu semantiku.

Pētījumā ir apskatīti praksē nepieciešamie darba plūsmas modelēšanas aspekti, kas nepieciešami darba plūsmas modelēšanai un procesa vadības sistēmas izstrādei. Uz šo nosacījumu bāzes ir izveidots AD profils un tam atbilstoša BPMN apakškopa. AD profila un BPMN apakškopas izvēle ir balstīta uz iepriekšējos pētījumos veikto atziņu pamata (. un 3. nodaļas), valodu līdzekļus papildinot ar B2B (*business-to-business*) sadarbības līdzekļiem. Abu valodu semantika ir analizēta, pievēršot uzmanību datu plūsmai sadalītos biznesa procesos, procesu veicēju aprakstīšanai, ko iespējams kompilēt uz izpildāmu biznesa procesu modelēšanas valodu BPEL (*Business Process Execution Language*) [7.3].

Izstrādātais transformāciju paņēmiens ir ilustrēts, izmantojot divas izplatītas biznesa modelēšanas valodas – UML aktivitāšu diagrammu (AD) [7.3] un biznesa procesu modelēšanas notāciju (*Business Process Modeling Notation* – BPMN [7.3], abu valodu standartus šobrīd uztur OMG grupa). Transformācijas ir realizētas MOLA modeļu transformācijas valodā. Izstrādātā transformāciju pieeja nav ierobežota tikai ar aprakstītajām modelēšanas valodām. Līdzīgi iespējams veikt gan pretēju transformāciju, gan arī pavisam savādākas transformācijas, izmantojot citas modelēšanas valodas.

Kā jau tika minēts 2.4. nodaļā, šobrīd ar AD un BPMN valodu apvienošanu nodarbojas OMG domēna darba grupa (*Domain task force*) [7.3], kas valodu jēdzienus salāgo ar kartēšanas paņēmienu [7.3]. Tomēr, ņemot vērā valodu jēdzienu sarežģīto attiecību, pētījumā ir parādīts, ka jēdzienu salīdzināšana ar modeļu transformācijām ir efektīvāka un jēdzienu attiecību parāda precīzāk.

5.1 Darba plūsmas definēšanas valodas

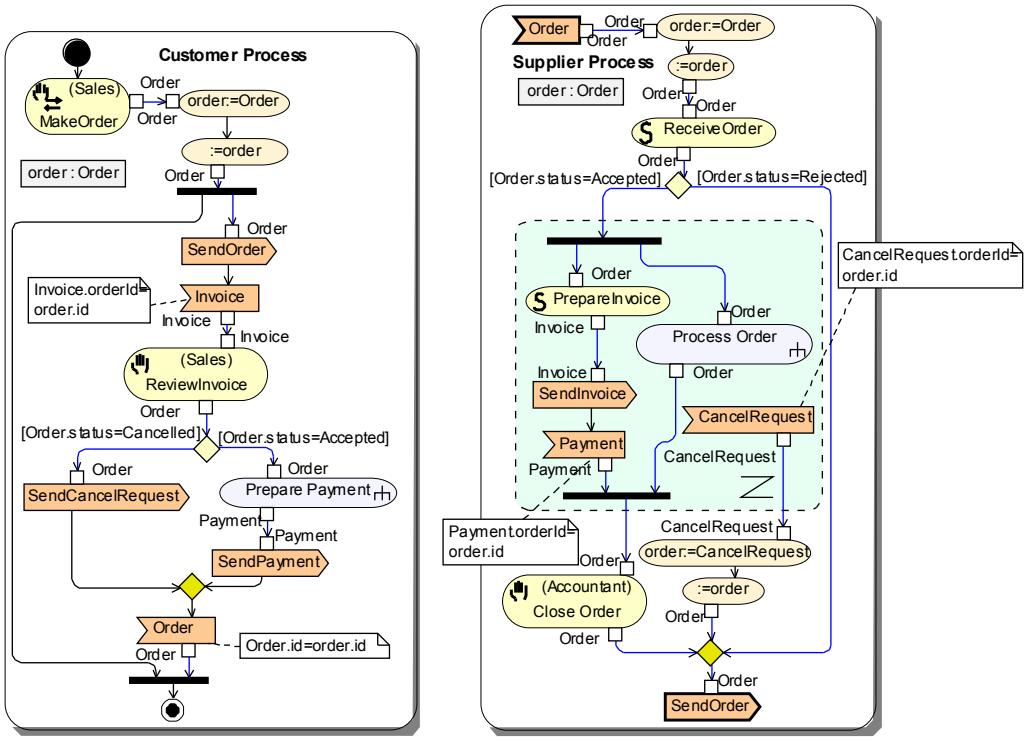
Darba plūsmas definēšanai ir nepieciešama viegli lasāma grafiska valoda ar precīzu izpildes semantiku [7.3,7.3]. No vairākām pētījumā apskatītajām modelēšanas valodām pieejas ilustrācijai ir izvēlētas UML AD un BPMN, jo tās vispilnīgāk apmierina darba plūsmas definēšanai nepieciešamos nosacījumus (skat. nod. 4.1). Tālāk ir analizēta šo valodu izpildes semantika, pievēršot uzmanību biznesa procesu specifikai un tam, kā valodu grafiskie līdzekļi to spēj attēlot.

Tā kā faktiski UML AD elementu semantika ir precizēta tā, ka labāk atbilst iegultajām sistēmām, biznesa procesu modelēšanai ir nepieciešama AD pielāgošana. Citiem vārdiem sakot, ir nepieciešams izveidot domēnam specifisku valodu (*Domain Specific Language* – DSL) kā AD profilu, kur UML ieviestie stereotipi novērstu pamanītās AD darba plūsmas definēšanas nepilnības. Kā transformācijas mērķa valoda ir izvēlēta BPMN. Arī tā ir guvusi zināmu rīku atbalstu [7.3]. Tomēr arī šai valodai ir

savas nepilnības, īpaši neformālā semantika un nepietiekams atbalsts datu struktūru veidošanai. Tāpēc īsumā ir apskatītas arī BPMN izmantošanas iespējas, un ir izvēlēta sadalītu biznesa procesu modelēšanai nepieciešamā apakškopa.
Lai precīzi noteiktu abu valodu izpildes semantiku sadalītos biznesa procesos, tās ir kartētas uz pagaidām vienīgo praktiski izpildāmo biznesa procesu modelēšanas valodu BPEL. Tajās vietās, kur standartā noteiktie valodas izteiksmes līdzekļi ir nepietiekami, tā ir paplašināta ar valodu implementējošo firmu *de-facto* rūpnieciski izmantotiem līdzekļiem. Tā kā pētījumā precīzi ir analizēti tikai ar procesu modeļu izpildi saistītie jēdzieni, konceptuālajā modelēšana satopami jēdzieni nav apskatīti. Tomēr to ieviešana principā nemaina piedāvātā paņēmiena izmantošanu.

5.2 UML aktivitāšu diagrammas pielāgošana darba plūsmas modelēšanai

Lai arī ir zināmi vairāki AD profili darba plūsmas definēšanai [7.3,7.3,7.3], neviens no tiem nenodrošina visas pētījumā izvirzītās darba plūsmas modelēšanai nepieciešamās prasības – sadalītu procesu sadarbību ar ziņojumiem, datu apstrādi, manuālu procesu veicēju aprakstīšanu un modeļa izpildes iespēju.
Tāpēc pētījumā ir veikta jauna AD profila izstrāde. No AD elementiem ir izvēlēti tie, kas nepieciešami darba plūsmas modelēšanai, saglabājot oriģinālo AD elementu nozīmi un ieviešot stereotipus tur, kur tiem nepieciešamas papildu īpašības vai arī elementu nozīmi ir nepieciešams precizēt sadalītas darba plūsmas vajadzībām.
Att. 27 parādītais biznesa process ar divām aktivitātēm ilustrē izstrādāto AD profilu. (Šī un citas diagrammas ir zīmētas GMF (*Generic Modeling Framework*) jeb EBM (*Exigen Business Modeler*) rīkā [7.3], kurā ir izveidoti papildus redaktori UML AD profilam un BPMN valodai). Diagrammā ir attēloti praktiski visi izvēlētie darba plūsmas definēšanas elementi. Lielākā daļa elementu ir izvēlēti jau no iepriekšējiem pētījumiem (., 3., un 4. nodaļa), pievienojot tikai tos elementus, kas raksturīgi B2B sadarbībai.



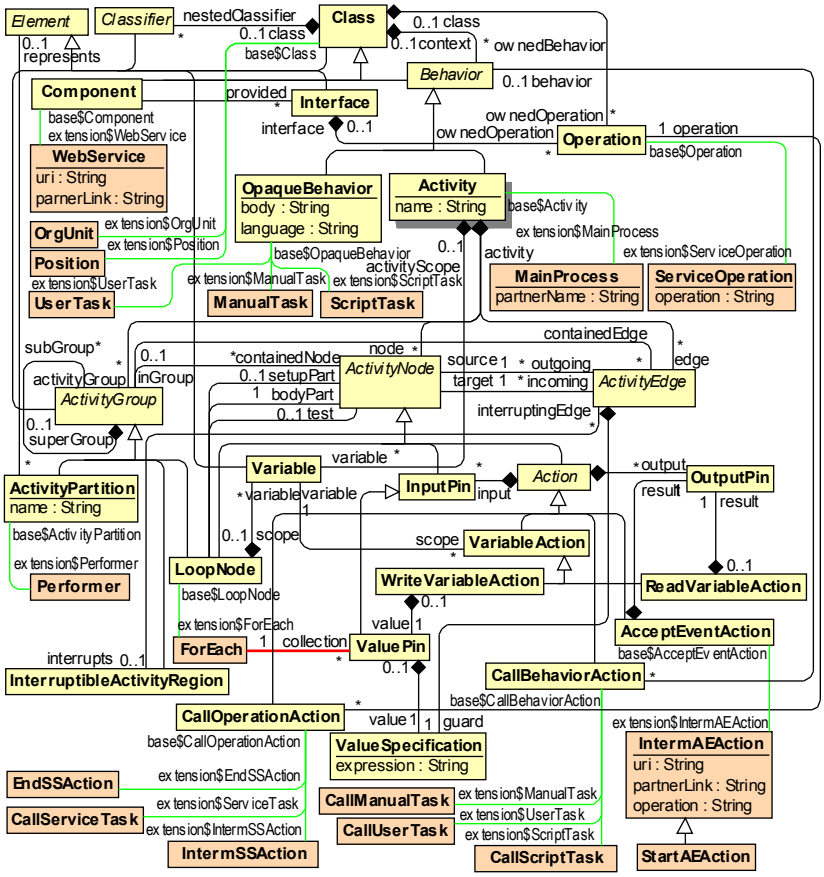
Att. 27. Procesa piemērs UML aktivitāšu diagrammās

Jaunie elementi ir signāla nosūtīšanas (*SendSignal*) un notikuma saņemšanas (*AcceptEvent*) darbības (attiecīgi apzīmētas ar ieliektu un izliektu karodziņu), kā arī stereotipi dažādiem darbību tipiem. Elements `order : Order` ir mainīgā `order` definīcija ar tipu `Order`, kura redzamības apgabals ir aktivitāte `Supplier Process`. Mainīgo operāciju darbības ir parādītas kā standarta OCL operācijas.

Notikuma saņemšanas darbība ar nosacījumu izejošajai šķautnei ir īpaša konstrukcija, kas nepieciešama procesa instances noteikšanai un kas ir ekvivalenta BPEL korelācijas kopai.

Ar stereotipiem ir precizēta esošo AD elementu nozīme sadalīta biznesa procesa modelēšanas vajadzībām. Att. 28 attēlotais metamodelis parāda AD ieviestos stereotipus:

- *Galvenais process* (*MainProcess*) ir aktivitātes stereotips, kas norāda, ka aktivitāte ir atsevišķs darba plūsmas process, ko izpilda atsevišķs darba plūsmas dzinējs. Grafiski tas parādīts kā aktivitāte ar ēnu.
- *Veicējs* (*Performer*) ir sadaļas (*Partition*) stereotips, kas norāda manuāli veicamas darbības veicēju. Tā `represents` asociācija norāda uz klasi ar *Position* vai *OrgUnit* stereotipu. Veicējs ir parādīts kā darbības nodalījums (*compartment*).
- *WebServiss* (*WebService*) ir komponentes (*Component*) stereotips, kas tiek izmantots, lai norādītu web servisu atribūtus. *CallServiceTask* darbības, kas izsauc dotā web servisa operācijas, iegūst nepieciešamos parametrus no šī stereotipa.
- *Starpposma signāla nosūtīšanas darbība* (*IntermSSAction*) nozīmē vienkāršu signāla nosūtīšanu web servisam (parādīts kā izliekts karodziņš), kamēr *beigu signāla nosūtīšanas darbība* (*EndSSAction*) papildus signāla nosūtīšanai nozīmē arī aktivitātes beigas (izliekts karodziņš ar ēnu).
- *Katram* (*ForEach*) ir stereotips cilpas virsotnei (*LoopeNode*). Papildu `ForEach.collection` asociācija uz `ValuePin` ir ieviesta iteratoram (iteratora nepieciešamību biznesa modelēšanas valodās norāda arī [7.3]). Tā kā papildus asociāciju nevar ieviest ar stereotipu, šī ir vienīgā "spēcīgās paplašināšanas" vieta, lai maksimāli saglabātu metamodeļa savietojamību.

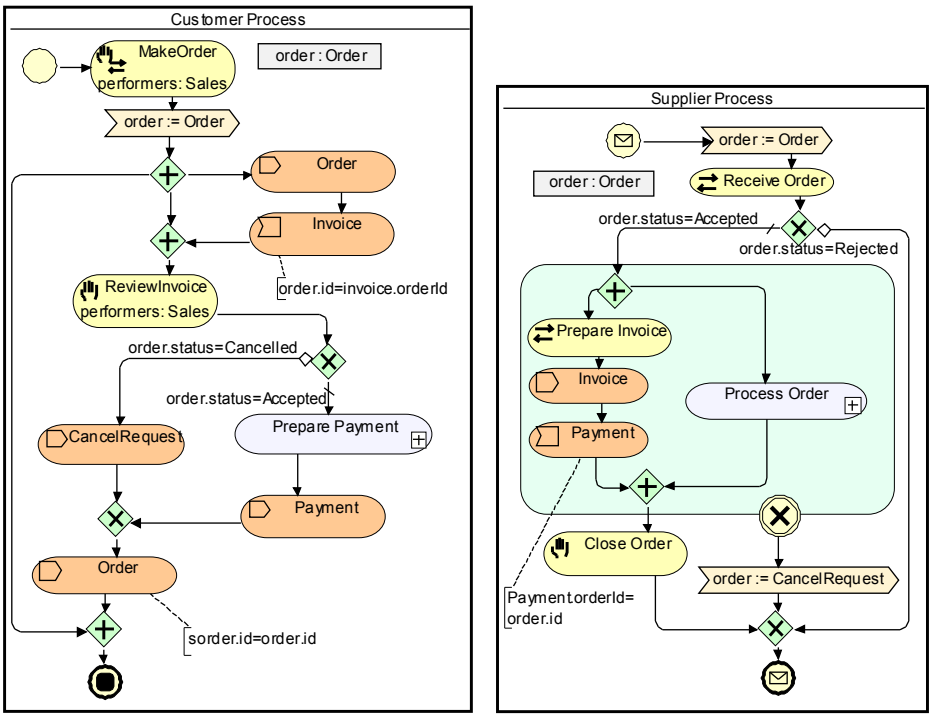


Att. 28. AD metamodeļa fragments (*transformācijas avots*)

Att. 28 ir parādīts izstrādātā UML AD metamodeļa fragments, kas ir "saplacināts", jo tajā ir izmestas nevajadzīgās abstraktās virsklases, un kam ir "uzlikts" izveidotais darba plūsmas profils. Saskaņā ar MOF standartu, profilētās klases un to stereotipi tiek attēloti ar asociācijām `extension$Position` <-> `base$Class`. Šī metamodeļa instances ir 5.4 nodaļā aprakstīto transformāciju piemēru avots.

5.3 BPMN diagramma kā otra valoda

Kā jau norādīts 5. nodaļas sākumā, arī BPMN ir izplatīta darba plūsmas modelēšanas valoda. Tā kā arī BPMN valodai ir lieki (raugoties no "darba plūsmas pilnības" viedokļa) elementi un defekti, ir nepieciešams izvēlēties BPMN apakškopu. Apakškopā ir iekļautas visu veidu slūžas (*Gateways*, attēloti kā rombi), visu veidu uzdevumi (*Tasks*) un apakšprocesi (*Subprocesses*, attēloti kā noapaļoti taisnstūri), sākuma (*Start*), beigu (*End*) un tie starpposmu (*Intermediate*) notikumi (attēloti kā apļi), kas ir definēti uz aktivitātes robežas ("pārtraukuma konstrukcija"), jo šiem elementiem ir dabīgs kartējums uz BPEL valodu.
Att. 29 ir parādīts biznesa process izvēlētajā BPMN valodas apakškopā. Abi procesi ir precīzi Att. 27 procesu analogi, kas ir iegūti, izpildot MOLA valodas transformācijas GMF rīkā un izmantojot rīka automātiskās grafiskās izvietošanas iespējas.

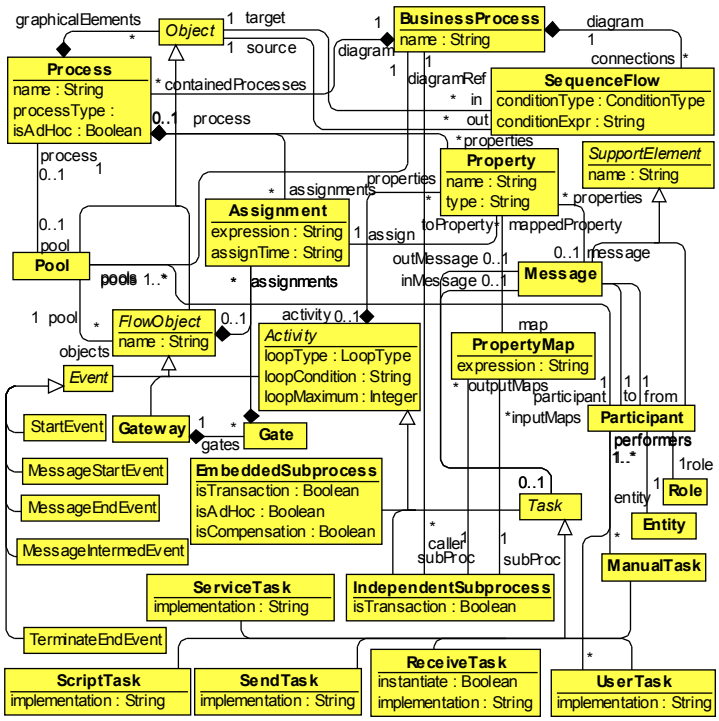


Att. 29. Procesa piemērs BPMN valodā

Ziņojumu nosūtīšanai un saņemšanai "nepārtraukumu" gadījumos tiek izmantoti nosūtīšanas un saņemšanas uzdevumi (noapaļoti taisnstūri ar karodziņiem iekšpusē). Lai arī BPMN metamodelis ir pieņemams, tomēr vairākiem vajadzīgajiem elementiem nav grafiskās prezentācijas. Grafiskā prezentācija ir ieviesta sekojošiem elementiem:

- ❑Uzdevumu tipiem tiek izmantoti stereotipi ar ikonām uzdevuma iekšpusē, uzdevuma nodalījumā ir minēti uzdevumu veicēji.
- ❑Īpašības (*properties*) ir parādītas kā čertstūri formā `vārds:tips`.
- ❑Lai padarītu redzamu datu apstrādi, piešķiršanas (*assignments*) ir parādītas ar lielām bultām un tekstuālu izteiksmi tajās.

Līdzīgi kā UML AD Att. 30 parāda BPMN fragmentu ar izvēlēto elementu apakškopu. Šo elementu instances parādās modeļa transformācijas rezultātā.

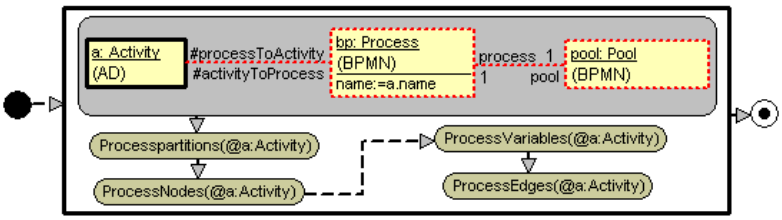


Att. 30. BPMN metamodeļa fragments (*transformācijas mērķis*)

5.4 AD transformācija uz BPMN

Lai arī vairāki darbi ir veltīti formālām biznesa modeļu transformācijām [7.3,7.3], netika atrasts darbs, kurā būtu aprakstīta AD transformācija uz BPMN, iekļaujot visus ar darba plūsmu definēšanu saistītos aspektus. Tāpēc ir izmantotas esošās AD kartēšanas shēmas uz BPMN [7.3,7.3,7.3,7.3,7.3,7.3], papildinot tās ar datu plūsmām, mainīgajiem, piešķiršanām un uzdevumu veicējiem.

Lai detalizētu jēdzienu kartējumu, ir parādīti arī da~i fragmenti no modeļu transformācijām, kas izveido BPMN modeli no AD modeļa. Transformācijas ir rakstītas MOLA modeļu transformācijas valodā [7.3]. Att. 28 un Att. 30 parāda attiecīgi transformāciju avota un mērķa metamodeļus, bet Att. 31 ilustrē vienu no transformācijām. ` ajā transformācijā katra aktivitāte tiek transformēta BPMN procesā (*Process*) un pūlā (*Pool*), un tiek izsauktas transformāciju apakaprogrammas, kas apstrādā pārējos aktivitātes elementus.



Att. 31. Transformācijas piemērs no AD uz BPMN MOLA valodā

MOLA transformāciju programma pilnīgi precīzi apraksta atbilstību starp aktivitāšu diagrammas un BPMN elementiem. Kā redzams Att. 31, aādai atbilstībai ir nepieciešams izmantot vairākas klases un konteksta nosacījumus, kas ir pietiekami sarežģīti, lai tos nevarētu viennozīmīgi aprakstīt tikai ar kartējuma asociācijām. Modeļu transformācija ir labākais veids kā definēt sarežģītu atbilstību starp divām dažādām valodām, ņemot vērā visas ar valodu nozīmi saistītās īpatnības.

Gadījumā, ja nepieciešams aprakstīt "darba plūsmas bagātākus" procesus, piemēram, BPMN transakcijas, kompensācijas plūsmas, aktivitāšu cilpas, pieeja principā

nemainās, jo no transformāciju viedokļa, tie ir tikai papildus jēdzieni, kas tiek apstrādāti līdzīgā veidā.
Transformācija ir realizēta GMF rīkā, izstrādājot jaunus AD un BMPN valodu redaktorus, un izmantojot MOLA valodas transformācijas vidi ar MySQL datu bāzi [7.3]. Realizācija pierādīja paņēmiena efektivitāti pētnieciskiem nolūkiem – nelieli modeļi ar vairākiem desmitiem klašu tiek transformēti dažu sekunžu laikā uz standarta personālā datora. Uz šo pētījumu pamata šobrīd tiek izstrādāta jauna MOLA transformāciju valodas izpildes vide ar operatīvajā atmiņā glabājamu datu repozitoriju, kas ir izmantojams transformāciju izpildei rūpnieciskos mērogos.

6 Noslēgums

Šo pētījumu mērķis bija novērtēt un izmantot metamodelēšanas paņēmienus precīzu (izpildāmu) biznesa procesu definēšanā. Dažas autora izstrādātās biznesa procesu modelēšanas idejas jau ir ieviestas esošajos modelēšanas rīkos, bet citas var izmantot turpmākai modelēšanas rīku attīstībai. Turpmāk dots autora ideju vērtējums jaunāko biznesa procesa modelēšanas notikumu kontekstā:

- Pētījuma sākumā izstrādātais "notācijas neatkarīgais" biznesa procesu metamodelis parāda biznesa modelēšanas jēdzienus un to saistību un ir izmantots kā visu turpmāko pētījumu bāze. Izvirzītās idejas sasaucas ar OMG vēlāk izstrādātajiem Biznesa motivācijas modeļa [7.3] un Procesu definīcijas standarta [7.3] projektiem. Tomēr autora izstrādātajā biznesa procesu metamodelī procesa resursu, ieeju un izeju, kā arī uzdevuma veicēju apraksts ir detalizētāks. Uz nedaudz modificēta biznesa procesu metamodeļa pamata autors ir izveidojis UML AD un BPMN modelēšanas valodu redaktorus GMF rīkā [7.3].
- Autora izvirzītā ideja biznesa modeļu jēdzienu kartēšanā no viena domēna uz vairākām prezentācijām ir izmantota GMF rīkā, kurā viena domēna modeli iespējams parādīt vairākās līdzīgās modelēšanas valodās. Ideja parādīt līdzīgas modelēšanas valodas kā "kanoniskās formas" skatus pēdējā laikā ir atspoguļojusies arī OMG grupas aktivitātēs, kas vienotā domēnā mēģina apvienot UML AD un BPMN modelēšanas valodas [7.3].
- Darba gaitā izstrādātie MOLA [7.3] valodas transformāciju piemēri biznesa modelēšanas vajadzībām ir devuši lielu ieguldījumu MOLA valodas un rīka validācijā, testēšanā un demonstrēšanā. MOLA rīkā realizētā ideja, transformējot biznesa modeļus no vienas modelēšanas valodas uz citu, ļauj vairākiem lietotājiem izmantot ērtāko modelēšanas valodu dažādās biznesa vadības sistēmu izstrādes fāzēs.
- LU MII šobrīd tiek izstrādāta jaunas paaudzes uz metamodeļiem un modeļu transformācijām balstīta rīku platforma TTF (*Transformation Tool Framework* [7.3]), kurā modeļu transformācijas tiek būvētas MOLA valodā. Kā viens no šīs jaunās paaudzes rīku platformas pielietojumiem ir dažādu specializētu modelēšanas rīku būve domēna specifiskām (DSL) valodām. Piemēram, jaunās paaudzes rīku platformā būs iespējams būvēt dažādu biznesa modelēšanas rīku redaktorus. Autora izstrādāto UML AD profilu un BPMN valodas apakškopu būs iespējams izmantot par pamatu redaktoru izstrādei šajā jaunās paaudzes modelēšanas rīku platformā. Savukārt autora izstrādāto aktivitāšu diagrammas virtuālo mašīnu un procesa mēru apstrādes satvaru būs iespējams izmantot modeļu imitācijas dzinēja izveidei. Tādējādi tiks nodrošināta visa biznesa procesu modelēšanai un imitācijai nepieciešamā funkcionalitāte.

7 Atsauces

7.1 Autora publikācijas recenzētos starptautisku konferenču materiālos

1.Vitolins Valdis, Audris Kalnins. ***Modeling Business***. Modeling and Simulation of Business Systems, Kaunas University of Technology Press, edited by H. Pranevicius, E. Zavadskas, B. Rapp, Vilnius, May 13-14, 2003, pp. 215.-220.
2.Vitolins Valdis, ***Business Process Measures***. Computer Science and Information Technologies, Databases and Information Systems Doctoral Consortium, University of Latvia Scientific Papers Vol. 673, edited by R. Freivalds, University of Latvia, 2004, pp. 186.-197.
3.Valdis Vitolins, Audris Kalnins, ***Semantics of UML 2.0 Activity Diagram for Business Modeling by Means of Virtual Machine***, Proceedings Ninth IEEE International EDOC Enterprise Computing Conference, edited by Danielle C. Martin, IEEE, 2005, pp. 181.-192.
4.Audris Kalnins, Valdis Vitolins, ***Use of UML and Model Transformations for Workflow Process Definitions***, Databases and Information Systems, BalticDB&IS'2006, edited by O. Vasilecas, J. Eder, A. Caplinskas, Vilnius, Technika, 2006, pp. 3.-15.

7.2 Citas autora publikācijas (tieši nesaistītas ar promocijas darba tēmu)

5.Vītoliņš Valdis, ***Darba plūsmas pārvaldība***, e-pasaule (2001) Nr. 5, 6
6.Vītoliņš Valdis, ***Microsoft dokumentu pārvaldības rīks SharePoint***, e-pasaule (2001), Nr. 7
7.Vītoliņš Valdis, ***IT projektu pārvaldība***, e-pasaule (2001), Nr. 9
8.Vītoliņš Valdis, ***Zināšanu pārvaldības sistēmas***, e-pasaule (2002), Nr. 9
9.Vītoliņš Valdis, ***Organizācijas modelis biznesa procesu automatizācijā***, e-pasaule (2002), Nr. 10
10.Vītoliņš Valdis. ***Biznesa modelēšanas rīki***, e-pasaule, (2003), Nr. 9.

7.3 Citi darbā izmatotie avoti

11.***Unified Modeling Language: Superstructure version 2.0***, OMG, 2005, http://www.omg.org/cgi-bin/doc?formal/05-07-04
12.***Unified Modeling Language (UML) Specification: Infrastructure, Version 2.0***, OMG, 2005, http://www.omg.org/cgi-bin/doc?formal/05-07-05
13.***OCL 2.0 Specification, Version 2.0***, OMG, 2005, http://www.omg.org/cgi-bin/doc?ptc/05-06-06
14.***MOF2 Versioning Final Adopted Specification***, OMG, 2006, http://www.omg.org/cgi-bin/doc?formal/2006-01-01
15.***Business Process Modeling Notation (BPMN), Final Adopted Specification***, OMG, 2006, http://www.omg.org/cgi-bin/doc?dtc/2006-02-01
16.***Business Process Execution Language for Web Services***, Version 1.0, IBM, 31 July 2002 ftp://www6.software.ibm.com/software/developer/library/ws-bpel1.pdf
17.Grady Booch, James Rumbaugh, ***The Unified Modeling Language User Guide***, Addison-Wesley Longman, 1999
18.***Business Process Specification Schema v1.01***, Business Process Team, 11 May 2001
19.Osterwalder, S. Ben Lagha, Y. Pigneur, ***An Ontology for Developing e-Business Models***, INFORGE, Encole des HEC, 2002
20.***ESPRIT project ADDE***, http://www.fast.de/ADDE
21.***The Zachman Institute for Framework Advancement (ZIFA)***, http://www.zifa.com
22.***System Architect,Tutorial***, 2001, Popkin Software, http://www.popkin.com/products/product_overview.htm
23.Andreas Dietzsch. ***Adapting the UML to Business Modelling's Needs - Experiences in Situational Method Engineering***, UML 2002. LNCS 2460, pp 73-83
24.International Standard, ***ISO/DIS 9001 Quality management systems — Requirements***, ***International Organization for Standardization***, http://www.iso.com
25.Sandy Tyndale-Biscoe, Business ***Modelling for Component Systems with UML***. Proceedings of the Sixth EDOC Conference (2002)
26.Osterwalder, ***An e-Business Model Ontology for the Creation of New Management Software Tools and IS Requirement Engineering***, Ecole des Hec, Université de Lausanne, 2002
27.***Business Process Modeling Notation. Working Draft (0.9)*** November 13, 2002, Business Process Management Initiative (BPMI), http://www.bpmi.org

28.Mark C. Paulk, et al., ***The Capability Maturity Model : Guidelines for Improving the Software Process***, Addison Wesley Professional, 1995
29.Douglas T. Hicks, Activity***-Based Costing : Making It Work for Small and Mid-Sized Companies***, 2nd Edition, John Wiley & Sons, 2002
30.Robert S. Kaplan, David P. Norton, ***The Strategy-Focused Organization: How Balanced Scorecard Companies Thrive in the New Business Environment***, Harvard Business School Press, 2000
31.Farok J.Contractor,Valuation ***of Intangible Assets in Global Operations***, Quorum Books, 2001
32.***ARIS 6 – Collaborative Suite***, System White Paper, IDS Scheer, 2003, http://www.ids-scheer.com/sixcms/media.php/1186/ARIS+6-2+SWP+en+2003-07.pdf
33.***System Architect, Tutorial***, 2001, Popkin Software, http://www.popkin.com/products/product_overview.htm
34.***QPR Process Guide White Paper***, QPR ScoreCard White Paper, 2002, QPR Software Plc, http://www.qpr.com/protected/whitepapers/QPR_ScoreCard_WhitePaper.pdf
35.***Casewise Corporate Modeler Product Info***, Casewise, http://www.casewise.com/products/corporate-modeler/corporate-modeler.php
36.Stephen A. White, ***Process Modeling Notations and Workflow Patterns***, March, 2004, http://www.omg.org/bp-corner/pmn.htm
37.R. Eshuis, R. Wieringa, ***Comparing Petri Net and Activity Diagram Variants for Workflow Modelling - A Quest for Reactive Petri Nets***, Lecture Notes in Computer Science: Petri Net Technology for Communication-Based Systems: Advances in Petri Nets, Volume 2472. 2003, Heidelberg, Germany: Springer-Verlag. pp. 321 - 351.
38.Harald Störrle, Jan Hendrik Hausmann, ***Towards a Formal Semantics of UML 2.0 Activities***, 2004, http://www.pst.informatik.uni-muenchen.de/~stoerrle/V/AD-11-Limits.pdf
39.Harald Störrle, ***Semantics and Verification of Data Flow in UML 2.0 Activities***, 2004, http://www.pst.informatik.uni-muenchen.de/~stoerrle/V/AD2b-DataFlow.pdf
40.***Rational Software Architect***, http://www-306.ibm.com/software/awdtools/architect/swarchitect/
41.Conrad Bock, ***UML 2 Activity and Action Models Part 4: Object Nodes***, in Journal of Object Technology, vol. 3, no. 1, pp. 27-41. http://www.jot.fm/issues/issue_2004_01/column3
42.Conrad Bock, ***UML 2 Activity and Action Models Part 2: Actions***, in Journal of Object Technology, vol. 2, no. 5, pp. 41-56. http://www.jot.fm/issues/issue_2003_09/column4
43.Tom Baeyens, ***The State of Workflow***, May 2004, http://www.theserverside.com/articles/content/Workflow/article.html
44.Derek Miers, Paul Harmon, ***The 2005 BPM Suites Report, Version 1.0***, March 15, 2005, http://www.bptrends.com/reports_toc_01.cfm
45.Curtis Hall, Paul Harmon, ***The 2005 Enterprise Architecture,Process Modeling & Simulation Tools Report***, Version 1.0, April 28, 2005, http://www.bptrends.com/reports_toc_02.cfm
46.Petia Wohed, et al., ***Pattern-based Analysis of UML Activity Diagrams***, 2004, http://is.tm.tue.nl/research/patterns/download/uml2patterns BETA TR.pdf
47.***WebSphere Business Integration Modeler***, IBM, http://www-306.ibm.com/software/integration/wbimodeler/
48.***Business Process Definition Metamodel, Revised Submission***, Object Management Group, 2004, http://www.omg.org/docs/bei/04-01-02.pdf
49.***Business Motivation Model (BMM) Specification, Adopted Specification***, Object Management Group, 2004, http://www.omg.org/docs/dtc/06-08-03.pdf
50.Behzad Bordbar, Athanasios Staikopoulos, ***On behavioural model transformation in Web services***, 5th International Workshop on Conceptual Modeling Approaches for e-Business eCOMO'2004, November 8-12, 2004
51.Jean Bezivin, et al., ***Applying MDA Approach to B2B Applications: A Road Map***, Work-shop on Model Driven Development (WMDD 2004) at ECOOP 2004, Springer-Verlag, LNCS, vol. 3344, June 2004
52.Tracy Gardner, ***UML Modelling of Automated Business Processes with a Mapping to BPEL4WS***, IBM, 2004, http://www-128.ibm.com/developerworks/rational/library/4593.html
53.Stephen A. White, ***Using BPMN to Model a BPEL Process***, BPTrends, March 2005
54.Andrew Watson, ***OMG's new modeling specifications***, ECMDA-FA 2005, Nuremberg, Germany, November 7-10, 2005, keynote speech
55.Armin Haller, Eyal Oren, Paavo Kotinurmi, ***An Ontology for Internal and External Business Processes***, WWW 2006, May 23–26, 2006, Edinburgh, Scotland.
56.***Web Services Business Process Execution Language Version 2.0, Primer Initial Draft***, 13th September, 2006
57.***BPELJ: BPEL for Java technology***, IBM Developerworks, 2004, http://www-128.ibm.com/developerworks/library/specification/ws-bpelj/

58.***Oracle BPEL Process Manager***, http://www.oracle.com/technology/products/ias/bpel/index.html
59.***Draft UML 1.4 Profile for Automated Business Processes with a Mapping to BPEL 1.0***, IBM, 2004, http://www-128.ibm.com/developerworks/rational/library/4593.html
60.***The Emerging Technologies Toolkit (ETTK)*** http://www.alphaworks.ibm.com/tech/ettk
61.***MagicDraw UML 10.5*** http://www.magicdraw.com/
62.***BPMI, Current Implementations of BPMN***, http://www.bpmn.org/BPMN_Supporters.htm#current
63.Mike Havey, ***Essential Business Process Modeling***, O'Reilly, 2005, ISBN: 0-596-00843-0
64.Ken Beck, Joshy Joseph, Germán Goldszmidt, ***Learn business process modeling basics for the analyst***, IBM Developerworks, 2005, http://www-128.ibm.com/developerworks/library/ws-bpm4analyst/
65.Alexandre Alves, ***BPEL4WS 1.1 To WS-BPEL 2.0 - An SOA Migration Path***, 2005, http://webservices.sys-con.com/read/155617_1.htm
66.Wil van der Aalst, ***Workflow Patterns***, http://is.tm.tue.nl/research/patterns/
67.Petia Wohed, et al., ***Pattern-based Analysis of UML Activity Diagrams, BETA Working Paper Series, WP 129***, Eindhoven University of Technology, Eindhoven, 2004
68.Petia Wohed, et al., ***Pattern-based Analysis of BPMN - an extensive evaluation of the Control-flow, the Data and the Resource Perspectives***, BPM Center Report BPM-05-26, BPMcenter.org, 2005
69.***Business Modeling & Integration Domain Task Force***, http://bmi.omg.org/
70.Bill Moore, et al., Using ***BPEL Processes in WebSphere Business Integration Server Foundation Business Process Integration and Supply Chain Solutions***, IBM, 2004
71.Martin Keen et al., ***BPEL4WS Business Processes with WebSphere Business Integration: Understanding, Modeling, Migrating***, IBM, 2004
72.Peter Swithinbank et al., ***Patterns: Model-Driven Development Using IBM Rational Software Architect***, IBM, 2005
73.David Skogan, Roy Grønmo, Ida Solheim, ***Web Service Composition in UML***, The 8th International IEEE Enterprise Distributed Object Computing Conference (EDOC), Monterey, Ca. Sept 2004
74.Roy Grønmo, Michael C. Jaeger, ***Model-Driven Semantic Web Service Composition***, 12th Asia-Pacific Software Engineering Conference (APSEC), Taipei, Taiwan. December 2005
75.Chun Ouyang, et al., Translating Standard Process Models to BPEL, BPM Center Report BPM-05-27, BPM, 2005
76.***Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification***, OMG, 2005, http://www.omg.org/docs/ptc/05-11-01.pdf
77.J.Bārzdiņš, J.Tenteris, Ē.Viļums, B***iznesa modelēšanas valoda GRAPES-BM 4.0 un tās lietošana***, Rīga, 1998
78.Celms E., A. Kalnins, L. Lace. ***Diagram definition facilities based on metamodel mappings***. OOPSLA'2003 (Workshop on Domain-Specific Modeling), Anaheim, California, USA, October 2003, pp. 23-32.
79.***MOLA Modeling language***, Institute of Mathematics and Computer Science University of Latvia, http://mola.mii.lu.lv/
80.A. Kalnins, J. Barzdins, E.Celms. ***Efficiency Problems in MOLA Implementation***. 19th International Conference, OOPSLA'2004, Vancouver, Canada, October 2004
81.Kalnins, J. Barzdins, E. Celms. ***Model Transformation Language MOLA***. Proceed-ings of MDAFA 2004 (Model-Driven Architecture: Foundations and Applications 2004), Linkoeping, Sweden, June 10-11, 2004. pp.14-28.
82.***ProGuide***, http://www.proformacorp.com/
83.***xBML***, http://www.xbmlinnovations.com/
84.Satish Thatte, ***XLANG Web Services for Business Process Design***, Microsoft Corporation, 2001
85.A. Ledeczi et al., ***The Generic Modeling Environment***, Workshop on Intelligent Signal Processing, Budapest, Hungary, May 17, 2001
86.L.Lace, E.Celms, A.Kalnins. ***Diagram definition facilities in a generic modeling tool***, 2003,- Proceedings of International Conference on Modelling and Simulation of Business Systems, Vilnius, 2003, pp. 220-224.
87.A. Kalnins, J. Barzdins, ***MDA Support by Transformation Based Tool,*** Proceedings of First International Workshop MoRSe 2006, Warsaw, Poland, October 2006, pp. 21-24.
88.A. Kalnins, E. Celms, A. Sostaks, ***Simple and Efficient Implementation of Pattern Matching in MOLA Tool***, Proceedings of the 7th International Baltic Conference on Databases and Information Systems (Baltic DB&IS'2006) , Vilnius, Lithuania, July 3-6, 2006, pp. 159-167.
89. Jorn Bettin. ***Model-Driven Software Development: An emerging paradigm for industrialized software asset development***. 2004. http://www.softmetaware.com/mdsd-and-isad.pdf.
90. Mili, H., Pachet, F., Benyahia, I., and Eddy, F. ***Metamodeling in OO***. 1995. OOPSLA '95 Workshop summary.

91. ARIS Expert Paper, ***Business Process Design as the Basis for Compliance Management***, Enterprise Architecture and Business Rules, March 2007, http://is.tm.tue.nl/staff/wvdaalst/publications/p74.pdf
92.Sandy Tyndale-Biscoe. ***Business Modelling for Component Systems with UML***. Proceedings of the Sixth EDOC Conference ('02)
93.A.Kalnins, J.Barzdins et al., ***Business Modeling Language GRAPES-BM and Related CASE Tools***, Proceedings of Baltic DB&IS'96, Institute of Cybernetics, Tallinn, 1996, pp. 3-16
94.D. Harel. ***Statecharts: A visual formalism for complex systems***. Science of Computer Programming, 8(3):231--274, June 2002.
95.Gianfranco Balbo et al., ***Petri Nets 2000, Introductory Tutorial***, 21st International Conference on Application and Theory of Petri Nets, Aarhus, Denmark, June 26-30, 2000
96.http://www.idef.com/IDEF3.html
97.http://en.wikipedia.org/wiki/Value_chain
98.http://www.tibco.com/software/business_process_management/iprocess_suite/default.jsp
99.http://en.wikipedia.org/wiki/Six_Sigma
100.G. Linde, ***Semantics and Equivalence of UML Class Diagrams***, Scientific Papers, University of Latvia, Vol. 669, Computer Science and Information Technologies, p. 107-116., Riga, 2004
101.http://modeling.telelogic.com/products/statemate/index.cfm
102.***OWL Web Ontology Language Guide, W3C Recommendation***, 10 February 2004, http://www.w3.org/TR/owl-guide/
103.http://en.wikipedia.org/wiki/Backus-Naur_form
104.***GRADE Business Modeling, Simulation Guide***,Infologistik GmbH, 1998
105.***GRADE Business Modeling, Language Reference***,Infologistik GmbH, 1998

8 Pielikums

8.1 Autora referāti par darba rezultātiem starptautiskās zinātniskās konferencēs vai semināros

1.Vitolins Valdis, Audris Kalnins. Modeling Business, Modeling and simulation of business systems, May 13-14, 2003, Vilnius, Lithuania.
2.Vitolins Valdis, Business Process Measures, 6th International Baltic Conference on Databases and Information Systems 2004, June 6-9, 2004, Riga, Latvia.
3.Valdis Vitolins, Audris Kalnins, Semantics of UML 2.0 Activity Diagram for Business Modeling by Means of Virtual Machine, Proceedings Ninth IEEE International EDOC Enterprise Computing Conference, 19-23 September 2005, Enschede, The Netherlands.
4.Audris Kalnins, Valdis Vitolins, Use of UML and Model Transformations for Workflow Process Definitions, Databases and Information Systems, BalticDB&IS'2006, 7th International Baltic Conference on Databases and Information Systems, July, 3-6 2006, Vilnius, Lithuania

8.2 Promocijas darbā iekļautās publikācijas un promocijas darba autora personiskais ieguldījums

Autori	Publikācija	Ieguldī-jums %	Promocijas darba autora ieguldījuma apraksts
Valdis Vītoliņš Audris Kalniņš	Modeling Business	90	•Esošo modeļu apkopošana un analīze •Vienotā metamodeļa izstrāde •Jēdzienu kartēšanas idejas izstrāde
Valdis Vītoliņš	Business Process Measures	100	•Esošo procesu mērīšanas metodiku analīze •Uz metamodeli balstīta mērīšanas paņēmiena izstrāde •Mēru metamodeļa un metametamodeļa izstrāde
Valdis Vītoliņš Audris Kalniņš	Semantics of UML 2.0 Activity Diagram for Business Modeling by Means of Virtual Machine	90	•UML aktivitāšu diagrammas darbības analīze un virtuālās mašīnas izveide •Vienkāršotās virtuālās mašīnas izveide, marķieru dzinēju darbības izstrāde
Valdis Vītoliņš Audris Kalniņš	Use of UML and Model Transformations for Workflow Process Definitions	90	•UML aktivitāšu diagrammas analīze, BPMN valodas metamodeļa izstrāde, metamodeļu kartējuma izstrāde •UML AD un BPMN valodu redaktoru izstrāde •Transformāciju izstrāde no AD uz BPMN