RĪGAS TEHNISKĀ UNIVERSITĀTE

A. Lektauers

Integrētas pieejas izstrāde diskrētu notikumu un nepārtrauktu sistēmu imitācijas modelēšanai un vizualizācijai

PROMOCIJAS DARBS

2008

RĪGAS TEHNISKĀ UNIVERSITĀTE
Datorzinātnes un informācijas tehnoloģijas fakultāte
Informācijas tehnoloģijas institūts

Arnis LEKTAUERS
Doktora studiju programmas "Vadības informācijas tehnoloģija" doktorants

INTEGRĒTAS PIEEJAS IZSTRĀDE DISKRĒTU NOTIKUMU UN NEPĀRTRAUKTU SISTĒMU IMITĀCIJAS MODELĒŠANAI UN VIZUALIZĀCIJAI

Promocijas darbs

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

Rīga 2008

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

ANOTĀCIJA

Datorgrafikas un vizualizācijas metodes aizvien plašāk tiek izmantotas imitācijas modelēšanas jomā, līdztekus imitācijas modelēšanas rezultātu prezentēšanas iespēju nodrošināšanai ļaujot paātrināt imitācijas modeļu verifikācijas un validācijas uzdevumu izpildi. Straujā datorgrafikas tehnoloģiju attīstība rada iespējas izveidot arvien sarežģītākus un reālistiskākus modelējamo sistēmu vizuālos attēlojumus, tajā pašā laikā izvirzot papildus prasības esošajiem imitācijas modelēšanas risinājumiem un radot nepieciešamību pēc jaunām pieejām un metodēm imitācijas modelēšanas un vizualizācijas integrācijā.

Promocijas darbā ir aprakstīti imitācijas modelēšanas un vizualizācijas integrācijas pētījumu rezultāti, kas iegūti, risinot tādas aktuālas problēmas, kā lietotāja interaktivitātes nodrošināšana imitācijas gaitā, kā arī imitācijas un vizualizācijas sinhronizācijas un sadarbspējas nodrošināšana kombinētu diskrētu notikumu un nepārtrauktu sistēmu imitācijas modelēšanā. Darbā ir analizēts un pamatots uz sistēmpieeju balstītās diskrētu notikumu sistēmu specifikācijas (DEVS) lietojums un priekšrocības, kā arī ir veikta vizualizācijas pieeju un metožu salīdzinošā analīze to integrācijai imitācijas modelēšanā, identificējot esošo vizuālo imitācijas modelēšanas sistēmu trūkumus un definējot prasības integrētai vizuālai interaktīvai imitācijas modelēšanas videi. Pamatojoties uz veiktajiem pētījumiem, darba ietvaros ir izstrādāts jauns sistēmteorētisks formālisms integrētai vizuālai diskrētu notikumu un nepārtrauktu sistēmu imitācijas modelēšanai, kas īstenots programmatūras prototipa veidā. Darbā ir izklāstītas izstrādātās programmatūras sistēmas pamatā esošās koncepcijas, realizācijas detaļas un eksperimentālās pārbaudes rezultāti. Pētījumu rezultātu praktiskā nozīme un pielietojuma iespējas ir parādītas ar automatizētas ražošanas sistēmas imitācijas modeļa izstrādi un veiktajiem imitācijas eksperimentiem. Par promocijas darba galvenajiem rezultātiem ir nolasīti referāti 8 starptautiskās konferencēs, kā arī tie ir atspoguļoti 7 zinātniskajās publikācijās.

Darbs sastāv no ievada, 4 nodaļām un secinājumiem. Tajā ir 127 lappuses, 66 attēli un 16 tabulas pamattekstā, 4 pielikumi un 123 nosaukumu literatūras saraksts.

ABSTRACT

The computer graphics and visualization methods are increasingly used in the field of simulation to provide presentation possibilities of the simulation results and to speed up the execution of verification and validation tasks for the simulation models. The rapid progress in computer graphics technologies enables the development of ever more complicated and realistic visual depictions of the modelled systems, at the same time additionally postulating the existing simulation solutions and creating the necessity for new approaches and methods in the integration of simulation and visualization.

This thesis describes the research results of simulation and visualization integration that are obtained through solving such topical problems as providing a user interactivity during the simulation execution, as well as simulation and visualization synchronization and ensuring interoperability during the simulation of combined discrete event and continuous systems. The work provides an analysis and justification of usage and advantages of the system approach based discrete event system specification (DEVS) with the purpose to determine major drawbacks of the existing visual simulation systems and to determine general requirements to an integrated visual interactive discrete event and continuous systems simulation environment. A comparative analysis of visualization approaches and methods for their integration in the simulation is performed as well. Based on the performed research, a new system theoretical formalism for an integrated visual simulation of discrete event and continuous systems is developed within the framework of this work that is realized in the form of a software prototype. The underlying concepts of the developed software system are laid out; implementation details and evaluation results are discussed. The practical importance and application possibilities of research results are demonstrated by developing a simulation model of an automated manufacturing system and executed simulation experiments. The main results of the thesis have been presented at 8 international conferences, as well as they are reflected in 7 scientific papers.

The thesis includes introduction, 4 chapters and conclusions. It contains 127 pages, 66 figures and 16 tables in the main text, 4 appendices and 123 titles in the bibliography.

SATURS

IEVADS

Imitācijas modelēšana ir uzskatāma par vienu no senākajām datorzinātnes nozarēm, jo tās vēsture ir pielīdzināma datortehnoloģiju attīstības laika kategorijām, kad 20.gadsimta 40.gadu sākumā parādījās pasaulē pirmās elektroniskās skaitļojamās mašīnas jeb mūsdienu terminoloģijā - datori. Un praktiski vienlaicīgi radās nepieciešamība ar datora palīdzību risināt skaitliskās modelēšanas uzdevumus, kur būtisks sasniegums ir 1940.gadu beigās izstrādātā Monte-Karlo metode [87], kas līdz šim laikam joprojām ir stohastisku sistēmu imitācijas modelēšanas pamatprincips. Kopš tā laika ir notikusi ārkārtīgi strauja informācijas tehnoloģiju attīstība, ieskaitot arī imitācijas modelēšanu, kuras attīstības gaitā ir izstrādātas neskaitāmas imitācijas modelēšanas metodes, valodas un sistēmas, ļaujot modelēt gan diskrētas, gan nepārtrauktas sistēmas un procesus. Kā īpaši nozīmīgs etaps imitācijas modelēšanas vēsturē ir uzskatāma 1960.gados kompānijas IBM izstrādātā diskrēto notikumu imitācijas modelēšanas valoda GPSS, kas kardināli mainīja priekšstatu par imitācijas modelēšanas nozīmi un pielietošanas iespējas sistēmu analīzē.

Imitācijas modelēšana kā esošas vai teorētiskas fiziskas sistēmas modeļu izveides, izpildes un analīzes disciplīna [25] plaši tiek izmantota visdažādākajās praktiskajās un zinātnes jomās, un šeit ir izceļamas galvenās un svarīgākās imitācijas modelēšanas pielietošanas priekšrocības:

- reālistisku jeb modelējamajai sistēmai maksimāli pietuvinātu modeļu iespējamība;
- iespējas izmēģināt un pārbaudīt dažādus modeļu variantus bez nepieciešamības veikt tiešus eksperimentus ar modelējamo sistēmu;
- veiktspējas mērījumu iespējas līdzīgi reālām sistēmām;
- iespējas modelēt neeksistējošas sistēmas;
- imitācijas rezultātu attēlošana vizuālā veidā kā modeļu izstrādes un validācijas palīglīdzeklis;
- nav nepieciešams pielietot sarežģītu matemātisko aparātu;
- pielietošanas iespējas gadījumos, kad neeksistē atbilstoši analītiski risinājumi, vai arī tie ir pieejami tikai vienkāršotā formā.

Pēdējā laikā arvien vairāk imitācijas modelēšanas sistēmās tiek izmantotas datorgrafikas un vizualizācijas metodes, nodrošinot pāreju no vēsturiski tradicionālām teksta orientētajām imitācijas modelēšanas izstrādes sistēmām uz vizuālām imitācijas modelēšanas vidēm, kurās ar grafiskiem līdzekļiem tiek balstīta imitācijas modeļu izveide un tiek attēloti imitācijas modelēšanas rezultāti pēcmodelēšanas laikā vai arī tieši eksperimentu izpildes gaitā. Sākotnēji vizuālajās imitācijas

modelēšanas vidēs bija iespējama tikai modelēšanas rezultātu pēcimitācijas laika attēlošana bez lietotāja mijiedarbības iespējām. Pašlaik eksistē daudzi pētījumi par vizuālām imitācijas modelēšanas metodēm, kas nodrošina gan modelēšanu, gan eksperimentēšanu vizuālā interaktīvā vidē. Kamēr vizualizācijas rīki ir paredzēti labākai izpratnes gūšanai par modeļa izejas datiem, interaktivitāte ļauj lietotājam vadīt un kontrolēt imitācijas eksperimentus, tādējādi ietekmējot un novērojot modeļa uzvedību tā izpildes laikā.

Tēmas aktualitāte

Promocijas darba tēmas aktualitāte ir saistīta ar pēdējos gados vērojamajiem centieniem padarīt imitācijas modelēšanas sistēmas pieejamākas un draudzīgākas imitācijas modeļu izstrādātājiem un lēmumu pieņēmējiem, jo imitācijas modelēšana līdz šim pārsvarā tiek izmantota tikai šaurā profesionāļu lokā, kuriem jābūt ne tikai ar padziļinātām zināšanām tajā pielietojuma sfērā, kurā tiek izstrādāts imitācijas modelis, bet ir labi jāorientējas arī programmēšanā, varbūtību teorijā un matemātiskajā statistikā. Par nozīmīgu un perspektīvu virzienu imitācijas modelēšanas attīstībā ir uzskatāma datorgrafikas un vizualizācijas metožu pielietošana un integrācija imitācijas modelēšanas sistēmās [41], jo vizualizācija, kalpojot kā primārais līdzeklis imitācijas modelēšanas rezultātu prezentēšanai, vienlaikus ir arī efektīvs imitācijas modeļu verifikācijas un validācijas palīglīdzeklis. Tādēļ lielākā daļa moderno imitācijas modelēšanas programmatūras sistēmu lietotājiem piedāvā vizuālas divdimensiju (2D) un trīsdimensiju (3D) imitācijas modelēšanas iespējas, praktiski aizstājot imitācijas modelēšanas attīstības sākumposmā plaši izmantotās teksta valodas.

Neraugoties uz vizualizācijas metožu un līdzekļu plašo un nu jau pašsaprotamo pielietojumu imitācijas modelēšanas sistēmās, imitācijas modelēšanas un vizualizācijas integrācijā joprojām pastāv vairākas pilnībā neatrisinātas problēmas, kuras, pēc autora domām, eksistē ne tik daudz konceptuālu vai tehnoloģisku ierobežojumu dēļ, bet vairāk gan imitācijas modelēšanas kā teorētiskas un praktiskas nozares vēsturisko tradīciju dēļ.

Viena no pastāvošām problēmām ir tā, ka ir maz teorētisku pētījumu un praktisku rezultātu imitācijas modelēšanas un vizualizācijas interaktīvas mijiedarbības jomā. Lielākā daļa imitācijas modelēšanas programmlīdzekļu nenodrošina pilnvērtīgas iespējas mijiedarboties ar modeli tieši imitācijas izpildes gaitā, bet izmaiņu veikšanai ir nepieciešams apstādināt imitācijas procesu un pēc tam to turpināt. Lai gan eksistē dažas imitācijas modelēšanas sistēmas, kurās ir iespējama lietotāja interakcija ar modeli tā imitācijas eksperimentu gaitā, piemēram, Model Vision Studium [117], tās lietotājam piedāvā šauri specializētu modelēšanas paradigmu un tādējādi nav paredzētas plašam imitācijas uzdevumu lokam.

Otra pastāvošā problēma ir saistīta ar imitācijas modelēšanu diskrētu notikumu un nepārtrauktu sistēmu izpētē, izmantojot atšķirīgas pieejas un metodes. Ar to ir izskaidrojams fakts, ka tradicionālajās nepārtrauktu procesu imitācijas modelēšanas sistēmās lielas grūtības sagādā diskrētu notikumu apstrāde. Tādēļ, piemēram, Simulink [95, 123] vidē problēmas rada diskrētu notikumu modeļu raksturlielumi: notikumi un stāvokļi, notikumu vadāma darbība. Savukārt diskrētu notikumu imitācijas modelēšanas sistēmās ir neiespējami vai arī ļoti grūti veikt nepārtrauktu sistēmu modelēšanu. Lai gan pēdējā laikā pieaug izstrādāto programmatūras risinājumu skaits kombinētu diskrētu notikumu un nepārtrauktu sistēmu imitācijas modelēšanai, tomēr joprojām pastāv problēma, kas saistīta ar imitācijas modelēšanas un vizualizācijas sinhronizācijas jautājumiem šādu sistēmu izpētē.

Lai imitācijas modeļu izstrādes procesā būtu iespējams izpildīt prasības, kas iziet ārpus tradicionālo paradigmu robežām, rodas nepieciešamība pielietot, piemēram, skripta valodas, sarežģītus ārēju programmatūras moduļu integrācijas līdzekļus, kas tradicionālajās imitācijas modelēšanas vidēs ievērojami sarežģī modeļu izstrādi.

Promocijas darba mērķis un uzdevumi

Promocijas darba mērķis ir izstrādāt integrētu pieeju un metodes diskrētu notikumu un nepārtrauktu sistēmu imitācijas modelēšanai un vizualizācijai, tās praktiski realizēt programmatūras sistēmā, un veikt izstrādātās sistēmas eksperimentālu pārbaudi.

Promocijas darba mērķa sasniegšanai ir izvirzīti šādi uzdevumi:

- imitācijas modelēšanas un vizualizācijas pieeju, metožu un sistēmu salīdzinošs pētījums ar mērķi identificēt to izstrādē un integrācijā neatrisinātos uzdevumus;
- integrētas vizuālās imitācijas modelēšanas pieejas un metožu izstrāde iepriekš identificēto trūkumu novēršanai;
- diskrētu notikumu un nepārtrauktu sistēmu interaktīvas vizuālās imitācijas arhitektūras izstrāde;
- diskrētu notikumu un nepārtrauktu sistēmu vizuālās imitācijas modelēšanas sistēmas praktiska realizācija;
- izstrādātās sistēmas eksperimentāla pārbaude, parādot tās praktiskās pielietošanas iespējas.

Pētījumu objekts un priekšmets

Darba pētījumu objekts ir imitācijas modelēšanas un vizualizācijas mijiedarbības process.

Darba pētījumu priekšmets ir imitācijas modelēšanas un vizualizācijas raksturīgās īpašības, kas nosaka integrētas vizuālās imitācijas modelēšanas vides funkcionalitāti, un metodes, kas nodrošina šo atbilstošo funkcionalitāti.

Pētījumu metodes

Šajā darbā teorētisko pētījumu veikšanai ir izmantota sistēmu teorija un diskrētu notikumu sistēmu specifikācijas elementi, kopu teorija, datorvizualizācijas teorija. Promocijas darba izstrādātā sistēmas prototipa realizācijai ir izmantotas programmatūras inženierijas, imitācijas modelēšanas un datorgrafikas metodes.

Darba zinātniskais jaunieguvums un vērtība

Izstrādātā promocijas darba galvenais zinātniskais jaunieguvums ir šāds:

1. Izstrādāts sistēmteorētisks interaktīvas vizuālās imitācijas modelēšanas formālisms, kas paredzēts integrētu vizuālo imitācijas modelēšanas sistēmu izstrādei, izmantojot kombinētu diskrētu notikumu un nepārtrauktu sistēmu imitācijas pieeju. No tipiskām imitācijas modelēšanas koncepcijām tas atšķiras ar to, ka ir universāli pielietojams un integrē imitācijas modelēšanas un vizualizācijas funkcionalitāti.

2. Balstoties uz vizuālās imitācijas modelēšanas formālismu, ir izstrādāta arhitektūra, metodika un algoritmi tā praktiskai realizācijai, kas no tradicionālām imitācijas modelēšanas sistēmām atšķiras ar vienotu modelēšanas, imitācijas, vizualizācijas un interaktīvo elementu traktējumu, kas unificē un vienkāršo imitācijas modeļu izstrādi un izmantošanu.

Izpētot un atrisinot darba gaitā sastaptās problēmas, līdztekus galvenajam jaunieguvumam ir sasniegti šādi zinātniskie rezultāti:

1. Ieviests vizualizācijas ietvara jēdziens integrētas vizuālās imitācijas modelēšanas konteksta definēšanai mijiedarbībā ar imitācijas un modelēšanas ietvaru.

2. Realizējot izstrādāto vizuālās imitācijas modelēšanas formālismu, ir konstatēts, ka modeļvadāmā pieeja uzlabo un vienkāršo imitācijas modeļa izstrādes un verifikācijas procesu.

3. Izstrādāts algoritms integrētai nepārtrauktu sistēmu kvantētu stāvokļu imitācijas vizualizācijai, kas ļauj būtiski palielināt kvantēšanas soli un tādējādi samazināt nepieciešamo skaitļošanas resursietilpību, nesamazinot iegūto imitācijas rezultātu precizitāti.

4. Aprobējot izstrādāto formālismu praktiska programmatūras prototipa veidā, ir izstrādāti divi simulatora varianti un ir pierādīts, ka prioritātes rindu algoritms ir ievērojami efektīvāks par hierarhisko simulatora algoritmu, tādējādi tas ir piemērotāks praktiskas vizuālās imitācijas modelēšanas sistēmas izstrādē un izmantošanā.

Darba praktiskā vērtība

Darba praktiskā vērtība ir izstrādātā integrētā imitācijas modelēšanas un vizualizācijas pieeja un realizētais sistēmas prototips, kas apvieno diskrētu notikumu un nepārtrauktu sistēmu imitācijas modelēšanas un 2D/3D vizualizācijas tehnikas. Minēto tehniku apvienojums vienotā integrētā vidē ļauj efektīvi nodrošināt vizuālu interaktīvu imitācijas modelēšanas uzdevumu risināšanu.

Darba aprobācija

Par promocijas darba galvenajiem rezultātiem ir nolasīti referāti 8 starptautiskās zinātniskajās konferencēs:

1. eLOGMAR-M Workshop, Riga, Latvia, September 22, 2006.
2. 19th European Conference on Modelling and Simulation, Riga, Latvia, June 1-4, 2005.
3. RTU 45. Starptautiskā zinātniskā konference. Rīga, 13-14.oktobris, 2004.
4. RTU 44. Starptautiskā zinātniskā konference. Rīga, 9-11.oktobris, 2003.
5. The International Workshop on Harbour, Maritime and Multimodal Logistics Modelling & Simulation (HMS2003), Riga, Latvia, September 18-20, 2003.
6. Tagung “Simulation und Visualisierung ’2003” der Otto-von-Guericke Universität Magdeburg, Magdeburg, Germany, 6-7 März, 2003.
7. RTU 43. Starptautiskā zinātniskā konference. Rīga, 10-14.oktobris, 2002.
8. RTU 42. Starptautiskā zinātniskā konference. Rīga, 11-13.oktobris, 2001.

Promocijas darba ietvaros veikto pētījumu rezultāti ir atspoguļoti 7 publikācijās starptautiskos Latvijas Zinātnes padomes atzītos zinātniskajos izdevumos:

1. Lektauers A., Merkuryev Y. 3D Visual Framework for Modelling and Simulation of Supply Chain Systems// Scientific Proceedings of the eLOGMAR-M Project. - Riga: JUMI Ltd., 2006. - 141-150 p.

2. Lektauers A. A Mixed Reality Framework for Visualization and Execution of DEVS-based Simulation Models// Proceedings of the 19^{th} European Conference on Modelling and Simulation "Simulation in Wider Europe", ECMS2005. - Riga: Publishing House of Riga Technical University, 2005. - 271-276 p.

3. Lektauers A. Developing a 3D Graphical User Interface for Object-Oriented Simulation Modelling// Rīgas Tehniskās universitātes zinātniskie raksti. 5.sēr., Datorzinātne. - Rīga: RTU, 2003. - 14.sēj., - 36-42 lpp.

4. Lektauers A. A Virtual 3D Environment for Logistics Modelling// Proceedings of the International Workshop on Harbour, Maritime and Multimodal Logistics Modelling & Simulation (HMS2003). - Riga: RTU, 2003. - 242-248 p.

5. Lektauers A. 3D-Projektierung von Simulationsmodellen in einer VR-Umgebung// Proceedings der Tagung "Simulation und Visualisierung '2003" der Otto-von-Guericke Universität Magdeburg (SimVis 2003). - Magdeburg: SCS Publishing House, 2003. - 313-322 S.

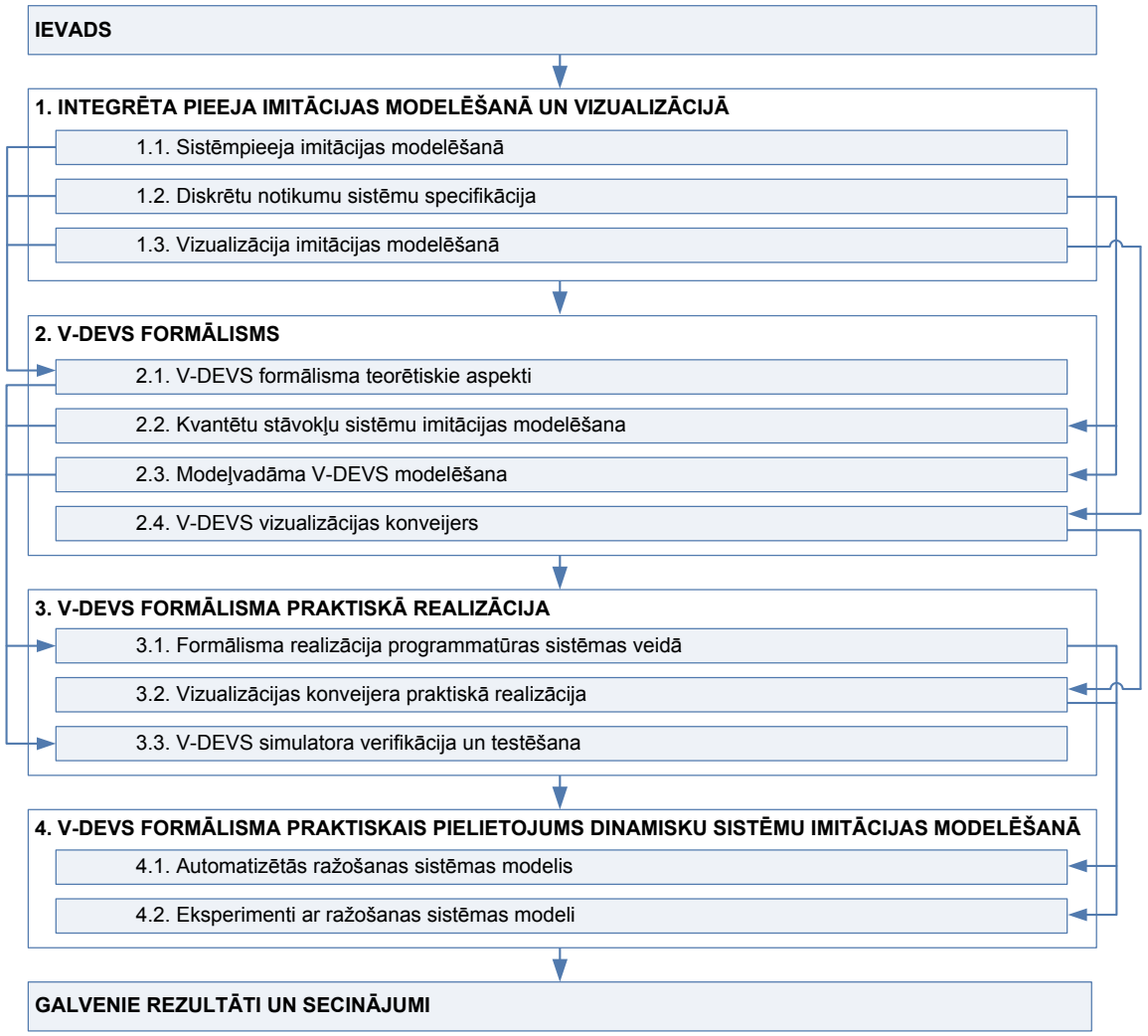
6. Lektauers A. Imitācijas modeļu izveide virtuālā 3D vidē// Rīgas Tehniskās universitātes zinātniskie raksti. 5.sēr., Datorzinātne. - Rīga: RTU, 2002. - 10.sēj., - 107-113 lpp.

7. Lektauers A. Imitācijas modeļu vizualizācijas tehnoloģija// Rīgas Tehniskās universitātes zinātniskie raksti. 5.sēr., Datorzinātne. - Rīga: RTU, 2001. - 5.sēj., - 103-109 lpp.

Promocijas darba struktūra

Promocijas darbs sastāv no ievada, 4 nodaļām, secinājumiem, 4 pielikumiem un literatūras avotu saraksta. Darbs ir rakstīts latviešu valodā. Promocijas darba uzbūve un loģiskā struktūra ir pakārtota izvirzīto darba uzdevumu risināšanai (1. attēls).

Ievadā ir pamatota izvēlētās tēmas aktualitāte, noformulēts pētījumu mērķis un uzdevumi, uzskaitītas promocijas darba izstrādē lietotās zinātniskās metodes, aprakstīts pētījumu zinātniskais jaunieguvums un darba iegūto rezultātu praktiskā vērtība, kā arī ir sniegts darba aprobācijas raksturojums.



1. att. Promocijas darba uzbūve un loģiskā struktūra

Promocijas darba 1. nodaļa "Integrēta pieeja imitācijas modelēšanā un vizualizācijā" ir teorētiska nodaļa, kurā ar mērķi identificēt esošo vizuālo imitācijas modelēšanas sistēmu trūkumus un vispārīgā līmenī definēt prasības integrētai vizuālai interaktīvai diskrētu notikumu un nepārtrauktu sistēmu imitācijas modelēšanas videi ir analizēts un pamatots sistēmpieejā sakņotās diskrētu notikumu sistēmu specifikācijas lietojums un priekšrocības, kā arī ir veikta vizualizācijas pieeju un metožu salīdzinošā analīze to integrācijai imitācijas modelēšanā.

Darba 2. nodaļā "V-DEVS formālisms" ir piedāvāta integrēta vizuālās imitācijas modelēšanas pieeja, kuru apraksta darba gaitā izstrādātais sistēmteorētisks interaktīvas vizuālās diskrētu notikumu un nepārtrauktu sistēmu imitācijas modelēšanas formālisms V-DEVS. Uz tā pamata ir izstrādāta interpolācijas metode kvantētu stāvokļu sistēmu vizuālajai imitācijai, izpētītas modeļvadāmās V-DEVS modelēšanas koncepcijas, kā arī izstrādāts V-DEVS vizualizācijas konveijers.

Darba 3. nodaļā "V-DEVS formālisma praktiskā realizācija", precizējot otrajā nodaļā de-

finētās vispārējās prasības interaktīvai imitācijas modelēšanas videi, ir aprakstīta V-DEVS formālisma praktiskā realizācija programmatūras sistēmas veidā, kas ietver simulatora, modeļvadāmās pieejas un vizualizācijas konveijera realizācijas elementus. Ar izstrādāto programmatūras prototipu ir veikti praktiski eksperimenti tā pamatā esošo koncepciju un realizācijas detaļu verifikācijai.

Darba 4. nodaļā "V-DEVS formālisma praktiskais pielietojums dinamisku sistēmu imitācijas modelēšanā" ir izklāstīta automatizētās ražošanas sistēmas V-DEVS modeļa realizācija un ir aprakstīti un analizēti iegūtie eksperimentālie rezultāti, parādot izstrādātā formālisma praktiskā pielietojuma iespējas dinamisku sistēmu interaktīvajā imitācijas modelēšanā.

Promocijas darba rezultāti un secinājumi ir apkopoti darba noslēguma daļā.

1. INTEGRĒTA PIEEJA IMITĀCIJAS MODELĒŠANĀ UN VIZUALIZĀCIJĀ

20.gadsimta otrajā pusē daudzās praktiskajās un zinātnes nozarēs plaši sāka pielietot terminu *sarežģītas sistēmas*, šo jēdzienu attiecinot uz sarežģītām dinamisku objektu vadības sistēmām [119, 120], sastāvošām no daudziem atšķirīgas uzbūves, mainīgas funkcionalitātes savstarpēji saistītiem komponentiem. Praktiski jebkura sarežģīta sistēma kopumā ir uzskatāma par kombinētu sistēmu. Ar *kombinētas sistēmas* jēdzienu tiek apzīmētas sistēmas, kurās ir novērojami gan nepārtrauktie, gan diskrētie darbības aspekti [117]. Literatūrā tiek izmantoti arī termini "hibrīdas sistēmas", "mainīgas struktūras sistēmas", "notikumu vadāmās sistēmas". Parasti tiek izdalīti trīs galvenie aspekti, kas nosaka nepieciešamību izmantot hibrīdas sistēmas:

1. kombinēta nepārtraukto un diskrēto objektu funkcionēšana;
2. nepārtraukto objektu kvalitatīvās izmaiņas;
3. sistēmā ietilpstošo nepārtraukto objektu skaita un sastāva izmaiņas funkcionēšanas gaitā.

1.1. Sistēmpieeja imitācijas modelēšanā

Sarežģītu (lielu) sistēmu analīzē un sintēzē plaši tiek izmantota sistēmpieeja, kas no klasiskās (induktīvās) pieejas atšķiras ar to, ka šī pieeja izmanto pāreju no vispārīgā uz detalizēto, kuras pamatā ir noteikts mērķis, pētāmo objektu izdalot no apkārtējās vides [118]. Jebkuru sarežģītu sistēmu, piemēram, dažāda līmeņa industriālo vadības sistēmu, projektēšanā un ekspluatācijā ir viena kopīga iezīme - nepieciešamība sasniegt noteiktu mērķi, kas kalpo par sistēmas definīcijas pamatu.

1.1.1. Sistēmu teorija

Sistēmu teorija ir saistāma ar sistēmu inženieriju - zinātnes jomu, kas pēta un izstrādā metodes, paņēmienus un rīkus sarežģītu sistēmu izveidei. Vispārējā sistēmu teorijā ir definēta kauzāla determinēta sistēma S, kas ir pamatā daudziem dažādiem formālismiem, piemēram, parastajiem diferenciālvienādojumiem, galīgo stāvokļu automātiem, Petri tīkliem u.t.t.. *Sistēma* S ir savstarpēji saistītu elementu kopums, kas darbojas ar kopēju mērķi izpildīt funkcijas, kuras katram tās elementam atsevišķi nav iespējamas [78]. Jebkura sistēma ir apskatāma noteiktas *ārējās vides* E kontekstā, kas definē jebkāda veida ārpus sistēmas eksistējošu elementu kopu, kuriem ir ietekme uz sistēmu vai kurus ietekmē pati sistēma.

Vispārējā gadījumā sistēmu iespējams aprakstīt ar operatoru (funkciju) $\mathscr{S}$, kas sasaista ieejas signāla vektoru $x(t)$ ar izejas signāla vektoru $y(t)$ [88]:

$$y(t) = \mathscr{S}x(t).$$

Sistēmas tiek iedalītas kategorijās, kuru robežas vispārējā gadījumā nav strikti nodalāmas:

- *Statiskas sistēmas*, kurām ir spēkā vienādība $y(t) = \mathscr{S}x(t)$, t.i., izejas funkcija y laika momentā t ir atkarīga tikai no ieejas x šajā pašā laika momentā.
- *Dinamiskas sistēmas* - starp ieeju un izeju nav statiskas atbilstības, bet ir jāpieņem, ka $y(t) = \mathscr{S}(x,t)$, t.i., izejas y vērtība laika momentā t var būt atkarīga no visām vērtībām $x(\tau)$, $\tau \in (-\infty, +\infty)$.
- *Determinētas sistēmas* - starp sistēmas ieejām un izejām pastāv viennozīmīga sakarība.
- *Nedeterminētas sistēmas* - izeju vērtības var atrasties noteiktas vērtību kopas robežās, t.i., katrā laika momentā t sistēmai ir piemērojama funkcija $y(t) = \mathscr{S}(x,t)$.
- *Stohastiskas sistēmas* - nedeterminētas sistēmas ar noteiktu izejas vērtību kopu, taču šajā gadījumā izejas vērtības ir asociējamas ar varbūtiskiem lielumiem.
- *Kauzālas sistēmas* - sistēmas izeja $y(t)$ jebkurā laika momentā t ir atkarīga no pagātnes ieejas vērtībām $x(t-\theta)$, $\theta \geqslant 0$.
- *Nekauzālas sistēmas* - sistēmas izeja $y(t)$ jebkurā laika momentā t ir atkarīga no nākotnes (nezināmām) ieejas vērtībām $x(t+\theta)$, $\theta \geqslant 0$.

Atšķirībā no iepriekš apskatītās sistēmu klasifikācijas, sistēmas parasti tiek klasificētas atkarībā no to ieejas un izejas vērtību telpas tipa. Atbilstoši šai klasifikācijai iespējams izdalīt vairākus sistēmu tipus:

- diskrēta laika, diskrētu notikumu sistēmas;
- diskrēta laika, nepārtrauktas sistēmas;
- nepārtraukta laika, diskrētu notikumu sistēmas;
- nepārtrauktas sistēmas.

Sistēmu teorija definē fundamentālu matemātisko formālismu dinamisku sistēmu aprakstam, izšķirot divus pamataspektus:

- sistēmas specifikācijas līmeņi;
- sistēmu specifikācijas formālismi.

Sistēmu specifikācijas līmeņi attiecas uz dažādām pakāpēm, kādās iespējams aprakstīt doto sistēmu ar esošajām zināšanām par to. 1.1. tabulā dots pārskats par sistēmu specifikācijas līmeņiem.

1.1. tabula

Sistēmu specifikācijas hierarhija (aizgūts no [111])

Līmenis	Specifikācija	Apraksts
0	Ieeju / izeju ietvars	Apraksta laika bāzi un ieeju / izeju vērtību kopas
1	Ieeju / izeju relāciju novērojumi	Apraksta visus iespējamos ieeju / izeju segmentu pārus
2	Ieeju / izeju funkciju novērojumi	Transformē relācijas uz fukciju kopu
3	Ieeju / izeju sistēma	Pievieno sistēmas iekšējos stāvokļus, globālo stāvokļu pāreju un izejas funkciju
4	Iteratīvā specifikācija	Aizstāj globālo stāvokļu pārejas funkciju ar iteratīvām stāvokļu pārejām
5	Strukturēta sistēmas specifikācija	Visas kopas tiek attēlotas multimainīgo kopās
6	Nemodulāra saistīta daudzkomponenšu sistēma	Ievieš saistītas sistēmas, kurās komponenti cits citu tiešā veidā ietekmē ar stāvokļu pārejas funkcijām
7	Modulāra saistīta daudzkomponenšu sistēma	Ievieš modularitāti, ierobežojot ietekmi uz ieejas un izejas interfeisiem

1.1.2. Sistēmu modelēšana

Sarežģītu sistēmu izpētes metožu augsto prasību dēļ ir vērojama pāreja no kvalitatīviem paņēmieniem uz kvantitatīvām metodēm. Šī pāreja īpaši ir redzama sistēmu modelēšanas jomā, kas ir viena no visplašāk pielietotajām sarežģītu sistēmu analīzes metodēm. Modelēšana ir process, kurā ar modeļu palīdzību notiek reālās pasaules abstrahēšana vienkāršākā formā. *Modelis* ir sistēmas vai apakšsistēmas formāls apraksts, kas apskata izvēlēto informāciju vai zināšanas. Sistēmu modelēšanā ir izšķirami pieci modeļu pamatveidi [44]:

- fiziskie modeļi (piemēram, reālu objektu mēroga kopijas);

- naturālie modeļi;
- uzskatāmie modeļi (piemēram, inženierrasējumi);
- simboliskie modeļi (piemēram, formālās specifikācijas);
- matemātiskie modeļi.

Sīkāks sistēmu modelēšanas veidu iedalījums ir attēlots 1. pielikumā.

Pāreju no kvalitatīvām metodēm uz kvantitatīvām metodēm parāda arvien pieaugošais simbolisko modelēšanas metožu un pieeju skaits, jo tās ir labi piemērotas kvantitatīvai sistēmu analīzei. Parasti simboliskie modeļi tiek izstrādāti, izmantojot speciālas modelēšanas valodas. Modelēšanas valodas atkarībā no modelējamās sistēmas veida var iedalīt trīs kategorijās:

1. nepārtrauktās valodas;
2. diskrētu notikumu valodas;
3. kombinētās jeb hibrīdās valodas.

Nepārtrauktās valodas izmanto nepārtrauktu (fizikālu) sistēmu un procesu modelēšanai. Diskrētu notikumu valodas, piemēram, GPSS [87], SIMAN (*SIMulation ANalysis*) [43], izmanto (fizikālu) sistēmu un procesu diskrētu notikumu norišu modelēšanai. Eksistē arī hibrīdās valodas, piemēram, MATLAB / Simulink [95], AnyLogic [122], kas nepārtrauktās un diskrētu notikumu īpašības apvieno vienotā formālismā. Iepriekš minēto modelēšanas valodu galvenais mērķis ir nodrošināt sistēmu dinamiskās darbības izpēti ar imitācijas modelēšanas līdzekļiem, ļaujot efektīvā un relatīvi vienkāršā veidā analizēt pētāmo sistēmu parametrus.

Pie matemātiskajiem modeļiem ir pieskaitāmi imitācijas modeļi, ar kuriem ir saistāms imitācijas modelēšanas jēdziens. *Imitācijas modelēšana* ir definējama kā reālas sistēmas modeļa izveides un eksperimentēšanas process ar mērķi izprast modelējamās sistēmas darbību vai arī izpētīt dažādas tās darbības stratēģijas [87]. Tā kā jebkurš sistēmu analīzes un sintēzes process ir saistāms ar noteiktu lēmumu pieņemšanu, tad arī imitācijas modelēšana ir uzskatāma par lēmumu pieņemšanas palīglīdzekli. Imitācijas modelēšana ir iedalāma trīs galvenajos etapos:

1. modeļa izveide;
2. modeļa izpilde;
3. modelēšanas rezultātu analīze.

Attiecībā uz sistēmpieeju imitācijas modelēšanā ir jāņem vērā vairāki aspekti [113]:

- diskrēto notikumu pārtrauktība;
- nepārtrauktība lielākajai daļai veiktspējas mērījumu;
- varbūtiska formulējuma svarīgums;
- hierarhiskas analīzes nepieciešamība;
- dinamikas esamība;
- praktiskās realizācijas iespējas.

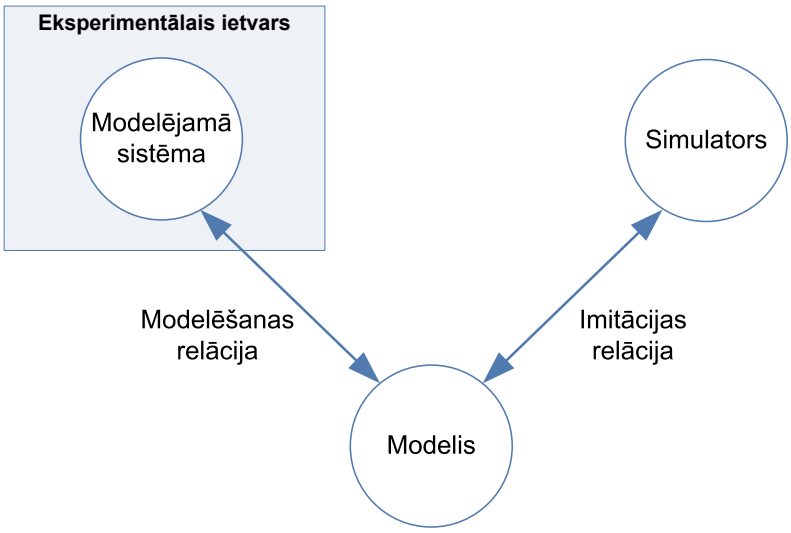
Sistēmpieejai imitācijas modelēšanā ir ļoti liela nozīme, jo tā modelējamo sistēmu, atšķirībā no klasiskās (induktīvās) pieejas, apskata kā pakāpenisku pāreju no vispārējā uz detalizēto, balstoties uz noteiktu mērķi un pētāmo objektu izdalot no apkārtējās vides [118].

1.1.3. Modelēšanas un imitācijas ietvars

Termini "modelis" un "imitācija" bieži vien tiek lietoti kā sinonīmi, kaut gan būtībā tiem ir stingri noteikta un atšķirīga nozīme, ko definē *modelēšanas un imitācijas ietvars* (angļu valodā *Framework for Modelling and Simulation*) [19]. Modelēšanas un imitācijas ietvars, balstoties uz sistēmu teoriju, nodrošina bāzes koncepcijas imitācijas modelēšanas programmatūras vidēm. Taču, lai gan sistēmu teorija nodrošina stingru modelēšanas un imitācijas ietvara matemātisko pamatu, tai trūkst konceptuālās un skaitļošanas nodrošinājuma infrastruktūras. Savukārt šādu infrastruktūru nodrošina objektorientētā metodoloģija.

Modelēšanas un imitācijas ietvarā centrālā loma ir četriem elementiem jeb *entītijām* - *modelējamai sistēmai*, *modelim*, *simulatoram* un *eksperimentālajam ietvaram*, kā arī to savstarpējām modelēšanas un imitācijas saitēm jeb *relācijām* (1.1. attēls). Katra entītija tiek formāli raksturota kā apakšsistēma noteiktā vispārējas dinamiskas sistēmas specifikācijas līmenī. Modelējamā sistēma, saukta arī par *etalonsistēmu*, *realitāti* [71] vai *reālo sistēmu* [40], ir reāla vai virtuāla sistēma, kas ir imitācijas modelēšanas interešu un mērķa objekts. Modelējamā sistēma modelēšanas un imitācijas ietvarā tiek apskatīta kā novērojamu datu avots laikā indeksētu mainīgo veidā, kur šie dati tiek aplūkoti vai iegūti modelētāja interešu eksperimentālajā ietvarā. Novērotos datus iespējams aprakstīt kā ieeju un izeju laika segmentu pārus [109], kā rezultātā sistēma ir reducējama uz 0. vai 1.specifikācijas līmeni (1.1. tabula). Bieži vien par šīs sistēmas iekšējo struktūru praktiski nekas nav zināms, un to iespējams pētīt tikai ar novēroto datu palīdzību.

Eksperimentālais ietvars ir specifikācija nosacījumiem, pie kādiem tiek veikti sistēmas eksperimenti un novērojumi. Tādējādi eksperimentālais ietvars nosaka imitācijas modelēšanas uzdevumu formulējumu, un ir viena no nākamajā nodaļā apskatāmās diskrēto notikumu sistēmas



1.1. att. Modelēšanas un imitācijas entītijas un to savstarpējā saistība (aizgūts no [109])

specifikācijas pamatkoncepcijām [111]. Eksperimentālais ietvars tiek realizēts kā sistēma, kas pie noteiktiem nosacījumiem mijiedarbojas ar modelējamo sistēmu interesējošo datu ieguvei. Un pamatprincips, kādēļ ir nepieciešams eksperimentālais ietvars, ir tāds, ka jebkurai imitācijas datu ieguvei (statistika, veiktspējas rādītāji u.c.) vai imitācijas izpildes vadībai (inicializācija, apstādināšana u.c.), kas nav konceptuāli tieši saistītas ar reālo sistēmu, jābūt formulētai nevis kā imitācijas modeļa, bet gan kā eksperimentālā ietvara sastāvdaļai. Eksperimentālā ietvara specifikācija ir iedalāma četrās apakšgrupās [107], kas eksperimentālā ietvara kā interaktīvas sistēmas realizācijas gaitā kļūst par vadības sistēmas komponentiem:

- *ieejas stimuli* - pieļaujamo laika atkarīgu ieejas iedarbju specifikācija;
- *vadība* - nosacījumu specifikācija, kas nosaka sistēmas modeļa inicializāciju, imitācijas izpildi un tās pabeigšanu;
- *metrika* - datu apkopošanas funkciju un mērījumu specifikācija kvantitatīvas vai kvalitatīvas imitācijas analīzes veikšanai;
- *analīze* - iegūto imitācijas datu analīzes līdzekļu specifikācija galīgo slēdzienu un lēmumu pieņemšanai.

Eksperimentālo ietvaru formāli iespējams definēt kā struktūru [108]:

$$EF = \langle T, I, C, O, \Omega_I, \Omega_C, SU \rangle, \tag{1.1}$$

kur T - laika bāze;

I - ieejas mainīgo kopa;

C - izpildes vadības mainīgo kopa;

O - izejas mainīgo kopa;

$\Omega_I \subset (X,T)$ - pieļaujamo ieejas segmentu kopa, kur

X - eksperimentālā ietvara definēto ieejas mainīgo kopa;

Ω_C - pieļaujamo izpildes kontroles mainīgo kopa;

SU - imitācijas rezultātu attēlošana kopa.

Lai eksperimentālo ietvaru būtu iespējams interpretēt gan kā reālās sistēmas, gan modeļa eksperimentēšanas koncepciju, eksistē divi atšķirīgi formulējumi:

- Imitācijas režīmā modeļa pieļaujamo ieejas segmentu kopu realizē ģenerators, kas ģenerē izejas segmentu $\omega \in \Omega_I$, izpildot to no piemērota sākuma stāvokļa un noteiktā novērojumu laika intervālā.

- Eksperimentu laikā ar reālo sistēmu eksperimentālā ietvara uzdevums ir nodrošināt nepieciešamo reālo ieejas datu izvēli, ko realizē akceptors, akceptējot vai noraidot akceptora ieejās saņemtos datus.

Eksperimentālais ietvars parasti satur trīs komponentus [111]:

1. *ģenerators* - sistēmas ieejas segmentu ģenerēšana;

2. *akceptors* - eksperimentālo nosacījumu monitorings;

3. *devējs* - sistēmas izejas segmentu novērošana un analīze.

Kaut gan klasiskais modelēšanas un imitācijas ietvars ir plaši akceptēts imitācijas un modelēšanas aprindās, darbā [98] ir piedāvāts šī ietvara uzlabojums, kura motivācija ir nepieciešamība formāli ņemt vērā arī modelējamās sistēmas izpētes kontekstu, kas būtībā ir svarīgs faktors jebkuras sistēmas imitācijas modelēšanā, jo jebkurš imitācijas modelēšanas etaps tiek veikts noteiktā kontekstā, no kā izriet arī ar eksperimentālo ietvaru saistāmie eksperimentu veikšanas nosacījumi. Tas ir saistīts ar iepriekš minēto imitācijas un modelēšanas ietvara dualitāti.

Imitācijas modelis ir abstrakta modelējamās sistēmas specifikācija instrukciju, likumu, vienādojumu vai ierobežojumu veidā ieeju un izeju dinamikas ģenerēšanai. Modeļus iespējams izteikt dažādu formālismu veidā, kurus var traktēt kā dinamisku sistēmu apakšklašu definēšanas līdzekļus.

Simulators ir noteikts skaitļošanas elements, piemēram, procesors vai algoritms, kas ir spējīgs izpildīt modeli tā dinamiskās uzvedības ģenerēšanai. Jo universālāks ir simulators, jo lielākas tā konfigurēšanas iespējas dažāda tipa modeļu izpildei. Simulators var būt gan modeļa realizācija, gan arī imitācijas izpildelements, kura uzdevums ir izpildīt modeļa definētās instrukcijas, un simulatoru izveidē ir iespējami abi varianti. Taču abstrakta simulatora izmantošanai ar iespēju

izpildīt dažādus modeļus, fizisko loģiku nodalot no vadības loģikas, ir daudzas priekšrocības, kā piemēram ātrākas jaunu modeļu izveides un modificēšanas iespējas, salīdzinot ar visu modeļu un simulatoru izveidi no jauna [72]. Fundamentāls modelēšanas un imitācijas ietvara princips, kas sakņojas sistēmu teorijā, ir modeļa un simulatora savstarpējā nodalīšana.

1.2. Diskrētu notikumu sistēmu specifikācija

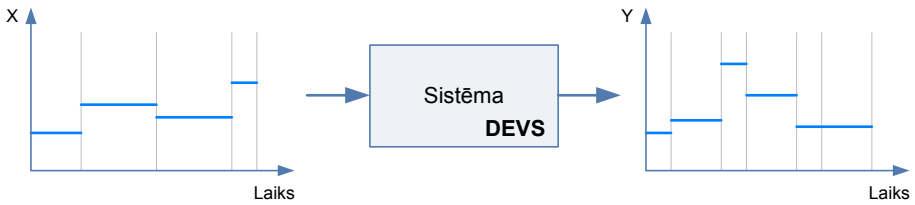
Lai nodrošinātu sarežģītu sistēmu imitācijas modeļu konceptualizāciju un specifikāciju, eksistē dažādas paradigmas, formālismi, modelēšanas metodoloģijas un imitācijas metodes. Petri tīkli, diferenciālvienādojumi un diskrētu notikumu sistēmu specifikācija (angļu valodā *Discrete Event System Specification*) (DEVS) [107] ir tipiski piemēri. Daudzi eksistējošie mēģinājumi ņemt vērā visus 1.1.2.apakšsekcijā minētos imitācijas modelēšanas aspektus un izstrādāt vienotu imitācijas modelēšanas ietvaru ir beigušies neveiksmīgi, taču eksistē arī dotās problēmas risinājums - tieši augstākminēto argumentu iespaidā izstrādātais DEVS formālisms.

Diskrētu notikumu sistēmu specifikācija ir vispārīgs universāls formālisms, kas paredzēts diskrēto notikumu sistēmu dinamikas aprakstīšanai un definēšanai, bet to ir iespējams pielietot arī tradicionālo formālismu aprakstīto sistēmu attēlošanā, kā piemēram, diferenciālvienādojumi (nepārtrauktās sistēmas) un diferenču vienādojumi (diskrētā laika sistēmas). DEVS formālismu ir izstrādājis amerikāņu zinātnieks Bernards Zeiglers 1970.gadu sākumā ar mērķi nodrošināt diskrēto notikumu imitācijas modeļu izstrādi hierarhiskā un modulārā veidā.

DEVS formālisms ļauj formāli aprakstīt jebkuru sistēmu, kuras ieeju un izeju izmaiņas iespējams definēt kā secīgu notikumu virknes pie nosacījuma, ka sistēmas stāvokļu izmaiņu apjoms ir galīgs jebkurā galīgā laika intervālā. Tādējādi DEVS formālisms apskata diskrētu notikumu sistēmas, kurās laika virzība tiek vadīta ar notikumu palīdzību. No tā izriet, ka ne tikai Petri tīkli, stāvokļu diagrammas, notikumu grafi un citas diskrētu notikumu valodas, bet jebkura diskrētu notikumu sistēma ir uzskatāma kā DEVS speciālgadījums. Uz DEVS formālismu balstītas sistēmas ir iespējams pielietot visdažādākajās jomās, kā piemēram, datorzinātnē, transporta, loģistikas un piegādes ķēžu vadībā, vides modelēšanā u.c.. Eksistē daudzas DEVS formālisma praktiskās realizācijas imitācijas modelēšanas programmatūras veidā, kā piemēram, DEVSJAVA [112], adevs [67], PowerDEVS [46], CD++ [102]. Bet ir jāpiebilst, ka visas šīs eksistējošās darba autoram zināmās DEVS programmatūras sistēmas ir izstrādātas vairāk kā teorētisko ideju realizācijas prototipi un nesatur tik plašas ar imitācijas modelēšanu saistītās iespējas (piemēram, apjomīgas modeļu bibliotēkas, eksperimentēšanas un rezultātu analīzes nodrošinājums), kā komerciālie imitācijas modelēšanas rīki.

DEVS sistēma apstrādā ieejas notikumus, un atbilstoši savas darbības iekšējai loģikai pēc

noteikta laika var ģenerēt izejas notikumus. Visiem notikumiem ir piekārtots noteikts laiks, un notikumu apstrādes laikā laika virzība notiek pa diskrētiem soļiem, taču notikumu biežums nav stingri definēts ar noteiktu frekvenci, bet gan tas ir gadījuma lielums. 1.2. attēlā ir parādīta sistēmas ieejas un izejas izmaiņu iespējamā sakarība laikā.



1.2. att. DEVS ieejas un izejas izmaiņas laikā

Oriģinālo DEVS formālismu dēvē arī par par *klasisko* DEVS, jo vēlāk tika izstrādāta uzlabota DEVS formālisma modifikācija, saukta par *paralēlo* DEVS jeb P-DEVS formālismu [108]. Paralēlais DEVS formālisms ļauj pārvarēt klasiskā DEVS formālisma ierobežojumus, kas saistīti ar agrīno datoru secīgo imitācijas modelēšanas izpildi, un paver pilnas paralēlisma izmantošanas iespējas imitācijas modelēšanā.

1.2.1. Imitācijas modeļu specifikācija

Atomārie modeļi

DEVS formālisms izmanto kopu teoriju precīzam modelējamās sistēmas formulējumam, un to ir iespējams izmantot gan matemātiskajai analīzei, gan kā pamatu efektīvai imitācijas modelēšanas metodoloģijas un tehnoloģijas izveidei. Paralēlās diskrētu notikumu sistēmu specifikācijas atomārie modeļi ir definējami šāda korteža veidā:

$$AM_{DEVS} = \langle X, Y, S, \delta_{int}, \delta_{ext}, \delta_{conf}, \lambda, ta \rangle, \tag{1.2}$$

kur X - ieejas notikumu vērtību kopa;

Y - izejas notikumu vērtību kopa;

S - secīgu stāvokļu kopa;

$\delta_{int} : S \to S$ - *iekšējā pārejas funkcija*, kas nosaka sistēmas pāriešanu jaunā stāvoklī pēc laika $ta(s)$;

$\delta_{ext} : Q \times X \to S$ - *ārējā pārejas funkcija*, kas ieejas notikuma saņemšanas momentā izraisa tūlītēju sistēmas pāriešanu jaunā stāvoklī,

kur $Q = \{(s,e) \mid s \in S, 0 \leq e \leq ta(s)\}$ - *kopējā stāvokļu telpa*;

$s \in S$ - stāvoklis, kādā atrodas sistēma kopš pēdējās stāvokļu pārejas;

e - pagājušais laiks kopš pēdējās stāvokļu pārejas;

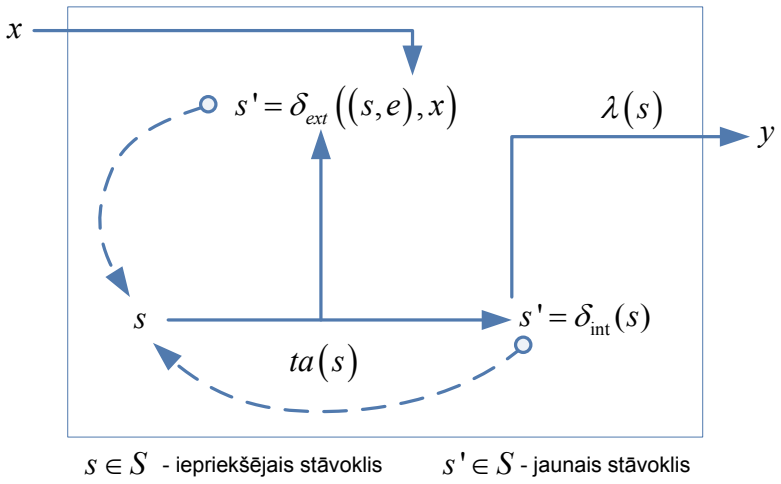
$\delta_{conf} : Q \times X \to S$ - *vienlaicīgā pārejas funkcija*;

$\lambda : S \to Y$ - *izejas funkcija*, kas pirms iekšējās stāvokļu pārejas funkcijas izpildes ģenerē izejas notikumu;

$ta : S \to \mathbb{R}_{0,+\infty}$ - *laika ritējuma funkcija*, kas, neesot ieejas notikumiem, nosaka laika intervālu līdz nākamajai iekšējai stāvokļu pārejai,

kur $\mathbb{R}_{0,+\infty} = [0, +\infty]$ - pozitīvu reālu skaitļu intervāls.

Imitācijas laika bāze T ir nepārtraukta un netiek tiešā veidā pieminēta: $T = \mathbb{R}$. Diskrētā laika momentā $t \in T \geq 0$ sistēma atrodas noteiktā stāvoklī $s \in S$. Ja nav ārēju notikumu, sistēma šajā stāvoklī s atrodas laika periodā $ta(s)$. Ja $ta(s) = 0$, tad sistēmas atrašanās stāvoklī s ir tik īsa, ka vairs nav iespējama ārējo notikumu apstrāde. Ja laika ritējuma funkcija $ta(s) = \infty$, kas matemātiskā nozīmē nav konkrēta vērtība, bet bezgalīga lieluma simbols, taču praktiskajā realizācijā iegūst reālu skaitļošanas platformas atkarīgu reālā tipa maksimālo vērtību, tad sistēma pastāvīgi atrodas stāvoklī s līdz brīdim, kad tiek saņemts ārējais paziņojums. Kad atrašanās laiks stāvoklī s ir beidzies, t.i, $ta(s) = e$, sistēma izdod izejas vērtību $\lambda(s)$ un pāriet jaunā stāvoklī $s' = \delta_{int}(s)$ (1.3. attēls). Ja ārējais notikums $x \in X$ notiek pirms iekšējā stāvokļa beigu laika (t.i., sistēma atrodas kopējā stāvoklī $(s, e) \in Q$, kur $e \leq ta(s)$), tad sistēma pāriet jaunā stāvoklī $s' = \delta_{ext}(s)$.



1.3. att. DEVS darbības shēma (aizgūts no [111])

Modelēšanas procesu ir iespējams vienkāršot, ieviešot ieejas un izejas portu jēdzienu, un tādējādi atomārā modeļa (1.2) ieejas un izejas kopas iegūst šādu formu:

$$X = \{(p, v) \mid p \in IP, v \in X_p\}, \tag{1.3}$$

kur IP - ieejas portu kopa, bet X_p - ieejas vērtību kopa;

$$Y = \left\{(p,v) \mid p \in OP, v \in Y_p\right\}, \tag{1.4}$$

kur OP - izejas portu kopa, bet Y_p - izejas vērtību kopa.

Ieejas / izejas portiem ir centrālā loma notikumu apmaiņai starp DEVS modeļiem.

Klasiskajā DEVS formālismā tikai viens ieejas ports saņem vērtību ārēja notikuma gadījumā, bet paralēlā DEVS formālisma variantā vairākiem ieejas portiem ir iespējama vienlaicīga ieejas vērtību apstrāde, šo iespēju panākot ar kolekciju datu struktūru izmantošanu paralēlo atomāro modeļu ieejās.

DEVS atomāro modeļu jēdzienam principiāli līdzīgs ir agregāta jēdziens [119], kas definē sarežģītas sistēmas matemātisko funkcionēšanas modeli.

Saistītie modeļi

Atomāros DEVS modeļus ir iespējams apvienot vienotā sistēmā, izveidojot *saistītos* DEVS modeļus. Saistītais DEVS modelis definē, kādā veidā atomārie un/vai saistītie apakšmodeļi (modeļa komponentes) ir savstarpēji jāsavieno, lai izveidotu jaunu modeli. Saistītais modelis sastāv no modeļa komponentiem un saitēm starp tiem:

$$CM = \langle X, Y, M, EIC, EOC, IC \rangle, \tag{1.5}$$

kur $X = \left\{(p,v) \mid p \in IP, v \in X_p\right\}$ - ieejas portu un atbilstošo vērtību kopa;

Y=$\left\{(p,v) \mid p \in OP, v \in Y_p\right\}$ - izejas portu un atbilstošo vērtību kopa;

$M = \{M_d \mid d \in D\}$ - saistītā modeļa komponentu kopa M_d,

kur D - komponentu nosaukumu kopa;

$EIC \subseteq \{((CM, ip_{CM}), (d, ip_d)) \mid ip_{CM} \in IP, d \in D, ip_d \in IP_d\}$ - *ārējā ieeju saistība*, savienojot saistītā modeļa ārējās ieejas ar komponentu ieejām;

$EOC \subseteq \{((d, op_d), (CM, op_{CM})) \mid op_{CM} \in OP, d \in D, op_d \in OP_d\}$ - *ārējā izeju saistība*, savienojot saistītā modeļa ārējās izejas ar komponentu izejām;

$IC \subseteq \{((a, op_a), (b, ip_b)) \mid a, b \in D, op_a \in OP_a, ip_b \in IP_b\}$ - *iekšējā saistība*, kas savieno komponentu izejas ar citu komponentu ieejām.

Saistītie modeļi tiek pielietoti lielu sistēmu modelēšanai ar daudziem savstarpējā mijiedarbībā esošiem elementiem, nodrošinot iespējas izveidot ļoti lielu daudzlīmeņu sistēmu modeļus hierarhiski organizētā un strukturētā veidā. Atšķirībā no atomārajiem modeļiem, saistītie modeļi tiešā veidā nedefinē modelējamās sistēmas dinamiku. Saistītā modeļa dinamiku nosaka tā sastāvā esošo pakārtoto atomāro modeļu dinamika.

1.2.2. DEVS formālisma paplašinājumi

Lai gan DEVS formālisms ir universāls diskrētu dinamisku sistēmu modelēšanas ietvars, eksistē vairāki specializēti DEVS formālisma paveidi, kas pielāgoti noteiktām sistēmu klasēm un problēmu apgabaliem. Tas ir saistīts ar to, ka daudzas reālās pasaules problēmas nav iespējams pietiekoši detalizēti aprakstīt un pētīt viena noteikta formālisma ietvaros. Piemēram, ražošanas konveijera darbību precīzi nav iespējams modelēt tikai ar diskrētās vai nepārtrauktās imitācijas paradigmas palīdzību. Šeit ir nepieciešama kombinēta diskrētu notikumu un nepārtrauktās imitācijas metodika, kas nodrošina *multiformālismu modelēšanas* pieeju. Multiformālismu modelēšana ir sistēmu izveides un analīzes koncepcija, kas sistēmas stāvokļu pāreju funkcijas izsaka kā apvienotu vairāku formālismu funkciju [99, 111].

DEVS formālisma paplašināšanai un adaptēšanai konkrētam problēmapgabalam ir trīs iespējami virzieni [110]:

1. DEVS formālisma paplašināšana jaunas apakšklases veidā;
2. jauna formālisma izstrāde un tā pilnīga iekļaušana esošās DEVS koncepcijas ietvaros;
3. jauna uz DEVS nebalstīta formālisma izveide un DEVS mijiedarbības saskarnes izstrāde.

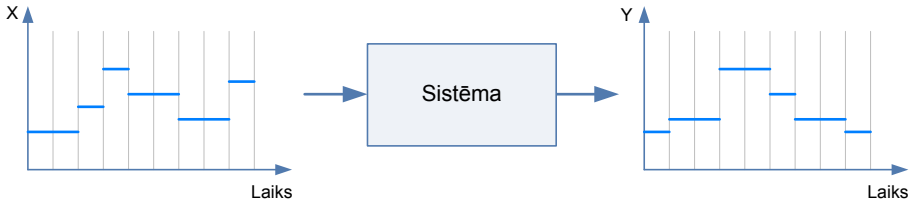
Tā kā šajā darbā izstrādātais formālisms ir izstrādāts uz četru DEVS formālisma paveidu - diskrēta laika sistēmu specifikācijas, diferenciālvienādojumu sistēmu specifikācijas, kombinētas diskrētu notikumu un diferenciālvienādojumu sistēmu specifikācijas un ģeometriski kinemātiskās diskrētu notikumu sistēmu sistēmu specifikācijas bāzes, tad ir nepieciešams šos formālisma paplašinājumus aprakstīt nedaudz detalizētāk. Taču, kā jau iepriekš minēts, izstrādāto DEVS formālisma paveidu skaits ir lielāks par šeit tuvāk apskatītajiem.

Diskrēta laika sistēmu specifikācija

Diskrēta laika sistēmas ir līdzīgas diskrētu notikumu sistēmām, jo tās izmanto diskretizētu laika bāzi. Taču diskrēta laika sistēmas atšķiras ar to, ka tās darbojas soļu režīmā ar fiksētu frekvenci - ieejas, stāvokļu un izejas informācija tiek apstrādāta fiksētos laika intervālos. 1.4. attēlā ir parādīts sistēmas ieejas un izejas izmaiņu atkarība laikā, kur redzams, ka ieejas vērtību apstrāde notiek diskrētos laika momentos, arī tad, ja nav ieejas vērtības izmaiņu.

Diskrēta laika sistēmas detalizēti apraksta diskrēta laika sistēmu specifikācija DTSS (angļu valodā *Discrete Time System Specification*) ar nepārtrauktu stāvokļu pārejas funkciju un diskrētu laika virzības funkciju, t.i., $t(x_n) = t(x_{n-1}) + \triangle t$. DTSS struktūra formālā veidā ir definējama sekojošā veidā:

$$DTSS = \langle X, Y, Q, \delta, \lambda, c \rangle,$$



1.4. att. DTSS ieejas un izejas izmaiņas laikā

kur X - ieeju kopa;

Y - izeju kopa;

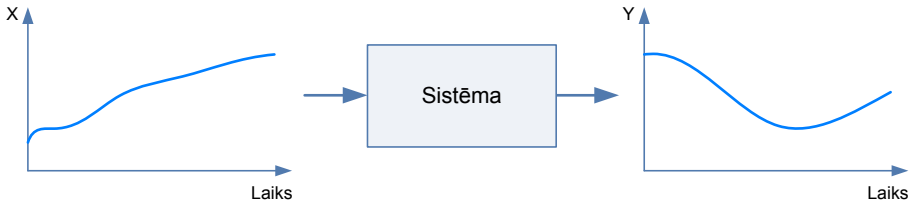
Q - stāvokļu kopa;

$\delta : Q \times X \rightarrow Q$ - stāvokļu pāreju funkcija;

$\lambda : Q \rightarrow Y$ vai $\lambda : Q \times X \rightarrow Y$ - izejas (Mūra vai Mīla tipa) funkcija.

Diferenciālvienādojumu sistēmu specifikācija

Nepārtraukta laika sistēmas parasti matemātiski apraksta ar diferenciālvienādojumu palīdzību, t.i. ar vienādojumiem, kas satur atvasinājumus. 1.5. attēlā ir parādīts nepārtraukta laika sistēmas piemērs, kas raksturo ieejas un izejas vērtību izmaiņas laikā.



1.5. att. DESS ieejas un izejas izmaiņas laikā

Diferenciālvienādojumu sistēmas no sistēmu teorijas viedokļa apraksta diferenciālvienādojumu sistēmu specifikācijas DESS (angļu valodā *Differential Equation System Specification*) formālisms. Diferenciālvienādojumu sistēmu specifikācija apraksta tradicionālos diferenciālvienādojumus ar nepārtrauktu laika virzības funkciju un nepārtrauktu stāvokļu pārejas funkciju. DESS struktūra ir definējama sekojošā veidā:

$$DESS = \langle X, Y, Q, f, \lambda \rangle,$$

kur X - ieeju kopa;

Y - izeju kopa;

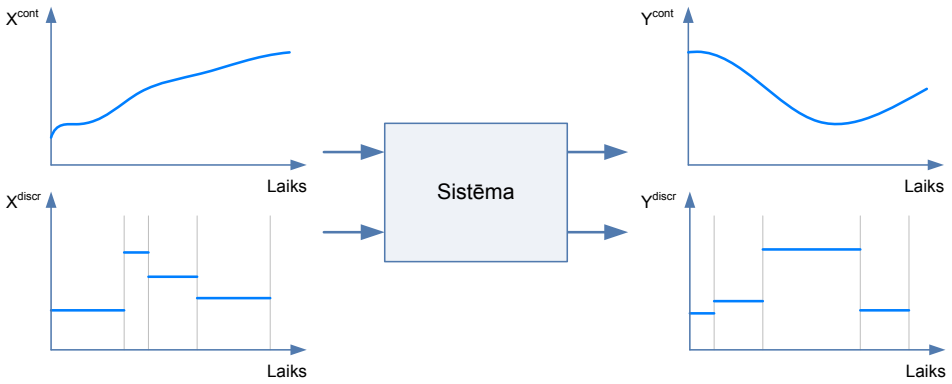
Q - stāvokļu kopa;

$f : Q \times X \rightarrow Q$ - izmaiņu ātruma funkcija;

$\lambda : Q \rightarrow Y$ vai $\lambda : Q \times X \rightarrow Y$ - izejas (Mūra vai Mīla tipa) funkcija.

Kombinēta diskrētu notikumu un diferenciālvienādojumu sistēmu specifikācija

Kombinēta diskrētu notikumu un diferenciālvienādojumu sistēmu specifikācija DEV&DESS ir dinamisku sistēmu imitācijas modelēšanas metodoloģija, kas balstīta uz DEVS formālismu un diferenciālvienādojumu sistēmu specifikāciju DESS, apvienojot nepārtraukto un diskrēta laika sistēmu darbību. 1.6. attēls vispārīgi ilustrē šādu sistēmu, kas satur gan nepārtrauktās, gan diskrētās ieejas un izejas.



1.6. att. DEV&DESS ieeju un izeju izmaiņas laikā

DEV&DESS formālisms definē komponentus un to interfeisus šādā formā [76]:

$$DEV\&DESS = \left\langle X^{discr}, X^{cont}, Y^{discr}, Y^{cont}, S^{discr}, S^{cont}, \delta_{ext}, C_{int}, \lambda^{discr}, f, \lambda^{cont} \right\rangle, \qquad (1.6)$$

kur X^{discr},Y^{discr}- diskrētu notikumu ieeju un izeju kopas;

X^{cont},Y^{cont} - nepārtraukto ieeju un izeju kopas;

$S = S^{discr} \times S^{cont}$ - stāvokļu kopa;

$\delta_{ext} : Q \times X^{cont} \times X^{discr} \rightarrow S$ - ārējā stāvokļu pārejas funkcija, kur

Q - summārā stāvokļu kopa;

$\delta_{int} : Q \times X^{cont} \rightarrow S$- iekšējā stāvokļu pārejas funkcija;

$\lambda^{discr} : Q \times X^{cont}$ - diskrētu notikumu izejas funkcija;

$\lambda^{cont} : Q \times X^{discr}$ - nepārtauktā izejas funkcija;

$f : Q \times X^{cont} \rightarrow S^{cont}$ - izmaiņu ātruma funkcija;

$C_{int} : Q \times X^{cont} \rightarrow Boolean$ - stāvokļu notikumu nosacījuma funkcija.

Kā redzams, DEV&DESS formālisms adaptē daudzus elementus no DEVS un DESS, kaut gan tas atšķiras ar divām izejas funkcijām - diskrēto un nepārtraukto. Papildus šai atšķirībai vēl viena atšķirība ir stāvokļu notikumu nosacījumu funkcija, kas tiek izmantota stāvokļu izmaiņu identificēšanai, izmainoties modeļa nepārtrauktajai daļai.

Tāpat kā ar bāzes formālismiem, arī DEV&DESS ir iespējams izveidot lielākas sistēmas no

atomāriem modeļiem, definējot sekojošu saistītā modeļa struktūru:

$$CM_{DEV\&DESS} = \left\langle X^{discr}, X^{cont}, Y^{discr}, Y^{cont}, D, \{M_d\}, EIC, EOC, IC, Select \right\rangle, \tag{1.7}$$

kur X^{discr} - diskrētu notikumu ieeju kopa;

X^{cont} - nepārtraukto ieeju kopa;

Y^{discr} - diskrētu notikumu izeju kopa;

Y^{cont} - nepārtraukto izeju kopa;

D - modeļa komponentu nosaukumu kopa;

$\{M_d\}$ - modeļa komponentu kopa, kur $d \in D$;

EIC - ārējo ieeju saites;

EOC - ārējo izeju saites;

IC - iekšējās saites;

$Select$ - modeļa komponentu prioritātes funkcija.

1.7.formulas aprakstīto struktūru, kas ir līdzīga citu DEVS formālismu saistītajiem modeļiem, sauc arī par *multiformālismu tīklu*. Atšķirība ir tikai tā, ka multiformālismu tīklā ir iespējams ietvert praktiski jebkuru citu formālismu, piedāvājot līdzekļus multiformālismu modelēšanai. Tādējādi DEVS, DTSS un DESS formālismus var uzskatīt par DEV&DESS speciālgadījumu [110].

GK-DEVS

GK-DEVS (angļu valodā *Geometric and Kinematic Discrete Event System Specification*) ir kombinēta uz DEVS formālisma balstīta imitācijas modelēšanas metodoloģija diskrētu un nepārtrauktu trīsdimensiju dinamisku multikomponenšu sistēmu modelēšanai [35]. GK-DEVS formālisms nodrošina ģeometrisko un kinemātisko modelēšanu, un ir definējams šādā formā:

$$GK-DEVS = \langle X, S, Y, \delta_{int}, \delta_{ext}, f, \lambda, ta, M, Z, Select \rangle, \tag{1.8}$$

kur $S = \left\langle S^{discr}, S^{cont} \right\rangle$ - secīgu stāvokļu kopa, sastāvoša no diskrēto stāvokļu kopas S^{discr}un nepārtraukto stāvokļu kopas S^{cont};

$\delta_{int} : S \to S$ - iekšējā pārejas funkcija;

$\delta_{ext} : Q \times X \to S$ - ārējā pārejas funkcija,

kur $Q = \{(s,e) \,|\, 0 \leqslant e \leqslant ta(s)\}$ - kopējā stāvokļu kopa;

$f : Q \to S^{cont}$ - atvasinājuma (izmaiņu ātruma) funkcija, kas nosaka nepārtraukto stāvokļu izmaiņu raksturu laikā atkarībā no tekošā modeļa stāvokļa;

M - apakšmodeļu kopa;

$Z \subseteq Y^H \times X^H$ - saistības kopa,

kur $Y^H = \bigcup_{m \in M} m.Y^H \cup Y$ - hierarhiska izejas notikumu kopa;

$X^H = \bigcup_{m \in M} m.X^H \cup Y$ - hierarhiska ieejas notikumu kopa;

Select - modeļa komponentu prioritātes funkcija.

1.2.3. Kvantētu stāvokļu sistēmas

Relatīvi jauna skaitlisko integratoru klase ir kvantētu stāvokļu integratori, kas balstās uz diskrētu stāvokļu izmantošanas pieeju nepārtrauktu sistēmu modelēšanas gadījumā, atšķirībā no tradicionālās diskrētā laika integratoru pielietošanas, kas plaši tiek izmantoti nepārtrauktu sistēmu imitācijas modelēšanā [95]. Kvantētu stāvokļu integratori balstās uz kvantēto sistēmu principa, kur jebkurš nepārtraukts ieejas un izejas lielums tiek sadalīts kvantos, radot iespēju nepārtrauktu sistēmu imitācijas modelēšanā pielietot DEVS formālisma principus. Kvants ir diskrēts mērījums, kas nosaka mazāko būtisko nepārtraukta lieluma izmaiņu amplitūdu.

Kvantētas sistēmas iespējams realizēt ar speciāla komponenta - kvantētāja palīdzību, kas, detektējot būtiskas ieejas lielumu izmaiņas, nepārtrauktu ieejas lielumu transformē diskrētā izejas lielumā. Šādas pieejas tālāka adaptēšana noved pie kvantēta integratora, kas ir standarta skaitliskā integratora funkcionāls analogs. Ir pierādīts, ka kvantētu sistēmu precizitāte ir līdzvērtīga diskrētajai pieejai, taču šo sistēmu būtiska priekšrocība ir mazākā ģenerēto notikumu ziņojumu skaitā imitācijas gaitā, kas uzlabo šo imitācijas modeļu darbības veiktspēju. Tas izskaidrojams ar to, ka izmaiņas modelī tiek apstrādātas tikai kvantizācijas līmeņu punktos.

Darbā [14] ir secināts, ka nepārtrauktu sistēmu modelēšanai reālā laikā vislabāk ir piemēroti fiksēta laika soļa tiešie vai daļēji netiešie integrēšanas algoritmi, bet mainīga soļa algoritmi ir atzīti par nepiemērotiem. Tradicionālos integrēšanas algoritmus ir problemātiski pielietot pārtrauktu sistēmu modelēšanai, kad ir grūti detektēt un apstrādāt sistēmas notikumus (tas ir atkarīgs no izvēlētā modelēšanas laika soļa). Kvantētu stāvokļu metodēm šādi iepriekš minētie klasisko algoritmu trūkumi nepiemīt, tādējādi tie ir laba izvēle kombinētu sistēmu imitācijai reālā laikā, ņemot vērā, ka parastos diferenciālvienādojumus ir iespējams aproksimēt ar diskrētā laika sistēmu palīdzību - izmantojot skaitliskās integrēšanas metodes. Tā kā diskrēta laika sistēmas ir DEVS speciālgadījums, tad no tā izriet, ka DEVS var aproksimēt arī nepārtrauktās sistēmas. DEVS pielietojuma iespējas nepārtrauktu sistēmu modelēšanā vēl vairāk uzsver fakts, ka eksistē speciālas skaitliskās metodes, piemēram, QSS, QSS2, QSS3, kuru imitācijas modeļus iespējams izveidot tikai ar DEVS formālisma palīdzību.

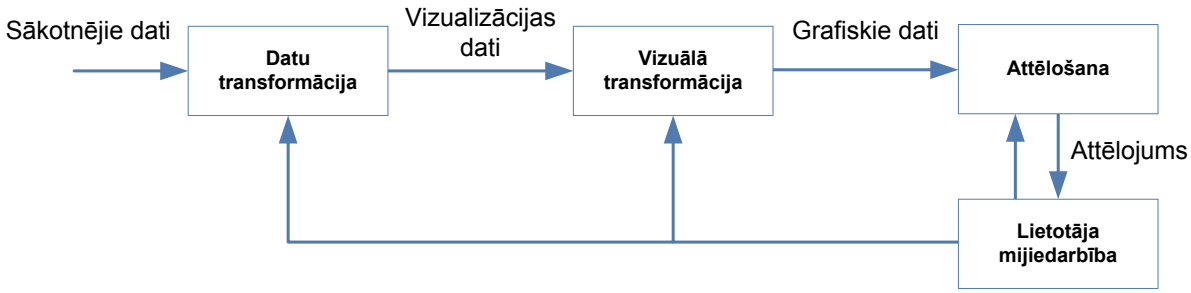
1.3. Vizualizācija imitācijas modelēšanā

Vairums imitācijas programmatūras līdzekļu piedāvā animācijas iespējas līdztekus ar modeļa rezultātu grafisko attēlojumu. Ir programmatūras sistēmas, kas nodrošina arī pašu imitācijas modeļu izstrādi grafiskā veidā, cenšoties tādā veidā vienkāršot izstrādes procesu. Taču eksistē arī pietiekoši daudz skeptisku viedokļu, kuros pausta doma, ka datorgrafikas iespējas imitācijas modelēšanā ir pārspīlētas. Protams, ar grafikas palīdzību nav iespējams padarīt nepareizu modeli par pareizu, bet eksistē vairāki veidi, kā vizualizācija un datorgrafika var kvalitatīvi ietekmēt un uzlabot imitācijas modelēšanas procesus.

Datorgrafikas un vizualizācijas jēdzienu saistībai ar modelēšanas un imitācijas jēdzieniem ir pieminami tādi termini kā "vizuālā interaktīvā imitācijas modelēšana" (angļu valodā *Visual Interactive Simulation*) [9] un "vizuālā interaktīvā modelēšana" (angļu valodā *Visual Interactive Modelling*) [101], taču tie galvenokārt netiek attiecināti uz imitācijas modelēšanu tās klasiskajā izpratnē, bet gan uz tādām informācijas tehnoloģiju jomām kā zinātniskā vizualizācija, datorizētā projektēšana, datorspēles un apmācības sistēmas.

1.3.1. Vizualizācijas jēdziens

Vizualizācija ir neatņemama mūsdienu dzīves sastāvdaļa, kur piemēri ir rodami visapkārt, sākot ar laika ziņu kartēm, beidzot ar reālistisku datorgrafiku izklaides industrijā. Tomēr, neraugoties uz vizualizācijas plašo pielietojumu, tai nav viennozīmīgas un noteiktas definīcijas, bet vizualizācijas jēdziena formulējums lielā mērā ir atkarīgs no tās pielietojuma sfēras un konteksta, taču vispārējā veidā var formulēt, ka vizualizācija ir datu vai informācijas pakāpenisks transformācijas process grafiskos attēlos (1.7. attēls). Saskaņā ar [114], vizualizācija ir datu,



1.7. att. Vispārējs vizualizācijas process

informācijas un zināšanu transformācijas process grafiskā attēlojumā ar mērķi nodrošināt datu analīzi, informācijas izpēti, informācijas izskaidrošanu, tendenču prognozēšanu, tēlu atpazīšanu u.c.. Savukārt datorgrafika ietver visus grafisko attēlu sintēzes, attēlojuma un manipulācijas aspektus [10].

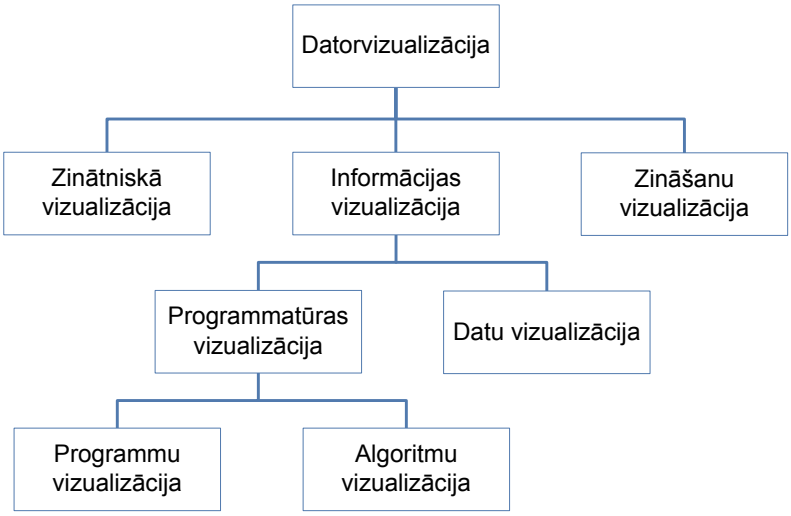
Lielākajai daļai eksistējošo vizualizācijas formulējumu un praktisko vizualizācijas sistēmu (piemēram, IRIS Explorer, Open Visualization Data Explorer, VTK) par formālo bāzi tiek izmantots Habera-Maknabba atsauces modelis [31], kas vizualizāciju apskata trīs secīgos pamatetapos:

1. sākotnējo datu transformācija - datu sagatavošana vizualizācijai;
2. datu vizuālā transformācija - skaitlisko datu transformēšana abstraktā ģeometriskajā attēlojumā;
3. grafiskā attēlošana jeb *renderēšana* (angļu valodā *Rendering*) - grafiskā attēla iegūšana.

Vizualizācijas kā formālas datorzinātnes disciplīnas pirmsākumi ir datējami ar 1987.gadu, kad tika publicēts raksts *"Visualization in scientific computing"* ("*Vizualizācija zinātniskajā skaitļošanā*") [60], kurā pirmo reizi tika lietots termins *zinātniskā vizualizācija*, kaut gan, protams, vizualizācijas pirmsākumi meklējami daudzus gadu desmitus un pat gadsimtus [29] pirms šīs publikācijas parādīšanās. Jēdziens *zinātniskā vizualizācija* sevī ietver lietotāja saskarnes nodrošinājuma, datu attēlojuma un apstrādes algoritmus, kā arī citu sensorisko informāciju, kā piemēram skaņu, ievadiekārtu datus. Lai uzsvērtu, ka vizualizācija tiek pielietota datorzinātnē, bieži vien tiek lietots termins *datorvizualizācija*, jo, protams, vizualizāciju iespējams pielietot arī bez datortehnoloģiju piesaistīšanas.

Datorvizualizācijas jomā ir attīstījušies vairāki pētījumu un praktiskās darbības virzieni, taču trūkst sistemātisku datorvizualizācijas klasifikācijas pētījumu, daudzus vizualizācijas jēdzienus lietojot kā pašsaprotamus, bet nesniedzot precīzu to definīciju un nenorādot to vietu salīdzinājumā ar līdzīgiem jēdzieniem. Par vienu no novatoriskākajiem vizualizācijas klasifikācijas mēģinājumiem var uzskatīt *vizualizācijas periodisko tabulu* [57], bet šī klasifikācija ir attiecināma uz vispārējo vizualizācijas jēdzienu ārpus datorzinātnes robežām, tādēļ tajā nav iekļauta tāda būtiska nozare kā zinātniskā vizualizācija. Šī darba ietvaros ir izveidota datorvizualizācijas klasifikācija (1.8. attēls), kas ir pielietojama arī imitācijas modelēšanas un vizualizācijas integrācijas vajadzībām.

Vizualizācijas aprakstīšanai līdztekus zinātniskajai vizualizācijai plaši izmanto arī terminu *informācijas vizualizācija* [114], kas vispārējā gadījumā definē plašāku jēdzienu nekā zinātniskā vizualizācija, jo informācijas vizualizācija ir pielietojama ārpus klasiskās zinātnes un inženierijas robežām, kā piemēram, finanšu, mārketinga vai uzņēmējdarbības datu vizualizācija. Galvenā zinātniskās vizualizācijas un informācijas vizualizācijas atšķirība ir tā, ka zinātniskās informācija nodarbojas ar telpisku fiziskas izcelsmes datu attēlošanu [58], bet informācijas vizualizācijas gadījumā attēlojamie dati ir abstrakti, un tiem trūkst telpiskās struktūras vai datu ģeometrija, tādējādi viens no informācijas vizualizācijas uzdevumiem ir šo telpiskumu ģenerēt. Savukārt informācijas vizualizācijai ir divi paveidi - *datu vizualizācija* un *programmatūras vizualizācija*,



1.8. att. Vizualizācijas kā datorzinātnes nozares vispārējā klasifikācija

datu vizualizāciju definējot kā kvantitatīvu datu vizuālo attēlošanu shematiskā veidā [57], bet programmatūras vizualizāciju attiecinot uz algoritmu [47] un programmu grafisko attēlojumu [92].

Relatīvi jauna vizualizācijas nozare ir *zināšanu vizualizācija*, kas tiek definēta kā vizuālā attēlojuma pielietošana zināšanu apmaiņai vismaz starp divām personām [12]. Lai noteiktu atšķirību starp informācijas vizualizāciju un zināšanu vizualizāciju, pirmām kārtām nepieciešams noteikt atšķirību starp jēdzieniem *informācija* un *zināšanas*, kur zināšanas ir praktiski vai teorētiski iegūta pieredze un izpratne.

Bieži vien pastāv neskaidrības jautājumā par to, kādas ir atšķirības, starp attēlveidošanu jeb attēlu apstrādi, datorgrafiku un vizualizāciju, tādēļ ir nepieciešams katru no šiem terminiem paskaidrot :

- *Attēlveidošana* (angļu valodā *Imaging*) jeb *attēlu apstrāde* (angļu valodā *Image Processing*) - darbības ar 2D attēliem, kas sevī ietver transformācijas metodes (rotēšana, mērogošana, nobīdīšana), informācijas iegūšanu, analīzi un uzlabošanu.

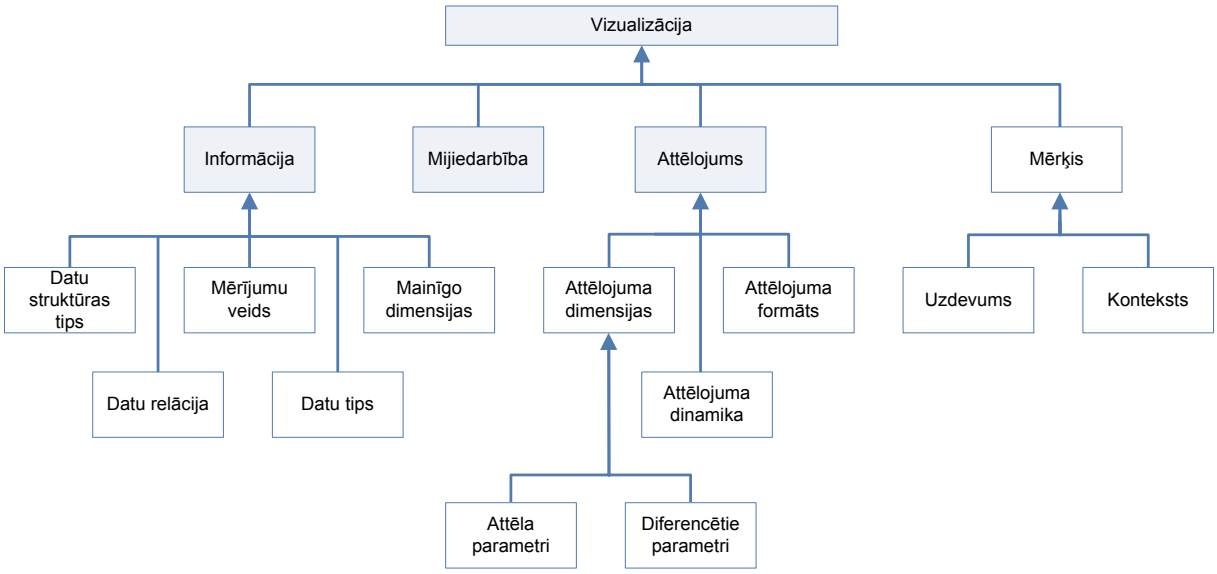
- *Datorgrafika* - attēlu veidošanas process, izmantojot datoru, ietverot gan 2D, gan 3D krāsošanas, zīmēšanas un renderēšanas metodes.

- *Vizualizācija* - datu izpētes, transformācijas un apskates process attēlu veidā (vai citā sensoriskā formā), lai gūtu ieskatu un izpratni par pētāmajiem datiem.

1.3.2. Vizualizācijas klasifikācija

Lai būtu iespējams izvēlēties piemērotas vizualizācijas metodes noteiktam mērķim un nolūkam, ir nepieciešams izveidot vizualizācijas klasifikācijas shēmu pēc noteiktām pazīmēm.

Tradicionāli vizualizācijas metodes tiek klasificētas pēc attēlojumo objektu un veidojamā attēla dimensiju kritērija. Attēli var būt gan simboliski (piemēram, 2D grafiki), gan arī reālistiski (piemēram, reālu objektu attēlojums). Protams, vienu un to pašu objektu ir iespējams attēlot dažādos veidos - piemēram, ģeogrāfiskās kartes ir iespējams attēlot, izmantojot gan 2D, gan 3D datorgrafiku. Eksistē arī dažādu citu veidu vizualizācijas klasifikācijas veidi, piemēram, klasifikācija datorgrafikas skatījumā vai klasifikācija informācijas skatījumā [105]. Šajā darbā vizualizācijas sasaistei ar imitācijas modelēšanu tiek piedāvāta vienota vizualizācijas taksonomija, kurā tiek ņemti vērā informācijas, mijiedarbības un attēlojuma aspekti (1.9. attēls).



1.9. att. Vienota vizualizācijas taksonomija

Informācijas aspekts vizualizācijā ir ļoti svarīgs, jo informācija ir nepieciešamo datu avots vizualizācijas procesā. Lai informācija būtu reāli pielietojama un noderīga, tai ir jābūt noteiktā veidā strukturētai un sistematizētai. Tādējādi informācijas - datu aspektā ir jāapskata izmantoto datu tips, datu relāciju struktūra. Vizualizācijas datus formāli apraksta šāda kopa:

$$VD = (G, A), \tag{1.9}$$

kur G - grafiskā objekta ģeometrija;

A - grafiskā objekta ģeometrijai un/vai topoloģijai piesaistīto datu atribūtu kopa.

Grafiskā objekta ģeometriju definē tā topoloģija saistībā ar virsotņu pozīciju kopu:

$$G = (K, V),$$

kur K - grafiskā objekta topoloģiju aprakstoša kombinatoriska struktūra;

$V = \{v_1, \ldots, v_n\}$ - n virsotņu pozīciju kopa $\mathbb{R}^3$ telpā, kas definē 3D grafiskā objekta formu.

Praktiskajā vizualizācijas sistēmu realizācijā eksistē daudz dažādu topoloģijas struktūras K realizācijas veidi, bet pamatā tie visi balstās uz topoloģijas teorijas *abstrakta simpliciāla kompleksa* jēdzienu [73, 116], kas definē virsotņu, šķautņu, trijstūru u.c. incidences. Abstrakts simpliciāls komplekss K sastāv no $\{1,\ldots,m\}$ virsotnēm kopā ar virsotņu apakškopām, kuras sauc par K *simpleksiem*. Simplekss σ^n tiek definēts kā izliekta figūra vektoru telpā $\sigma^n \subset \mathbb{R}$, kurai ir $(n+1)$ virsotnes $\sigma^n = (\alpha_0, \alpha_1, \ldots, \alpha_n)$ [121]. Abstrakta simpliciāla kompleksa K *ģeometrisko realizāciju*, kas definē topoloģijas telpas attēlojumu ģeometriskajā virsotņu telpā, sauc par *simpliciālu kompleksu* jeb *daudzskaldni*.

Darbā [85] ir piedāvāta efektīva datu struktūra vizualizācijas datu topoloģijas definēšanai, kas sastāv no topoloģiskiem elementiem - *šūnām*. Grafiskā objekta topoloģiju ir iespējams aprakstīt kā šūnu kopu:

$$K = (C, GR),$$

kur $C = \{c_i | \, i = 1,\ldots,n\}$ - šūnu kopa, kur katra šūna c_i sastāv no k virsotnēm;
$c_i = \left\{ \left(v_j | \, j = 1,\ldots,k\right), C_t \right\}$, definējot $k-1$ kārtas simpleksu; C_t - šūnas tips.

GR - {*Attēla dati, Taisnvirziena tīkls, Strukturēts tīkls, Nestrukturēti punkti, Poligonāli dati, Nestrukturēts tīkls*} - topoloģijas tips, kas nosaka topoloģijas ģeometrisko realizāciju.

Šūnas var būt gan lineāras, gan nelineāras [84]:

- lineāras šūnas: $C_l \subseteq C_t$ ={*Virsotne, Polivirsotne, Līnija, Polilīnija, Trīsstūris, Trīsstūru sloksne, Četrstūris, Pikselis, Daudzstūris, Četrskaldnis, Heksaedrs, Vokselis, Prizma, Piramīda*};

- nelineāras šūnas: $C_n \subseteq C_t$ ={*Kvadrātiska šķautne, Kvadrātisks trīsstūris, Kvadrātisks četrstūris, Kvadrātisks četrskaldnis, Kvadrātisks heksaedrs*}.

Visbiežāk vizualizācijā pielietotie datu atribūti ir skalāri, vektori, normāles, tekstūras koordinātes un tenzori, tādēļ vizualizācijas datu atribūtu kopu var definēt šādā formā:

$$A = \{\textit{Skalāri, Vektori, Normāles, Tekstūras koordinātes, Tenzori}\}.$$

Divdimensiju un trīsdimensiju vizualizācija

Kā iepriekš minēts, galvenā vizualizācijas ideja ir datu transformēšana grafiskā formā, kas tiek panākta ar datorgrafikas palīdzību. Datorgrafikai, līdzīgi kā vizualizācijai, eksistē dažādi klasifikāciju veidi, bet viens no pamatklasifikācijas parametriem ir attēlojamo objektu tips (dimensionalitāte) un attēlu veids. Grafiskie attēli var būt pilnībā simboliski (2D grafi) vai reālistiski (reālu objektu 3D attēlojumi), protams, neizslēdzot iespēju vienu un to pašu objektu grafiski attēlot dažādos veidos.

Jautājumā par to, kāds attēlojuma veids ir labāks - 2D vai 3D, nav viennozīmīgas un universālas atbildes. Vairākos pētījumos [90, 96] ir minēts, ka 2D skatījums ir salīdzinoši labāk piemērots noteikta objekta detalizētai attēlošanai, precīzai grafiskajai navigācijai vai attāluma noteikšanai. Darbā [96] ir apskatīti vairāki 3D attēlojuma sistēmu trūkumi un ir pieminēts, ka nav iespējama precīza pozicionēšana un navigācija, izņemot atsevišķus gadījumus, kas saistīta ar attāluma un leņķiskajiem kropļojumiem 3D perspektīvas z dimensijā. Tādējādi ir apgrūtinoši attēlot 3D telpiskos datus uz 2D datora ekrāna tādā veidā, lai vienlaicīgi precīzi tiktu attēlota grafiskās scēnas objektu forma un arī precīza telpiskā distance.

Arī 3D attēlojumam ir vairākas priekšrocības, salīdzinot ar 2D attēlojumu. Piemēram, 3D attēlojums ir piemērots pārskata gūšanai 3D telpā, 3D objekta formas uztverei un aptuvenai navigācijai. Trešās dimensijas izmantošana ļauj atrisināt vairākas problēmas, kas raksturīgas 2D vizuālajām modelēšanas sistēmām [51]:

- 2D grafiskā attēlojuma simbolisma nepilnības;
- ekrāna telpas ierobežojumi;
- modeļa estētiskie aspekti.

$2D$ un $3D$ telpā attēlojamās informācijas I_{nD} apjoms teorētiski ir izsakāms ar sekojošu vienādojumu [23, 103]:

$$I_{2D} = (I_{1D})^2 \qquad I_{3D} = (I_{2D})^{\frac{3}{2}} \tag{1.10}$$

Tomēr, praktiski realizējot vizuālās informācijas attēlošanu uz 2D datora ekrāna, vizuāli uztveramās informācijas apjoms ir ievērojami mazāks:

$$I_{3D} = C \times I_{2D}, \tag{1.11}$$

kur $C \leq 3.0$ ir konstants lielums.

Ja cilvēka vizuālo uztveri definē kā redzes funkciju $V(I)$ un uztvertās vizuālās informācijas apjomu jeb stimulu apzīmējot ar I, tad, piemēram, vienas reālas scēnas un tās reālistiskas datorimitācijas uztveres procesā vajadzētu eksistēt sekojošai sakarībai [22]:

$$V(I_{IM}) \approx V(I_{SC}), \tag{1.12}$$

kur I_{IM} - uztvertā informācija no datora attēla

I_{SC} - uztvertā informācija no reālā objekta

No sakarības 1.12 izriet:

$$I_{IM} \approx V^{-1} V(I_{SC}).$$

Taču redzes funkcija ir ļoti sarežģīta un nav invertējama, pie tam, dažādi stimuli var attēlot vienu un to pašu scēnu. Arī datora ģenerētajiem attēliem ir ierobežojumi, salīdzinot ar reālo optisko informāciju. Tas izskaidro vienādojumu 1.11, jo vizuālās informācijas grafiskajam attēlošanai ir nepieciešams ņemt vērā daudzus vizualizācijas fizikālos un cilvēka uztveres kognitīvos aspektus.

Tādējādi no visa iepriekš minētā var secināt, ka gan 2D, gan 3D vizualizācijai ir noteiktas priekšrocības un trūkumi. Kādu konkrēto veidu izvēlēties, ir atkarīgs no risināmās problēmas apgabala un konkrētās situācijas. Kā iepriekš minēts, 2D attēlojums ir labāk piemērots atsevišķiem konkretizētiem telpiskajiem uzdevumiem, kamēr 3D attēlojums ir izteiksmīgāks un sniedz labāku vispārējo priekšstatu par attēlojamo objektu. Tā kā 2D un 3D skatījums parasti tiek izmantots dažādiem mērķiem, tad abu skatījumu vienlaicīgai izmantošanai ir iespējamas priekšrocības [97] noteiktu uzdevumu veikšanai, kā piemēram, objektu savstarpējai orientēšanai un pozicionēšanai, tādējādi abu vizualizācijas veidu pielietošana viena otru neizslēdz, bet gan papildina. Tomēr ir jāņem vērā, ka noteikti grafiskie un vizualizācijas veidi un risinājumi ir jāpielieto tikai tad, ja tie ir piemēroti un nodrošina efektīvu informācijas attēlojumu.

1.3.3. Vizualizācijas renderēšanas sistēmu arhitektūra

Reālā laika vizualizācijas un datorgrafikas renderēšanas sistēmu izveidē plaši tiek izmantotas divas konceptuālas pamatpieejas:

1. scēnas grafs;
2. vizualizācijas konveijers.

Scēnas grafs

Scēnas grafs [7, 20, 24] ir vispārināta un datorgrafikā plaši pazīstama datu struktūra, kas sevī ietver grafiskās scēnas loģisko un telpisko struktūru. Grafisko scēnu ir iespējams modelēt kā statisku vienotu veselumu, vai arī kā no atsevišķiem diskrētiem komponentiem veidotu kopumu. Scēnas kā vienota veseluma modelēšana ir saistīta ar ģeometrisko objektu izveidošanu un izvietošanu noteiktās telpiskajās pozīcijās un ar noteiktu orientāciju. Šādi statiski strukturēti scēnas modeļi ir piemēroti un tiek izmantoti lielu grafisko datu kopu renderēšanai, vizualizācijai un apskatei gadījumos, kad nav nepieciešams veikt izmaiņas atsevišķos grafiskās scēnas elementos, vai arī, ja šīs izmaiņas ir nepieciešamas relatīvi reti. Savukārt, no diskrētu hierarhiski sakārtotu komponentu veidotas grafiskās scēnas ir piemērotas dinamisku izmaiņu (piemēram, pozicionēšana, orientēšana, mērogošana) veikšanai atsevišķos scēnas elementos. Šāda atsevišķu scēnas elementu vadība ir būtisks nosacījums animācijas vajadzībām.

Scēnas grafu formāli var definēt kā orientētu aciklisku grafu $G(N,E)$ [15, 24], kura virsotnes satur grafiskos objektus un to atribūtus, bet loki attēlo saites starp virsotnēm. Grafs sastāv no virsotņu kopas N un loku kopas E. Orientētā grafā lokus E veido sakārtoti virsotņu $(v,w) \mid v \in N, w \in N$ pāri. Saistība starp virsotnēm tiek realizēta formā "vecāki-bērni", balstoties uz objektu īpašību pārmantojamību. Dažāda līmeņa virsotnes attēlo dažādus abstrakcijas līmeņus. Scēnas grafs satur informāciju par vairākām attēlojamo objektu fiziskajām un telpiskajām īpašībām - pozīciju un orientāciju telpā, materiāla parametrus, tekstūras informāciju u.c.. Objektu ģeometrija, materiāla un tekstūras parametri ir nepieciešami telpiskā izkārtojuma attēlošanai.

Vizualizācijas konveijers

Vairums vizualizācijas programmatūras sistēmu izmanto datu plūsmu paradigmu, kur dati tiek apstrādāti *vizualizācijas konveijerā* (jeb *vizualizācijas tīklā*) [84, 106] - vizualizācijas konveijerā tiek padoti ieejas dati, tie tiek apstrādāti un rezultātā transformēti grafisko attēlu formātā. Vizualizācijas konveijers sastāv no elementiem, kas attēlo datus (*datu elementi*), no elementiem, kas datus apstrādā (*apstrādes elementi*), un no elementiem, kas norāda datu plūsmas virzienu (datu un apstrādes elementu saites). Apstrādes elementi veic ieejas datu apstrādi un izejas datu ģenerēšanu, un tie ir iedalāmi trīs apakšgrupās, atkarībā no to darbības raksturojuma [18, 86]:

- *avota elementi* - ģenerē izejas datus, nolasot ieejas datus no ārēja datu avota vai arī ģenerējot tos algoritmiski;
- *filtra elementi* - transformē ieejas datus pēc iepriekš definēta algoritma;
- *kartēšanas elementi* - pārveido vizualizācijas datus par grafiskajiem datiem.

Vizualizācijas konveijera priekšrocība ir tā caurlaides spējas neatkarība no elementu skaita, un tikai lēnākais elements (šaurā vieta) ietekmē konveijera caurlaides spēju. Eksistē daudz dažādās zinātnes un praktiskajās jomās plaši izmantotu vizualizācijas sistēmu, kas balstās uz datu plūsmu modeļiem, kā piemēram, AVS/Express [2], OpenDX [100], Visualization Toolkit (VTK) [84].

Lai iegūtu nepieciešamo grafisko rezultātu, vizualizācijas konveijeram ir jāapstrādā vizualizācijas dati, kur kopējais datu apstrādes process tiek saukts par vizualizācijas tīkla *izpildi*. Vizualizācijas tīkla izpilde iespējama *pieprasījuma vadāmā* vai *notikumu vadāmā* veidā [84]. Pieprasījuma vadāmā sistēmā vizualizācijas tīkls tiek izpildīts tikai tad, kad to pieprasa tīkla izejas elements, kā arī izpildīta tiek tikai tā tīkla daļa, kas ietekmē rezultātu. Notikumu vadāmā sistēmā katra apstrādes objekta vai tā ieeju izmaiņa izsauc atkārtotu tīkla izpildi. Notikumu vadāmā paņēmiena priekšrocība ir tā, ka tīkls vienmēr rezultātus korekti ataino atbilstoši notikušajām izmaiņām. Savukārt pieprasījuma vadāmās metodes priekšrocība ir iespēja akumulēt ieejas

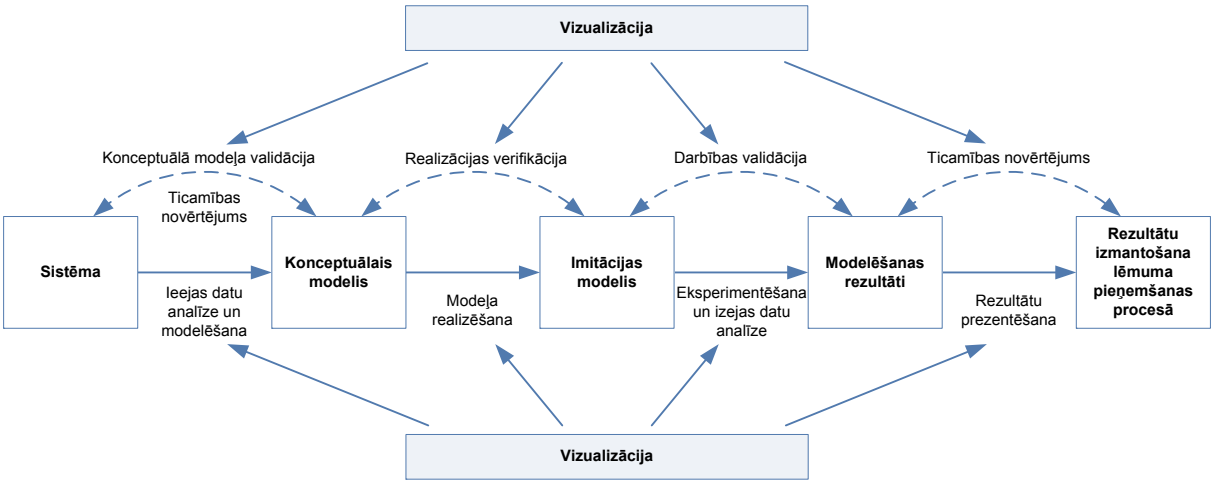
parametru izmaiņas bez to starpapstrādes nepieciešamības, tādējādi uzlabojot vizualizācijas sistēmas ātrdarbību.

Vizualizācijas tīkla izpildes gaitā ir nepieciešama apstrādes objektu sinhronizācija, un šim nolūkam eksistē divi vispārēji sinhronizācijas veidi: *netiešā izpilde* un *tiešā* izpilde [84]. Netiešās izpildes pieejas gadījumā katrs apstrādes objekts apstrādā tikai savas ieejas parametru izmaiņas. Galvenā netiešās izpildes priekšrocība ir šīs pieejas vienkāršība, jo katrs apstrādes objekts pats kontrolē savu darbību. Taču pie trūkumiem ir pieskaitāmas grūtības, kas pastāv šīs pieejas pielietošanā dalītā vizualizācijas apstrādē.

Tiešās izpildes gadījumā tiek izmantots centralizēts vizualizācijas konveijera *pārvaldnieks*, kas tiešā veidā koordinē visu apstrādes objektu darbību. Tiešās izpildes priekšrocība ir tā, ka vizualizācijas sinhronizācijas un modifikāciju analīzi pilnībā realizē pārvaldnieka objekts, ļaujot optimizēt vizualizācijas tīkla datu plūsmu apstrādi. Īpaši svarīga šī iespēja ir paralēlas un dalītas vizualizācijas apstrādes vajadzībām, vizualizācijas algoritmus izpildot ar vairākiem procesoriem vai datoriem. Tajā pašā laikā konveijera pārvaldnieka objekts tiešajā izpildē ir arī šīs pieejas trūkums, jo katrs apstrādes objekts ir atkarīgs no konveijera pārvaldnieka elementa.

1.3.4. Imitācijas modelēšanas un vizualizācijas integrācija

Imitācijas modelēšanas un vizualizācijas integrācija ir svarīgs aspekts vizuālo imitācijas modelēšanas sistēmu izstrādē, jo vizualizācija ir pielietojama visos imitācijas modelēšanas etapos, sākot ar konceptuālā modeļa izstrādi, beidzot ar modelēšanas rezultātu iegūšanu un to izmantošanu lēmuma pieņemšanas procesā (1.10. attēls), kalpojot par palīglīdzekli imitācijas modeļa verifikācijā un validācijā dažādos tā izstrādes etapos.

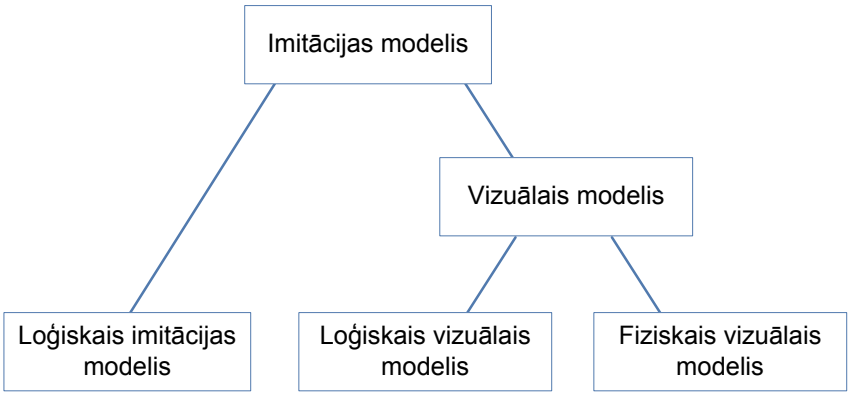


1.10. att. Imitācijas modelēšanas un vizualizācijas saikne

Vizualizāciju iespējams pielietot imitācijas modelēšanā gan statiskā, gan dinamiskā formā.

Imitācijas modeļu, eksperimentu vai rezultātu datus iespējams attēlot statiskā formā, t.i., tabulu vai diagrammu formā. Tāpat grafiskās attēlošanas procesā iespējama lietotāja mijiedarbība ar imitācijas modeli un tieša tā manipulācija.

Tā kā terminam "modelis" atkarībā no tā lietošanas konteksta var būt daudzas nozīmes [26], tad integrētā imitācijas modelēšanas un vizualizācijas skatījumā nepieciešams precizēt arī modeļa terminu. Modelis var būt gan modelējamās etalonsistēmas darbības loģikas un dinamikas, gan arī fizisko īpašību un telpiskā izskata attēlojums divās vai trīs dimensijās, tādēļ darba autors piedāvā klasifikāciju, kurā ir izšķirams loģiskā imitācijas modeļa un vizuālā modeļa jēdziens (1.11. attēls). Savukārt vizuālais modelis, atkarībā no attēlojuma konteksta, ir iedalāms loģiskā vizuālā modeļa un fiziskā vizuālā modeļa apakštipos.



1.11. att. Imitācijas modeļu klasifikācija integrētā skatījumā

Loģiskais imitācijas modelis kā jēdziens ir izveidots ar nolūku izcelt tā lietojumu modelējamās sistēmas darbības aprakstam, bet būtībā ir sinonīms 1.1.3.apakšnodaļā dotajai imitācijas modeļa definīcijai.

Vizuālais modelis - vizuālā formā definē imitācijas modeļa loģisko un/vai fizisko struktūru.

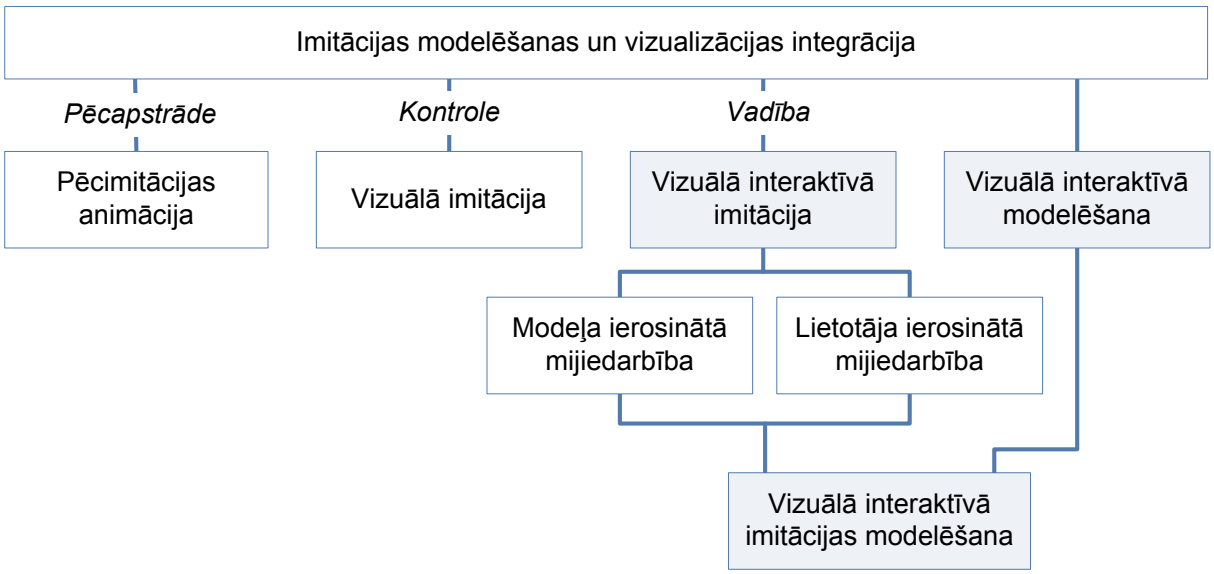
Loģiskais vizuālais modelis - modeļa diagrammas, piemēram, 3D vienkāršotas hierarhiskas aktivitāšu cikla diagrammas [56].

Fiziskais vizuālais modelis definē modelējamās sistēmas izkārtojumu un ģeometriju, kā arī tās statisko un dinamisko elementu vizuālos atribūtus.

Vizualizācijas pielietojuma mērķi ir atšķirīgi atkarībā no konkrētas imitācijas modelēšanas stadijas, kā arī no tā, kas ir konkrēta imitācijas modeļa mērķa auditorija - piemēram, imitācijas modelēšanas speciālistam vizualizācija var būt saistoša kā kognitīvs līdzeklis modeļa validācijas uzlabošanai un paātrināšanai, savukārt apmācību vai prezentāciju laikā vizualizācijas galvenokārt saistāma ar estētiskajem un reprezentatīvajiem aspektiem. Tādējādi dažādās imitācijas modelēšanas stadijās vizualizācijas pielietojums atkarībā no tās izmantošanas mērķa un auditorijas ir atšķirīgs:

- vizualizācija kā informācijas ieguves rīks;
- imitācijas modeļu verifikācijas palīglīdzeklis modelēšanas procesā;
- imitācijas modeļu validācijas palīglīdzeklis [49];
- analīzes rīks dažādu imitācijas laikā noritošo procesu izskaidrošanai;
- saziņas līdzeklis imitācijas modelēšanas speciālistiem un plānotājiem vai lietotājiem;
- reprezentācijas līdzeklis;
- apmācības līdzeklis.

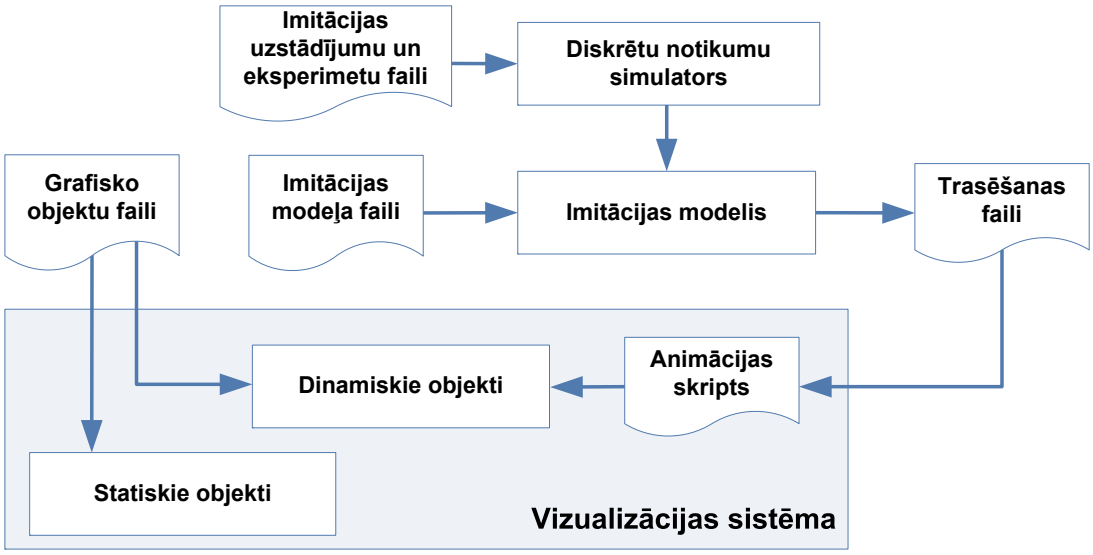
Galvenā vizuālās interaktīvās imitācijas motivācija ir vizualizācijas rīku un interaktīvo vadības rīku integrācija, ļaujot lietotājam novērot modeļa darbības gaitu un mijiedarboties ar to eksperimentu laikā. Modeļa, kā arī vizuālā attēlojuma parametrus un mainīgos ir iespējams modificēt eksperimentu veikšanas gaitā ar tūlītēju ietekmi uz imitācijas procesu. Atkarībā no lietotāja un modeļa mijiedarbības pakāpes ir izšķiramas trīs mijieadarbības jeb interakcijas klases: *pēcapstrāde, kontrole un vadība* [59], uz kā pamata iespējams izveidot imitācijas modelēšanas un vizualizācijas integrācijas veidu klasifikāciju (1.12. attēls).



1.12. att. Imitācijas modelēšanas un vizualizācijas integrācijas veidu klasifikācija

Imitācijas vidēs, kas nodrošina pēcapstrādes iespējas, lietotājam nav iespēju novērot imitācijas procesa starprezultātus. Pēcapstrādes sistēmās izejas dati tiek saglabāti failos vai datu bāzēs, no kurienes vēlāk šos datus var izmantot grafiskie pēcimitācijas laika līdzekļi (1.13. attēls). Tādējādi lietotāja mijiedarbība ir ierobežota tikai ar pēcapstrādes analīzes iespējām, un galvenais vizualizācijas veids šeit ir *pēcimitācijas animācija*. Pēcapstrādes imitācijas datu grafisko

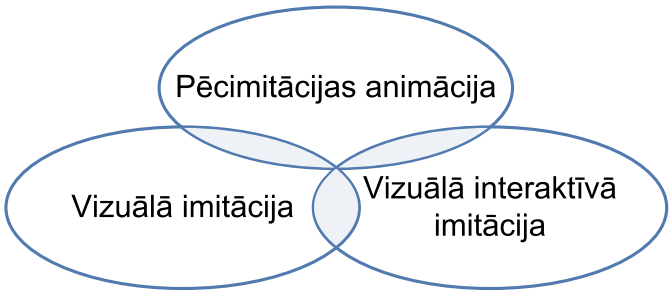
attēlošanu nodrošina tādas sistēmas kā Proof Animation [33] un ProModel [32]. Imitācijas vides ar kontroles iespējām tiek sauktas par *vizuālās imitācijas sistēmām*, un tajās tiek piedāvāti resursi imitācijas starprezultātu novērošanai, tādējādi lietotājam ir iespējas sekot eksperimenta gaitai ar dažādiem vizuālajiem līdzekļiem, piemēram, daudzveidīgs modeļa elementu grafiskais attēlojums, daudzlogu grafiskais režīms, statistikas grafiku ģenerēšana, animācijas resursu izmantošana. Lietotājs var arī vadīt eksperimentu tādā nozīmē, ka ir iespējas uzstādīt kopējo imitācijas laiku, definēt novērojamo parametru un mainīgo kopu, kā arī saglabāt un apstrādāt imitācijas modeļa starpstāvokļus. Tajā pašā laikā vizuālās imitācijas vides nepieļauj modeļa struktūras un parametru izmaiņas imitācijas eksperimenta izpildes gaitā. Vizuālās imitācijas iespējas nodrošina lielākā daļa moderno diskrētu notikumu un nepārtraukto imitācijas modelēšanas sistēmu, piemēram, Rockwell Arena [6, 43], AutoMod [1, 91], QUEST [8, 21], ProModel.



1.13. att. Tipiska pēcimitācijas animācijas sistēmas arhitektūra

Imitācijas vides, kas tiek klasificētas kā vadības interaktīvās sistēmas, piedāvā iespējas un līdzekļus lietotājam mijiedarboties ar modeli eksperimentu izpildes gaitā, tādējādi ietekmējot imitācijas modelēšanas rezultātus. Paralēli pēcapstrādes un kontroles iespējām lietotājam ir resursi modeļa parametru, mainīgo un atribūtu modificēšanai. Šādas imitācijas vides sauc par *vizuālajām interaktīvajām imitācijas sistēmām*. Ar vizuālajām interaktīvajām imitācijas sistēmām cieši saistīts un bieži vien kā sinonīms tiek lietots *virtuālās interaktīvās imitācijas* jēdziens, kas apzīmē interaktīvu imitācijas procesu, kurā lietotājam ir interaktīvas imitācijas modeļa vadības iespējas imitācijas laikā, izmantojot virtuālās realitātes saskarni. Eksistē imitācijas modelēšanas sistēmas, kurās ar interaktivitāti tiek traktētas iespējas imitācijas laikā modeļa darbību apstādināt, turpināt, iegūt un novērot statistiskos datus [91], taču tā ir tikai daļa no tām iespējām, ko var nodrošināt pilnīga lietotāja mijiedarbība, bet tas parāda, ka vizuālās imitācijas un interaktīvās

imitācijas robežas nav strikti nodalāmas (1.14. attēls).



1.14. att. Vizuālās imitācijas un vizuālās interaktīvās imitācijas robežas

Koncepcijas, kas sākotnēji ieviestas vizuālās interaktīvās imitācijas jomā, galvenokārt ir attiecināmas uz eksperimentēšanas, izejas datu analīzes un rezultātu prezentēšanas fāzēm, taču tikpat veiksmīgi tās ir iespējams pielietot visos citos imitācijas modelēšanas etapos, tajā skaitā imitācijas modeļa realizēšanas fāzē. Pēdējā laikā ir izstrādāts daudz imitācijas modelēšanas programmatūras sistēmu, piemēram, Rockwell Arena, kas satur vizuālus interaktīvos līdzekļus imitācijas modeļu izstrādei. Šo rīku galvenais uzdevums ir palielināt modeļa izstrādātāja produktivitāti, uzsvaru liekot uz imitācijas modeļu grafisko izstrādes procesu. Tādēļ šeit ir lietojams *vizuālās interaktīvās modelēšanas* jēdziens. Ja vizuālā interaktīvā imitācija ir orientēta uz modeļa lietotāja vajadzībām, tad vizuālā interaktīvā modelēšana ir pielāgota modeļa izstrādātāja prasībām. Jebkura vizuālā interaktīvā modelēšanas vide ietver vismaz trīs dažādas apakšsistēmas imitācijas modeļu izstrādei [101]:

1. modeļa ģenerators, kas ļauj modeļa izstrādātājam grafiski izveidot un vizualizēt modeli;
2. modeļa analizators, kas pārbauda modeļa korektumu;
3. modeļa translators, kas modeļa specifikāciju transformē izpildāmā imitācijas struktūrā.

Imitācijas modeļa izveidi iespējams sadalīt divās fāzēs: paša imitācijas modeļa apraksta izstrāde un imitācijas modelēšanas scenārija izveide. Eksistē daudz praktiski izstrādātu grafisko redaktoru, kas nodrošina vizuālu interaktīvo imitācijas modeļu izveidi. Daudzās sistēmās modeļus iespējams izstrādāt ar speciālām vizuālajām valodām. Ņemot vērā gan imitācijas datu, gan modeļa realizēšanas vizualizāciju, iespējams atrast analoģiju starp imitācijas modelēšanas valodām un vizuālajām programmēšanas valodām. Līdztekus imitācijas modeļa struktūras grafiskajai izveidei, lietotājam ir arī jādefinē modeļa loģika, izmantojot atbilsošu programmēšanas valodu. Šādu valodu galvenais uzdevums ir aprakstīt sistēmas komponenšu darbību laikā jeb dinamiku. Šim nolūkam ir izstrādātas dažādas grafiskās metodes, piemēram, Petri tīkli, aktivitāšu ciklu diagrammas [68, 69] un to varianti [51, 52, 53].

Vizuālās imitācijas modeļu izstrādes elementi

Imitācijas modeļu izveide vienotā modelēšanas - vizualizācijas vidē ir iedalāma, atkarībā no konkrētās realizācijas, divos secīgos vai paralēli veicamos etapos:

1. modeļa loģiskās struktūras definēšana;

2. grafiskā attēlojuma izveide.

Ar šiem diviem modelēšanas etapiem ir saistāms statiskās vizualizācijas jēdziens. Imitācijas modeļu statiskā vizualizācija tiek veikta uz grafiskā attēlojuma pamata, ko sauc par *grafisko izkārtojumu* [3]. Grafiskais izkārtojums definē ne tikai vizualizācijas struktūru, bet arī ir ļoti svarīgs elements modeļu izveidē. Veids, kādā tiek izstrādāts grafiskais izkārtojums, lielā mērā nosaka vizuālā imitācijas modeļa kvalitāti. Grafiskā izkārtojuma radīšanā, it īpaši, 2D attēlojuma gadījumā, visbiežāk tiek pielietoti tādi grafiskie pamatelementi, kā diagrammas, zīmējumi, ikonas, kartes, gleznas, fotogrāfijas, shēmas.

Modeļu darbības loģiskās struktūras vizualizācijai plaši tiek izmantoti dažādi diagrammu veidi (1.2. tabula). Kombinētu sistēmu modelēšanā vispiemērotākie ir Petri tīkli un vienotās modelēšanas valodas UML stāvokļu diagrammas, jo tās ir ērtas gan diskrētu, gan nepārtrauktu sistēmu aprakstīšanai. Modeļu vizualizācija diagrammu veidā ļauj relatīvi vienkārši attēlot komplicētas modelēšanas aktivitātes un procedūras.

1.2. tabula

Grafisko modeļu diagrammu veidi

Darbības diagrammas	Attiecību diagrammas
Datu / kontroles plūsmu diagrammas	Plūsmu diagrammas
Lēmumu koki	Objektu diagrammas
Dekompozīcijas diagrammas	Petri tīkli
Atkarību diagrammas	Stāvokļu pārejas diagrammas
Aktivitāšu diagrammas	Struktūras diagrammas
Bloku diagrammas	

Iepriekš aprakstītie grafiskā izkārtojuma elementi ir hierahiski strukturējami, veidojot hierarhisku izkārtojuma struktūru un līdz ar to sekmējot vizuālā imitācijas modeļa izstrādi.

Animācijas pielietojums imitācijas modelēšanā

Dinamiskā vizualizācija jeb animācija attēlo modelējamos objektus, kuri maina savus fiziskos vai loģiskos atribūtus un stāvokļus imitācijas procesā. Animācijas veikšanai tiek izmantoti modeļu izpildes moduļa izejas dati, taču būtiska iespēja ir veikt animāciju bez nepārtrauktas

saistības ar simulatoru. Tas ir lietderīgi diskrētu sistēmu modelēšanā, kad animācija tiek vadīta ar notikumu palīdzību, bet notikumu starplaikā grafiskie objekti tiek attēloti pēc noteikta animācijas algoritma. Arī 2D un 3D datu histogrammu un grafiku animācija, piemēram, rotācija un panoramēšana, var notikt neatkarīgi no citiem sistēmas moduļiem.

Dinamiskā vizualizācija izmanto grafiskā izkārtojuma elementus, kuriem ir piekārtoti dinamiskie loģiskā modeļa parametri. Modelēšanā, izmainoties šiem parametriem, piemēram, transporta līdzekļa atrašanās vietas koordinātēm, attiecīgi izmainās grafiskā elementa koordinātes, kas attēlo šo transporta līdzekli. Tādējādi rodas animācijas efekts.

Vislielāko vizuālo efektu dod tieši trīsdimensiju animācijas izmantošana, kas palielina informācijas apjoma uztveres iespējas un kvalitāti. Taču 3D grafisko modeļu izveidei ir nepieciešami lielāki izstrādāšanas darbietilpības un laika resursi, tomēr lielu un sarežģītu sistēmu modelēšanā tās izmantošanai ir nozīmīgi rezultāti. Dinamiskās vizualizācijas augstākais līmenis ir virtuālā realitāte, kas lietotājam ļauj maksimāli iejusties modelējamo objektu darbības vidē un dod vistiešākās mijiedarbības iespējas.

Imitācijas modelēšanas un vizualizācijas mijiedarbībā svarīgs aspekts ir laika faktors, kuru noteikti jāņem vērā imitācijas un vizualizācijas procesu sinhronizācijai. Imitācijā ir izšķirami divi tās pamatveidi:

- reālā laika imitācija;
- virtuālā laika imitācija.

Vizuālajā imitācijas modelēšanā izšķir statiskos un dinamiskos objektus [15, 51]. Statiskie objekti visā modelēšanas laikā paliek fiksētā stāvoklī, taču tie var saturēt kustīgus komponentus, kas to darbības gaitā pārvietojas, bet kustība tiek ierobežota statiskā objekta funkcionālajās vai fiziskajās robežās. Objekta funkcionālās robežas nosaka kā objektu aptverošu apgabalu, kurā viens vai vairāki objekta elementi var mainīt savu telpisko novietojumu. Statisku objektu piemēri ir noliktavas, stacionāri roboti, virpas u.c.. Dinamiskie objekti pieder pie objektu klases, kuri var brīvi pārvietoties to darbības laikā. Dinamiskie objekti var pārvietoties paši, tos var pārvietot ārējas iedarbes rezultātā, vai arī tie var saturēt vienu vai vairākus kustīgus komponentus. Dinamisku objektu piemērs ir automašīnas, pacēlāji, roboti.

Animācija tiek iedalīta divās kategorijās [65, 70]:

- tradicionālā animācija;
- atslēgkadru (angļu valodā *Key-Frame*) animācija.

Animācija sniedz iespējas lietotājam vizuāli novērot imitācijas modeļa darbības dinamiku un palielina modeļa darbības efektu, sniedzot pilnīgāku pārskatu par modelējamo sistēmu, tādēļ tā

var kalpot kā efektīvs imitācijas modelēšanas palīglīdzeklis dažādās ar imitācijas modelēšanu saistītās jomās un aspektos [81]:

- verifikācijas un validācija;
- rezultātu izpratnes gūšana;
- rezultātu apspriešana;
- uzticības iegūšana no skeptiski noskaņotiem cilvēkiem;
- imitācijas ticamības iegūšana.

Ļoti svarīgs faktors interaktīvai divpusējai modelēšanas un vizualizācijas sistēmu sasaistei ir animācijas un modelēšanas sinhronizācija. Pieņemsim, ka t ir sistēmas reālais laiks. Ar $t_s(t)$ apzīmēsim simulatora modelēšanas laiku, ar $t_a(t)$ - animācijas laiku. Animācijas laiku $t_a(t)$ raksturo tas, ka tas ir proporcionāls reālajam laikam ar diskretizāciju vienādos laika intervālos, tādējādi iegūst vienādojumu:

$$t_a(t) = c \cdot t,$$

kur c - animācijas laika proporcionalitātes koeficients.

Ja starpību starp modelēšanas un vizualizācijas laiku definē kā $\triangle t_{sa}(t) = t_s(t) - t_a(t)$, tad ir spēkā vienādojums:

$$t_s(t) = t_a(t) + \triangle t_{sa}(t).$$

Modelēšanas un vizualizācijas sistēmu sinhronizācijai ir izstrādātas vairākas metodes, kuras balstās uz reālā laika [63] un loģiskā laika sinhronizāciju [80, 93]. Sistēmās ar reālā laika sinhronizāciju simulatora un animācijas laiki ir cieši vienoti un bieži vien tieši proporcionāli reālajam laikam, un $\Delta t_{sa}(t) \to 0$. Diskrēto notikumu modelēšanā tiek pieņemts, ka stāvokļu maiņas imitācijas modelēšanas izpildes gaitā notiek momentāni, taču praktiski ir ir jāņem vērā laiks t_{sp}, ko datorsistēma patērē stāvokļu maiņas izkalkulēšanai. Reālā laika sinhronizācijas pielietošanas gadījumā ir jābūt spēkā nevienādībai:

$$t_{sp} < \triangle t_{sa}(t),$$

kas nozīmē, ka modelēšanas notikumu apstrādes laikam ir jābūt mazākam par modelēšanas un animācijas laika diferenci. Pretējā gadījumā, kas ir novērojams sarežģītu imitācijas modeļu izpildes gaitā, laiks t_{sp} var pieaugt tā, ka

$$t_{sp} > \Delta t_{sa}(t),$$

kas noved pie tā, ka animācijas procesā notiek nevēlamas aizkaves un pārtraukumi, gaidot uz imitācijas modelēšanas procesa izpildi.

Pielietojot loģiskā laika sinhronizāciju, parasti tiek izmantots centralizēts loģiskā modelēšanas un animācijas laika sinhronizācijas mehānisms. Imitācijas modelēšanas laika virzībā ir iespējamas nianses atkarībā no modelēšanas veida:

- diskrēta laika imitācijas modelēšana;
- diskrētu notikumu imitācijas modelēšana.

Diskrēta laika imitācijas modelēšanas sistēmās laika diskretizācija notiek vienādos laika intervālos, kas ļauj nodrošināt vienotu laika secības sinhronizāciju starp imitācijas modelēšanas un animācijas moduļiem. Diskrētu notikumu imitācijas modelēšanas sistēmās laika secību nosaka attiecīgie imitācijas notikumu laika momenti. Tādējādi diskrētu notikumu imitācijas modelēšanas laiks nav vienots ar animācijas laiku, un animācijas laiks atpaliek no modelēšanas laika.

Apskatot initegrētu imitācijas un animācijas procesu, ir jāpiemin tas, kādā veidā šī integrācija tiek nodrošināta. Visplašāk izplatītais mehānisms ir nepārtrauktā imitācijas un animācijas saistība, kas nosaka, ka grafiskā scēna vienmēr tiek renderēta katrā imitācijas solī (renderēšanas un imitācijas fāzes ir saistītas). Nepārtrauktajai imitācijas saistībai ir divi būtiski trūkumi:

- visu grafisko objektu pārbaude, neatkarīgi no tā, vai tie reāli ir iesaistīti tekošā imitācijas soļa apstrādē;
- grafisko objektu atlases frekvence ir vienāda visiem sistēmas objektiem un tā nav adaptējama konkrētu objektu vajadzībām.

Iepriekšminētās nepārtrauktās imitācijas un animācijas sistēmu saistības trūkumus ļauj novērst diskrētas saistības ieviešana [30] imitācijas un animācijas procesu integrācijai, kas nodala imitācijas fāzi no grafisko objektu apstrādes fāzes.

Lietotāja mijiedarbība

Praktiski visas komerciālās imitācijas modelēšanas sistēmas ir pieskaitāmas pie vizuālās imitācijas sistēmu klases, kas nozīmē, ka imitācijas procesa izpildes gaitā lietotājam ir tikai iespējams vizuāli novērot imitācijas izpildi, bet nav iespēju interaktīvi tajā iesaistīties. Taču interaktīvas mijiedarbības nodrošinājuma esamība imitācijas izpildes gaitā paver būtiskas

priekšrocības modeļa lietotājam - iespējas mainīt modeļa parametrus un uzvedību bez nepieciešamības apstādināt imitācijas ciklu un atsākt to atkal no jauna. Šāda iespēja visnoderīgākā ir imitācijas modeļa verifikācijas un validācijas laikā, kad tiek veikta modeļa parametru un struktūras noskaņošana. Tādējādi interaktīvā mijiedarbība ar modeli imitācijas procesa laikā ir uzskatāma par modeļa verifikācijas un validācijas palīglīdzekli.

Raksturojot lietotāja mijiedarbību ar imitācijas modelēšanas sistēmu, ir nepieciešams uzsvērt, ka personai, kas izstrādā imitācijas modeli, nav obligāti jābūt arī šī modeļa lietotājam. Pastāv pieņēmums [4], ka ir vēlams un pat nepieciešams nodalīt imitācijas modeļa izstrādātājus un lietotājus, pielietojot šiem šiem diviem lietotāju tipiem apzīmējumus *izstrādātājs* un *inženieris*. Šādai divu lomu - izstrādātāja un inženiera - izdalīšanai ir vairākas priekšrocības:

- inženierim nav detalizēti jāizprot izmantojamās programmatūras zemākā līmeņa arhitektūra, programmatūras inženierijas metodes un imitācijas modelēšanas paņēmieni;
- inženierim ir iespējas vairāk laika veltīt rezultātu iegūšanai un analīzei, nevis modeļa izstrādei;
- inženieris var pielietot savas profesionālās zināšanas noteiktā praktiskajā darbības sfērā;
- izstrādātājam ir iespējams pielietot savu uzkrāto programmēšanas pieredzi programmatūras izstrādē;
- izstrādātājam ir laiks un zināšanas robusta un konkrētām inženiera vajadzībām pielāgota risinājuma izstrādei;
- izstrādātājam ir iespējams ņemt vērā plašāka mēroga jautājumus, nevis fokusēties tikai uz specifiska projekta vajadzībām.

1.3.5. Prasības interaktīvai vizuālai imitācijas modelēšanas videi

1.3. tabulā ir sniegts salīdzinošs pārskats par vairākiem komerciāliem diskrētu notikumu un nepārtraukto sistēmu imitācijas programmatūras līdzekļiem, par salīdzināšanas kritēriju izvēloties to grafisko nodrošinājumu un iespējas. No šajā tabulā uzskaitītajām programmatūras sistēmām vizuālo interaktīvo imitāciju, t.i., iespējas mijiedarboties ar imitācijas modeli tā izpildes gaitā nodrošina tikai AnyLogic, Model Vision Studium, Powersim un iThink rīki. Taču visi šie programmatūras līdzekļi izmanto noteiktu specializētu pieeju imitācijas uzdevumu risināšanai. Piemēram, AnyLogic izmanto hibrīdu aģentu imitācijas modelēšanas metodi, Model Vision Studium - Harela stāvokļu pāreju diagrammas, bet Powersim un iThink - sistēmu dinamikas metodoloģiju. No uzskaitītajiem modelēšanas līdzekļiem 3D vizuālo interaktīvo

imitāciju piedāvā tikai Model Vision Studium. Tādēļ joprojām pastāv problēma, kas saistīta ar integrētas vizuālās imitācijas modelēšanas sistēmas izveidi, kurā lietotājam būtu brīvas iespējas izvēlēties modelēšanas realizācijas un abstrakcijas līmeni, kā arī tiktu nodrošinātas pilnvērtīgas 3D vizualizācijas un mijiedarbības iespējas.

1.3. tabula

Komerciālo vizuālo imitācijas modelēšanas sistēmu salīdzinājums

Programmatūra	Pēcimitācijas animācija	Vizuālā imitācija	Vizuālā interaktīvā imitācija	Vizuālā interaktīvā modelēšana
AnyLogic [122]	-	2D	2D	2D
Arena [6, 43]	-	2D	-	2D
AutoMod [1, 91]	3D	3D	-	3D
Plant Simulation (eM-Plant) [89]	3D	3D	-	2D
Enterprise Dynamics [37]	-	3D	-	2D
ExtendSim [36]	3D	3D	-	3D
Flexsim [27]	-	3D	-	3D
MATLAB / Simulink [95]	-	3D	-	2D
Model Vision Studium [117]	-	3D	3D	2D
Micro Saint Sharp [62]	-	3D	-	2D
QUEST [8, 21]	-	3D	-	2D
Powersim [75]	2D	2D	2D	2D
Proof Animation [33]	2D	-	-	-
ProModel [32]	2D	2D	-	-
ShowFlow (Taylor II) [38]	-	3D	-	2D
iThink [39]	-	2D	2D	2D
Witness [48]	-	3D	-	2D

Balstoties uz sarežģītu dinamisku sistēmu īpašību un to vizuālās projektēšanas īpatnību

analīzes pamata, ir iespējams formulēt šādas prasības interaktīvas vizuālās imitācijas modelēšanas sistēmas instrumentālajiem līdzekļiem:

1. Modelēšanas valodas prasības:

   (a) nepieciešamība izveidot vienotu sistēmteorētisku ietvaru, kas nodrošina gan imitācijas modelēšanas, gan interaktīvās vizualizācijas prasības;

   (b) uz izstrādātā formālisma izveidotajiem modelēšanas bāzes elementiem jābūt pēc iespējas vienkāršiem;

   (c) iespējas lietotājam modificēt modelēšanas bāzes elementus, vai arī tos modulārā veidā apvienot;

   (d) lietotāja iespējas brīvi izvēlēties nepieciešamo modeļa abstrakcijas līmeni.

2. Imitācijas eksperimentu izpildes prasības:

   (a) imitācijas rezultātu vizualizācija nevis pēc eksperimenta, bet tieši tā veikšanas gaitā;

   (b) iespējas interaktīvi mijiedarboties ar imitācijas modeli un ietekmēt imitācijas gaitu.

3. Integrētās vides prasības:

   (a) vienkārša un intuitīva lietotāja saskarne;

   (b) datu importa un eksporta iespējas;

   (c) 2D / 3D vizuālās vides nodrošinājums.

1.4. Kopsavilkums un secinājumi

Šajā promocijas darba nodaļā ir veikts vizuālo imitācijas modelēšanas pieeju, metožu un sistēmu salīdzinošs pētījums ar mērķi identificēt to izstrādē neatrisinātās problēmas un uzdevumus, kā arī vispārīgā līmenī definēt prasības interaktīvai imitācijas modelēšanas sistēmai.

Šīs nodaļas ietvaros paveiktais:

- pamatota sistēmpieejas nepieciešamība un efektivitāte imitācijas modelēšanā;
- ir analizēts uz sistēmpieeju balstītais DEVS formālisms un galvenie tā paveidi;
- aprakstīti vizualizācijas lietojuma un integrācijas principi imitācijas modelēšanā.

Sasniegtie rezultāti ir šādi:

- ir izstrādāta vizualizācijas klasifikācija, kas orientēta uz pielietojumu vizuālo imitācijas modelēšanas sistēmu teorētiskajā izpētē un praktiskajā realizācijā;
- ir identificētas priekšrocības, ko sniedz lietotāja interaktīvā mijiedarbība imitācijas procesa gaitā;
- vispārīgā līmenī ir formulētas prasības interaktīvai imitācijas modelēšanas sistēmai.

Galvenie secinājumi ir šādi:

- uz sistēmpieeju balstītais DEVS formālisms nodrošina universālu formālu bāzi un multiformālismu pieeju kombinētu diskrētu notikumu un nepārtrauktu sistēmu modelēšanai;
- vizualizācija, atkarībā no tās izmantošanas mērķa un auditorijas, ir efektīvs palīglīdzeklis dažādos imitācijas modelēšanas etapos:
  - vizualizācija kā informācijas ieguves rīks;
  - imitācijas modeļu verifikācijas palīglīdzeklis modelēšanas procesā;
  - imitācijas modeļu validācijas palīglīdzeklis;
  - analīzes rīks dažādu imitācijas laikā noritošo procesu izskaidrošanai;
  - saziņas līdzeklis imitācijas modelēšanas speciālistiem un plānotājiem vai lietotājiem;
  - reprezentācijas līdzeklis;
  - apmācības līdzeklis;
- interaktīvā vizuālā imitācijas modelēšana rada iespējas lietotājam mainīt modeļa parametrus un ietekmēt tā darbību imitācijas izpildes gaitā, ļaujot paātrināt imitācijas modeļu verifikācijas un validācijas procesu.

Nodaļā ir izvirzītas vispārīgas prasības imitācijas modelēšanas videi, kurai jāatbalsta uz sistēmpieeju bāzēta diskrētu notikumu un nepārtrauktu sistēmu imitācijas modelēšana interaktīvā vizuālā vidē.

Jaunais teorētiskais rezultāts šajā nodaļā ir izstrādātā vizualizācijas klasifikācija integrētā skatījumā ar imitācijas modelēšanu.

2. V-DEVS FORMĀLISMS

Apskatot integrētas pieejas iespējas imitācijas modelēšanā un vizualizācijā, var secināt, ka ir nepieciešams izveidot vienotu sistēmteorētisku ietvaru, kas nodrošina gan imitācijas modelēšanas, gan interaktīvās vizualizācijas prasības. Iepriekšējā nodaļā aplūkotie teorētiskie aspekti un problēmas, kas saistās ar imitācijas modelēšanas un vizualizācijas integrāciju, kalpo par pamatu šajā nodaļā piedāvātajam V-DEVS formālismam.

Tā kā 1.2. apakšnodaļā aprakstītā diskrētu notikumu sistēmu specifikācija (DEVS) ir hierarhisks, modulārs diskrētu notikumu sistēmu formālisms, visiem modeļiem definējot interfeisus ar ieejas un izejas portiem, tad modeļus iespējams izstrādāt no hierarhiski savienotiem apakšmodeļiem, kā arī ir iespējama neatkarīga un modulāra modeļu projektēšana un testēšana. Taču DEVS formālisms nemodificētā veidā nav piemērots integrētai interaktīvai imitācijas modelēšanai un vizualizācijai, kam pamatā ir divi iemesli:

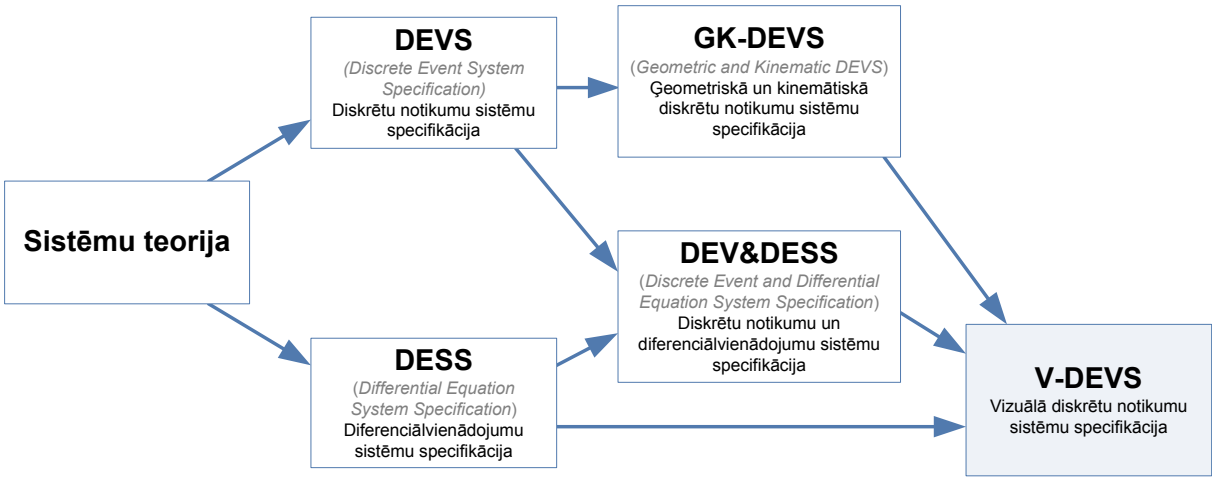
1. tradicionālā DEVS formālisma hierarhiskā struktūra ir atšķirīga no 3D vizualizācijai nepieciešamā grafiskā attēlojuma objektu hierarhijas scēnas grafa veidā (sk. 1.3.3.apakšnodaļu);

2. 2D/3D vizualizācijas dinamikai ir nepieciešams specifisks nepārtraukto stāvokļu sistēmas skatījums.

Tādēļ šajā darbā tiek piedāvāts jauns formālisms - *vizuālā diskrētu notikumu sistēmu specifikācija* (V-DEVS) (angļu valodā *Visual Discrete Event System Specification*), kas paredzēta diskrētu un nepārtrauktu sistēmu imitācijas modelēšanas un 2D/3D vizualizācijas integrācijai. Izstrādātais formālais aparāts ir izmantojams par teorētisku bāzi šādu iespēju realizācijai:

- diskrētu notikumu un nepārtrauktu sistēmu imitācijas modelēšana;
- reālā laika imitācija;
- interaktīva vizuālā imitācijas modelēšana.

V-DEVS formālisma pamatideja un diskrētu notikumu imitācijas modelēšanas struktūra ir aizgūta no klasiskā DEVS formālisma, bet nepārtraukto stāvokļu imitācijas elementi aizgūti no 1.2.2. apakšnodaļā aprakstītajiem DESS, DEV&DESS un GK-DEVS formālismu veidiem (2.1. attēls).



2.1. att. V-DEVS formālisma saikne ar citiem DEVS formālisma paplašinājumiem

1. pielikumā ir parādīta V-DEVS vieta vispārējā no darba [118] aizgūtā imitācijas modelēšanas klasifikācijā. 2.1. tabulā ir sniegts V-DEVS formālisma salīdzinājums ar citiem saistītajiem DEVS formālisma paplašinājumiem, kur redzamas V-DEVS formālisma jaunās iespējas, salīdzinot ar klasisko DEVS, DEV&DESS, DESS un GK-DEVS formālismu.

2.1. tabula
V-DEVS un tā bāzes formālismu salīdzinājums ("+" - iespēja ir nodrošināta, "-" - iespēja nav nodrošināta)

Formālisma iespējas	**V-DEVS**	GK-DEVS	DESS	DEV&DESS	Klasiskais DEVS
Diskrētu sistēmu modelēšana	+	+	+	+	+
Nepārtrauktu sistēmu modelēšana	+	-	-	+	-
Objektu kinemātikas animācija	+	+	-	-	-
Imitācijas modelēšanas un vizualizācijas integrācija	+	-	-	-	-
Interaktīvā imitācija	+	-	-	-	-

2.1. V-DEVS formālisma teorētiskie aspekti

V-DEVS formālisma teorētiskie aspekti balstās uz vienotu pieeju diskrētu notikumu un nepārtrauktu sistēmu modelēšanai, vizualizāciju traktējot kā nepārtrauktu procesu. Vizualizācijas un citu nepārtraukto lielumu modelēšanai tiek izmantota laika diskretizācija, bet diskrētajiem

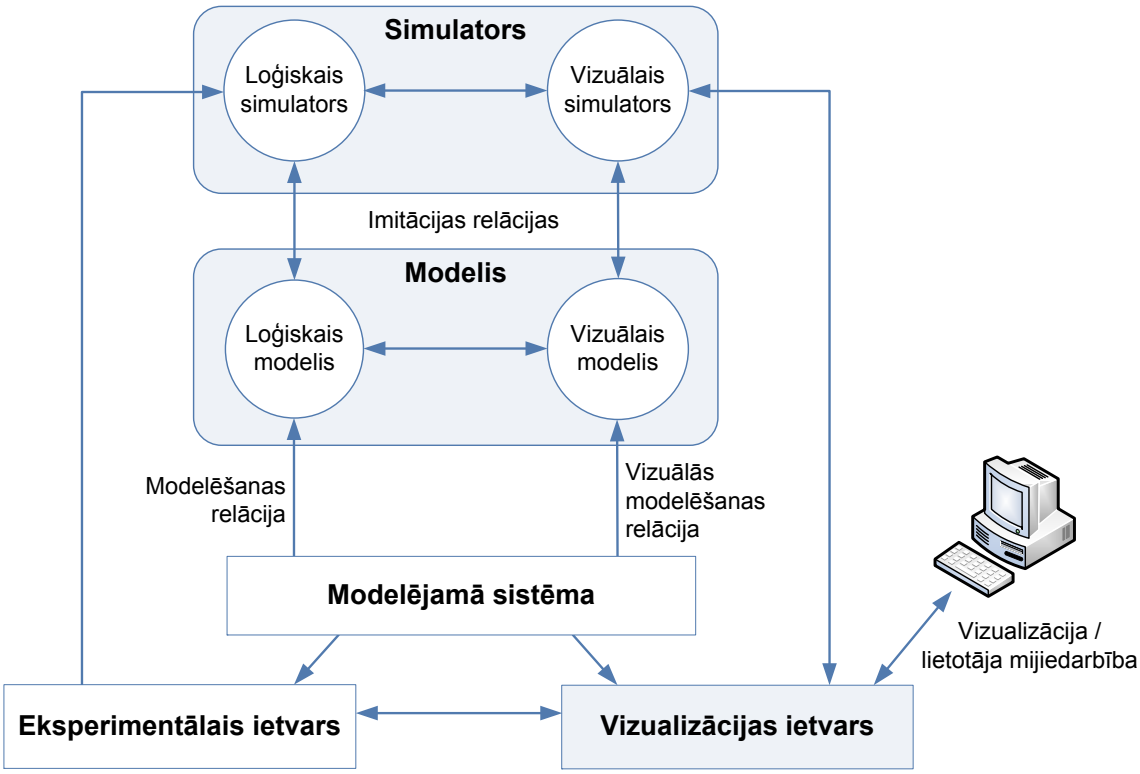
lielumiem izmantota imitācijas notikumu diskretizācija. Tajā pašā laikā V-DEVS formālisms ir universāli izmantojams un to ir iespējams pielietot dažādu dinamisku kombinētu sistēmu imitācijas modelēšanai, arī bez vizualizācijas pielietošanas.

2.1.1. Vizualizācijas ietvars

Līdzīgi kā modelēšanas un imitācijas ietvarā modelējamās sistēmas novērojumu vai eksperimentu nosacījumu uzdošanai ir definēts eksperimentālais ietvars, modelējamās sistēmas eksperimentālo nosacījumu definēšanai attiecībā uz imitācijas modelēšanas grafisko procesu ir nepieciešams ieviest vizualizācijas ietvara jēdzienu.

2.1. definīcija: *Vizualizācijas ietvars* ir imitācijas modelēšanas struktūra, kas definē nosacījumus, pie kuriem notiek modelējamās sistēmas vizuālā modeļa izveide, izpilde un lietotāja mijiedarbība imitācijas gaitā.

2.2. attēlā ir parādīta vizualizācijas ietvara vieta modelēšanas un imitācijas ietvarā mijiedarbībā ar eksperimentālo ietvaru, modelējamo sistēmu un simulatoru.



2.2. att. Vizualizācijas ietvars modelēšanas un imitācijas struktūrā

Vizualizācijas ietvara priekšrocība modelēšanas un imitācijas struktūrā ir iespējas nodalīt imitācijas modelēšanas procesu no grafiskā attēlojuma, definējot formālu bāzi vizuālajai imitācijai, kā arī vizuālajai interaktīvajai imitācijai un modelēšanai, atbilstoši 1.3.4. nodaļā aplūkotajai

klasifikācijai. Imitācijas modelēšanas un grafiskā attēlojuma nodalīšanai ir līdzīgas priekšrocības kā imitācijas modeļa atdalīšanai no simulatora imitācijas modelēšanas ietvarā: vienu un to pašu imitācijas modeli dotā eskperimentālā ietvara definētajās robežās ir iespējams izpildīt dažādos vizualizācijas ietvaros, tādējādi nodrošinot sadarbspēju un pārnesamību. Piemēram, konteineru termināla imitācijas modelēšanā vizualizācijas ietvars definē termināla fiziskā izkārtojuma vizuālo modeli, termināla transporta līdzekļu grafiskos modeļus, kā arī lietotāja saskarnes interaktīvos elementus, piemēram, ļaujot interaktīvi mainīt termināla transporta ātrumu, pārvietošanās maršrutus u.c..

Eksperimentālais ietvars un vizualizācijas ietvars pēc savas struktūras ir līdzīgi elementi, viens otru papildinot vienotā imitācijas nosacījumu definēšanā un eksperimentu izpildes procesā.

Vizuālā modeļa saikne ar vizualizācijas ietvaru iespējama divos veidos:

1. vizuālais modelis ir realizēts analoģiski imitācijas modelim pēc viena un tā paša formālā pamatprincipa
2. vizuālais modelis ir realizēts no loģiskā imitācijas modeļa atšķirīgā veidā kā neatkarīgs komponents, balstoties uz noteiktiem vizuālā modeļa izveides principiem.

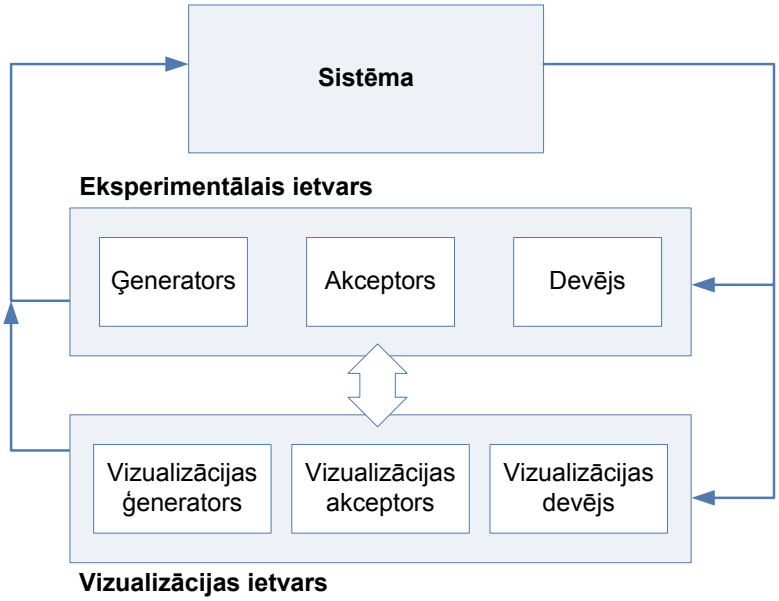
Vizualizācijas ietvars konceptuāli nodala imitācijas modelēšanas un vizualizācijas procesus, nodrošinot divus būtiskus imitācijas modelēšanas aspektus:

- izvēles iespējas imitācijas modeļa izstrādātājam izmantot vai neizmantot vizualizācijas līdzekļus modeļa izstrādes un imitācijas procesa gaitā;
- imitācijas procesa atsaiste no grafiskā attēlošanas procesa.

Tāpat kā eksperimentālā ietvara gadījumā, arī vizualizācijas ietvari konkrētai etalonsistēmai var būt vairāki, vai arī viens vizualizācijas ietvars var tikt attiecināts uz vairākām sistēmām.

Modelēšanas un imitācijas ietvara realizācijā eksperimentālais ietvars sastāv no trim komponentiem: ģeneratora, kas ģenerē sistēmas ieejas segmentus, akceptora, kas seko līdzi eksperimentālo nosacījumu izpildei un devēja, kas novēro un analizē sistēmas izejas segmentus. Analoģiski komponenti ir iekļaujami arī vizualizācijas ietvara sastāvā (2.3. attēls):

- *vizualizācijas ģenerators* - sistēmas vizuālo interaktīvo ieejas datu ģenerēšana;
- *vizualizācijas akceptors* - eksperimentālo nosacījumu vizuālais monitorings, ko veic imitācijas modeļa lietotājs;
- *vizualizācijas devējs* - grafiskā attēlojuma ģenerators, balstoties uz sistēmas izejas datiem.



2.3. att. Vizualizācijas ietvars un tā komponenti

2.1.2. Atomārā modeļa definīcija

Atomārs V-DEVS modelis ir definējams šāda korteža veidā:

$$AM_{V-DEVS} = \left\langle X, Y, S, \delta_{ext}, \delta_{int}^{discr}, \lambda^{discr}, ta^{discr}, \delta_{int}^{cont}, \lambda^{cont}, ta^{cont} \right\rangle, \tag{2.1}$$

kur $X = \left\langle X^{discr}, X^{cont} \right\rangle$ - diskrēto un nepārtraukto ieeju kopa;

$Y = \left\langle Y^{discr}, Y^{cont} \right\rangle$ - diskrēto un nepārtraukto izeju kopa;

$S = S^{discr} \times S^{cont}$ - secīgu stāvokļu kopa kā diskrētu stāvokļu S^{discr} un nepārtrauktu stāvokļu S^{cont} kopu Dekarta reizinājums;

$\delta_{ext} : Q \times X \to S$ - ārējā pārejas funkcija, kur

$Q = \left\{ (s,e) \mid s \in S, 0 \leq e \leq ta^{discr}(s) \right\}$ - summārā stāvokļu kopa;

$s \in S$ - stāvoklis, kādā sistēma atrodas kopš pēdējās stāvokļu pārejas;

e - laiks kopš pēdējās diskrētās stāvokļu pārejas;

$\delta_{int}^{discr} : S \to S$ - diskrētu notikumu iekšējā pārejas funkcija;

$\lambda^{discr} : S \to Y^{discr}$ - diskrētu notikumu izejas funkcija;

$ta^{discr} : S \to \mathbb{R}_{0,\infty}^{+}$- diskrētu notikumu laika ritējuma funkcija;

$\delta_{int}^{cont} : S^{cont} \to S^{cont}$ - nepārtrauktā iekšējā pārejas funkcija;

$\lambda^{cont} : S \to Y^{cont}$ - nepārtrauktā izejas funkcija;

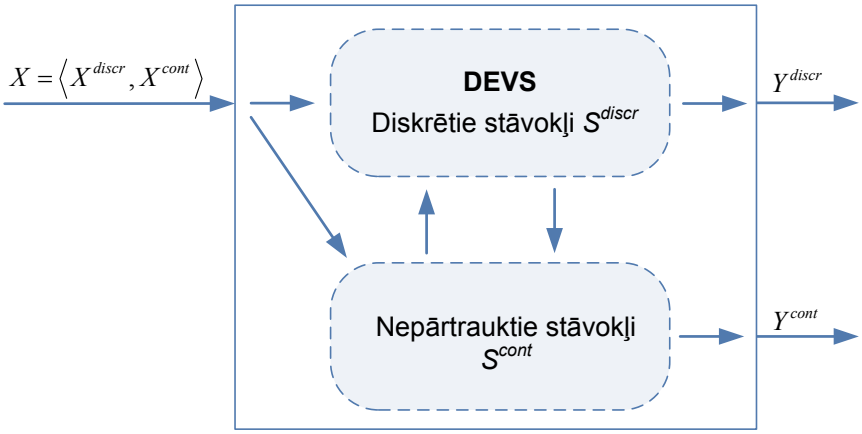
$ta^{cont} : S \to \mathbb{R}_{0,\infty}^{+}$ - nepārtrauktā laika ritējuma funkcija.

Nepārtraukto stāvokļu kopa var saturēt arī dinamiskās vizualizācijas vajadzībām nepieciešamos datus, tādējādi $S^{cont} = \left\langle VD, S^{cont-V} \right\rangle$, kur

VD - vizualizācijas kopa, kuru apraksta struktūra 1.9;

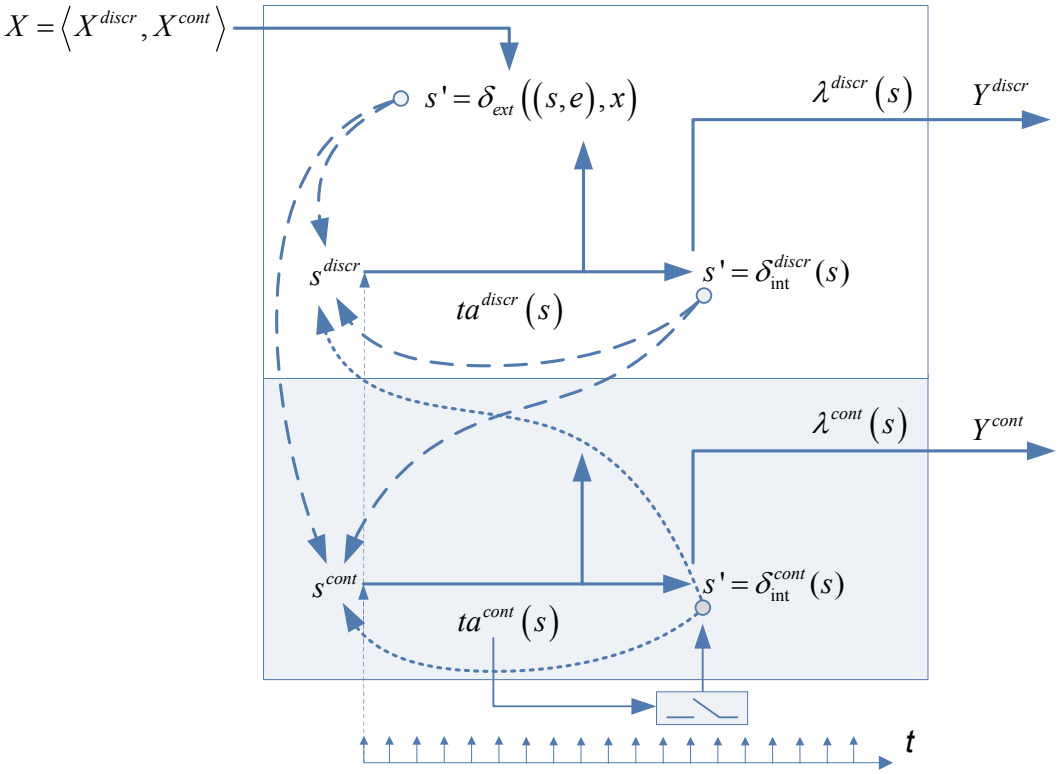
$S^{cont-VS} = S^{cont} \setminus VD$ - nepārtraukto stāvokļu kopa bez vizualizācijas kopas.

Atomārs V-DEVS modelis (2.1) apraksta gan diskrētus, gan nepārtrauktus procesus, paralēli apvienojot klasisko DEVS struktūru un modificētu DEVS struktūru (2.4. attēls).



2.4. att. V-DEVS modeļa struktūra

2.5. attēlā ir parādīta V-DEVS modeļa darbības dinamika.



2.5. att. V-DEVS modeļa dinamika

Saskaņā ar 1.1.1. apakšnodaļā apskatīto sistēmu teorijas terminoloģiju V-DEVS definētajai struktūrai ir šāds raksturojums:

- V-DEVS laika bāze ir reālu skaitļu kopa $\mathbb{R}$.
- Globālā stāvokļu pārejas funkcija $\Delta : Q \times \Omega \to Q$ ir definēta šādā veidā:

$$\Delta((s,e),x) = \begin{cases} \left(\delta_{int}^{discr}(s),0\right), & \text{ja } e = ta^{discr}(s),\ x^{discr} = \varnothing \\ \left(\delta_{ext}^{discr}\left(s,e,x^{discr}\right),e\right), & \text{ja } 0 < e \leq ta^{discr}(s),\ x^{discr} \neq \varnothing \\ \left(\delta_{int}^{cont}(s),0\right), & \text{ja } e = ta^{cont}(s),\ x^{cont} = \varnothing \\ \left(\delta_{ext}^{cont}(s,e,x^{cont}),e\right), & \text{ja } 0 < e \leq ta^{cont}(s),\ x^{cont} \neq \varnothing \end{cases} \tag{2.2}$$

- Globālā izejas funkcija $\Lambda : Q \to Y$ ir definēta šādi:

$$\Lambda(s,e) = \begin{cases} \lambda^{discr}(s), & \text{ja } e = ta^{discr}(s) \\ \lambda^{cont}(s), & \text{ja } e = ta^{cont}(s) \\ \varnothing, & \text{ja } e < ta^{cont}(s) \end{cases}$$

Ja formālisma nepārtrauktā daļa netiek izmantota, tad V-DEVS ir pilnībā ekvivalents klasiskajam DEVS formālismam $AM_{V-DEVS} \Longleftrightarrow AM_{DEVS}$, tādējādi kalpojot par formālu bāzi klasisko diskrētu notikumu sistēmu imitācijas modelēšanai. Ja $AM_{V-DEVS} \Longleftrightarrow AM_{DEVS}$, tad 2.1 struktūru ir iespējams pierakstīt saīsinātā veidā:

$$AM^{*}_{V-DEVS} = \left\langle X, Y, S, \delta_{ext}, \delta_{int}^{discr}, \lambda^{discr}, ta^{discr} \right\rangle. \tag{2.3}$$

V-DEVS nepārtrauktā daļa netiek izmantota pie sekojošiem minimālajiem nosacījumiem:

$$ta^{cont}(s) = +\infty \wedge \delta_{ext}^{cont}\left(s, x^{cont}, e\right) = \varnothing.$$

Līdzīgā veidā ir iespējams V-DEVS variants, kad netiek izmantota formālisma diskrētā daļa, kas formāli ir raksturojams sekojošā veidā:

$$ta^{discr}(s) = \infty \wedge \delta_{ext}^{discr}\left(s, x^{discr}, e\right) = \varnothing.$$

Šajā gadījumā notiek tikai nepārtraukto procesu imitācija.

Tā kā V-DEVS satur divas paralēli darbojošas struktūras, tad būtisks nosacījums V-DEVS darbībā ir laika sinhronizācija starp diskrētu notikumu un nepārtraukto V-DEVS daļu. Šo nosacījumu ir iespējams izpildīt definējot, ka ir jābūt spēkā nevienādībai

$$ta^{cont}\left(s^{cont}\right) \leq ta^{discr}\left(s^{discr}\right), \tag{2.4}$$

ja $ta^{discr}(s) \neq \infty \wedge \delta_{ext}^{discr}(s, x^{cont}, e) \neq \varnothing$.

V-DEVS formālismā nav tiešā veidā definēta vienlaicīgās pārejas funkcija $\delta_{conf}(s,x)$, kā tas ir paralēlajā DEVS formālismā (sk. 1.2.1. apakšnodaļu), taču no 2.2vienādojuma redzams, ka nav viennozīmīgi definēta modeļa uzvedība šādos gadījumos. Ja modelis saņem ārējos notikumus vienlaicīgi ar tā iekšējo stāvokļu pāreju laika momentā $ta(s)$, tad V-DEVS formālismā tiek izmantota vienlaicīgās pārejas funkcijas definīcija pēc noklusējuma, tāpat kā tas ir klasiskajā DEVS formālismā [111]:

$$\delta_{conf}(s,x) = \delta_{ext}(\delta_{int}(s), 0, x).$$

Tātad gadījumos, kad modelis saņem ārējos notikumus vienlaicīgi ar tā iekšējo stāvokļu pāreju, tad vispirms tiek izpildīta iekšējā pārejas funkcija, bet pēc tam - ārējā pārejas funkcija, ņemot vērā iekšējās pārejas funkcijas rezultātā ieguto jauno modeļa stāvokli.

$\Phi_q : \langle t_1, t_2 \rangle \rightarrow Q$ - stāvokļu trajektorijas segments laika intervālā $[t_1, t_2]$, kas ir saistīts ar summāro stāvokļu kopu $\left(s^{discr}, s^{cont}, e\right) \in Q$, ja ir spēkā šādi nosacījumi:

1. $\Phi_q(t_1) = \left(s^{discr}, s^{cont}, e\right)$ - sistēmas stāvoklis laika momentā t_1;
2. $\frac{d\Phi_q(t)}{dt} = f\left(\Phi_q(t)\right), t \in \langle t_1, t_2 \rangle$ - stāvokļu trajektorijas izmaiņu funkcija.

No tā izriet, ka sistēmas stāvokli jebkurā laika intervāla $[t_1, t_2]$ momentā t nosaka šāda sakarība:

$$\Phi_q(t) = \left(s^{discr}, s^{cont} + \int_{t_1}^{t} f\left(\Phi_q(\tau)\right) d\tau, e + t \right).$$

V-DEVS modeļa saskarne

DEVS modeļi satur funkcionālās un strukturālās zināšanas, t.i, informāciju par stāvokļu mainīgo vērtībām, funkcionalitāti, komponenšu dekompozīciju un šo komponenšu relācijām. Modeļu efektīvai pielietošanai, it īpaši to atkārtotai izmantošanai un automatizētai modeļu sintēzei ir nepieciešamas arī dažāda veida papildus zināšanas. Šīs zināšanas par modeļu vispārējiem parametriem un parametru virzību pa modeļu hierarhiju tiek sauktas par *taksonomiskajām zināšanām* [108].

V-DEVS modeļa (sk. vienādojumu 2.1) ieeju X un izeju Y strukturizēšanai ir iespējams pielietot DEVS *modeļa saskarnes* koncepciju [13], kas sakņojas no klasiskā DEVS formālisma zināmās portu izmantošanas modeļa ieeju un izeju definēšanā, papildus piesaistot tiem tipa un atribūtu kopas. Modeļa interfeisu I apraksta struktūra

$$I = \left\langle P^X, P^Y \right\rangle,$$

kur $P^X = \langle N^X, U^X, V^X \rangle$ - ieejas portu kopa,

$P^Y = \langle N^Y, U^Y, V^Y \rangle$ - izejas portu kopa,

kur N^Xun N^Y - ieejas un izejas portus identificējošās nosaukumu kopas,

$U^X \subseteq U$ un $U^Y \subseteq U$ - portiem piesaistīto vērtību tipu kopas,

$V^X = \{v_u^X \mid u \in U^X\} \subseteq V$ un $V^Y = \{v_u^Y \mid u \in U^Y\} \subseteq V$- ieejas un izejas portu vērtību kopas.

Kopa U satur visus V-DEVS modeļa definētos vērtību tipus, bet kopa V - visas modeļa definētās vērtības.

Imitācijas entītijas

Kopas $\{U,V\}$ elementi ir izmantojami ne tikai statiskai modeļa atribūtu piesaistei portiem, bet arī modeļa izpildes laikā dinamisko objektu definēšanai, šajā darbā ieviešot *imitācijas entītijas* jēdzienu

$$E = \{U^E \subseteq U, V^E \subseteq V\}, \tag{2.5}$$

kur U^E - imitācijas entītijas vērtību tipu kopa;

V^E - imitācijas entītijas vērtību kopa.

Imitācijas entītija ir konceptuāls imitācijas laika objekts visa veida dinamisku objektu modelēšanai, un ar to ir saistāmi modelējamās sistēmas plūsmu objektu, kā piemēram, pircēji, detaļas u.t.t.. Imitācijas entītija pēc konceptuālās nozīmes ir līdzīga daudzās diskrētu notikumu imitācijas modelēšanas sistēmās lietotajam transaktu jēdzienam [43]. Tajā pašā laikā imitācijas entītija kā imitācijas laika objekts atšķiras no imitācijas modelēšanā lietotā vispārējā entītijas jēdziena [61], kas apzīmē jebkuru imitācijas modeļa izveides gaitā apskatāmo sistēmas objektu. Imitācijas entītijām ir noteikti specifiski raksturlielumi:

- imitācijas entītija attiecībā uz sevi ir pasīvs objekts - tā nevar mijiedarboties pati ar sevi;
- imitācijas entītija attiecībā uz citiem modeļa elementiem ir aktīvs objekts - tā ir dinamisks objekts, kas ieejas notikumu veidā piedalās imitācijas modeļa izpildē.

V-DEVS formālisma ietvaros iepriekš ieviestais entītijas jēdziens ir paplašināts, lai nodrošinātu :

$$E_{V-DEVS} = \{U^E, V^E, AM_G, t_c\},$$

kur AM_G - atomārs V-DEVS modelis entītijas vizuālā objekta darbības imitācijai;

t_c - imitācijas entītijas izveides laiks.

2.1.3. Saistītā modeļa definīcija

V-DEVS saistīto modeļu struktūra ir līdzīga diskrēto notikumu specifikācijā dotajai saistītā modeļa definīcijai (1.5), bet galvenā atšķirība šeit ir tā, ka ieejas un izejas vērtību kopas ir diskrētu un nepārtrauktu vērtību apvienojums. V-DEVS saistīto modeli formāli apraksta šāda struktūra:

$$CM_{V-DEVS} = \langle X, Y, M, EIC, EOC, IC \rangle, \tag{2.6}$$

kur $X = \left\langle X^{discr}, X^{cont} \right\rangle$ - diskrēto un nepārtraukto ieeju kopa;

$X = \left\langle Y^{discr}, Y^{cont} \right\rangle$ - diskrēto un nepārtraukto izeju kopa;

$M = \{M_d | d \in D\}$- modeļa komponentu kopa,

kur D - komponentu nosaukumu kopa;

$EIC \subseteq CM_{V-DEVS}.X \times M.X$ - ārējo ieeju saites, kas savieno saistītā modeļa ārējās ieejas ar komponentu ieejām;

$EOC \subseteq M.Y \times CM_{V-DEVS}.Y$ - ārējo izeju saites, kas savieno komponentu izejas ar saistītā modeļa ārējām izejām;

$IC \subseteq M.Y \times M.X$ - iekšējās saites, kas savieno modeļa komponentu izejas ar citu komponentu ieejām.

V-DEVS struktūra ir līdzīga arī 1.2.2. apakšnodaļā aprakstītajai DEV&DESS saistītā modeļa struktūrai, ar to atšķirību, ka V-DEVS formālisms neizmanto *Select* funkciju.

Tāpat kā klasiskajā DEVS struktūrā, arī V-DEVS struktūrā nav atļautas tiešās atgriezeniskās saites, t.i., nav atļauta saistītā modeļa izejas portu savienošana ar tā paša modeļa ieejas portiem.

2.1.4. V-DEVS modeļu imitācija

Ieviešot vizualizācijas ietvara koncepciju, detalizētāk ir jāapskata arī simulators kā imitācijas modeļa izpildes elements. Simulatora realizācijai saistībā ar eksperimentālo un vizualizācijas ietvaru var būt divi iespējamie pamatvarianti:

1. universāls simulators, kas izpilda gan loģisko, gan vizuālo modeli;
2. divi realizācijas neatkarīgi, bet savstarpēji sinhronizēti simulatori loģiskā un vizuālā modeļa izpildei - loģiskais simulators un vizuālais simulators.

2.2. definīcija: *Loģiskais simulators* ir imitācijas izpildelements, kas izpilda un vada loģiskā imitācijas modeļa darbību.

2.3. definīcija: *Vizuālais simulators* ir imitācijas izpildelements, kas izpilda un vada vizualizācijas procesa norisi.

2.4. definīcija: *Universālais simulators* ir imitācijas izpildelements, kas izpilda un vada gan loģiskā, gan vizuālā modeļa darbību, apvienojot loģiskā un vizuālā simulatora funkcionalitāti.

Gan universālajam, gan atsevišķu loģiskā un vizuālā simulatora kombinācijai ir savas priekšrocības un trūkumi. Universālā simulatora priekšrocība ir iespējā nodrošināt precīzāku sinhronizāciju starp imitācijas un dinamiskās vizualizācijas procesiem. Taču universālā simulatora trūkums ir pārāk cieša iespējamā modeļa loģikas un vizuālās apstrādes sasaiste. Šo minēto trūkumu ir iespējams novērst, ļauj novērst divu savstarpēji sinhronizētu loģiskā un vizuālā simulatoru kombinācijas izmantošana. Atsevišķu simulatoru izmantošanā nozīmīga priekšrocība ir arī tā, ka paveras relatīvi vienkāršas iespējas izmantot dažādas simulatoru praktiskās realizācijas un pārslēgties starp šiem realizāciju variantiem, vai arī izmantot tos vienlaicīgi, ar nosacījumu, ka simulatori realizē vienotu simulatora interfeisu.

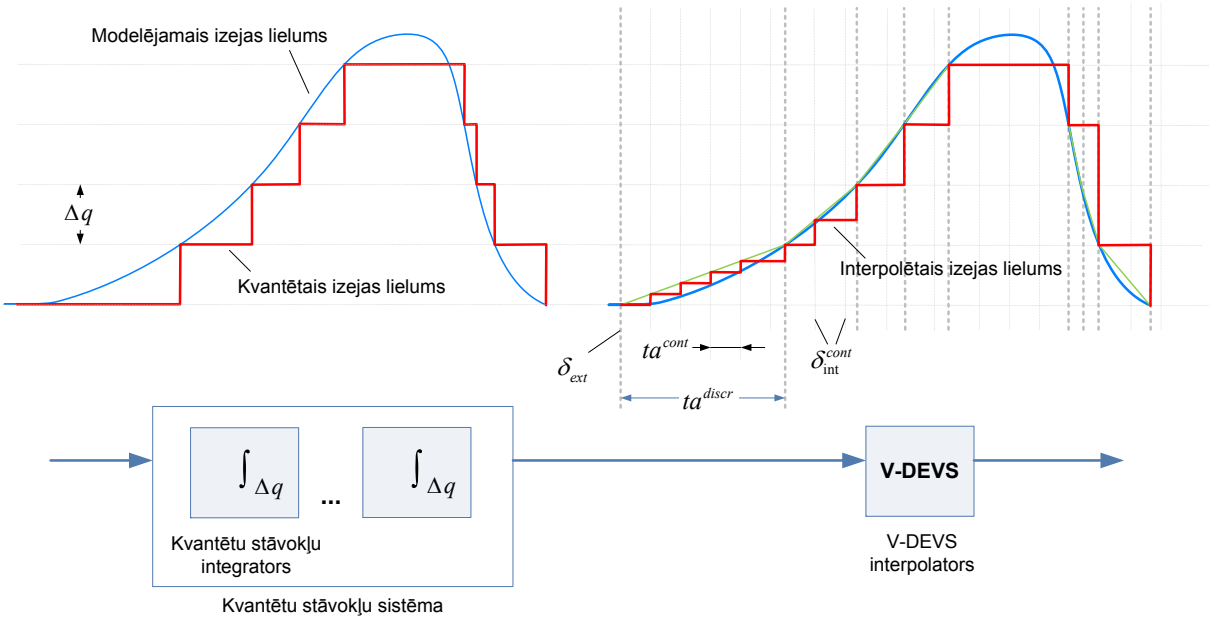
2.2. Kvantētu stāvokļu sistēmu imitācijas modelēšana

1.2.3. apakšnodaļā ir aplūkots fakts, ka DEVS formālisms nodrošina formālu bāzi nepārtrauktu sistēmu imitācijai ar diskrētiem līdzekļiem, pielietojot kvantētu stāvokļu algoritmus. Kvantētu stāvokļu algoritmu izmantošana ļauj samazināt nepieciešamo imitācijas soļu skaitu, vienlaicīgi nodrošinot, ka neviens modelējamajā sistēmā būtiskais notikums netiek palaists garām. Taču darba gaitā, izstrādājot V-DEVS vizualizācijas konveijeru, un veicot praktiskus eksperimentus kvantētu stāvokļu imitācijas modeļu integrācijā ar vizualizāciju ir novērota problēma, ka ne vienmēr tas kvantēšanas solis $\triangle q$, kas ir adekvāts imitācijas eksperimentu vajadzībām, ir vienlīdz piemērots dinamiskās vizualizācijas jeb animācijas izpildei, jo modeļa izejas lielumi var mainīties pa relatīvi lieliem diskrētiem soļiem. Animācijas vajadzībām ir nepieciešams, lai modeļa izejas lielumi mainītos vienmērīgi, radot imitācijas nepārtrauktības efektu. Dotās problēmas risināšanai šajā darbā tiek piedāvāta V-DEVS interpolatora koncepcija, kas efektīvi ļauj sasaistīt kvantētu stāvokļu modelēšanu un vizualizāciju.

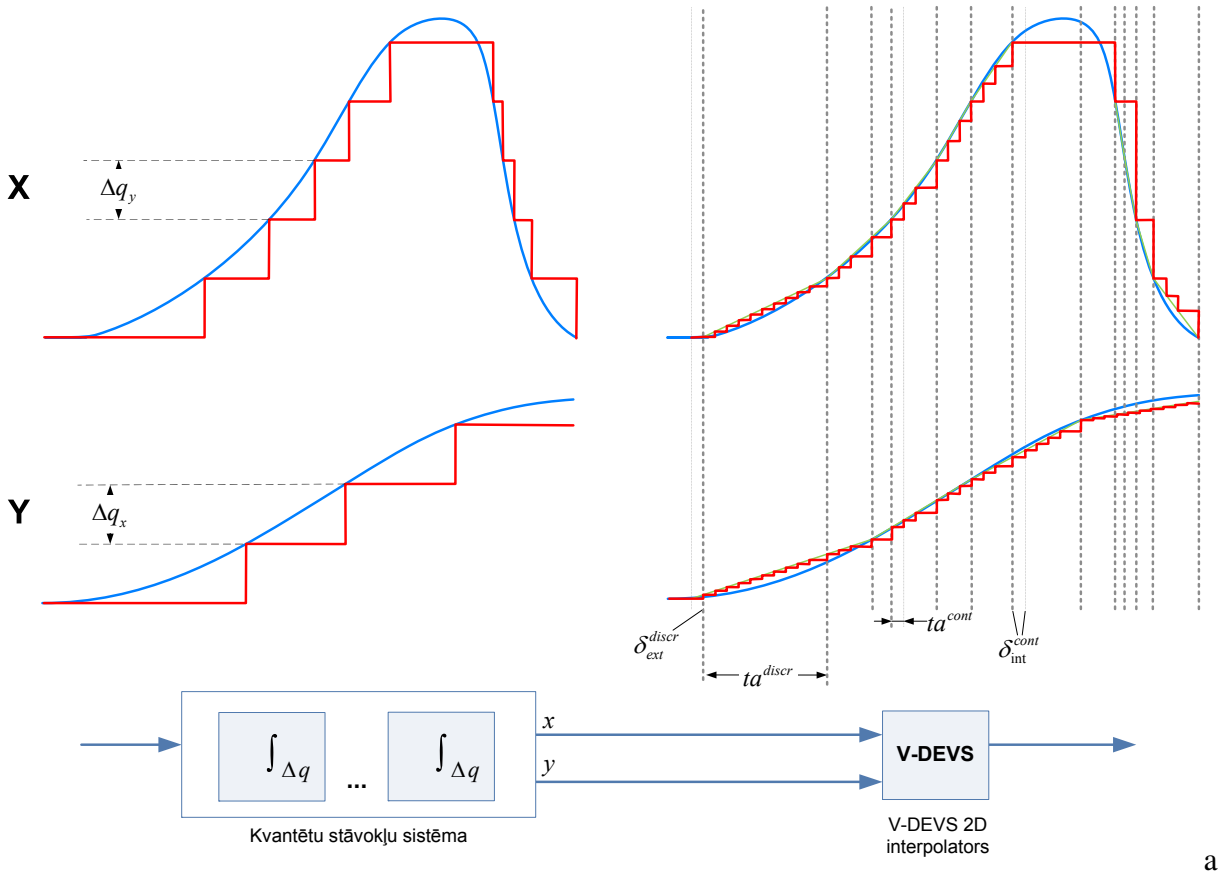
2.6. attēlā ir parādīta V-DEVS interpolatora koncepcija, kur kreisajā pusē ir attēlota kvantētu stāvokļu sistēma, bet labajā pusē - šīs sistēmas izejā pievienotais V-DEVS interpolators. Imitācijas procesa kvantēto stāvokļu sistēmas izejas lielumi mainās ar diskrētu mainīga lieluma soli $\triangle q$ laikā ta^{discr}. Interpolatora uzdevums ir šīs diskrētās izmaiņas transformēt vienmērīgā diskrēta laika ta^{cont} izmaiņu plūsmā. Lielums ta^{cont} šajā gadījumā nosaka renderēšanas laika intervālu, definējot, cik bieži tiek veikta attēla ģenerēšana.

Interpolācijas uzdevums kļūst sarežģītāks, ja modeļa izejā tiek apstrādāti divi vai vairāki savstarpēji saistīti lielumi, piemēram, veicot fizikālu dinamisku sistēmu imitāciju. Vairāku saistītu lielumu interpolācijas gadījumā tā ir jāveic sinhronizēti, lai starp interpolējamo lielumu

vērtībām nebūtu vērojama nobīde laikā, ņemot vērā renderēšanas periodu ta^{cont}. 2.7. attēlā shematiskā veidā ir parādīta 2D interpolatora darbības shēma.



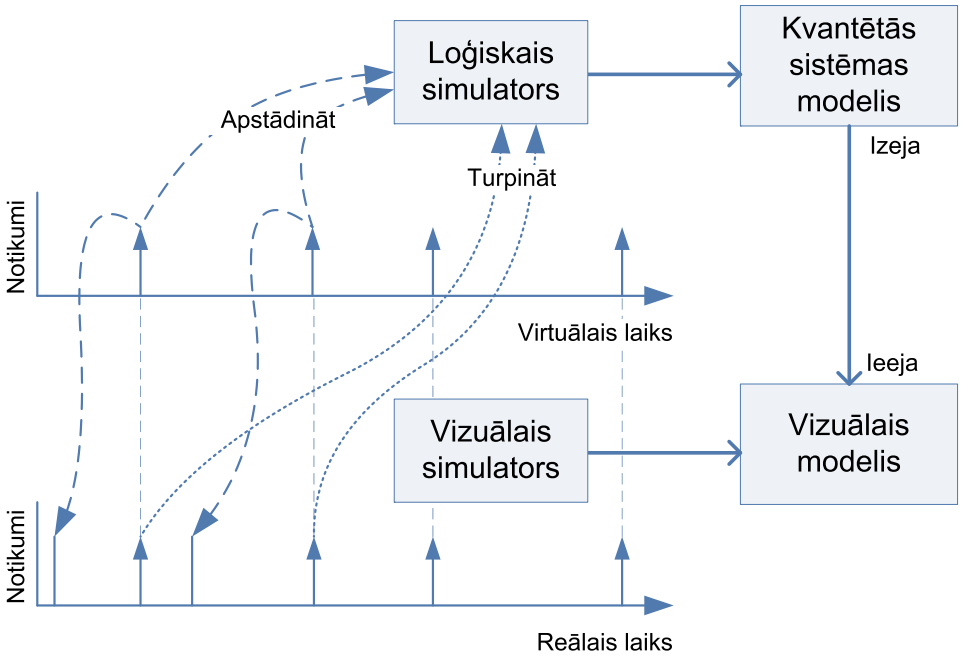
2.6. att. V-DEVS interpolatora darbības shēma



a

2.7. att. V-DEVS 2D interpolatora darbības shēma

Lai būtu iespējams izpildīt interpolāciju starp divām dažādām imitācijas notikumu vērtībām, tad jebkurā tekošā notikuma $e_n = (v,t) \in V \times T$ brīdī ir nepieciešams zināt dotā notikuma vērtību $v \in V$ un laiku $t \in T$, kā arī nākošā notikuma e_{n+1} vērtību un laiku. Tas nozīmē, ka simulatoram ir jāizdara vismaz viens imitācijas cikls uz priekšu nākotnē attiecībā pret tekošo notikumu, lai uzzinātu nākamo notikumu. Tā kā integrētā imitācijas modelēšanas un vizualizācijas sistēmā tiek izmantots mērogots fiziskais laiks ct ($c > 0$), tad tiešā veidā iepriekšminēto prasību nav iespējams izpildīt. Kā dotās problēmas risinājums šeit ir pielietojami divi paralēli simulatori - viens simulators loģiskajam modelim, bet otrs simulators - vizuālajam modelim. Divu simulatoru izmantošana ļauj darbināt katru no tiem savā laika mērogā, bet, protams, integrētas imitācijas nodrošināšanai nepieciešams šos simulatorus savā starpā sinhronizēt. Loģiskā modeļa simulators darbojas virtuālajā imitācijas laikā, tātad, cik vien ātri iespējams, bet vizuālais simulators darbojas reālā laikā, imitācijas lietotājam nodrošinot animētu modeļa darbības attēlojumu. Loģiskā un vizuālā simulatoru sinhronizācija ir iespējama tādējādi, ka loģisko simulatoru pēc noteikta notikumu skaita nepieciešams apstādināt, un pēc vajadzības atkal turpināt imitācijas procesu. Loģiskā simulatora apstādināšanas un turpināšanas procesu var nodrošināt vizuālais modelis. No tā izriet, ka kontrolēt simulatora darbību nepieciešams ne tikai eksperimentālajam ietvaram, bet arī pašam simulatora vadītajam imitācijas modelim. Loģiskā simulatora variantā modelis ietekmē savu bāzes simulatoru, bet vizuālā simulatora vadītais modelis iedarbojas uz loģisko simulatoru. 2.8. attēlā ir parādīts loģiskā un vizuālā simulatoru sadarbības piemērs.



2.8. att. Loģiskā un vizuālā simulatora sinhronizācija kvantētu stāvokļu sistēmu imitācijas modelēšanā

Loģiskais simulators sāk kvantētās sistēmas modeļa izpildi, līdz sistēmas izejā tiek ģenerēts

imitācijas notikums. Šis notikums nonāk vizuālā modeļa ieejā, kurš realizēts tādā veidā, lai varētu apstādināt / turpināt loģiskā simulatora darbību. Saņemot ieejas notikumu, ieejas vizuālais modelis dod komandu loģiskajam simulatoram apstādināt savu darbību.

2.2.1. Kvantētu stāvokļu interpolatora modelis

Šajā nodaļā tiek aplūkots divdimensiju kvantētu stāvokļu interpolatora modelis, kas paredzēts reālā laika animācijas nodrošināšanai kvantētu stāvokļu sistēmu imitācijas gadījumos, balstoties uz iepriekš aprakstīto koncepciju. Kvantētu stāvokļu interpolators ir atomārs V-DEVS modelis ar diviem ieejas portiem ip_x un ip_y, un kādā no tiem laika momentā t_x tiek saņemta ieejas vērtība v_x. 2.1.algoritms definē interpolatora reakciju uz ieejas notikumiem, izpildot iekšējās pārejas funkciju δ_{ext}.

2.1. algoritms Interpolatora ārējās pārejas funkcija $\delta_{ext}(s,e,x)$

ja $p_x = ip_x$ **tad**
  pievienot (v_x,t_x) $rinda_x$
citādi ja $p_x = ip_y$ **tad**
  pievienot (v_x,t_x) $rinda_y$
beigas (ja)
ja $\sigma^{discr} \neq \infty$ **tad**
  apstādināt *saknes_koordinators // apstādina saknes koordinatoru*
citādi ja $garums(rinda_x) \geq 2 \wedge garums(rinda_y) \geq 2$ **tad**
  $(v_{x0},t_{x0}) \Leftarrow rinda_x(0)$
  $(v_{x1},t_{x1}) \Leftarrow rinda_x(1)$
  $(v_{y0},t_{y0}) \Leftarrow rinda_y(0)$
  $(v_{y1},t_{y1}) \Leftarrow rinda_y(1)$
  $\sigma_x^{discr} \Leftarrow t_{x1} - t_{x0}$
  $\sigma_y^{discr} \Leftarrow t_{y1} - t_{y0}$
  ja $garums(rinda_x) = 2 \vee garums(rinda_y) = 2$ **tad**
    $\sigma^{discr} \Leftarrow min(\sigma_x^{discr}, \sigma_y^{discr})$
  beigas (ja)
  apstādināt *saknes_koordinators*
beigas (ja)

Interpolatora modelis satur divas notikumu laika prioritātes rindas $rinda_x$ un $rinda_y$ katra ieejas porta saņemto vērtību un laika uzkrāšanai. Ja ārējās pārejas ziņojuma saņemšanas brīdī interpolatora modelis neatrodas pasīvā stāvoklī, t.i., laika intervāls līdz nākošajai iekšējai diskrētajai pārejai $\sigma^{discr} \neq \infty$, tad tiek dota komanda apstādināt kvantētu stāvokļu sistēmas modeļa saknes koordinatoru (2.8. attēls), pretējā gadījumā, ja kādā no ieejas rindām ir uzkrāti vismaz divi elementi (v_x,t_x), tad tiek aprēķināts jauns laika intervāls līdz nākošajai iekšējai

diskrētajai pārejai, pēc tam apstādinot saknes koordinatoru. Rindu elementu bāzes indekss ir 0, un 0.elements apzīmē vecāko rindas elementu. 2.2.algoritms apraksta diskrēto izejas funkciju $\lambda^{discr}(s)$, kas veic vecāko rindas $rinda_x$ un $rinda_y$ elementu dzēšanu, modeļa izejā atgriežot interpolēto 2D vērtību.

2.2. algoritms Interpolatora diskrētā izejas funkcija $\lambda^{discr}(s)$

$(v_{x0}, t_{x0}) \Leftarrow rinda_x(0)$
$(v_{x1}, t_{x1}) \Leftarrow rinda_x(1)$
$(v_{y0}, t_{y0}) \Leftarrow rinda_y(0)$
$(v_{y1}, t_{y1}) \Leftarrow rinda_y(1)$
ja $t_{x1} < t_{y1}$ **tad**
  $v_{y0} \Leftarrow v_{y0} + (v_{y1} - v_{y0})/(t_{y1} - t_{y0}) * (t_{x1} - t_{x0})$
  $t_{y0} \Leftarrow t_{x1}$
  $(x, y) \Leftarrow (v_{x0}, v_{y0})$
  izdzēst 0.elementu no $rinda_x$ // izdzēš vecāko prioritātes rindas elementu
citādi ja $t_{x1} > t_{y1}$ **tad**
  $v_{x0} \Leftarrow v_{x0} + (v_{x1} - v_{x0})/(t_{x1} - t_{x0}) * (t_{y1} - t_{y0})$
  $t_{x0} \Leftarrow t_{y1}$
  $(x, y) \Leftarrow (v_{x0}, v_{y1})$
  izdzēst 0.elementu no $rinda_y$
citādi
  $(x, y) \Leftarrow (v_{x1}, v_{y1})$
  izdzēst 0.elementu no $rinda_x$
  izdzēst 0.elementu no $rinda_y$
beigas (ja)
$izeja = (x, y)$

2.3.algoritms apraksta diskrēto laika funkciju, kas atgriež laika intervālu līdz nākamajai iekšējai diskrētajai pārejai σ^{discr}.

2.3. algoritms Interpolatora diskrētā laika ritējuma funkcija $ta^{discr}(s)$

atgriezt σ^{discr}

2.4.algoritms apraksta diskrēto iekšējās pārejas funkciju $\delta_{int}^{discr}(s)$, kas veic jaunu nākošās iekšējās diskrētās pārejas laika intervāla σ^{discr} aprēķinu, pēc tam turpinot iepriekš apstādināto kvantētu stāvokļu sistēmas saknes koordinatora darbību.

2.4. algoritms Interpolatora diskrētā iekšējās pārejas funkcija $\delta_{int}^{discr}(s)$

ja $garums(rinda_x) \geq 2 \wedge garums(rinda_y) \geq 2$ **tad**
$(v_{x0}, t_{x0}) \Leftarrow rinda_x(0)$
$(v_{x1}, t_{x1}) \Leftarrow rinda_x(1)$
$(v_{y0}, t_{y0}) \Leftarrow rinda_y(0)$
$(v_{y1}, t_{y1}) \Leftarrow rinda_y(1)$
$\sigma_x^{discr} \Leftarrow t_{x1} - t_{x0}$
$\sigma_y^{discr} \Leftarrow t_{y1} - t_{y0}$
$\sigma^{discr} \Leftarrow min(\sigma_x^{discr}, \sigma_y^{discr})$
citādi
$\sigma^{discr} \Leftarrow \infty$
beigas (ja)
turpināt *saknes_koordinators // turpina saknes koordinatora darbību*

2.3. Modeļvadāma V-DEVS modelēšana

Imitācijas modeļa izveide agrīnā sistēmas izstrādes fāzē ir sarežģīts uzdevums, jo tas prasa padziļinātas modelēšanas metožu, sistēmas problēmu apgabala un modeļa izpildes paradigmu zināšanas. Šim nolūkam ir nepieciešama sadarbība starp sistēmas ekspertiem un modelēšanas ekspertiem, kuriem parasti ir diametrāli pretējas zināšanas un pieredze. Tādēļ ir nepieciešams praktisks un efektīvs veids 1.1.3. apakšnodaļā aprakstītā modelēšanas un imitācijas ietvara pielietošanai sistēmas modelēšanā jau agrīnā izstrādes etapā. Eksistē dažādi paņēmieni un līdzekļi šī jautājuma risināšanā, kur vienas no visplašāk izmantotajām koncepcijām un līdzekļiem ir vienotā modelēšanas valoda UML [79, 115] un modeļvadāmā arhitektūra (angļu valodā *Model-Driven Architecture* (MDA)) [28]. Modeļvadāmās arhitektūras pamatā ir ideja par sistēmas izstrādi, izmantojot augsta līmeņa grafisko notāciju, ļaujot izstrādājam koncentrēties uz problēmas būtību, mazāk uzmanības pievēršot sīkākām detaļām.

Imitācijas modelēšanas jomā reālas sistēmas tiek abstrahētas modeļu veidā, lai būtu iespējams veikt virtuālus eksperimentus. Modeļu izveidē tiek izmantotas dažādas metodoloģijas un notācijas, piemēram, diferenciālvienādojumi, Petri tīkli vai stāvokļu automāti. Tomēr saistībā ar to, ka sarežģītas sistēmas raksturo gan liels elementu skaits, gan šo elementu jeb komponentu dažādība, sarežģītu sistēmu modeļi praksē tiek izstrādāti ar dažādas uzbūves un darbības (piemēram, diskrētu notikumu vai nepārtrauktu procesi) komponentiem. Tādējādi vairums gadījumu ir nepieciešamas dažādas notācijas katra sistēmas komponenta aprakstīšanai un realizācijai, kas ir saistāms ar multiformālismu modelēšanas jēdzienu [99].

2.3.1. Metamodelēšana

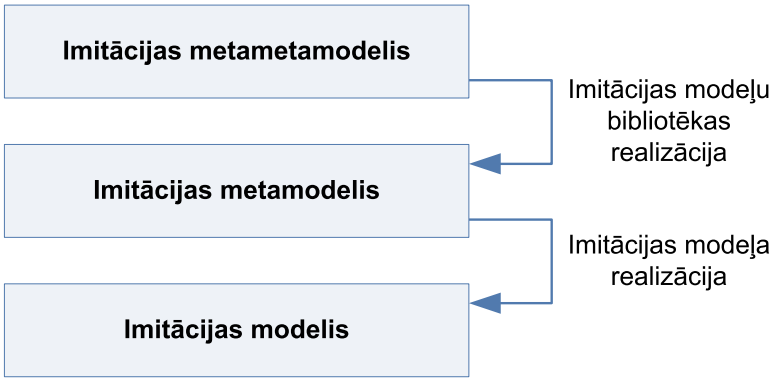
Tā kā V-DEVS formālisms nodrošina modeļu izveides iespējas savstarpēji saistītu hierarhisku komponentu veidā, tad dabiska izvēle šādu modeļu izveidei ir grafiskās notācijas izmantošana. Šāda pieeja ir pielietota vairākos uz DEVS formālismu bāzētos izpētes projektos, piemēram, *PythonDEVS* simulatoram paredzētais *AToM*^{3} metamodelēšanas vidē izstrādātais modelēšanas rīks.

Metadatus tradicionāli saprot kā "datus par datiem", aprakstot dažādas modelēšanas un datu attēlošanas aktivitātes. Piemēram, bibliotēka satur informāciju (metadatus) par publikācijām (dati). Vienas lietojumprogrammas metadati var būt citas lietojumprogrammas dati, un arī pašus metadatus ir iespējams aprakstīt ar metadatiem.

Aplūkojot metamodelēšanas iespējas V-DEVS imitācijas modelēšanā, ir nepieciešams definēt vairākus jēdzienus:

- *Metamodelis* - modelis, kas formāli definē noteikta problēmapgabala specifiskas modelēšanas vides sintaksi un semantiku [74].
- *Metamodelēšanas vide* - instrumentāls ietvars metamodeļu izveidei, validācijai un translēšanai.
- *Metametamodelis* - modelis, kas formāli definē dotās metamodelēšanas vides sintaksi un semantiku.

Modelēšana un metamodelēšana pēc būtības ir identiskas aktivitātes, taču atšķirīga ir to interpretācija. Ja modelēšanas pamatā ir reālās pasaules sistēmu abstrakcijas, tad savukārt process, kad modelēšanas pamatā ir citi modeļi, tiek saukts par metamodelēšanu. Tādējādi modelēšanas koncepcijas ir attiecināmas arī uz metamodelēšanu un metametamodelēšanu. Taču, tā kā modelēšanas, metamodelēšanas un metametamodelēšanas mērķi ir pilnīgi atšķirīgi, tad metamodelēšanas pētījumos plaši tiek izmantots četru slāņu ietvars, kuru ir iespējams adaptēt arī imitācijas modelēšanas vajadzībām hierarhisku trīs slāņu veidā (2.9. attēls).



2.9. att. Imitācijas metamodelēšanas etapi

2.3.2. Metamodelēšanas vides prasības

Metamodeļu transformācija par modeļiem tiek veikta ar modelēšanas valodu palīdzību, kuras raksturo noteikta sintakse un semantika. Metamodelēšanas videi ir izvirzāmas šādas prasības:

1. *Sintaktiskās specifikācijas:*

   (a) *abstrakta sintakse* - valodas sintakse bez konkrētas realizācijas detaļām, nodrošinot būtiskāko formālisma sastāvdaļu attēlojumu;

   (b) *konkrēta sintakse*, kas ietver konkrētu attēlojumu, piemēram, tekstuāla valoda, kas balstīta uz Bakusa-Naura formas (BNF) konstrukcijām.

2. *Semantiskās specifikācijas*, kas sevī iekļauj modeļa izveides ierobežojumus, lai būtu iespējams definēt domēna specifiskas koncepcijas un ierobežojumus. Semantiskās specifikācijas var iedalīt sīkāk divos veidos:

   (a) *statiska semantika* - semantika, kas atbilst modelēšanas valodas konstrukcijām, definē modeļa invariantos nosacījumus [66] un kuru iespējams pārbaudīt modeļa izveides gaitā (piemēram, elektronisko komponenšu skaits, kuras iespējams pieslēgt loģiskās ķēdes izejā, lai nepārsniegtu maksimāli atļautās izejas slodzes parametrus);

   (b) *dinamiska semantika* - semantika, kas atbilst modelēšanas konstrukciju interpretācijai un kuru iespējams pārbaudīt tikai modeļa izpildes gaitā (piemēram, noteikšana, vai ir sasniegts noteikts modeļa stāvoklis).

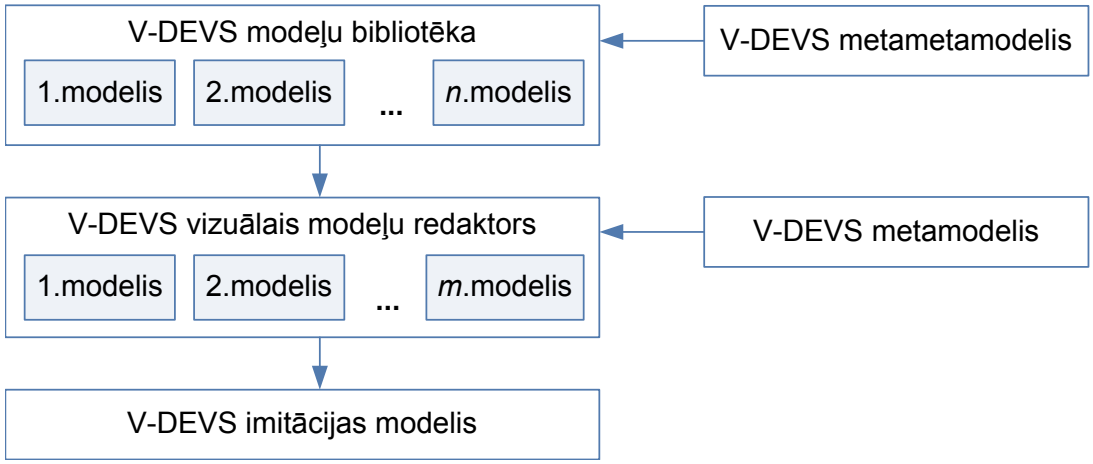
3. *Attēlojuma specifikācijas* - svarīga prasība pilnīgas modelēšanas vides izveidei, nodrošinot modeļa elementu, saišu un atribūtu vizuālā attēlojuma parametrus.

4. *Interpretatora specifikācijas* - ir nepieciešamas konkrētas modeļa informācijas ieguvei, piemēram, modeļa dokumentēšanas un izpildes vajadzībām. Tādējādi interpretatora specifikācija ir uzskatāma par konkrētu dinamiskās semantikas realizāciju.

Modeļvadāmajai pieejai V-DEVS formālisma vajadzībām ir nosakāmas šādas prasības:

- metamodelim jānodrošina sintaktiska struktūra modelējamās sistēmas aprakstam;
- modelēšanas elementu grafiskā attēlojuma eksistences nepieciešamība;
- piedāvātajam metamodelim jānodrošina pietiekoši elastīgs mehānisms konsistentam modelējamās sistēmas aprakstam.

2.10.attēlā ir attēlota V-DEVS imitācijas metamodelēšanas vispārējā arhitektūra. V-DEVS metamodelis definē izstrādājamā V-DEVS imitācijas modeļa sintaktisko un semantisko struktūru, un ir pamats loģisko un vizuālo imitācijas modeļu aprakstam. V-DEVS metametamodelis definē V-DEVS metamodeļa sintaksi un semantiku, un ir paredzēts modeļa bibliotēkas elementu aprakstam.



2.10. att. V-DEVS imitācijas metamodelēšanas vispārējā arhitektūra

2.4. V-DEVS vizualizācijas konveijers

Uz 1.2. apakšnodaļā aprakstītā V-DEVS formālisma pamata iespējams izstrādāt ne tikai koncepciju diskrētas un nepārtrauktas imitācijas modelēšanas un vizualizācijas integrācijai, bet šo formālismu iespējams pielietot arī pašu vizualizācijas pamatā esošo datorgrafikas attēlošanas sistēmu realizācijai. 1.3.3. apakšnodaļā ir apskatītas divas grafiskās attēlošanas jeb renderēšanas pamatkoncepcijas - scēnas grafs un vizualizācijas konveijers. Tā kā vizualizācijas konveijera

koncepcija balstās uz savstarpēji saistītu hierarhisku strukturētu komponentu sistēmas izmantošanu, tad šeit rodas iespēja pielietot V-DEVS matemātisko aparātu, tādējādi unificējot ne tikai imitācijas modelēšanas, bet arī datorgrafikas aspektus vienotā kontekstā.

2.4.1. Notikumu vadāms vizualizācijas konveijers

Netiešās izpildes notikumu vadāms V-DEVS vizualizācijas konveijers pēc savas strukturālās uzbūves ir pats vienkāršākais, salīdzinot ar citiem aplūkotajiem vizualizācijas konveijera paveidiem. Tā kā katra konveijera elementa izejas notikumi vada nākošā saistītā elementa darbību, tad nav nepieciešamības pēc papildus ieejām un parametriem katra vizualizācijas konveijera elementa darbības vadībai un kontrolei. Taču netiešās izpildes notikumu vadības gadījumā būtisks trūkums ir tas, ka izmaiņas jebkurā elementā izsauc visu saistīto secīgo elementu izpildi.

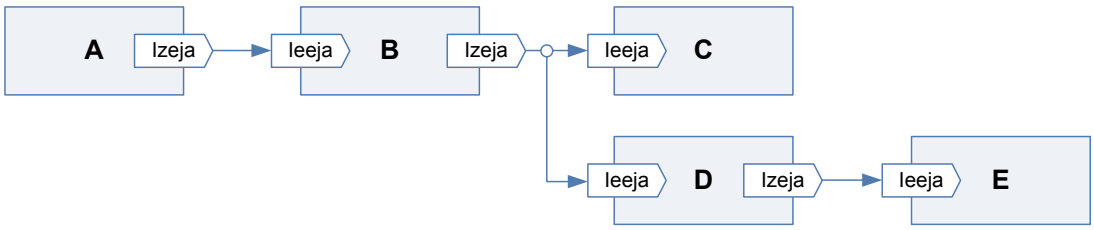
Viens no diviem netiešās izpildes vizualizācijas konveijera variantiem ir netiešās izpildes notikumu vadāms vizualizācijas konveijers, kura realizācijas pamatprincipi ar V-DEVS formālismu ir vienkārši - katra konveijera elementa ieejā no iepriekšējā elementa pienākošais notikums izsauc dotā elementa ārējo un iekšējo stāvokļu pāreju, ģenerējot jaunu notikumu dotā elementa izejā (2.11. attēls). Konveijera elementa izejā ģenerētais notikums, nonākot nākošā piesaistītā elementa ieejā, tiek apstrādāts analoģiski, veidojot secīgu vizualizācijas apstrādes operāciju ķēdi. Notikumu vadāma vizualizācijas konveijera nepieciešamo ieejas/izejas portu skaits N_P informācijas apmaiņai starp konveijera elementiem ir nosakāms pēc šādas formulas:

$$N_P = S_O + 2F + S_I, \tag{2.7}$$

kur S_O - avota elementu skaits;

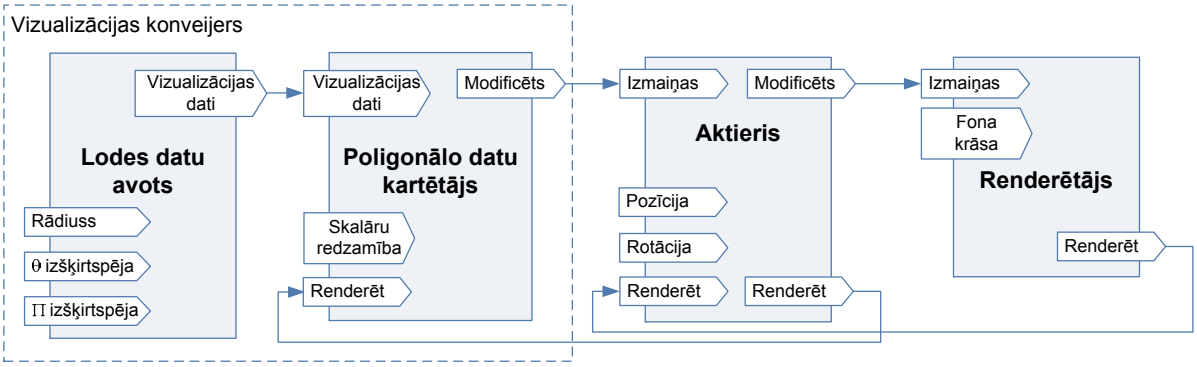
F - filtra elementu skaits;

S_I - izejas elementu skaits.



2.11. att. Netiešās izpildes notikumu vadāma V-DEVS konveijera struktūra

V-DEVS netiešās izpildes notikumu vadāma vizualizācijas konveijera darbības ilustrēšanai tiek izmantots vienkāršs formāls vizualizācijas konveijera piemērs (2.12. attēls), kura izpildes uzdevums ir sfēras jeb lodes grafiskā primitīva ģenerēšana.



2.12. att. Notikumu vadāma V-DEVS vizualizācijas konveijera piemērs

Lodes datu avota modelis ģenerē lodes poligonālos vizualizācijas datus VD, balstoties uz uzdotajām lodes rādiusa, kā arī garuma θ un platuma π virziena izšķirtspējas (poligonālās diskretizācijas) vērtībām. Lodes datu avota V-DEVS modelis formāli ir aprakstāms šādi:

$$AM_{\text{lodes datu avots}} = \langle X, Y, S, \delta_{ext}, \delta_{int}^{discr}, \lambda^{discr}, ta^{discr} \rangle$$

Ieejas notikumu mainīgie: $X = (\text{Rādiuss}, \Theta\text{izšķirtspēja}, \Pi\text{izšķirtspēja})$,

kur $\text{Rādiuss} = (\text{"rādiuss"}, v) \,|\, v \in \mathbb{R}_{0,+\infty}$ - ieejas ports lodes rādiusa definēšanai;

$\Theta\text{izšķirtspēja} = (\text{"}\Theta\text{izšķirtspēja"}, v) \,|\, v = [0,\ 360]$ - ieejas ports punktu skaitam ģeogrāfiskā garuma virzienā;

$\Pi\text{izšķirtspēja} = (\text{"}\Pi\text{izšķirtspēja"}, v) \,|\, v = [0,\ 180]$- ieejas ports punktu skaitam ģeogrāfiskā platuma virzienā.

Stāvokļu mainīgie: $S = \{(\text{fāze}, \text{rādiuss}, \theta\,\text{izšķirtspēja}, \pi\,\text{izšķirtspēja})\}$,

kur $\text{fāze} = \{\text{"pasīvs"},\ \text{"modificēts"}\}$;

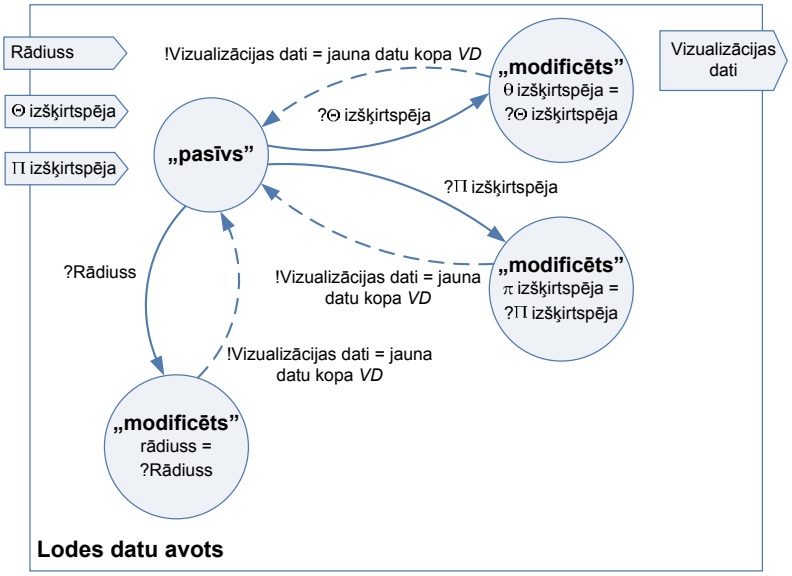
rādiuss - lodes rādiuss;

θ izšķirtspēja - punktu skaits ģeogrāfiskā garuma virzienā;

π izšķirtspēja - punktu skaits ģeogrāfiskā platuma virzienā.

Izejas notikumu mainīgie: $Y = (\text{Izeja})$,

kur $\text{Vizualizācijas dati} = (\text{"vizualizācijas dati"}, VD)$ - izejas ports, kurā tiek ģenerēta lodes vizualizācijas datu kopa VD (1.9 formula). 2.13. attēlā ir parādīta notikumu vadāma lodes datu avota stāvokļu pāreju diagramma, izmantojot DEVS modelēšanā zināmos modeļa elementu apzīmējumus. Piemēram, "**?**" simbols tiek lietots ārējo ieeju apzīmēšanai, bet "**!**" simbols - izejas notikumu apzīmēšanai.



2.13. att. Notikumu vadāma lodes datu avota stāvokļu pāreju diagramma

Poligonālo datu kartētājs ir vizualizācijas konveijera elements, kas vizualizācijas datus konvertē par grafiskajiem primitīviem, respektīvi, izpilda grafisko renderēšanas procesu. Poligonālo datu kartētāja V-DEVS modelis formāli ir aprakstāms šādā formā:

$$AM_{\text{vizualizācijas datu kartētājs}} = \left\langle X, Y, S, \delta_{ext}, \delta_{int}^{discr}, \lambda^{discr}, ta^{discr} \right\rangle.$$

Ieejas notikumu mainīgie: $X = (\text{Vizualizācijas dati}, \text{Skalāru redzamība}, \text{Renderēt})$,

kur $\text{Rādiuss} = (\text{”rādiuss”}, v) \,|v \in \mathbb{R}_{0,+\infty}$ - ieejas ports lodes rādiusa definēšanai;

$\text{Skalāru redzamība} = (\text{”skalāru redzamība”}, \{true, false\})$ - ieejas ports skalāro atribūtu vērtību iekļaušanu renderēšanas procesā;

$\text{Renderēt} = (\text{”renderēt”}, \{true\})$ - ieejas ports renderēšanas izpildei.

Stāvokļu mainīgie: $S = \{(\text{fāze}, \text{vizualizācijas dati}, \text{skalāru redzamība})\}$,

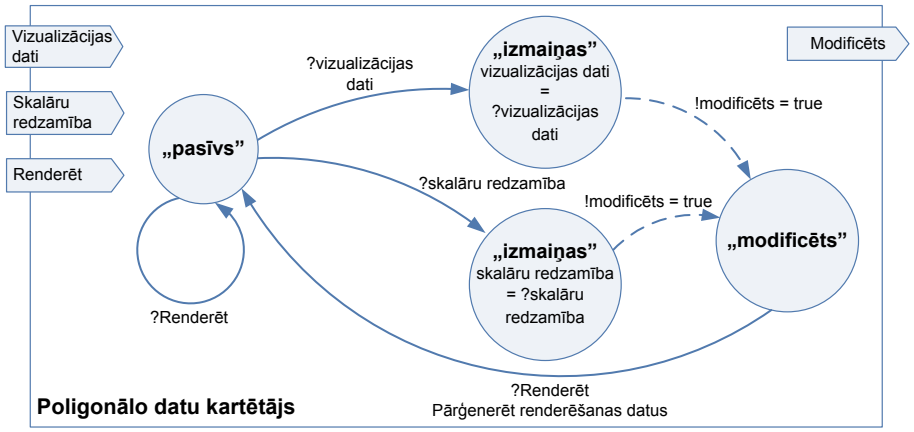
kur fāze = {”pasīvs”, ”modificēts”};

vizualizācijas dati - no ieejas porta apstrādei saņemtie vizualizācijas dati;

skalāru redzamība $\in \{true, false\}$ - no ieejas porta saņemtā skalāru redzamības parametra vērtība.

Izejas notikumu mainīgie: $Y = (\text{Modificēts})$,

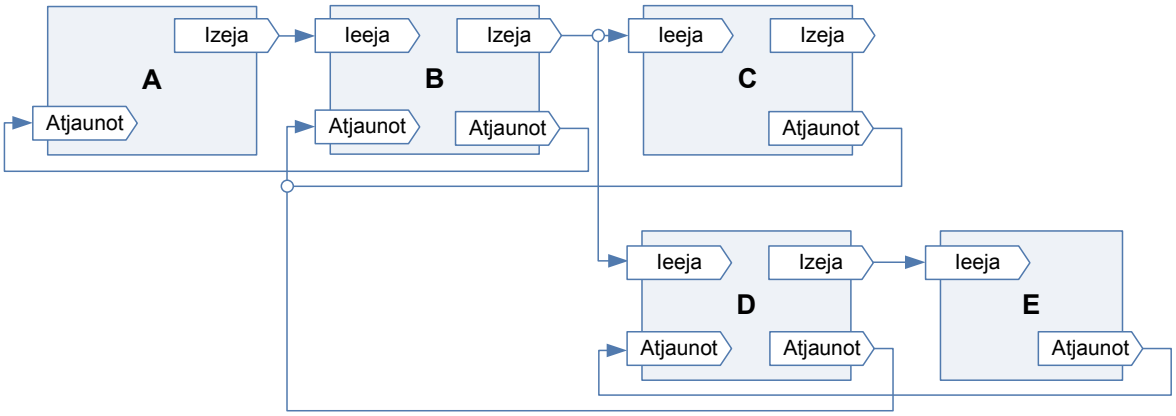
kur $\text{Modificēts} = (\text{”modificēts”}, \{true\})$ - izejas ports, kurā tiek ģenerēti notikumi, kas indicē ieejas parametru izmaiņas.



2.14. att. Notikumu vadāma poligonālo datu kartētāja stāvokļu pāreju diagramma

2.4.2. Pieprasījuma vadāms vizualizācijas konveijers

Ar V-DEVS formālisma palīdzību salīdzinoši vienkārši ir realizējams pieprasījuma vadāms vizualizācijas konveijers, kura galvenā priekšrocība, salīdzinot ar notikumu vadāmo pieeju, ir iespējas izpildīt vizualizācijas tīklu tikai pēc atbilstoša izejas pieprasījuma un tikai to vizualizācijas tīkla daļu, kas tieši ietekmē rezultātu. 2.15. attēlā ir parādīta vispārēja V-DEVS pieprasījuma vadāma vizualizācijas konveijera struktūra.

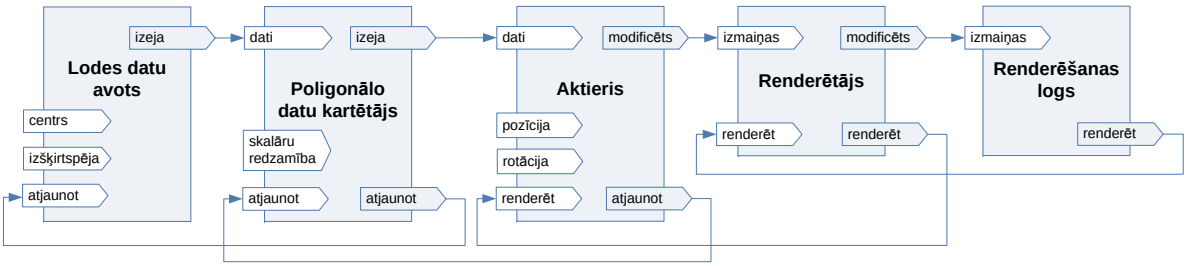


2.15. att. Pieprasījuma vadāma V-DEVS konveijera struktūra

Galvenā strukturālā atšķirība pieprasījuma vadāmam V-DEVS vizualizācijas konveijeram ir tā, ka vizualizācijas konveijera elementiem ir nepieciešami speciāli papildus ieejas un izejas porti pieprasījuma notikumu apmaiņai starp saistītajiem elementiem. Tas ir arī pieprasījuma vadāmā konveijera trūkums, jo tā struktūra ir sarežģītāka, nekā notikumu vadāmam vizualizācijas konveijeram. Pieprasījuma vadāma vizualizācijas konveijera nepieciešamo ieejas/izejas portu skaits N_P informācijas apmaiņai starp konveijera elementiem ir nosakāms pēc sekojošas formulas:

$$N_P = 2S_O + 4F + 2S_I. \tag{2.8}$$

2.16. attēlā ir parādīts vienkāršs pieprasījuma vadāma V-DEVS vizualizācijas konveijera piemērs.



2.16. att. Pieprasījuma vadāma V-DEVS vizualizācijas konveijera piemērs

2.5. Kopsavilkums un secinājumi

Šajā promocijas darba nodaļā ir aprakstīts promocijas darba ietvaros izstrādātais V-DEVS formālisms, definējot tā pamatkoncepcijas integrētas imitācijas modelēšanas un vizualizācijas kontekstā.

Nodaļā paveiktais:

- izstrādāta integrēta sistēmteorētiska pieeja diskrētu notikumu un nepārtrauktu sistēmu interaktīvai vizuālajai imitācijas modelēšanai;
- definēti un aprakstīti izstrādātās pieejas komponenti un saikne starp tiem;
- ir veikta imitācijas modelēšanas un vizualizācijas integrācijas izpēte kvantētu stāvokļu sistēmu modelēšanā.

Sasniegtie rezultāti:

- ieviests vizualizācijas ietvara jēdziens integrētas vizuālās imitācijas modelēšanas konteksta definēšanai mijiedarbībā ar imitācijas un modelēšanas ietvaru;
- izstrādāts uz sistēmteoriju balstīts V-DEVS formālisms;
- izveidota metodika imitācijas modelēšanas un vizualizācijas integrācijai kvantētu stāvokļu sistēmu imitācijā, pielietojot kvantētu stāvokļu interpolāciju;
- izstrādāta modeļvadāma modelēšanas pieeja V-DEVS formālisma realizācijai;
- izstrādāts V-DEVS vizualizācijas konveijers imitācijas modelēšanas un vizualizācijas procesu integrēšanai.

Galvenie secinājumi ir šādi:

- izstrādātais V-DEVS formālisms ir kombinēta diskrētu notikumu un nepārtraukto sistēmu imitācijas modelēšanas koncepcija, jo satur gan diskrētu notikumu, gan nepārtrauktu procesu imitācijas modelēšanas komponentus:
  - diskrētais komponents ir ekvivalents klasiskā DEVS formālisma definētajam atomārajam modelim diskrētu notikumu modelēšanai;
  - nepārtrauktais komponents definē nepārtrauktu procesu modelēšanu, un ir universāli pielietojams gan imitācijas modelēšanas, gan vizualizācijas uzdevumiem.
- V-DEVS formālisms nodrošina formālu bāzi interaktīvai vizuālai kombinētu sistēmu modelēšanai;
- V-DEVS vizualizācijas konveijera priekšrocības:
  - sistēmpieeja vizualizācijas procesa nodrošināšanai;
  - ar V-DEVS vienots ieeju un izeju uzbūves, kā arī sinhronizācijas princips;
  - notikumu vai pieprasījuma vadīta vizualizācijas plūsmas apstrāde;
  - datorgrafikā plaši izmantotā scēnas grafa arhitektūras un tradicionālā vizualizācijas konveijera apvienojuma iespējas.

V-DEVS vizualizācijas konveijera elementi satur tikai diskrēto notikumu apstrādes daļu, tādēļ tos iespējams aprakstīt gan ar klasiskā DEVS, gan ar šajā darbā piedāvātā V-DEVS formālisma palīdzību.

3. V-DEVS FORMĀLISMA PRAKTISKĀ REALIZĀCIJA

Šī nodaļa ir veltīta iepriekšējā nodaļā sniegtā sistēmteorētiskā V-DEVS formālisma praktiskajai realizācijai un ar to saistīto jautājumu risināšanai, iepriekš iegūtos teorētiskos rezultātus praktiski realizējot interaktīvas vizuālās 3D V-DEVS imitācijas modelēšanas sistēmas prototipa veidā.

3.1. Formālisma realizācija programmatūras sistēmas veidā

V-DEVS imitācijas modelēsanas programmatūras prototipa realizācijas sākuma stadijā ir formulētas un specifikācijas procesā ņemtas vērā vairākas prasības:

- modernas programmatūras izstrādes koncepcijas;
- programmas koda atkārtotas lietojamības iespējas;
- ātras izstrādes, uzturēšanas un paplašināšanas nodrošinājums;
- atklātā pirmkoda izstrādes rīku, programmatūras bibliotēku un moduļu pieejamība un izmantošanas iespējas.

Šīm minētajām prasībām kā vislabāk atbilstošā tika izvēlēta programmēšanas valoda Java, tādēļ V-DEVS formālisma programmatūras prototips un visas nepieciešamās apakšsistēmas ir izstrādātas ar Java programmēšanas valodu. Darba izstrādes gaitā ir bijuši V-DEVS formālisma realizācijas mēģinājumi ar C++ [50], Object Pascal [54] un C# [55] programmēšanas valodām un vidēm, taču gala rezultātā par piemērotāko ir atzīta Java programmēšanas valoda.

Programmatūras risinājuma izstrādē ir pielietotas šādas informācijas tehnoloģijas koncepcijas:

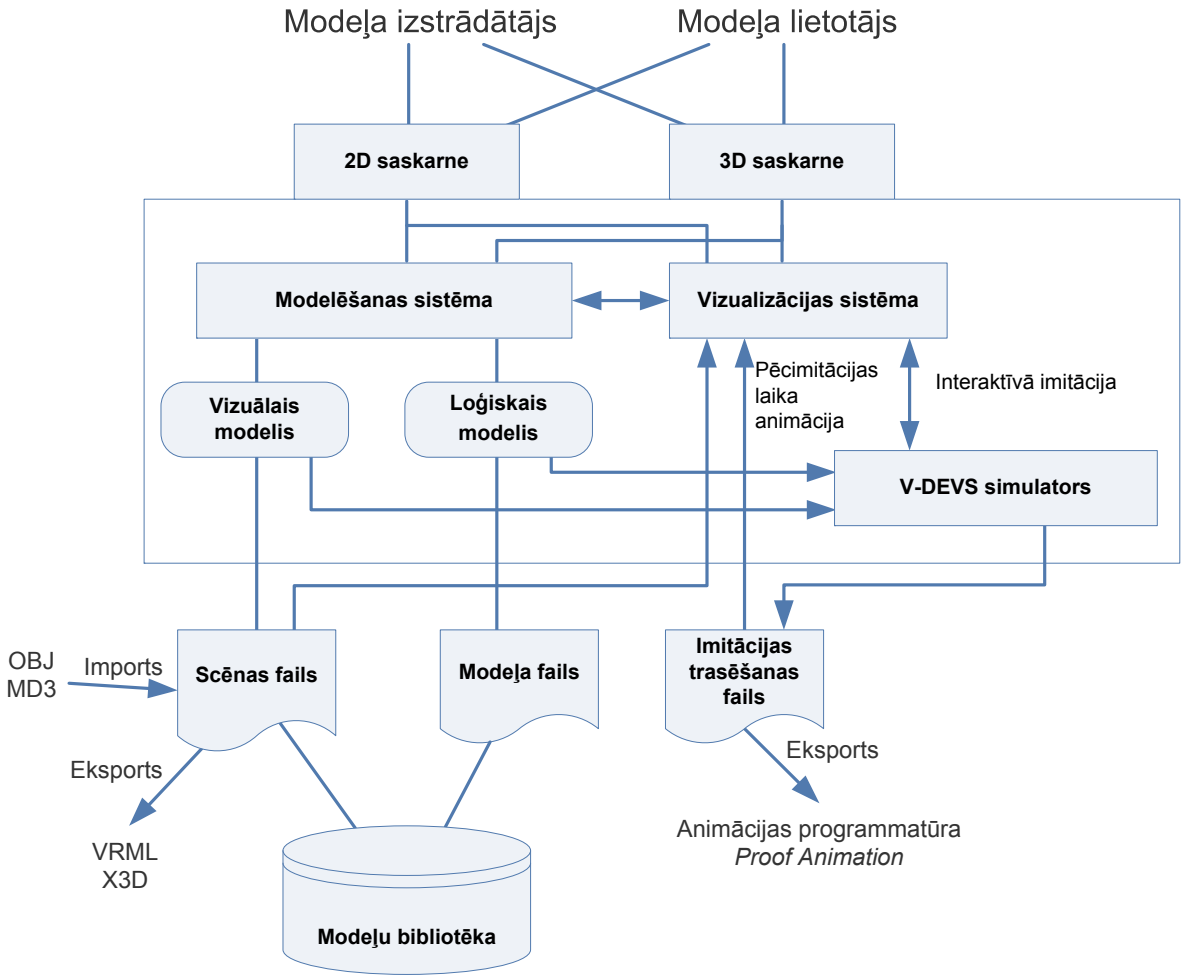
- modularitāte;
- strukturēts kods;
- objektorientētā programmēšana.

V-DEVS programmatūras prototipa izstrādei ir izmantoti trīs atklāta pirmkoda programmatūras projekti:

- scēnas grafa bibliotēka JME [42], kas pielietota 3D grafiskajai renderēšanai;
- vizualizācijas bibliotēka VTK [84], kas pielietota 3D vizualizācijai;

- Eclipse izstrādes platforma un EMF, GEF, GMF programmatūras bibliotēkas [94].

3.1. attēlā ir parādīta darba autora izstrādātā V-DEVS formālisma praktiskās realizācijas prototipa arhitektūra [55, 56], kas sastāv no trim apakšsistēmām - modelēšanas sistēmas, vizualizācijas sistēmas un V-DEVS simulatora. Sistēmas arhitektūras izstrādē ir pielietota pēdējā laikā imitācijas modelēšanas sistēmās arvien plašāk pielietotā integrētā pieeja [21], apvienojot dažādu avotu un veidu ieejas, piemēram, statistiskās un grafiskās informācijas apstrādi vienotā platformā. Izstrādātās sistēmas arhitektūra ir atvērta, tādējādi dotajam programmatūras prototipam ir iespējams pievienot papildus moduļus, piemēram, jauktās realitātes lietotāja saskarnes nodrošināšanai [55].



3.1. att. V-DEVS sistēmas realizācijas arhitektūra

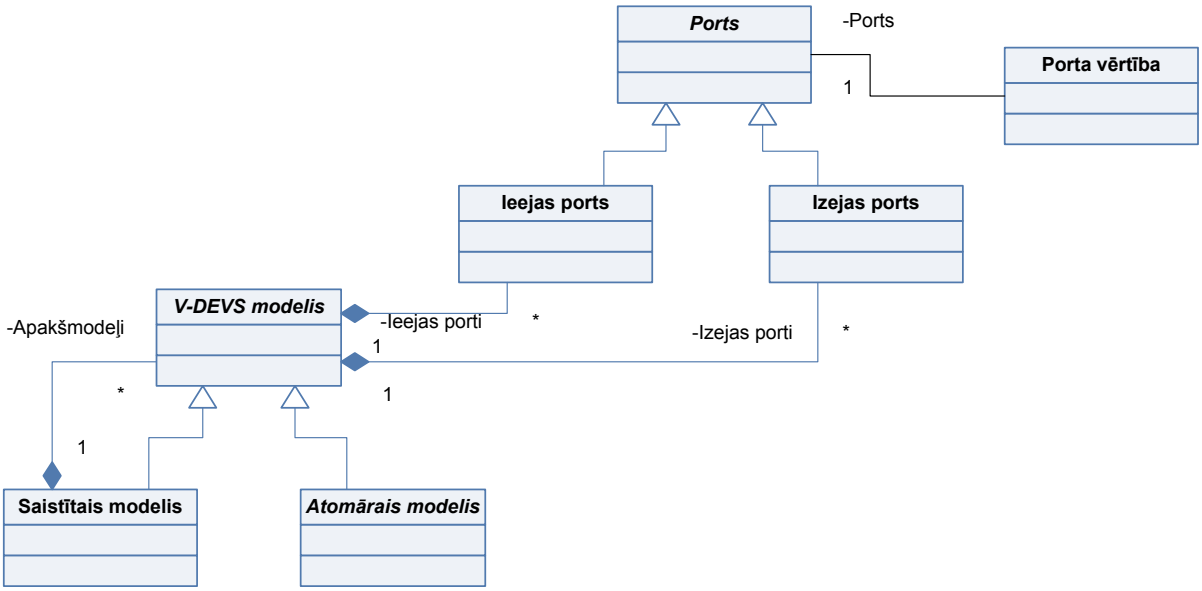
Modelēšanas sistēma mijiedarbībā ar vizualizācijas sistēmu nodrošina imitācijas modeļa izstrādātāju ar nepieciešamajiem instrumentiem imitācijas modeļu izveidei virtuālā 2D/3D vidē. Fizisko vizuālo modeli ir iespējams izveidot ar sistēmā iebūvētajiem līdzekļiem, vai arī to ir iespējams importēt no ārējas datorgrafikas modelēšanas vai CAD programmatūras, izmantojot plaši pazīstamus grafiskos formātus, piemēram, MD3, OBJ. Izveidotais fiziskais modelis tiek

saglabāts scēnas failā, vai arī eksportēts X3D vai VRML formātā izmantošanai citos datorgrafikas programmatūras līdzekļos.

3.1.1. Simulatora arhitektūra

V-DEVS simulatora apakšsistēma ir galvenais interaktīvās vizuālās imitācijas modelēšanas vides komponents, kas veic loģiskā imitācijas modeļa definēto darbības instrukciju izpildi. Imitācijas izpildes rezultāti var tikt saglabāti Proof Animation trasēšanas faila formātā, nodrošinot pēcimitācijas animācijas iespējas.

Imitācijas sistēmas bāzes struktūru veido modeļu klases V-DEVS modelis, Atomārais modelis un Saistītais modelis, kā arī portu klases Ports, Ieejas ports un Izejas ports (3.2. attēls).



3.2. att. V-DEVS formālisma realizācijas bāzes klašu diagramma

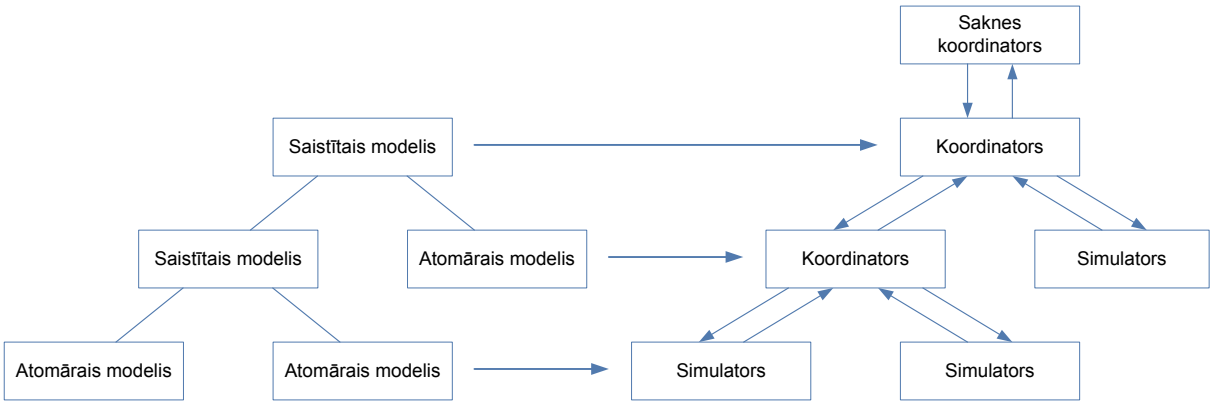
V-DEVS modelis ir abstrakta modeļu bāzes klase, kas satur ieejas un izejas portu sarakstu. Noteiktu ieejas un izejas portu pievienošana šim sarakstam ir atvasināto klašu realizācijas uzdevums. Ports ir abstrakta bāzes klase, kas realizē 2.1.2. apakšnodaļā aprakstīto V-DEVS modeļa saskarni. Abstrakta klase Atomārais modelis definē V-DEVS formālisma atomārā modeļa struktūras 2.1 realizāciju, bet klase Saistītais modelis realizē V-DEVS saistītā modeļa struktūru 2.6. Jauns V-DEVS atomārais modelis tiek veidots, atvasinot jaunu klasi no klases Atomārais modelis.

V-DEVS saknes koordinatora vispārējais darbības algoritms (P2.1. attēls 2. pielikumā) realizē imitācijas cikla apstrādi, kas sastāv no diskrētu notikumu un nepārtrauktu sistēmas notikumu apstrādes procedūrām, par imitācijas procesa virzītāju ņemot imitācijas pulksteņa laiku. Imitācijas pulkstenis var darboties virtuālā vai arī reālā laikā, tādējādi nodrošinot gan virtuālo,

gan mērogotu reālā laika imitācijas izpildi.

Hierarhiskais simulatora algoritms

Hierarhiskā un modulārā V-DEVS modeļa struktūra ir balstīta uz DEVS simulatoru klasisko specifikāciju [111]. Katram atomārajam modelim ir piesaistīts simulatora objekts, savukārt katram saistītajam modelim ir piesaistīts koordinatora objekts (3.3. attēls), tādējādi saistīto modeļu skaits ir vienāds ar koordinatoru skaitu, bet atomāro modeļu skaits ir vienāds ar simulatoru skaitu. Simulators veic atomārā modeļa izpildi, bet koordinators - saistītā modeļa izpildi.

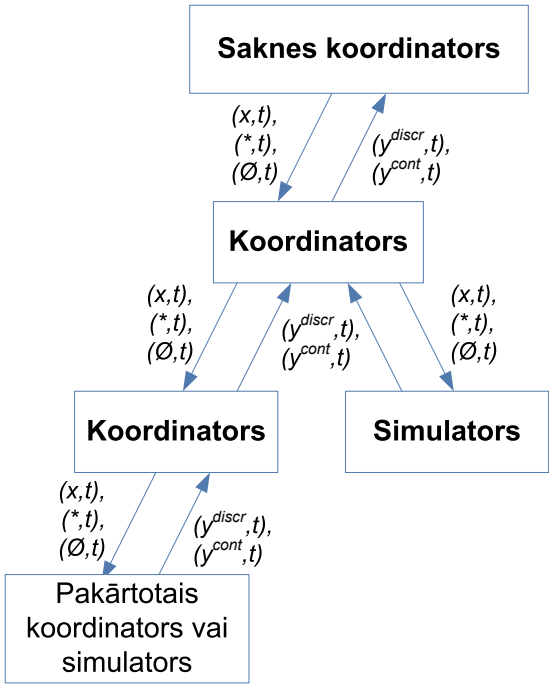


3.3. att. Hierarhiskā modeļa un simulatora saikne

Imitācijas sistēmas hierarhijas augšgalā atrodas saknes koordinators, kas vada imitācijas ciklu un darbojas pēc 2. pielikuma P2.1. attēlā redzamā vispārējā darbības algoritma. Diskrētu notikumu un nepārtraukto stāvokļu imitācijai tiek izmantoti pieci ziņojumu tipi:

- (x,t) - ārējās pārejas ziņojums, kas ziņojuma saņēmējam, ja tas ir simulatora objekts, liek izpildīt V-DEVS ārējās pārejas funkciju δ_{ext};
- $\left(y^{discr},t\right)$ - diskrētās izejas ziņojums ;
- (y^{cont},t) - nepārtrauktās izejas ziņojums;
- $(*,t)$ - diskrētās iekšējās pārejas ziņojums;
- $(\varnothing,t)$ - nepārtrauktās iekšējās pārejas ziņojums.

3.4. attēlā ir parādīta vispārēja ziņojumu apmaiņas shēma V-DEVS modeļa hierarhijā. Katrs koordinatora objekts nodod ziņojumus (x,t), $(*,t)$ un $(\varnothing,t)$ saviem pakārtotajiem koordinatora vai simulatora objektiem, no tiem atpakaļ saņemot ziņojumus $\left(y^{discr},t\right)$ un (y^{cont},t).



3.4. att. Ziņojumu apmaiņa dažādos hierarhijas V-DEVS modeļa līmeņos

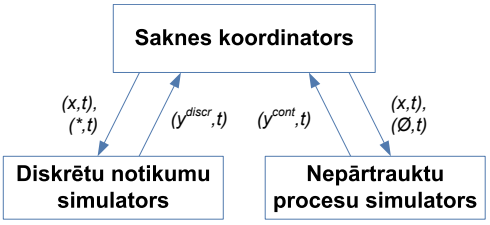
Prioritātes rindu simulatora algoritms

Hierarhiskā klasiskās DEVS specifikācijas realizācija var būt neefektīva gan datora atmiņas, gan veiktspējas ziņā lielu un sarežģītu modeļu imitācijai. Pēdējos gados DEVS simulatoru realizācijas efektivitātes jautājumiem tiek pievērsta arvien lielāka uzmanība [64]. Atmiņas lietojuma neefektivitāte rodas tādēļ, ka hierarhiskās realizācijas gadījumā imitācijas sistēmā tiek izmantots liels skaits koordinatora un simulatora objektu. Būtībā simulatori ir nepieciešami tikai aktīvajiem modeļiem, taču hierarhiskais algoritms izsauc simulatora objektu katram atomārajam modelim. Šo problēmu iespējams atrisināt, daudzu koordinatoru un simulatoru vietā pielietojot vienu kopēju simulatora objektu. Imitācijas laika un ar to saistītās veiktspējas neefektivitāte ir izskaidrojama ar to, ka hierarhiskajā algoritmā tiek izmantota tieši no DEVS specifikācijas atvasināta metožu kopa nākošā notikuma laika noteikšanai un notikumu maršrutēšanai modelī.

Šajā darba daļā tiek piedāvāts jauns V-DEVS simulatora realizācijas algoritms, kam par pamatu izmantoti darbā [64] aprakstītie un *adevs* imitācijas sistēmā [67] realizētie principi, imitācijas arhitektūru veidojot tādā veidā, lai tiktu uzlabota imitācijas efektivitāte šādos aspektos:

- atteikšanās no nevajadzīgu simulatora un koordinatora objektu izmantošanas;
- notikumu plānošanas paātrināšana, apstrādājot tikai aktīvos modeļus;
- atteikšanās no nevajadzīgu iekšējās sinhronizācijas ziņojumu izmantošanas;
- atteikšanās no nevajadzīgu notikumu maršrutēšanas ziņojumu izmantošanas.

Prioritātes rindu simulatora koordinācija tiek veikta pēc tāda paša principa, kā hierarhiskā simulatora variantā to realizē saknes koordinators (P2.1. attēls). Taču galvenā atšķirība šeit ir tā, ka saknes koordinators prioritātes rindu simulatora variantā koordinē divus pakārtotos simulatorus (3.5. attēls), nevis vienu pakārtoto koordinatoru, kā tas ir hierarhiskā simulatora realizācijā.



3.5. att. Prioritātes rindu V-DEVS simulatora vispārējā arhitektūra

Diskrētu notikumu apstrādi realizē diskrētu notikumu simulators, bet nepārtrauktu stāvokļu apstrādi - nepārtrauktu procesu simulators. Abi simulatori pēc uzbūves ir identiski, nākošo notikumu plānošanai izmantojot prioritātes rindu. Prioritātes rinda ir elements, kas visbūtiskāk ietekmē simulatora veiktspēju, tādēļ izvēlētajai datu struktūrai prioritātes rindas realizācijā ir prioritāra nozīme. Binārās kaudzes datu struktūras izvēle prioritātes rindas realizācijai ir optimāla, jo nodrošina elementu pievienošanu un dzēšanu $O(logn)$ laikā.

3.1 algoritms definē prioritātes rindu simulatora apakšprogrammu *nākošās_izejas_aprēķins*, kas tūlītējās atomāro modeļu kopas *tūlītējā_kopa* elementiem izsauc izejas funkciju λ. Tūlītējā kopa satur tos atomāros modeļus, kurus nepieciešams tekošajā imitācijas solī. Šī kopa tiek aizpildīta, balstoties uz prioritātes rindas datiem. Pēc izejas funkcijas λ izsaukšanas modeļa izejas notikumi tiek maršrutēti uz saistīto modeļu ieejām.

3.1. algoritms Prioritātes rindu algoritma apakšprogramma *nākošās_izejas_aprēķins*

```
izdzēst kopas tūlītējā_kopa elementus
atjaunot kopu tūlītējā_kopa no prioritātes rindas
visiem atomārs_modelis ∈ tūlītējā_kopa izpildīt
   modelim atomārs_modelis izsaukt funkciju λ
   visiem ports ∈ atomārs_modelis izpildīt
      maršrutēt modeļas izejas
   beigas
beigas
```

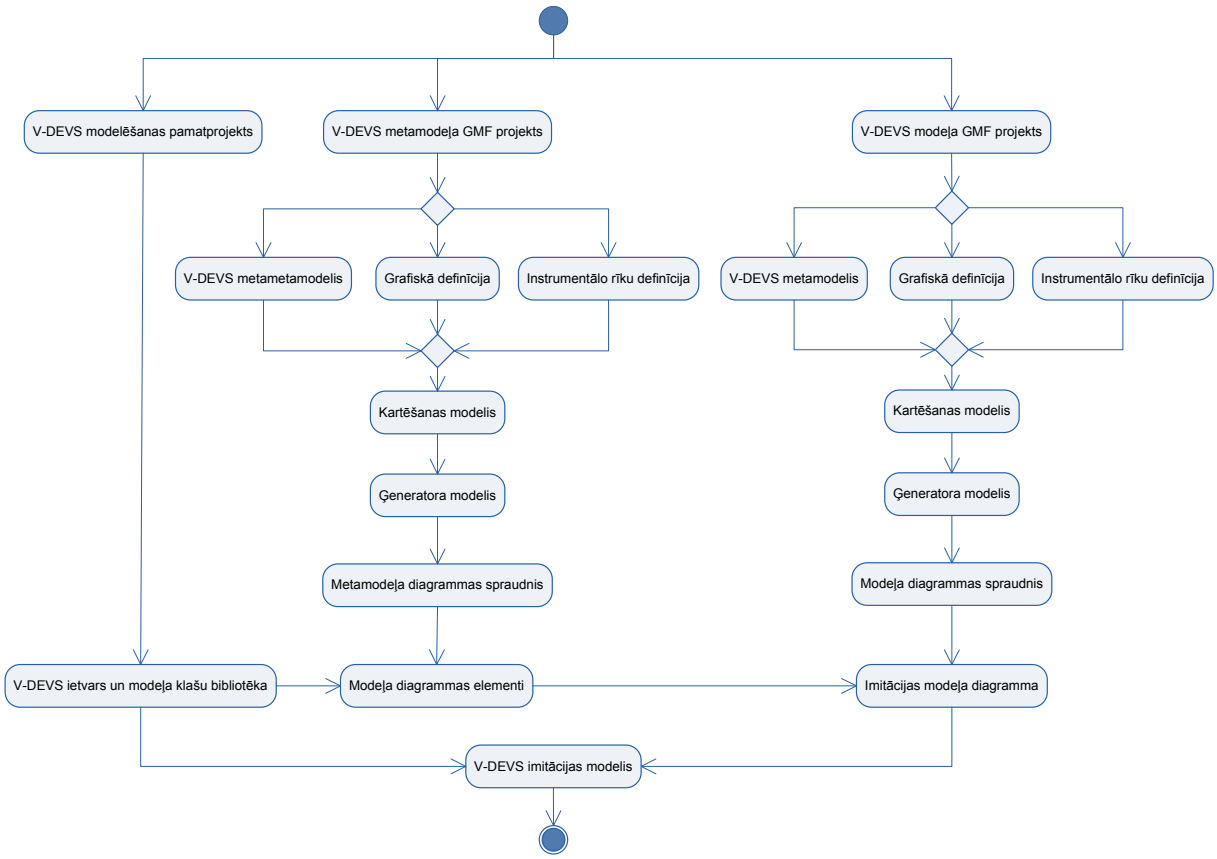
3.1.2. Modeļvadāmā pieeja praktiskajā realizācijā

Modeļvadāmā pieeja V-DEVS formālisma praktiskajā realizācijā tiek izmantota loģisko imitācijas modeļu izstrādei, balstoties uz teorētiskajām metamodelēšanas nostādnēm, kas ir

aprakstītas 2.3. apakšnodaļā. Tā kā visas imitācijas modelēšanas programmatūras komponenti ir realizēti ar Java programmēšanas valodu, tad arī Java tehnoloģiju izvēle modeļvadāmās pieejas praktiskajai realizācijai ir piemērota, izmantojot Eclipse GMF (*Graphical Modeling Framework*) projektu un ar to saistītās tehnoloģijas (EMF (*Eclipse Modeling Framework*), GEF (*Graphical Editing Framework*)) V-DEVS loģisko modeļu izstrādes vajadzībām.

GMF ir ģeneratīvs komponents un izpildes laika infrastruktūra grafisko redaktoru izstrādei, balstoties uz EMF un GEF tehnoloģijām. EMF ir modelēšanas ietvars un koda ģenerēšanas rīks modeļvadāmu sistēmu izveidei. Balstoties uz modeļa specifikāciju, EMF piedāvā rīkus Java klašu ģenerēšanai un to izpildes laika darbības nodrošināšanai. GEF projekts piedāvā līdzekļus grafisko redaktoru izstrādei. Tādējādi GMF ir tehnoloģija, kas apvieno EMF modelēšanas un GEF grafiskās metodes vienotā ietvarā.

Darba ietvaros GMF projekts tiek izmantots vizuālo loģisko V-DEVS modeļu redaktora izveidei, izmantojot modeļvadāmo pieeju. 3.6. attēlā ir dota shematiska V-DEVS metamodelēšanas moduļu izstrādes etapu un projektu shēma.



3.6. att. V-DEVS metamodelēšanas moduļu izstrādes etapi Eclipse GMF vidē

V-DEVS modelēšanas pamatprojekts ir izstrādāts ar standarta Java valodas līdzekļiem, kas realizē 3.1.1. apakšnodaļā aplūkoto V-DEVS simulatoru un atomāros modeļus. V-DEVS metamodeļa GMF projekts ir paredzēts V-DEVS imitācijas modeļu diagrammas vizuālo elementu

izstrādei, balstoties uz V-DEVS metametamodeli (P3.2. attēls 3.pielikumā). Bet V-DEVS modeļa GMF projekts ir paredzēts imitācijas modeļu diagrammu izveidei, balstoties uz V-DEVS metamodeli (P3.1. attēls 3.pielikumā). Tādējādi imitācijas modelēšana ir trīspakāpju processs, 1.etapā izstrādājot modeļu bibliotēku, 2.etapā ar V-DEVS metamodeļa redaktoru izveidojot modeļu grafiskos elementus, bet 3.etapā no esošu modeļa elementu bibliotēkas izstrādājot imitācijas modeli.

4. pielikumā ir parādīts pa kāpnēm lejup ripojošas lodes modelis (sīkāk šis modelis ir analizēts 3.3.2. apakšnodaļā), kas izstrādāts ar V-DEVS vizuālās imitācijas modelēšanas prototipu, izmantojot Eclipse GMF / EMF modeļvadāmo pieeju.

3.2. Vizualizācijas konveijera praktiskā realizācija

V-DEVS vizualizācijas konveijera praktiskā realizācija balstās uz 2.4. apakšnodaļā aplūkotajām vizualizācijas konveijera teorētiskajām nostādnēm.

3.2.1. Integrācija ar datorgrafikas un vizualizācijas programmatūras sistēmām

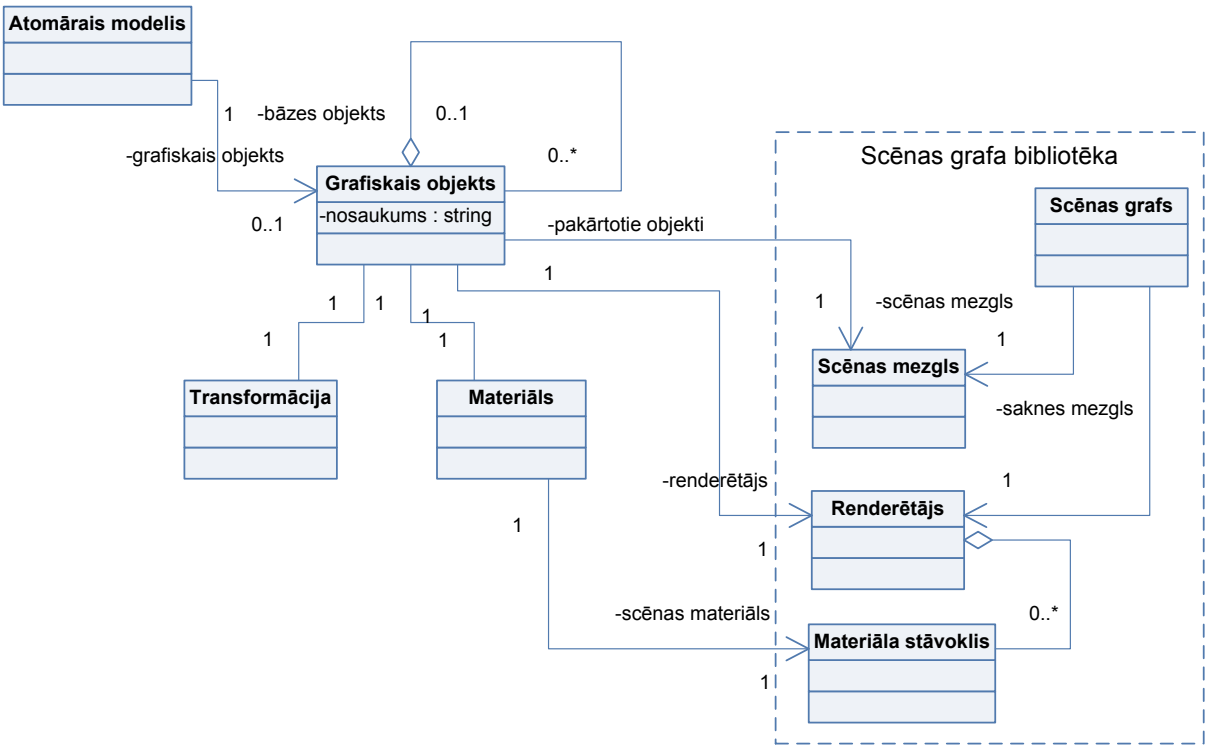
V-DEVS vizualizācijas konveijera definētā pieeja ir piemērota jaunas imitācijas modelēšanas vides projektēšanai un realizācijai, kad programmatūras sistēmas projektētāju un izstrādātāju rīcībā ir pietiekoši daudz izstrādes laika un cilvēkresursu. Taču datorgrafikas un vizualizācijas programmatūras sistēmas izstrāde ir ļoti sarežģīts un darbietilpīgs uzdevums [24], tādēļ praktiskas vizuālās imitācijas modelēšanas vides izstrādes gaitā bieži vien rodas uzdevums, kādā veidā imitācijas un modelēšanas apakšsistēmā integrēt esošas datorgrafikas un vizualizācijas programmatūras sistēmas.

Gatavu programmatūras bibliotēku vai pakotņu izmantošana atvieglo un paātrina imitācijas modelēšanas sistēmas izstrādi, tādēļ šī darba ietvaros ir veikta V-DEVS vizualizācijas konveijera teorētiskās koncepcijas praktiskā realizācija, integrējot to ar esošām datorgrafikas un vizualizācijas programmatūras sistēmām. V-DEVS formālisms kā universāla diskrētu notikumu un nepārtrauktu sistēmu modelēšanas paradigma, kas ir neatkarīga no konkrētas realizācijas, ir relatīvi vienkārši integrējams gan ar eksistējošām scēnas grafa, gan ar datu plūsmu vizualizācijas sistēmām.

Integrācija ar scēnas grafa balstītām sistēmām

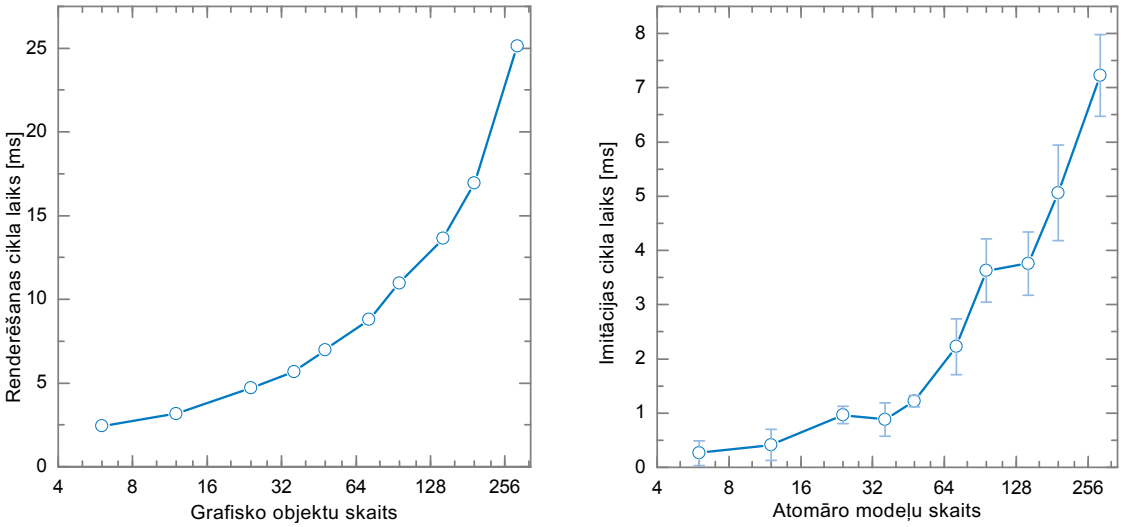
3.7. attēlā ir parādīta vispārēja klašu struktūra V-DEVS simulatora integrācijai ar scēnas grafa balstītām sistēmām. Darba izstrādes gaitā realizētajā V-DEVS prototipā ir veikta simulatora

integrācija ar atklātā pirmkoda scēnas grafa bibliotēku JME (JMonkeyEngine), taču šie paši realizācijas principi ir līdzīgā veidā ar nelielām izmaiņām pielietojami arī ar citām scēnas grafa bibliotēkām, piemēram, Java3D, OpenSceneGraph u.c.. Atomārais modelis Grafiskais objekts ietver scēnas grafa objektu funkcionalitāti, definējot transformāciju un objekta materiālu, kas nosaka tā vizuālo attēlojumu.



3.7. att. Uz integrāciju ar scēnas grafa bibliotēkām orientēta V-DEVS vizualizācijas konveijera bāzes klašu diagramma

Lai noskaidrotu V-DEVS simulatora veiktspēju mijiedarbībā ar scēnas grafa sistēmu, prototipa izstrādes gaitā ir veikti vairāki eksperimentāli testi, kuri rāda (3.8. attēls), ka simulatora kopējo veiktspēju visvairāk ietekmē tieši izmantotā scēnas grafa bibliotēka. Kā redzams 3.8. attēlā, palielinot dinamisko grafisko objektu skaitu imitācijas modelī 43 reizes (intervālā $6 - 258$ grafiskie objekti), renderēšanas cikla laiks palielinās ~ 10 reizes - no $2,5$ līdz $25,1$ milisekundēm. Savukārt imitācijas viena soļa cikla laiks palielinās ~ 24 reizes - no $0,3$ līdz $7,3$ milisekundēm. Kaut gan simulatora veiktspēja samazinās straujāk, nekā scēnas grafa renderēšanas veiktspēja, absolūtā laika mērogā scēnas grafa bibliotēkas veiktspēja atstāj lielāku negatīvo iespaidu uz kopējo vizuālā imitācijas modeļa darbību. No tā izriet secinājums, ka sarežģītāku imitācijas modeļu izstrādei ir nepieciešams uzlabot gan V-DEVS vizualizācijas konveijera, gan simulatora ātrdarbību.

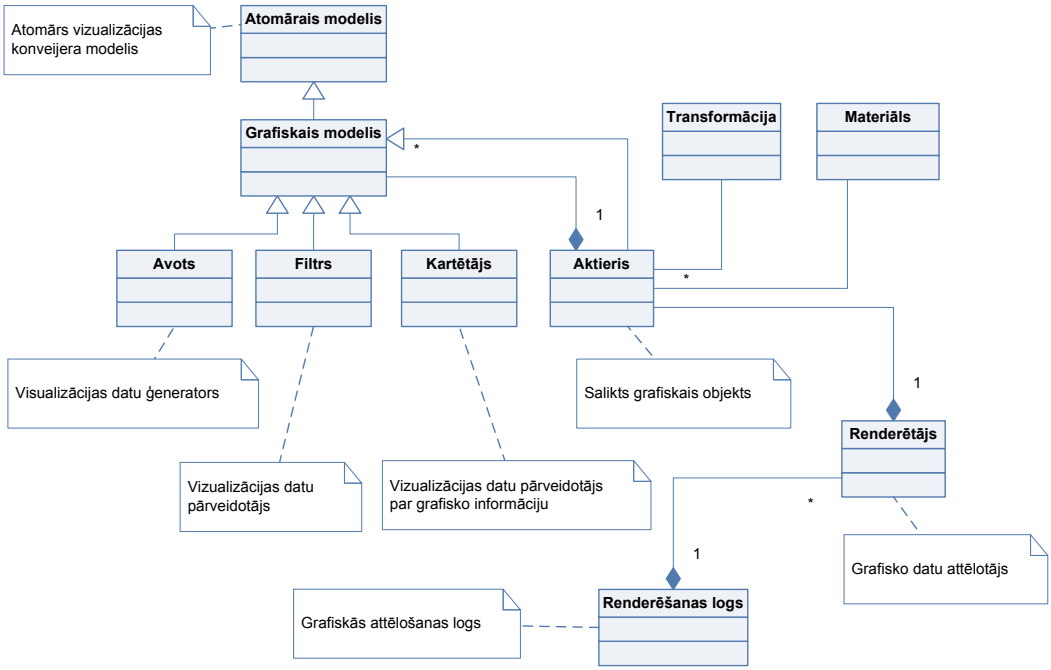


(a) Scēnas grafa renderēšanas laika atkarība no grafisko objektu skaita

(b) V-DEVS imitācijas cikla laika atkarība no scēnas grafa atomāro modeļu skaita

3.8. att. V-DEVS simulatora veiktspēja mijiedarbībā ar scēnas grafa bibliotēku

Integrācija ar datu plūsmu balstītām sistēmām



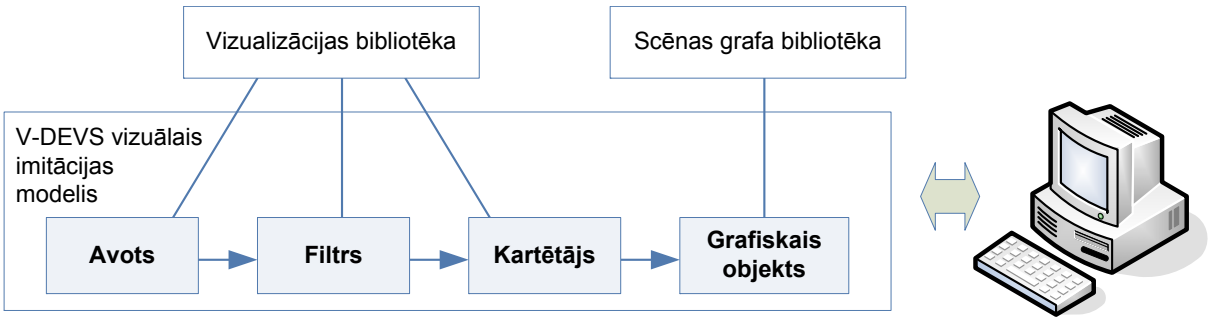
3.9. att. Uz integrāciju ar vizualizācijas bibliotēkām orientēta V-DEVS vizualizācijas konveijera bāzes klašu diagramma

V-DEVS sistēmas prototipa integrācijai ar datu plūsmu balstītām vizualizācijas sistēmām ir izmantota atklātā pirmkoda bibliotēka VTK. 3.9. attēlā ir parādīta ar vizualizācijas bibliotēku

integrēta V-DEVS vizualizācijas konveijera klašu struktūra. Katrs atomārais V-DEVS modelis ietver noteiktu izmantotās vizualizācijas bibliotēkas komponentu.

Vizualizācijas konveijera apvienojums ar vizualizācijas un scēnas grafa bibliotēkām

Scēnas grafu un vizualizācijas konveijeru ir iespējams apvienot un izmantot vienā lietojumā, un šādā gadījumā vizualizācijas konveijers ir grafisko primitīvu ģenerators, bet scēnas grafs vada grafiskās scēnas renderēšanas procesu (3.10. attēls). Tā kā V-DEVS formālisms definē hierarhisku modulāru modelēšanas pieeju, tad vizulizācijas un scēnas grafa sistēmas apvienošana ir relatīvi vienkārša. Galvenais nosacījums ir tāds, lai vizualizācijas modeļu izejas saskarne (2.1.2. apakšnodaļa) būtu savietojama ar scēnas grafa modeļa ieejas saskarni. Piemēram, 3.10. attēlā kartētāja modeļa izejas datu tipam ir jāsakrīt ar grafiskā objekta modeļa ieejas uzturēto datu tipu, lai grafiskais objekts spētu apstrādāt ieejā saņemtos ģeometrijas datus.



3.10. att. V-DEVS vizualizācijas konveijera apvienojuma piemērs ar vizualizācijas un scēnas grafa bibliotēku

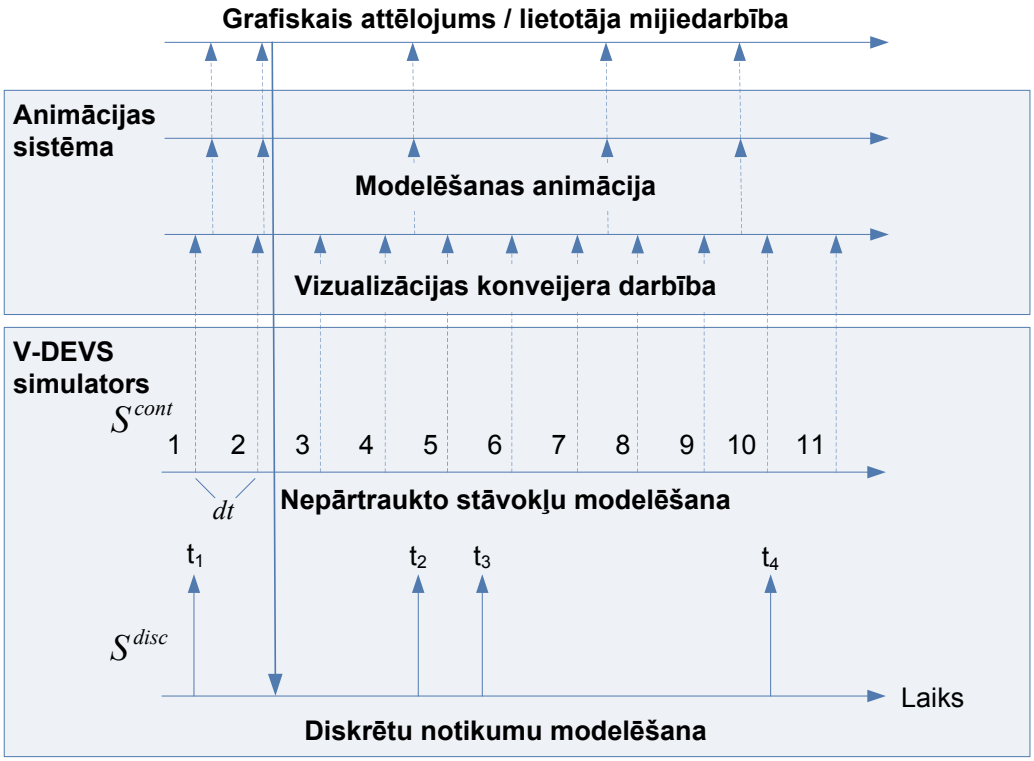
3.2.2. Lietotāja mijiedarbība

V-DEVS formālisms sniedz iespējas imitācijas modeļa lietotājam veikt interaktīvu mijiedarbību ar modeli tieši imitācijas procesa laikā. Vienīgais pamatnosacījums šeit ir reālā laikā mērogots imitācijas process, jo ir nepieciešams nodrošināt, ka lietotāja mijiedarbību ir iespējams sinhronizēt ar imitācijas procesu. Virtuālā jeb loģiskā laika imitācijas gaitā imitācijas laika virzība ir maksimāli ātra, tādēļ šādā imitācijas procesā nav iespējama lietotāja mijiedarbības un imitācijas procesa sinhronizācija.

Lietotāja mijiedarbību V-DEVS formālisma kontekstā ir iespējams definēt kā ārēju diskrētu notikumu virkni, kas dinamiski iedarbojas uz imitācijas modeli tā darbības laikā. No tā izriet, ka lietotāja mijiedarbībai ar imitācijas modeli ir piemērojami tie paši V-DEVS formālisma pamatprincipi, kā citiem imitācijas modeļiem, realizējot mijiedarbības apstrādes uzdevumu atomāra modeļa veidā. Būtiska atšķirība šeit ir tā, ka atomārais mijiedarbības modelis nevar prognozēt, cik ilgi tas atradīsies noteiktā stāvoklī. Tādēļ laika virzības funkcija vienmēr atgriež

bezgalīgu laika intervālu $ta = \infty$, kas nozīmē, ka mijiedarbības imitācijas modelis ir pasīvs V-DEVS modelis.

Diskrētu notikumu imitācijas izpilde ir V-DEVS simulatora kodola realizācijas pamatmērķis, kurā tāpat kā klasiskajā diskrētu notikumu imitācijā tiek plānoti nākošie notikumi. Piemēram, kā redzams 3.11. attēlā, notikums t_2 tiek plānots pēc notikuma t_1. V-DEVS sistēmā tiek ņemti vērā arī ārēji notikumi, kurus ģenerē modeļa lietotājs interaktīvās mijiedarbības laikā.



3.11. att. V-DEVS imitācijas modelēšanas procesa virzība laikā

Jebkuru interaktīvo lietotāja saskarnes elementu ir iespējams atvasināt no diskrēta bāzes V-DEVS modeļa saskarnes modeļa:

$$\text{Saskarnes modelis} = \left\langle X, Y, S, \delta_{ext}, \delta_{int}^{discr}, \lambda^{discr}, ta^{discr} \right\rangle \tag{3.1}$$

Modeļa ieejas: $X = \{\varnothing\}$

Modeļa izejas: $Y = \{(\text{Izeja}, \text{Saskarnes elements vērtība})\}$

Stāvokļu mainīgie: $S = \{$”pasīvs”, ”aktīvs”$\}$;

$\delta_{ext}(s, e, x) = \varnothing$;

δ_{int}^{discr} (”aktīvs”) = ”pasīvs”;

λ^{discr} (”aktīvs”) = saskarnes lietotāja ievadītā vērtība;

ta^{discr} (”aktīvs”) $= 0$;

ta^{discr} (”pasīvs”) $= +\infty$.

Saskarnes V-DEVS modelis apstrādā lietotāja ievadītās vērtības, piemēram, bīdņa tekošo pozīciju vai izvēles rūtiņas statusu, modeļa izejā ģenerējot atbilstošu notikumu. Reaģējot uz lietotāja ievadi, modelis pāriet aktīvā stāvokli, ģenerē izejas notikumu un pāriet atpakaļ pasīvā stāvoklī līdz nākošajai lietotāja reakcijai.

3.3. V-DEVS simulatora verifikācija un testēšana

V-DEVS formālismu realizējot praktiskā imitācijas sistēmas prototipā, ir nepieciešams pārliecināties, ka izstrādātās sistēmas pamatā izmantotie algoritmi ir korekti - tātad ir nepieciešams veikt sistēmas darbības verifikāciju un testēšanu. Šim nolūkam ir izstrādāti divi relatīvi vienkārši, viegli saprotami imitācijas modeļi. Pirmais imitācijas modelis modelē otrās kārtas rimstoša lineāra oscilatora kā nepārtrauktas dinamiskas sistēmas darbību. Otrs apskatāmais modelis modelē pa kāpnēm lejup ripojošas bumbas trajektoriju, ietverot sevī gan diskrētu notikumu, gan nepārtrauktu sistēmas darbību. Šī pētījuma ietvaros veiktajiem V-DEVS simulatora testiem ir šādi mērķi:

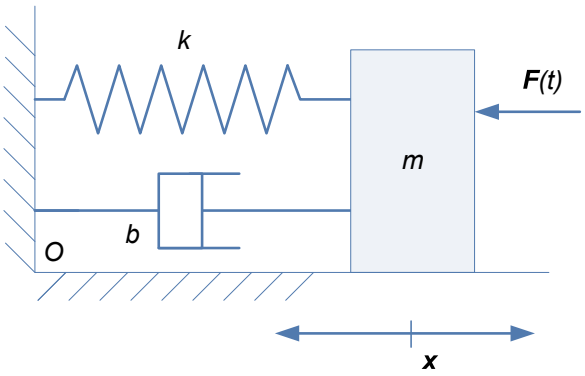
- Tā kā tālāk apskatāmie imitācijas modeļi ir realizēti daudzās imitācijas modelēšanas sistēmās, tad šādu modeļu izmantošana ir V-DEVS simulatorā realizēto algoritmu un iegūto rezultātu verifikācijai.
- Veiktie testi ir izstrādātās sistēmas validācijas pamatojums attiecībā uz tās realizētajām tehniskajām iespējām un pielietojumu.

3.3.1. Otrās kārtas rimstoša lineāra oscilatora modelis

Otrās kārtas rimstoša lineāra oscilatora modelis apraksta no fizikas kursa zināmu vienkāršu mehānisku sistēmu, kas sastāv no atsperei un amortizatoram piestiprināta priekšmeta. Dotais modelis ir izveidots V-DEVS imitācijas modelēšanas vidē ar mērķi verificēt un nodemonstrēt V-DEVS modeļu realizācijas pamatprincipus.

Problēmas formulējums

Materiāls punktveida ķermenis ar masu m, kurš piestiprināts atsperei un amortizatoram, pārvietojas izstieptas vai saspiestas atsperes spēka iedarbības rezultātā, radot mehāniskās svārstības. Atspere un amortizators ir nostiprināti koordinātu sistēmas sākumpunktā O (3.12. attēls).



3.12. att. Mehāniska sistēma ar atsperi un amortizatoru

Tā kā materiālā ķermeņa atrašanās vietu katrā laika momentā raksturo tā vektoriālās koordinātes x, tad šī ķermeņa kustība, balstoties uz otro Ņūtona likumu, ir izsakāma sekojošā lineāra otrās kārtas diferenciālvienādojuma formā:

$$m\ddot{x} = -k\mathbf{x} - b\dot{x}, \tag{3.2}$$

kur x - priekšmeta pozīcijas vektors;

$\dot{x} = v$ - priekšmeta momentānais ātrums;

$\ddot{x} = a$ - priekšmeta momentānais paātrinājums;

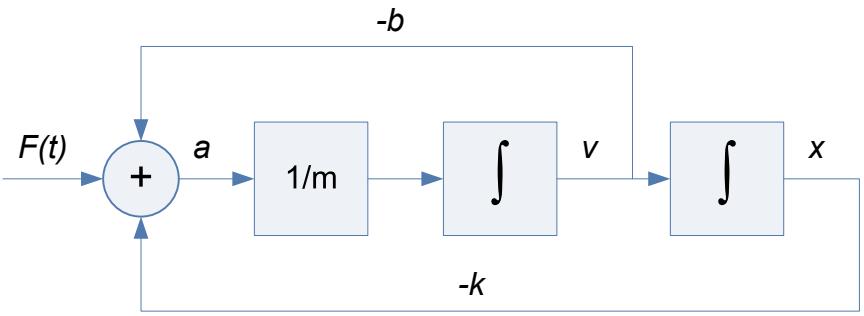
m - priekšmeta masa;

k - atsperes stinguma koeficients;

b - amortizācijas (berzes) koeficients.

Modeļa apraksts

3.13. attēlā ir parādīta otrās kārtas rimstoša oscilatora modeļa struktūrshēma, kas praktiski realizē 3.2 vienādojumu.



3.13. att. Otrās kārtas rimstoša lineāra oscilatora struktūrshēma

3.13. attēlā redzamie modeļa elementu apzīmējumi (3.1. tabula) ir aizgūti no imitācijas

modelēšanas programmatūras PowerDEVS [46].

3.1. tabula

Otrās kārtas rimstoša lineāra oscilatora modeļa elementu apzīmējumi

Elementa apzīmējums	Nosaukums
	Summators
	Konstants reizinātājs
	Integrators

Imitācijas modelim ir definējami šādi parametri:

$X = \{\text{Sākt}, \text{Sākuma\_pozīcija}, \text{Sākuma\_ātrums}, \text{Elastība}, \text{Berze}, \text{Masa}\}$;

$Y^{discr} = \varnothing$;

$Y^{cont} = \{\text{Transformācija}, \text{Krāsa}\}$;

$S^{discr} = \{(\text{fāze}, k, b, m) \mid \text{fāze} \in \{\text{APSTĀDINĀTS}, \text{KUSTĪBA}\}\}$;

$S^{cont} = \{T, \dot{p}, \ddot{p}, v\}$;

$\dot{p}, \ddot{p}, v \in \mathbb{R}^3$;

$k, b, m \in \mathbb{R}$.

Sākuma stāvoklis:

fāze = APSTĀDINĀTS;

$T.P = [0,0,0]$;

$v = [0,0,0]$.

Tā kā jebkura V-DEVS modeļa uzvedību nosaka to iekšējās, ārējās pārejas, izejas un laika ritējuma funkcijas, tad arī šajā apakšnodaļā apskatāmajam modelim ir nepieciešams definēt modeļa dinamiku noteicošās funkcijas. 3.2 algoritms definē modeļa reakciju uz ieejas notikumiem. 3.3 algoritms definē modeļa iekšējās diskrētās stāvokļu pāreju apstrādi. Modeļa diskrētā laika ritējuma funkcija atgriež konstantu vērtību IMITĀCIJAS_ILGUMS (3.4 algoritms), kas nosaka oscilatora svārstību imitācijas laiku līdz modeļa apstādināšanas brīdim. Nepārtrauktā laika ritējuma funkcija atgriež konstantu lielumu RENDERĒŠANAS_PERIODS (3.5 algoritms), kas nosaka oscilatora svārstību imitācijas laika soli un renderēšanas laika intervālu. Iekšējā nepārtrauktā pārejas funkcija tiek izpildīta ar konstantu laika intervālu RENDERĒŠANAS_PERIODS, nosakot momentāno ķermeņa ātrumu un pozīciju (3.6 algoritms). 3.7 algoritms apraksta nepārtraukto izejas funkciju.

3.2. algoritms Oscilatora modeļa ārējās pārejas funkcija $\delta_{ext}(s,e,x)$

ja $x =$ Sākuma_pozīcija **tad**
  T.P $\Leftarrow$ Sākuma_pozīcija
citādi ja $x =$ Sākuma_ātrums **tad**
  v $\Leftarrow$ Sākuma_ātrums
citādi ja $x =$ Elastība **tad**
  $k \Leftarrow$ Elastība
citādi ja $x =$ Berze **tad**
  $b \Leftarrow$ Berze
citādi ja $x =$ Masa **tad**
  $m \Leftarrow$ Masa
citādi ja $x =$ Sākt **tad**
  fāze $\Leftarrow$ KUSTĪBA
beigas (ja)

3.3. algoritms Oscilatora modeļa diskrētās iekšējās pārejas funkcija $\delta_{int}^{discr}(s)$

ja fāze = KUSTĪBA **tad**
  fāze $\Leftarrow$ APSTĀDINĀTS
beigas (ja)

3.4. algoritms Oscilatora modeļa diskrētā laika ritējuma funkcija $ta^{discr}(s)$

atgriezt IMITĀCIJAS_ILGUMS

3.5. algoritms Oscilatora modeļa nepārtrauktā laika ritējuma funkcija $ta^{cont}(s)$

atgriezt RENDERĒŠANAS_PERIODS

3.6. algoritms Oscilatora modeļa nepārtrauktās iekšējās pārejas funkcija $\delta_{int}^{cont}(s)$

ja fāze = KUSTĪBA **tad**
  $\dot{\mathbf{p}} \Leftarrow \mathbf{v}$
  $\ddot{\mathbf{p}} \Leftarrow \frac{-k \times \mathbf{T.P} - b \times \mathbf{v}}{m}$
citādi
  $\dot{\mathbf{p}} \Leftarrow [0,0,0]$
  $\ddot{\mathbf{p}} \Leftarrow [0,0,0]$
beigas (ja)

3.7. algoritms Oscilatora modeļa nepārtrauktās izejas funkcija λ^{cont}

Transformācija.P $\Leftarrow$ **T.P**
Krāsa $\Leftarrow \ddot{\mathbf{p}}$

3.3.2. Pa kāpnēm lejup ripojošas lodes modelis

Tipisku kombinētas diskrētu notikumu un nepārtrauktas sistēmas imitācijas modelēšanas piemēru raksturo pa kāpnēm lejup ripojošas elastīgas lodes (bumbas) modelis [14, 45]. Dotais modelis ir izstrādāts ar mērķi veikt 2.2. apakšnodaļā aprakstītā V-DEVS kvantētu stāvokļu interpolatora praktiskās realizācijas verifikāciju un novērtēt tā darbības efektivitāti imitācijas modelēšanas uzdevumu risināšanā.

Problēmas formulējums

Ripojot lejup pa kāpnēm, lode pārvietojas divās dimensijās, tādējādi ripošanas nosacījumi ir atkarīgi no diviem mainīgajiem (x un y). Balstoties uz pieņēmumu, ka apskatāmajai sistēmai tiek piemērots viens modelis, kamēr lode atrodas atlēcienā gaisā un cits modelis lodes ripošanas laikā, pa kāpnēm lejup ripojošas lodes modeli iespējams definēt sekojošā veidā:

$$\dot{x} = v_x$$

$$\dot{v}_x = -\frac{b_a}{m} \cdot v_x$$

$$\dot{y} = v_y$$

$$\dot{v}_y = -g - \frac{b_a}{m} \cdot v_y - s_w \cdot \left[\frac{b}{m} \cdot v_y + \frac{k}{m} \left(y - int\left(h + 1 - x\right) \right) \right],$$

kur m - bumbas masa, b_a - gaisa pretestības koeficients, g - gravitācijas konstante, b - berzes koeficients, k - elastības koeficients.

$$s_w\left(t\right) = \begin{cases} 1, & \text{kad lode atrodas uzkāpnēm} \\ 0, & \text{kad lode atrodas gaisā} \end{cases}$$

Funkcija $int(h+1-x)$ definē kāpņu augstumu dotajā pozīcijā x (h ir pirmā kāpņu soļa augstums).

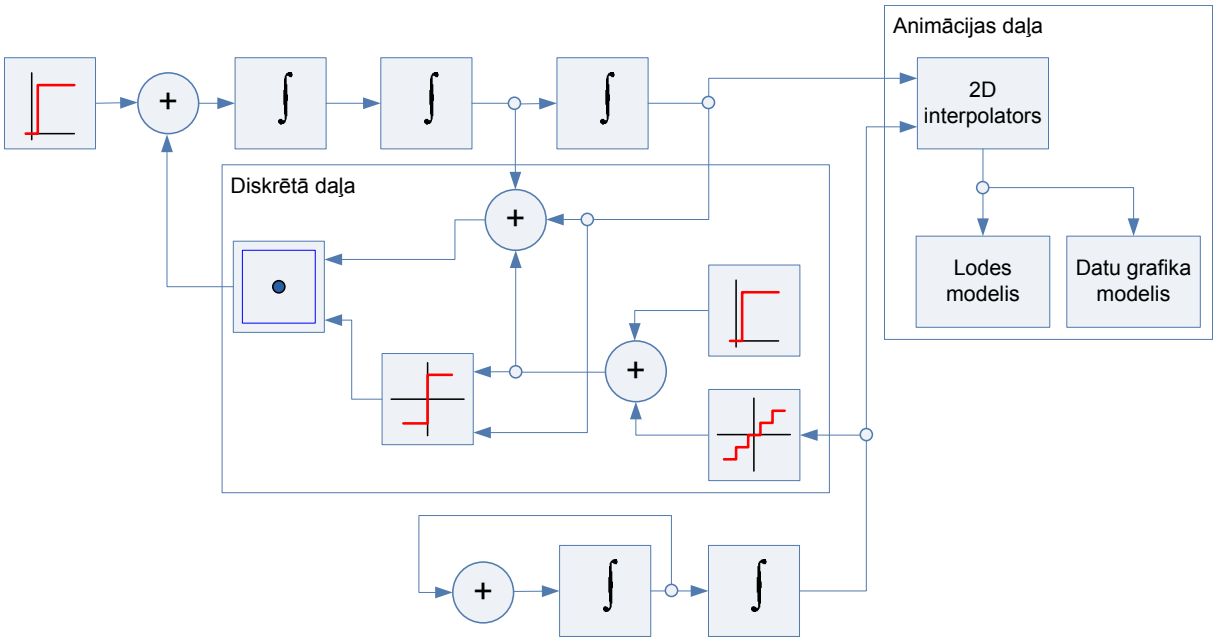
Diskrēto stāvokļu notikumi tiek ģenerēti, kad ir spēkā vienādība

$$y = int(h+1-x).$$

Pielietojot kvantētu stāvokļu V-DEVS vizuālo imitācijas modelēšanu šī uzdevuma izpildē, rodas problēma, ka pietiekoši precīzu modelēšanas rezultātu iegūšanai izmantojamais kvantēšanas solis Δq nav derīgs vienmērīgas imitācijas animācijas procesa nodrošināšanai.

Modeļa apraksts

V-DEVS imitācijas modelis, kura struktūra redzama 3.14. attēlā, realizē iepriekš apskatīto pa kāpnēm lejup ripojošas lodes kustības matemātisko formulējumu. Vienmērīgas imitācijas izpildes animācijas procesa nodrošināšanai tiek izmantots V-DEVS 2D kvantētu stāvokļu interpolators. 4. pielikumā ir parādīts 3.14. attēlā redzamās struktūras vizuālais loģiskais modelis, kas izstrādāts izmantojot Eclipse GMF / EMF modeļvadāmo pieeju.

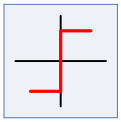
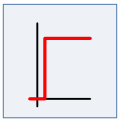
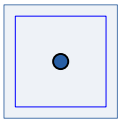
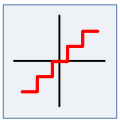


3.14. att. Pa kāpnēm lejup ripojošas lodes modeļa struktūra

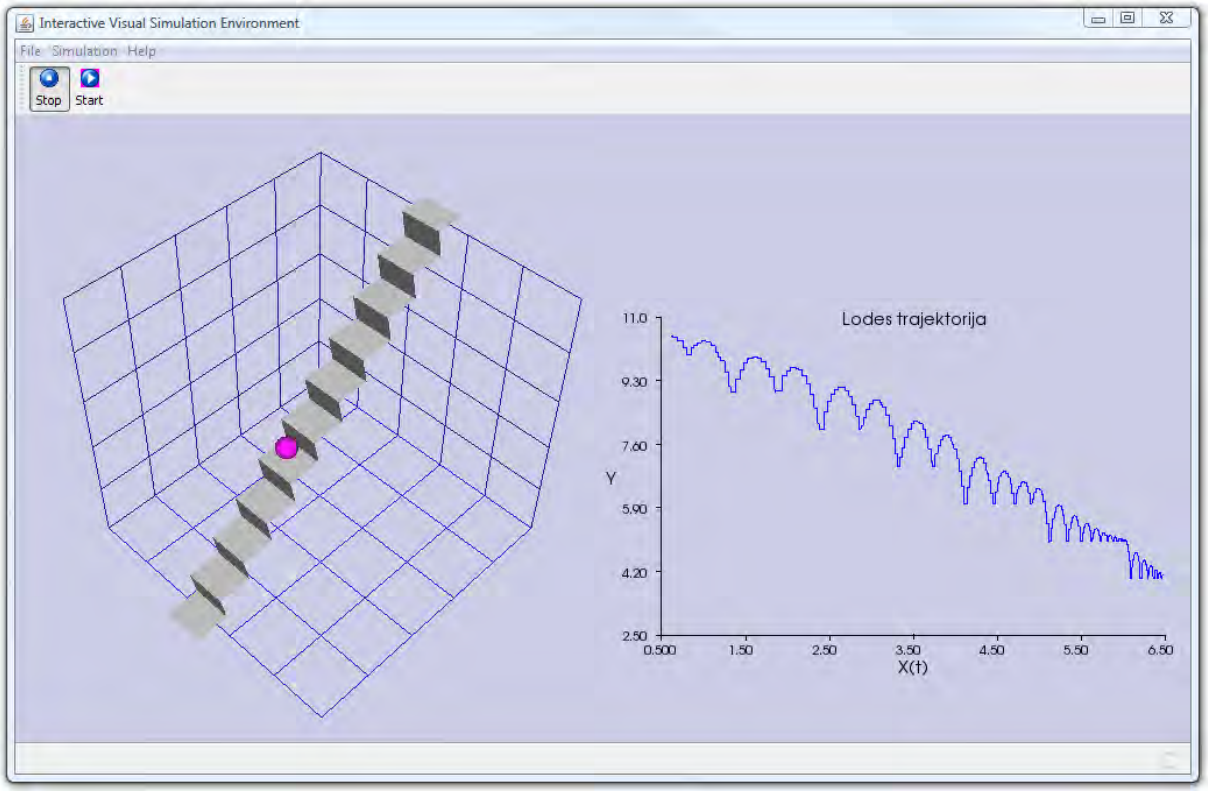
Šajā imitācijas modelī ir ieviesti vairāki papildus modeļa elementi (3.2. tabula) nepieciešamās funkcionalitātes nodrošināšanai, kuru izveidei izmantoti V-DEVS formālisma principi. Uz ekrāna redzamo grafisko attēlu ģenerē lodes un grafika modelis, kas ir pievienots 2D kvantētu stāvokļu interpolatora izejai. Lodes un grafika modeļi ir realizēti, izmantojot V-DEVS vizualizācijas konveijera pieeju.

3.2. tabula

Pa kāpnēm lejup ripojošas lodes modelī jaunieviesto elementu apzīmējumi

Elementa apzīmējums	Nosaukums	Elementa apzīmējums	Nosaukums
	Komparators		Soļa ģenerators
	Reizinātājs		Kvantētājs

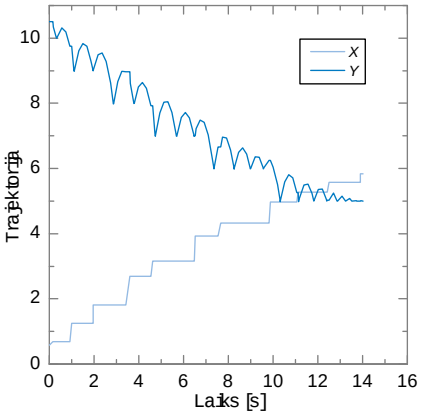
3.15. attēlā ir parādīta pa kāpnēm lejup ripojošas lodes modeļa 3D attēls, kas iegūts ar interaktīvās vizuālās V-DEVS imitācijas modelēšanas vides prototipu.



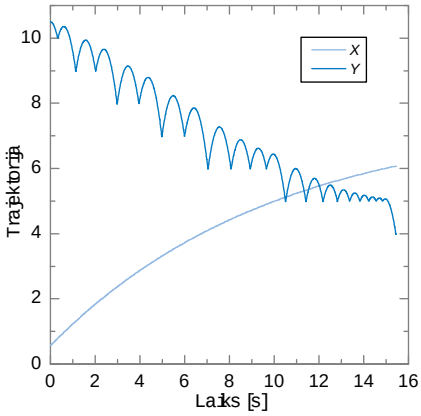
3.15. att. Pa kāpnēm lejup ripojošas lodes modeļa piemērs

Izstrādātā V-DEVS kvantētu stāvokļu interpolatora efektivitātes noteikšanai ar doto imitācijas modeli ir veikta virkne eksperimentu, salīdzinot imitācijas gaitā iegūtos lodes kustības trajektorijas datus ar un bez kvantētu stāvokļu interpolatora izmantošanas. 3.16. attēlā ir parādīta lodes modeļa ar kvantētu stavokļu interpolatoru kustības trajektorijas maiņa pa X un Y asi pie divām dažādām kvantēšanas soļa Δq vērtībām, un redzams, ka, samazinot kvantēšanas soli, palielinās arī modelētās trajektorijas vienmērība.

3.17. attēlā parādīta lodes kustības trajektorija dažādām kvantēšanas soļa Δq vērtībām ar un bez kvantētu stāvokļu interpolatora izmantošanas. Kā redzams, pat pie liela kvantēšanas soļa interpolators ļauj ievērojami uzlabot trajektorijas vienmērību.

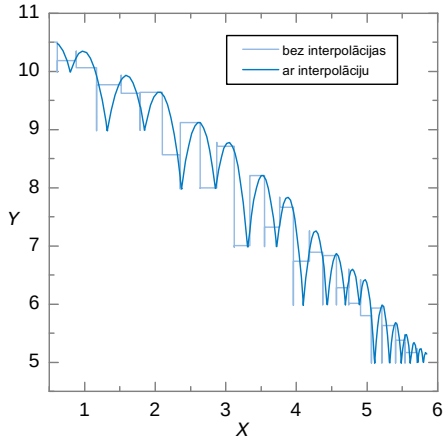


(a) $\Delta q = 1,0 \times 10^{-3}$

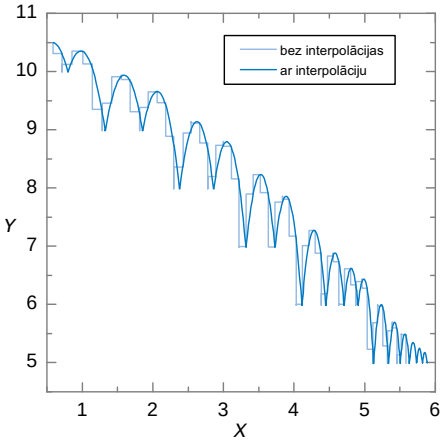


(b) $\Delta q = 1,0 \times 10^{-7}$

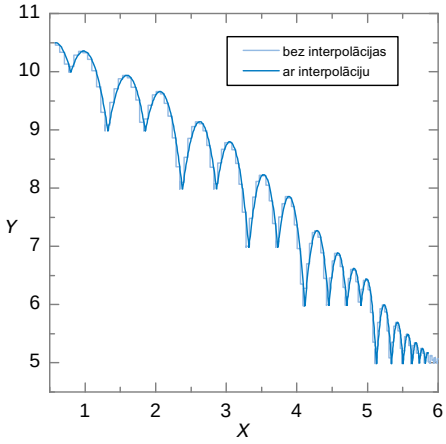
3.16. att. Lodes modeļa kustības trajektorijas maiņa laikā pa X un Y asīm pie dažādām kvantēšanas soļa Δq vērtībām, izmantojot V-DEVS kvantētu stāvokļu interpolatoru



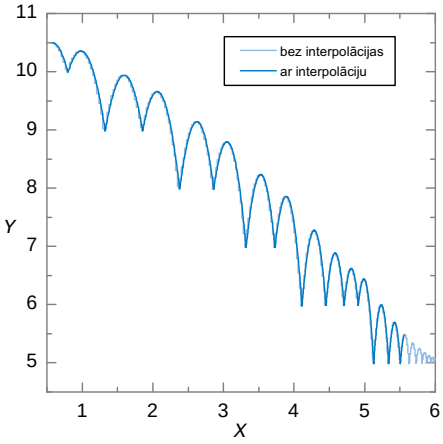
(a) $\Delta q = 1,0 \times 10^{-4}$



(b) $\Delta q = 1,0 \times 10^{-5}$



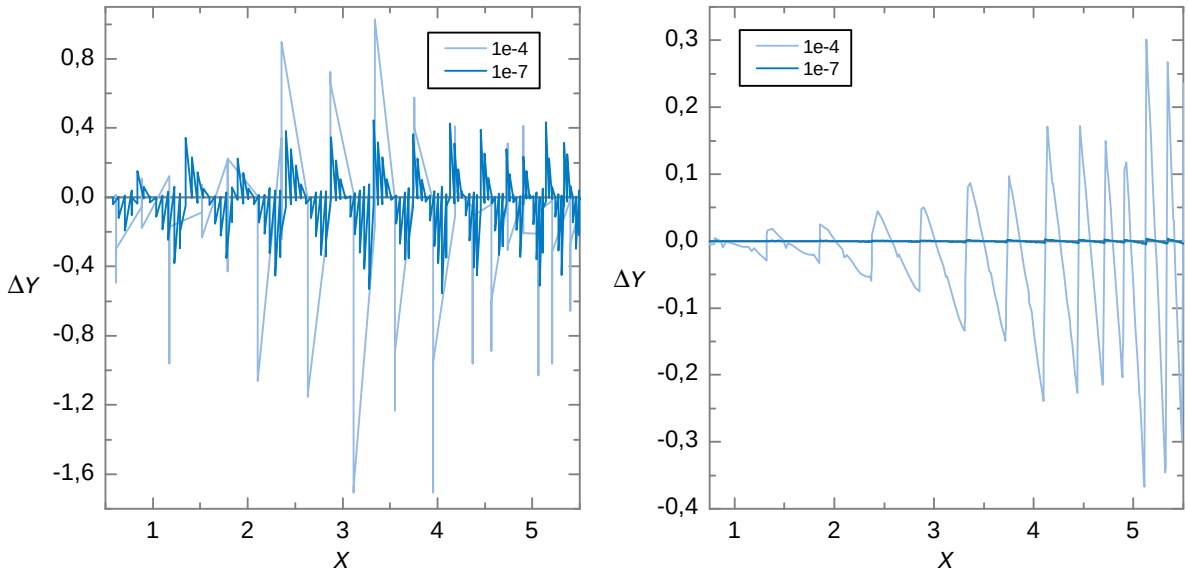
(c) $\Delta q = 1,0 \times 10^{-6}$



(d) $\Delta q = 1,0 \times 10^{-7}$

3.17. att. Lodes kustības trajektorija pa X un Y asīm pie dažādām parametra Δq vērtībām ar un bez interpolācijas pielietošanas

Ar mērķi veikt kvantitatīvu salīdzinošu bumbas kustības trajektorijas precizitātes analīzi ir izmantota starpības funkcija $\Delta Y = Y_{\triangle qe} - Y_{\triangle q}$ starp etalontrajektorijas $Y_{\triangle qe}$ un salīdzināmās trajektorijas datiem $Y_{\triangle q}$. Par etalontrajektoriju ir izvēlēti kustības trajektorijas dati pie parametra $\Delta q = 1,0 \times 10^{-7}$, imitācijas modelēšanu veicot ar V-DEVS interpolatora palīdzību. 3.18. attēlā ir parādīti iegūtie starpības dati, salīdzinot etalontrajektorijas datus ar iegūtajiem datiem, imitāciju veicot ar mazāku parametra $\triangle q$ vērtību.



(a) Starpības dati, neizmantojot interpolāciju

(b) Starpības dati, izmantojot interpolāciju

3.18. att. Imitācijas modeļa darbības precizitāti raksturojoši lodes kustības trajektorijas starpības dati $\Delta Y = Y_{1,0\times10^{-7}} - Y_{\triangle q}$ pie dažādām kvantēšanas soļa $\triangle q$ vērtībām

Kā redzams, jo mazāka izvēlētā parametra $\triangle q$ vērtība (lielāka imitācijas modeļa precizitāte), jo mazāks arī iegūtās trajektoriju starpības lielums, un tādējādi, salīdzināmo trajektoriju dati vairāk tuvinās etalontrajektorijai. No 3.18. attēlā redzamajiem imitācijas modelēšanas rezultātiem var secināt, ka V-DEVS interpolatora izmantošana ļauj aptuveni 5 reizes uzlabot modeļa izejas precizitāti pie nemainīgas precizitātes parametra Δq vērtības.

Lai savā starpā būtu iespējams statistiski salīdzināt lodes pārvietošanās trajektorijas līknes, kas iegūtas pie dažādām parametra $\triangle q$ vērtībām, tiek izmantota kroskorelācijas noteikšanas metode [77], kas ir statistisks divu signālu līdzības mērs. Divu vienāda garuma M signālu $f(n)$ un $g(n)$ korelāciju ir iespējams aprēķināt pēc šādas formulas:

$$y(m) = \sum_{n=0}^{M-1} f(n)\, g(n-m),$$

kur $m = 0, 1, \ldots, M-1$ - kroskorelācijas nobīde.

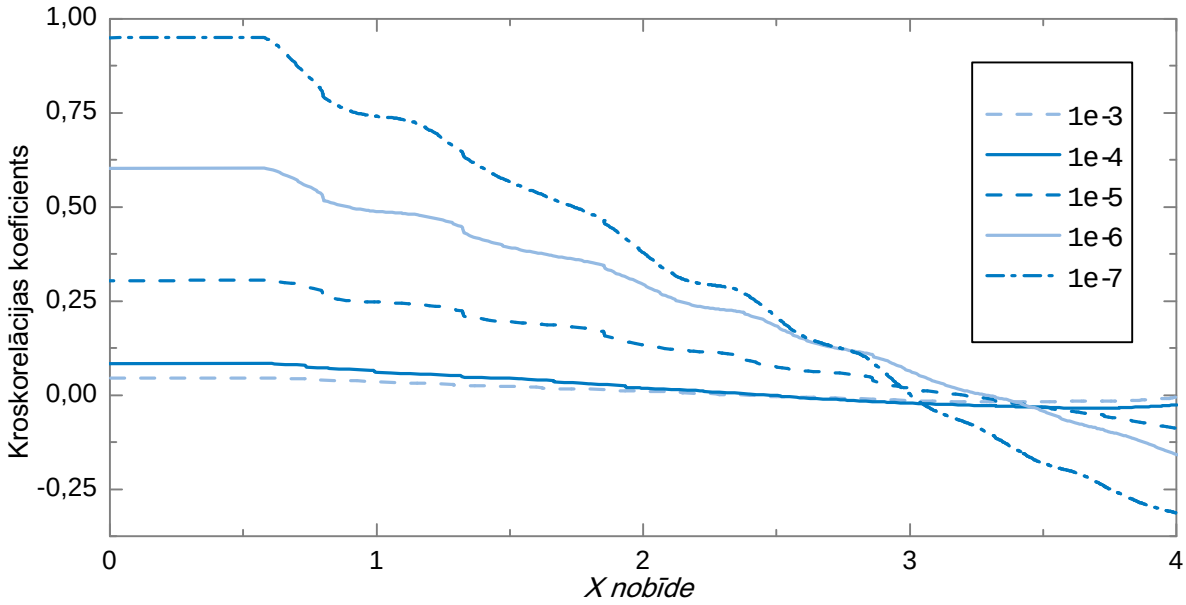
Aprēķinātā korelācijas vērtība parāda salīdzināmo signālu līdzību - jo lielāka šī vērtība, jo

lielāka šo divu signālu savstarpējā līdzība. Ir jāņem vērā, ka lielums $y(m)$ var mainīties ļoti plašā diapazonā, tādēļ praktiskajā lietošanā ērtāka ir normalizētā kroskorelācija:

$$y(m) = \sum_{n=0}^{M-1} f_{norm}(n)\, g_{norm}(n-m), \tag{3.3}$$

kur $f_{norm}(n) = \frac{f(n)}{\sqrt{\sum_{i=0}^{M-1}(f(n))^2}}$, $g_{norm}(n) = \frac{g(n)}{\sqrt{\sum_{i=0}^{M-1}(g(n))^2}}$ - normalizētie signāli.

Pa kāpnēm lejup ripojošas lodes modeļa trajektoriju salīdzināšanai ir izmantota matemātiskās statistiskas programmatūras OriginPro 8.0 iebūvētā kroskorelācijas noteikšanas funkcionalitāte, kas aprēķiniem izmanto iepriekš minēto 3.3 formulu, datu apstrādes paātrināšanai aprēķinus veicot frekvenču apgabalā, šim nolūkam izmantojot diskrēto Furjē transformāciju. 3.19. attēlā ir redzami iegūtie kroskorelācijas rezultāti, raksturojot to, ka samazinot kvantēšanas soli, palielinās modeļa darbības precizitāte. Palielinoties modeļa darbības precizitātei, samazinās atšķirība (palielinās kroskorelācija) starp interpolatora un modeļa tiešajiem rezultātiem. Tādējādi var secināt, ka interpolatora algoritma realizācija ir precīza un adekvāta izvirzītajam uzdevumam.



3.19. att. Lodes kustības trajektorijas datu kroskorelācija pie dažādām kvantēšanas soļa $\triangle q$ vērtībām ar un bez interpolācijas pielietošanas

3.4. Kopsavilkums un secinājumi

Šajā promocijas darba nodaļā ir aprakstītas promocijas darbā piedāvātā V-DEVS formālisma praktiskās realizācijas detaļas programmatūras sistēmas veidā, kā arī eksperimentālās verifikācijas un testēšanas rezultāti.

Nodaļā paveiktais:

- projektēta un praktiski realizēta interaktīva vizuālā imitācijas modelēšanas sistēma, kas pamatojas uz piedāvāto V-DEVS formālismu;
- definēta modeļa vadāmā pieeja V-DEVS formālisma praktiskajā realizācijā;
- eksperimentāli pārbaudīts izstrādātais imitācijas sistēmas prototips.

Sasniegtie rezultāti:

- izstrādāta praktiska V-DEVS formālisma arhitektūra;
- izstrādāts vizualizācijas konveijers interaktīvai mijiedarbībai ar imitācijas modeli;
- izstrādāti divi V-DEVS formālisma realizācijas algoritmu varianti; pirmais algoritms balstās uz klasisko DEVS simulatoru specifikāciju, savukārt otrs algoritms ir balstīts uz viena universāla simulatora un prioritāšu rindas izmantošanas principiem.

Galvenie secinājumi ir šādi:

- V-DEVS formālisma praktiskā realizācija un veiktās eksperimentālās pārbaudes parāda, ka piedāvātās V-DEVS teorētiskās koncepcijas ir pareizas un uz to bāzes ir iespējams izstrādāt reālu imitācijas modelēšanas sistēmu;
- izstrādāto simulatora algoritmu priekšrocība ir tā, ka tie nav tiešā veidā atkarīgi no izmantotās vizualizācijas sistēmas, jo vizualizācijas uzdevums tiek realizēts atomāro modeļu līmenī;
- izstrādātā kvantētu stāvokļu interpolatora izmantošana nepārtrauktu sistēmu imitācijā ļauj ievērojami palielināt vizualizācijas precizitāti, nepalielinot nepieciešamo skaitļošanas resursus.

4. V-DEVS FORMĀLISMA PRAKTISKAIS PIELIETOJUMS DINAMISKU SISTĒMU IMITĀCIJAS MODELĒŠANĀ

Šajā nodaļā ir aprakstīts V-DEVS formālisma praktiskais pielietojums dinamisku sistēmu imitācijas modelēšanā, apskatot automatizētās ražošanas sistēmas imitācijas modeli. Galvenais imitācijas modeļa izstrādes mērķis ir parādīt izstrādātā V-DEVS formālisma praktiskā pielietojuma iespējas un eksperimentāli pārbaudīt realizēto algoritmu veiktspēju.

Apskatāmā ražošanas sistēmas modeļa apraksts un analīze tiek veikta, balstoties uz V-DEVS teoriju un koncepcijām, raksturojot modeļa arhitektūru, identificējot komponentus, to savstarpējo mijiedarbību un ietekmi, kā arī nepieciešamās specifikācijas tipu, līdzīgi kā 1.1. tabulā aprakstītajā 7.līmeņa globālā sistēma specifikācijā un 0.līmeņa strukturētā atsevišķa komponenta līmenī. Modeļa aprakstam un attēlojumam tiek izmantota notācija, kas ir līdzīga darbā [111] lietotajai notācijai, kas savukārt ir līdzīga datu plūsmu diagrammām. Izstrādātā industriālās automatizētās ražošanas sistēmas V-DEVS modeļa pamatā esošo metodiku iespējams pielietot industriālo sistēmu pētījumos gan atsevišķi, gan arī mijiedarbībā ar citiem automatizācijas elementiem rūpnieciskajās sistēmās.

4.1. Automatizētās ražošanas sistēmas modelis

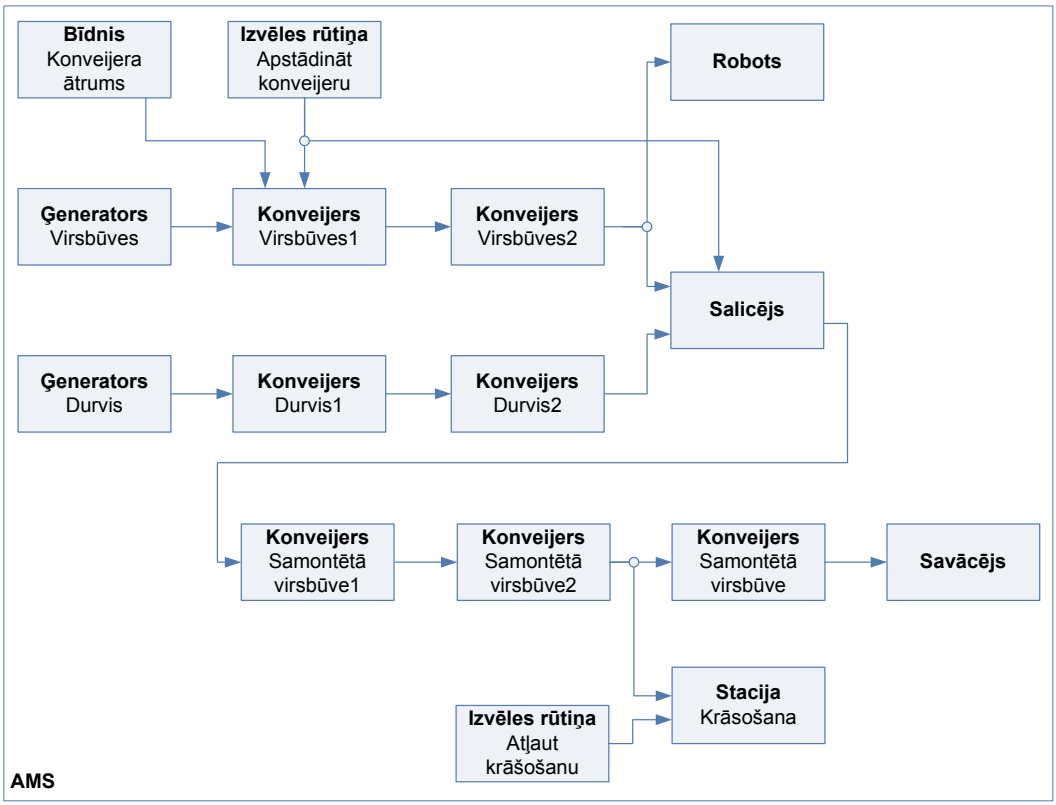
Lai ražošanas uzņēmumi spētu konkurēt globālā tirgus attīstības un konkurences apstākļos, tiem nepārtraukti ir jāattīsta un jāuzlabo savas ražošanas iespējas, ieviešot modernas *automatizētās ražošanas sistēmas* [16]. Darbā [49] ir minēti vairāki iemesli, kādēļ imitācijas modelēšanai ir arvien pieaugoša loma un nozīme ražošanas sistēmu jomā:

- automatizētās sistēmas ir tik sarežģītas, ka imitācijas modelēšana praktiski ir vienīgais pieejamais to analīzes līdzeklis;
- attīstoties datortehnoloģijām, samazinās nepieciešamie skaitļošanas resursi un izmaksas;
- imitācijas modelēšanas attīstības ietekmē iespējams samazināt modeļu izstrādes laiku;
- vizuālo līdzekļu pieejamība uzlabo imitācijas rezultātu izpratni un pielietojamību.

DEVS bāzētas pieejas pielietošanas pētījumi automatizēto ražošanas sistēmu ir realizēti arī iepriekš [17], taču V-DEVS formālismā sakņota imitācijas modelēšanas vide ietver gan diskrētus, gan nepārtrauktus procesus, kādi neapšaubāmi noris automatizētajās ražošanas sistēmās.

Automatizētajās ražošanas sistēmās izšķir trīs veida fiziskos ražošanas resursus: *apstrādes resursi* kā piemēram, darba galdi, montāžas centri, *transportēšanas resursi* kā piemēram, automātiski vadītie transporta līdzekļi, konveijeri un *glabāšanas resursi* kā piemēram, automātiskās glabāšanas sistēmas. Bieži vien imitācijas modelēšanas rīkus, tos pielietojot automatizētās ražošanas sistēmās, sauc par *rūpnīcu simulatoriem* [17]. Parasti rūpnīcu simulatori ir industriālu daudzfunkcionālu vadības informācijas sistēmu, sauktas par *ražošanas izpildes sistēmām* [34], sastāvā ietilpstoši moduļi, nevis atsevišķi pieejamas programmatūras sistēmas. Vieni no populārākajiem rūpnīcu simulatoriem ir *Siemens Simatics* un *ABB Robot Studio*, kuri nodrošina vizuālas interaktīvas 3D automatizēto ražošanas vadības sistēmu modelēšanas iespējas ar piesaisti reālajam ražošanas procesam. Taču šo sistēmu trūkums ir to šaurā specializācija un mazā pieejamība plašam lietotāju lokam savas augstās cenas dēļ, tādēļ tās ir piemērotas izmantošanai tikai augsti tehnoloģiskā liela apjoma ražošanas vidē.

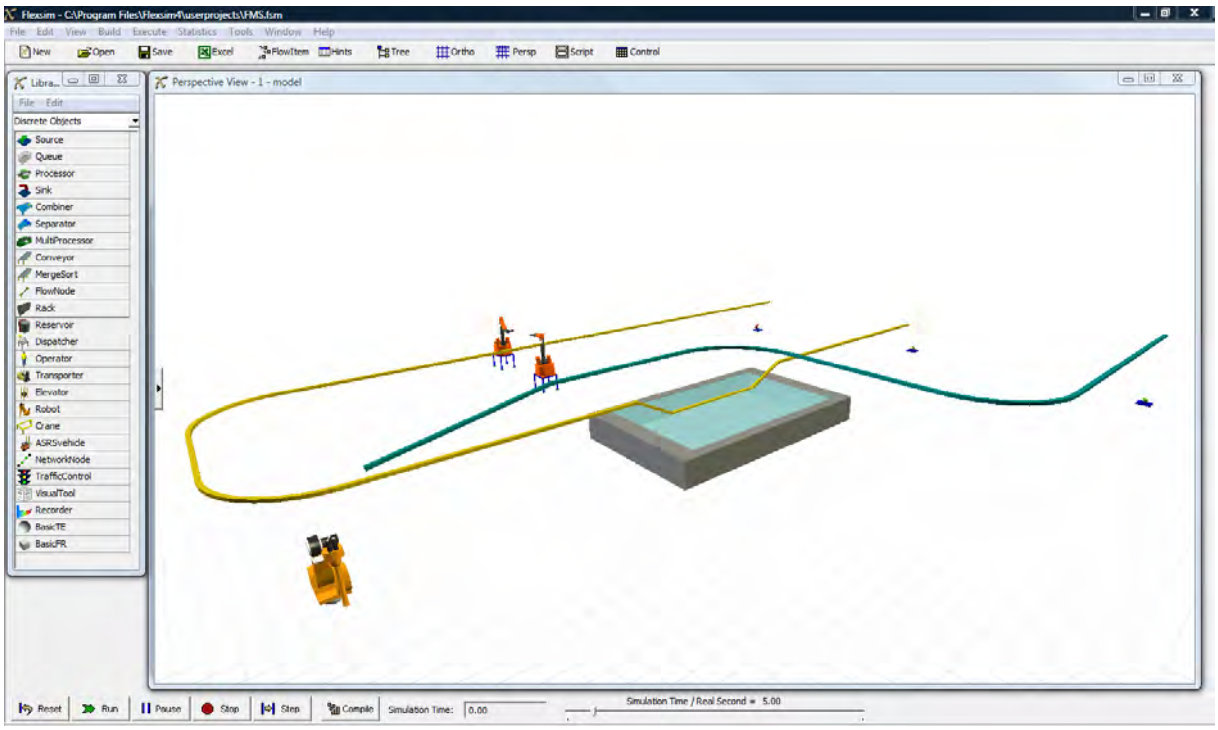
Apskatāmā automatizētās ražošanas sistēma sastāv no speciālām ražošanas materiālu plūsmu apstrādes stacijām un konveijeriem, kas nodrošina materiālu plūsmu transportēšanu starp stacijām. Modeļa pamatstruktūru (4.1. attēls) veido ģeneratori, imitācijas entītiju savācējs, interaktīvie elementi (bīdnis, izvēles rūtiņa), ražošanas produkcijas montāžas darbstacijas (krāsošanas stacija, montāžas robots, montāžas stacija jeb salicējs) un konveijeri produkcijas transportēšanai starp darbstacijām.



4.1. att. Automatizētās ražošanas sistēmas modelis

Ģeneratora modeļi ģenerē sistēmā ienākošo materiālu plūsmu - automašīnu virsbūves un durvis. Automašīnu virsbūves un durvis pa diviem dažādiem konveijeriem nonāk līdz salicēja modelim, kas modelē durvju montāžas procesu automašīnas virsbūvē. Tālāk pa citu konveijeru samontētā virsbūve virzās cauri krāsošanas stacijai, līdz nonāk savācējā. Savācēja modelis ir elements, kas veic saņemto entītiju - samontēto automašīnas virsbūvju iznīcināšanu, kas atbilst materiālu plūsmas iziešanai ārpus modelējamās sistēmas robežām.

Lai pārbaudītu un nodemonstrētu apskatāmās ražošanas sistēmas imitācijas modeļa realizācijas iespējas esošā komerciālā imitācijas sistēmā, kā arī, lai to salīdzinātu ar izstrādāto V-DEVS imitācijas modelēšanas vides prototipu, ir veikta imitācijas modeļa izstrāde un tā eksperimentāla pārbaude Flexsim vidē (4.2. attēls). 4.1. tabulā ir dots Flexsim imitācijas modelēšanas vides un V-DEVS prototipa galveno īpašību salīdzinājums ražošanas sistēmas modeļa izstrādē. Veiktie eksperimenti parāda sistēmas konceptuālā modeļa adekvātumu, taču Flexsim vidi nav tiešā veidā iespējams salīdzināt ar V-DEVS sistēmas prototipa veikto veiktspējas eksperimentu rezultātiem, jo Flexsim algoritmiskā realizācija un arhitektūra nav atvērta un pieejama izpētei.



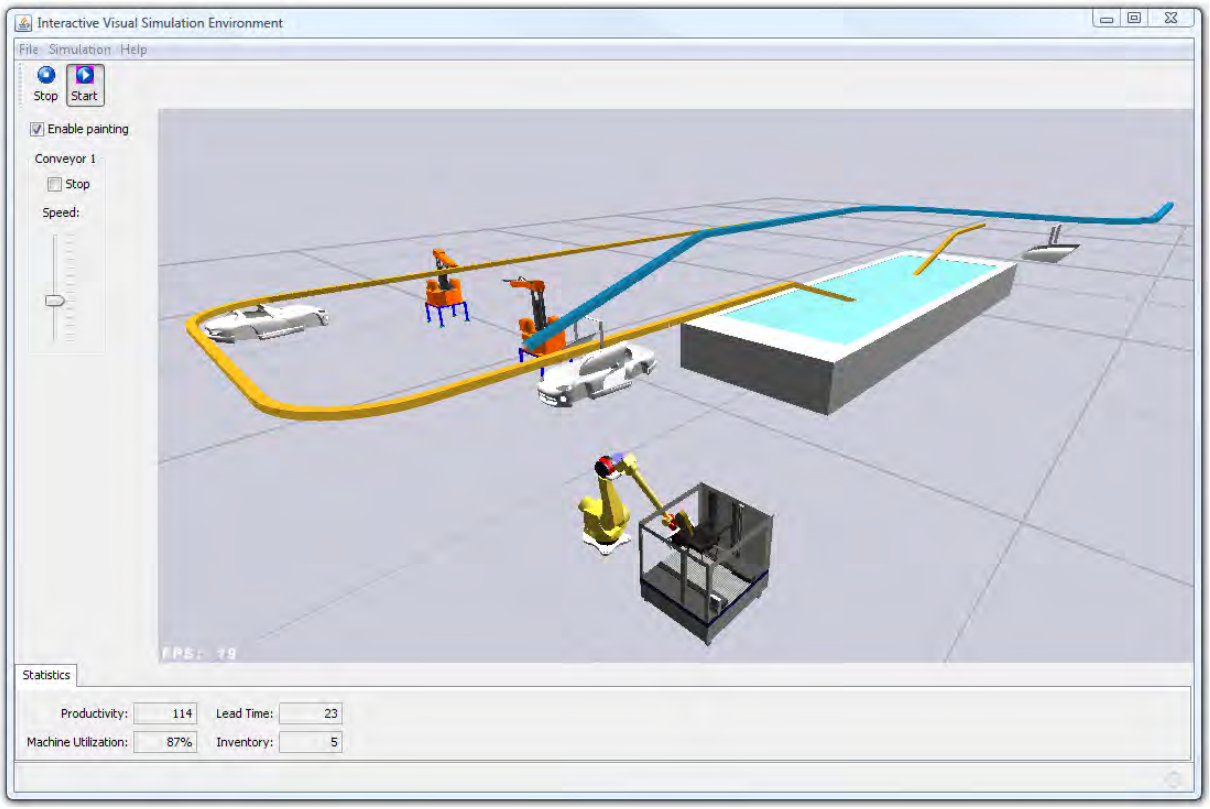
4.2. att. Automatizētās ražošanas sistēmas modelis Flexsim imitācijas modelēšanas vidē

4.3. attēlā parādīts imitācijas modeļa 3D attēls, kas izstrādāts ar V-DEVS imitācijas modelēšanas prototipu. No lietotāja viedokļa galvenās atšķirības un priekšrocības, salīdzinot ar tipiskām komerciālajām imitācijas modelēšanas sistēmām, ir kombinētas diskrētu notikumu un nepārtrauktu procesu imitācijas modelēšanas, kā arī nemodālas mijiedarbības iespējas ar modeli imitācijas procesa gaitā.

4.1. tabula

Flexsim imitācijas modelēšanas vides un V-DEVS prototipa galveno īpašību salīdzinājums ražošanas sistēmas modeļa izstrādē

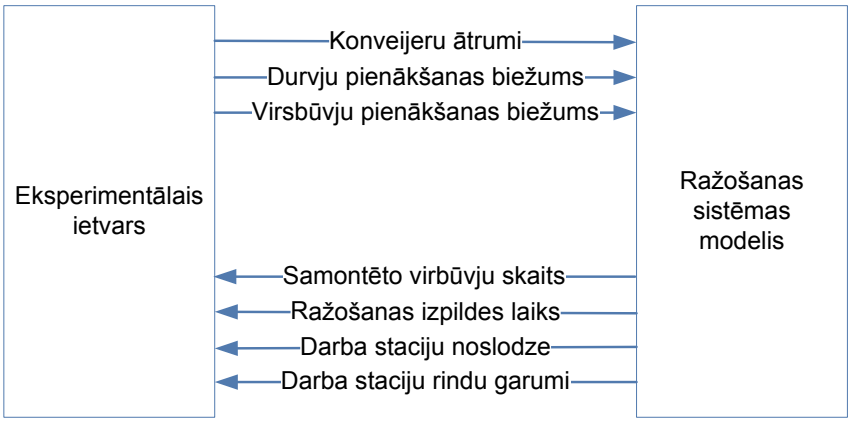
Īpašība	Flexsim	V-DEVS
Pielietošanas līmenis	Modelēšanas sistēma	Modelēšanas sistēma
Sistēmas tips	Kombinēta	Kombinēta
Pasaulskats	Uz dinamiskiem objektiem orientēts	Uz notikumiem orientēts
		Uz dinamiskiem objektiem orientēts
Modelēšanas jēdzieni	Modeļa bloki Orientācija uz objektiem	Modeļa bloki Orientācija uz objektiem
Specializācijas līmenis	Universāls	Universāls
Vizualizācija	3D	3D
Imitācijas interaktivitāte	Nav	Ir



4.3. att. Automatizētās ražošanas sistēmas modelis V-DEVS vizuālās imitācijas modelēšanas vidē

4.1.1. Eksperimentālais ietvars

Saskaņā ar 1.1.3. apakšnodaļā sniegto aprakstu, etalonsistēma ir reālās pasaules sistēma, bet eksperimentālais ietvars definē nosacījumus, pie kādiem un kādā veidā jāveic imitācijas eksperimenti. 4.4. attēlā ir parādīta automatizētās ražošanas sistēmas saistība ar modelējamo sistēmu.



4.4. att. Eksperimentālais ietvars

Eksperimentālā ietvara ģenerators modelē durvju un virsbūvju pienākšanu ražošanas sistēmā. Balstoties uz rindu teoriju, durvju un virsbūvju pienākšana izstrādājamajā imitācijas modelī ir traktējama kā gadījuma notikumu plūsma, kuru apraksta eksponenciālais sadalījuma likums [5, 49, 87] ar blīvuma funkciju

$$f(x) = \begin{cases} \lambda \cdot e^{-\lambda x}, & ja\, x \geq 0 \\ 0, & ja\, x < 0 \end{cases},$$

kur $\lambda > 0$ - eksponenciālā sadalījuma parametrs (vidējais pieprasījumu pienākšanas laika intervāls masu apkalpošanas sistēmās).

Eksperimentālajā ietvarā tiek uzdots arī modelējamās ražošanas sistēmas konveijeru darbības sākuma ātrums.

Ģeneratora atomārais modelis

Ģenerators ir diskrēts atomārs V-DEVS modelis bez ieejām, kas ģenerē modeļa ieejas notikumus, un tas ir eksperimentālā ietvara komponents, kuru apraksta 2.1.2. apakšnodaļā definētajai diskrētajai V-DEVS formai 2.3 ekvivalenta formāla struktūra:

$$\text{Ģenerators} = \langle X, Y, S, \delta_{ext}, \delta_{int}^{discr}, \lambda^{discr}, ta^{discr} \rangle$$

Modeļa ieejas: $X = \varnothing$.

Stāvokļu mainīgie: $S = \{\text{"pasīvs"}, \text{"aktīvs"}\}$.

Modeļa izejas: $Y^{discr} = \{(\text{"izeja"}, E)\}$.

$\delta_{ext}(s, e, x) = s$;

$\delta_{int}^{discr}(s) = \text{"aktīvs"}$;

$\lambda^{discr}(\text{"aktīvs"}) = (\text{"izeja"},\ \text{jaunizveidota entītija}\, E)$;

$ta^{discr}(\text{"aktīvs"}) = \text{notikumu ģenerēšanas periods}\, T_G$.

Tā kā ģeneratora modelim nav ieeju, tad saskaņā ar [119, 111] tā ir autonoma sistēma, kuras darbības dinamiku nosaka tikai iekšējās pārejas funkcija. Ģeneratora izejas portā izejas funkcijas λ^{discr} ģenerētie notikumi satur jaunizveidotos imitācijas entītiju E objektus, kuri imitācijas procesā virzās cauri modeļa saišu struktūras definētajiem elementiem, līdz tie nonāk imitācijas entītiju savācējā. Izejas notikumi tiek ģenerēti ar uzdotu notikumu ģenerēšanas periodu T_G, kas var būt konstants lielums vai arī pēc noteikta sadalījuma likuma (piemēram, eksponenciālā sadalījuma) ģenerēts gadījuma lielums.

Imitācijas entītiju savācēja atomārais modelis

Imitācijas entītiju savācējs ir atomārs modelis, kas darbojas eksperimentālā ietvara kontekstā, un ir paredzēts imitācijas procesu pabeigušo entītiju savākšanai, iznīcināšanai un statistikas uzkrāšanai. Imitācijas entītiju savācēju formāli apraksta sekojoša struktūra:

$$\text{Savācējs} = \left\langle X, Y, S, \delta_{ext}, \delta_{int}^{discr}, \lambda^{discr}, ta^{discr} \right\rangle$$

Modeļa ieejas: $X = \{(\text{"ieeja"}, E)\}$.

Stāvokļu mainīgie: $S = \{\text{"pasīvs"},\ \text{"aktīvs"}\}$.

Modeļa izejas: $Y = \varnothing$.

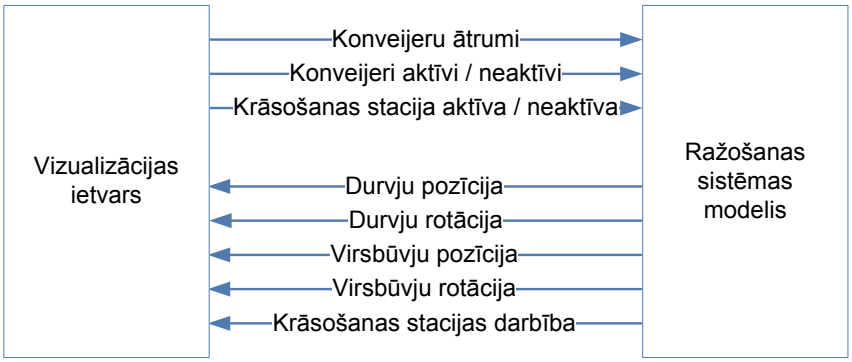
$\delta_{ext}(\text{"aktīvs"}, e, \text{"ieeja"}) = (\text{"aktīvs"}, \text{iznīcināt entītiju}\, E)$;

$\delta_{int}^{discr}(s) = s$;

$ta^{discr}(S) = +\infty$.

4.1.2. Vizualizācijas ietvars

Vizualizācijas ietvars definē automatizētās ražošanas sistēmas vizuālo un interaktīvo parametru saikni ar modelējamo sistēmu (4.5. attēls). Imitācijas modeļa lietotājam ir iespējams interaktīvi imitācijas laikā mainīt tādus modeļa parametrus kā konveijeru ātrumi, konveijeru darba apstādināšana un turpināšana, kā arī krāsošanas stacijas darba apstādināšana un turpināšana.



4.5. att. Ražošanas sistēmas modeļa vizualizācijas ietvars

Grafiskās lietotāja saskarnes atomārie modeļi

Tāpat kā modelējamās sistēmas loģiskās struktūras definēšanai un vizualizācijas vajadzībām tiek izmantoti V-DEVS modeļi, arī grafiskās lietotāja saskarnes mijiedarbība tiek apstrādāta ar speciālu V-DEVS modeļu palīdzību. Dotā automatizētās ražošanas sistēmas imitācijas modeļa izstrādē tiek izmantoti trīs veidu grafiskās lietotāja saskarnes atomārie modeļi: bīdnis, izvēles rūtiņa un rediģēšanas logs. Visi trīs šo modeļu tipi ir atvasināti no abstraktas bāzes grafiskā lietotāja saskarnes atomārā modeļa klases Saskarnes modelis (sk. 3.2.2. apakšnodaļu).

Bīdnis

Atomārais modelis Bīdnis ietver lietotāja saskarnes bīdņa elementu, reaģējot uz bīdņa pozīcijas maiņu ar atbilstošu notikumu modeļa izejā.

$$\text{Bīdnis} = \left\langle X, Y, S, \delta_{ext}, \delta_{int}^{discr}, \lambda^{discr}, ta^{discr} \right\rangle$$

Modeļa izejas: $Y = \{(\text{"izeja"},\ \text{bīdņa pozīcija})\}$,
kur bīdņa pozīcija $\in \mathbb{R}$.

Izvēles rūtiņa

Atomārais modelis Izvēles rūtiņa ietver lietotāja saskarnes izvēles rūtiņas elementu, reaģējot uz bīdņa pozīcijas maiņu ar atbilstošu notikumu modeļa izejā.

$$\text{Izvēles rūtiņa} = \left\langle X, Y, S, \delta_{ext}, \delta_{int}^{discr}, \lambda^{discr}, ta^{discr} \right\rangle$$

Modeļa izejas: $Y = \{(\text{"izeja"}, \text{izvēles ķeksītis})\}$,
kur izvēles ķeksītis $\in (true, false)$.

Rediģēšanas logs

Atomārais modelis Rediģēšanas logs ietver lietotāja saskarnes izvēles rūtiņas elementu, reaģējot uz bīdņa pozīcijas maiņu ar atbilstošu notikumu modeļa izejā.

$$\text{Rediģēšanas logs} = \left\langle X, Y, S, \delta_{ext}, \delta_{int}^{discr}, \lambda^{discr}, ta^{discr} \right\rangle$$

Modeļa izejas: $Y = \{(\text{"izeja"},\ \text{ievadītais teksts})\}$,
kur ievadītais teksts $\in \mathbb{R}$.

4.1.3. Sistēmas modeļa realizācija

Automatizētās ražošanas sistēmas modelis tiek attēlots kā saistītais V-DEVS modelis, kas sastāv no vairākiem savstarpēji saistītiem apakšmodeļiem (4.1. attēls). Modelējamajā sistēmā darbojas trīs konveijeru līnijas, kas paredzētas durvju, virsbūvju un samontēto virsbūvju transportēšanai. Katru konveijera līniju veido divi secīgi saistīti konveijeri, tādējādi dotajā modelī bāzes variantā tiek izmantoti 6 konveijeru modeļi. Katrs konveijers sastāv no taisnām vai liektām sekcijām, un to raksturo noteikts kopējais garums (4.2. tabula).

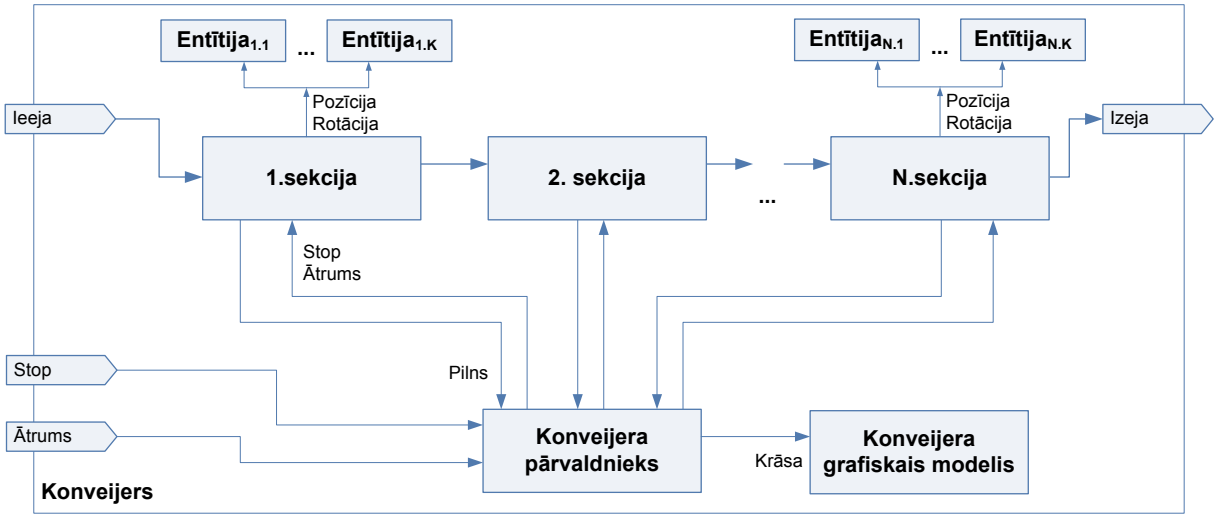
4.2. tabula

Konveijeru parametri

Konveijers	Taisno sekciju skaits	Liekto sekciju skaits	Garums (m)
Virsbūves 1	5	0	66,5
Virsbūves 2	1	0	10
Durvis 1	2	2	55,71
Durvis 2	2	0	18,8
Saliktās virsbūves 1	3	2	45,71
Saliktās virsbūves 2	1	0	50

Konveijera saistītais modelis

Konveijera saistītais modelis satur konveijera pārvaldnieka un vienu vai vairākus konveijera sekciju apakšmodeļus (4.6. attēls).



4.6. att. Konveijera saistītā modeļa struktūra

Konveijera saistītais modelis formāli ir aprakstāms sekojošas struktūras veidā:

$$\text{Konveijers} = \langle X_K, Y_K, M_K, EIC_K, EOC_K, IC_K \rangle$$

Ieejas notikumu mainīgie: $X_K = \{\text{Ieeja, Ātrums, Stop}\}$, kur

Ieeja = (”ieeja”, E);

Ātrums = (”ātrums”, v) | $v \in \mathbb{R}^+$;

Stop = (”stop”, $\{true\}$).

Izejas notikumu mainīgie: $Y_K = \{(\text{”izeja”}, E)\}$.

Modeļa komponenti:

$M_K = \{1.\text{sekcija}, 2.\text{sekcija}, \ldots, \text{N.sekcija}, \text{Konveijera pārvaldnieks},$
Konveijera grafiskais modelis$\}$.

Ārējo ieeju saites:

$EIC_K = \{(\text{Ieeja}, 1.\text{sekcija.Ieeja}), (\text{Stop}, \text{Konveijera pārvaldnieks.Stop}),$
$(\text{Ātrums}, \text{Konveijera pārvaldnieks.Ātrums})\}$.

Ārējo izeju saites: $EOC_K = \{(\text{N.sekcija.Izeja}, \text{Izeja})\}$.

Iekšējās saites:

$IC_K = \{(1.\text{sekcija.Izeja}, 2.\text{sekcija.Ieeja}), \ldots, (\text{N-1.sekcija.Izeja}, \text{N.sekcija.Ieeja}),$
$(1.\text{sekcija.Pilns}, \text{Konveijera pārvaldnieks.Pilns\_1}), \ldots,$
$(\text{N.sekcija.Pilns}, \text{Konveijera pārvaldnieks.Pilns\_N}),$
$(\text{Konveijera pārvaldnieks.Stop}, 1.\text{sekcija.Stop}), \ldots,$
$(\text{Konveijera pārvaldnieks.Stop}, \text{N.sekcija.Stop}),$
$(\text{Konveijera pārvaldnieks.Ātrums}, 1.\text{sekcija.Ātrums}), \ldots,$
$(\text{Konveijera pārvaldnieks.Ātrums}, \text{N.sekcija.Ātrums}),$
$(\text{Konveijera pārvaldnieks.Krāsa}, \text{Konveijera grafiskais modelis.Krāsa})\}$.

Konveijera pārvaldnieka modelis

Konveijera pārvaldnieka atomārais modelis ir centrālais konveijera vadības elements, kas veic visu konveijera sekciju darbības koordinēšanu.

$$\text{Konveijera pārvaldnieks} = \langle X, Y, S, \delta_{ext}, \delta_{int}^{discr}, \lambda^{discr}, ta^{discr} \rangle$$

Ieejas porti: $X = \{\text{Darbība}, \bar{\text{A}}\text{trums}, \text{Pilns}_1, \ldots, \text{Pilns}_N\}$, kur

Darbība $= (\text{"darbība"}, b) \mid b = \{true, false\}$;

$\bar{\text{A}}$trums $= (\text{"ātrums"}, v) \mid v \in \mathbb{R}^+$;

$\text{Pilns}_i = (\text{"pilns\_"} + i, \{true\}) \mid i = 1, \ldots, N$;

N - konveijera sekciju skaits.

Stāvokļu mainīgie: $S^{discr} = \{\text{fāze, ātrums, pēdējā sekcija}\}$, kur

fāze $= (\text{"pasīvs"}, \text{"apstādināts"}, \text{"palaists"})$;

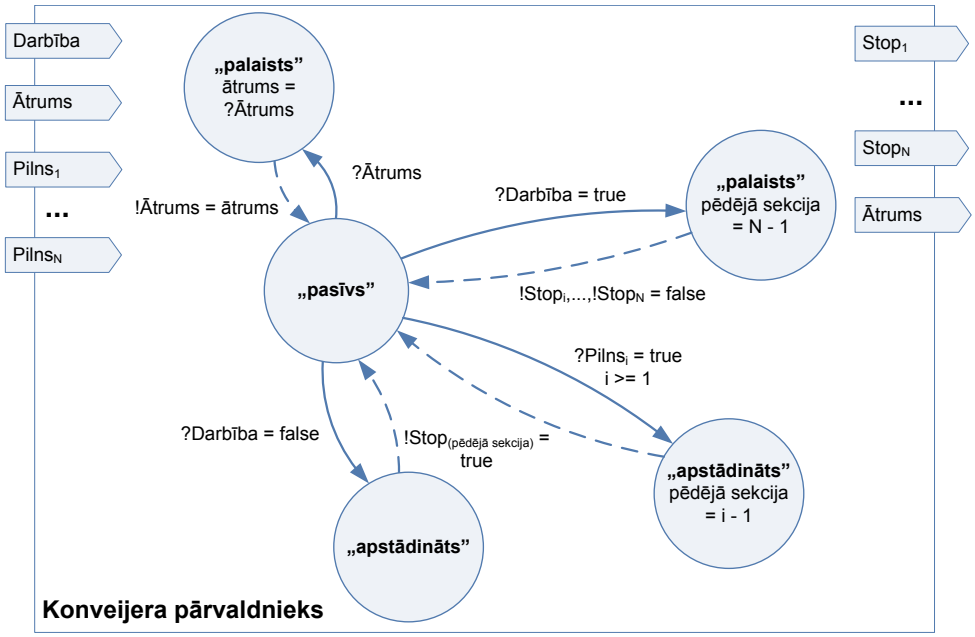
ātrums $\in \mathbb{R}^+$;

pēdējā sekcija $= 0, \ldots, N-1$.

Izejas porti: $Y = (\text{Stop}_1, \ldots, \text{Stop}_N)$, kur

$\text{Stop}_i = (\text{"stop\_"} + i, b) \mid b = \{true, false\}; i = 1, \ldots, N$.

4.7. attēlā ir parādīta konveijera pārvaldnieka modeļa stāvokļu pāreju diagramma.



4.7. att. Konveijera pārvaldnieka modeļa stāvokļu diagramma

Konveijera sekcijas modelis

Konveijera sekcijas modelis realizē atsevišķas konveijera sekcijas darbības imitāciju, ietverot gan diskrētu notikumu, gan nepārtraukto apstrādi. Sekcijas modelis apraksta šāda V-DEVS

balstīta struktūra:

$$\text{Konveijera sekcija} = \left\langle X, Y, S, \delta_{ext}, \delta_{int}^{discr}, \lambda^{discr}, ta^{discr}, \lambda^{cont}, ta^{cont} \right\rangle$$

Ieejas notikumu mainīgie: $X = \{\text{Ieeja, Stop, }\}$, kur

Ieeja = (”ieeja”, Entītija);

Stop = (”stop”, b) | $b = \{true, false\}$;

Ātrums = (”ātrums”, v)| $v \in \mathbb{R}^+$.

Stāvokļu mainīgie: $S = \{\text{fāze, rinda, elementi}\}$, kur

fāze = {”pasīvs”, ”aktīvs”, ”pilns”};

rinda = $(ED_1, \ldots, ED_M)$, kur

ED - entītijas dati;

M - tekošais konveijera sekcijas rindas garums;

elementi = $(ED_1, ..., ED_N)$, kur

N - tekošais transportēšanas procesā ešošo elementu skaits.

Izejas notikumu mainīgie: $Y^{discr} = \{\text{”Izeja”, ”Pilns”}\}$, kur

Izeja = (”izeja”, Entītija);

Pilns = (”pilns”, $\{true\}$);

$Y^{cont} = \{\text{Pozīcija, Rotācija}\}$, kur

Pozīcija = (”pozīcija”, p) | $p \in \mathbb{R}^3$;

Rotācija = (”rotācija”, r) | $r \in \mathbb{R}^3$.

Imitācijas entītiju apstrādei konveijera sekcijā to transportēšanas, gan rindā atrašanās laikā tiek izmantota palīgstruktūra - entītijas dati ED, kura formāli aprakstāma šādā veidā:

$$ED = \left\langle E, s_{cur}, s_{prev}, t_{enter} \right\rangle,$$

kur E - imitācija entītija;

s_{cur} - entītijas pašreizējais pārvietojums sekcijā;

s_{prev}- entītijas pārvietojums iepriekšējā laika momentā;

t_{enter} - entītijas ienākšanas sākuma laiks konveijera sekcijā.

Darba stacijas modelis

Darba stacijas atomārais modelis realizē materiālu resursu apstrādes procesu (piemēram, automašīnu virsbūvju krāšošana) imitāciju.

$$\text{Darbstacija} = \left\langle X, Y, S, \delta_{ext}, \delta_{int}^{discr}, \lambda^{discr}, ta^{discr}, \lambda^{cont}, ta^{cont} \right\rangle$$

Ieejas porti: $X = \{\text{Ieeja}, \text{Darbības atļauja}\}$, kur

Ieeja = (”ieeja”, E) , kur E - imitācijas entītija;

Darbības atļauja = $\{true, false\}$;

Stāvokļu mainīgie: $S = \{\text{fāze}, \text{atļauts}\}$, kur

fāze = (”pasīvs”, ”darbība”);

atļauts = $\{true, false\}$.

Izejas porti: $Y = \{(\text{”izeja”}, E)\}$;

$\delta_{ext}(s, e, \text{Darbības atļauja}) = (\text{atļauts} = x.v)$;

δ_{int}^{discr} (”darbība”) = ”pasīvs”;

λ^{discr} (”darbība”) = (”izeja”, E);

ta^{discr} (”darbība”) = apstrādes laiks.

Salicēja modelis

Salicēja atomārais modelis realizē materiālu resursu vienību komplektēšanas procesu, vienības no divām ieejām apvienojot vienā izejā. Tas nozīmē, ka entītiju dati no divām ieejas entītijām tiek apvienoti un novirzīti modeļa izejā.

$$\text{Salicējs} = \left\langle X, Y, S, \delta_{ext}, \delta_{int}^{discr}, \lambda^{discr}, ta^{discr}, \lambda^{cont}, ta^{cont} \right\rangle$$

Ieejas porti: $X = \{\text{1.ieeja}, \text{2.ieeja}, \text{Darbības atļauja}\}$, kur

1.ieeja = (”1.ieeja”, E) , kur E - imitācijas entītija;

2.ieeja = (”2.ieeja”, E);

Darbības atļauja = $\{true, false\}$.

Izejas ports: $Y = \{\text{”izeja”}, E\}$.

4.1.4. Robota modelis

Automatizētās ražošanas sistēmas imitācijas modelī tiek izmantots sešu brīvības pakāpju robota modelis. Šāda veida roboti, piemēram, industriālais robots PUMA 560, tiek plaši izmantoti arī reālās automatizētajās industriālajās sistēmās. Apskatāmais industriālais robots sastāv no secīgu segmentu kopas, kas ir relatīvi saistīti viens ar otru ar rotējošu vai bīdes savienojumu palīdzību, tādējādi tas ir pieskaitāms pie industriālo manipulatoru klases [83]. Tā kā robots sastāv no vairākiem savstarpēji saistītiem elementiem - segmentiem un to savienojumiem, tad, lai būtu iespējams definēt robota darbību, robota elementu koordinātes ir nepieciešams izteikt pasaules koordinātu sistēmās un savienojumu koordinātu sistēmās. Šim nolūkam ērts līdzeklis ir vispārzināmais Denavita-Hartenberga (DH) koordinātu transformāciju attēlojums [82, 104], kura

princips ir tāds, ka robota savienojumu koordinātes tiek veidotas tādā veidā, lai starp jebkuriem diviem robota segmentiem būtu spēkā divi ierobežojumi:

1. ass x_i ir perpendikulāra asij z_{i-1};
2. ass x_i krusto asi z_{i-1}.

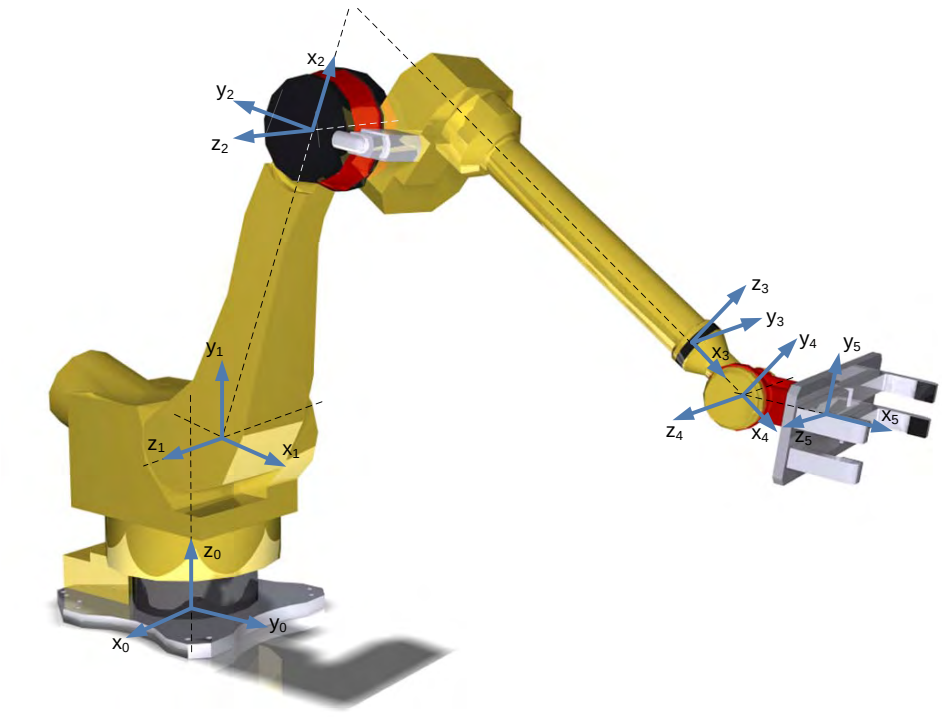
Šādu ierobežojumu ieviešana ļauj vienkāršot koordinātu sistēmu transformācijas, jo ir iespējams pāriet no sešām brīvības pakāpēm uz četrām.

4.3. tabula

DH parametri (aizgūts no [82])

a_i	Attālums pa x_i asi no x_i un z_{i-1}krustpunkta līdz o_i
α_i	Leņķis starp z_{i-1} un z_i asīm; rotācija ap x_i asi
d_i	Attālums pa z_{i-1} asi no o_{i-1} līdz x_i un z_{i-1}krustpunktam
θ_i	Leņķis starp x_{i-1} un x_i asīm; rotācija ap z_{i-1} asi

Dotajā imitācijas sistēmā robota modelis sastāv no 5 segmentiem, robota beigu efektoru (satvērējmehānismu) neiekļaujot robota brīvības pakāpju noteikšanā. Katrs segments tiek aprakstīts ar četriem DH parametriem, kas definē katra segmenta savienojuma pārvietojumu vai rotāciju (4.4. tabula) attiecībā pret iepriekšējo segmentu. 4.8. attēlā ir parādīta robota modeļa kinemātiskā ķēde.



4.8. att. Robota modeļa kinemātiskā ķēde

Kā redzams 4.4. tabulā, dotā robota modeļa kinemātiku nosaka seši mainīgie θ_1, θ_2, θ_3, θ_5, α_4.

4.4. tabula

Robota modeļa DH parametru tabula (attālumi a_i un d_i doti metros)

Segments	a_i	d_i	α_i	θ_i
1	0,1	1,25	90	θ_1
2	1,75	0	0	θ_2
3	1,3	0	0	θ_3
4	0,4	0,3	α_4	0
5	0,4	0	0	θ_5

Robota bāzes segmenta transformācija attiecībā pret robota roku ir izsakāma sekojoši:

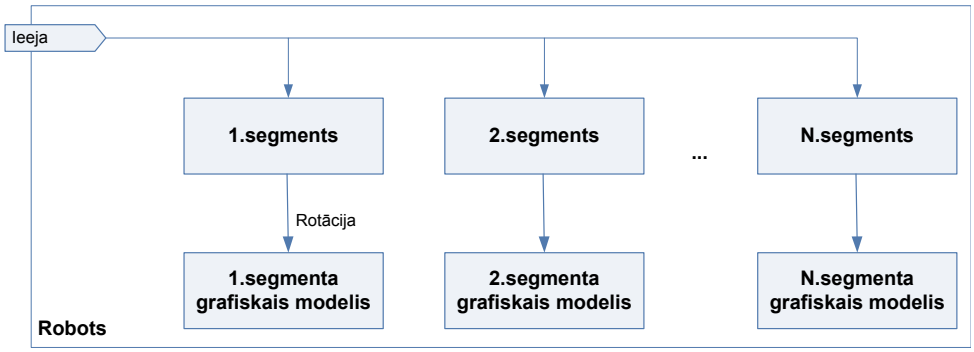
$$T(\theta_1, \theta_2, \theta_3, \alpha_4, \theta_5) = A_1 \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5, \tag{4.1}$$

kur A_i - segmenta i transformācijas matrica [82]:

$$A_i = \begin{pmatrix} cos\theta_i & -sin\theta_i cos\alpha_i & sin\theta_i sin\alpha_i & a_i cos\theta_i \\ sin\theta_i & cos\theta_i cos\alpha_i & cos\theta_i sin\alpha_i & a_i sin\theta_i \\ 0 & sin\alpha_i & cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Robota saistītais modelis

Robota saistītais modelis satur vienu vai vairākus segmentu un to grafiskos apakšmodeļus (4.9. attēls).



4.9. att. Robota saistītā modeļa struktūra

Robota saistītais modelis formāli ir aprakstāms šādas struktūras veidā:

$$\text{Robots} = \langle X_R, Y_R, M_R, EIC_R, EOC_R, IC_R \rangle$$

Ieejas notikumu mainīgie: X_R=$\{$Ieeja$\}$.

Izejas notikumu mainīgie: $Y_R = \varnothing$.

Modeļa komponenti:

$M_R = \{1.\text{segments}, \ldots, \text{N.segments}, 1.\text{segmenta grafiskais modelis}, \ldots,$
N.segmenta grafiskais modelis$\}$.

Ārējo ieeju saites: $EIC_R = \{(\text{Ieeja}, 1.\text{segments.Ieeja}), \ldots, (\text{Ieeja}, \text{N.segments.Ieeja})\}$.

Ārējo izeju saites: $EOC_R = \varnothing$.

Iekšējās saites:

$IC_R = \{(1.\text{segments.Rotācija}, 1.\text{segmenta grafiskais modelis.Rotācija}), \ldots,$
$(\text{N.segments.Rotācija}, \text{N.segmenta grafiskais modelis.Rotācija})\}$.

Robota segmenta modelis

Robota segmenta atomārais modelis realizē atsevišķa robota segmenta un tā savienojuma kustības imitāciju. Doto modeli formāli apraksta šāda struktūra:

$$\text{Segments} = \left\langle X, Y, S, \delta_{ext}, \delta_{int}^{discr}, \lambda^{discr}, ta^{discr}, \delta_{int}^{cont}, \lambda^{cont}, ta^{cont} \right\rangle$$

Ieejas porti: $X = \{$Ieeja$\}$.

Stāvokļu mainīgie: $S = \{$fāze, turpvēstā kustība, pauze, atpakaļvērstā kustība$\}$, kur

fāze = (”pasīvs”,”turpvērstā kustība”,”pauze”,”atpakaļvērstā kustība”);

ātrums $\in \mathbb{R}^+$;

pēdējā sekcija $= 0, \ldots, N-1$.

Izejas porti: $Y = \varnothing$.

4.2. Eksperimenti ar ražošanas sistēmas modeli

Galvenais imitācijas eksperimentu veikšanas mērķis ar doto automatizētās ražošamas sistēmas modeli ir realizēto algoritmu verifikācija, to veiktspējas analīze, kā arī lai nodemonstrētu, ka izstrādātais V-DEVS formālisms un uz tā bāzes realizētais imitācijas modelēšanas vides prototips ir pielietojams praktiskai sistēmu imitācijas modelēšanai. Eksperimentu izpildei ar ražošanas sistēmas modeli ir izvēlēts imitācijas laika mērogs 8:1, kas nozīmē, ka imitācijas modelis tiek darbināts 8 reizes ātrāk, nekā reālā laika sistēma.

4.2.1. Simulatora algoritmu veiktspējas analīze

Tā kā V-DEVS formālisma praktiskās realizācijas nolūkiem ir izstrādāti divi dažādi simulatora algoritmi (3.1.1. apakšnodaļa), tad ir nepieciešams šos algoritmus eksperimentāli pārbaudīt un veikt to veiktspējas salīdzinošu analīzi, lai novērtētu to praktiskās pielietošanas iespējas sarežģītu kombinētu sistēmu imitācijas modelēšanā. Dotā uzdevuma izpildei ir veikti 20 imitācijas eksperimenti, katru eksperimentu atkārtojot 10 reizes. Katra eksperimenta imitācijas laiks ir 1 minūte, kas reālajā laika mērogā atbilst 8 minūšu ražošanas sistēmas darbībai. No klasiskās imitācijas modelēšanas viedokļa tas ir pārāk mazs laika periods adekvātu imitācijas statistisko rezultātu iegūšanai, taču simulatora algoritma veikstpējas novērtēšanai izvēlētais imitācijas ilgums ir pietiekošs.

Veicot veiktspējas mērījumus, viens no uzdevumiem ir noteikt tās galējās robežas, tādēļ imitācijas eksperimenti jāveic ar pēc iespējas lielāku simulatora noslodzi. Šim nolūkam ir veikti eksperimenti ar dažādu konveijeru un virsbūvju ģeneratoru skaitu pie konstanta konveijeru ātruma 0,625 m/s.

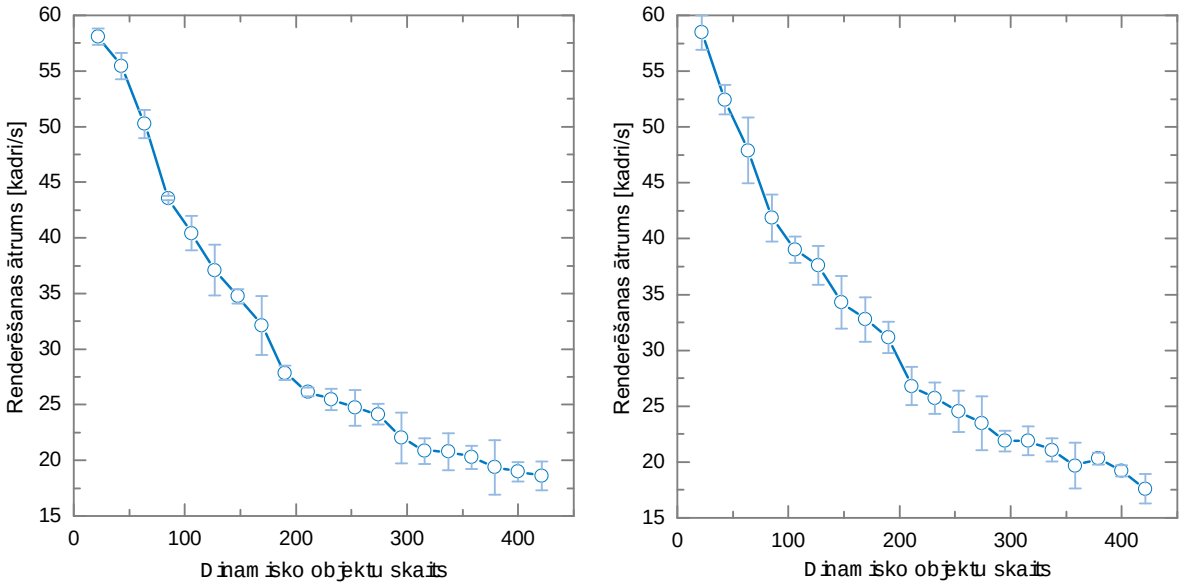
Imitācijas process ir integrēts ar reālā laika vizualizāciju, tādēļ rodas jautājums, kāda ir vizualizācijas ietekme uz imitācijas procesa veiktspēju. Pirmais eksperimentu cikls ir veikts atbildes gūšanai tieši uz šo jautājumu, šim nolūkam mainot konveijeru un ģeneratoru skaitu intervālā $[6;25]$. 4.5. tabulā ir parādīta ģenerēto imitācijas entītiju skaita atkarība no virsbūvju ģeneratoru un konveijeru skaita modelī.

4.5. tabula

Ģenerēto imitācijas entītiju skaita atkarība no konveijeru skaita sistēmā

Konveijeru skaits	6	7	8	9	10	11	12	13	14	15
Imitācijas entītiju skaits	22	43	64	85	106	127	148	169	190	211
Konveijeru skaits	16	17	18	19	20	21	22	23	24	25
Imitācijas entītiju skaits	232	253	274	295	316	337	358	379	400	421

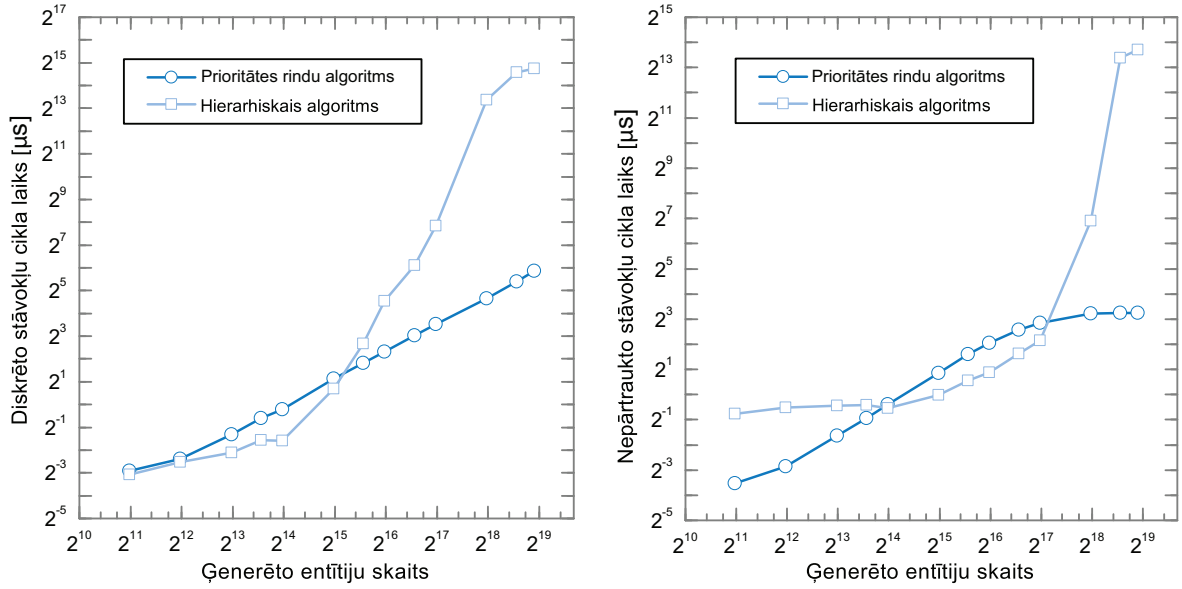
4.10. attēlā ir parādīta renderēšanas ātruma atkarība no ģenerēto imitācijas entītiju jeb dinamisko objektu skaita modelī, pielietojot hierahisko un prioritātes rindu algoritmus. Palielinot dinamisko objektu skaitu modelī, gan hierarhiskā, gan prioritātes rindu V-DEVS simulatora balstītā renderēšanas veiktspēja samazinās.



(a) Hierarhiskais algoritms (b) Prioritātes rindu plānošanas algoritms

4.10. att. Imitācijas modeļa renderēšanas ātruma atkarība no dinamisko objektu skaita modelī

4.11. attēlā ir parādīta diskrēto S^{discr} un nepārtraukto stāvokļu S^{cont} apstrādes cikla laika atkarība no ģenerēto imitācijas entītiju skaita modelī. Palielinot ģenerēto entītiju skaitu modelī, hierarhiskā V-DEVS simulatora algoritma veiktspēja samazinās straujāk, nekā prioritātes rindu algoritma izmantošanas gadījumā.

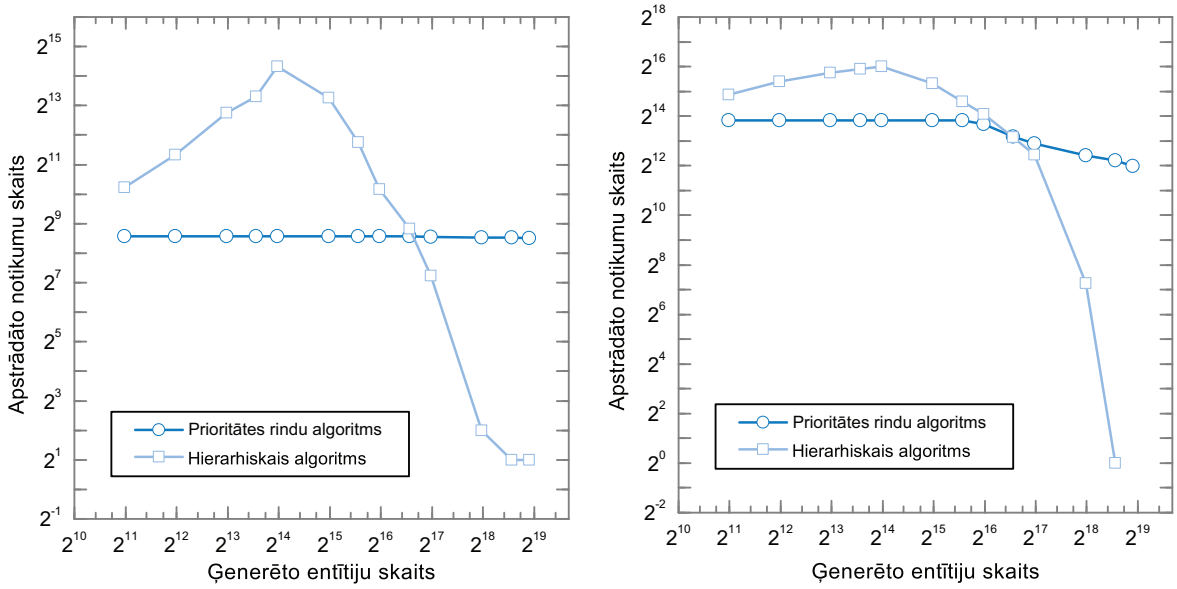


(a) Diskrēto notikumu imitācija (b) Nepārtraukto notikumu imitācija

4.11. att. Diskrēto S^{discr} un nepārtraukto S^{cont} stāvokļu apstrādes cikla laika atkarība no ģenerēto imitācijas entītiju skaita modelī

4.12. attēlā ir parādīta diskrēto un nepārtraukto notikumu skaita atkarība no ģenerēto

imitācijas entītiju skaita modelī.



(a) Diskrēto notikumu imitācija

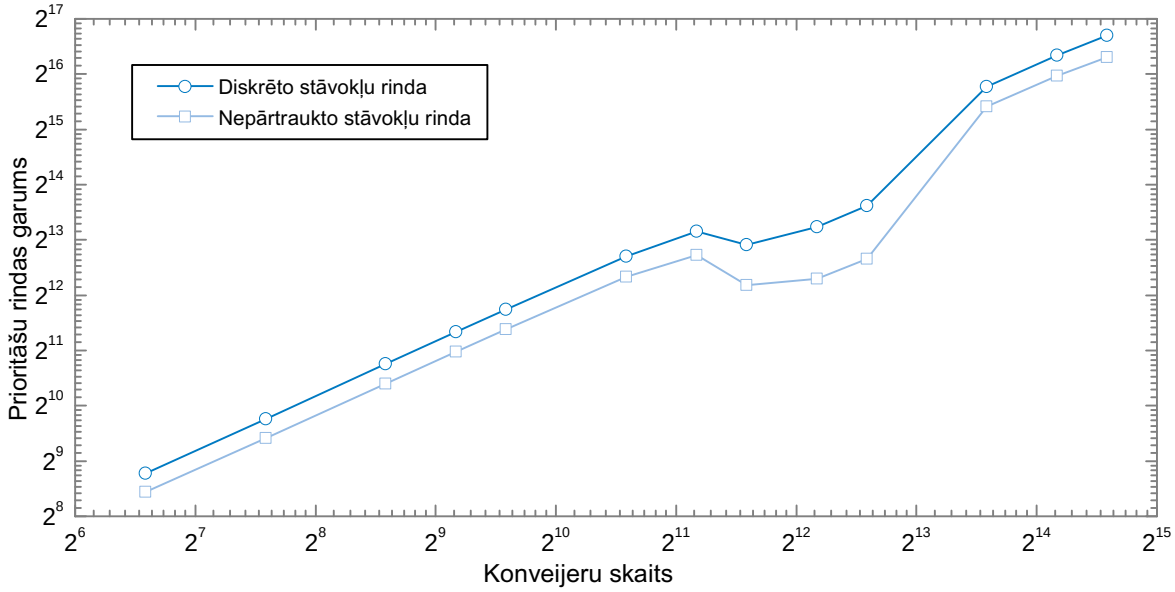
(b) Nepārtraukto notikumu imitācija

4.12. att. Diskrēto un nepārtraukto notikumu skaita atkarība no ģenerēto imitācijas entītiju skaita modelī

Hierarhiskā simulatora apstrādāto diskrēto un nepārtraukto stāvokļu notikumu skaits ievērojami palielinās, pieaugot ģenerēto entītiju skaitam, līdz tiek sasniegts piesātinājuma punkts, kad imitācijas izpildes laika sistēma vairs nespēj apstrādāt visu ģenerētos notikumus. 4.12. attēlā šāds hierarhiskā simulatora diskrēto un nepārtraukto stāvokļu notikumu apstrādes piesātinājuma punkts ir pie 2^{14} ģenerēto entītiju skaita, pēc kura sāk samazināties apstrādāto notikumu skaits. Un tas nozīmē, ka šādi iegūtie imitācijas rezultāti vairs nav ticami, jo sistēma nespēj apstrādāt visus notikumus.

Savukārt prioritātes rindu algoritma diskrēto notikumu apstrādes veiktspēja veikto eksperimentu ietvaros ir praktiski neatkarīga no ģenerēto entītiju skaita. Bet nepārtraukto stāvokļu apstrādes veiktspēja prioritātes rindu algoritmam sāk samazināties pie 2^{16} ģenerēto entītiju skaita. Tas nozīmē, ka nepārtrauktu procesu apstrādē prioritātes rindu algoritmam ir $2^{16}/2^{14} = 4$ reizes lielāka veiktspēja nekā hierarhiskajam simulatora algoritmam.

4.13. attēlā ir redzama plānošanas algoritma diskrētu un nepārtrauktu stāvokļu prioritātes rindu garuma atkarība no ģenerēto imitācijas entītiju skaita modelī.



4.13. att. Plānošanas algoritma prioritātes rindu vidējā garuma atkarība no ģenerēto imitācijas entītiju skaita modelī

Palielinot konveijeru un ģeneratoru skaitu un tādējādi pieaugot ģenerēto entītiju skaitam, palielinās arī apstrādājamo notikumu skaits un prioritātes rindu garums.

4.2.2. Modeļa izejas datu regresijas analīze

Ar imitācijas modeli ir veikti 19 eksperimenti, mainot konveijeru ātrumu robežās 0,25-2,5 m/s. Katrs eksperiments ar 1 minūtes imitācijas ilgumu ir atkārtots 10 reizes, tādējādi kopējais eksperimentu skaits ir $19 \times 10 = 190$. Veiktspējas analīzes veikšanai tiek izmantots konstants durvju un virsbūvju pienākšanas laika intervāls $6 \cdot 8 = 48$ sek.;

Pielietojot regresijas analīzi imitācijas modeļa ārējās pārejas δ_{ext} notikumu izmaiņu pētīšanai atkarībā no konveijeru ātruma, ir iegūti rezultāti (4.14. attēls), kas statistiskos datus aptuveni ļauj aprakstīt ar šādas eksponenciālas funkcijas palīdzību:

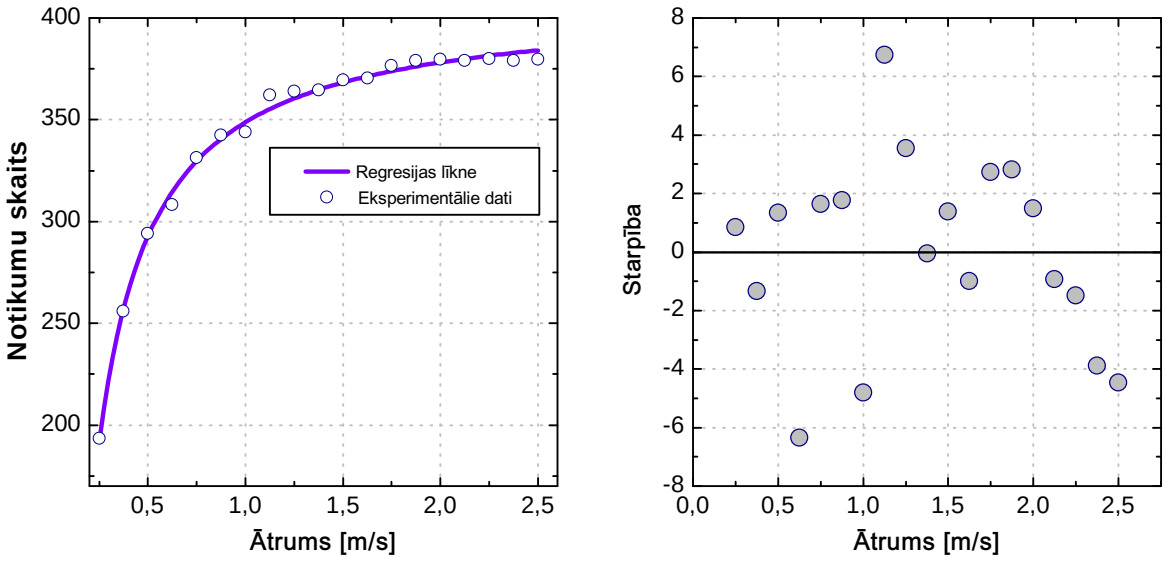
$$y = ae^{\frac{b}{x+c}}. \tag{4.2}$$

Dotās funkcijas noteiktās parametru vērtības ir attēlotas 4.6. tabulā, tādējādi iegūtais eksponenciālās regresijas vienādojums ir

$$y = 408,315 \cdot e^{\frac{-0,15}{x-0,051}}.$$

Lai novērtētu regresijas kvalitāti, kā palīglīdzeklis ir izmantota reziduālā analīze, kas parāda regresijas starpības lielumu varianci un mainīgo neatkarību. 4.14. attēlā parādītajai ārējās pārejas

funkcijas δ_{ext} regresijas starpību atkarībai no konveijeru ātruma nav vērojams pieaugošs vai dilstošs trends, kas norāda uz to, ka regresijas kļūdai ir konstanta variance.



(a) Regresijas līkne

(b) Starpību izkliede

4.14. att. Imitācijas modeļa ārējās pārejas δ_{ext} notikumu skaita regresijas līkne un starpību atkarība no konveijeru ātruma.

4.6. tabula

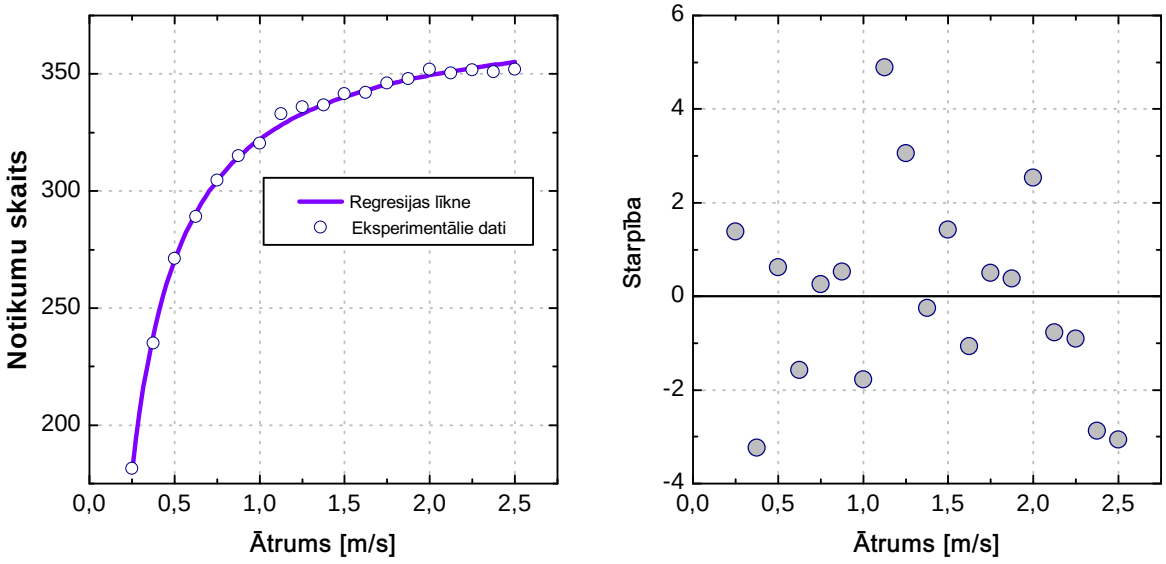
Ārējās pārejas funkcijas δ_{ext} regresijas vienādojuma parametri

			Stjūdenta kritērijs	
Parametrs	Vērtība	Standartnovirze	t-vērtība	p-vērtība
a	408,315	2,417	168,906	0
b	-0,15	0,009	-17,582	0
c	-0,051	0,013	-4,08	0,001

Pēc aprēķinātā Fišera kritērija $F = 64668,297$ eksponenciālās regresijas modelis ir būtisks pie būtiskuma līmeņa $\alpha = 0,05$, jo aprēķinātā p-vērtība $S_F = 0,00 < 0,05$. Arī pēc Stjūdenta kritērija dotais eksponenciālais modelis ir būtisks, jo visu regresijas modeļa parametru t-kritēriju p-vērtības (4.6. tabula) ir mazākas par uzdoto būtiskuma līmeni.

Diskrētās iekšējās pārejas funkcijas δ_{int}^{discr} ģenerēto stāvokļu pāreju atkarība no konveijeru ātruma (4.15. attēls) ir aprakstāma pēc tādas pašas sakarības (4.2.formula) kā ārējās pārejas funkcijas δ_{ext} pārejas gadījumā. Dotās funkcijas noteiktās parametru vērtības ir attēlotas 4.7. tabulā, tādējādi iegūtais eksponenciālās regresijas vienādojums ir

$$y = 377,783 \cdot e^{\frac{-0,152}{x-0,045}}.$$



(a) Regresijas līkne

(b) Starpību izkliede

4.15. att. Imitācijas modeļa diskrēto iekšējās pārejas δ_{int}^{discr} notikumu skaita regresijas līkne un starpību atkarība no konveijeru ātruma.

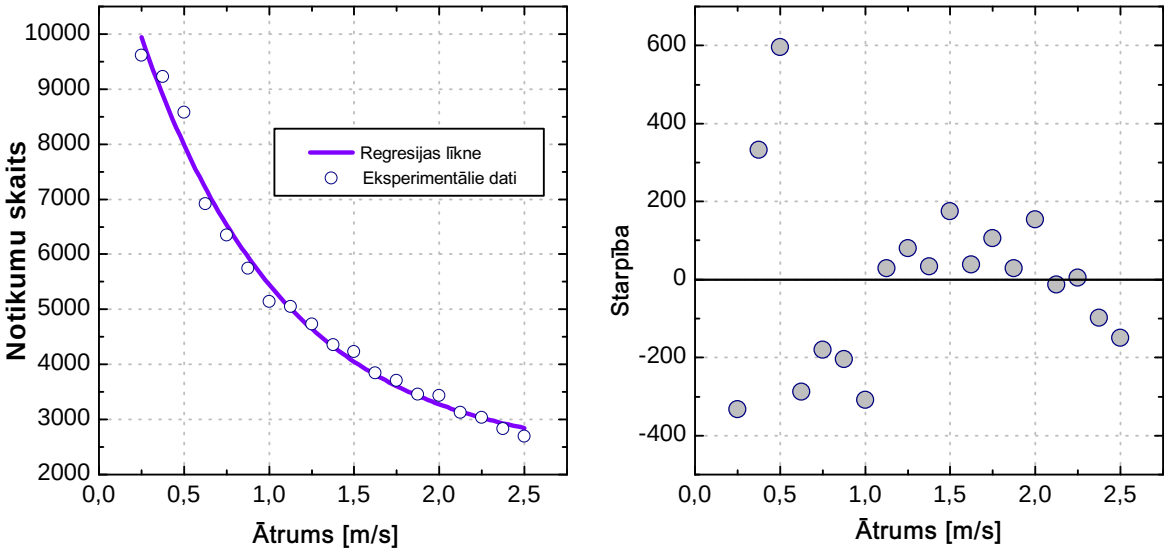
4.7. tabula

Diskrētās iekšējās pārejas funkcijas δ_{int}^{discr} regresijas vienādojuma parametri

			Stjūdenta kritērijs	
Parametrs	Vērtība	Standartnovirze	t-vērtība	p-vērtība
a	377,783	1,61	234,767	0
b	-0,152	0,006	-24,492	0
c	-0,045	0,009	-4,808	0

Pēc aprēķinātā Fišera kritērija $F = 129271,80627$ eksponenciālās regresijas modelis ir būtisks pie būtiskuma līmeņa $\alpha = 0,05$, jo aprēķinātā p-vērtība $S_F = 0,00 < 0,05$. Arī pēc Stjūdenta kritērija dotais eksponenciālais modelis ir būtisks, jo visu regresijas modeļa parametru t-kritēriju p-vērtības (4.7. tabula) ir mazākas par uzdoto būtiskuma līmeni.

Regresijas analīze, kas pielietota imitācijas modeļa nepārtraukto iekšējās pārejas δ_{int}^{cont} notikumu izmaiņu pētīšanai atkarībā no konveijeru ātruma, rāda (4.16. attēls), ka δ_{int}^{cont} notikumu izmaiņas modelī aptuveni raksturo eksponenciāla funkcija $y = y_0 + Ae^{R_0 x}$. Dotās funkcijas noteiktās parametru vērtības ir attēlotas 4.8. tabulā, tādējādi eksponenciālās regresijas vienādojums ir $y = 2314,707 + 10252,605 \cdot e^{-1,184 \cdot x}$.



(a) Regresijas līkne

(b) Starpību izkliede

4.16. att. Imitācijas modeļa nepārtraukto iekšējās pārejas δ_{int}^{cont} notikumu skaita regresijas līkne un starpību atkarība no konveijeru ātruma

4.8. tabula

Nepārtrauktās iekšējās pārejas funkcijas δ_{int}^{cont} regresijas vienādojuma parametri

			Stjūdenta kritērijs	
Parametrs	Vērtība	Standartnovirze	t-vērtība	p-vērtība
y_0	2314,707	231,729	9,989	0
A	10252,605	303,708	33,758	0
R_0	-1,184	0,103	-11,491	0

Pēc aprēķinātā Fišera kritērija $F = 3240,551$ eksponenciālās regresijas modelis ir būtisks pie būtiskuma līmeņa $\alpha = 0,05$, jo aprēķinātā p-vērtība $S_F = 0,00 < 0,05$. Arī pēc Stjūdenta kritērija dotais eksponenciālais modelis ir būtisks, jo visu regresijas modeļa parametru t-kritēriju p-vērtības (4.8. tabula) ir mazākas par uzdoto būtiskuma līmeni.

4.2.3. Ražošanas sistēmas veiktspējas statistiskie rādītāji

Apskatāmās automatizētās ražošanas sistēmas veiktspējas statistiskie rādītāji tiek noteikti ar mērķi parādīt, ka izstrādātais V-DEVS formālisms un uz tā bāzētais imitācijas vides prototips ir pielietojams praktisku sistēmu imitācijas modelēšanā un ar to iegūtie statistiskie rezultāti ir adekvāti. Ar izstrādāto prototipu iegūto rezultātu adekvātuma pārbaudei tiek veikta to salīdzināšana ar līdzīgiem rezultātiem, kas iegūti ar komerciālu imitācijas modelēšanas līdzekli

Flexsim.

Modelējamās automatizētās ražošanas sistēmas veiktspējas statistisko rādītāju noteikšanā tiek pielietota darbos [5, 11, 61] izmantotā metodika. Vidējā ražošanas izpildes laika $\overline{LT}$ noteikšanai tiek izmantota šāda formula:

$$\overline{LT} = \frac{1}{N}\sum_{p=1}^{N} LT_p,$$

kur N - izgatavoto produkcijas vienību skaits plānošanas laika intervālā $[0, T]$;

$LT_p = T_F - T_A$ - vienas produkcijas vienības ražošanas izpildes laiks, kur

T_A - ražošanas cikla uzsākšanas laiks;

T_F - ražošanas cikla pabeigšanas laiks.

Vidējais ražošanas iekārtas apstrādes rindas garums:

$$\overline{AQ} = \frac{1}{T}\sum_{i=1}^{Q_{max}} iT_i^Q,$$

kur Q_{max} - maksimālais rindas garums;

T_i^Q - kopējais laiks intervālā $[0, T]$, kādā detaļas ar apjomu i atrodas gaidīšanas rindā.

Veiktspējas statistisko rādītāju iegūšanai ir veikti 5 eksperimenti, mainot konveijeru ātrumu intervālā $[0,1; 0,5]$ m/s. Katrs eksperiments ir atkārtots 10 reizes, un katra imitācijas cikla garums ir izvēlēts atbilstošs 720 reālā laika ražošanas stundām jeb 30 dienām. 4.9. tabulā ir parādīti ar interaktīvās vizuālās imitācijas modelēšanas vides prototipu iegūtie ražošanas sistēmas veiktspējas vidējie statistiskie rādītāji.

4.9. tabula

V-DEVS vizuālās imitācijas modelēšanas vidē iegūtie automatizētās ražošanas sistēmas veiktspējas vidējie statistiskie rādītāji

			Salicējs	
Konveijeru ātrums (m/s)	Samontēto virsbūvju skaits	Ražošanas izpildes laiks $\overline{LT}$ (min.)	Virsbūvju rindas garums $\overline{AQ_V}$	Durvju rindas garums $\overline{AQ_D}$
0,1	18097	68,86	0,38	0,47
0,2	23379	56,25	0,5	0,26
0,3	23294	51,6	0,55	0,23
0,4	22989	49,1	0,58	0,19
0,5	23747	47,83	0,7	0,26

4.10. tabulā ir parādīti Flexsim imitācijas modelēšanas vidē iegūtie ražošanas sistēmas veiktspējas vidējie statistiskie rādītāji.

4.10. tabula
Flexsim vidē iegūtie automatizētās ražošanas sistēmas veiktspējas vidējie statistiskie rādītāji

Konveijeru ātrums (m/s)	Samontēto virsbūvju skaits	Ražošanas izpildes laiks $\overline{LT}$ (min.)	Salicējs	
			Virsbūvju rindas garums $\overline{AQ_V}$	Durvju rindas garums $\overline{AQ_D}$
0,1	18149	68,87	0,38	0,45
0,2	23433	56,2	0,5	0,26
0,3	23210	51,69	0,59	0,25
0,4	23028	49,1	0,59	0,19
0,5	23762	47,9	0,73	0,25

Salīdzinot ar V-DEVS imitācijas modelēšanas prototipu un Flexsim rīku iegūtos modelēšanas rezultātus, kas redzami 4.9. un 4.10. tabulā, ir redzams, ka tie ir līdzīgi, kas parāda, ka izstrādātais prototips ļauj iegūt adekvātus imitācijas modelēšanas rezultātus.

4.3. Kopsavilkums un secinājumi

Šajā nodaļā ir aprakstīts V-DEVS formālisma praktiskais pielietojums dinamisku sistēmu modelēšanā, balstoties uz automatizētās ražošanas sistēmas piemēru un pielietojot izstrādāto interaktīvā vizuālās imitācijas modelēšanas sistēmas prototipu.

Nodaļas ietvaros paveiktais:

- ir veikta automatizētās ražošanas sistēmas imitācijas modeļa izstrāde, izmantojot komerciālo vizuālo imitācijas modelēšanas rīku Flexsim un promcijas darba ietvaros izstrādāto V-DEVS imitācijas modelēšanas sistēmas prototipu;
- ir veikti imitācijas eksperimenti ar izstrādāto automatizētās ražošanas sistēmas modeli;
- ir novērtēta imitācijas modelēšanas sistēmas prototipa vizualizācijas un imitācijas veiktspēja, izmantojot hierarhisko un prioritātes rindu imitācijas algoritmus.

Sasniegtie rezultāti ir šādi:

- ir izveidots Flexsim un V-DEVS sistēmu salīdzinājums, balstoties uz izstrādāto automatizētās ražošanas sistēmas modeli;
- ir iegūti imitācijas modelēšanas sistēmas prototipa veiktspējas mērījumi;
- ir iegūti automatizētās ražošanas sistēmas imitācijas modelēšanas statistiskie rezultāti.

Galvenie secinājumi ir šādi:

- V-DEVS formālismu ir iespējams pielietot praktisku interaktīvu imitācijas modelēšanas sistēmu izveidē, ko pierāda ar imitācijas programmatūras prototipu izstrādātā automatizētās ražošanas sistēmas modeļa eksperimentālie rezultāti;
- uz V-DEVS formālismu balstīta interaktīva vizuālā imitācijas modelēšanas sistēma ir efektīvi pielietojama un var konkurēt ar komerciālām programmatūras sistēmām sarežģītu dinamisku sistēmu imitācijas modelēšanā.
- neraugoties uz hierarhiskā algoritma vienkāršo konceptuālo arhitektūru, tas ir ar ievērojami mazāku veiktspēju, nekā uzlabotais prioritātes rindu algoritms. No tā izriet, ka pieaugot imitācijas modeļa elementu un to savstarpējo saišu skaitam, prioritātes rindu V-DEVS simulators ir labāk piemērots, nekā hierarhiskais algoritms.

GALVENIE REZULTĀTI UN SECINĀJUMI

Promocijas darbā tika izvirzīts mērķis, balstoties uz imitācijas modelēšanas un vizualizācijas integrācijas salīdzinošu pētījumu un neatrisināto problēmu identificēšanu, izstrādāt integrētu pieeju un metodes diskrētu notikumu un nepārtrauktu sistēmu imitācijas modelēšanai un vizualizācijai, tās praktiski realizēt programmatūras sistēmā, un veikt izstrādātās sistēmas eksperimentālu pārbaudi. Izvirzītā mērķa sasniegšanai ir atrisināti šādi uzdevumi:

- imitācijas modelēšanas un vizualizācijas pieeju, metožu un sistēmu izpēte, kas ir ļāvusi identificēt to izstrādē un integrācijā neatrisinātos uzdevumus, kā arī definēt vispārīgas prasības interaktīvai vizuālai imitācijas modelēšanas videi;
- integrētas vizuālās imitācijas modelēšanas pieejas un metožu izstrāde iepriekš identificēto trūkumu novēršanai;
- uz izstrādātās pieejas balstītas diskrētu notikumu un nepārtrauktu sistēmu interaktīvas vizuālās imitācijas arhitektūras izstrāde un praktiskā realizācija;
- izstrādātās sistēmas eksperimentāla pārbaude tās praktiskās pielietošanas iespēju noteikšanai.

Pētījumu rezultātā ir identificēti šādi neatrisinātie uzdevumi integrētu imitācijas modelēšanas un vizualizācijas sistēmu izstrādē:

- maz teorētisku pētījumu un praktisku rezultātu imitācijas modelēšanas un vizualizācijas interaktīvas mijiedarbības jomā, imitācijas modeļa lietotājiem nav iespēju mijiearboties ar modeli tā izpildes laikā;
- atšķirīgas pieejas un metodes diskrētu notikumu un nepārtrauktu sistēmu imitācijas modelēšanā, un ar to saistītās vizualizācijas sinhronizācijas problēmas.

Darba teorētiskie rezultāti

Risinot minētās problēmas, promocijas darbā ir sasniegi šādi galvenie teorētiskie rezultāti:

- pamatota diskrētu notikumu un nepārtrauktu sistēmu imitācijas modelēšanas un vizualizācijas integrācijas nepieciešamība;
- izstrādāts sistēmteorētisks V-DEVS formālisms vizuālai interaktīvai diskrēto notikumu un nepārtrauktu sistēmu modelēšanai;

- izstrādāta metodika un algoritmi V-DEVS formālisma praktiskajai realizācijai;
- piedāvāta kombinēta pieeja nepārtrauktu procesu imitācijas modelēšanas un vizualizācijas sinhronizācijai;
- izstrādāta modeļvadāma pieeja V-DEVS modelēšanai.

Darba praktiskie rezultāti

Darba izstrāde ir ļāvusi sasniegt šādus praktiskos rezultātus:

- izstrādāta arhitektūra, metodika un algoritmi V-DEVS formālisma praktiskai realizācijai, kas no tradicionālām imitācijas modelēšanas sistēmām atšķiras ar vienotu modelēšanas, imitācijas, vizualizācijas un interaktīvo elementu traktējumu, unificējot un vienkāršojot imitācijas modeļu izstrādi un izmantošanu;
- realizēts V-DEVS vizuālās interaktīvās imitācijas modelēšanas sistēmas prototips, pārbaudītas un nodemonstrētas tā iespējas.

V-DEVS imitācijas modelēšanas vides prototipa izstrāde un eksperimentālās pārbaudes rezultāti ļauj izdarīt šādus secinājumus:

- darbā piedāvātais V-DEVS vizualizācijas konveijers ļauj adaptēt scēnas grafa, gan klasiskā vizualizācijas konveijera sistēmas imitācijas modelēšanas uzdevumiem, apvienojot šīs arhitektūras vienotā sistēmā.
- izstrādātajam prioritātes rindu V-DEVS simulatora algoritmam ir lielāka efektivitāte, nekā hierarhiskajam V-DEVS simulatoram, kas izstrādāts, balstoties uz klasisko DEVS formālisma imitācijas algoritmu, tādējādi prioritātes rindu algoritms ir labāk piemērots sarežģītu imitācijas modelēšanas uzdevumu risināšanai.
- V-DEVS sistēmas prototipa eksperimentālā pārbaude apliecina kvantētu stāvokļu interpolatora izmantošanas efektivitāti vizualizācijas nolūkiem nepārtrauktu sistēmu kvantētu stāvokļu imitācijas modelēšanā;
- modeļvadāmā pieeja uzlabo un vienkāršo imitācijas modeļa izstrādes un verifikācijas procesu.

Turpmāko pētījumu iespējamie virzieni

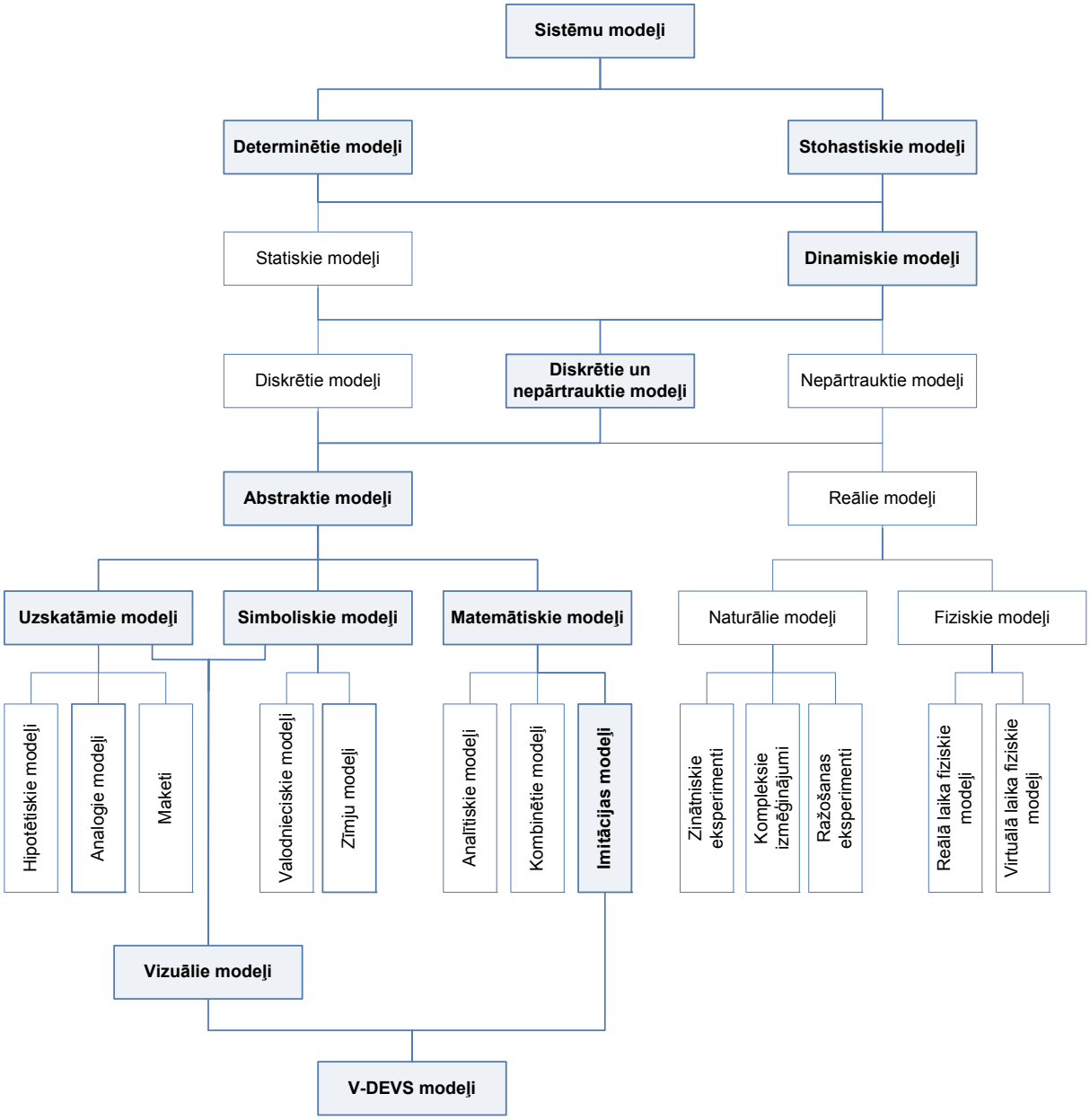
Darba autors, izstrādājot integrētu interaktīvu vizuālu imitācijas modelēšanas sistēmu, saskata tālākas piedāvāto koncepciju pilnveidošanas un attīstības iespējas, kas ļauj formulēt vairākus turpmāko pētījumu virzienus. Galvenie šo turpmāko pētījumu iespējamie virzieni ir šādi:

- V-DEVS formālisma pielāgošana dinamiskas struktūras sistēmu modelēšanai, kas ir nepieciešams arī tiešās izpildes V-DEVS vizualizācijas konveijera realizācijai;
- dalītās un paralēlās apstrādes principu realizācija V-DEVS balstītu sistēmu darbības nodrošināšanai daudzprocesoru un/vai tīmekļa vidēs;
- daudzlietotāju režīma atbalsts interaktīvā imitācijas izpildes vidē, nodrošinot dažādas piekļuves tiesības dažādām lietotāju grupām un sinhronizāciju starp lietotājiem;
- V-DEVS simulatora veiktspējas uzlabošanas iespēju izpēte nepārtrauktu sistēmu imitācijas modelēšanā;
- kvantētu stāvokļu V-DEVS imitācijas stabilitātes pētījumi nelineāru nepārtrauktu sistēmu imitācijas modelēšanā;
- lietotāju interaktīvās mijiedarbības matemātiskā pamatojuma izstrāde;
- V-DEVS integrācija ar virtuālās / jauktās realitātes vizualizācijas tehnoloģijām.

PIELIKUMI

1.pielikums

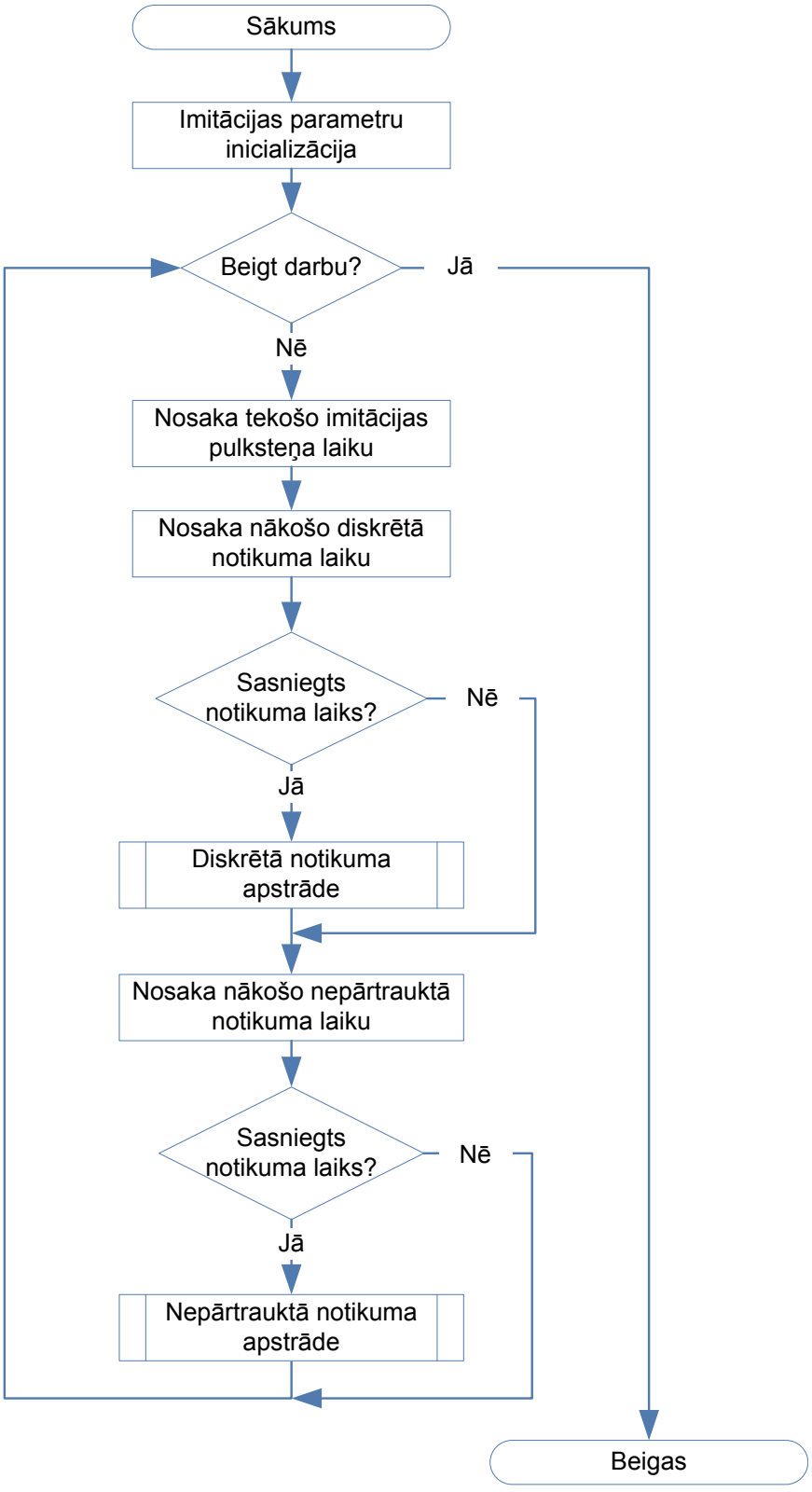
V-DEVS formālisma vieta vispārējā sistēmu modelēšanas veidu klasifikācijā



P1.1. att. V-DEVS formālisma vieta vispārējā sistēmu modelēšanas veidu klasifikācijā (adaptēts no [118])

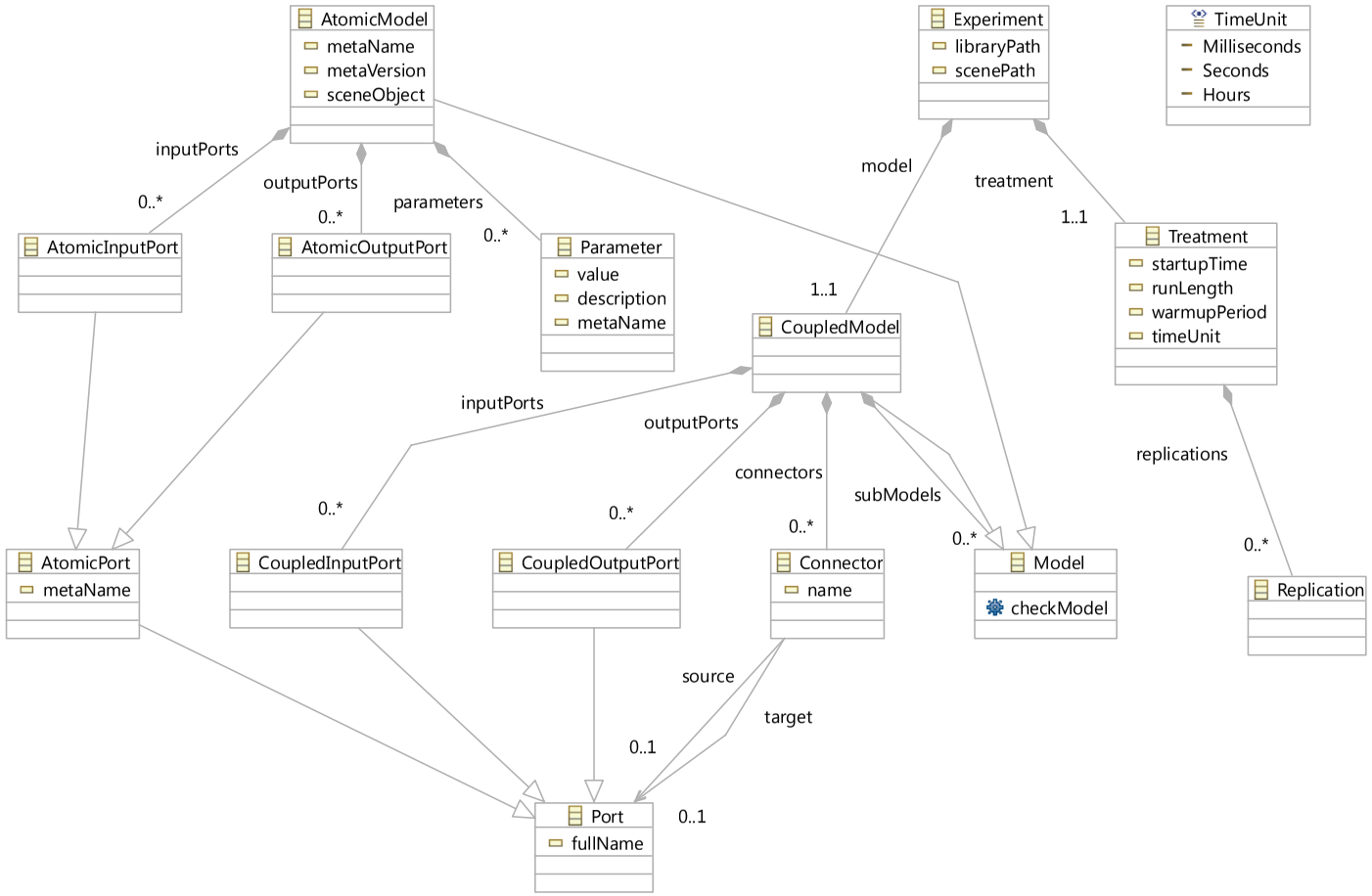
2.pielikums

V-DEVS saknes koordinatora vispārējs darbības algoritms

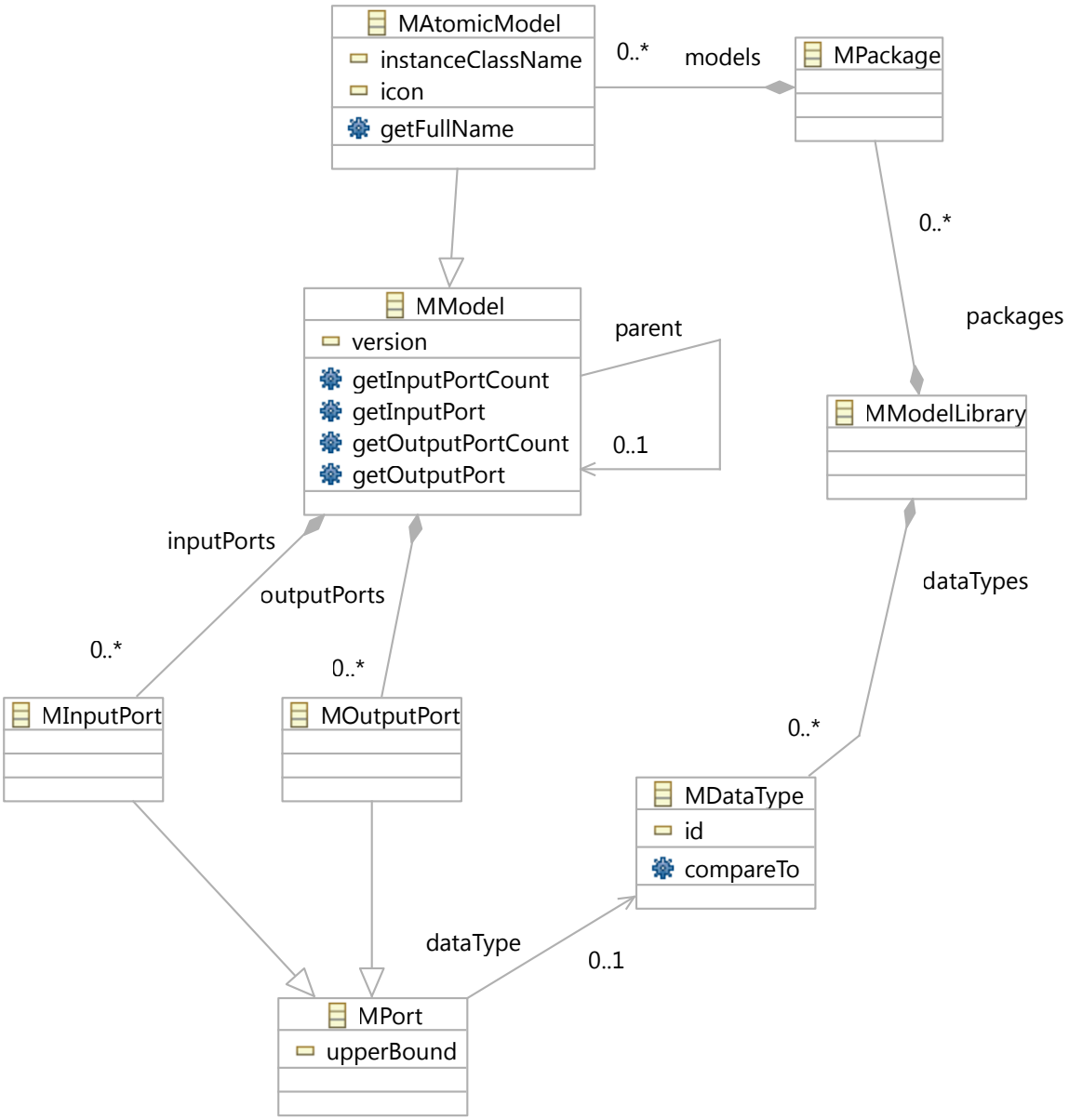


P2.1. att. V-DEVS saknes koordinatora vispārējs darbības algoritms

V-DEVS modeļvadāmā realizācija

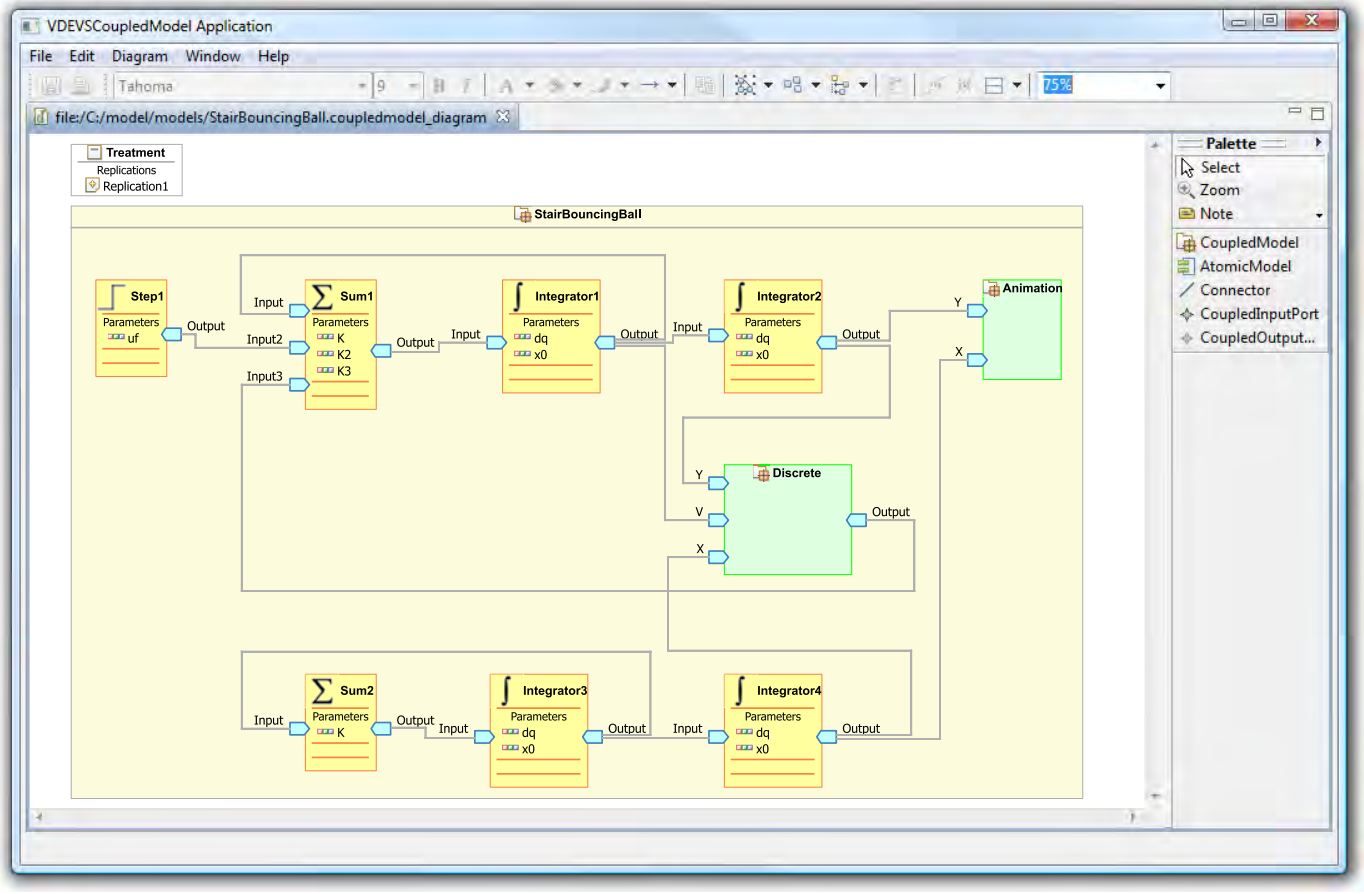


P3.1. att. V-DEVS metamodeļa klašu diagramma vizuālo imitācijas modeļa diagrammu izveidei, kas V-DEVS programmatūras prototipā realizēta ar Eclipse EMF tehnoloģiju

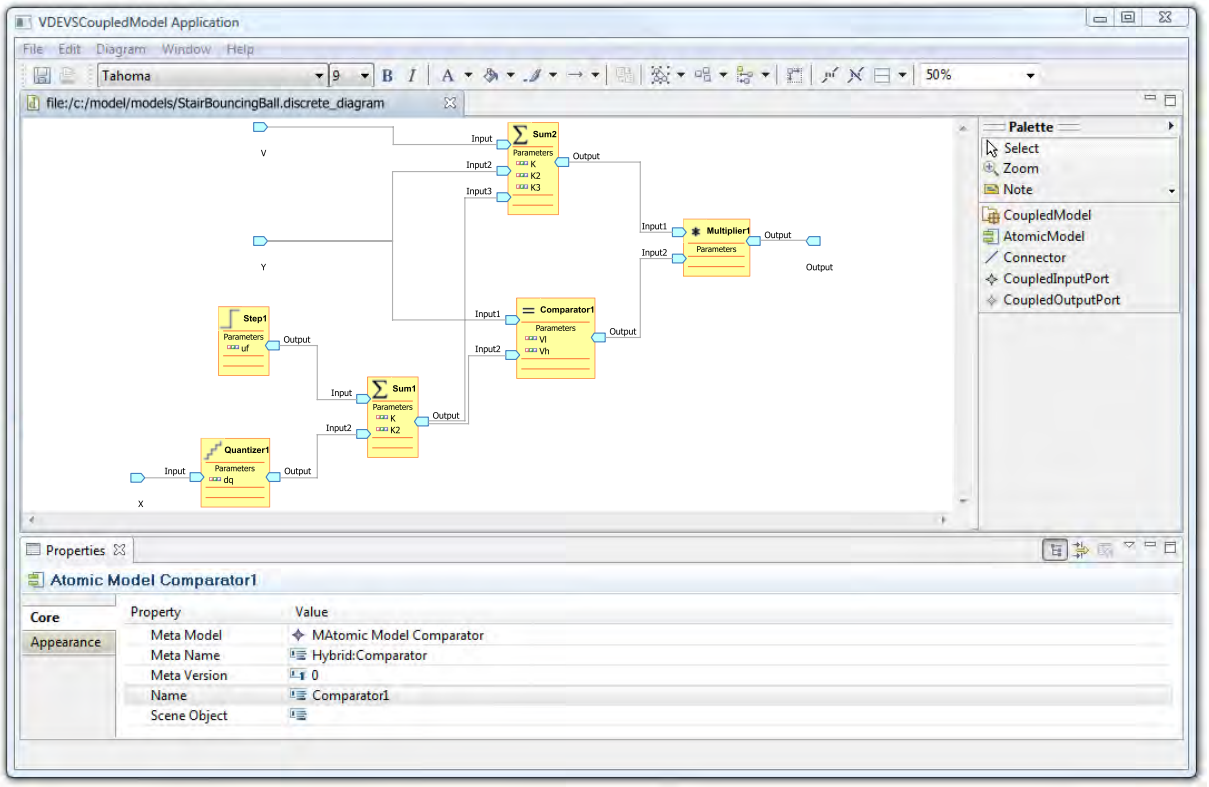


P3.2. att. V-DEVS metametamodeļa klašu diagramma vizuālo imitācijas modeļa elementu izveidei, kas V-DEVS programmatūras prototipā realizēta ar Eclipse EMF tehnoloģiju

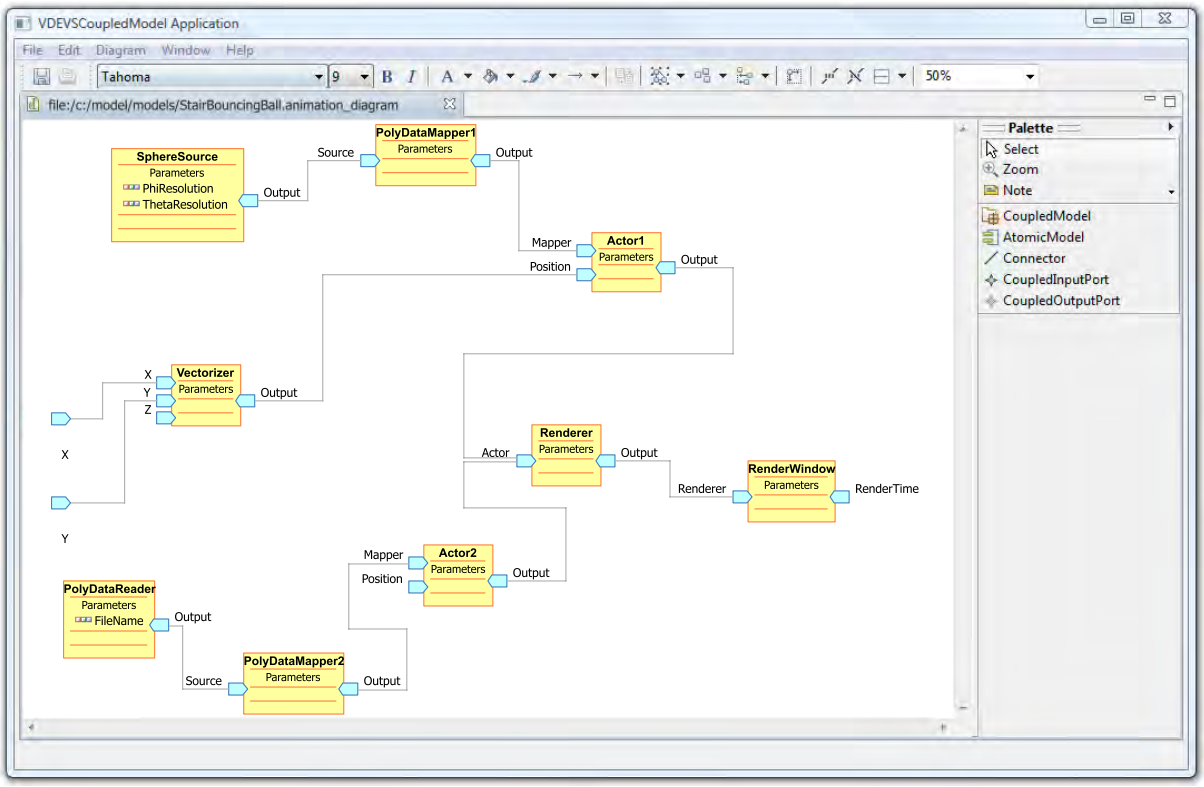
Pa kāpnēm lejup ripojošas lodes modelis



P4.1. att. Pa kāpnēm lejup ripojošas lodes V-DEVS modelis, kas izstrādāts ar V-DEVS vizuālās imitācijas modelēšanas prototipu, izmantojot modeļvadāmo pieeju



P4.2. att. Pa kāpnēm lejup ripojošas lodes diskrētās daļas V-DEVS apakšmodelis



P4.3. att. Pa kāpnēm lejup ripojošas lodes V-DEVS animācijas apakšmodelis

LITERATŪRA

[1] Applied Materials, Inc.: *Applied AutoMod^{TM}: Simulation and modeling software* / Internets. - http://www.brookssoftware.com/pages/245_automod_overview.cfm.

[2] AVS Advanced Visual Systems: *AVS / Express* / Internets. - http://www.avs.com/software/soft_t/avsxps.html.

[3] Balci O., Nance R.E. *A taxonomy of layout composition techniques for visual simulation// WSC '98: Proceedings of the 30th conference on Winter simulation*. - Los Alamitos, CA, USA: IEEE Computer Society Press, 1998. - July 19-22. - 279–288 p.

[4] Ball P., Love D. *The key to object-oriented simulation: Separating the user and the developer// WSC '95: Proceedings of the 27th conference on Winter simulation*. - Washington, DC, USA: IEEE Computer Society, 1995. - 768–774 p.

[5] *Discrete-Event System Simulation*, Banks J., Carson J.S., L. N.B., Nicol D.M. - Upper Saddle River, NJ: Pearson Prentice Hall, 2005. - 4th ed. - 624 p.

[6] Bapat V., Sturrock D.T. *The Arena product family: Enterprise modeling solutions// WSC '03: Proceedings of the 35th conference on Winter simulation*. - New Orleans, Louisiana, 2003. - 210–217 p.

[7] Bar-Zeev A.: *Scenegraphs: Past, present, and future* / Internets. - http://www.realityprime.com/articles/scenegraphs-past-present-and-future.

[8] Barnes M.R. *An introduction to QUEST// WSC '97: Proceedings of the 29th conference on Winter simulation*. - Washington, DC, USA: IEEE Computer Society, 1997. - 619–623 p.

[9] Bell P., O'Keefe R. *Visual interactive simulation - history, recent developments, and major issuses//* Simulation. - 1987. - No. 49. - Vol. 3. - 109–116 p.

[10] Blundell B.G. *An Introduction to Computer Graphics and Creative 3-D Enviroments*. - London: Springer Verlag, 2008. - 501 p.

[11] Borenstein D. *A visual interactive multicriteria decision analysis model for FMS design//* The International Journal of Advanced Manufacturing Technology. - 1998. - No. 11. - Vol. 14. - 848–857 p.

[12] Cañas A.J., Carff R., Hill G., Carvalho M., Arguedas M., Eskridge T.C., Lott J., Carvajal R. *Concept maps: Integrating knowledge and information visualization// Knowledge and Information Visualization*. Springer, 2005. - 205–219 p.

[13] Carsten T. *Interface-oriented classification of DEVS models.// Proceedings of the Fifth Annual Conference on AI, Simulation and Planning in High Autonomy Systems.* IEEE Computer Press, 1994. - 208–213 p.

[14] Cellier F.E., Kofman E. *Continuous System Simulation*. - New York: Springer, 2006. - 643 p.

[15] Chawla R., Banerjee A. *A virtual environment for simulating manufacturing operations in 3D// WSC '01: Proceedings of the 33nd conference on Winter simulation*. - Washington, DC, USA: IEEE Computer Society, 2001. - December 12. - 991–997 p.

[16] Choi B.K., Han K., Park T. *Object-oriented graphical modeling of FMSs//* International Journal of Flexible Manufacturing Systems. - 1996. - Vol. 8. - 159–182 p.

[17] Choi B.K., Park B.C., Park J.H. *A formal model conversion approach to developing a DEVS-based factory simulator//* Simulation. - 2003. - No. 8. - Vol. 79. - 440–461 p.

[18] Churcher N., Irwin W. *Informing the design of pipeline-based software visualisations//* Hong S.H. (ed.): *Asia Pacific Symposium on Information Visualisation (APVIS 2005)*, vol. 45 of *ACM International Conference Proceeding Series*. - Sydney, Australia, Australian Computer Society, Inc., 2005. - 59–68 p.

[19] Cloud D., Rainer L. *Applied Modelling & Simulation: An Integrated Approach to Development and Operation*. - New York: AIAA American Institute of Aeronautics and Astronautics, 1998. - 736 p.

[20] Döllner J., Hinrichs K. *A generic 3D rendering system.//* IEEE Transactions on Visualization and Computer Graphics. - 2002. - No. 2. - Vol. 8. - 99–118 p.

[21] Donald D.L. *A tutorial on ergonomic and process modeling using QUEST and IGRIP// WSC '98: Proceedings of the 30th conference on Winter simulation*. - Los Alamitos, CA, USA: IEEE Computer Society Press, 1998. - 297–302 p.

[22] Durand F. *An invitation to discuss computer depiction.// NPAR '02: Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*. - New York, NY, USA: ACM, 2002. - June. - 111–124 p.

[23] Dwyer T. *Three dimensional UML using force directed layout//* Eades P., Pattison T. (eds.): *Australasian Symposium on Information Visualisation, InVis.au, Sydney, Australia, 3-4 December 2001*, vol. 9 of *CRPIT*. Australian Computer Society, 2001. - 77–85 p.

[24] Eberly D.H. *3D Game Engine Architecture: Engineering Real-Time Applications with Wild Magic*. - Amsterdam; Boston: Morgan Kaufmann Publishers, 2005. - 2nd ed. - 752 p.

[25] Fishwick P.A. *Computer simulation: Growth through extension// Proceedings of European Simulation Multiconference*. - 1994. - 3–20 p.

[26] Fishwick P.A. *A visual object-oriented multimodeling design for physical modeling*. - Techn. rep., ACM Transactions on Modeling and Computer Simulation, 1997. - 22 p.

[27] Flexsim Software Products, Inc.: *Flexsim simulation software* / Internets. - http://www.flexsim.com.

[28] Frankel D.S. *Model Driven Architecture: Applying MDA to Enterprise Computing*. - Indianapolis: Wiley Publishing, 2003. - 352 p.

[29] Friendly M. *A brief history of data visualization.* Chen C.H., Härdle W., Unwin A. (eds.): *Handbook of Computational Statistics: Data Visualization*, vol. III, Springer-Verlag, Heidelberg, 2008, - 15–56 p.

[30] Garcia I., Molla R. *Videogames decoupled discrete event simulation*// Computers & Graphics. - 2005. - No. 29. - Vol. 29. - 195–202 p.

[31] Haber R., McNabb D. *Visualization idioms: A conceptual model for scientific visualization systems*. Nielson G., Shriver B., Rosenblum L. (eds.): *Visualization in Scientific Computing*, IEEE Computer Society Press, 1990, - 74–93 p.

[32] Harrell C.R., Price R.N. *Simulation modeling using ProModel technology*// *WSC '03: Proceedings of the 35th conference on Winter simulation*. - 2003. - 175–181 p.

[33] Henriksen J.O. *Adding animation to a simulation using Proof*// *WSC '00: Proceedings of the 32nd conference on Winter simulation*. - San Diego, CA, USA: Society for Computer Simulation International, 2000. - 191–196 p.

[34] Herrmann J.W. (ed.) *Handbook of Production Scheduling*. International Series in Operations Research & Management Science: Springer Science+Business Media, Inc., 2006. - 330 p.

[35] Hwang M.H., Choi B.K. *GK-DEVS: Geometric and kinematic DEVS formalism for simulation modeling of 3-dimensional multi-component systems*// Transactions of the Society for Modeling and Simulation International. - 2001. - No. 3. - Vol. 18. - 159–173 p.

[36] Imagine That Inc.: *ExtendSim* / Internets. - http://www.extendsim.com.

[37] Incontrol Enterprise Dynamics: *Enterprise Dynamics simulation software* / Internets. - http://www.enterprisedynamics.com.

[38] Incontrol Enterprise Dynamics GmbH: *Showflow simulation software* / Internets. - http://www.showflow.com.

[39] isee systems, inc.: *iThink: Systems thinking for business* / Internets. - http://www.iseesystems.com/Softwares/Business/ithinkSoftware.aspx.

[40] Jacobs P.H.M. *The DSOL simulation suite: Enabling multi-formalism simulation in a distributed context*. - PhD thesis, Delf, 2005. - 216 p.

[41] Jain S. *Simulation in the next millenium*// *Proceedings of the 1999 Winter Simulation Conference*. - Phoenix, Australia, AZ, 1999. - December 5-8. - 1478–1484 p.

[42] jMonkeyEngine.com: *jME (jMonkey Engine): High performance open source Java-based game engine* / Internets. - http://www.jmonkeyengine.com.

[43] Kelton D.W., Sadowski R.P., Sadowski D.A. *Simulation with Arena*. - Boston, Mass.: McGraw-Hill, 2003. - 672 p.

[44] Khoshnevis B. *Discrete Systems Simulation.* - New York: McGraw-Hill, 1994. - 416 p.

[45] Kofman E. *Discrete Event Based Simulation and Control of Continuous Systems*. - PhD thesis, Universidad Nacional de Rosario, Argentina, 2003. - 167 p.

[46] Kofman E., Lapadula M., Pagliero E. *PowerDEVS: A DEVS-based environment for hybrid system modeling and simulation*. - Tech. Rep. LSD0306, School of Electronic Engineering, Universidad Nacional de Rosario, Rosario, Argentina, 2003. - 25 p.

[47] Korhonen A. *Visual Algorithm Simulation*. - PhD thesis, Helsinki University of Technology, Department of Computer Science and Engineering, Laboratory of Information Processing Science, 2003. - 136 p.

[48] Lanner Group Limited: *Witness: Simulation software for professional users* / Internets. - http://www.lanner.com/en/witness.cfm.

[49] Law A.M. *Simulation Modeling and Analysis*: McGraw-Hill, 2007. - 4th ed. - 800 p.

[50] Lektauers A. *Imitācijas modeļu vizualizācijas tehnoloģija// Rīgas Tehniskās universitātes zinātniskie raksti*, 5.sēr., Datorzinātne. - Rīga: RTU, 2001. - 103–109 p.

[51] Lektauers A. *Imitācijas modeļu izveide virtuālā 3D vidē// Rīgas Tehniskās universitātes zinātniskie raksti*, 5.sēr., Datorzinātne. - Rīga: RTU, 2002. - 107–113 p.

[52] Lektauers A. *3D-Projektierung von Simulationsmodellen in einer VR-Umgebung// Tagung Simulation und Visualisierung 2003*. - Magdeburg: Otto-von-Guericke Universität Magdeburg, SCS Publishing House, 2003. - 313–322 p.

[53] Lektauers A. *Developing a 3D graphical user interface for object-oriented simulation modelling// Rīgas Tehniskās universitātes zinātniskie raksti*, vol. 14 of *5*. - Rīga: RTU, 2003. - 36–42 p.

[54] Lektauers A. *A virtual 3D environment for logistics modelling// Proceedings of the International Workshop on Harbour, Maritime and Multimodal Logistics Modelling & Simulation (HMS2003)*. - Riga: RTU, 2003. - September 18-20. - 242–248 p.

[55] Lektauers A. *A mixed reality framework for visualization and execution of DEVS-based simulation models// Proceedings of the* 19^{th} *European Conference on Modelling and Simulation "Simulation in Wider Europe", ECMS2005*. - Riga: Publishing House of Riga Technical University, 2005. - June 1-4. - 271–276 p.

[56] Lektauers A., Merkuryev Y. *3D visual framework for modelling and simulation of supply chain systems// Scientific Proceedings of the eLOGMAR-M Project*. - Riga, 2006. - 141–150 p.

[57] Lengler R., Eppler M. *Towards a periodic table of visualization methods for management// Proceedings of the Conference on Graphics and Visualization in Engineering*. - Clearwater, Florida, USA, 2007. - 1–6 p.

[58] Mackinlay J.D. *Opportunities for information visualization//* Computer Graphics and Applications. - 2000. - No. 1. - Vol. 20. - 22–23 p.

[59] Marshall R., Kempf J., Dyer S. un Yen C. *Visualization Methods and Simulation Steering for a 3D Turbulence Model of Lake Erie//* In:Riesenfeld R. un Sequin C. (Hrsg.): *Computer Graphics: 1990 Symposium on Interactive 3D Graphics*, Bd. 24. - 1990. - 89–97 lpp.

[60] McCormick B., DeFanti T., Brown M. *Visualization in scientific computing - a synopsis//* IEEE Computer Graphics and Applications. - 1987. - No. 7. - Vol. 7. - 61–70 p.

[61] *Sistēmu imitācijas modelēšanas tehnoloģija*, Merkurjevs J., Merkurjeva G., Pečerska J. un Tolujevs J. - Rīga: Rīgas Tehniskās universitātes Datorzinātnes un informācijas tehnoloģijas fakultātes Informācijas tehnoloģijas institūts, 2008. - 120 lpp.

[62] Micro Analysis & Design, Inc.: *Micro Saint Sharp simulation software* / Internets. - http://www.maad.com/index.pl/micro_saint.

[63] Mueck B., Dangelmaier W., Fischer M., Klemisch W. *Bi-directional coupling of simulation tools with a walkthrough-system*// Schulze T., Schlechtweg S., Hinz V. (eds.): *Simulation und Visualisierung 2002 (SimVis 2002)*. - Magdeburg: SCS European Publishing House, 2002. - 28. Februar - 1. März. - 71–84 p.

[64] Muzy A., Nutaro J. *Algorithms for efficient implementations of the DEVS & DSDEVS abstract simulators.*// *Proceedings of the 1st Open International Conference on Modeling & Simulation*. - France, ISIMA/Blaise Pascal University, 2005. - June. - 401–407 p.

[65] Nischwitz A., Fischer M., Haberäcker P. *Computer Graphik und Bildverarbeitung*. - Wiesbaden: Friedr. Vieweg & Sohn Verlag, 2007. - 2nd ed. - 873 p.

[66] Nordstrom G.G. *Metamodeling – Rapid Design and Evolution of Domain-Specific Modeling Environments*. - PhD thesis, Vanderbilt University, Nashville, Tennessee, 1999. - 158 p.

[67] Nutaro J.: *adevs (A Discrete EVent System simulator)*. - http://www.ece.arizona.edu.

[68] Odhabi H.I., Paul R.J., Macredie R.D. *Developing a graphical user interface for discrete event simulation*// Paul R. (ed.): *WSC '98: Proceedings of the 30th conference on Winter simulation*, vol. 1. - Los Alamitos, CA, USA: IEEE Computer Society Press, 1998. - 429–436 p.

[69] Odhabi H.I., Paul R.J., Macredie R.D. *Making simulation more accessible in manufacturing systems through a 'four phase' approach*// Paul R. (ed.): *WSC '98: Proceedings of the 30th conference on Winter simulation*, vol. 2. - Los Alamitos, CA, USA: IEEE Computer Society Press, 1998. - 1069–1075 p.

[70] Parent R. *Computer Animation: Algorithms and Techniques*. The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. - San Francisco: Morgan Kaufmann Publishers, 2002. - 527 p.

[71] Pelz G. *Mechatronic Systems: Modelling and Simulation with HDLs*. - Hoboken, NJ: John Wiley & Sons, 2003. - 236 p.

[72] Pidd M., Castro R.B. *Hierarchical modular modelling in discrete simulation*// *WSC '98: Proceedings of the 30th conference on Winter simulation*. - Los Alamitos, CA, USA: IEEE Computer Society Press, 1998. - 383—-390 p.

[73] Popović J., Hoppe H. *Progressive simplical complexes*// Computer Graphics. - 1997. - No. Annual Conference Series. - Vol. 31. - 217–224 p.

[74] Posse E., Bolduc J.S. *Generation of DEVS modelling & simulation environments*// *Proceedings of the 2003 Summer Simulation MultiConference (SCSC'03)*. - Montréal, Canada, 2003. - July.

[75] Powersim Software AS: *Powersim Software's Studio* / Internets. - http://www.powersim.com.

[76] Praehofer H. *System Theoretic Foundation for Combined Discrete-Continuous System Simulation*. - PhD thesis, Department of Systems Theory, University of Linz, Austria, 1991. - 173 p.

[77] *Numerical Recipes in C: The Art of Scientific Computing*, Press W.H., Flannery B.P., Teukolsky S.A., Vetterling W.T.: Cambridge University Press, 1992. - 2nd ed. - 994 p.

[78] Radatz J. *The IEEE Standard Dictionary of Electrical and Electronic Terms*. - New York, NY, USA: IEEE Standards Office, 1997. - 6th ed. - 1278 p.

[79] Risco-Martín J.L., Mittal S., Zeigler B.P., de la Cruz J.M. *From UML state charts to DEVS state machines using XML// Proceedings of the Workshop on Multi-Paradigm Modeling: Concepts and Tools*, vol. 1 of *BME-DAAI Tech. Rep. Series*. - Budapest Univ. of Tech. and Economics Dep. Automation and Applied Informatics, 2007. - 35–48 p.

[80] Ritter K.C., Klein U., Straßburger S., Diessner M. *Web-basierte Animation verteilter Simulationen auf Basis der High Level Architecture (HLA)//* Lorenz P., Preim B. (eds.): *Simulation und Visualisierung 1998 (SimVis 1998)*. - Magdeburg: SCS Publishing House e.V., 1998. - 5-6 März. - 41–52 p..

[81] Rohrer M.W. *Seeing is believing: the importance of visualization in manufacturing simulation// WSC '00: Proceedings of the 32nd conference on Winter simulation*, vol. 2. - San Diego, CA, USA: Society for Computer Simulation International, 2000. - 1211–1216 p.

[82] Sahin F., Pushkin K. *Practical and Experimental Robotics*. - Boca Raton, FL: CRC Press, Taylor & Francis Group, 2008. - 439 p.

[83] Sandler B.Z. *Robotics: Designing the Mechanisms for Automated Machinery*. - San Diego: Academic Press, 1999. - 2nd ed. - 433 p.

[84] Schroeder W., Martin K., Lorensen B. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. - Upper Saddle River, N.J.: Pearson Education, Inc., 2004. - 3rd ed. - 474 p.

[85] Schroeder W., Yamrom B. *A compact cell structure for scientific visualization. SIGGRAPH '94 Course Notes CD-ROM, Course 4: Advanced Techniques for Scientific Visualization*, ACM SIGGRAPH, 1994, - 53–59 p.

[86] Schroeder W.J., Avila L.S., Hoffman W. *Visualizing with VTK: A tutorial//* IEEE Computer Graphics and Applications. - 2000. - No. 5. - Vol. 20. - 20–27 p.

[87] Shannon R.E. *Systems Simulation: The Art and Science*. - Englewood Cliffs, N.J.: Prentice-Hall Inc., 1975. - 368 p.

[88] Shmaliy Y. *Continuous-Time Systems*. - Dordrecht, The Netherlands: Springer, 2007. - 639 p.

[89] Siemens Product Lifecycle Management Software Inc.: *Tecnomatics® Plant Simulation software* / Internets. - http://www.emplant.com.

[90] St. John M., Cowen M.B., Smallman H.S., Oonk H.M. *The use of 2D and 3D displays for shape-understanding versus relative-position tasks. Human Factors*, vol. 43, Human Factors and Ergonomics Society, 2001, - 79—-98 p.

[91] Stanley B. *AutoMod : The AutoMod product site tutorial*// *Proceedings of the 33nd conference on Winter simulation*. - Washington, DC: IEEE Computer Society, 2001. - 209–216 p.

[92] *Software Visualization: Programming as a Multimedia Experience*, Stasko J., Domingue J., Brown M., Price B.A. - Cambridge, Mass.: MIT Press, 1998. - 580 p.

[93] Strassburger S., Schulze T. *Temporally parallel coupling of discrete simulation systems with virtual reality systems*// *Proceedings of the 2005 Winter Simulation Conference*. - Orlando, FL, 2005. - December. - 1949–1957 p.

[94] The Eclipse Foundation: *Eclipse - an open development platform* / Internets. - http://www.eclipse.org.

[95] The MathWorks, Inc. *Simulink: Model-Based and System-Based Design*. - Natick, MA: The Mathworks, Inc., 2002. - 5th ed. - 472 p.

[96] Tory M., Kirkpatrick A.E., Atkins M.S., Möller T. *Visualization task performance with 2D, 3D, and combination displays*// IEEE Transactions on Visualization and Computer Graphics. - 2006. - No. 1. - Vol. 12. - 2–12 p.

[97] Tory M., Möller T., Atkins M.S., Kirkpatrick A.E. *Combining 2D and 3D views for orientation and relative position tasks*// *SIGCHI Conference on Human Factors in Computing Systems*. - Vienna, Austria, 2004. - 24-29 April.

[98] Traore M.K., Muzy A. *Capturing the dual relationship between simulation models and their context*// Simulation Modelling Practice and Theory. - 2006. - Vol. 14. - 126–142 p.

[99] Vangheluwe H., de Lara J., Mosterman P.J. *An introduction to multi-paradigm modelling and simulation*// *Proceedings AI Simulation and Planning AIS-2002*. - 2002. - 9–20 p.

[100] Visualization and Imagery Solutions, Inc.: *OpenDX: The open source software project based on IBM's Visualization Data Explorer* / Internets. - http://www.opendx.org.

[101] Wagner P., Freitas C., Wagner F. *A new paradigm for visual interactive modeling and simulation*// *8th European Simulation Symposium - ESS'96*, vol. 1. - Genova, Italy: Society for Computer Simulation, 1996. - October 24-26. - 142–145 p.

[102] Wainer G. *CD++: A toolkit to develop DEVS models*// Software: Practice and Experience. - 2002. - No. 32. - Vol. 13. - 1261–1306 p.

[103] Ware C., Franck G. *Viewing a graph in a virtual reality display is three times as good as a 2d diagram*// *Proceedings of IEEE Symposium on Visual Languages*. - 1994. - 182–183 p.

[104] Watt A., Watt M. *Advanced Animation and Rendering Techniques: Theory and Practice*. - Reading, Mass.: Addison-Wesley, 1992. - 472 p.

[105] Wenzel S., Bernhard J., Jessen U. *A taxonomy of visualization techniques for simulation in production and logistics*// Chick S., Sanchez P.J., Ferrin D., Morrice D.J. (eds.): *Proceedings of the 2003 Winter Simulation Conference*, vol. 1. - 2003. - 729–736 p.

[106] Wright H. *Introduction to Scientific Visualization*. - London: Springer, 2007. - 147 p.

[107] Zeigler B.P. *Theory of Modelling and Simulation.* - New York: John Wiley & Sons, 1976. - 435 p.

[108] Zeigler B.P. *Multifacetted Modelling and Discrete Event Simulation.* - London: Academic Press, 1984. - 372 p.

[109] Zeigler B.P. *DEVS today: recent advances in discrete event-based information technology// MASCOTS Conference*. - Orlando, FL, 2003. - October. - 148–161 p.

[110] Zeigler B.P. *Embedding DEV&DESS in DEVS: Characteristic behavior of hybrid models*. *Simulation Series*, vol. 38, Society for Computer Simulation, Huntsville, Alabama, 2006, - 125–132 p.

[111] Zeigler B.P., Praehofer H., Kim T.G. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems.* - San Diego, California: Academic Press, 2000. - 2nd ed. - 510 p.

[112] Zeigler B.P., Sarjoughian H.S.: *Introduction to DEVS modeling and simulation with Java: Developing component-based simulation models* / Internets. - www.acims.arizona.edu/EDUCATION/ECE575Fall03/notes/Manuscript_MSDJ_090903.pdf.

[113] Zeigler B.P., Vahie S. *DEVS formalism and methodology: Unity of conception / diversity of application// Proceedings of the 25th Winter Simulation Conference*. ACM Press, 1993. - 573–579 p.

[114] Zhang J. *Visualization for Information Retrieval*. The Informational Retrieval Series. - Berlin Heidelberg: Springer-Verlag, 2008. - 292 p.

[115] Zinoviev D. *Mapping DEVS models onto UML models// Proceedings of the 2005 DEVS Integrative M&S Symposium*. - San Diego, CA, 2005. - April. - 101–106 p.

[116] Zomorodian A.J. *Topology for Computing*. - Cambridge: Cambridge University Press, 2005. - 243 p.

[117] Бенкович Е., Колесов Ю., Сениченков Ю. *Практическое моделирование динамических систем*. - Санкт-Петербург: БХВ-Петербург, 2002. - 464 стр.

[118] Советов Б.Я., Яковлев С.А. *Моделирование систем*. - Москва: Высшая школа, 2001. - 343 стр.

[119] Бусленко, Н.П. *Моделирование сложных систем*. - Москва: Наука, 1978. - 400 стр.

[120] Растригин, Л.А. *Адаптация сложных систем: методы и приложения*. - Рига: Зинатне, 1981. - 375 стр.

[121] Новиков, С.П. *Топология*. - Москва-Ижевск: Институт компютерных исследований, 2002. - 336 стр.

[122] Карпов Ю. *Имитационное моделирование систем: введение в моделирование с* AnyLogic *5*. - Санкт-Петербург: БХВ-Петербург, 2005. - 400 стр.

[123] Дьяконов, В.П. MATLAB *6.5* SP*1/7* + Simulink *5/6 в математике и моделировании*. Библиотека профессионалов. - Москва: СОЛОН-Пресс, 2005. - 576 стр.