36th European Modeling & Simulation Symposium, 010
21th International Multidisciplinary Modeling & Simulation Multiconference

2724-0029 © 2024 The Authors.
doi: 10.46354/i3m.2024.emss.010

Easy Communication Environment for Heterogeneous and Distributed Simulation Models Design

Artis Aizstrauts[1] and Egils Ginters[2,*]

[1]Sociotechnical Systems OU, Sakala 7-2, Tallinn, 10141, Estonia
[2]Riga Technical University, Zunda Krastmala 10-522, Riga, LV-1048, Latvia

[*]Corresponding author. Email address: egils.ginters@rtu.lv

Abstract

The rapid and pervasive development of digital technologies not only affects but determines the life of society and the economy, as well as changing our living models. Today, society can be described as a complex socio-technical system, the model of which is heterogeneous, stochastic, and the results are therefore difficult to predict. To simulate the functioning of such a system, a distributed and heterogeneous model is needed. However, the designing of such simulation models is difficult due to the lack of easy-to-use communication tools that are not laborious and do not require specific software engineering knowledge from the modeler. The article explains the Easy Communication Environment (ECE) architecture and functionality that has been developed and validated over time, changing implementation stack but keeping the basic idea of designing a communication environment unchanged. The ECE methodology developed by the authors provides the modeler, who does not have specific skills in software engineering, the ability to design distributed and heterogeneous simulation models, as well as to ensure their interaction. The content of the article will be useful both for researchers and professionals in various fields who need to use simulation for the analysis of complex systems.

Keywords: Distributed simulation; Easy Communication Environment; High-Level Architecture; Simulation technologies

1. Introduction

Digital technologies have become the cornerstone of our society and economy. The quality of these technologies determines our well-being and living conditions. The market for digital technologies is growing at immeasurable speed and volume. There is a digital transformation of our lives, where digital technologies are changing not only industry, but also our behaviour and communication patterns. The penetration of artificial intelligence (AI) and development of Big Data analytics, cloud computing, mobile technologies, business intelligence and Internet of Things (IoT), as well as the global and pervasive impact of social media have a significant impact on our life.

The size of the global digital transformation market in 2023 was $880.28 billion, but it is expected that it reaches $1,070.43 billion in 2024. The market growth is characterized by a compound annual growth rate of about 27.6%. Projections suggest that the size of the digital transformation market will reach $4,617.78 billion by 2030 (Grand View Research, 2024).

The existence of society today can be described as a sociotechnical system, the sustainability of which depends on the development of the sustainability of digital technologies. The stochastic nature of digital technology, which includes a hidden set of influences (Ginters and Revathy, 2021), as well as a pervasive nature, has the most diverse and unexpected impacts on the development of society, the economy, and the environment, as well as on the daily life of everyone.

The number of resources available and explored by mankind is decreasing, climate changes are taking place, the life cycle of technologies is shortening, AI tools are creating both a positive effect and have a negative impact on society. The above determines the increasing relevance of the need for reasonable forecasts. In turn, the development of a reliable forecast and the validation of various scenarios require modeling of the problem.

The model of a sociotechnical system is a simplified reflection of objective reality, however, respecting the high proportion of stochastic influence factors, it remains complex. Complex problems cannot be explained by primitive and/or homogeneous patterns. The model of a real sociotechnical system is heterogeneous and usually requests for different simulation technologies use. If, for example, discrete-event simulation allows for good modeling of processes related to queues and delays, but it is better to use agent-based models (ABM) to study the behaviour and interaction of individual objects, then the overall changes will be better specified by system dynamics simulation equations. If also the sections of applied data processing and analysis must be added, then a heterogenous distributed modeling system must be designed and exploited.

In 2012, when conducting an analysis of distributed simulation platforms and tools, the authors (Aizstrauts et al., 2012) found that the number of options is very limited with Distributed Interactive Simulation (DIS), High-Level Architecture (HLA), Common Object Request Broker Architecture (CORBA), or customizable web service-based solutions, but there are practically no universal and easy to use tools for designing distributed simulation systems models.

And even today, nothing fundamental has changed. Separate sufficiently fast communication tools Robot Operating System (ROS) and Lightweight Communications and Marshalling (LCM) have emerged, which were created for real-time robotics applications, and the situation has forced modelers to use those to somehow compensate for the lack of tools needed to design distributed models. (Xu et al., 2021). Also, the development of IoT has contributed to the development of special device-device communication tools such as IoTivity (Xu et al., 2021). However, these tools serve poorly to provide heterogeneous model collaboration. A ray of hope was the protocol and standard Data Distribution Service (DDS) (OMG, 2015) created by Object Management Group for real-time distributed operational systems. However, these are only instructions and guidelines to achieve model interoperability, but to bring the recommendations to life, the author of each model will have to invest serious time and software engineering resources.

It must be honestly acknowledged that the still dominant tool to ensure the interaction of simulation models is the High-Level Architecture (HLA), which was created by the US Department of Defence back in 1995 (Dahmann, Fujimoto and Weatherly, 1997), but it is also supplemented, improved and widely used in the civilian sectors (Jabbour, Possik and Zacharewicz, 2023; Possik, 2021). In fact, HLA has become the standard solution for providing distributed models interoperability. One of the main drawbacks of HLA was and still is "*Standard is too "heavy", i.e. very complex, difficult to learn and thus time consuming to adopt and use*" (Strassburger, Schulze and Fujimoto, 2008), and without good skills in software engineering its use is impossible.

The use of the above-mentioned solutions requires well financial justification, since high level software engineering knowledge is untypical for representatives of other industries who are aimed at creating distributed problem-oriented models. So, for professionals in other industries, this path to creating a distributed model is still closed, or it is necessary to attract software engineering specialists who will have to translate the insights and regularities of the industry. So, the life cycle of this model will most likely end with a loss of interest from the software engineering specialist. Moreover, the licensing terms of commercial tools will impose an additional financial burden on further distribution of the designed model.

To address the above problem, simulation environment developers incorporate multiple simulation technologies at the same time, which makes it possible to construct heterogeneous models, such as AnyLogic (Borshchev, 2014). However, the question remains how to integrate models already created earlier from other simulation environments, such as FlexSim (Wang, Wang and Zhang, 2022), NetLogo (Wilensky and Rand, 2015) and STELLA (Dissanayake, 2016), etc., as well as provide a link to the subsystems of data analysis.

There is an urgent need for a tool that would enable modelers to make the previously impossible possible, and to allow industry professionals with basic knowledge in information technology and modeling to create distributed heterogenous simulation models themselves, without the involvement of coders and additional financial expenses.

If the modeler works with ABM, then he has at least a minimal understanding of object-oriented programming. If the modeler uses discrete-event tools, then he has knowledge of statistical processing of data and understands probabilities. If the modeler develops system dynamics models, then he is familiar with differential equations. That is, he has enough knowledge to create a distributed and heterogenous simulation model using Easy Communication Environment (ECE), which has been developed and improved since 2008 (Aizstrauts et al., 2012).

The aim of the article is to discuss the Easy Communication Environment (ECE) concept, which provides heterogeneous and distributed simulation models design capabilities to any specialist in other fields with appropriate knowledge in modeling.

The audience of the article is researchers and industry professionals engaged in the modeling of processes and phenomena, as well as designers of data processing systems who want to create distributed data processing and exchange systems. The authors' approach could also be of interest to designers of digital twins.

In Section 2, the authors review the development of ECE, starting with the first attempts to replace HLA with CORBA and similar solutions used in software engineering, and ending with the design of an HLA alternative. Section 3 of the article analyzes the structural architecture and functionality of ECE, while Section 4 examines a specific application example and various issues related to the use of ECE.

2. State of the art

In 2007, the authors conducted research on Ligatne Natural Trails, which is part of the Gauja National Park (Ginters and Silins, 2007). In a limited area, moose, deer, wild boar, wolves, bears, and other representatives of the national fauna live there in a natural environment. In order not to damage natural resources, the sightseeing places connect pedestrian tourist trails. However, for the convenience of lazy visitors, an asphalt road has been arranged, for which private cars can move along a certain route and order, since there is a parking lot next to each sight. The question of the research was, what is the permissible load on the natural resource, so that the damage caused by visitors does not lead to irreversible consequences and the resource is able to recover at a certain time interval?

It was necessary to create a discrete-event model that provides an analysis of the movement of cars to understand how large parking spaces should be allocated and how equipping them with WC and other should be carried out. The model was implemented in the Extend Suite environment and served as a data source for the second ABM model in the NetLogo environment, which assessed the regenerative capacity of the natural resource.

Then the question arose, how to ensure the interoperability of models? An analysis of existing distributed simulation support mechanisms was carried out. DIS and HLA were found to be usable. However, the implementation of an interoperability mechanism between the models in the HLA was laborious and complex. It became clear that designing distributed models using the tools HLA and similar is impossible without good knowledge of software engineering, and for nature researchers this path is closed.

The authors continued looking for distributed simulation communication tools that are usable to other industry professionals (Ginters, Silins and Andrusaitis, 2007).

An analysis of the suitability of the Aggregate Level Simulation Protocol (ALSP), CORBA, The Foundation for Physical Intelligent Agents (FIPA) and HLA was carried out. Special attention was paid to the CORBA mechanism, whose Object Request Broker (ORB) was used to ensure communication between Extend Suite and NetLogo models. Simulation environments extension modules C++ (Extend Suite) and Java (NetLogo) were implemented. The solution provided models communication, unfortunately, change management without serious software engineering knowledge was impossible.

In 2008, the adaptation of the CORBA ORB mechanism for distributed simulation continued (Ginters and Silins, 2008a), however, the low CORBA performance, which affected the modelling quality, began to cause problems.

The authors (Ginters and Silins, 2008b) returned to performance measurements and made comparisons between CORBA and HLA communication mechanisms. The suspicions were confirmed as the length of the message packets increased, significant HLA advantages over CORBA began. It was decided that the use of CORBA in distributed simulation is unsustainable.

In 2010, the authors published a solution (Silins, Ginters and Aizstrauta, 2010) that eased access to the HLA federation. The solution was based on the Communication Adapter, which provided a link between the simulation model and HLA. The adapter was created as part of the HLA environment and interpreted the HLA data in a format that was understandable by the simulation model. The communication adapter listened to the models and accumulated the sent messages in the data storage. Another model that needed the data could read them from the data storage on demand. At the same time, data storage provided time synchronization options. Each simulation environment required a library to interact with the adapter. Cooperation between the simulation tool and adapter was initiated by requests in XML format. In conformity with the model parameters the adapter generated the Federation Object Model (FOM), which allowed to connect to the HLA federation. However, this adapter should have been easy enough to manage since the parameters of the simulation models are variable. The implementation of this requirement turned out to be more complicated than it seemed at first glance, as it provoked corresponding changes in the chain of compatibility with HLA. Thus, the first version of Easy Communication Environment (ECE) was born.

In 2011, the improvement of the first version of ECE continued, working on supplementing the simulation tools Communication library. An intermediate

Communication Gateway was created that provided the data exchange among Communication Adapters prior to connection to the HLA environment (Aizstrauts, Ginters and Aizstrauta, 2011). The improvement reduced the modeler's workload by creating HLA-based distributed simulation models.

Since the first versions of the ECE were based on the presence of HLA, the search for more suitable communication environments, architectures, protocols, and message data formats continued (Ginters et al., 2011).

The authors attempted to apply the ideology of ISO OSI 7498 in distributed simulation, creating the Simulation Highway concept as a multi-layered open architecture. The Simulation Highway was formed by the set of simulation cells, but the distributed simulation task was accomplished by the inclusion of appropriate simulation cells in the task chain. The requirements for specific software engineering skills were eased, as high-level languages such as SimAL and SimQL were allowed for modeling requests in accordance with the client-server approach. However, the communication backbone was still the HLA. In addition, the implementation of Simulation Highway required very significant resources. After a self-assessment of the sustainability development of Simulation Highway, the authors abandoned this idea.

In 2012, the FP7 project No.287119 FUPOL was launched. In the framework of the project, the development of several heterogeneous simulators was carried out both for the design of the zoning of the public park in Zagreb, the forecasting of the occupancy of tourist facilities and the planning of the bicycle route network of the city of Skopje (Ginters et al., 2014).

The FUPOL project was the next step of ECE development (Aizstrauts et al., 2012), as it made it possible to validate the quality of the previously developed ECE version. To implement and test response time boundary intervals in a territorially distributed model, the ECE was deployed in a virtual computing environment of Amazon Elastic Compute Cloud (EC2), while simulation model management was done with Remote Desktop Protocol (RDP). In this version of the ECE and in the future, the authors abandoned the use of HLA, expanding the functionality of the previously designed Communication Gateway.

One of the main tasks of FUPOL project was to create a simulation-based support mechanism for policy decision-making. The consortium conducted an analysis of more than 60 generic and problem-oriented simulation tools, which confirmed earlier suspicions that most problem-oriented simulation environments are closed, and it is not possible to attach external models. Unfortunately, closed environments usually offer limited functionality that did not provide an implementation of heterogeneous distributed models.

The basic concept of the FUPOL simulator was developed (Aizstrauts et al., 2013), where Fuzzy Cognitive Maps and Colored Petri Networks were used at the higher level of design, while simulation models were implemented in RePast Symphony. However, STELLA, a system dynamics simulation environment, was used to simulate system changes over time. Data processing and analysis were implemented in Python. The data was stored in PostgreSQL base. To improve the visualization of the results and facilitate the management of the modeling sessions, the NetLogo environment was used, because the visualization mechanism of the RePast Symphony was rather poor. Data exchange in FUPOL simulation subsystem was provided by the ECE. To ensure an interoperability with other FUPOL systems an ECE alignment with the Enterprise Service Bus was designed. The FUPOL project provided excellent validation opportunities for the ECE concept.

One of FUPOL's use cases where the ECE environment was validated was the Skopje Bicycle Routes Simulator (Aizstrauts et al., 2015). Simulator was aimed to provide route choices for cyclists, depending on the load on the route, the quality of the pavement and infrastructure, as well as meteorological conditions. Here the ECE had to ensure interoperability between NetLogo, RePast and Python.

The analysis of several Message Brokers was carried out, recognizing as the most promising RabbitMQ, which was subsequently used as an ECE component. The Advanced Message Queuing Protocol (AMQP) was used by simulation model to communicate with the Message Broker. Henceforth the authors of the ECE abandoned the use of any clouds for communication needs, which did not provide a guaranteed response time and raised doubts about the timeline of the simulation session. The FUPOL validation results replenished the ECE set of requirements, and specifically, not only the convenience of the modeler is important, but also the universality of the communication environment.

In the following a prototype of the cycle route planning tool Velorouter was created (Aizstrauts et al., 2020). The research was initiated by previous FUPOL results and FP7 FLAG-ERA FuturICT 2.0 (2017-2020) project. Velorouter users had the ability to develop their own routes, as well as recommend them to the municipality, providing feedback through ticketing, which is a basic rule for the development of sustainable management and control.

Here, for the first time, the multi-layer architecture of the ECE was presented and the functionality of each layer was defined. Based on the universality and compatibility requirement, the ECE communication and data exchange mechanism was based on JSON data formats and the AMQP protocol disclaiming further use of XML.

The ECE versions were validated in several national and international research projects:

- No. 2006/11 “Simulation Tools EXTEND and NetLogo use for Ecosystems Analysis” funded by the Latvian Ministry of Education and Science.
- No. 2007/1-17/26 “Communication Environment of Hybrid Simulation Systems” funded by the Latvian Ministry of Education and Science.
- No.2DP/2.1.1.2.0/10/APIA/VIAA/001 “Support for preparation of IST FP7 STRE project “Simulation Highway”” funded by ERDF.
- FP7 No.287119 FUPOL „Future Policy Modelling” funded by European Commission.
- FP7 FLAG-ERA FuturICT 2.0 (2017-2020) "Large scale experiments and simulations for the second generation of FuturICT" funded by European Commission.

The research results were validated and used in the study course "Sociotechnical Systems Modeling" (Vidzeme University of Applied Sciences and Riga Technical University). Research results have been reported at 15 international conferences.

The ECE's multi-layer architecture and a description of the functionality will be discussed below.

3. Methodology

The structural and functional description of the Easy Communication Environment explains the basic principles of architecture and layer interoperability.

3.1. Structural Model of Easy Communication Environment

The Easy Communication Environment (ECE) is a versatile and user-friendly communication tool. The primary goal is to facilitate communication between diverse simulation models. Additionally, it aims to be user-friendly for distributed simulation model designers, even those with limited software engineering knowledge.

Easy Communication Environment's structural multi-layer model has an open system architecture based on the basic building principles of OSI ISO 7498 (Ginters et al., 2011) and Dijkstra machines (Dijkstra, 1968).

The ECE structural model (1) (see Figure 1) consists of four layers, each of which is responsible for specific and functional tasks - ECE framework messaging protocol (A), ECE framework notation layer (B), ECE framework messaging library (C) and Simulation sub-model (D).

ECE framework is determined by

$$ECE = \langle A, B, C, D \rangle \quad (1)$$

where

A – ECE framework messaging protocol. Messaging protocol consists of two functional parts – message transportation and message carrier format.

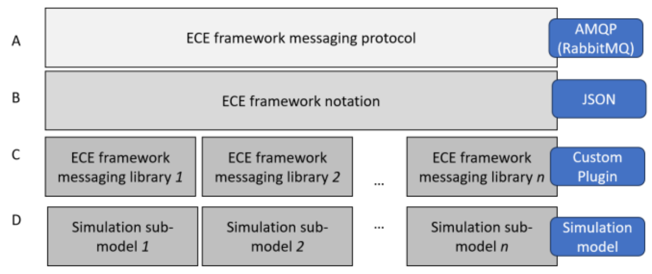


Figure 1. ECE structural model.

For message transportation ECE uses Advanced Message Queuing Protocol (AMQP) protocol. It is an open standard application layer protocol for message-oriented middleware. The defining features of AMQP are message orientation, queuing, routing (including point-to-point and publish-and-subscribe) and reliability (Naik, 2017). Reliability is one of the core features of AMQP, and it offers two preliminary levels of Quality of Service (QoS) for delivery of messages – “At most once” and “At least once”.

ECE uses “At most once” service level, the recipient does not acknowledge receipt of the message. Data storage functionality is implemented at ECE architecture abstraction layer C. For the message carrier ECE uses JSON data format.

ECE combines AMQP and JSON to deliver sophisticated communication architecture that is highly reliable and easy maintainable. AMQP ensures that message is delivered to the right client/subscriber (and resilience) and JSON allows to transmit complicated data that can be parsed and understood by most of the programming languages.

B – ECE framework notation layer. Notation layer is one of the highest levels in ECE communication architecture. This layer ensures that each node (sub-model) under-stands each other and speaks the same language. ECE notation is based on JavaScript Object Notation (JSON) data format and requires three fields for the message:

- *tick* - represents simulations time when this message has been sent.
- *datatype* – describes what type of data has been sent.
- *data* – the data to send to other simulation models.

C – ECE framework messaging library. Messaging library is custom built software that integrates into modelling toolkit, simulation suits, etc. as extensions or plugins. The abstraction layers of the ECE architecture are generic and open, so it is possible to develop such extensions/plugins for different simulation tools independently.

The ECE messaging library (C-layer) includes the basic functions that enable interoperability with the D-layer (see Table 1).

Table 1. ECE messaging library (C-layer).

Function	Description
init	Initialises the connection to AMQP server.
Close	Closes the connections to AMQP server.
isConnected	Returns the status of the connections to AMQP server.
Subscribe	Subscribes to communications topic. Will receive and store all the messages that are sent to that topic.
Unsubscribe	Unsubscribes from topic, will not receive any messages to this topic and purges all messages that were received before.
sendString	Sends String type message to specific topic.
sendInteger	Sends Integer type message to specific topic.
sendDouble	Sends Double type message to specific topic.
sendBoolean	Sends Boolean type message to specific topic.
getStringData	Returns String data for specific topic at specific tick.
getIntegerData	Returns Integer data for specific topic at specific tick.
getDoubleData	Returns Double data for specific topic at specific tick.
getBooleanData	Returns Boolean data for specific topic at specific tick.
getLastStringData	Returns String data for specific topic with the latest (largest) tick.
getLastIntegerData	Returns Integer data for specific topic with the latest (largest) tick.
getLastDoubleData	Returns Double data for specific topic with the latest (largest) tick.
getLastBooleanData	Returns Boolean data for specific topic with the latest (largest) tick.
getLastTick	Returns the latest (largest) tick for specific topic.

D – Simulation sub-model. Sub-model is any program or algorithm that consumes or produces any data. Such sub-model uses API of extensions/plugin from C-layer to send or receive data from other models. To ensure mutual interoperability of two or more simulation models (D-layer) and data exchange during the session, a chain of cooperation between ECE layers is established (see Figure 1).

The next chapter will explain the ECE communication algorithm in more detail.

3.2. Functionality and Implementation

The interoperability of the ECE A, B, C and D-layers is shown in the BPMN2 diagram (see Appendix A), demonstrating the collaboration between the two simulation models (D-layer). The processes occurring in each layer during the simulation session proceed in parallel. Parallel processes exchange messages and signals, and a process that occurs in one layer can initiate and/or affect another process in the other layer.

The ECE initialization operation is activated at the D-layer "Simulation sub-model" by sending a signal to the C-layer "ECE framework messaging library". If the model wants to send data, then the appropriate library in the C-layer is called. If the model does not have any data to be transmitted, and the simulation session is over, then a signal is sent that stops the plug-in of the model in the C-layer.

C-layer processes are activated by a signal that is received from the D-layer simulation model. Upon receiving a signal about the launch of the simulation session, the C-layer provides a connection to the AMQP server. If a data message is received from the D-layer, then a raw data object is created from it, which is transferred to the B-layer "ECE framework notation layer". Receiving a data object activates the B-layer, where the message is transformed in the JSON data format, and then the message is sent back to the C-layer. After that the message in JSON format is sent to the AMQP server on the A-layer "ECE framework messaging protocol". If the C-layer has not received a signal about the end of the simulation session, then the process of sending data continues, otherwise the connection to the AMQP server is interrupted. The A-layer is activated by data message received from the C-layer. The message is recorded to an AMQP storage queue corresponding to a specific data topic. At the same time, the AMQP server broadcasts a queued data message.

The sent message receives/process the model whose topic it is needed. In the C-layer of the other simulation model, the received AMQP message is converted to internal raw format and recognized, but then converted to JSON format again in B-layer and sent back to C-layer, where the message is stored in topic local storage. If the process is not interrupted by a signal received from the D-layer about the end of the simulation session, then it is divided in two directions - a request to read data from the D-layer is pending and a signal to subscribe to topic is expected at the same time. The data object in the internal format is then sent to the simulation model (D-layer), while the message in JSON format goes to the queues in the A-layer, which correspond to the relevant topic and are deployed in the AMQP storage.

4. Results and Discussion

The extract from FUPOL's cycling route planning heterogeneous and distributed simulation model, where communication is provided by the ECE, is discussed below.

The task of the distributed and heterogenous model is to simulate the load in different segments of the route, so that the cyclist can combine the route that suits him.

It is not the full model of Skopje network simulation, but its middleware, which is aimed to show the use of the ECE.

The model concept map (see Figure 2) includes simulator objects that provide the necessary functionality.

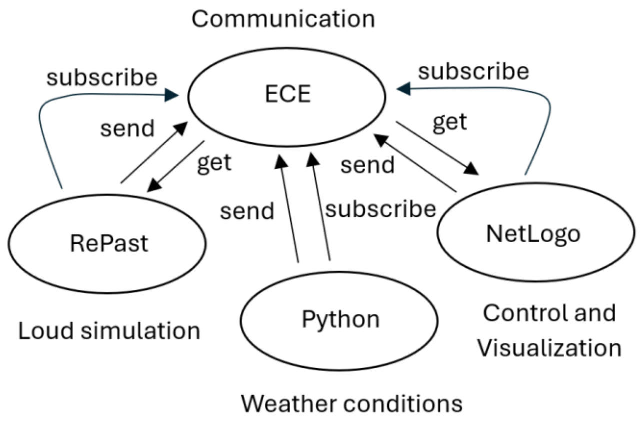


Figure 2. Simulator concept map.

Segments load simulation is provided by RePast agent-based model. Weather data is coming from the Python part, while simulator control and information visualization are implemented in the NetLogo environment.

All simulator sub-models are connected to the ECE, which provides communication and data exchange options. Models and requests to libraries make layer D of the ECE. Each model specification (D-layer) includes commands that ensure that notifications/data are sent and received (subscribe/send/get).

The task of the modeler is to embed the appropriate commands in the model specification. ECE parts A, B and C are not the subject of interest to the modeller.

Causal loop diagram (CLD) (see Figure 3) describes the concept of simulator functionality.

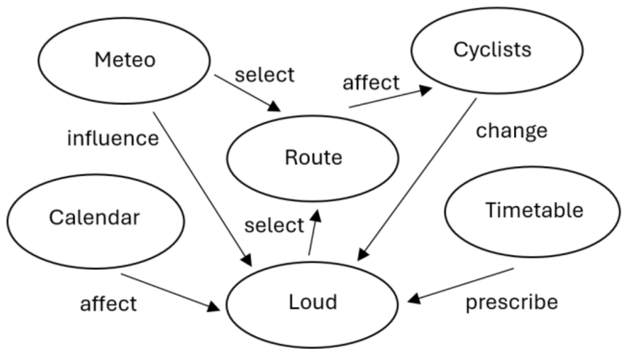
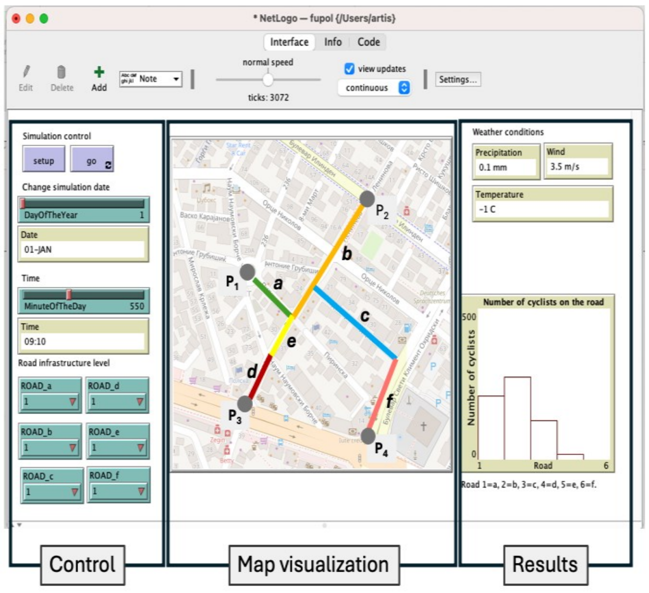


Figure 3. Causal loop diagram of route planning.

The simulator helps to the cyclist in selecting right segments of travel route. The selection of segment is influenced by meteorological conditions, which can impair the quality of the path surface. On the other hand, in bad weather, the load on routes decreases due to a decrease in the number of cyclists. Route load is affected by the season (winter, summer) and calendar (weekdays and holidays), during which the number of cyclists increases/decreases. The load affects the choice of a particular route segment, since the profiles of the cyclist are different and determined by the set of attributes: skills, type of bike, age, presence of children in the group, size of the group, etc. Each selection increases the number of cyclists on the segment, which in turn leads to an increase in load. The increase in loud reduces the probability of route selection in further iterations. The initial conditions of the simulation session are adjusted with real load data creating a balanced system.

Simulation workbench (see Figure 4) is implemented in NetLogo and ensures control and visualization options. The user interface has three compartments – Control, Map visualization and Results.



ticks: 3072

DayOfTheYear 1

Date 01-JAN

Time

MinuteOfTheDay 550

Time 09:10

ROAD_a 1
ROAD_d 1
ROAD_b 1
ROAD_e 1
ROAD_c 1
ROAD_f 1

Precipitation 0.1 mm

Wind 3.5 m/s

Temperature -1 C

Road 1=a, 2=b, 3=c, 4=d, 5=e, 6=f.

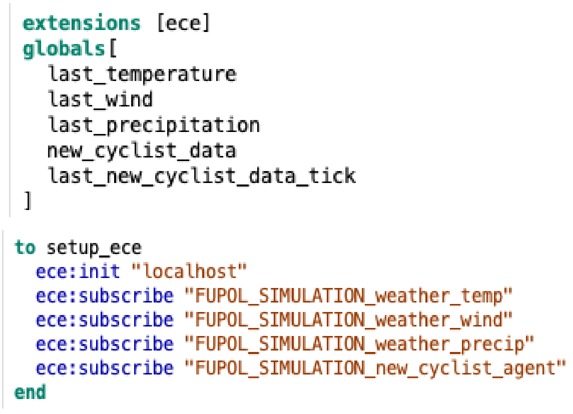
Figure 4. Simulation model workbench.

Setup button initializes model and ECE communication, that is, defines global variables and subscribe for ECE session. In this case all the global values will be received from other models via ECE. ECE Netlogo plugin instance (C-layer) will asynchronously receive data for each of these subscribed topics and store it locally with specific timestamp (tick from the message itself) (see Figure 5).

The C-layer libraries at the beginning of the simulation session must connect to ECE AMQP server. It is done with commands *ece: init*, in this case connection is to *localhost*.

Go button start/stop simulation process. Starting time and selected travel date are manually adjustable. Time is defined as minute of 24 hours, but date is consecutive day of year.

User can define the bicycle lane coverage and infrastructure quality with drop-down menus (ROAD_i). In this model, the user simulates the load on six segments of a possible route. The map shows four starting points/destinations (P_1, P_2, P_3 and P_4), bicycle lanes among these points are split into six edges or segments (a, b, c, d, e, and f). Lane load is simulated in RePast for each segment separately.



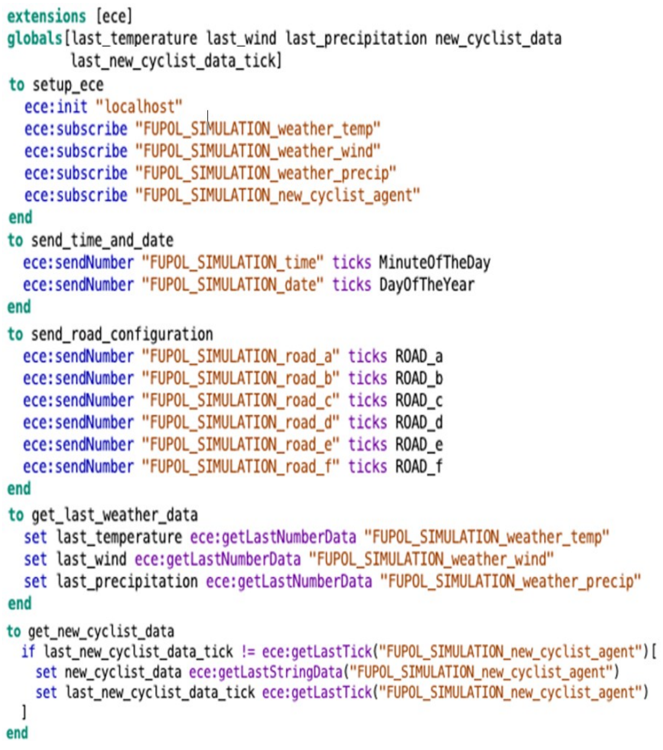
```
extensions [ece]
globals[
  last_temperature
  last_wind
  last_precipitation
  new_cyclist_data
  last_new_cyclist_data_tick
]
```

```
to setup_ece
  ece:init "localhost"
  ece:subscribe "FUPOL_SIMULATION_weather_temp"
  ece:subscribe "FUPOL_SIMULATION_weather_wind"
  ece:subscribe "FUPOL_SIMULATION_weather_precip"
  ece:subscribe "FUPOL_SIMULATION_new_cyclist_agent"
end
```

Figure 5. NetLogo Setup implementation.

The Results compartment displays the possible weather conditions provided by Python part for the selected day and time (precipitation, wind speed and temperature) and shows a bar chart representing the potential load for each segment simulated. According to the load results, the cyclist can decide to include this segment in his route or reject it.

NetLogo model software (D-layer), includes requests for C-layer libraries (see Figure 6).



```
extensions [ece]
globals[last_temperature last_wind last_precipitation new_cyclist_data
        last_new_cyclist_data_tick]
to setup_ece
  ece:init "localhost"
  ece:subscribe "FUPOL_SIMULATION_weather_temp"
  ece:subscribe "FUPOL_SIMULATION_weather_wind"
  ece:subscribe "FUPOL_SIMULATION_weather_precip"
  ece:subscribe "FUPOL_SIMULATION_new_cyclist_agent"
end
to send_time_and_date
  ece:sendNumber "FUPOL_SIMULATION_time" ticks MinuteOfTheDay
  ece:sendNumber "FUPOL_SIMULATION_date" ticks DayOfTheYear
end
to send_road_configuration
  ece:sendNumber "FUPOL_SIMULATION_road_a" ticks ROAD_a
  ece:sendNumber "FUPOL_SIMULATION_road_b" ticks ROAD_b
  ece:sendNumber "FUPOL_SIMULATION_road_c" ticks ROAD_c
  ece:sendNumber "FUPOL_SIMULATION_road_d" ticks ROAD_d
  ece:sendNumber "FUPOL_SIMULATION_road_e" ticks ROAD_e
  ece:sendNumber "FUPOL_SIMULATION_road_f" ticks ROAD_f
end
to get_last_weather_data
  set last_temperature ece:getLastNumberData "FUPOL_SIMULATION_weather_temp"
  set last_wind ece:getLastNumberData "FUPOL_SIMULATION_weather_wind"
  set last_precipitation ece:getLastNumberData "FUPOL_SIMULATION_weather_precip"
end
to get_new_cyclist_data
  if last_new_cyclist_data_tick != ece:getLastTick("FUPOL_SIMULATION_new_cyclist_agent")[
    set new_cyclist_data ece:getLastStringData("FUPOL_SIMULATION_new_cyclist_agent")
    set last_new_cyclist_data_tick ece:getLastTick("FUPOL_SIMULATION_new_cyclist_agent")
  ]
end
```

Figure 6. NetLogo model D-layer specification.

Weather simulation is developed in Python programming language. ECE library (C-layer) provides Python ECE class collection (D-layer) with functions to communicate via ECE framework. The main purpose of this model is to broadcast the weather conditions for travel date and time.

Load simulation is implemented in RePast Symphony according to a previously validated algorithm (Ginters, Aizstrauts et al., 2016). RePast C-layer library is designed in generic pattern and can be implemented in any Java application. The user's D-layer includes requests for C-layer libraries, such as NetLogo and Python.

And now a little about challenges and problems. In the example considered, to access the ECE, the task of the modeler is to include D-layer commands in the sub-model code, while the higher layers of the ECE are responsible for the rest of the communication. If the modeler has sufficient knowledge to prepare NetLogo, RePast, or other simulation model code, then he will have sufficient skills to incorporate the necessary ECE commands. The level of complexity is not even comparable to those software engineering skills required for HLA use.

The performance of the distributed model is determined by the model's interoperability algorithm designed by the modeler, while ECE provides only communication tube. The same can be said for HLA. The exciting question would be, which of the communication environments HLA or ECE has higher performance?

Since ECE is based on AMQP, which is one of the fastest protocols, the possible latency associated with ECE operation will be lower than in an HLA environment. AMQP is significantly faster than HTTP, so if the modeler deploys the simulations on the web environment, then ECE will not be the reason for the delay.

Is AMQP fast enough for real-time applications when a modeler creates digital twins for real-time control of IoT objects? IoT applications usually use more primitive protocols, such as Message Queue Telemetry Transport (MQTT), because shorter data messages are used to control the technical system. If ECE will be based on the MQTT protocol, then its use for heterogeneous systems design would be severely limited. However, the RabbitMQ stack also includes MQTT, so customizing ECE wouldn't be a problem.

In recent years, the term "digital twins" has become popular, which describes a simulation model working in parallel with the controlled object. The simulation model receives data from the external environment and simulates the operation of a real object. The peculiarity is that the result of modeling must be achieved in real time, otherwise an error is detected. This is nothing new, because successfully working simulation models that control real physical objects were tested back in the late 90s. For example, Alexander Verbraeck of Delft University of Technology demonstrated to the authors the Automatic Guided Vehicle (AGV), which transported flowers through underground tunnels between Schiphol Amsterdam and Flower Auction (Versteegt and Verbraeck, 2002). The CORBA environment was used. This was a very interesting example, since it was problematic in underground tunnels to provide control feedback, so the possible location of the AGV was calculated by a simulation model that worked in real time. Could the ECE be used here? Of course, because CORBA and ECE performance are similar, but the convenience of applying ECE is significantly better.

HLA is rarely used in real-time applications. The cornerstone of HLA is not maximum performance, but safety. HLA provides callback functions that reside in each federate, which increases safety and the load of computing resources, which in turn causes delays in communication. The ECE broadcasting mechanism is simpler and therefore faster.

ECE will provide the modeler with reasonable timeliness, scalability, modularity, safety, and performance parameters, however, if the modeler wants to design distributed simulation systems that support human life-critical functions, and programming in C++ does not cause any problems, then it would still be more reasonable to use the old, the well-tested, standardized, heavy, inconvenient, complex and expensive HLA mechanism.

Still, attempts to humanize HLA and replace it with Service Oriented Architecture (SOA) in critical simulation applications have failed (Iagaru, 2022).

5. Conclusions

The sociotechnical systems that permeate our society and industry are complex because they are influenced by many different stochastic factors, of which human effect is the most significant and least predictable. To check scenarios of the development of various situations and the possible consequences of our actions, simulation is used. Analytical solutions in conditions of high uncertainty and real-time, when the system evolves during a modeling session, are not sufficiently usable. Modeling of the behaviour and interaction of individual objects using analytical approach are especially difficult.

Complex processes and phenomena cannot be specified by homogeneous simulation models. Patterns become distributed and heterogeneous. A major problem is ensuring the exchange of data between these models and coordinating their operation over time.

This is amazing, but still distributed simulation uses HLA, which is not possible to modelers without specific knowledge and good skills in software engineering.

This is a problem that slows down adequate policy decision-making and analysis of various scenarios of development in any industry, as it becomes dependent on the coding skills of third parties. In addition, part of the model knowledge usually disappears in translation.

A second option remains, to use one of the multifunctional but closed simulation environments and hope that this environment will be sustainable and versatile, and with a long enough life cycle.

Over 17 years, the authors have developed and validated the Easy Communication Environment (ECE) data exchange environment for distributed and heterogeneous simulation, which is accessible to professionals in other industries without specific software engineering skills.

A stack of flexible protocols and data formats has been designed, the changes of which do not cause problems in running simulation models. Multi-layer open architecture provides interoperability of various simulation technologies.

The advantages of ECE are its application not only in the development of distributed simulation models, but also in the provision of software subsystems interoperability, so ECE audience is not only researchers and modelers, but also developers of distributed data processing systems.

When developing the ECE communication mechanism, the authors deliberately did not use message confirmation, which creates a delay in information exchange. Therefore, ECE is not recommended for simulating systems with specific safety requirements.

The concept and functionality of the ECE have been validated in several national and European Commission funded projects, while research results have been published in various sources and announced at international conferences.

For the time being, ECE lacks methodological and training materials, as well as not having a large enough library of simulation tools, however, the authors will continue to work on the preparation of these deliverables. The next step of ECE will be connecting heterogeneous simulation models to a Bayesian network to support cyber threat forecasting for individual nodes connected to the data transmission network.

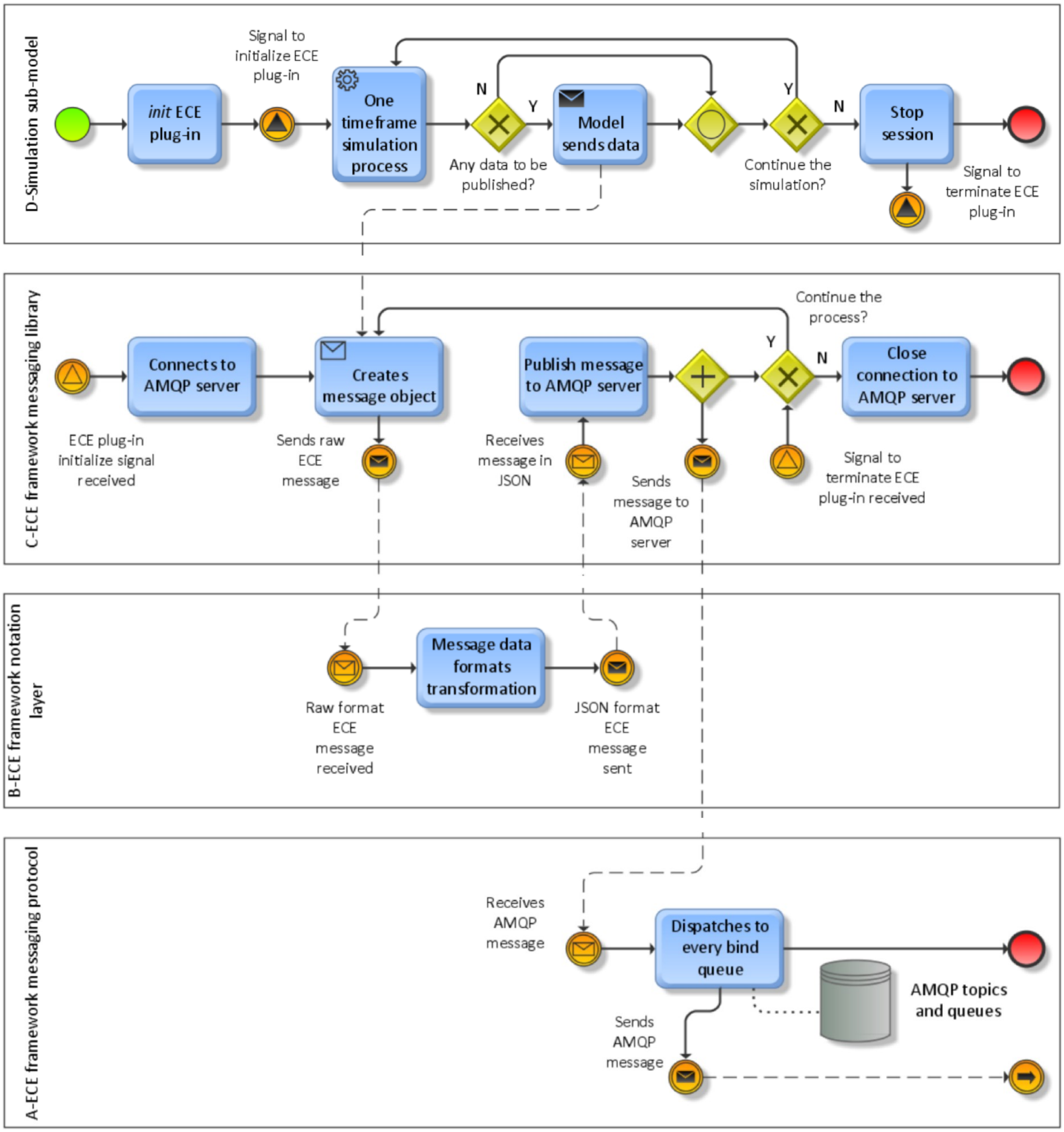
Acknowledgements

The authors would like to thank Vidzeme University of Applied Sciences and Information Technology Institute of Riga Technical University for their support during the development of ECE research.

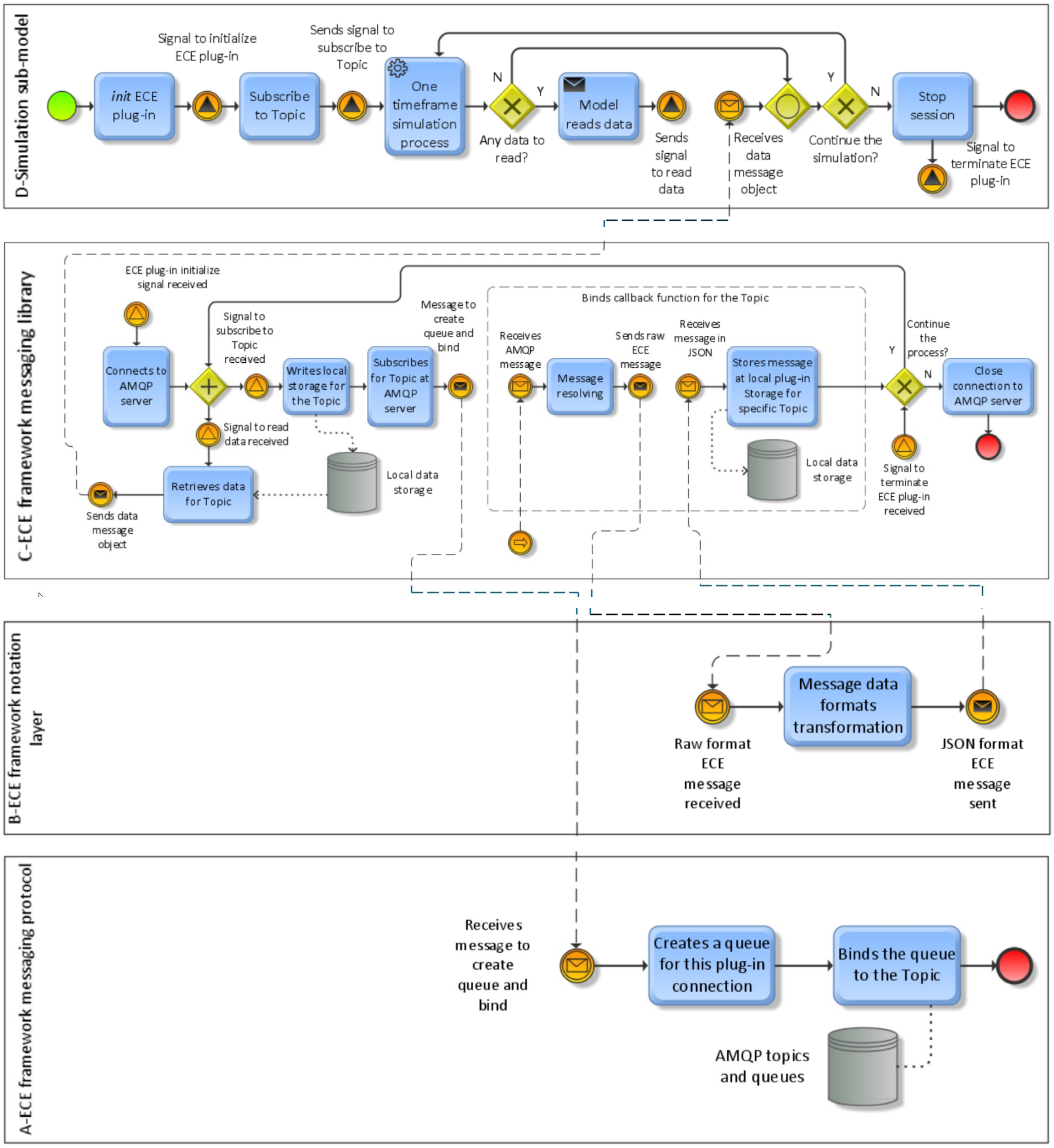
The development of the ECE is the longest project in the simulation life, so the authors would like to thank not only their colleagues who endured the never-ending improvements and modifications of the ECE, but also their families, who reconciled and allowed them to devote time to ECE research.

Appendix A. BPMN2 diagram of ECE layer interactions

The diagram describes the send/receive operations between two simulation models, looking at the macro-level interoperability of the ECE layers implemented at both ends of one and the other model.


D-Simulation sub-model
init ECE plug-in
Signal to initialize ECE plug-in
One timeframe simulation process
N
Y
Any data to be published?
Model sends data
Y
N
Continue the simulation?
Stop session
Signal to terminate ECE plug-in
C-ECE framework messaging library
Connects to AMQP server
ECE plug-in initialize signal received
Creates message object
Sends raw ECE message
Publish message to AMQP server
Receives message in JSON
Continue the process?
Y
N
Sends message to AMQP server
Close connection to AMQP server
Signal to terminate ECE plug-in received
B-ECE framework notation layer
Raw format ECE message received
Message data formats transformation
JSON format ECE message sent
A-ECE framework messaging protocol
Receives AMQP message
Dispatches to every bind queue
AMQP topics and queues
Sends AMQP message

Appendix A. BPMN2 diagram of ECE layer interactions (continuation)


D-Simulation sub-model
Signal to initialize ECE plug-in
init ECE plug-in
Subscribe to Topic
Sends signal to subscribe to Topic
One timeframe simulation process
N
Y
Any data to read?
Model reads data
Sends signal to read data
Receives data message object
Continue the simulation?
Y
N
Stop session
Signal to terminate ECE plug-in
C-ECE framework messaging library
ECE plug-in initialize signal received
Connects to AMQP server
Signal to subscribe to Topic received
Writes local storage for the Topic
Subscribes for Topic at AMQP server
Message to create queue and bind
Signal to read data received
Local data storage
Retrieves data for Topic
Sends data message object
Binds callback function for the Topic
Receives AMQP message
Message resolving
Sends raw ECE message
Receives message in JSON
Stores message at local plug-in Storage for specific Topic
Local data storage
Continue the process?
Y
N
Close connection to AMQP server
Signal to terminate ECE plug-in received
B-ECE framework notation layer
Message data formats transformation
Raw format ECE message received
JSON format ECE message sent
A-ECE framework messaging protocol
Receives message to create queue and bind
Creates a queue for this plug-in connection
Binds the queue to the Topic
AMQP topics and queues

References

Aizstrauts, A., Burkhardt, D., Ginters, E., & Nazemi, K. (2020). On Microservice Architecture Based Communication Environment for Cycling Map Developing and Maintenance Simulator. In *2020 61st International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS)* (pp. 1-4). doi: 10.1109/ITMS51158.2020.9259299. Retrieved from https://www.researchgate.net/publication/347072223_On_Microservice_Architecture_Based_Communication_Environment_for_Cycling_Map_Developing_and_Maintenance_Simulator#fullTextFileContent

Aizstrauts, A., Ginters, E. & Aizstrauta, D. (2011). Easy Communication Approach for Data Exchange in Distributed Simulation Environment. In *Proceedings of the 13th WSEAS international conference on Automatic control, modelling & simulation (ACMOS'11)* (pp. 34-38). Stevens Point: World Scientific and Engineering Academy and Society (WSEAS). Retrieved from http://www.wseas.us/e-library/conferences/2011/Lanzarote/ACMOS/ACMOS-04.pdf

Aizstrauts, A., Ginters, E., Aizstrauta, D., & Sonntagbauer, P. (2012). Easy Communication Environment on the Cloud as Distributed Simulation Infrastructure. In *Proceedings of the 5th WSEAS congress on Applied Computing conference, and Proceedings of the 1st international conference on Biologically Inspired Computation (BICA'12)* (pp. 173-178). Stevens Point, Wisconsin: World Scientific and Engineering Academy and Society (WSEAS). Retrieved from http://www.wseas.us/e-library/conferences/2012/Algarve/BICA/BICA-29.pdf.

Aizstrauts, A., Ginters, E., Baltruks, M., & Gusev, M. (2015). Architecture for Distributed Simulation Environment. *Procedia Computer Science*, *43*, pp. 18-25. doi: 10.1016/j.procs.2014.12.004. Retrieved from https://www.sciencedirect.com/science/article/pii/S1877050914015725

Aizstrauts, A., Ginters, E., Lauberte, I., & Piera Eroles, M.A. (2013). Multi-level Architecture on Web Services Based Policy Domain Use Cases Simulator. In *Lecture Notes in Business Information Processing* (vol 153, pp.130-146). Berlin: Springer. doi:10.1007/978-3-642-41638-5_9. Retrieved from https://www.researchgate.net/publication/282289690_Multi-level_Architecture_on_Web_Services_Based_Policy_Domain_Use_Cases_Simulator#fullTextFileContent

Borshchev, A. (2014). Multi-method modelling: AnyLogic. In *Discrete-Event Simulation and System Dynamics for Management Decision Making* (eds S. Brailsford, L. Churilov and B. Dangerfield). doi: 10.1002/9781118762745.ch12

Dahmann, J.S., Fujimoto, R.M., & Weatherly, R.M. (1997). The Department of Defense High Level Architecture. In *Proceedings of the 29th conference on Winter simulation (WSC '97)* (pp. 142-149). IEEE Computer Society. doi: 10.1145/268437.268465

Dijkstra, E.W. (1968). The Structure of the "THE" - Multiprogramming System. *Commun. ACM*, *11*, 342–346.

Dissanayake, S. T. M. (2016). Using STELLA Simulation Models to Teach Natural Resource Economics. *The Journal of Economic Education*, 47(1), 40–48. doi: 10.1080/00220485.2015.1106358

Ginters, E., Aizstrauts, A., Baltruks, M., Merkuryev, Y., Novickis, L., Grundspenkis, J., & Grabis, J. (2016). VeloRouter - Technology for Urban Transport Intermodal Sustainability. In *City Planning and Urban Design Conference, 07-09 April (CPUD'16)* (pp. 211-220). Istanbul: DAKAM. ISBN 978-605-9207-21-8. Retrieved from https://www.dakamconferences.org/_files/ugd/bac820_3252301ff80b453e928fb159e78a9910.pdf

Ginters, E., Aizstrauts, A., Dreija, G., Ablazevica, M., Stepucev, S., Sakne, I., Baltruks, M., Piera Eroles, M.-A., Buil, R., Gusev, M., & Velkoski, G. (2014). Skopje Bicycle Inter-modality Simulator – e-Involvement Through Simulation and Ticketing. In *Proceedings of 26th Europen Modelling & Simulation Symposium (EMSS 2014)* (pp. 557-563). Retrieved from https://www.msc-les.org/proceedings/emss/emss2014/emss2014_557.html

Ginters, E., Sakne, I., Lauberte, I., Aizstrauts, A., Dreija, G., Aquilar Chinea, R.-M., Merkuryev, Y., Novitsky, L., & Grundspenkis, J. (2011). Simulation Highway – Direct Access Intelligent Cloud Simulator. In *Proceedings of 23rd European Modelling & Simulation Symposium (EMSS 2011)* (pp. 62-72). Retrieved from https://docplayer.net/16307931-Simulation-highway-direct-access-intelligent-cloud-simulator.html

Ginters, E., & Silins, A. (2007). Multi-level approach for environmental systems modelling in the Ligatne Natural Trails. *WSEAS Transactions on Systems*, 6(4), 795-801. Retrieved from https://www.researchgate.net/publication/297118169_Multi-level_approach_for_environmental_systems_modelling_in_the_Ligatne_Natural_Trails

Ginters, E., & Silins, A. (2008a). Exchange Mechanisms in Distributed Simulation of Sociotechnical Systems. In *Information Society and Modern Business. Proceedings of 3rd International conference, 21-22*

September 2007, Ventspils (p. 362). Ventspils: Ventspils University College. Retrieved from https://daugavpils.biblioteka.lv/alisepac/Details?title=Information-Society-and-Modern-Business.-3rd-International-conference-%3A-proceedings-21-22-September,-2007,-Ventspils&Id=85013&Ident=1218592&InstanceId=13&LibraryId=0#bibliographic

Ginters, E., & Silins, A. (2008b). Simulation Data Exchange in Distributed E-learning Environment. In *Proceedings of the 4th WSEAS/IASME International Conference on Education Technologies (EDUTE'08)* (pp. 138-143). Corfu: WSEAS. Retrieved from http://www.wseas.us/e-library/conferences/2008/corfu/edute/edute23.pdf

Ginters, E., Silins, A., & Andrusaitis, J. (2007). Communication in Distributed Simulation Environment. In *ICOSSSE'07: Proceedings of the 6th WSEAS international conference on System science and simulation in engineering* (pp. 217-221). Stevens Point: World Scientific and Engineering Academy and Society. doi: 10.5555/1974442.1974475 Retrieved from http://www.wseas.us/e-library/conferences/2007venice/papers/600-139.pdf

Ginters, E., & Revathy, J. G. (2021). Hidden and Latent Factors' Influence on Digital Technology Sustainability Development. *Mathematics,* 9(21), 2801. doi: 10.3390/math9212801

Grand View Research,. (2024). *Digital Transformation Market Size, Share, Growth & Trends Analysis Report by Solution, By Deployment, By Service, By Enterprise Size, By End-use, By Region, And Segment Forecasts, 2024 – 2030. Report ID: GVR-1-68038-851-0.* Retrieved from https://www.grandviewresearch.com/industry-analysis/digital-transformation-market

Iagăru, E. (2022). Comparative Analysis Between High Level Architecture (HLA) and Service Oriented Architecture (SOA) in the Field of Military Modelling and Simulation. *Scientific Bulletin,* 27(1), pp. 30 - 40. doi: 10.2478/bsaft-2022-0004

Jabbour, J., Possik, J., & Zacharewicz, G. (2023). A Modeling and Distributed Simulation Platform for Immersive Experience. In *JIAE 2023 - Journées sur l'Interopérabilité des Applications d'Entreprise, Pôle Grand Sud-Ouest (PGSO)*. Lille, France. Retrieved from https://hal.science/hal-04199808/document

Naik, N. (2017). Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP. In *2017 IEEE International Systems Engineering Symposium (ISSE)* (pp. 1-7). IEEE. doi: 10.1109/SysEng.2017.8088251

OMG (Object Management Group),. (2015). *About the Data Distribution Service Specification Version 1.4*. Retrieved from https://www.omg.org/spec/DDS/1.4

Possik, J., Gorecki, S., Asgary, A., Solis, A.O., Zacharewicz, G., Tofighi, M., Shafiee, M.A., Merchant, A.A., Aarabi, M., Guimaraes, A., & Nadri, N. (2021). A Distributed Simulation Approach to Integrate AnyLogic and Unity for Virtual Reality Applications: Case of COVID-19 Modelling and Training in a Dialysis Unit. In *2021 IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)* (pp. 1-7). Valencia: IEEE.

Silins, A., Ginters, E., & Aizstrauta, D. (2010). Easy Communication Environment for Distributed Simulation. In *Computational Intelligence in Business and Economics* (pp. 91-98). https://doi.org/10.1142/9789814324441_0014. Retrieved from https://www.researchgate.net/publication/282288675_Easy_Communication_Environment_for_Distributed_Simulation#fullTextFileContent

Strassburger, S., Schulze, T., & Fujimoto, R. (2008). Future Trends in Distributed Simulation and Distributed Virtual Environments: Results of a Peer Study. In *2008 Winter Simulation Conference* (pp. 777-785). Miami, FL, USA. doi: 10.1109/WSC.2008.4736140. Retrieved from https://ieeexplore-ieee-org.resursi.rtu.lv/stamp/stamp.jsp?tp=&arnumber=4736140

Versteegt, C., & Verbraeck, A. (2002). The Extended Use of Simulation in Evaluating Real-Time Control Systems of AGVs and Automated Material Handling Systems. In *Proceedings of the 2002 Winter Simulation Conference* (pp. 1659-1666). San Diego. doi: 10.1109/WSC.2002.1166448

Xu, L., Lin, SY., Hlynka, A.W. *et al.* (2021). Distributed Simulation Platforms and Data Passing Tools for Natural Hazards Engineering: Reviews, Limitations, and Recommendations. *Int J Disaster Risk Sci*, *12*, 617–634. doi: 10.1007/s13753-021-00361-7

Wang, S., Wang, S.M., & Zhang, N. (2022). Flexsim-based Simulation and Optimization of Green Logistics Distribution Center. In *Proceedings of the 14th International Conference on Computer Modeling and Simulation (ICCMS '22)* (pp.76-82). New York: Association for Computing Machinery. doi: 10.1145/3547578.3547590

Wilensky, U., & Rand, W. (2015). An Introduction to Agent-Based Modeling. Modeling Natural, Social, and Engineered Complex Systems with NetLogo. MIT Press.