Simulation Modelling Practice and Theory 135 (2024) 102977

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

An HLA-based automated approach for the interoperable simulation of collaborative business processes[☆]

Mariane El Kassis[a], Paolo Bocciarelli[b,*], François Trousset[a], Nicolas Daclin[a], Andrea D'Ambrogio[b], Gregory Zacharewicz[a]

[a] *Laboratory for the Science of Risks, IMT Mines Alès, 6 Avenue de Clavières, Alès, 30100, France*
[b] *Department of Enterprise Engineering, University of Rome Tor Vergata, Via del Politecnico 1, Rome, 00175, Italy*

ARTICLE INFO

Keywords:
Modeling & simulation (M&S)
Business process
Collaborative simulation
Distributed simulation
HLA
Model-driven Architecture (MDA)
Model-driven Engineering (MDE)

ABSTRACT

As collaborative simulations gain prominence, there is a pressing need for methodologies that seamlessly integrate interoperability while safeguarding intellectual property. This paper presents a novel approach rooted in Model-Driven Architecture and combined with the High-Level Architecture standard. This approach expedites the simulation process by automating the generation of Federation Object Model files and federate codes. Distinctly, we produce two federates: one exclusively in Java and another integrating Java with the Discrete Event System Specification, addressing diverse simulation paradigms. Utilizing the Unified Modeling Language and the Business Process Model and Notation standards, we devise a systematic procedure for modeling business operations while maintaining confidentiality. While our automation framework is robust, certain intricacies of federate behavior necessitate manual adjustments to ensure secure data transmission and protection of proprietary knowledge. The efficacy of our approach, striking a balance between interoperability and confidentiality in High-Level Architecture-based simulations, is demonstrated through a comprehensive experiment.

1. Introduction

Modern business processes operate within a complex network of interconnected subsystems, each contributing to the overall operational efficiency [1]. Business process simulation is crucial for navigating this network, as it helps in understanding system dynamics and identifying areas of improvement [2]. However, the challenge arises in enabling collaborative simulation while maintaining the confidentiality of intellectual property [3].

In this context, we employ the High-Level Architecture (HLA) framework to foster a collaborative simulation environment where entities with unique models can communicate and operate in a unified setting while protecting their intellectual property. This approach promotes heterogeneity and autonomy, allowing each collaborator to simulate their models independently using their preferred methods and fostering a nurturing ground for innovation and diversity [4].

As we delve deeper into the mechanisms facilitating collaborative simulations, our approach leverages the Model-Driven Architecture (MDA) methodology coupled with the HLA standard [5]. This strategy automates the generation of federates and Federation Object Model (FOM) files, bestowing each participant with autonomy during the simulation. We further integrate Unified Modeling Language (UML) and Business Process Model and Notation (BPMN) standards to draft a UML conceptual model

☆ This document is the results of the research project funded by "Région Occitanie", France.

* Corresponding author.

E-mail addresses: mariane.el-kassis@mines-ales.fr (M. El Kassis), paolo.bocciarelli@uniroma2.it (P. Bocciarelli), francois.trousset@mines-ales.fr (F. Trousset), nicolas.daclin@mines-ales.fr (N. Daclin), dambro@uniroma2.it (A. D'Ambrogio), Gregory.Zacharewicz@mines-ales.fr (G. Zacharewicz).

https://doi.org/10.1016/j.simpat.2024.102977
Received 24 October 2023; Received in revised form 15 March 2024; Accepted 3 June 2024
Available online 4 June 2024

of the federation, and apply Model-to-Text transformation to create the FOM and initial BPMN collaboration model, preserving the confidentiality of individual business processes.

Despite the substantial automation achieved, the refined delineation of federate behavior necessitates manual intervention in the generated federate code. This endeavor underscores secure and restricted information exchange, fostering interoperability without divulging detailed insights into their proprietary business processes [6]. By adhering to a low-code development paradigm, our methodology markedly diminishes the manual labor traditionally required in HLA-based simulation of BPMN-guided process collaborations.

To validate our strategy, we undertake an experiment delineating a UML conceptual model of the federation and automating the generation of the FOM and preliminary BPMN collaboration model. This process culminates in a simulation execution, wherein each participant adopts a bespoke simulation methodology embodied in the generated federate code.

The ensuing sections are organized as follows: Section 2 positions this study within existing literature and furnishes the requisite background. Section 3 elaborates on the proposed methodology for distributed simulation of business process (BP) collaborations adhering to the BPMN standard, discussing underlying assumptions and the creation process of JAVA-based and Discrete Event System Specification (DEVS)-based federates. Section 4 details a case study involving two research institutions who independently develop federate codes for a collaborative simulation experiment, demonstrating the interoperation under the HLA standard. Finally, Section 5 recapitulates the central contributions and delineates avenues for future exploration.

This manuscript serves as a comprehensive guide to our strategy for simplifying the distributed simulation of BP collaborations, guided by the BPMN standard, illustrating the journey from fundamental concepts to detailed developmental processes.

2. Research positioning and related works

As outlined in Section 1 this paper investigates methods and approaches for easing the development of distributed simulations. The proposed contribution specifically deals with the simulation of independent and heterogeneous entities that operate separately and interact through a message exchange. In this respect, the method introduced in this work along with the application scenario adopted for its experimental evaluation, builds upon and extends authors' previous contributions:

- The idea of introducing a transformation-based approach to support the development of HLA-based simulations has been investigated in [7]. Specifically, such a contribution introduces a model-to-text transformation to automatically generate the Federation Object Model from a conceptual model of the federation under study specified by the use of a UML Class Diagram. The aforementioned contribution also introduces a preliminary version of the *FOM-HLA Profile*, an HLA-oriented UML Profile which allows the identification of HLA-related concepts and entities in a federation abstract model, making it able to provide the information required by the model transformation. As clarified in Section 3, the proposed method exploits the above-mentioned transformation for generating the required FOM. Moreover, this work reuses and extends the *FOM-HLA Profile*.
- The adoption of model-driven techniques for reducing the implementation effort of Java-based distributed simulations systems has been proposed in [8]. Such a contribution deals with the distributed simulation of BPMN Collaborations and introduces a model-transformation approach for supporting the generation of a Java-based discrete event simulation system. The proposed method has also been applied to a simple yet effective concrete case to investigate its feasibility. In this respect, this work goes beyond our previous work and specifically focuses on the distributed simulation of heterogeneous and independent federates. Furthermore, this work extensively discusses the results of an experimentation of the proposed method.
- In the study [9], we explored accurate resource allocation within business process simulations, proposing a metamodel-based transformation approach for extending BPMN models with Business Process Simulation Interchange (BPSIM) elements into DEVS simulation models. This approach was directly applied in the example experiment discussed in Section 4, where we created a resource allocator Broker based on the transformation mapping delineated in this paper. This facilitated improved decision-making and coordination in business process management, proving instrumental in the development of the DEVS-based federate.
- The incremental transformation methodology introduced in [10] advances the process of transforming BPMN models enriched with BPSIM elements into DEVS simulations. This methodology innovatively employs an intermediate modeling layer and executes a dual-phase transformation, considerably streamlining the transformation process. Elaborated in Section 3.5, this approach not only simplifies the transition from modeling to simulation but also plays a pivotal role in facilitating the efficient integration of complex BPMN models into the DEVS formalism. By evolving from the foundational work detailed in prior studies to implementing this streamlined process, we demonstrate our commitment to advancing the efficiency and effectiveness of DEVS-based federate development, making it more accessible and practical for a wider range of simulation scenarios.

To the best of our knowledge, no contributions have been proposed which demonstrate how model-driven technologies can be effectively used in practice to support the development of a distributed simulation based on heterogeneous and independent federates.

In order to better outline the *state-of-the-art* and underline the novelty of this contribution, this section reviews existing literature in the Distributed Simulation (DS) field, specifically with regards to (i) the adoption of approaches to ease the DS development and, (ii) the use of HLA for simulating the interaction of independent entities which cooperates throughout an exchange of messages.

The idea of introducing approaches and technologies to mitigate the difficulties of developing software systems is not new and has been largely investigated in the past. Recently, the so-called *low-code* development paradigm which promotes the adoption of automated techniques for supporting the implementation of software systems has gained a considerable attention in the software engineering field [11–13]. The adoption of automation for easing the development of a DS implementation is a pillar of the proposed method which strongly relies on principles introduced in the Model-driven Engineering field and makes use of methods, technologies and standards provided by the Model Driven Architecture [14]. In this respect, as underlined in [8], MDA-based model-transformations can be effectively introduced as a part of a concrete approach founded on low-code paradigm principles. Moreover, various contributions such as [15,16] underline the added value provided by Model-driven Engineering principles and standards in developing DS systems.

In addition to the authors' past work previously mentioned, which constitutes the conceptual foundation of this paper, various previous contributions dealing with HLA and BPM have influenced the proposed research. In [17], a BPMN- and HLA-based method to integrate discrete event simulation components has been introduced, while in [18,19], a framework to generate simulation code from BPMN models by use of automated model transformations is discussed. The latter exploits the Java-based simulation platform introduced in [20,21] that supports both the execution of sequential simulations and HLA-based distributed simulations. In [22], the use of BPMN for DS development is investigated. In such an approach BPMN collaborations are used to describe the behavioral view of an HLA federation.

This work starts from results achieved by such contributions but also goes significantly beyond. First, the proposed approach, in a low-code perspective, encompasses all the activities required for developing an HLA-based DS and takes into consideration the development of both the FOM and substantial portions of federates. Specifically, the latter includes the code required to invoke the HLA Run-Time Infrastructure (RTI) services and handle the related *callbacks*. Moreover, this research takes into consideration a realistic scenario in which federation participants are required to only agree on a shared data model, with no assumptions, restrictions or constraints on technologies adopted for each federate's implementation.

As regards modeling of HLA federations, in [23], the lack of modeling formalisms specifically addressing the behavioral description of HLA federates is underlined. In addition, in [24], authors highlight the lack of established practices for dealing with behavioral models in DS. In this respect, the model of the HLA federation adopted in this paper approach makes use of both a UML Class Diagram, to specify the information model describing federate participants along with the exchanged data, and a BPMN Collaboration, to provide the behavioral description.

Finally, it should be underlined that the BPMN standard does not provide primitives for data modeling, as also underlined by the Object Management Group (OMG) itself, which states that this is not a BPMN 2.0 objective [25]. In order to specify the data exchanged by federates representing participants of a process collaboration, this work exploits a BPMN extension introduced in [8] and briefly outlined in Section 3.3.

3. Method for the distributed simulation of BP collaborations

This section illustrates the proposed method for easing the distributed simulation of BP Collaboration specified by use of the BPMN standard. The method hereby outlined starts from and significantly extends previous contributions, specifically [7] which focuses on the generation of the FOM starting from a federation model, and [8] where a complete low-code development process has been introduced to generate a JAVA-based federation. In this work the method has been considerably revised and extended in order to address the concrete case briefly introduced in Section 1 and characterized by the following assumptions:

- the development of a DS system is a challenging task often undertaken by different partners which might be interested in simulating a common scenario, where each partner is responsible for developing its own federate. The proposed approach does not assume or impose any restriction on the number of participants. In this respect, Section 4 discusses a concrete case which deals with the design and implementation of a DS, whose development involves two entities: the Institut Mines-Télécom Mines Ales (IMT) and the University of Tor Vergata (UTV);
- partners shall agree on the information to be exchanged during the simulation execution. In this respect, they cooperate to define a common data model;
- each partner might not be interested in sharing any detail about the internal behavior/design of their federate(s), making only available data and capabilities that each federate provides to and expects from other federates;
- each partner develops its own federate according to the agreed data model, and make the federate available in a distributed infrastructure. No assumptions are made on technologies upon which each federate implementation is based;

In this respect, Section 3.1 provides an overview of the method by illustrating the various activities which have to be undertaken in order to simulate a BP Collaboration. Section 3.2 outlines a UML profile introduced for annotating the Class Diagram which provides a conceptual model of the federation under study, Section 3.3 introduces an extension of the BPMN metamodel which the proposed method exploits to specify the BPMN Collaboration model describing the interaction exchanged by federates during the simulation execution. Finally, Sections 3.4 and 3.5 gives insight into the development process of the JAVA- and DEVS-based federate, respectively.

3.1. Overview

The development process at the basis of the proposed method for the distributed simulation of BP collaboration is shown in Fig. 1.

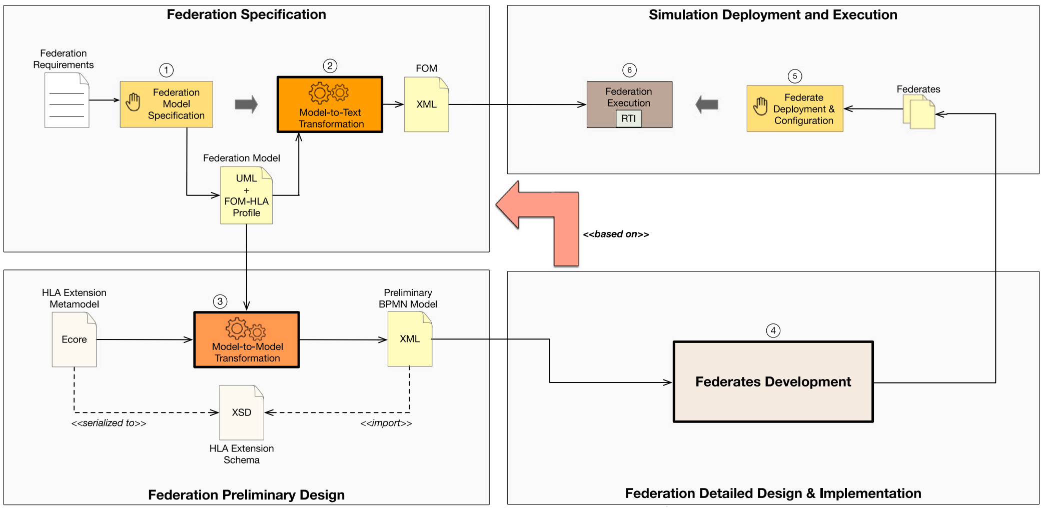


Fig. 1. Development process of the distributed simulation.

The proposed approach makes use of two different models: a UML Class Diagram for specifying a conceptual model of the federation under study that also provides a description of the adopted data model and a BPMN Collaboration for defining the flow of messages that the RTI and the involved federates exchange during the simulation execution. As these two artifacts constitute the input of automated model transformations that support the implementation of the simulation code, the information they shall provide constitute an essential aspect to be addressed. In this respect, the BPMN Collaboration describes the data model entity that is transferred in each message exchanged between a federate and the RTI. Unfortunately, the BPMN metamodel does not provide metaclasses to model the structure and details of data exchanged in the business process. Therefore, it is necessary to introduce a BPMN extension to enrich the BPMN semantics, allowing the specification of messages that the federate under study exchanges with the RTI.

The proposed method is inspired by the conceptual pillars of the Distributed Simulation Engineering and Execution Process [26], which introduces a standardized process for managing distributed simulations, and exploits *model-driven* standards and technologies to ease the development effort. Model transformations are introduced to generate preliminary models and a significant portion of the required HLA implementation code, written in Java. Specifically, the method makes use of the following software components:

- the *FOM-HLA UML Profile*, introduced in [7], which allows the annotation of the UML Federation Model to map HLA-related concepts to UML model elements. The FOM-HLA Profile, which has been revised and extended in this work, is illustrated in Section 3.2.
- a BPMN extension, to enrich the BPMN Collaboration model with details about messages that the federate under study exchanges with the RTI.
- two automated generation activities, the first one to support the initial specification of the BPMN Collaboration model, and the second one to generate the federation implementation;

The process includes four main activities, each further structured in different steps (see numbers from 1 to 6 in Fig. 1) which are described below:

- **Federation Specification:** In this activity, the federation requirements are initially identified and then are used to define a conceptual model of the federation. Specifically:

  1. *Federation Model Specification:* in the first step, the federation requirements are provided to an HLA expert who is in charge of manually specifying a UML-based Federation Model. Such a model aims at allowing federation's participants to agree on (i) a common *data model* which describes model elements which play a role in an HLA-based federation (e.g., entities which have to be mapped to *Object Classes* or *Interaction Classes*), (ii) the *capabilities* each participant shall provide (e.g., who publishes/subscribes what) and, finally, (iii) other HLA-related information required for describing the federation (e.g., synchronization points). The output of the step 1 is a UML model annotated with the FOM-HLA Profile.
  2. *Model-to-Text Transformation:* an automated *model-to-text* transformation is executed to generate the Federation Object Model (FOM) from the annotated federation model. The generation of an XML-based FOM from a UML model annotated with the FOM-HLA Profile has been introduced in [7] and will not be further discussed in this work.

- **Federation Preliminary Design:** This activity deals with the specification of a BPMN Collaboration Model which provides a behavioral view of the federation under study. Such model aims at helping each federate's designer in (i) defining the activity flow performed by the federate under his/her responsibility, and (ii) describing the interaction with other federates in terms of the messages flow exchanged among the collaboration's participants. Specifically:

3. *Model-to-Model Transformation:* at the third step a *model-to-model* transformation is executed to generate the *Preliminary BPMN Collaboration Model*. Specifically, the BPMN model includes the one BPMN `Pool` element for each federate (plus one Pool for the RTI) and a preconfigured list of the messages which are exchanged during the federation execution. Such elements are generated according to the stereotypes included in the UML federation model.

- **Federation Detailed Design and Implementation:** In this activity each partner is engaged in the development of its own federate.

4. *Federates Development:* as previously stated, on the one hand the proposed method aims at providing a general framework for supporting the federation implementation and, in this respect, introduces standards and architectures such as HLA or MDA to pursue interoperability and reduce the development effort. On the other hand, the method does not make any assumption on strategies and technologies adopted for developing the required federates. Indeed, in a general perspective, the implementation of each federate might be based on different strategies, methodologies and technologies. As stated in Section 1, along with the definition of the method, this paper also discusses its actual adoption in a concrete case in which a JAVA-based simulator and a DEVS-based one are federated in a joint simulation experiment (see Section 4. In this respect, the specific process adopted by each federation participants to develop the relevant federate is discussed in Sections 3.4 and 3.5, respectively.

- **Simulation Deployment and Execution:** The last activity deals with the actual execution of the distributed simulation system:

5. *Federate Deployment and Configuration:* the fifth step consist in a manual activity in which the federates generated at step 4 are deployed and the federation environment is configured;
6. *Federation Execution:* finally, at the last step, the federation is ready to be executed to carry out the simulation study for the addressed scenario.

3.2. UML profile for annotating the federation model

The UML Federation Model specified at the first step of the methodology illustrated in Fig. 1 plays a primary role in the proposed approach. On the one hand, it constitutes the input of the *model-to-text* transformation in charge of generating the FOM (step 2 in Fig. 1). On the other hand, it is also used for deriving the BPMN Preliminary Collaboration Model (step 3 in Fig. 1). In order to generate the expected outputs, the above-mentioned model transformations have to be provided with appropriate information which describe the federation model from an HLA-oriented perspective. To this purpose, a UML profile which extends the UML semantics and, ultimately, allows designers to enrich the federation model with the required HLA-related concepts. An initial version of the adopted *FOM-HLA Profile* has been proposed in [7]. In this work, the profile has been extended, in order to make the UML Federation Model suitable for driving the above-mentioned model transformations.

The FOM-HLA Profile consists of a set of stereotypes which have been defined as extensions of standard UML metaclasses. Each stereotype allows the annotation of appropriate UML model elements so as to specify the required information need for characterizing the element in terms of relevant HLA properties.

The metamodel of the FOM-HLA Profile is shown in Fig. 2. The profile includes the *HLADatatypes* package, illustrated in Fig. 3, which provides a set of enumerated types used for describing the attributes of available stereotypes. The profile stereotypes and the relevant data types have been both based on HLA Object Model Template, the standard that defines the format of HLA FOMs [27].

The red boxes in Figs. 2 and 3 enclose the profile parts which have been revised or added in this work. Specifically, as regard the profile stereotypes, new stereotypes have been introduced, i.e., `<<SimpleDataType>>`, `<<FixedRecordDataType>>`, `<<ArrayDataType>>`, `<<EnumeratedDatatype>>` and `<<VariantRecordDatatype>>` for extending the UML metaclasses *Datatype* and *Class* and allows the specification of relevant types provided by the Object Model Template. Moreover, the new stereotypes `<<SyncPoint>>` `<<Register>>` and `<<Achieve>>` have been introduced as extension of the *Association* UML Metaclass, for allowing the appropriate representation of each federate responsibility in achieving and/or registering the required synchronization points. It is worth underlining that the `<<Register>>` stereotype can also be used for denoting that a federated is required to register a given Object Class. Finally, the `<<Federate>>` has been revised and various attributes have been introduced in order to allow the specification of the following implementation-related properties:

- **timeRegulating**: a boolean value which is set to *true* if the federate is time-regulated, *false* otherwise;
- **timeConstrained**: a boolean value which is set to *true* if the federate is time-constrained, *false* otherwise;
- **timeManagement**: an enumeration which denotes whether a federate is *time-stepped* or *event-driven*. The attribute is specified by the *TimeManagementKind* type.
- **timeImplementation**: an enumeration which denotes whether a federate logical time is implemented as a *float* or an *integer* value. The attribute is specified by the *TimeImplementationKind* type.
- **lookadead**: the (optional) value of the federate lookahead;
- **generate**: a boolean value which denotes the federate under study. It is set to *true* if the federate code has to be generated, *false* otherwise;

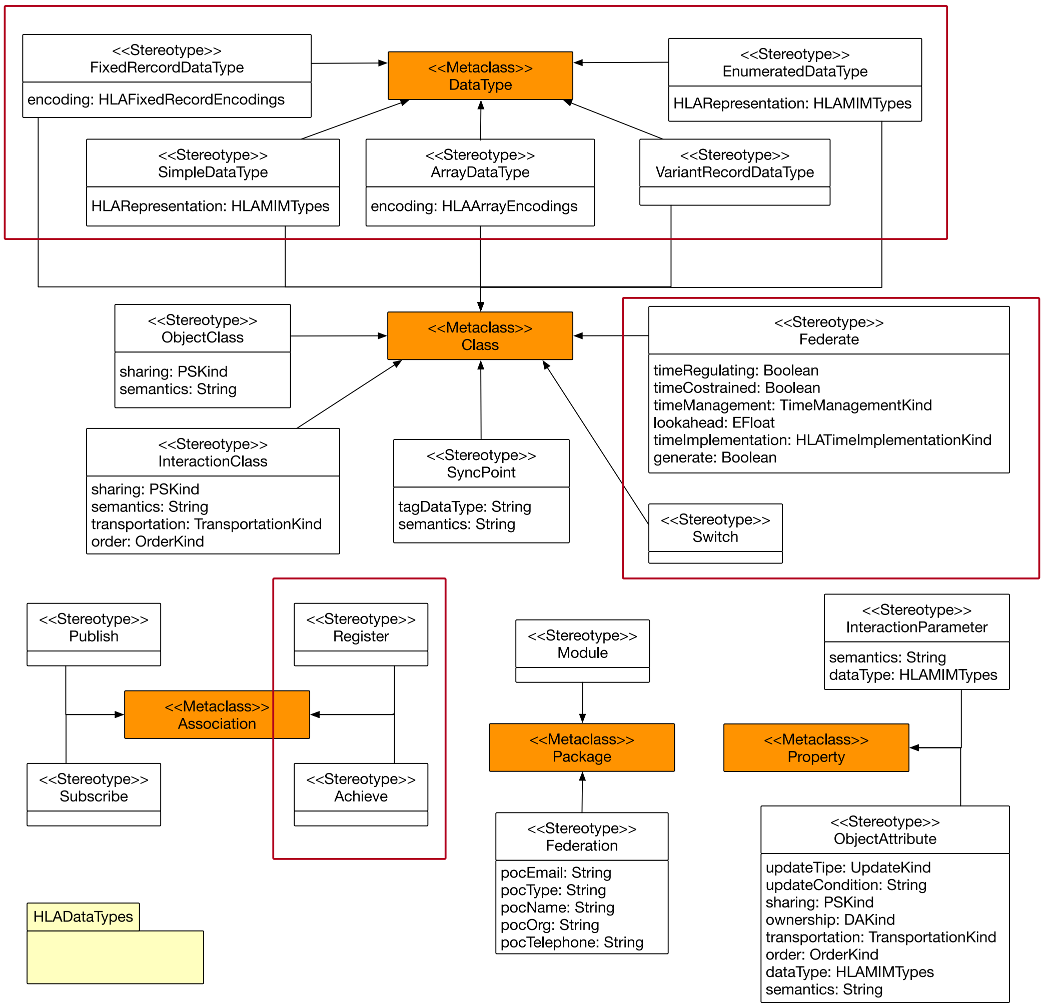


Fig. 2. Metamodel of the FOM-HLA Profile.

As regard the *HLADatatypes* package, five new enumerations have been introduced. Specifically, `HLAArrayEncodings`, `HLAFixedREcordEncodings` and `HLAVariantRecordEncodings` have been introduced for enabling a compliant representation of relevant HLA data types. Finally, `HLATImeImple- mentationKind` and `TimeManagementKind` have been introduced to include in the model implementation-level information related to the federate logical time and its management.

3.3. BPMN extension

As mentioned in Section 3.1, the third step of the proposed methodology introduces an automated transformation step in charge of generating the preliminary BPMN Collaboration Model, from the Federation Model.

In an HLA federation, federate interaction follows a message-passing paradigm which uses the HLA RTI as a *broker*. In this respect, such a preliminary BPMN Collaboration model includes the following elements:

- a BPMN *participant* element for each class of the federation model annotated with the `<<federate>>` stereotype;
- a BPMN *participant* which represents the HLA Run Time Infrastructure;
- an empty BPMN *Pool* for each *participant* introduced in the two previous steps, which constitutes its visual representation; a list of *message* elements representing the information each participant exchanges with the RTI.

More specifically, messages are included to represent, e.g., a participant which sends a message to the RTI for publishing or subscribing an interaction class or an object class. Messages are also exchanged when interactions are sent or received or for notifying the update of an object attribute. In this respect, the preliminary BPMN Collaboration includes the semantic description of the various messages, but the corresponding visual elements are not added to the model: the actual message flow will be specified during a

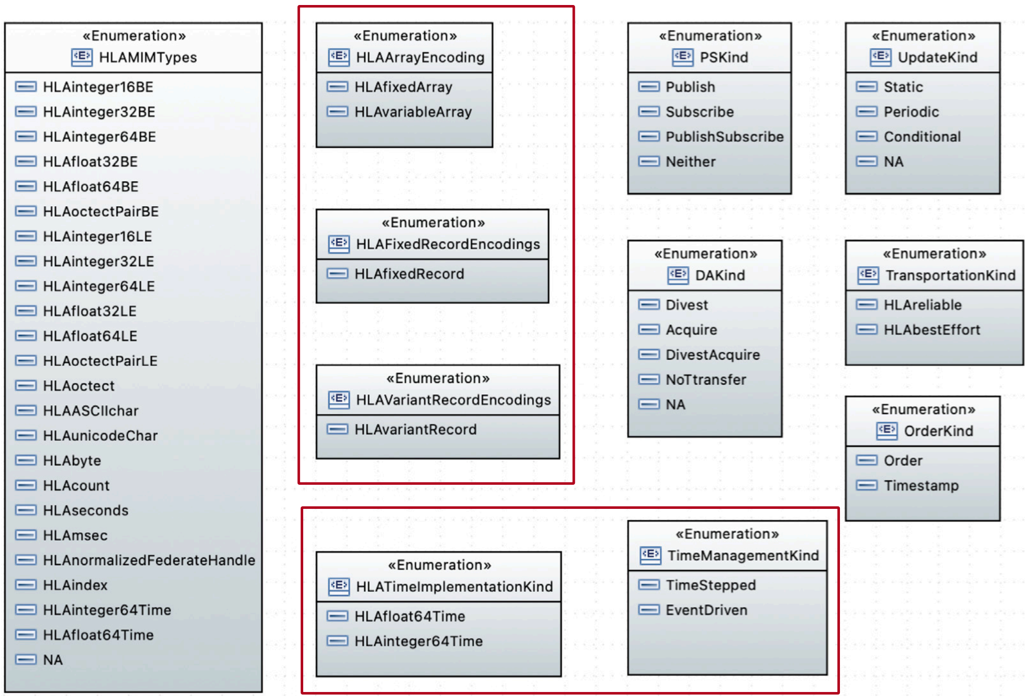


Fig. 3. Metamodel of the FOM-HLA Profile (HLADatatypes package).

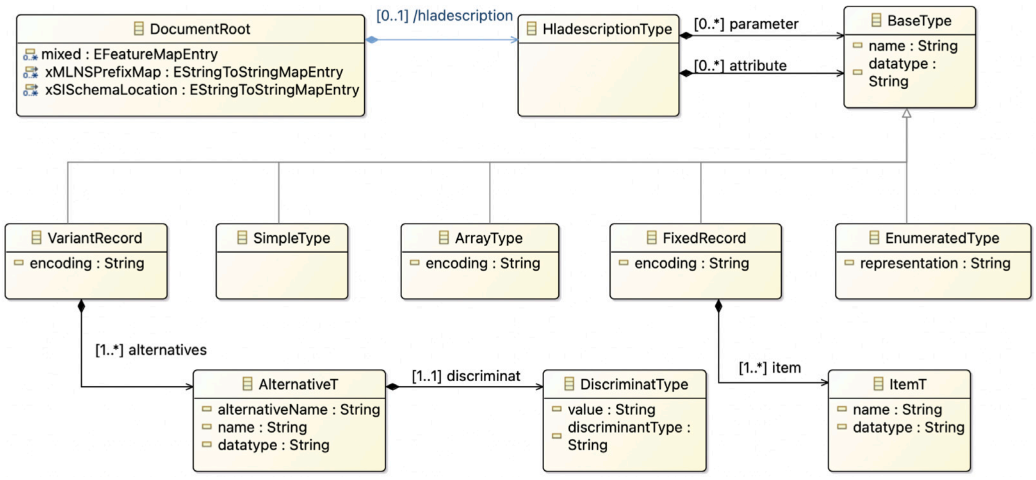


Fig. 4. Metamodel of the HLA-oriented BPMN extension.

subsequent *refinement* step. As discussed in Section 3.1, during the *Federate Development* step, the message flow will be manually specified by selecting the appropriate message from the list of existing ones, according to the BP execution flow.

As underlined in Section 3.1, the FOM specifies a common data model defining the data type of each information entity produced and consumed at federation execution time. In an HLA-based simulation, such information entity can be an *object class*, which represents a persistent entity, an *interaction class*, which represents an event, or a synchronization message. In this respect, the proposed approach exploits the UML Class Diagram constituting the Federation Model to specify the data model. Differently, the BPMN Collaboration model aims at specify the flow of messages, as described in Section 3.

As highlighted in Section 3.1, the BPMN standard metamodel lacks metaclasses that allow the description of HLA messages exchanged during the federation execution. Therefore, a BPMN extension addressing the specification of data exchanged in a message flow is required. For this purpose, this work makes use of the BPMN extension introduced in [8], which is hereby summarized for the sake of completeness.

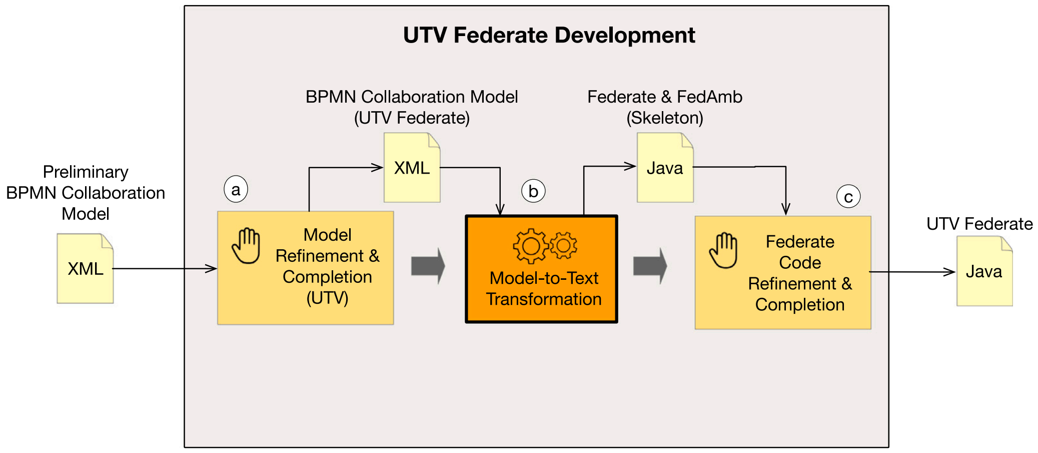

UTV Federate Development
Preliminary BPMN Collaboration Model
XML
a
Model Refinement & Completion (UTV)
BPMN Collaboration Model (UTV Federate)
XML
b
Model-to-Text Transformation
Federate & FedAmb (Skeleton)
Java
c
Federate Code Refinement & Completion
UTV Federate
Java

Fig. 5. MDA-based development process for the Java Federate.

Various approaches can be followed for extending BPMN, each providing specific advantages and weaknesses. An exhaustive comparison among the various approaches for extending BPMN is out of the scope of this paper. Interested readers are sent to [8] where a detailed discussion is provided.

Specifically, the above-mentioned extension has been initially specified from a conceptual point of view. A metamodel compliant to the OMG Meta-Object Facility standard [28] has been introduced to describe the various extension elements and clarify how they extend the relevant BPMN base classes.

The metamodel has been serialized to a corresponding XSD schema. The BPMN process collaboration model includes an `Extension` element in each message which represents an interaction between a federate and the RTI, whose structure is provided according to the relevant XML schema. Finally, such message extension is handled by the automated transformation steps in charge of generating the preliminary collaboration model and the federate implementation.

The metamodel shown in Fig. 4 represents from a conceptual point of view the BPMN semantic extension adopted in this work. The metamodel implementation is based on *Ecore*, the Eclipse reference implementation of the Meta-Object Facility standard. The extension core is constituted by the main class *HLAdescriptionType* which allows the specification of *object class* attributes and *interaction class* parameters. The various subclasses of the *BaseType* class allow the appropriate and complete description of the actual type used for defining each attribute or parameter. A complete description of each BPMN extension element is provided in [8].

The metamodel is finally serialized to an XSD-based XML schema, which is used for specifying a corresponding `hlaExt` namespace.

3.4. Development of the JAVA-based federate

The fourth step of the development process shown in Fig. 1 deals with the development of the required federates. As illustrated in Section 3, the proposed method does not make any assumption on strategies and technologies adopted for the actual development of each federate, but only assumes that the main input is constituted by the preliminary BPMN Collaboration Model. In this respect, this section illustrates how a Java-based federate is generated starting from the XML representation of the collaboration model, by introducing appropriate MDA-based model transformations. The development process, which is shown in Fig. 5, is structured in three steps:

(a) **Refinement of the Collaboration Model:** the Preliminary Collaboration model which constitutes the input of this step is the result of the automated *Model-to-Model* transformation executed at step 3 (see Fig. 1. Such a model includes empty BPMN pools, each associated to one federate (plus one pool representing the RTI) and a pre-configured list of the messages that each participant is expected to exchange with the RTI. It should be underlined that the BPMN specification of such messages is augmented according to the BPMN extension illustrated in Section 3.3. The refinement step is a manual activity carried out by a human expert which specifically focuses on a designated participant (i.e., a BPMN *Pool*) that in this specific case is the JAVA-based federate. The model refinement is accomplished by creating the message flow exchanged by the designated participant and the RTI, by selecting the concrete message from the pre-configured list of existing messages.

(b) **Execution of the Model-to-Text Transformation:** at the second step of the proposed method, an automated Model-to-Text Transformation that generates a preliminary implementation of the federate under study is executed. The transformation has been specified by use of Acceleo [29], a plugin of the Eclipse development environment. Acceleo implements the OMG MOFM2T standard [30] and provides a *model-to-text* transformation engine which takes as input an XMI model that conforms to a given Ecore-based source metamodel and yields as output a text document. The latter is obtained by use of a *template-based approach*, where fixed text is completed with the information retrieved from the input model. Specifically, the JAVA implementation of the designated federate generated at this step is structured as follows:

- a *Datatypes package*, which includes the Java classes required for implementing the elements used for typing attributes of UML classes stereotyped as `<<ObjectClass>>` and `<<InteractionClass>>`;
- an *Interactions package*, which includes the Java classes that implement UML classes stereotyped as `<<Interaction-Class>>`;
- an *Objects package*, which includes the Java classes that implement UML classes stereotyped as `<<ObjectClass>>`;
- a *Federate package*, which includes the Java implementation for the *federate* and the *federate ambassador*.
- a *Simulation package*, which includes the implementation of Java classes for initializing and starting the simulation environment.

(c) **Java Code Refinement and Completion:** the automated model transformation is able to generate a significant portion of the required Java code, specifically those strictly tied to HLA (e.g. object classes and interaction classes declaration, publication and subscription, attributes and parameters definition, types encoding and decoding, etc.). In order to finalize the federate code and make it actually *executable*, a manual step is required to take into consideration the implementation of features and aspects that cannot be automatically derived from the input model (e.g., details on federate's internal behavior not captured by the BPMN Collaboration model).

3.5. Development of the DEVS-based federate

The primary objective of this section is to outline a methodology for transforming an existing BPMN model, enriched with BPSIM parameters, into a Java Federate capable of performing event-driven simulation using the DEVS formalism. In the approach used in this paper, we utilize DEVS-Suite [31] to simulate the resulting DEVS model.

The methodology commences with the Preliminary BPMN Collaboration Model, as referenced in Section 3. This model is already augmented with a BPMN extension for detailing messages exchanged with the RTI. An adept modeling user refines this model, with particular attention to a designated collaborator (or pool). The refinement encompasses:

- Integrating proprietary business processes (BPMN+BPSIM) into the designated collaborator.
- Connecting all the inputs and outputs of the model to the RTI pool using the interaction send and receive predefined messages.
- Incorporating two supplementary BPMN tasks into the flow:
`PS_Int_Classes` and `PS_Obj_Classes`. These tasks are interconnected with the RTI pool through predefined messages.

Upon completion of these tasks, the refined BPMN+BPSIM model stands ready for the transformation phase.

3.5.1. Automatic transformation

In the course of developing this methodology, a comprehensive analysis of possible BPMN and BPSIM combinations was conducted. The extensive range of allowed combinations posed a significant challenge, demanding an effective solution for preserving vital functionalities in the transformation.

Instead of specifying the behavior of these combinations and then creating transformation rules conforming to this specification, we chose to describe these specifications in a model similar to BPMN, which we call the Generic Interaction Model.

The transformation of a BPMN+BPSIM model to DEVS is then directed by this Generic Interaction Model, streamlining the transformation process. To further refine this approach, we introduced the Intermediate Interaction Model (I2M) [10].

I2M allows for a separation of concerns, breaking down functionalities and modeling their interactions separately in a BPMN-like manner. Importantly, this methodology focuses on modeling interactions rather than resorting to code-based solutions. This is advantageous for maintaining flexibility, as any changes in system dynamics can be easily accounted for by modifying the interaction model, without the need for coding adjustments. Once the BPMN+BPSIM model is translated into I2M, the subsequent transformation to DEVS is a straightforward one-to-one mapping. We have developed a dedicated library of I2M models in DEVS-Suite; the final DEVS model is simply a coupling of the corresponding elements from this library, streamlining the entire transformation process.

To encapsulate the potential relationships a BPMN task can establish within the BPMN or BPSIM frameworks, we formulated a Generic Interaction Model for a BPMN Task. To elucidate the aforementioned transformation process, we refer to a simple example showcased in the Generic Interaction Model in Fig. 6, while noting that it does not represent all potential combinations.

During the initial transformation, the focus remains on pinpointing the interactions between the BPMN and BPSIM elements in the user's model using the Generic Interaction Model. Subsequently, non-essential I2M elements are discarded to derive the final I2M model.

For example, when examining a BPMN Receive Task that incorporates BPSIM time and resources, the corresponding I2M representation aligns with the Generic Interaction Model for a task, omitting unnecessary elements, specifically i2 m:Send(msg) and i2 m:Timer(Prob. Dist), that are not required for this particular instance.

The final step in the transition to DEVS only necessitates a DEVS library that accommodates the I2M elements in a one-to-one relationship. Leveraging the DEVS-Suite where we have housed a dedicated library of I2M models facilitates this procedure, with the ultimate DEVS model simply being a connection of the corresponding components from this library. Detailed mappings between

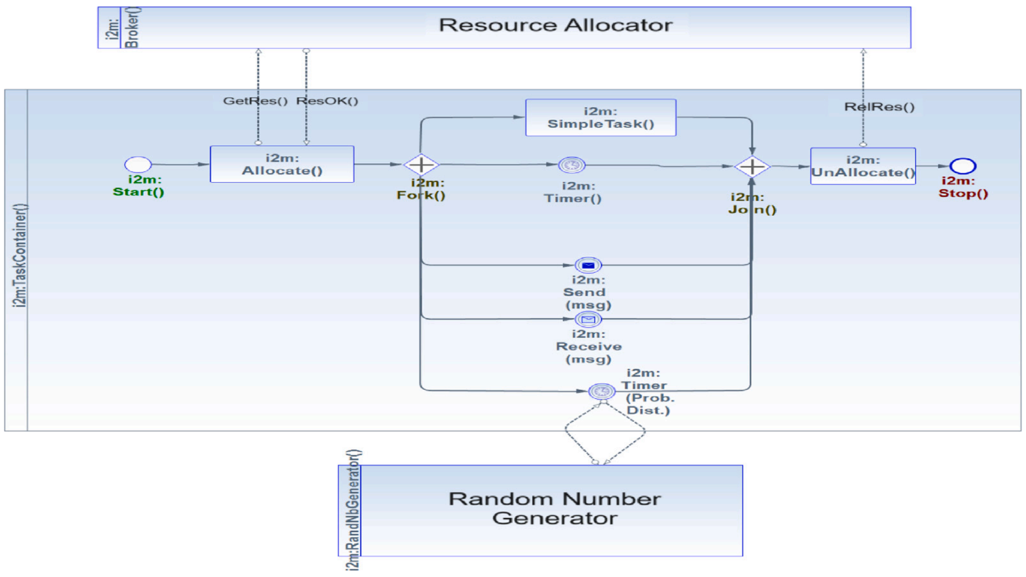


Fig. 6. Generic Interaction Model of a Task.

I2M elements and their DEVS equivalents are provided in [10], ensuring accurate and functional representations within the DEVS framework.

We now facilitate a dual-phase transformation, first from BPMN+BPSIM to I2M and then from I2M to DEVS. These transformations are executed using the Atlas Transformation Language, a pivotal tool for describing complex transformations in a readable, declarative, and maintainable manner [32]. With the DEVS model in hand, a Model-to-Text transformation is carried out using Acceleo [29].

This process generates Java code compliant with the DEVS model. Leveraging HLA-dedicated libraries and a selection of APIs created to communicate with DEVS-Suite, a Java DEVS Federate is constructed. Our approach incorporates a dual-phase Model-to-Model transformation followed by a Model-to-Text transformation, which ultimately facilitates the generation of the federate code for the DEVS-based Java federate.

This approach provides an efficient and flexible means to transform BPMN models, enriched with BPSIM parameters, into a simulation-ready format. The advantage of this methodology lies not only in its accuracy in capturing the dynamic nature of business processes but also in its adaptability, allowing for incremental changes and updates without the need for extensive code revisions. By utilizing this method, we contribute to preserving intellectual property, as the two different federates (Java-only federate and DEVS-based federate) are generated separately, thereby promoting interoperability.

Our comprehensive methodology ensures a seamless transition from a BPMN model to an event-driven simulation, facilitating in-depth analysis and potential optimization of complex systems.

4. Example application

This section describes an application case that highlights the dynamic interactions between two main entities: Collaborator 1 (IMT) and Collaborator 2 (UTV).

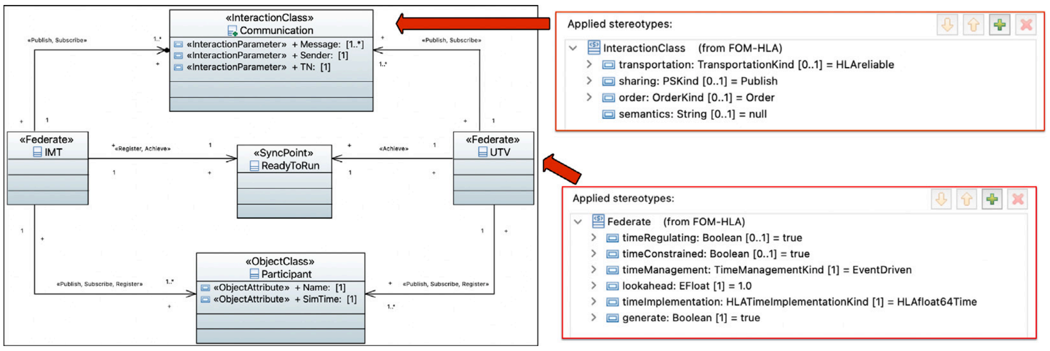
In compliance with the federation requirements, UTV dispatches a “Specification” message to IMT. After processing this specification, IMT sends back a “Design” message to UTV.

A UML conceptual model for the federation, illustrated in Fig. 7, is constructed based on these interactions. This model outlines vital components, symbolizes entities with HLA terminology, identifies roles such as publishers and subscribers, and incorporates HLA-specific markers including synchronization points. The FOM-HLA UML Profile annotations are incorporated for enhanced clarity.

Following the completion of the UML Federation Model, the Model-to-Text transformation phase starts. This UML model is instrumental in guiding the FOM file’s creation, which outlines the federation’s structure, interactions, and adherence to established standards.

The *Preliminary BPMN Collaboration Model* is then derived using Model-to-Model transformation, with the UML model as its basis. Components from the UML are converted into their BPMN counterparts. Entities like ’Collaborator 1 (IMT)’, ’Collaborator 2 (UTV)’, and ‘RTI’ are represented as BPMN Pools, derived directly from the UML. Simultaneously, a message list is auto-generated, emphasizing the key communications. See Section 3.1.

The BPMN diagram in Fig. 8 displays the theoretical synchronized workflow between UTV and IMT. UTV initiates the process with a predesign phase, transmitting specifications to IMT, and subsequently awaits the design. Concurrently, IMT processes UTV’s


«InteractionClass»
Communication
«InteractionParameter» + Message: [1..*]
«InteractionParameter» + Sender: [1]
«InteractionParameter» + TN: [1]
«Federate»
IMT
«Federate»
UTV
«SyncPoint»
ReadyToRun
«ObjectClass»
Participant
«ObjectAttribute» + Name: [1]
«ObjectAttribute» + SimTime: [1]
«Publish, Subscribe»
«Register, Achieve»
«Achieve»
«Publish, Subscribe, Register»
1
1..*
+

```
Applied stereotypes:
  InteractionClass    (from FOM-HLA)
    transportation: TransportationKind [0..1] = HLAreliable
    sharing: PSKind [0..1] = Publish
    order: OrderKind [0..1] = Order
    semantics: String [0..1] = null
```

```
Applied stereotypes:
  Federate    (from FOM-HLA)
    timeRegulating: Boolean [0..1] = true
    timeConstrained: Boolean [0..1] = true
    timeManagement: TimeManagementKind [1] = EventDriven
    lookahead: EFloat [1] = 1.0
    timeImplementation: HLATimeImplementationKind [1] = HLAfloat64Time
    generate: Boolean [1] = true
```

Fig. 7. UML Federation Model annotated with the FOM-HLA Profile.

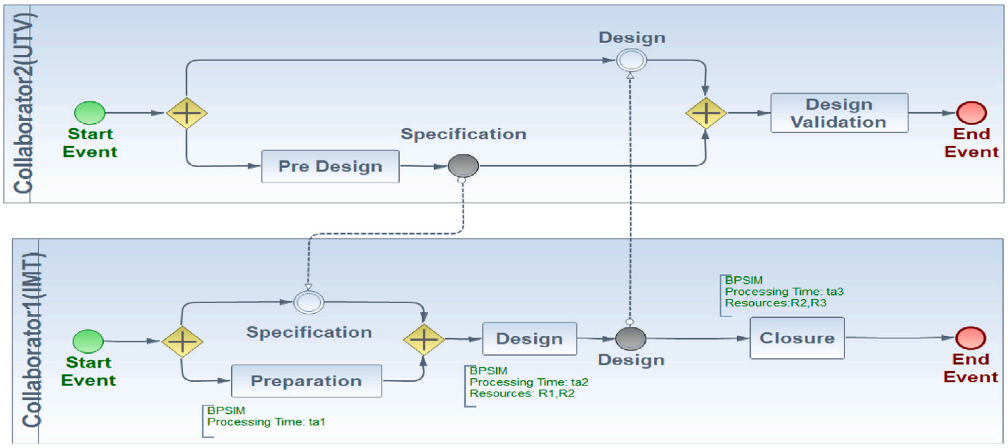


Fig. 8. Application case.

specifications, moves to the design phase, and forwards the design to UTV for validation. In real life, these collaborators are only aware of the exchanged messages but not each other's flow. The real communication between these two flows will be done only through the RTI. A key aspect of their collaboration is the protection of intellectual property. The primary goal of this simulation is to ensure that proprietary processes of both IMT and UTV remain confidential. While the BPMN serves as a mutual reference, each collaborator refines its portion within the BPMN model separately.

Both IMT and UTV will then independently refine their respective models. The outcomes of these refinement efforts are illustrated in Figs. 9 and 10. This is done according to Section 3.

Both federates implement the Interaction and Participant Object configuration during their initial setups, correlating them with HLA methods. The interactions and Objects are defined in the auto-generated FOM file. Configuration details include:

- Communication Configuration Messages:
  - Send_Int_Communication (SEND) and Receive_Int_Communication (RECV): Relate to HLA methods for sending and receiving interactions: sendInteraction, receiveInteraction.
- Initialization Configuration Messages:
  - ROP (Register_Obj_Participant): For registering a participant object: Relate to HLA method for object configurations: registerObjectInstance.
  - PIC (Publish_Int_Communication) and SIC (Subscribe_Int_Communication): For configuring the publish and subscribe mechanisms for interactions: Relate to HLA methods for publishing and subscribing to interactions: subscribeInteractionClass and publishInteractionClass.
  - POP (Publish_Obj_Participant) and SOP (Subscribe_Obj_Partici- pant): For configuring the publish and subscribe mechanisms for objects: Relate to HLA methods for publishing and subscribing to object attributes: subscribeObjectClassAttributes and publishObjectClassAttributes.

Post configuration, the Java-Based federate is prepped for RTI communication based on the FOM file.

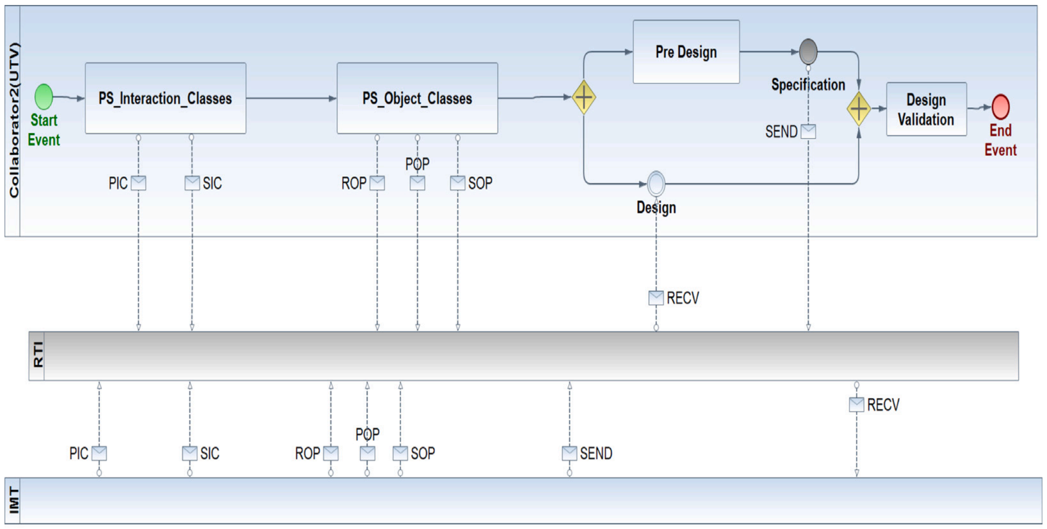


Fig. 9. Refined UTV model.

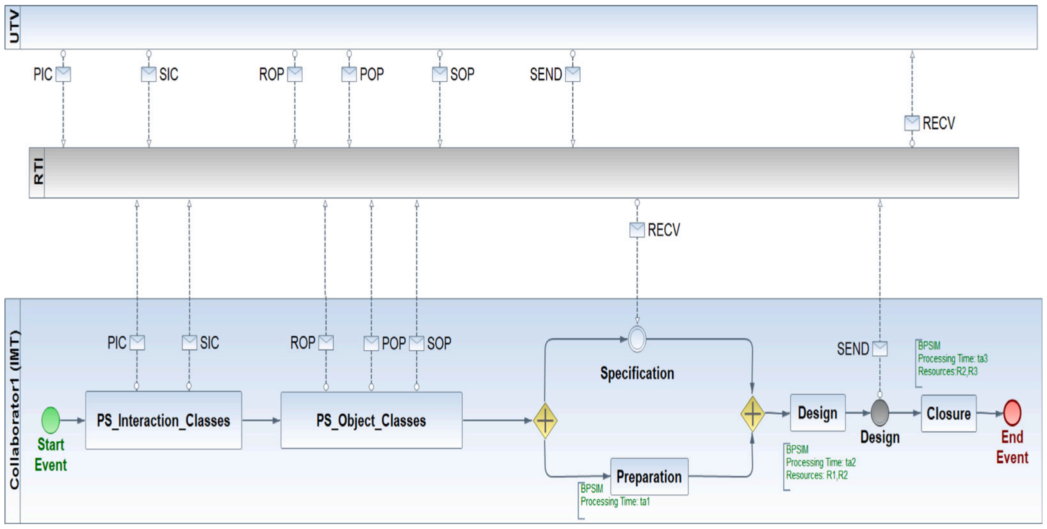


Fig. 10. Refined IMT model.

The Java-based federate is developed following the method illustrated in Section 3.4. The availability of an execution environment, such as the Eclipse Modeling Framework, makes the methodology easy to be actually carried out. The adoption of MDA-based model transformations allow the seamless execution of the various steps and guarantees the straightforward coherence of the different artifacts (data model, preliminary and complete collaboration model, federate implementation). The federate implementation step highly benefits from the availability of automated transformation, as the proposed approach considerably reduces the required effort. In this respect, it should be underlined that observed advantages are not limited to a mere reduction of the lines of code to be developed by hand. Indeed, the transformation step specifically addresses the automated generation of the HLA-related code, which is the most difficult portion to be developed and the one which requires significant know-how and technical skills.

For the DEVS-based federate, we follow the method described in Section 3.5. The BPMN+BPSIM model undergoes conversion into an I2M model, subsequently transforming into a DEVS simulation that encapsulates essential parameters and task dependencies.

This dual-stage transformation, from BPMN+BPSIM to I2M (Fig. 11) and subsequently to DEVS (Fig. 12), offers a holistic perspective on the process, setting the foundation for anticipated improvements and augmented efficiency.

Having generated the final DEVS model, we will proceed with the Model-to-Text transformation to produce the DEVS-based federate. The transformation also includes connections with the DEVS-Suite APIs. A DEVS Coupled coordinator is then instantiated

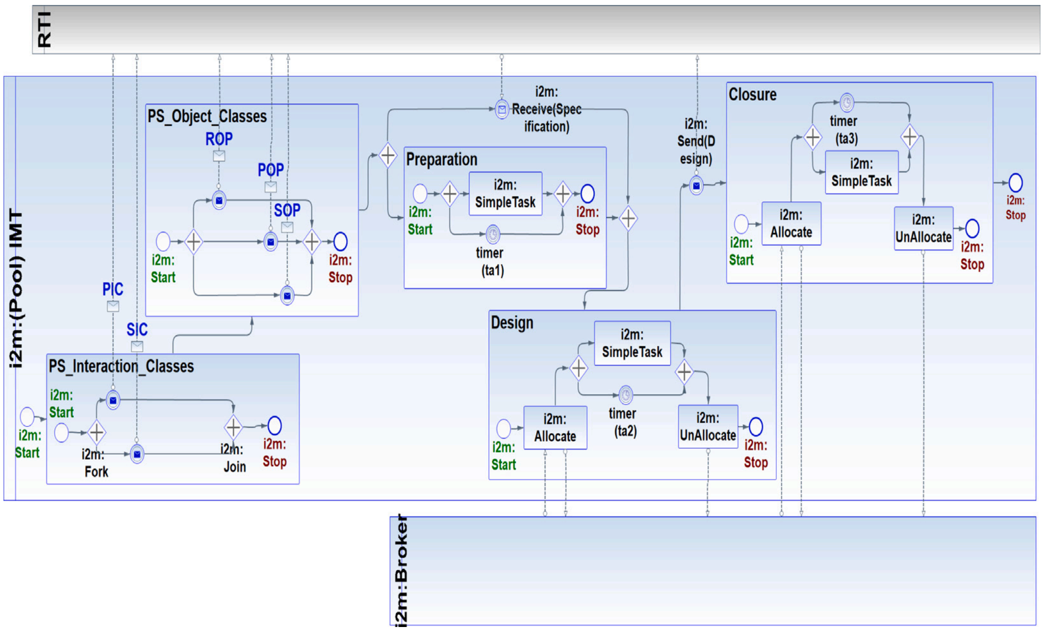


Fig. 11. I2M Model corresponding to the IMT participant.

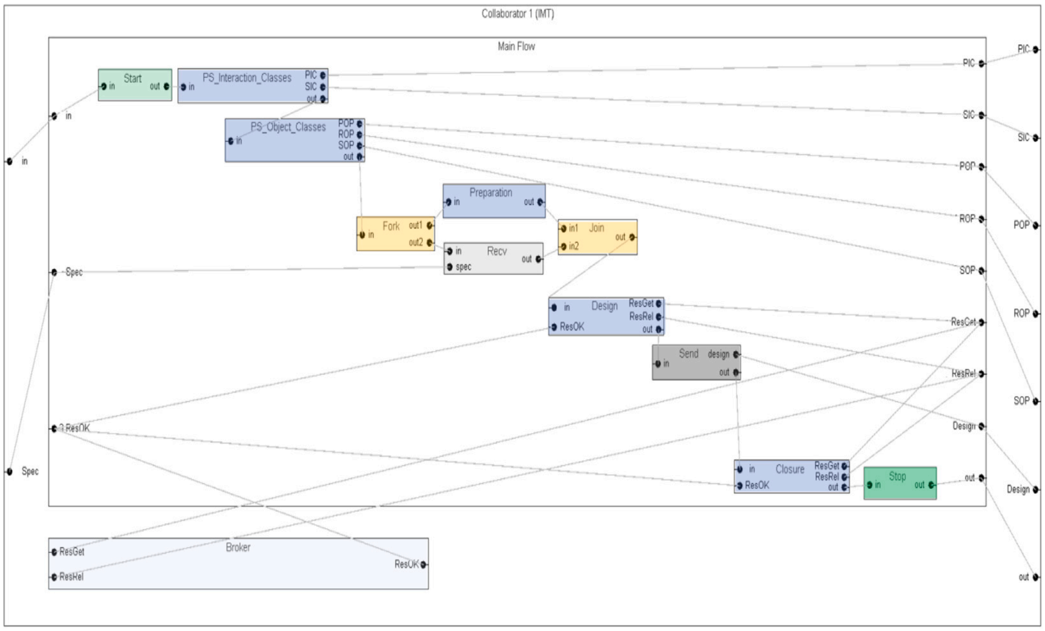
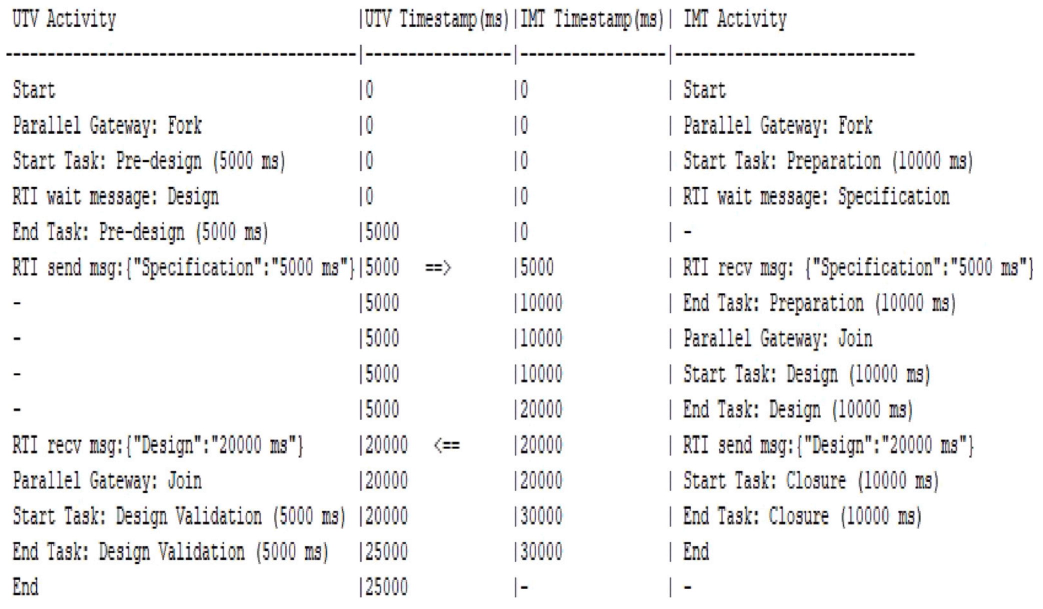


Fig. 12. DEVS Model resulting from I2M.

and configured for the IMT model. Integration with this DEVS Coordinator facilitates simulation in DEVS-Suite and allows for the injection of simulation events. This two-tiered transformation procedure offers a thorough insight into the complete workflow, forming a foundation for prospective improvements.

Fig. 13 displays the logs derived from the collaborative simulation illustrated in Fig. 8. In this example, the task durations for UTV and IMT are set at 5000 ms and 10 000 ms, respectively. Starting their processes individually at time 0 ms, both UTV and IMT progressed according to their own operational methodologies.

The cornerstone of their collaboration lies in the synchronization facilitated by secure message exchanges through the RTI. This synchronization, first evident at the 5000 ms mark, allows for secure sharing of sensitive data, bridging the separate timelines of UTV and IMT into a unified workflow. It showcases the effectiveness of message-driven synchronization in collaborative environments.



```
UTV Activity                               |UTV Timestamp(ms)|IMT Timestamp(ms)| IMT Activity
-------------------------------------------|-----------------|-----------------|-----------------------------
Start                                      |0                |0                | Start
Parallel Gateway: Fork                     |0                |0                | Parallel Gateway: Fork
Start Task: Pre-design (5000 ms)           |0                |0                | Start Task: Preparation (10000 ms)
RTI wait message: Design                   |0                |0                | RTI wait message: Specification
End Task: Pre-design (5000 ms)             |5000             |0                | -
RTI send msg:{"Specification":"5000 ms"}|5000   ==>       |5000             | RTI recv msg: {"Specification":"5000 ms"}
-                                          |5000             |10000            | End Task: Preparation (10000 ms)
-                                          |5000             |10000            | Parallel Gateway: Join
-                                          |5000             |10000            | Start Task: Design (10000 ms)
-                                          |5000             |20000            | End Task: Design (10000 ms)
RTI recv msg:{"Design":"20000 ms"}         |20000   <==      |20000            | RTI send msg:{"Design":"20000 ms"}
Parallel Gateway: Join                     |20000            |20000            | Start Task: Closure (10000 ms)
Start Task: Design Validation (5000 ms)    |20000            |30000            | End Task: Closure (10000 ms)
End Task: Design Validation (5000 ms)      |25000            |30000            | End
End                                        |25000            |-                | -
```

Fig. 13. UTV-IMT collaboration simulation log.

The log reveals a notable point that the UTV federate completes its process at 25 000 ms, while the IMT federate concludes at 30 000 ms. Despite the difference in completion times, the successful synchronization of tasks and message exchanges guide the independent simulations of UTV and IMT towards a harmonized conclusion, illustrating the efficacy of our approach in managing collaborative simulations with disparate task durations and operational strategies.

5. Conclusions

In response to the growing demand for collaborative simulations, our research introduces a methodology that integrates the High-Level Architecture with the Model Driven Architecture. Through detailed model-to-model and model-to-text transformations, our approach facilitates the transformation of a conceptual UML federation model into the Federation Object Model and a Preliminary BPMN Collaboration Model. As a result, this framework has led to the automatic generation of code for two distinct federates, each embracing a unique simulation approach: one based on Java and the other anchored in the DEVS formalism. Despite their differing simulation paradigms, these federates demonstrated impressive interoperability within the HLA framework. Their ability to work together, given their distinct simulation foundations, emphasizes the versatility and resilience of our introduced method.

A core accomplishment of our research is its dedication to preserving confidentiality in collaborative simulations. We have devised a mechanism that enables varied business models to interact efficiently, yet with assurance that their proprietary information remains uncompromised. Through this selective and controlled information exchange methodology, we maintain the essential confidentiality of proprietary processes while enabling effective collaboration in a shared simulation environment.

In summation, this study provides a comprehensive approach to distributed simulations, striking a balance between the advantages of collaboration and the imperative need for autonomy and security. Future work will focus on refining this methodology, enhancing its operational efficiency and reinforcing its security mechanisms.

Data availability

Data will be made available on request.

Acknowledgments

IMT authors extend their appreciation to the “Région Occitanie, France” for their financial support, instrumental in the realization of the research encompassing the “BusinEss procEss webbased Platform for multi modeling and SIMulation” (Allocation Doctorale BeePSIM - n° 105743 / 21013151). Andrea D’Ambrogio’s contribution has been supported by the Spoke 1 “FutureHPC & BigData” of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by MUR Missione 4 Componente 2 Investimento 1.4: Potenziamento strutture di ricerca e creazione di “campioni nazionali” di R&S (M4C2-19) - Next Generation EU (NGEU).

References

[1] K. Mehdouani, N. Missaoui, S.A. Ghannouchi, An approach for business process improvement based on simulation technique, Procedia Comput. Sci. 164 (2019) 225–232.

[2] F. Steiner, et al., Industry 4.0 and business process management, Teh. Glasnik 13 (4) (2019) 349–355.

[3] C. Gomes, C. Thule, J. Deantoni, P.G. Larsen, H. Vangheluwe, Co-simulation: the past, future, and open challenges, in: Leveraging Applications of Formal Methods, Verification and Validation. Distributed Systems: 8th International Symposium, ISoLA 2018, Limassol, Cyprus, November 5-9, 2018, Proceedings, Part III 8, Springer, 2018, pp. 504–520.

[4] G. Zacharewicz, N. Daclin, G. Doumeingts, H. Haidar, Model driven interoperability for system engineering, Modelling 1 (2) (2020) 94–121.

[5] IEEE, IEEE standard for modeling and simulation (M&S) high level architecture (HLA)– framework and rules, in: IEEE Std 1516-2010 (Revision of IEEE Std 1516-2000), 2010, pp. 1–38, http://dx.doi.org/10.1109/IEEESTD.2010.5553440.

[6] C. Gomes, C. Thule, D. Broman, P.G. Larsen, H. Vangheluwe, Co-simulation: a survey, ACM Comput. Surv. 51 (3) (2018) 1–33.

[7] P. Bocciarelli, A. D'Ambrogio, A. Giglio, E. Paglia, Automated generation of fom modules for HLA-based distributed simulations, in: 2019 Spring Simulation Conference, SpringSim, 2019, pp. 1–12.

[8] P. Bocciarelli, A. D'Amgrogio, A low-code approach for simulation-based analysis of process collaborations, in: C.G. Corlu, S.R. Hunter, H. Lam, B.S. Onggo, J. Shortle, B. Biller (Eds.), Proceedings of the 2023 Winter Simulation Conference, Institute of Electrical and Electronics Engineers, Inc., Piscataway, New Jersey, 2023, pp. 2530–2541.

[9] M. El Kassis, F. Trousset, G. Zacharewicz, N. Daclin, Bridging the gap between business process and simulation: transformation from BPMN to DEVS, IFAC-PapersOnLine 56 (2) (2023) 11888–11893.

[10] M. El Kassis, F. Trousset, G. Zacharewicz, N. Daclin, Incremental transformation of BPSIM-enriched BPMN models into DEVS, in: 2023 Winter Simulation Conference, WSC, IEEE, 2023, pp. 2542–2553.

[11] P. Vincent, K. Iijima, M. Driver, J. Wong, Y. Natis, Magic Quadrant for Enterprise Low-Code Application Platforms, Gartner report, 2019.

[12] A.C. Bock, U. Frank, Low-code platform, Bus. Inf. Syst. Eng. 63 (6) (2021) 733–740, http://dx.doi.org/10.1007/s12599-021-00726-8.

[13] R. Waszkowski, Low-code platform for automating business processes in manufacturing, IFAC-PapersOnLine 52 (10) (2019) 376–381.

[14] OMG, MDA Guide Revision 2.0 (ORMSC/14-06-01), 2003, https://www.omg.org/cgi-bin/doc?ormsc/14-06-01.pdf, Accessed 29 January 2022.

[15] O. Topçu, U. Durak, H. Oğuztüzün, L. Yilmaz, Distributed Simulation: A Model Driven Engineering Approach, Springer, 2016.

[16] G. Zacharewicz, C. Frydman, N. Giambiasi, G-DEVS/HLA environment for distributed simulations of workflows, Simulation 84 (5) (2008) 197–213.

[17] J. Possik, A. D'Ambrogio, G. Zacharewicz, A. Amrani, B. Vallespir, A BPMN/HLA-based methodology for collaborative distributed DES, in: 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE, IEEE, 2019, pp. 118–123.

[18] P. Bocciarelli, A. D'Ambrogio, A. Giglio, E. Paglia, D. Gianni, A transformation approach to enact the design-time simulation of bpmn models, in: 2014 IEEE 23rd International WETICE Conference, 2014, pp. 199–204, http://dx.doi.org/10.1109/WETICE.2014.27.

[19] P. Bocciarelli, A. D'Ambrogio, A. Giglio, E. Paglia, BPMN-based business process modeling and simulation, in: N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, Y.-J. Son (Eds.), Proceedings of the 2019 Winter Simulation Conference, Institute of Electrical and Electronics Engineers, Inc., Piscataway, New Jersey, 2019, pp. 1439–1453, http://dx.doi.org/10.1109/WSC40007.2019.9004960.

[20] P. Bocciarelli, A. D'Ambrogio, A. Giglio, E. Paglia, D. Gianni, Empowering business process simulation through automated model transformations, in: Simulation Series, 46, The Society for Modeling and Simulation International, 2014, pp. 278–286, URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84901985241&partnerID=tZOtx3y1.

[21] P. Bocciarelli, A. D'Ambrogio, E. Paglia, A Language for Enabling Model-driven Analysis of Business Processes, in: Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development, MODELSWARD '14, SciTePress, Lisbon, Portugal, 2014b.

[22] P. Bocciarelli, A. D'Ambrogio, E. Paglia, A. Giglio, An HLA-based BPMN extension for the specification of business process collaborations, in: 2017 IEEE/ACM 21st International Symposium on Distributed Simulation and Real Time Applications (DS-RT), 2017, pp. 1–8, http://dx.doi.org/10.1109/DISTRA.2017.8167668.

[23] O. Topçu, M. Adak, H. Ovguztüzün, A metamodel for federation architectures, ACM Trans. Model. Comput. Simul. 18 (3) (2008) 10:1—-10:29, http://dx.doi.org/10.1145/1371574.1371576.

[24] W.S. McKenzie, Process modeling for simulation: Observations and issues, Proceedings of the 2016 Winter Simulation Conference (2016) 1072–1083.

[25] OMG, Business Process Model And Notation (BPMN) version 2.0.2, 2014, https://www.omg.org/spec/BPMN/, Accessed 4 March 2022.

[26] IEEE, 1730-2010 Distributed simulation engineering and execution process (DSEEP), 2010.

[27] IEEE, IEEE standard for modeling and simulation (M&S) high level architecture (HLA)- object model template (OMT) specification, in: IEEE Std 1516.2-2010, 2010, http://dx.doi.org/10.1109/IEEESTD.2010.5557731.

[28] OMG, Meta Object Facility (MOF) Specification, Version 2.4.2, 2017, https://www.omg.org/spec/MOF/2.4.2/, Accessed 29 January 2022.

[29] Eclipse Foundation, Acceleo, 2012, https://www.eclipse.org/acceleo/, Accessed 29 January 2022.

[30] OMG, MOF Model to Text Transformation Language (MOFM2T), 1.0, 2008, https://www.omg.org/spec/MOFM2T, Accessed 29 January 2022.

[31] H. Sarjoughian, A. Mahmoodi Markid, EMF-DEVS Modeling, in: Simulation Series, vol. 44, 2012.

[32] Jouault, Allilaire, Bézivin, Kurtev, ATL: A model transformation tool, Sci. Comput. Program. 72 (1–2) (2008) 31–39.