Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

A LOW-CODE APPROACH FOR SIMULATION-BASED ANALYSIS OF PROCESS COLLABORATIONS

Paolo Bocciarelli
Andrea D'Ambrogio

Department of Enterprise Engineering
University of Rome Tor Vergata
Via del Politecnico, 1
00133 Rome, ITALY

ABSTRACT

The simulation-based analysis of process collaborations introduces significant challenges, such as the ability to focus on the interchange of information and data without disclosing any internal details of collaboration participants' processes. The use of distributed simulation (DS) provides good opportunities to face these challenges. However, properly using DS standards and technologies requires significant technical know-how and effort. This paper introduces a largely automated approach to carry out distributed simulations of process collaborations. The DS standard addressed by the paper is the High Level Architecture (HLA), which is used to analyze process collaborations specified by using the Business Process Model and Notation (BPMN). The degree of automation is obtained by using a low-code development paradigm based on automated model transformations that reduce the amount of manual effort required to code the HLA-based simulation. An example application is also discussed to underline the pros and cons of the proposed approach.

1 INTRODUCTION

In a *process collaboration* different organizations engage in collaborative tasks for collectively contributing to an overall business process (BP). Process collaborations are based on the interchange of information and data among distributed participants that are interested to create value by interoperating with each other without however disclosing the details of their internal business processes (Law and Kelton 2007; Falcone et al. 2017).

The ability to properly analyze and evaluate the potential of opening to collaboration is extremely important for organizations that plan to share their resources in collaborative workflows. In this respect, the adoption of simulation-based analysis techniques is acknowledged as a valuable option to get a quantitative evaluation of different alternative scenarios in terms of measures of performance and effectiveness.

However, conventional discrete-event simulation tools do not provide the required functionality and computational capabilities to deal with the inherently distributed, heterogeneous and interoperable workspace introduced by process collaborations. The use of distributed simulation (DS) approaches naturally meets the requirements of simulation-based analysis of process collaborations. Specifically, the reference standard in the DS field, namely HLA (High Level Architecture) (IEEE 2010b), allows one to specify and implement a distributed simulation as a *federation* of simulation components that represent how single participants (*federates* in HLA terms) contribute to a process collaboration. The federates taking part to an HLA-based simulation act as black-box components that have only to agree about what information is to be exchanged, without disclosing any information about their implementation details. An organization interested to open their processes to collaboration could thus provide the corresponding HLA federates and evaluate the impact of prospective market opportunities.

Unfortunately, the aforementioned advantages of DS approaches are to be balanced with some well-known difficulties, such as the significant expertise, technical know-how and effort that are required to properly use DS standards, e.g. HLA, and the related implementation technologies.

This paper aims at overcoming these difficulties by proposing a model-based approaches that introduces a significant degree of automation to implement HLA-based simulations of process collaborations specified in BPMN (Business Process Model and Notation), the standard language for specifying BP models (OMG 2014).

BPMN allows covering a wide range of uses and enables the representation of both *private* processes and *public* processes. A private process describes the internal process executed by single organizations, providing a detailed view of the process flow. Instead, a public process describes the inter-organizational cooperation of a given participant in a process collaboration with other entities.

The manual effort that is typically required to implement an HLA-based simulation of a BPMN-based process collaboration is largely reduced in the proposed approach by using a *low-code* development paradigm, which has originated and is gaining increasing attention in the software engineering field (Vincent et al. 2019; Bock and Frank 2021; Waszkowski 2019). According to the main principles of this paradigm, a development process should foster the adoption of code generation environments to pursue the reduction of the manual effort required to build the application code. In the paper case, the HLA implementation code is partially obtained by introducing appropriate model transformations that are driven by the BPMN-based process collaboration model.

In order to fully specify the information required to enact the low-code approach, the BPMN-based behavioral model is enriched with a UML-based structural model (i.e., a class diagram) that provides the relevant data to specify the so-called federation object model (FOM), which describes the data produced and consumed by each federate in the HLA-based simulation.

An appropriate extension is introduced to properly adapt the modeling capabilities provided by BPMN to the HLA domain. Indeed, the standard semantics of BPMN does not allow the detailed specification of the messages exchanged by participants of a process collaboration. For such a reason, this paper contribution also includes a BPMN extension that allows the specification of messages exchanged by collaboration participants, which are represented as federates in the HLA-based simulation.

The remainder of this paper is structured as follows: Section 2 provides the literature review. Section 3 describes the BPMN extension at the basis of the proposed model-based method, which is then introduced in Section 4 along with a case study in the manufacturing domain. Section 5 provides a detailed description of the model-based method and shows its actual application to the addressed case. Finally, Section 6 provides concluding remarks.

2 RELATED WORK

This section reviews existing literature in the DS field, specifically with regards to i) the adoption of approaches to facilitate DS development and, ii) the investigation of the joint role of BPMN and HLA-based DS to support the simulation of process collaborations.

The idea of introducing approaches and technologies to mitigate the difficulties of DS development has been largely investigated in the past. Among various relevant contributions, in Topçu et al. (2016) and Zacharewicz et al. (2008), authors argue about the need of easing the DS development by discussing the added value provided by model-driven engineering principles and standards.

In previous contributions we have presented approaches and tools to enhance and make easier the development of DS systems, which also take into consideration the business process management domain. In Possik et al. (2019), a BPMN- and HLA-based a method to integrate discrete event simulation components has been introduced, while in (Bocciarelli et al. 2014; Bocciarelli et al. 2019), 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 (Bocciarelli et al. 2014; Bocciarelli et al. 2014b) that supports both the execution of sequential simulations and HLA-based distributed simulations. In

Bocciarelli et al. (2019), an automated approach which focuses on the development of the HLA Federation Object Model (FOM) is proposed. In Bocciarelli et al. (2017), 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 RTI services and handle the related *callbacks*.

As regards modeling of HLA federations, in Topçu et al. (2008), the lack of modeling formalisms specifically addressing the behavioral description of HLA federates is underlined. In addition, in Wagner, G. and Seck, M. and McKenzie, F. (2016), authors highlight the lack of established practices for dealing with behavioral models in DS. In this respect, the model of the HLA federaton 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 (OMG 2014). In order to specify the data exchanged by federates representing participants of a process collaboration, this paper introduces a BPMN extension, as described in Section 3, which has been designed and implemented according to the BPMN standard extension mechanism.

3 BPMN EXTENSION FOR DATA MODELING

In an HLA federation, federate interaction follows a message-passing paradigm, according to which each federate sends and receives messages by using a broker denoted as the HLA *Run-Time Infrastructure (RTI)*. As underlined in Section 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 approach proposed in this paper makes use of a UML Class Diagram, to specify the data model, and a BPMN Collaboration, to define the flow of messages, as described in Section 4. Both artifacts are then used to generate the federation implementation. In this scenario, an essential aspect to be addressed is related to the information that these two artifacts shall provide. 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 offer metaclasses to model the structure and details of data exchanged in the business process. It is thus necessary to introduce a BPMN extension, which can be carried out by using various strategies, such as Zarour et al. (2020):

- **BPMN Metamodel extension:** the BPMN metamodel can be extended with additional metaclasses to provide the concepts required for representing the required information. The main advantage of this approaches lies in its flexibility and in the clear separation of concerns between the extension concepts and the actual model that is instantiated from the metamodel. A drawback is that extended models are not strictly conforming with the BPMN standard and cannot be edited by BPMN compliant tools;
- **BPMN native extension mechanism:** the BPMN standard provides a set of extension elements which can be used to extend the expressiveness of standard elements (OMG 2014). Specifically, an `ExtensionDefinition` element identifies a named group of new attributes that can be used by other BPMN elements. The `ExtensionDefinition` is in turn associated to a constellation of other BPMN elements which contribute to specify the extension (e.g., `ExtensionAttributeDefintion`, `ExtensionAttributeValue`, etc.) as shown in Figure 1. The major advantage of this approach is that the extended models are still conforming to

the BPMN standard, so that they can be handled by any BPMN compliant tools (included those which are not required to *understand* the extension semantics). On the other hand this approach shows several flaws (Braun 2015; Stroppi et al. 2011):

- the extension mechanism is discussed from a syntactical point of view but there is a lack of any methodological guide;
- an extension definition is related to many other BPMN elements (i.e., the extended elements and those that are required to be defined for specifying the extension itself). Thus, dealing with complex extensions may result hard;
- any BPMN element can access each extension definition, causing semantic irregularities and an unclear separation of concerns;
- the native extension does not make any difference between attribute extension and element extension;
- the native extension violates the separation of abstraction layers that the OMG defines in its Model Driven Architecture effort (OMG 2003). Specifically, an `ExtensionAttributeValue` class, which is at M2 (metamodel) layer, is further specified by an `Element` class, which is the most generic class at M3 layer (OMG 2004) (see Figure 1, which refers to MOF, the Meta Object Facility, as the standard language for specifying metamodels).

• **Reference to an external data model:** the extension can be provided by an external data model which is referred by use of an existing attribute of the BPMN metamodel. As noted in Stroppi et al. (2011), the specification of an extension directly in the model constitutes a valuable and self-consistent option. However, in this paper case, the BPMN Collaboration is taken as input by a model transformation in charge of generating the federation code, and the use of an external data structure makes harder the transformation implementation.

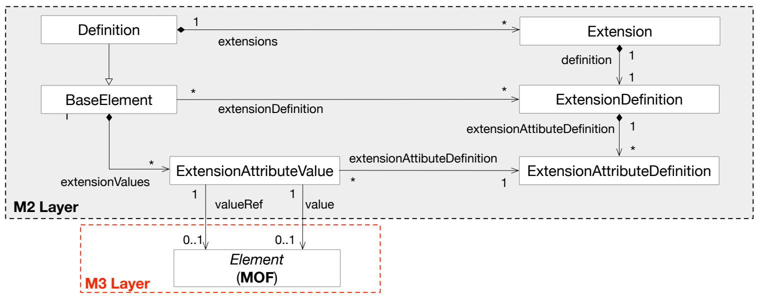

Figure 1: Abstraction issue of the BPMN native extension mechanism: the `ExtensionAttributeValue` class, which belongs to the M2 layer, refers to the MOF *Element* class at M3 layer.

Finally, it should be underlined that the `ExtensionDefinition`, `ExtensionAttributeDefinition` and `ExtensionAttributeValue` classes are not applicable when the XML-based serialization of models and metamodels is used, as stated in the BPMN specification. Thus, at XML level, the reference to the *ExtensionDefinition* class provided by the `Extension` class is simply an XML qualified name.

In order to overcome the limitations of above-mentioned strategies and leverage on the relevant advantages, the extension proposed in this paper jointly exploits a metamodel-extension and the native BPMN extension mechanism, also defining a methodology which provide the required guidance to appropriately leverage the extension for DS development.

Specifically, as clarified in Section 5.2, in order to address the BPMN extension from a conceptual point of view, the extension has been initially designed as a MOF-based metamodel. The latter has been then 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, as better explained in 5.3, such message extension is handled by the automated transformation steps in charge of generating the preliminary collaboration model and the federate implementation.

4 MODEL-BASED DISTRIBUTED SIMULATION OF PROCESS COLLABORATIONS

The core of the proposed contribution is constituted by the model-based method hereby outlined, which starts from and significantly extends a previous contribution (Bocciarelli et al. 2019). As underlined in Section 2, the common objective of the two approaches is to provide means to ease the development of HLA-based distributed simulations. However, the previous approach only focuses on the generation of the FOM, while this work addresses the entire HLA development process.

Specifically, the proposed approach addresses the development of a simulation component (e.g, a federate) that has to be deployed and integrated with other federates in a DS implementation. Generally speaking, in a DS scenario, the various participants contribute to the development of the complex simulation experiment by first defining a common agreement on the information to be exchanged during the simulation execution and then providing the required simulation components. In this respect, the proposed method takes into consideration the distributed simulation from the perspective of one specific participant that is interested to i) understand and possibly contribute to the definition of a common data model, ii) develop its own federate according to the agreed data model, and iii) make the federate available in a distributed infrastructure, in order to eventually run the DS code (and thus evaluate the potential of the simulated process collaboration).

This section outlines the various steps that composes the proposed method and specifies the required inputs and the expected outputs of each step. A concrete application scenario dealing with the simulation of a manufacturing system is then introduced in Section 4.1. Finally, Section 5 discusses the actual application of the method to the addressed case and provides a detailed description of each step.

The proposed method is inspired by the conceptual pillars of the Distributed Simulation Engineering and Execution Process (DSEEP) (IEEE 2010a), which introduces a standardized process for managing distributed simulations, and makes use of a *low-code development* approach.

Appropriate guidance is provided to specify a UML Class Diagram describing the federation structure along with its data model, as well as a BPMN Collaboration model describing the federation behavior from the point of view of a given federate (denoted as the *federate under study*). Moreover, the method effectively eases the implementation of the federate under study by introducing a model transformation that generates a significant portion of the required HLA implementation code, written in Java. The following software components are used:

- the FOM-HLA UML Profile, introduced in Bocciarelli et al. (2019), which allows the annotation of the structural part of the UML Class Diagram to map HLA-related concepts to UML model elements;
- 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 proposed method is shown in Figure 2. The first step deals with the generation of the FOM. This activity has been deeply discussed in (Bocciarelli et al. 2019), which provides the relevant details. For the sake of completeness, it is worth noting that this step takes as input the UML class diagram annotated with the FOM-HLA UML Profile and yields as output the FOM.

The second step addresses the generation of a preliminary BPMN Collaboration model which includes the specification of 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.

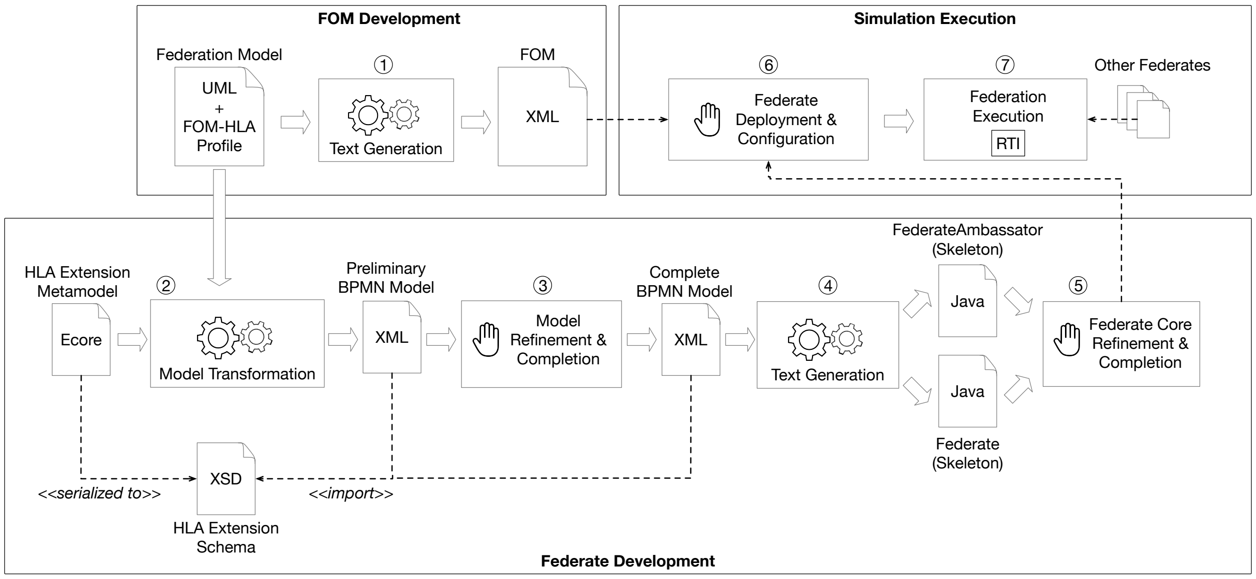

Figure 2: Model-based method for the HLA-based simulation of process collaborations.

At the third step, such a preliminary model is refined in order to build the complete BPMN Collaboration model.

At the fourth step, an additional automated model transformation is executed to generate the implementation of the federate under study in terms of the *federate ambassador*, which implements the interface with the RTI, and the federate code, which implements both HLA-related code and the simulation logic of the federate. It is worth noting that, while such an automated step is able to generate a significant portion of the required Java code, a manual refinement is needed to finalize the federation implementation (see Figure 2, step 5).

Finally, once all required federates are available, the federation is deployed, configured and executed to carry out the simulation study for the addressed scenario.

The next section introduces a case study that is used as a running example application of the proposed method.

4.1 Case Study Description

Let us consider a manufacturing system that involves elements from three different organizations: a *production plant* where electric, hybrid and traditional vehicles are assembled, a *component supplier* that provides engines and batteries, and a *consulting company* in charge of measuring and analyzing the performance of the production process.

A simulation-based analysis has to be carried out to study the supply chain and define the most effective vehicle assembly line configuration. Specifically, the simulation experiment is implemented as an HLA-based simulation that includes three federates: the *production plant federate*, the *supplier federate* and the *business analysis federate*.

Let us assume that the last two federates are already available and reused in the addressed HLA-based simulation. Thus, in order to build and execute a distributed simulation of the vehicle production supply chain, the production plant organization is required to carry out the following steps:

- define and share with other federates a data model so to build the FOM;
- develop the federate that mimics the behavior of the production plant;
- integrate the federate in the overall federation.

The next Section 5 discusses in detail how the proposed method eases and supports the execution of the aforementioned steps.

5 METHOD DETAILS AND EXPERIMENTATION

This section shows how the method introduced in Section 4 can be actually adopted for the DS-based analysis of the scenario provided in Section 4.1. The actual implementation of the model-based method is based on the Eclipse Modeling Framework (EMF) (Eclipse Foundation 2016), which provides an extensible environment for the specification of metamodels and the design and execution of model transformations.

5.1 Specification and Annotation of the Federation Model

The main input of the proposed method, according to the DSEEP prescriptions, is the conceptual federation model that identifies the various entities involved in the federation execution. The federation model is not developed merely for a *contemplative* purpose, e.g., to *describe* the federation. Differently, the federation model is actually taken as input by the automated code generation steps. In this respect, in order to extend the model so to include the additional information that drive the subsequent transformations, the *FOM-HLA* profile has been used to annotate the UML classes that play a relevant role in the HLA federation.

Figure 3 shows the federation model that describes the *Vechile Production Supply Chain*. The various stereotypes provided by the *FOM-HLA* profile allow the specification of HLA-related concepts. As an example, the *ProductionPlant* class and the *EngineSupplier* both act as `federates`. The *PartRequest* class is an `interactionclass` that is `published` by the *ProductionPlant* and `subscribed` by the *EngineSupplier*.

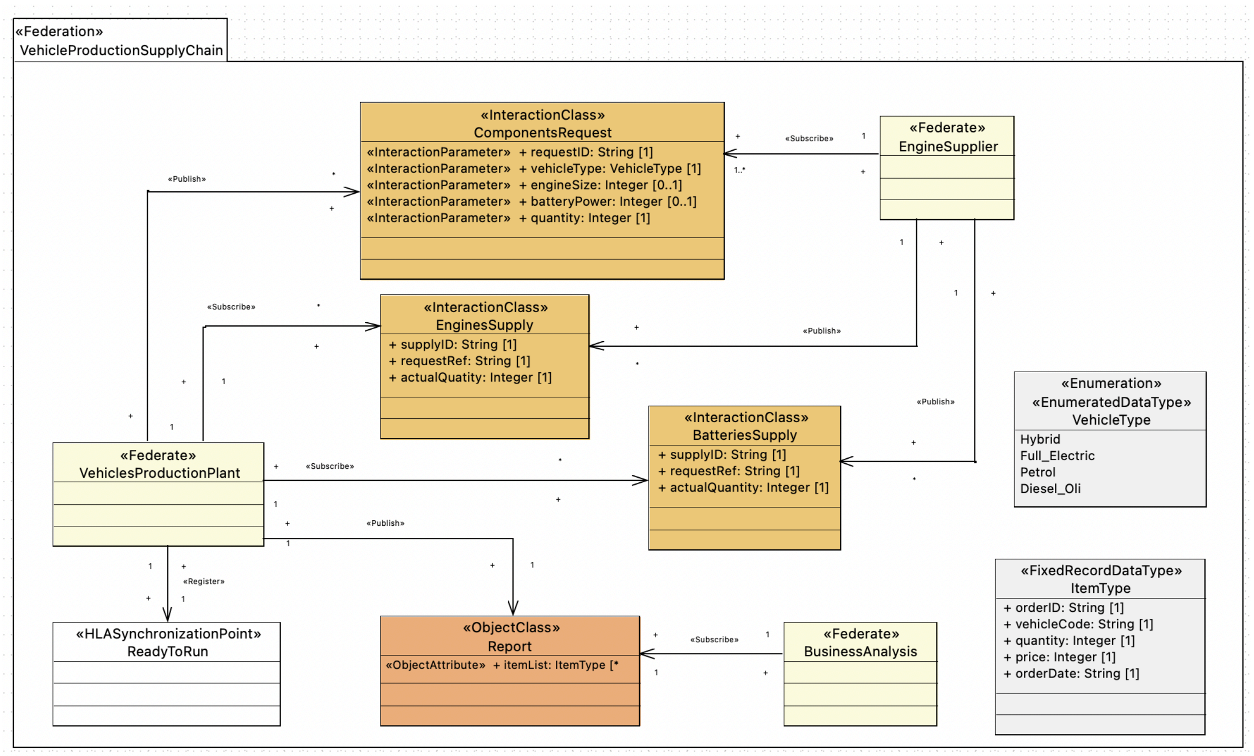
«Federation» VehicleProductionSupplyChain
«InteractionClass» ComponentsRequest
«InteractionParameter» + requestID: String [1]
«InteractionParameter» + vehicleType: VehicleType [1]
«InteractionParameter» + engineSize: Integer [0..1]
«InteractionParameter» + batteryPower: Integer [0..1]
«InteractionParameter» + quantity: Integer [1]
«Federate» EngineSupplier
«InteractionClass» EnginesSupply
+ supplyID: String [1]
+ requestRef: String [1]
+ actualQuatity: Integer [1]
«Enumeration» «EnumeratedDataType» VehicleType
Hybrid
Full_Electric
Petrol
Diesel_Oli
«InteractionClass» BatteriesSupply
+ supplyID: String [1]
+ requestRef: String [1]
+ actualQuantity: Integer [1]
«Federate» VehiclesProductionPlant
«FixedRecordDataType» ItemType
+ orderID: String [1]
+ vehicleCode: String [1]
+ quantity: Integer [1]
+ price: Integer [1]
+ orderDate: String [1]
«HLASynchronizationPoint» ReadyToRun
«ObjectClass» Report
«ObjectAttribute» + itemList: ItemType [*
«Federate» BusinessAnalysis
«Publish»
«Subscribe»
«Register»
1
1..*
+
*

Figure 3: Federation Model of the Vehicle Production Supply Chain.

The annotated federation model obtained in the previous step is then given as input to a model transformation that generates the FOM (see Figure 2 step 1).

As stated before, this step is not further discussed in this paper, interested readers are sent to Bocciarelli et al. (2019).

5.2 Metamodel of the BPMN Extension

As clarified in Section 3, this work introduces a metamodel, shown in Figure 4, which represents from a conceptual point of view the BPMN semantic extension required to provide information about messages exchanged during the federation execution.

The metamodel implementation is based on *Ecore*, the Eclipse reference implementation of the MOF standard, and includes the following metaclasses:

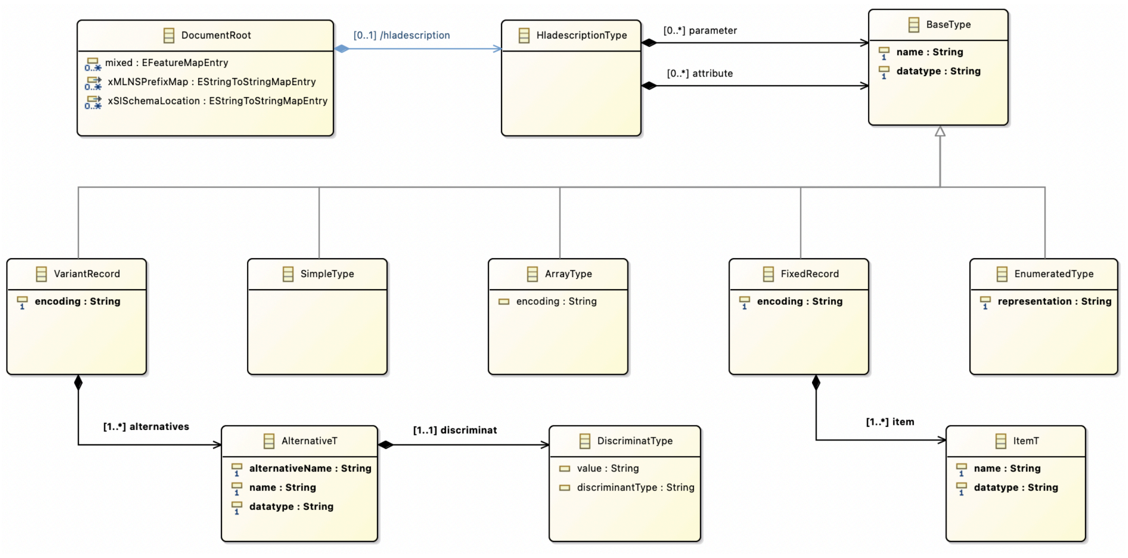

Figure 4: Metamodel of the HLA-oriented BPMN extension.

- *HLAdescriptionType*, which is the main class of the metamodel, used to specify messages by using the two following elements:
  - *parameter*, which describes the parameter of an HLA *interaction class*;
  - *attribute*, which describes the attribute of an HLA *object class*.
- *BaseType*, which is the supertype providing the common elements shared by all relevant subclasses:
  - *name*, which indicates the name of the parameter or attribute;
  - *datatype*, which identifies the attibute/parameter type, used, e.g., to describe a complex type defined in the federation model (and in the FOM, accordingly).
- SimpleType, which maps the HLA simple type;
- EnumeratedType, which maps the HLA enumerate type. It provides the *representation* attribute to specify the type of the enumeration literals;
- ArrayType, which maps the HLA array type. It provides the *encoding* attribute to specify how the array elements have to be encoded;
- FixedRecord, which maps the HLA fixed record type. It provides the following attributes:
  - *encoding*, which specifies how the record has to be encoded;
  - *item*, which describes the *name* and the *datatype* of each record element.
- VariantRecord, which maps the HLA variant record type. It provides the following attributes:
  - *encoding*, which specifies how the record has to be encoded;
  - *alternative*, which describes the structure of each alternative in terms of *name*, *datatype* and relevant value assumed by the *discriminant*.

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

5.3 Generation of the preliminary BPMN model

In order to support the complete specification of the HLA federation, the method includes an automated step dealing with the generation of a *preliminary* BPMN Collaboration model. Specifically, an XML-based representation of a BPMN Collaboration model is generated starting from the annotated UML Class Diagram and the BPMN extension metamodel. This preliminary model includes:

- a `participant` element, along with its visual counterpart (i.e., an empty pool) for each class of the federation model annotated with the «`federate`» stereotype;
- a participant representing the RTI with the corresponding empty pool;
- a list of `message` elements representing the information each participant exchanges with the RTI. As an example, a participant sending a message to the RTI to publish or subscribe an interaction class or an object class. Messages are exchanged to send/receive interactions or to notify an update to an object attribute. In this respect, the BPMN Collaboration does not include any visual element corresponding to the messages. As discussed in Section 5.4, during the BPMN refinement step, the message flow has to be manually specified by selecting the appropriate message from the list of existing ones, according to the BP execution flow. Finally, the `message` elements include an `<extensions>` element, which describes the information that characterizes the message, according to the extension metamodel (and the corresponding XML schema).

Listing 1: XML serialization of the Vehicle Production Supply Chain Preliminary BPMN Model (fragment).

```
<?xml version="1.0" encoding="UTF-8"?>
<bpmn2:definitions xmlns:hlaExt="http://sel.uniroma2.it/hlaExtension"
 ...
 <bpmn2:import importType="http://www.w3.org/2001/XMLSchema" ...>
 <bpmn2:message id="_KDBfcs7YEe2g76bqx6x0pA" name="Send_Int_ComponentsRequest">
  <bpmn2:extensionElements>
   <hlaExt:hladescription>
    <parameter xsi:type="hlaExt:EnumeratedType" representation="HLAunicodeString">
     <name xsi:type="xs:anyType">vehicleType</name>
     <datatype xsi:type="xs:anyType">VehicleType</datatype>
    </parameter>
   </hlaExt:hladescription>
  </bpmn2:extensionElements>
 ...
</bpmn2:definitions>
```

A fragment of the corresponding XML representation is provided in Listing 1. Specifically, the fragment shows the HLA extension which includes the different parameters specifying the `ComponentsRequest` *send interaction* message.

5.4 Specification of the Collaboration

The refinement of the preliminary BPMN Collaboration model is carried out by the following activities:

1. specification of the process diagram executed by the participant representing the federate under study. Specifically, in the considered experimentation, the process to be specified is the one executed by the `VehicleProductionPlant` participant;
2. specification of the message flow that the `VehicleProductionPlant` participant exchanges with the RTI to interact with other participants, i.e., the `EngineSupplier` participant and the `BusinessAnalysis` participant.

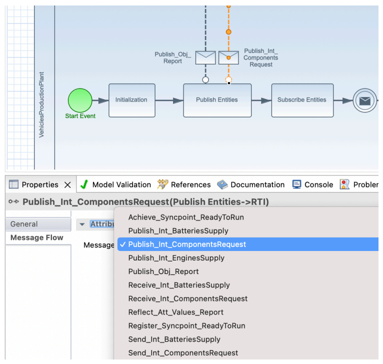
VehiclesProductionPlant
Publish_Obj_ Report
Publish_Int_ Components Request
Start Event
Initialization
Publish Entities
Subscribe Entities
Properties
Model Validation
References
Documentation
Console
Probler
Publish_Int_ComponentsRequest(Publish Entities->RTI)
General
Message Flow
Attribu
Message
Achieve_Syncpoint_ReadyToRun
Publish_Int_BatteriesSupply
✓ Publish_Int_ComponentsRequest
Publish_Int_EnginesSupply
Publish_Obj_Report
Receive_Int_BatteriesSupply
Receive_Int_ComponentsRequest
Reflect_Att_Values_Report
Register_Syncpoint_ReadyToRun
Send_Int_BatteriesSupply
Send_Int_ComponentsRequest

Figure 5: Refinement of the preliminary BPMN model.

The preliminary BPMN Collaboration model includes all possible messages exchanged by the various participants. The model refinement is carried out by creating a `message flow` between two participants and then selecting the concrete message from the pre-configured list of existing messages, as shown in Figure 5.

The complete Vehicle Production Supply Chain BPMN model obtained at the end of the *Model Refinement and Completion* step is given in Figure 6.

5.5 Generation of the Federate Implementation

From a general perspective, the implementation of a federate includes various components, which are designed and developed according to the specific simulation requirements. This paper makes use of the Java language for generating the federate implementation, which 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 «`InteractionClass`»;
- 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.

5.6 Preliminary Evaluation of Experimentation Results

The example application described in previous Section has been used to demonstrate the feasibility of the proposed approach. The availability of an execution environment, such as the Eclipse Modeling Framework, makes the methodology easy to be actually carried out. Moreover, the BPMN extension and the 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).

A measurable metric to evaluate the advantages of the proposed approach can be defined in terms of lines of code (LOCs) to be developed, which may hamper the use of a DS-based approach for analyzing process collaborations. In the addressed case, we have observed that the LOCs to be manually developed during the *federate refinement and completion* are less than 40% of the total federate size. For the *FederateAmbassador* class, this value is around 10%. It should be noted that the saved effort is not limited to a mere reduction of LOCs to be developed by hand, since the code generation specifically addresses the automated generation

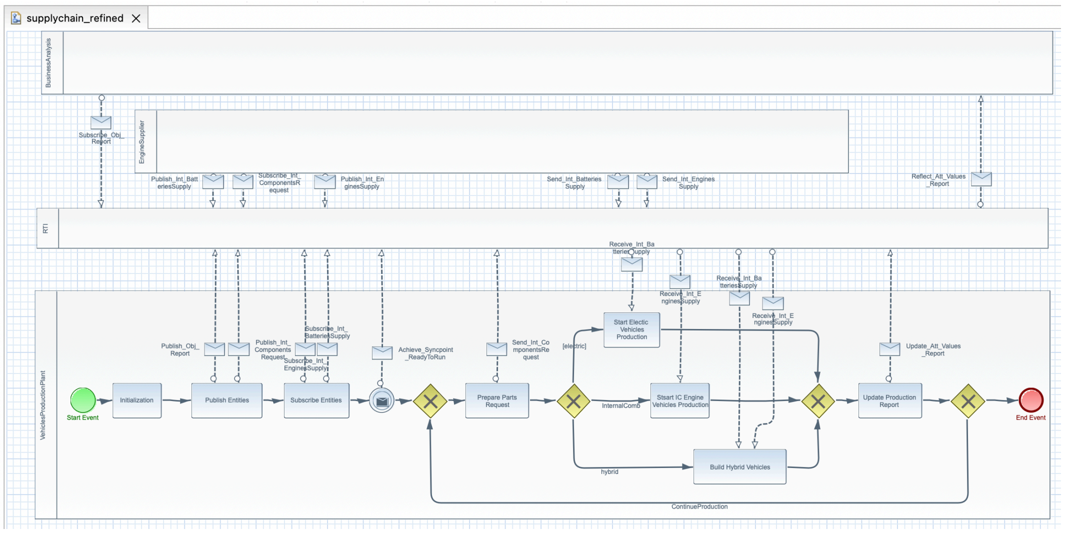
supplychain_refined
BusinessAnalysis
EngineSupplier
RTI
VehiclesProductionPlant
Subscribe_Obj_Report
Publish_Int_BatteriesSupply
Subscribe_Int_ComponentsRequest
Publish_Int_EnginesSupply
Send_Int_Batteries Supply
Send_Int_Engines Supply
Reflect_Att_Values_Report
Receive_Int_BatteriesSupply
Receive_Int_EnginesSupply
Receive_Int_BatteriesSupply
Receive_Int_EnginesSupply
Publish_Obj_Report
Publish_Int_Components Request
Subscribe_Int_BatteriesSupply
Subscribe_Int_EnginesSupply
Achieve_Syncpoint_ReadyToRun
Send_Int_ComponentsRequest
Update_Att_Values_Report
Start Event
Initialization
Publish Entities
Subscribe Entities
Prepare Parts Request
[electric]
Start Electic Vehicles Production
InternalComb
Stsart IC Engine Vehicles Production
hybrid
Build Hybrid Vehicles
Update Production Report
ContinueProduction
End Event

Figure 6: Vehicle Production Supply Chain BPMN model.

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.

6 CONCLUSIONS

This paper has introduced a largely automated (low-code) approach for the HLA-based distributed simulation of process collaboration specified in BPMN. In order to fully specify the information required to enact the low-code approach, the BPMN-based behavioral model is enriched with a UML-based structural model (i.e., a class diagram) which describes the data produced and consumed by each federate in the HLA-based simulation. Moreover, an appropriate extension has been introduced to properly adapt the modeling capabilities provided by BPMN to the HLA domain, so to overcome the limitations in terms of BPMN ability to specify the details of messages exchanged by participants of a process collaboration. The feasibility of the proposed approach has been evaluated by using an example application to a case study dealing with a process collaboration in the manufacturing domain. Ongoing work includes a more extensive experimentation to additional cases, as well as the investigation of the most effective tradeoff for the automated code generation steps, so to minimize the need of manual refinement while taking into account the additional complexity of the modeling activity required to drive the code generation.

ACKNOWLEDGMENTS

This work 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

Bocciarelli, P., A. D'Ambrogio, A. Giglio, and E. Paglia. 2019. "Automated Generation of Fom Modules for HLA-Based Distributed Simulations". In *2019 Spring Simulation Conference (SpringSim)*, 1–12.

Bocciarelli, P., A. D'Ambrogio, A. Giglio, E. Paglia, and D. Gianni. 2014. "Empowering business process simulation through automated model transformations". In *Simulation Series*, Volume 46, 278–286: The Society for Modeling and Simulation International.

Bocciarelli, P., A. D'ambrogio, A. Giglio, E. Paglia, and D. Gianni. 2014. "A Transformation Approach to Enact the Design-Time Simulation of BPMN Models". In *2014 IEEE 23rd International WETICE Conference*, 199–204.

Bocciarelli, P., A. D'Ambrogio, and E. Paglia. 2014b. "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. Lisbon, Portugal: SciTePress.

Bocciarelli, P., A. D'Ambrogio, E. Paglia, and A. Giglio. 2017. "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)*, 1–8.

Bocciarelli, P., A. D'Amgrogio, A. Giglio, and E. Paglia. 2019. "BPMN-Based Business Process Modeling and Simulation". In *Proceedings of the 2019 Winter Simulation Conference*, edited by N. Mustafee, K.-H. G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, 1439–1453. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Bock, A. C., and U. Frank. 2021, December. "Low-Code Platform". *Business & Information Systems Engineering* 63(6):733–740.

Braun, R. 2015. "Behind the Scenes of the BPMN Extension Mechanism Principles, Problems and Options for Improvement". In *2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, 1–8. IEEE.

Eclipse Foundation 2016. *Eclipse Modeling Framework (EMF)*. https://eclipse.org/modeling/emf/, Accessed 20th December 2022.

Falcone, A., A. Garro, A. D'Ambrogio, and A. Giglio. 2017. "Engineering Systems by Combining BPMN and HLA-based Distributed Simulation". In *2017 IEEE International Systems Engineering Symposium (ISSE)*, 1–6. IEEE.

IEEE 2010a. *1730-2010 Distributed Simulation Engineering and Execution Process (DSEEP)*.

IEEE 2010b. *IEEE 1516 - Standard for Modeling and Simulation High Level Architecture - Framework and Rules*.

Law, A. M., and W. D. Kelton. 2007. *Simulation Modeling and Analysis*, Volume 3. Mcgraw-hill New York.

OMG 2003. "MDA Guide revision 2.0 (ormsc/14-06-01)". https://www.omg.org/cgi-bin/doc?ormsc/14-06-01.pdf, accessed 15th April 2022.

OMG 2004. *Meta Object Facility (MOF) Specification, version 2.0*. https://www.omg.org/mof, accessed 15th April 2022.

OMG 2014. "Business Process Model And Notation (BPMN) version 2.0.2". https://www.omg.org/spec/BPMN/, accessed 15th April 2022.

Possik, J., A. D'Ambrogio, G. Zacharewicz, A. Amrani, and B. Vallespir. 2019. "A BPMN/HLA-based Methodology for Collaborative Distributed DES". In *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, 118–123. IEEE.

Stroppi, L. J. R., O. Chiotti, and P. D. Villarreal. 2011. "Extending BPMN 2.0: Method and Tool Support". In *Business Process Model and Notation*, edited by R. Dijkman, J. Hofstetter, and J. Koehler, 59–73. Berlin, Heidelberg: Springer Berlin Heidelberg.

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

Topçu, O., M. Adak, and H. Ovguztüzün. 2008. "A Metamodel for Federation Architectures". *ACM Trans. Model. Comput. Simul.* 18(3):10:1—-10:29.

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

Wagner, G. and Seck, M. and McKenzie, F. 2016. "Process modeling for simulation: Observations and issues". In *Proceedings of the 2016 Winter Simulation Conference*, edited by T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, 1072–1083. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Waszkowski, R. 2019. "Low-code Platform for Automating Business Processes in Manufacturing". *IFAC-PapersOnLine* 52(10):376–381.

Zacharewicz, G., C. Frydman, and N. Giambiasi. 2008. "G-DEVS/HLA Environment for Distributed Simulations of Workflows". *Simulation* 84(5):197–213.

Zarour, K., D. Benmerzoug, N. Guermouche, and K. Drira. 2020. "A Systematic Literature Review on BPMN Extensions". *Business Process Management Journal* 26(6):1473–1503.

AUTHOR BIOGRAPHIES

PAOLO BOCCIARELLI is a postdoc researcher at the University of Rome Tor Vergata (Italy). His research interests include business process management, modeling and simulation, model-based engineering applied to the development of software and systems. His email address is paolo.bocciarelli@uniroma2.it.

ANDREA D'AMBROGIO is an associate professor at the University of Rome Tor Vergata (Italy). His research interests are in the fields of model-based systems and software engineering, dependability engineering, distributed and web-based simulation. His email address is dambro@uniroma2.it.