# A BPMN/HLA-Based Methodology for Collaborative Distributed DES

Jalal Possik, Aicha Amrani, Bruno Vallespir, Andrea d'Ambrogio, Grégory Zacharewicz

# A BPMN/HLA-Based Methodology for Collaborative Distributed DES

Jalal Joseph Possik, Aicha Amrani, Bruno Vallespir
IMS- University of Bordeaux
33405 Talence cedex, France
{jalal.possik, aicha.amrani, bruno.vallespir}@u-bordeaux.fr

Andrea D'Ambrogio
Dept. of Enterprise Engineering
University of Rome "Tor Vergata"
Rome, Italy
dambro@uniroma2.it

Gregory Zacharewicz
LG2IP – IMT- Mines Ales
30100 Alès, France
gregory.zacharewicz@mines-ales.fr

***Abstract*— In many domains, Discrete-Event Simulations (DES) are usually used to reproduce the behavior of a certain system or process, where events are processed one after another in chronological and sequential order. Classical DES will no longer be a possible solution for Complex and Large-scale systems, System of Systems (SoS), and Performance Evaluation Systems that compare multiple different simulations running simultaneously in parallel. Advances in network and communications made the Distributed Simulation (DS) approach one of the best solutions for the aforementioned Systems Simulations. One of the challenges faced when developing a DS from DES components is the federation behavior including time management and synchronization between these components. In most of the traditional DES platforms, simulations cannot exchange messages, nor change the configuration at run time. This makes the DES connection and integration very hard and at times, impossible to implement. This article presents the method used to integrate different DES components, using High-Level Architecture (HLA) Evolved Standard, Business Process Model and Notation (BPMN), and Jaamsim, a Java open source DES.**

***Keywords*— *Discrete-Event Simulation (DES), High Level Architecture (HLA), Business Process Model and Notation (BPMN), Distributed Simulation (DS).***

## I. INTRODUCTION

Across different sectors, Modeling and Simulation (M&S) has become one of the best ways to try, explore, analyze and optimize systems structure, behavior and performance prior to the implementation process. Simulation is necessary to deal with real-world uncertainties, variations, and complexities [1].

DES is an effective tool for process improvement [2]. It is a method to simulate real system or process and it is nowadays used in different environments such as manufacturing plants, queuing systems, distribution systems, inventory and delivery systems, health-care, transportation networks, communication networks, and many others [3]. In DES, the simulated system changes state or value at discrete points in time, and the simulation moves from one state to another upon an event occurrence [4].

In some cases, DES alone is not an effective solution. The simulation system must be disassembled into subsystems or nodes in order to be parallelized or distributed on a multiprocessing environment for performance enhancements [5]. In other cases, a collection of interacting simulations is needed to form a more complex system that offers additional functionalities to the existing ones [6]. There are also scenarios where users need to compare many different DESs, and this cannot be run sequentially and needs to be also parallelized or distributed on a network of processors [7]. For all the aforementioned scenarios, time management and synchronization protocols are necessary to avoid timing discrepancies and to ensure precise event interconnections and data communication between subsystems or simulations.

In this article, we will study the case where multiple DESs run in parallel on a network of processors. This work is part of a project developed to test the behavior of Lean tools and techniques during context changes. Lean Manufacturing is a systematic method that uses multiple tools and techniques in order to eliminate wastes from the manufacturing processes, improve inventory, quality, and customer satisfaction [8]. The goal of this project is to guide the companies willing to implement Lean Manufacturing in their industries to choose the right Lean tools that suit their production processes and economic contexts.

We use the HLA Evolved Standard to develop a collaborative distributed DES. HLA is an architecture for interoperation and reuse of interacting simulations. The US Department of Defense first developed this Standard in the 90's and now it became an international IEE standard for distributed simulation. HLA was first used in the military domain. Nowadays, it is applicable in many application domains and across wide simulation areas [9] [10]. HLA has three versions, HLA 1.3 published by the Defense Modeling and Simulation Office in 1998, HLA IEEE 1516-2000 published by IEEE in 2000, and HLA 1516-2010 published by IEEE in 2010 and known as HLA Evolved [11].

BPMN is a business process-modeling standard that offers a graphical notation based on a flowcharting technique. BPMN represents the end-to-end flow of a process. The Business Process Management Initiative (BPMI) developed the Business Process Modeling standard. In 2005, this group merged with the OMG (Object Management Group). In 2011, OMG released the BPMN 2.0 version release and changed the name of the method to Business Process Model and Notation. This Business Process Modeling standard became more detailed by using a richer set of symbols and notations for business process diagrams. The main goal of BPMN is to deliver a standard notation easily readable by non-expert users. In the presented work, we used BPMN to clear up the proposed methodology and simplify the understanding of the integration and collaboration between discrete event simulators.

The technical part of the methodology is also discussed in the present work. In this part, we will discuss the Java implementation of this methodology; the methods used to create, join, or destroy an HLA federation, the publish/subscribe mechanism, interactions/parameters communication, objects/attributes communication, time management, DES configuration. The DES used in this paper is JaamSim. We chose Jaamsim as a Discrete Event Simulator because it is a free and open source simulator written in Java language. Using Eclipse or any other integrated development environment, users are able to add/change Jaamsim objects or add their own code in Java, which is familiar to many programmers [12]. Jaamsim is not developed for distributed simulations and communication to external systems; in this

article, we showed that even with such simulators, we could implement the distributed simulation.

The rest of this paper is structured as follows: Section II reviews the background, related work and relevant contributions to distributed simulation. Section III shows the material and methods used to develop the distributed simulation system. The last section is the conclusion and future work.

## II. Background and Related Work

### A. HLA Standard

Distributed Simulations are very useful in systems engineering. The HLA standard describes a set of services and rules for distributed simulations' implementation. The HLA approach promotes interoperability and reusability. In HLA, the system is considered a federation, a federation is a collection of federates, federates are interconnected through a Run-Time Infrastructure (RTI). The RTI role is to ensure a smooth run of the simulation. Fig. 1 describes the overall architecture of an HLA simulation. In the example we worked on, Jaamsim models are federates that interconnect and connect to other external DESs.

The HLA standard defines:

- Ten architectural rules describing the responsibilities of the entire federation. One of the rules specifies that all data exchanges between federates must go through the RTI.
- A federate interface specification delineating the set of services provided by the RTI. These services are required to manage federates during simulation execution.
- A Federation Object Model (FOM) that describes the shared objects and interactions used to exchange data.

HLA also supports optional services for time management, allowing the coordination of event exchanges between the existing federates. Time management is responsible for the mechanism of regulating the progression of each federate on the federation time axis. Each federate has a logical time. The RTI guarantees the time synchronization of the federates by consistently advancing the logical times of each federate. The logical time is equivalent to the simulation time in the classical literature of DES.

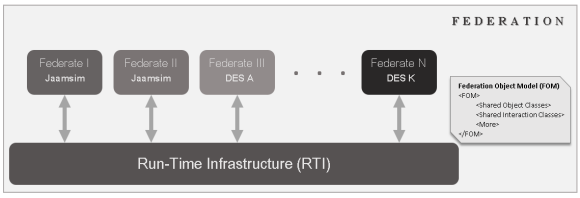


Fig. 1. Global Orchestration

### B. BPMN Standard

BPMN standard is a flowchart method that models the steps of a business process from the ground up. One of the key elements of BPMN is to provide a detailed visual representation of the complete sequence of business activities and information flows. Its purpose is to increase efficiency. It is the enterprise equivalent of the Unified Modeling Language (UML) used in software design. More details about BPMN standard are found in [13].

Fig. 3 represents an Aeronautic Assembly line model. This model is used as the basic model from which multiple scenarios, with different configurations, are generated. Using HLA, we were able to run these scenarios in parallel and compare the outputs to get the most efficient scenarios. This Aeronautic industry produces four references of an aeronautic fastener. The sales order is sent to the cutting shop, where, based on the product reference, raw material metal is cut into specific dimensions. In the cutting shop, setup time is required to change the settings, for the machine to be able to cut a different line of product; if consecutive products have the same reference, no setup time is required. After the cutting process, the product is sent to the treatment shop, where a thin layer of Zinc is added. The product is then delivered to the Assembly shop. The first machine in the assembly shop will also require a setup time in case of a product reference change. The last process in the production line is the Machining process, where, based on the reference type, a gear is installed to the product. This process also requires a changeover time. The finished good is finally delivered to the warehouse and is ready for delivery process.

### C. Related Work

This Section reviews the existing related work and relevant contributions to the distributed simulation domain.

Jayadev Misra proposed a solution that partitioned a simulation to multiple components running on different processors and presented different techniques for deadlock detection and avoidance [5]. The main objective of his work was to enhance the simulation performance.

In [14], the authors proposed a model-driven method that allowed the generation of an HLA-based code from a BPMN model using a chain of models to text transformations. BPMN and HLA have different techniques and objectives. The authors were able to combine both standards into a model-driven method that makes the distributed simulation techniques easier to understand for developers and engineers. Others introduced a BPMN extension to address data structures' definition associated with the information exchanged during the execution of the Business Process collaborations [15]. Their paper defines a model-driven framework for DS based analysis of Business Process in order to fill the gap between HLA and BPMN.

In [16], using HLA standard, authors developed an Enterprise Operating System architecture to control and monitor the enterprise's operations. Authors used HLA standard to integrate large distributed environments and synchronize data and actions among existing federates. The authors, in [17], presented a reference architecture to inter-integrate manufacturing distributed simulations and connect them to other manufacturing software and data repositories.

In a previous study, we developed a Co-Simulation system for industries and enterprises based on MECSYCO (Multi-agent Environment for Complex System CO-simulation), in which users specify the simulations input on a web interface then launch the system from the interface [18]. The system initializes the simulators and runs the simulations simultaneously in parallel. The respective output results appear in a graphical presentation during the simulation run-time process. Each of these simulations represents a different setup or configuration for the same aeronautic assembly line. By varying the inputs, users can easily choose the best configuration/setup that suits the production assembly line.

The purpose of our work is to extend the scope of the previous studies by proposing a methodology to connect different separated DESs using HLA publish/subscribe mechanism and time management. This methodology will be

discussed using BPMN for easy access to non-expert users. For the technical savvy, the java coding that has been used during the implementation process will be discussed.

## III. Material And Methods

In this part, we will use an example connecting three federates, Master federate, Scenario 1 federate, and Scenario 2 federate. Scenario 1 represents the DES model of Fig. 3; Scenario 2 represents the same model with some modifications in the production design. The aim of this distributed simulation is to determine how these scenarios react to changes in attributes and parameters, and compute the best behavior scenarios. These federates are linked via the RTI constituting a federation. These federation elements use a common Federation Object Model (FOM), an XML file that defines the objects/attributes and the interactions/parameters of the federation. We used the Java library of Pitch pRTI platform [19] to develop the following part.

### A. Federation related Services

First, a federation should be created. As per Fig. 2, when the Master platform starts the simulation, it creates the Federation by calling the RTI Ambassador; this HLA service creates the Federation using a unique Federation name and links it to its corresponding FOM XML file. The method in *RTIAmbassador* class used to create the federation execution is *RTIAmbassador.createFederationExecution("Federation Name", xmlFOMfile)*. After creating the federation, the Master, as a federate, joins the federation using the *RTIAmbassador.joinFederationExecution("Federate Name", "Federate Type", "Federation name to join")* method. Next, the Master launches the two DES scenarios that also join the created federation using the same method and parameters.

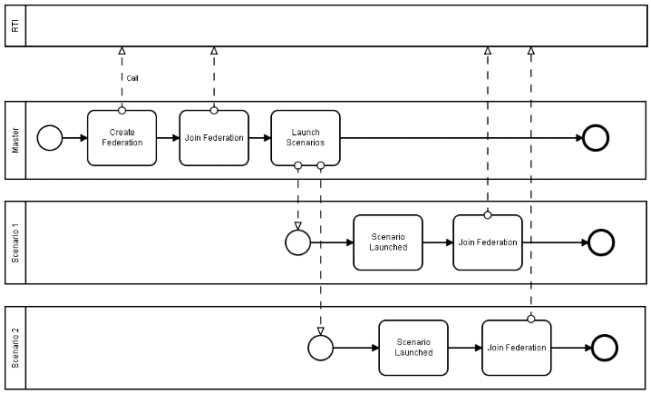


Fig. 2. Create/Join Federation

In the example we developed, as per Fig. 4, the object classes created are "Scenario" and "Machine". The Scenario Object Class has the following attributes: Name, SimTime, Run Duration, Material Buffer, and Stock Keeping Unit (SKU). "Name" represents the Scenario Name, "SimTime" is the Simulation Time of the DES during the run time, "RunDuration" determines in years the duration of the simulation scenario, "MaterialBuffer" represents the number of raw materials waiting for the production process, and "SKU" represents the number of units remaining in stock. The Machine Object Class attributes include the number of goods processed, the number of goods in progress, the working time of the machine, the number of workers needed per machine, and the Work In Progress (WIP) that represents the in-process inventory. We subsequently created eight interactions: Scenario Load, Scenario Loaded, Scenario Error, Start/Pause/Stop Simulation, Order Demand Increase, and Order Demand Decrease. For each interaction, we have one or more parameters listed as noted in Fig. 4. The objects/attributes and the interactions/parameters sharing mechanisms (Publish "p", Subscribe "s", Publish/Subscribe "ps") are also listed in Fig. 4.

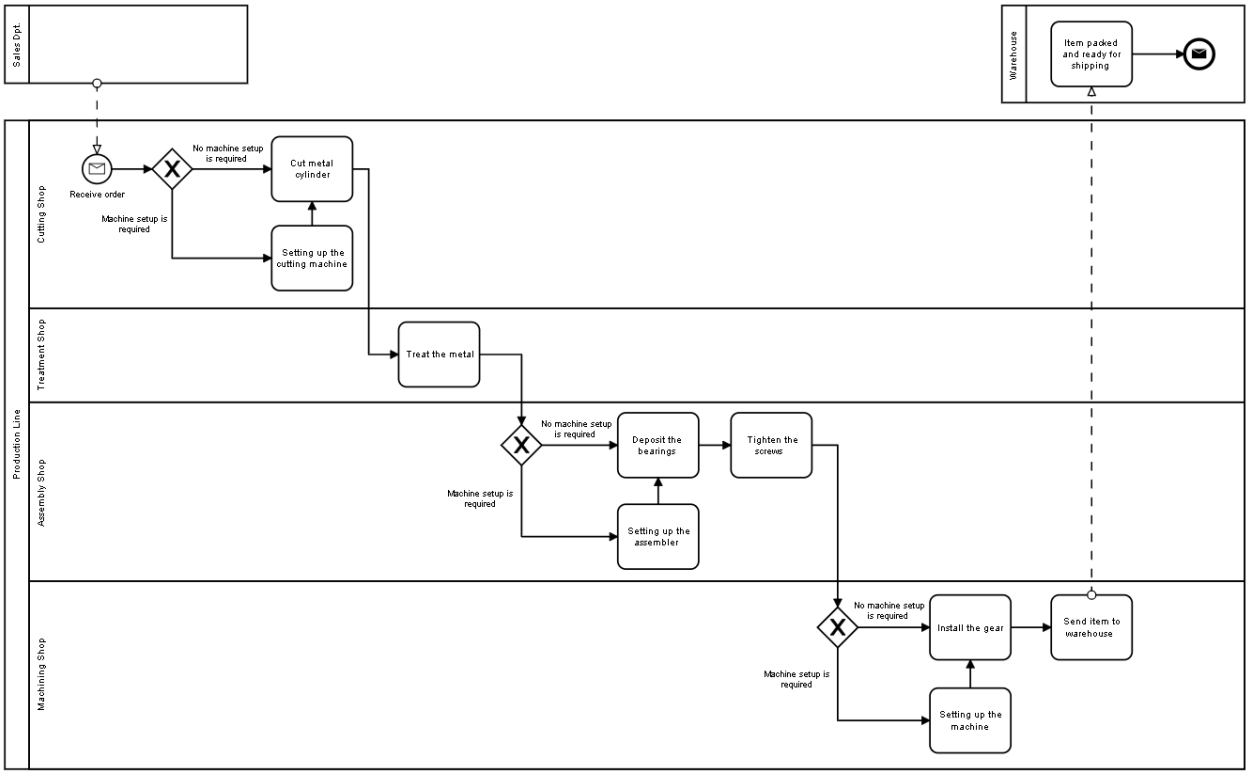


Fig. 3. Aeronautic assembly line model

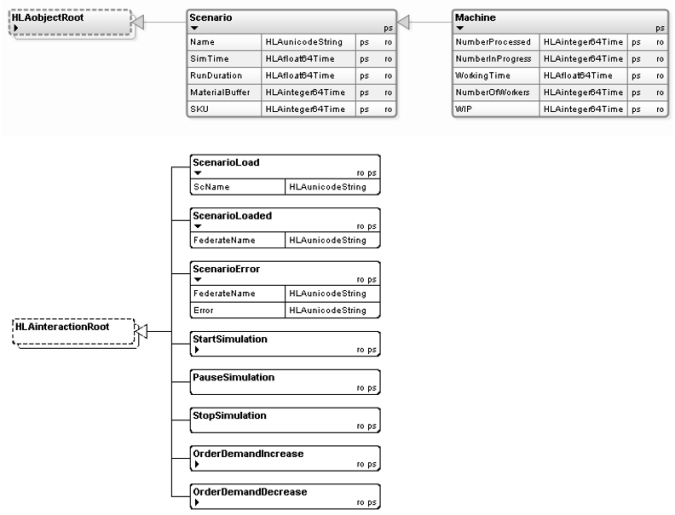


Fig. 4. Federation Object Model (FOM)

## B. Declaration Management Services

Fig. 6 illustrates the steps used to select the Publish/Subscribe interests of the Object Classes. Each object should first get the handle for the actual object class in order to be published. The method used for this service is *RTIAmbassador.getObjectClassHandle("Object Class")*. In our example, "Object Class" could be "Scenario" or "Machine". The next step involves creating an Attribute Handle Set using the method *create()* in the *AttributeHandleSetFactory* class. Next, one should get the Attribute Handle using *RTIAmbassador.getAttributeClassHandle("Object Class Handle", "Attribute")* method. One of the attributes could be "Name" that exists in the Object Class "Scenario". Next, the Attribute Handle Set should be added using the method *add()* in the *AttributeHandleSet* class. The last step in the declaration part is to Publish/Subscribe the AttributeHandleSet of the Object Class using *RTIAmbassador.publishObjectClassAttributes("Object Class Handle", "Attribute Handle Set")* and *RTIAmbassador.subscribeObjectClassAttributes("Object Class Handle", "Attribute Handle Set")* methods.

After the publish Object Class Attribute, a callback from the RTI accesses the *startRegistrationForObjectClass("ObjectClass Handle")* method.

Registering the Publish/Subscribe for the interaction classes is more straightforward. First, one should get the Interaction Class Handle using the method: *RTIAmbassador.getInteractionClassHandle("Interaction Class")* then get the Parameter Handle using *RTIAmbassador.getParameterHandle("Interaction Class Handle", "Parameter")* method.

## C. Object Management

Fig. 5 shows the required services to register/discover object instances. *RTIAmbassador.registerObjectInstance("Object Class Handle", "the Object Name")* is required to register the object instance. After the registration process, a callback is sent to the other existing federates, accessing the method *discoverObjectInstance("Object Instance Handle", "Object Class Handle", "the Object Name"). turnUpdatesOnForObectInstance()* callback method is accessed in the federate that registered the object instance.

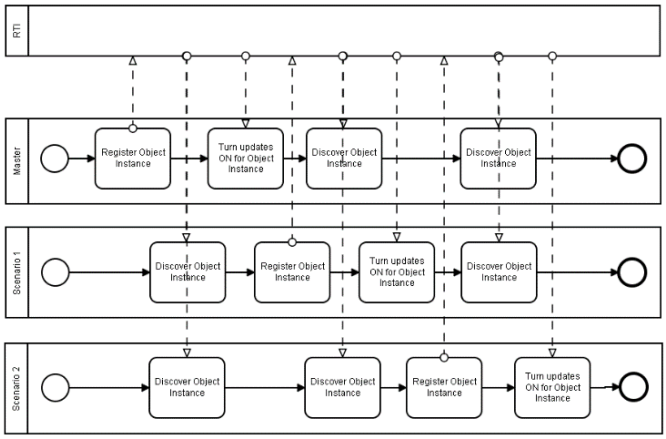


Fig. 5. Object Instance Registration

The method *updateAttributeValues()* of the RTIAmbassador class is used to update the attributes related to the registered object instance. After the attribute update, *ReflectAttributeValues()* callback method is accessed in the other existing federates as per Fig. 7. As for the interactions, the same concept is used with the *sendInteraction()* method and *receiveInteraction()* callback method.

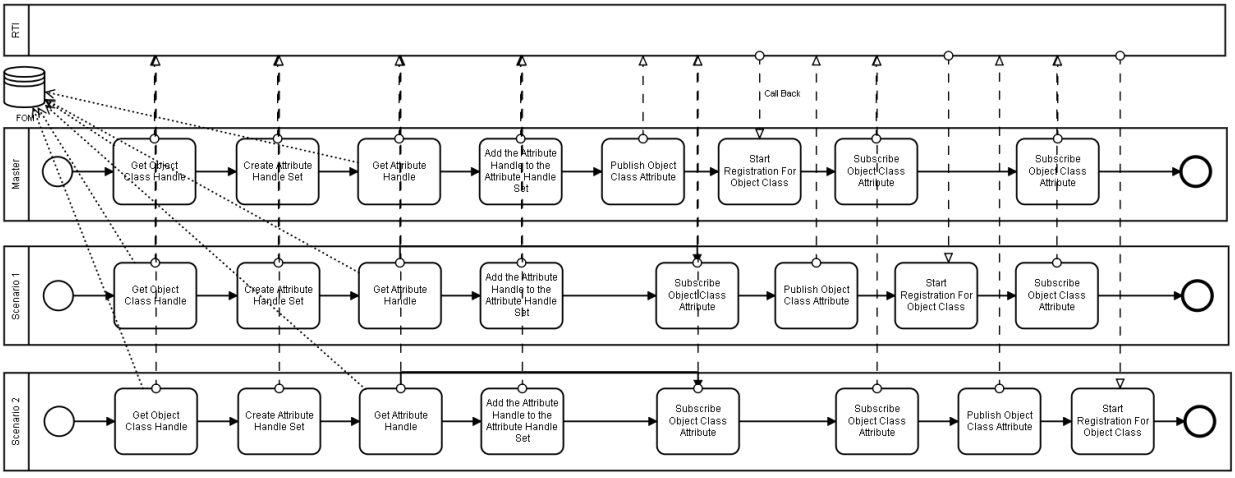


Fig. 6. Object/Attribute Declaration

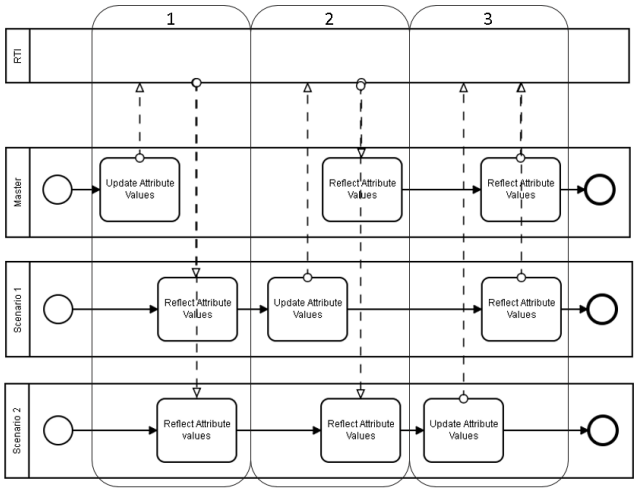


Fig. 7. Objects/Attributes update

### D. Time Management

By default, federates have the time regulating service and the time constrained service disabled. To enable the time management services, a federate requests to be a time regulating federate using the method *EnableTimeRegulation(),* or to be time constrained using the method *EnableTimeConstrained()*. A federate could be time regulating/constrained at the same time. When these two methods are used, the Federate Ambassador calls back the *TimeRegulationEnabled()* and *TimeConstrainedEnabled()* methods. In this work, all DESs have the time regulating/constrained enabled.

The DES Federates in this paper are developed using Jaamsim. Jaamsim is not designed for communications to external systems and not fitted for DS; it is viewed as a black box simulator. As it is an open source software, we were able to run the *startSimulation()* and *pauseSimulation()* methods in Jaamsim to Start/Pause the simulations. These methods are linked to the aforementioned interaction's parameters StartSimulation and PauseSimulation. As per Fig. 8, when the DES runs, Jaamsim will add, with each event, a new output line to the log file issued using the Entity Logger option. When an event is processed, the federate will ask for time advancement to send new events using the *nextEventRequest()* method. The RTI ensures that it will not deliver any message with a Time Stamped Order less than the lookahead time and the federate actual time combined, the lookahead being the time delay that cannot be exceeded between simulations. The larger the lookahead value, the longer it takes for messages to reach the other federates. With a zero lookahead, messages should reach the other federates instantly. These HLA services are used to avoid out of order messages delivery.

If a time advance is granted to the federate, we test the value of X:

- A value that is equal to one means that Jaamsim was already running and there is no need to run it again.
- A value that is equal to zero means that Jaamsim was paused and we need to start/resume the simulation again, then assign the value of one to the variable X.

If the "LogicalTimeAlreadyPassed" exception is raised, it implies that Jaamsim simulation should be paused:

- A value of X that is equal to zero means that Jaamsim was already paused and there is no need to pause it again.
- A Value of X that is equal to one means that Jaamsim was running and we have to pause it, then assign the value of zero to the variable X.

This way, all Jaamsim DESs will run in parallel sending the output of the simulations almost simultaneously depending on the lookahead value.

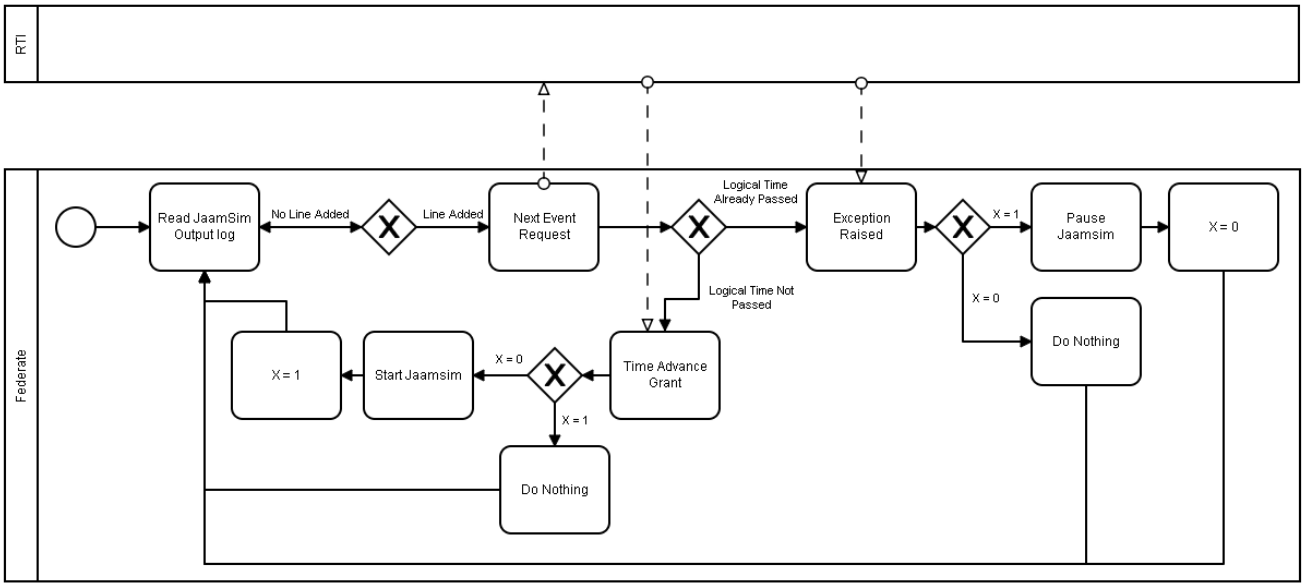


Fig. 8. Time Advancement, Event-Based

## IV. Conclusion and Future Work

HLA standard does not propose nor precise any particular language to describe the behavioral process of the federation, i.e. group of federates, before to setup and implement. This IEEE standard also does not point out any specific programing language or software use. In this article, we described the methodology followed to define the desired interconnections and data exchange between DESs while running simulations

in parallel on a network of processors. The implementation steps are explained using BPMN and the Java library of pRTI. BPMN provides a standard straightforward notation easily readable by non-expert users while Java language describes the technical implementation part.

Many industries are inefficiently implementing Lean tools in their organizations and are facing quality, management, financial, and other failures in their Lean implementations. This simulation system aims to conduct manufacturing industries in choosing the right Lean Manufacturing tools that lead the implementation to success with explaining the planned behavior. Our future work is to develop more models, contexts, and scenarios using this developed distributed simulator in order to obtain relative hypothesis that contributes, helps, and conducts Lean tools implementations in the production domain.

## REFERENCES


[1] T. I. Ören and B. P. Zeigler, "Concepts for advanced simulation methodologies," *SIMULATION,* vol. 32, pp. 69-82, 1979.

[2] C. D. Barnes and K. R. Laughery, "Advanced uses for Micro Saint simulation software," in *1998 Winter Simulation Conference. Proceedings (Cat. No.98CH36274)*, 1998, pp. 271-274 vol.1.

[3] G. S. Fishman, *Discrete-event simulation: modeling, programming, and analysis*: Springer Science & Business Media, 2013.

[4] R. M. Fujimoto, "Parallel discrete event simulation," *Commun. ACM,* vol. 33, pp. 30-53, 1990.

[5] J. Misra, "Distributed discrete-event simulation," *ACM Computing Surveys (CSUR),* vol. 18, pp. 39-65, 1986.

[6] A. Falcone, A. Garro, A. DxAmbrogio, and A. Giglio, "Using BPMN and HLA for SoS engineering: lessons learned and future directions," *2018 IEEE International Systems Engineering Symposium (ISSE),* pp. 1-8, 2018.

[7] J. J. Possik, A. A. Amrani, and G. Zacharewicz, "Development of a co-simulation system as a decision-aid in Lean tools implementation," presented at the Proceedings of the 50th Computer Simulation Conference, Bordeaux, France, 2018.

[8] A. Amrani, J. Possik, Y. Ducq, and G. Zacharewicz, "Contribution to a Lean Maturity Evaluation: Leanness Metrics Calculation," presented at the PMA 2018 - Performance Measurement and Management in a Globally Networked World, Warsaw, Poland, 2018.

[9] J. S. Dahmann, F. Kuhl, and R. Weatherly, "Standards for Simulation: As Simple As Possible But Not Simpler The High Level Architecture For Simulation," *SIMULATION,* vol. 71, pp. 378-387, 1998.

[10] F. Yilmaz, U. Durak, K. Taylan, and H. Oğuztüzün, "Adapting functional mockup units for HLA-compliant distributed simulation," in *Proceedings of the 10 th International Modelica Conference; March 10-12; 2014; Lund; Sweden*, 2014, pp. 247-257.

[11] "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)-- Framework and Rules," *IEEE Std 1516-2010 (Revision of IEEE Std 1516-2000),* pp. 1-38, 2010.

[12] D. H. King and H. S. Harrison, "Open-source simulation software "JaamSim"," in *2013 Winter Simulations Conference (WSC)*, 2013, pp. 2163-2171.

[13] O. M. Group. (1997-2019, March 5, 2019 ). *Business Process Model and Notation*. Available: http://www.bpmn.org/

[14] A. Garro, A. Falcone, A. D'Ambrogio, and A. Giglio, "A Model-Driven Method to Enable the Distributed Simulation of BPMN Models," *2018 IEEE 27th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE),* pp. 121-126, 2018.

[15] P. Bocciarelli, A. D. Ambrogio, E. Paglia, and 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.

[16] J. Youssef, G. Zacharewicz, D. Chen, and F. Vernadat, *EOS: enterprise operating systems*, 2017.

[17] C. McLean and F. Riddick, "Simulation in the international IMS MISSION project: the IMS MISSION architecture for distributed manufacturing simulation," presented at the Proceedings of the 32nd conference on Winter simulation, Orlando, Florida, 2000.

[18] J. Possik, A. Amrani, and G. Zacharewicz, "WIP: Co-simulation system serving the configuration of lean tools for a manufacturing assembly line," presented at the Works in Progress Symposium, WIP 2018, Part of the 2018 Spring Simulation Multiconference, SpringSim 2018, Baltimore, United States, 2018.

[19] P. Technologies. (March 2, 2019). *Pitch pRTI*. Available: http://pitchtechnologies.com/products/prti/