Business models for distributed-simulation orchestration and risk management

Simon Gorecki, Jalal Possik, Grégory Zacharewicz, Yves Ducq, Nicolas Perry

information

Article

Business Models for Distributed-Simulation Orchestration and Risk Management

Simon Gorecki [1], Jalal Possik [2,3], Gregory Zacharewicz [4,*], Yves Ducq [5] and Nicolas Perry [6]

1 Groupe de Recherche en Economie Théorique et Appliquée (GREThA) UMR CNRS 5113, University of Bordeaux, 33608 Pessac, France; simon.gorecki@u-bordeaux.fr
2 Advanced Disaster, Emergency and Rapid Response Simulation (ADERSIM), York University, Toronto, ON M3J 1P3, Canada; jpossik@yorku.ca or jalal.possik@lau.edu.lb
3 Department of Computer Science and Mathematics, Lebanese American University, Block A, Byblos 1401 2010, Lebanon
4 Laboratoire des Sciences des Risques (LSR), Institut Mines-Telecom (IMT) Mines Ales, CEDEX, 30319 Alès, France
5 Intégration du Matériau au Système (IMS) UMR CNRS 5218, University of Bordeaux, CEDEX, 33405 Talence, France; yves.ducq@u-bordeaux.fr
6 Arts et Métiers Bordeaux-Talence—I2M Bordeaux UMR 5295, Esplanade des Arts et Métiers, 33400 Talence, France; nicolas.perry@ensam.eu
* Correspondence: gregory.zacharewicz@mines-ales.fr; Tel.: +33-(0)4-6678-5000

Abstract: Nowadays, industries are implementing heterogeneous systems from different domains, backgrounds, and operating systems. Manufacturing systems are becoming more and more complex, which forces engineers to manage the complexity in several aspects. Technical complexities bring interoperability, risk management, and hazards issues that must be taken into consideration, from the business model design to the technical implementation. To solve the complexities and the incompatibilities between heterogeneous components, several distributed and cosimulation standards and tools can be used for data exchange and interconnection. High-level architecture (HLA) and functional mockup interface (FMI) are the main international standards used for distributed and cosimulation. HLA is mainly used in academic and defense domains while FMI is mostly used in industry. In this article, we propose an HLA/FMI implementation with a connection to an external business process-modeling tool called Papyrus. Papyrus is configured as a master federate that orchestrates the subsimulations based on the above standards. The developed framework is integrated with external heterogeneous components through an FMI interface. This framework is developed with the aim of bringing interoperability to a system used in a power generation company.

Keywords: HLA; FMI; risk management; interoperability; modeling and simulation; Papyrus; JaamSim; Industry 4.0

Citation: Gorecki, S.; Possik, J.; Zacharewicz, G.; Ducq, Y.; Perry, N. Business Models for Distributed-Simulation Orchestration and Risk Management. *Information* **2021**, *12*, 71. https://doi.org/10.3390/info12020071

Academic Editor: Luis Rabelo
Received: 6 January 2021
Accepted: 29 January 2021
Published: 7 February 2021

1. Introduction

Industry 4.0 is considered the fourth industrial revolution. It follows the arrival of production automation [1]. The concept of Industry 4.0 emerged at the beginning of the 21st century. Kagermarin et al. [2] refer to the digitization of production and describe the vision of an intelligent factory, characterized by the networking of all production processes in order to operate interactively and to increase productivity and efficiency in the use of resources. Companies are facing increasing competition in their markets, leading managers to constantly seek new methods, approaches, and strategies to improve operational flexibility, efficiency, innovation, and responsiveness through an accurate control supported by 4.0 technologies [3,4]. The use of digital approaches anticipates the development and control of intelligent production systems by contributing, for example, to the control of production flow's behavior.

In the context of Industry 4.0, the need for modeling and simulation (M&S) becomes more and more important as it allows users to manipulate virtual prototypes to easily reproduce experiments in different situations while working on accessible virtual models [5]. This approach to modeling and simulation relates to the four pillars of Industry 4.0: (1) Digitization: the interconnection of systems and the management of large amounts of data allow the enrichment of a model's input data; (2) Flexibility: the simulation of an industrial process allows an efficient representation and analysis of its operations. This results in an increased ability to make decisions and adapt to its environment; (3) Logistic tools: An M&S tool can be connected to an information system as a decision support module; and (4) Human–machine interface: as in all operations involving human and machine resources, the need for efficiency and ease of use are necessary.

The Industry 4.0 business models rely on data to control processes. The collected and predicted data help in analyzing and understanding complex systems and facilitate the control of their operation. The ability to virtualize workspaces or systems bridges the gap between the physical world driven by machines and the virtual world. According to Madni et al. [6], the digital twin (DT) creates the ideal environment for data collection on all aspects of the manufacturing process for analysis and simulation. When data are accurately collected and a digital twin is designed, systems integrators, data analysts, and others can use it to drive business policies and improve decision-making processes. According to Mangles [7], by 2021, nearly half of the large industrial companies will use DT to assess the technical performance and risks of their systems. However, as systems become larger and more complex, the difficulty of simulating these systems and their associated risks increases proportionately. As stated by Taylor et al. [8], when the model is complex and requires high-performance resources, the classical simulation approach becomes insufficient. The model execution must be divided and distributed among a large number of processors or machines, based on appropriate modeling tools. Therefore, distributed simulation (DS) becomes essential: one simulation is divided into multiple subsimulations (or models) of a complex system. Each autonomous simulation can be executed on a different computer and sometimes on a different LAN (local area network). From a general point of view, this solution divides the complex problems into simpler subproblems, but also raises interoperability issues.

The model we propose in this project takes into consideration the notion of complexity distribution and risk management. Thus, we propose the association of three main contributions:

First, the interoperability between heterogeneous simulation components to allow component reuse, load balancing between separate components, and reduction of development costs. High-level architecture (HLA) and functional mockup interface (FMI) are both standards for distributed simulation providing interfaces to solve interoperability issues between tools and simulations. Creating a bridge between these two standards allows us to connect components from several different domains. The bridging tool used to interconnect these two standards is Papyrus [9], an open-source modeling and simulation tool;

Second, throughout its lifecycle, every project is subject to numerous internal or external risks and hazards. The management of these risks is critical for the success of the project. In this project, the process and risk are both modeled and taken into consideration. The data input of a simulation is divided into two main parts. On the one hand is an industrial process model with its input data set. On the other hand is a model describing the risks and hazards affecting the simulated process;

Third, an orchestrator component that manages the interconnected distributed simulations. Moreover, a model describing the business process is developed. This model refers to the subcomponents that represent the functional or technical processes.

The rest of this article is structured as follows: A literature review section analyzes the state of the art. The next section shows the materials and methods used to develop the distributed-simulation system. Sections 4 and 5 explain the development steps and mechanisms. The risk-management implementation is detailed in Section 6. Section 7

describes the case study and the simulation results. The last section is the conclusion and perspective that presents the general outcome of the work.

2. Literature Review

The industrial processes of complex systems require a method and a framework to interconnect a set of heterogeneous components operating on different platforms and operating systems [10]. Manufacturing modeling and simulation systems are built from the set of processes, the set of exchanged data, the interconnection of services, and the collaboration between supply chain processes. Therefore, it requires a modeling language to capture all the elements of the system. Then, at the time of execution, a synchronization of the data exchange is necessary. Moreover, the interoperability between components is required during the simulation run. The literature details interesting contributions to enterprise modeling, interoperability, and distributed-simulation standards. These areas are recalled in the following section.

In [11], Falcone et al. propose the use of the business process management and notation (BPMN) standard to model and better understand the internal mechanisms of the standard. Their approach is divided into two parts: the first aims to use the standard to model the internal functioning mechanisms of a federation; the second aims to use the standard to model the different phases of the whole HLA federation life cycle. In [12] and [13], the authors present a collaborative framework based on business process management and notation (BPMN) as well as a middleware that implements the functionalities of BPMN by using basic functions of the HLA standard. BPMN enables the definition and simulation of workflow engines using HLA. Taylor et al. [14] presented a proposal that improves a commercial simulation tool (commercial-off-the-shelf simulation packages (CSPs)) using distributed-simulation standards. The authors describe the communication between existing business process models (modeled in BPMN) through a network of HLA federates.

Bocciarelli et al. [15] presented a method for the automated transformation of BPMN processes into EQN (extended queuing network) models. The EQN models are then run as distributed simulations in an e-commerce scenario. This research converts the BPMN into executable code. Bazoun et al. [16] presented a tool called "SLMToolBOX". This developed tool is a graphical modeler, a model transformer, and a simulation engine capable of converting a BPMN diagram into DEVS models to simulate the behavior of a process. The DEVS models can be transformed and executed as components of an HLA federation [17]. Ultimately, Lee [10] developed a workflow simulation system that combines both BPMN and HLA functionalities. This system allows experts to describe an industrial system in the form of BPMN diagrams. The resulting model will be converted into a set of HLA components that will be simulated using HLA–BPMN middleware.

After the above analysis, some limits can be identified. According to the HLA standard, federation members are autonomous. In a reusability approach, federation members should be reworked (redefine the structure, code, parameters, exchange protocols, etc.), which can be problematic in an environment with high component reuse. In the literature, some articles have already highlighted the benefits that a graphical modeling language would bring to the HLA standard. Despite this observation, we note that few studies have attempted to use a graphical language to orchestrate a distributed HLA simulation. Representing the federate execution sequence and their interactions by a graphical language would increase the potential for reusability and modularity of the standard. The above studies make it possible to use various methods to run distributed workflow languages such as BPMN via the HLA standard. Some studies use the graphical standards, such as BPMN, to represent the internal mechanisms and the interaction of the HLA simulation components. However, one can notice the need of a graphical model that orchestrates and manages a distributed simulation.

In state of the art, various authors have attempted approaches to reduce or resolve interoperability problems specific to the cosimulation standard, in particular FMI. Some

are attempting to create linkages between the FMI and HLA, such as Yilmaz et al. [18] who presented in their paper a method to automatically generate an FMU (functional mockup units) entity able to join an HLA federation as member. A wrapper reads the FMU specification and generates the HLA layer. To demonstrate the FMUFd (FMU-federate) usage, authors developed a simple DS example with MAK HLA Run Time Infrastructure (RTI). In this paper, time synchronization between the two standards (FMI & HLA) is solved by updating the federate time at each running step of the FMU model.

Falcone et al. [19] presented an integrated way to merge the functional mockup interface (FMI) and the IEEE 1516—High Level Architecture standard. The authors propose two approaches in order to merge the standards. The first approach is an adapter-based approach where a hybrid federate is created. The federate contains an FMU (the behavior of the component to simulate and its solver) coupled with an adapter managing all interactions between HLA-RTI and the FMU component. The second approach is a mediator-based approach where a set of FMUs, an HLA federate that can use FMU to simulate a specific component, and a mediator layer are created in order to coordinate the behavior of the whole system.

Awais et al. [20] proposed how the strengths of both the HLA and the FMIs can be utilized to realize a distributed hybrid simulation platform. Two different algorithms were proposed. To demonstrate the correctness of algorithms and their performance comparison, a simulation example was chosen from the domain of complex energy systems.

Most risk-management tools are derived from formal methods such as: Obeng, risk checklist, brainstorming [21], ISSRM, and influence diagrams [22]. They are frequently implemented in serious games in order to make users aware of the raised problems. Pontakorn et al. [23] presented a serious game that helps to simulate risk management in software engineering. During the simulation, the participants followed the principles of risk management and aimed to achieve their objectives while saving their expenses. In addition, Jurjens et al. [24] observed that security problems often arise during the definition of the business processes. It was on the basis of this observation that Olga Altuhhova et al. [25] stated how security concerns are neglected in business modeling processes, specifically in the modeling information systems. In [25], the objective was to use the BPMN standard to model the security of business processes and to suggest extensions to the language convergence toward security risk management. In order to achieve the above, authors selected the IS security risk management (ISSRM) domain model and aligned it semantically with the BPMN standard through the development of a dedicated extension. Ralf Laue et al. [26] presented an evaluation of the business process simulation standard (BPSIM) specification. For simulation purposes, a model must contain a certain amount of information (tasks, resource management, start and end events, etc.) [24]. The BPSIM standard [27] is an extension of the BPMN language for process simulation. In BPSIM, parameters are added to an XML file to allow the simulation of the BPMN process. Ralf Laue et al. precised that the language allows modeling independently of simulation tools; however, in practical tests they find that some tools that offer an implementation of the standard only implement a subset of the standard. In addition, some implementations provide useful but proprietary extensions. Since the specification is relatively recent, user interface problems appear (such as the need to go through the editing of text files to simulate certain resources). The authors also point out that BPSIM does not grant an accurate description of resources as usually achieved in traditional simulation tools.

In [28], Fan et al. focused on ecological risk assessment using a probabilistic approach to characterize risk and deal with uncertainty using Monte Carlo simulations. This simulation relies on data from the American army to evaluate the probabilistic distributions of risk parameters related to the exposure to depleted uranium. Haiyi Lu [29] developed a tool designed to study UML models that deal with environmental risks. In [30], Bixio et al. presented the case of the Belgian wastewater treatment plants which faced many hazards during a structure renovation process. The article of Bixio et al. describes a method that combines a probabilistic (Monte Carlo) simulation with deterministic simulations.

Originating from the IO-SUIT prototypes [31] and Pro-(R)isk [32], the RIO-Suit tool inherits a modular architecture. Common to several research projects, the tool supports risk management and interoperability within collaborative organizations (RIOs). Via a module that manages different situations, the user can declare the model of a collaborative system in the form of graphs (Neo4J) that include a set of objectives related to the collaboration between organizations [33].

In a high-risk context, uncertainties must be taken into account from the early stages of the modeling process—the sooner, the better to avoid unwanted events. Taking into account risk management in simulation involves two difficulties:

- Adding a representation of risks with their parameters (impacts, occurrences, probabilities, and side effects) to a system drastically increases the difficulty of the modeling process. Due to this complexity, some authors propose the outsourcing of the hazards. However, the model will no longer be autonomous. Thus, the user will not be able to test a risk-free scenario;
- Simulating a global system implies access and communication with all the subsystems. In the majority of cases, the different subsystems are from vast and varied domains. Therefore, there is a significant problem of the components' homogeneity.

In the sections below, we provide methods and solutions to the above-mentioned difficulties faced by the engineers. We propose a method separating the input data from the risk-management parameters. Risk parameters are linked to a user-configurable database that may or may not be connected to the main model. This database can be updated during the simulation run. Moreover, distributed-simulation standards (HLA and FMI) are used to solve the heterogeneity issue.

3. Materials and Methods

This section presents the materials and methods followed in this project to propose both, the conceptual and technical implementations of the distributed-simulation system.

3.1. Process-Modeling Workflow

Workflow is the modeling and computer-assisted management of all the tasks and the various actors involved in the realization of a business process [34]. The Workflow Management Coalition (WfMC) has developed standards in association with the leading actors of the domain [17]. Those standards define a workflow reference model that represents the workflow components and contains the process definition tool, the administrator tool, the workflow client application, the invoked applications, and the link between other workflow environments. A workflow defines a set of processes that models business activities. A process consists of a sequence of procedures, named tasks, and logical expressions or controllers that describe the pathways for the items. A workflow can be modeled by a graphical representation in which tasks are represented with rectangles, controllers with nodes, and arrows to determine the task flows. There are many environments for the representation and simulation of workflows. Nevertheless, they are based only on ad-hoc execution software engines; consequently, they do not have the advantage of the concepts offered by discrete event simulation theory [35]. In fact, this theory separates the modeling phase from the simulation, allowing the reuse of the validated representation in other domains.

3.2. Distributed Simulation

In the literature, the concept of distributed simulation can be associated with cosimulation standards. In this chapter, we will first examine the theory of the HLA distributed-simulation standard and then a more recent European alternative: the functional model interface (FMI) as a cosimulation standard. Finally, we will see the main implementations of these two standards, their advantages, and disadvantages.

3.2.1. High-Level Architecture

The HLA standard originated in 1983 as the SIMNET project supported by DARPA (defense advanced research projects agency). SIMNET was the first distributed system for virtual reality applications and simulations. Then, over time and through several evolutions, it was adopted by the American Department of Defense (DoD). In 2000, the standard was refined and adopted by the IEEE and named "HLA IEEE 1516". In 2010, a new version with some updates/improvements became available. This new version is known as "HLA Evolved" [36]. The development of the new version of HLA, named HLA 4.0, started in January 2016 by SISO and is currently in progress. It will be probably called HLA 1516-20XX (HLA 4).

The HLA standard offers an architecture that facilitates the reuse of distributed components by simplifying interoperability issues. It also describes a set of services and rules for the implementation of these components. However, it does not propose any language (textual or graphical) to describe the choreography of its components during the simulation execution [35].

HLA standard has synchronization advantages. It enables the management of data exchange between simulations: messages are sent right on time, in the right order, and without violating causal constraints. HLA and its RTI (runtime infrastructure) provide various mechanisms for time management and synchronization. In HLA standard, the global simulation is called "federation" composed of different components called "federates". A federate can be a simulation system or an interface to a physical system connected to the RTI in order to send/receive data and be synchronized with other connected federates. The federation object model (FOM) is an XML file that contains a description of the elements exchanged between the federates connected to the same federation. Data can be exchanged as objects/attributes (persistent data), or as interactions/parameters (ephemeral data).

3.2.2. Functional Mockup Interface

FMI is a European standard designed in 2011 by MODELISAR [37,38] to improve the design of systems and software embedded in vehicles. The standard is designed and developed for industrial applications, in particular for cyber–physical systems, to facilitate data exchanges. One of its objectives is to simplify collaboration between industrial partners, by providing them with a standard system to exchange models, while guaranteeing the security of their industrial systems. The standard supports two interfaces for two possible uses: model exchange (ME) and cosimulation (CO) [39].

FMI offers a set of generic rules and functions to manipulate specific components called FMUs (functional mockup units). An FMU can therefore be:

- A component intended for cosimulation;
- A component for model exchange.

An FMU component is a compressed file containing:

- A cosimulation interface: a set of C functions for the execution control and the details on exchanged input/output data;
- A cosimulation description schema: it defines the structure and content of the federate using an XML file. This XML file, specific to the execution process, contains "static" information about the model (input and output variables, parameters, etc.) and the solver/simulator. "Flags" characterize the execution support into advanced decision algorithms that use variable communication time stamps, higher-order signal extrapolation, etc. Note that the FMI standard can only support standard variables.
- It can also contain FMU source and header files. Those files are optional.

According to the FMI standard, a master should load the FMU by reading its XML file and managing its local time. FMI allows a hidden implementation to secure intellectual property.

3.3. Simulation Tools

A simulation is designed to study the result(s) of an action on a specific element(s) without performing the experiment on the real element(s). In most of the cases, it is a numerical calculation system compatible with a formalism (in our case, event driven models) capable of executing a given model and predicting its behavior. In this paper, we are going to model and simulate different subjects. On the one hand is modeling and simulating a business process using Papyrus [9]. On the other hand is modeling and simulating industrial components using JaamSim [40].

3.3.1. Process Modeling with Papyrus

Papyrus is a modular tool based on Eclipse IDE that includes a modeling module by default, as well as an execution engine called "Moka". It has a graphical editing tool that allows the editing of all types of UML 2.0 diagrams. It provides an integrating UML profile mechanism that allows extensions definition to basic standards for better modeling flexibility. The UML profiles are the mechanisms specific to the UML 2.0 standard that allow the extension of the metaclasses related to the existing metamodels. By doing this, the UML profiles can be adapted to any context.

Papyrus also has a UML model execution plugin: Moka, an execution engine compatible with the OMG foundational subset for executable UML models (fUML) standard [41]. The objective of this standard is to serve as an intermediary between "surface" models (used to describe business processes) and "platform" languages (can be executed on machines). fUML is therefore a pivot language necessary for the simulation of UML models.

Papyrus is widely used by the community. It is designed with a very high modularity, easily extensible, and simple to use. Using this tool, the engineers can represent business models and connect them to external tools.

3.3.2. Industrial Simulation Using JaamSim

JaamSim is a free Java-based and open-source simulation package developed by an Australian Engineering company named Ausenco. The main feature that differentiates JaamSim from other off-the-shelf commercial DES (discrete event simulators) is that users can design and make their own high-level object palettes [40]. Users can use the GUI of JaamSim to add entities and create the simulation model or write/edit a configuration file (.cfg) in which all entities/objects can be added and configured. Some users may prefer the graphical user interface (GUI) to drag and drop their entities and configure them on the graphical user interface (GUI). Others, especially programmers, will find it faster and easier to create or modify the configuration file (.cfg). This software is used in our research work, because of its transparency, reliability, and recent adaptation to the HLA standard.

JaamSim was not originally designed for communications to external systems and is not suitable for DS; it is considered to be a black-box simulator. This contribution uses an extended version of JaamSim [42] which opens it to distributed simulation via the HLA standard. In [42], the author proposes to edit the source code of the tool in order to make it communicate with different instances of JaamSim via the HLA standard. We will use this proposal to make JaamSim, as Papyrus, collaborate with external systems.

4. Distributed-Simulation Orchestration

According to the HLA standard [37], a federation cannot be managed from a component since each federate is autonomous and cannot be controlled. Moreover, there is no graphical tool allowing the implementation of a distributed-simulation scenario that defines an order of execution for different simulations constituting an HLA federation. This point is an important aspect since it would allow the distributed-simulation behavior to be defined using a simple graphical-modeling language.

Defining a simulation scenario means centralizing the information that decides the order of execution of the different distributed simulations. The concept of "order of execution" introduces the notion of pending federates. The main objective is to connect

the federates to a specific federation, execute them, and wait for their respective orders in order to launch their simulations appropriately. This diagram must therefore have priority over all the federates of a federation, and this is currently impossible in the HLA standard as described by the HLA specifications. Each federate is autonomous and can only be prioritized by the order of the messages exchanged. This is why a distributed HLA simulation orchestration has never been proposed before.

In order to set up a distributed-simulation scripting system, it is imperative to implement a decision-maker/master mechanism. The decision maker having, as a resource, a graphical diagram describing the execution order of the federates must be able to drive and control the federates described by the model. At this point, the concept of control is interpreted by a set of pre-established rules that will allow the communication between the master and the different federate entities (the slaves) through HLA interactions.

According to the HLA standard, a master federate piloting other federates is not an existing concept in the standard because a federate is an autonomous simulation, which only needs a Run Time Infrastructure (RTI) to work. In the following part, we will describe the development process of an HLA standard upper layer to implement the decision-maker/master mechanism.

4.1. Slave Federate

This proposal is an upper layer to the HLA standard illustrated in Figure 1, which will be a mandatory layer for the good functioning of the mechanisms already implemented by the various frameworks used. The federate executor is a classic federate, on which a management layer decision maker/master is encapsulated. This layer is responsible for controlling the simulation of the federate.

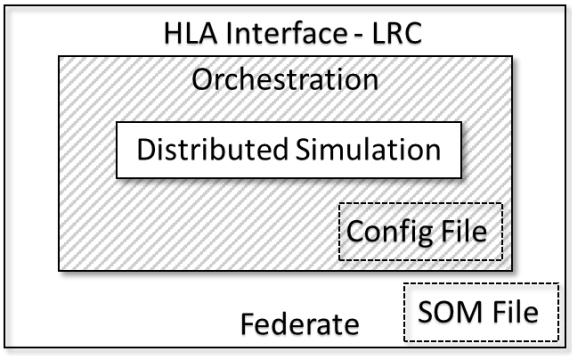

Figure 1. Layered view of a slave federate.

Figure 1 shows the layered architecture of a slave federate. The communication between federates is based on the publish/subscribe mechanisms of the standard. The HLA layer is the bottom layer that will serve as a pillar on the upper stages. It contains all the necessary components that allow a federate to be part of a federation, namely the LRC (local RTI component), allowing the interface with the remote RTI, as well as its SOM (simulation object model) file, describing shared data with other federates.

The SOM file contains all the "Objects" and "Interactions" related to the distributed simulation itself. It also contains a set of objects and interactions necessary for the proper functioning of the top layer "Orchestration" allowing the management of the master system, as well as time management of the connected federates. In this application, we will not be in need of the time management between federates; this is why the time regulating/constrained of the federates will be disabled.

The intermediate layer "Orchestration" has two functions:

- Drives the distributed top-layer simulation;
- Manages the communications between the federate decision maker and the federate executor.

The distributed simulation is controlled by messages sent by the decision maker to the executor. Depending on the content of the received message, the federate will perform one of the following actions:

- Start the simulation: "START" message;
- Pause the simulation: "PAUSE" message;
- Resume the simulation: "RESUME" message;
- Stop the simulation and destroy the federate: "KILL" message.

When a federate is launched, after the initialization steps imposed by the HLA standard, the orchestration layer automatically pauses the simulation. It will be waiting for the reception of the "START" signal from the master. Upon receipt of this message, the simulation is released. Conversely, when a PAUSE signal is received, the orchestration layer freezes the progress of the simulation, its internal processes, and the communication to the federates. When a "KILL" signal is received, the master federate destroys all connected federates, as well as the whole federation.

4.2. Master Federate

As for the slave federate, the decision maker (or master federate) is a normal federate, on which we encapsulated the orchestration layer. This layer (visible in Figure 2) implements the same communication mechanisms as described above. The major difference between this federate and the others (slaves) is in the last software layer.

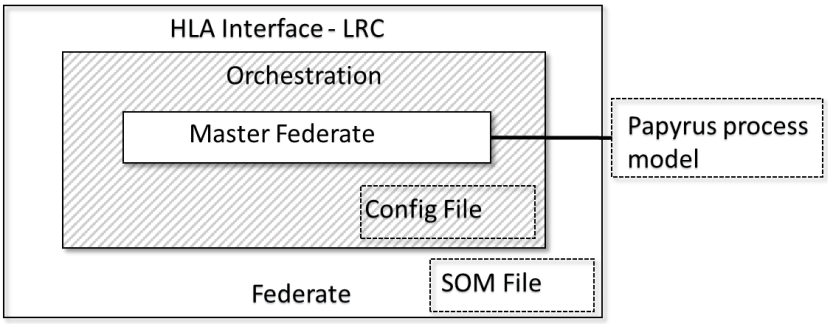

Figure 2. Layered view of a master federate.

Figure 2 shows the contents of a master federate. It has the same orchestration layer as its peers. The difference is located in the center of the figure on the last software layer. It is a federate dedicated to the management of the distributed-simulation execution. It is recalled that the federate control consists of the ability to put on hold, launch, and wait to stop the latter in order to direct them like black boxes. The master federate is connected to Papyrus process model and is able to send orders to federates of the federation.

The role of the Papyrus process model is to describe the execution order of the federates. This execution order will be read and interpreted by the Papyrus Moka engine, in order to launch the federations in a specific order.

Thus, if the configuration tells us about four federates with fed1, fed2, fed3 and fed4 identifiers, the process model diagram must describe the behavior of these four federates using the identifiers entered in the configuration.

When launching a distributed simulation, the master federate must be launched first. It will create the federation and initialize the orchestration layer.

After the HLA initialization steps are completed, the federate reads the configuration file information and verifies that the BPMN file is correctly written, and that the names of the federates handled match the federates of the configuration file. If an error appears, the simulation is not run and the origin of the error is indicated in the simulation report. Following these tests, two functions are possible. The "hot-plug" mode (or "hot-connected") starts the simulation without waiting for all federates to connect. Conversely, in the

"cold-plug" mode, the decision maker waits until all federates are connected to start the distributed simulation.

In the rest of this paper, we consider that the master federate will be a Papyrus Moka extension able to control HLA federate designated by a custom UML profile added over Papyrus modeler.

5. Papyrus as a Middleware Component

5.1. Papyrus Extension Mechanism

Papyrus can be extended by UML profile and Moka extension. We will see both of these mechanisms in the next sections.

5.1.1. UML Profile

According to the UML standard [43], a profile is a means of providing a generic extension mechanism to customize basic UML models. It allows one to customize the graphical and semantic language in order to bring it closer to a particular context. Additions made to the base language cannot contradict the semantics of the original standard language.

Profiles are defined through "stereotypes" that are applied to elements of the base language (such as classes, attributes, etc.). They are therefore a set of extensions that customize the standard for a specific context. A stereotype can be extended to several parameters in order to add semantic value over the UML Profile. From the Papyrus point of view, a UML Profile declared and applied on a task allows one to add data over a simple UML task. However, these data cannot be interpreted by the execution engine, and we need to declare a Moka extension.

5.1.2. Moka Extension

Moka is an event-based execution engine. At each event, a set of components will be queried in order to authorize or not the execution of a task. These components are called visitors and advices. In Moka, a visitor is instantiated at each element of the basic UML diagram: starting points, stopping points, transitions, steps, etc. Each of these visitors contains a list of advices associated with it. Each advice allows one to add an additional behavior to the execution engine. During simulation execution, each of the visitors will be consulted by Moka at the moment its own task is activated. When a visitor is consulted, all the advices associated with the visitor are executed in order to perform the behavior they describe.

From the point of view of the Moka runtime engine, we can therefore represent the interactions between the basic model and its various Visitors, as shown in Figure 3.

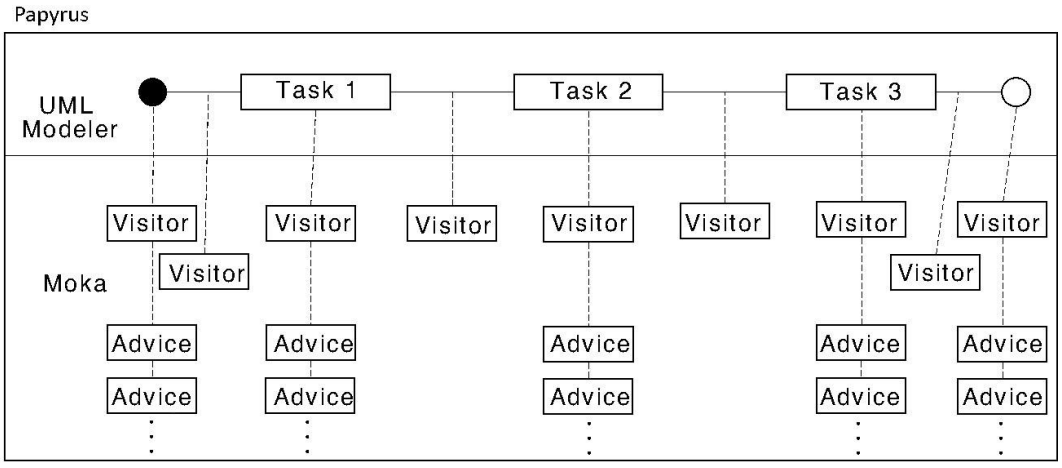

Figure 3. Visitors and advices connected to Papyrus (UML) model.

Each element of the UML diagram is linked to its associated visitor. Each visitor of the simulation can have none, one, or several associated advice(s). It is therefore possible to extend the Papyrus execution engine to add advices to the visitors, and thus add custom conditions/behaviors when executing a Papyrus model.

5.2. Papyrus Extension for Coupling to External Components

Distributed components are designed to gravitate around the Papyrus tool: the simulation orchestrator component. Its role is to coordinate the different tools to obtain a single and homogeneous platform. Its modeling tool (extensible through UML profiles) contains the global business simulation scenario referring to related functions. Its Moka execution engine (also extensible) interprets the scenario and interacts with connected tools in Papyrus.

Each of the technologies exploited by this contribution (risk management and HLA) is handled by a Papyrus software layer (UML profile + Moka extension), which is stacked to build the overall proposal illustrated in Figure 4).

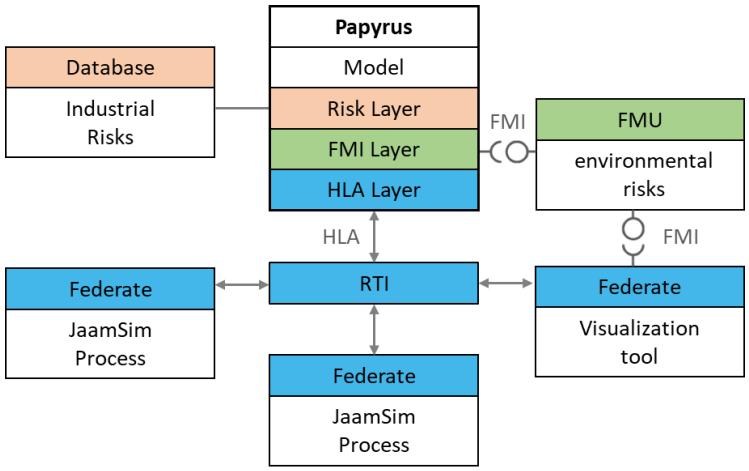

Figure 4. Technical architecture.

5.2.1. Risk-Management Layer

The risk-management layer consists of a UML profile added to the papyrus UML modeling tool, and an extension of the Moka runtime allowing a customized behavior. We emphasize that the function of a UML profile is to semantically customize a basic language. Here, we add the notion of a unique identifier ("ID" in Figure 5) to each task (UML Action in Figure 5) of the general model.

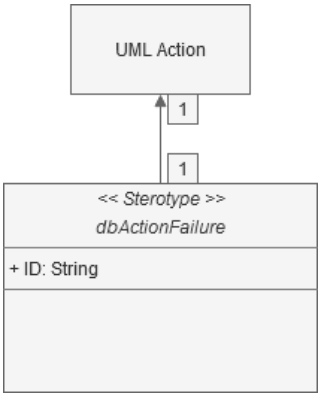

Figure 5. Risk-management Papyrus profile.

Industrial risk information is stored in a remote database; thus, the link between database and Papyrus is done through the definition of a new Moka extension. The Eclipse EMF Generator provides the basis for the implementation of a Papyrus extension: a set of classes and methods for implementing an additional advice to the visitor of each task extended by the UML "dbActionFailure" profile. When the model is executed, the Moka engine identifies a running task and performs a time-stamped query in the risk database. Upon receipt of this query, the risk-management tool identifies the impact of the current risks as a function of the simulated time and returns to Papyrus the value of the temporal impact on the simulation.

5.2.2. FMI–Environmental Risks Layer

The implementation of communications between Papyrus and the distributed FMU components is established through an additional Papyrus software layer. To ensure communication between the Papyrus tool and the simulation environment provided by the FMI standard, we are adding three components to the system:

First, a UML profile (superimposed on the previous one) allowing a user to define a link between a task in the Papyrus diagram and an FMU component. The "Path" parameter in Figure 6 contains the path to the FMU file to be executed;

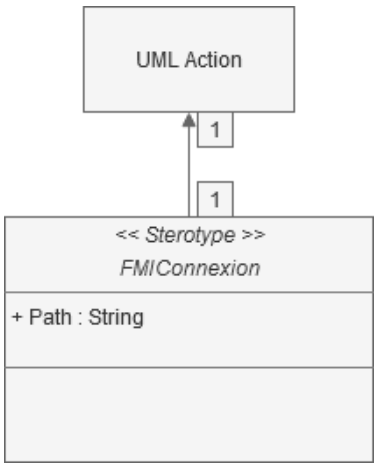

Figure 6. Functional mockup interface (FMI) Papyrus profile.

Second, a Papyrus configuration file containing additional information on the nature of the FMU supported by papyrus. A json configuration file is accessible at the root of Papyrus and describes the data related to the external components handled by the tool. In our case, it will contain the nature and value of the input parameters of the FMU files loaded by the tool, as well as the output values of the components;

Third, a Moka extension (described in Figure 6) retrieving on the one hand the FMU path available in the UML profile and on the other hand the different information about the FMU component in the file described above.

When performing a task extended by the described profile, the Moka engine makes a time advance request (time value defined by the configuration file) to the FMU indicated by the "Path" field in Figure 6. Upon receipt of this request, the FMU advances its own time, and returns values to the extension, which are recorded in the FMU configuration file. From a Papyrus model, you can order the execution of FMI compatible components.

5.2.3. HLA–Technical Simulation Layer

The Papyrus layer allowing the piloting of an HLA federation takes up the proposal of an upper layer applied to the federations in order to implement a decision-maker/executing mechanism between each of the elements of the federation. To do this, the Papyrus process-modeling tool is used to orchestrate related federates.

As for the previous extensions, this Papyrus layer requires the definition of a custom UML profile applied to Papyrus tasks representing an HLA component. Here, we declare a new stereotype that will be applied to UML actions and that will contain a Federate-Name field (see Figure 7) in order to identify the federate that will be launched by Moka at runtime.

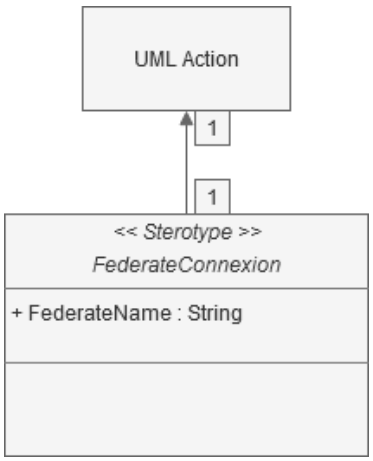

Figure 7. High-level architecture (HLA) Papyrus profile.

At simulation start, Moka instantiates a MasterFederate class which allows it to create a preconfigured federation by the developer (definition of the name of the federation, content of FOM files, expected number of federates, etc.). Moka waits for the connection of all the federates needed for the simulation to go further. Once the right number of federates have been connected to the simulation, the Papyrus model is run until it reaches a task extended by the UML profile meaning the launch of a federate. Then, the UML task visitor queries a function of each advice associated with it. In our case, our custom advice will refer to the FederateManager which will launch the federate associated with the FederateName field described by the UML profile through the MasterFederate class. In other words, the Moka engine gives an execution order to an executing federate. In a second step, Moka queries a getDuration() function of each advice. According to Moka, this function must return the simulated cumulated time of all the advices of the current task. When this function is executed for a federate task, the FederateManager puts the Moka engine on standby and loops on the change of state of the LastDuration variable of the MasterFederate class. On its side, the MasterFederate class is waiting (subscribed) for the "FederateEnd" interaction described in Figure 8.

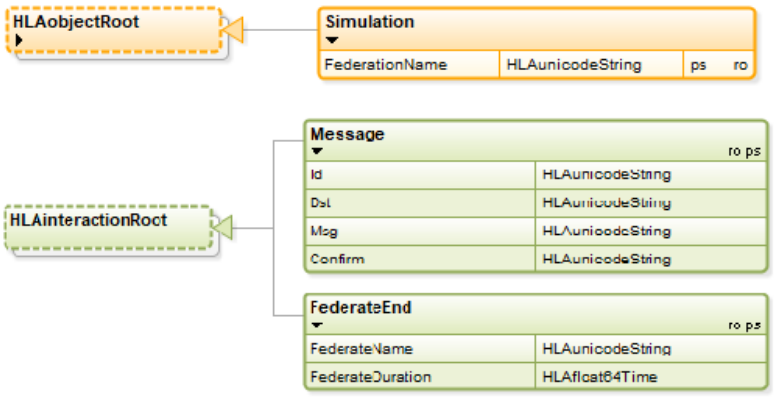

Figure 8. HLA objects and interactions for Papyrus orchestration.

The FederateEnd interaction is sent by a federate once it has completed its simulation. It sends a message to Papyrus to let it know that its operation is finished and to send its own simulation duration. This time will be retrieved by the MasterFederate and stored in a local variable. FederateManager will then be able to retrieve it and transmit it to the Moka scheduler to represent the duration of the distributed task.

6. Risk-Management Tool

The consequence of a risk is to cause the degradation of the duration of one or several tasks. It is necessary that this disruption acts on the durations during the simulation runtime. In order to have the most flexible tool possible, the user defines each of the risks present in the simulation in an external database. Therefore, there is a risk-modeling phase to be undertaken in parallel with the process modeling. The risks can be identified in different ways depending on the context. A risk can be used in two different ways and can have two different objectives:

It can be an item referenced by a unique identifier that can have an impact on the duration of one or more tasks of a Papyrus process. This impact can be a constant, the result of an equation, or based on a semirandom factor;

It can also be an element aggravating the impact of another risk in the simulation.

In order to meet these two requirements, risk management and generation is handled by one or more Excel files. Risk modeling can be static, defined in hard in the Excel table, or dynamic, defined according to Excel formulas which can take in parameters: pseudo random values or the dates of the current simulation. As shown in Figure 9, the communication with the Excel file is time-stamped.

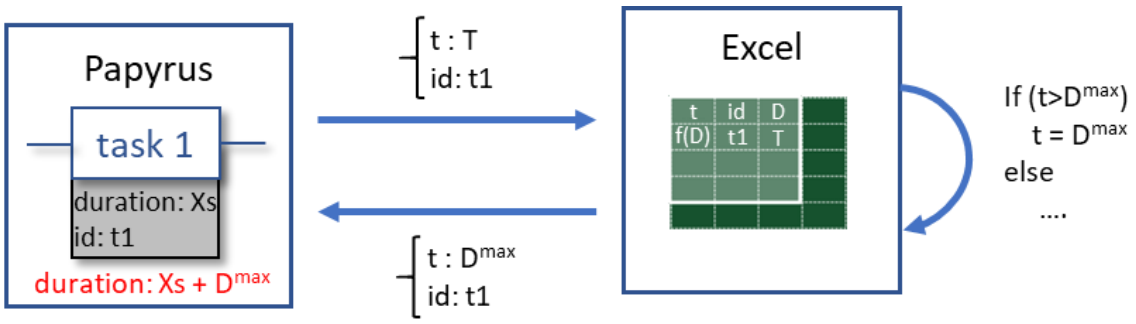

Figure 9. Dynamic risk management for Excel.

When a task extended by the risk-management profile is executed, the Moka engine queries the Excel file by providing the id of the current task (id = t1), and the current simulation date (t = T). This last information is entered in the table (see rule relative to column 3 of the interface table), and then all the Excel formulas are updated. The first column is then updated and retrieved by the Moka engine. In the case of Figure 9, the duration of task 1 is therefore Xs (its initial duration) + Dmax (its additional duration relative to the risks incurred).

From the risk simulation point of view, according to Papyrus, the duration of a UML task is calculated by the cumulative effect of all the advices associated with the task. As can be seen in Figure 10, for a UML element is associated a visitor which contains one or more advices. The visitor calls the getDuration() methods of all his advices and sums their results.

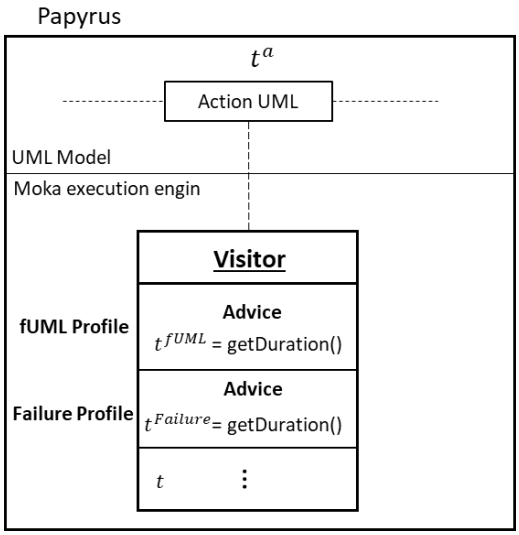

Figure 10. Task life duration from UML Profile.

The getDuration() method returns a double representing a duration in nanoseconds. Therefore, the sum of the durations of the advices represents the total time of the task (t) as the following equation:

$$t^a = t^{fUML} + t^{Failure} + \sum_{i=0}^{n} t^i \quad (1)$$

It is via instances of a custom advice and the override of getDuration method that we can query the Excel file, get a risk impact for a specific task, and degrade the task time. Risk can also impact distributed simulations controlled by Papyrus business process while injecting risk parameters to connected devices.

7. Simulation Case Study and Results

These researches and proposition were supported by industrial context. At the crossroads of several disciplines, ALSOLENTECH expresses the need for a tool capable of grouping simulation standards, simplifying process modeling by outsourcing risks, and connecting external tools of its environment. In this section, we group together all of the contributions mentioned above to form a global application for the unification of tools and technologies. Papyrus, the orchestration tool, interconnects blocks of functionality, each representing a part of the production process.

ALSOLENTECH is a French company for the design, construction, and operation of the thermodynamic and photovoltaic solar power plants. Based on Fresnel mirrors, the power plant technology proposed makes it possible to supply isolated areas simultaneously with electricity, drinking water, refrigeration, and industrial heat. The technical and economic context of concentrated heat-energy production is a relatively young field of research, which is confronted with notions of uncertainty and competitiveness. This creates a need for risk management. Moreover, our industrial context is a grouping of various fields and technologies, whose intersection allows one to ensure the development and the operation of the solar power plants. All the actors of the industrial chain include tools and knowledge that will need to communicate, without a standardized exchange interface. This need involves issues of interoperability between heterogeneous components, which are addressed by distributed-simulation concepts: each component represents a domain, which is a subpart of a global system to experiment and analyze the performance of the whole system.

In this context, ALSOLENTECH needs to model the process of setting up and deploying a solar-panels field. Figure 11 shows the process of setting up a solar field. It consists of a set of tasks referring to external tools.

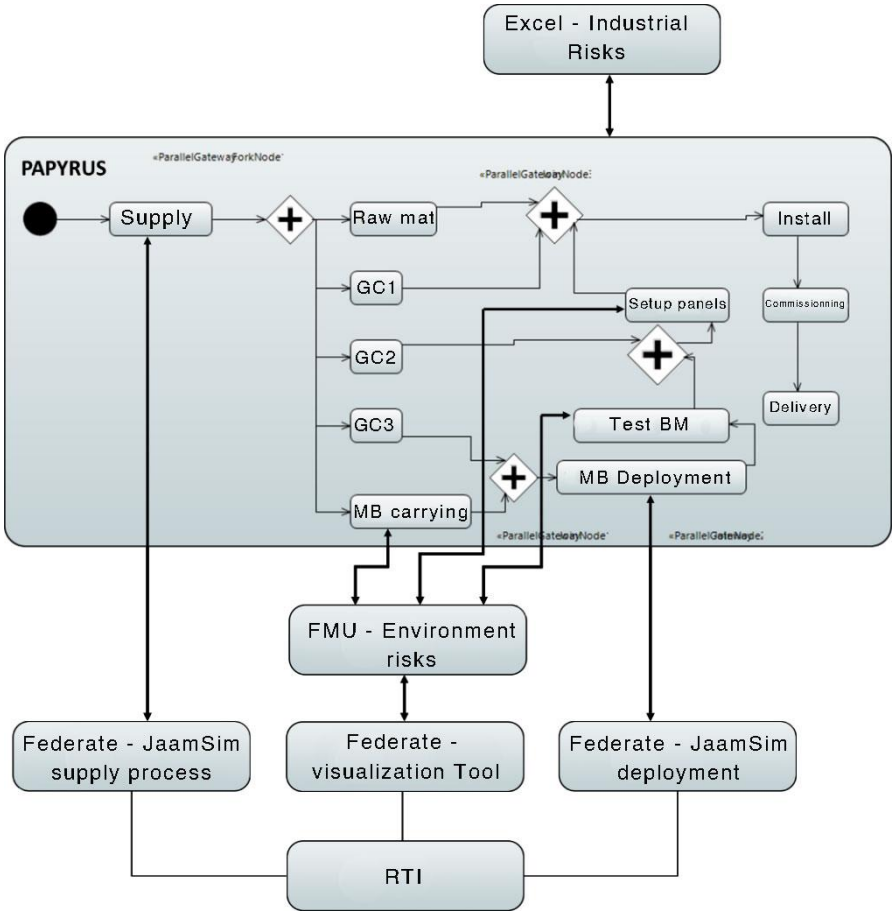

Figure 11. Papyrus orchestration process.

The first step of this model corresponds to the simulation of the materials supply necessary for the different constructions that make up the entire power plant. Then, a set of tasks are launched in parallel:

- the delivery of materials necessary for the operation of the solar power plant: raw materials;
- the construction of the three structures for the power plant: civil engineering (GC) 1, 2, and 3;
- the delivery of the mobile factory to the production site: mobile factory carrying.

Once the mobile factory has been delivered to the production site, and civil engineering 3 (GC3) has been completed, the deployment and operation of the plant can be simulated: "MB Deployment" which simulates the deployment of the plant. Once the solar fields have been deployed, a test battery is carried out to determine the effective power of the plant: "Test mobile factory production". Then, the solar panels are turned on for testing purposes before delivery.

The first task ("Appro" task in Figure 11) consists in the supply delivery by airplane of two categories of products. This task is subject to risks and will be deported to a JaamSim simulation. We therefore have two profiles applied to this UML action:

- FailureProfile to connect the task to the risk-management tool;
- HLAProfile to connect the task to a HLA federate.

The task ID is transmitted to the Excel file via the ID field in the FailureProfile -> SiteExploitation. The HLAprofile is also applied to it, which points to the name of the federate to be executed -> Appro. As this task is subject to risks, Papyrus queries the environmental risk-management tool, in order to influence the input parameters of the JaamSim simulation. Parameters are calculated and then transmitted to the JaamSim federate by modifying the HLA object visible in Figure 12.

Scenario_Appro			ps
Federate_name	HLAunicodeString	ps	ro
SimTime	HLAfloat64Time	ps	ro
N_HPriority	HLAinteger64Time	ps	ro
N_LPriority	HLAinteger64Time	ps	ro
F_HPriority	HLAunicodeString	ps	ro
F_LPriority	HLAunicodeString	ps	ro
D_Loading	HLAunicodeString	ps	ro
D_TR	HLAunicodeString	ps	ro
D_VOL	HLAunicodeString	ps	ro
D_TR2	HLAunicodeString	ps	ro
D_Depal	HLAunicodeString	ps	ro

Figure 12. HLA object: supply delivery parameters.

Here, all parameters are user-configured variables. These are the nominal parameters of the simulation which do not take into account the risk. All these values are modified by the risk-management tool through Moka engine. In a second step, Papyrus sends an execution order interaction to the federate that runs the simulation visible in Figure 13. The supply delivery simulation is done with an industrial modeling and simulation tool: JaamSim. JaamSim process generating two types of entities (H_Priority and L_Priority). Their number and production rate are input variables. Moreover, the different execution times related to transport, trans-shipment, loading, and unloading are also input parameters (Figure 12) that can be modified by the risk-management tools. During this simulation, the various performance indicators are recorded and updated at the RTI for each crossing of several blocks in Figure 13. These parameters (Figure 12) are updated using the RTI ambassador's updateAttributeValues() function.

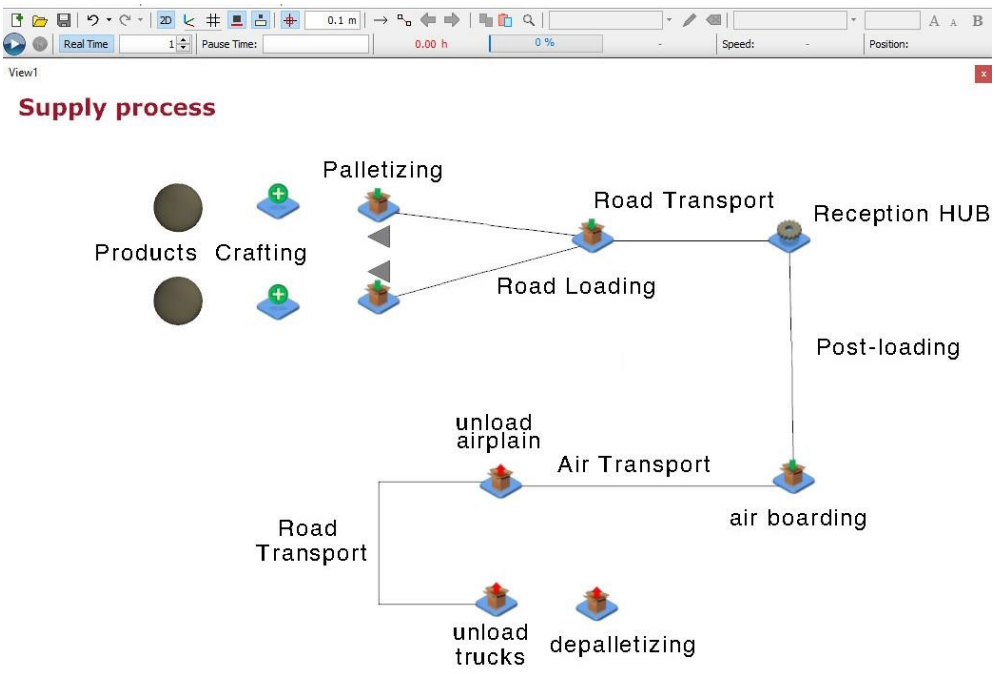

Figure 13. JaamSim supply delivery process.

The next steps of the Papyrus process (Figure 11) are executed in parallel:

- The delivery of raw materials for the solar field, and construction steps (raw materials, GC1, GC2, GC3) are tasks simulated by Papyrus and disrupted by the industrial risk-management database;
- The delivery of a mobile factory is simulated by Papyrus and disrupted by the environmental risk-management tool. This external component is a system composed of different states performing meteorological queries in order to disrupt the progress of a task. In our case, this is the first time that a Papyrus task uses the FMU. It is therefore in its initial state: queries on the humidity rate (precipitation) of a geographical area for a time described in the configuration file of the simulation.

The next task of the orchestration model concerns the deployment of the power plant. Here, Papyrus once again relocates the complexity of the simulation through a JaamSim model impacted by industrial risks. The "Mobile factory deployment" of Figure 11 is thus extended by the HLAProfile (which indicates the name of the JaamSim federate to be executed) and the FailureProfile (which introduces hazards in the simulator inputs).

The deployment of the power plant is a task simulated by JaamSim. It involves building and deploying the solar reflectors directly on the production site. Its operation (illustrated in Figure 14) is a succession of steps transforming steel sheets into mirrors placed on motorized gantries.

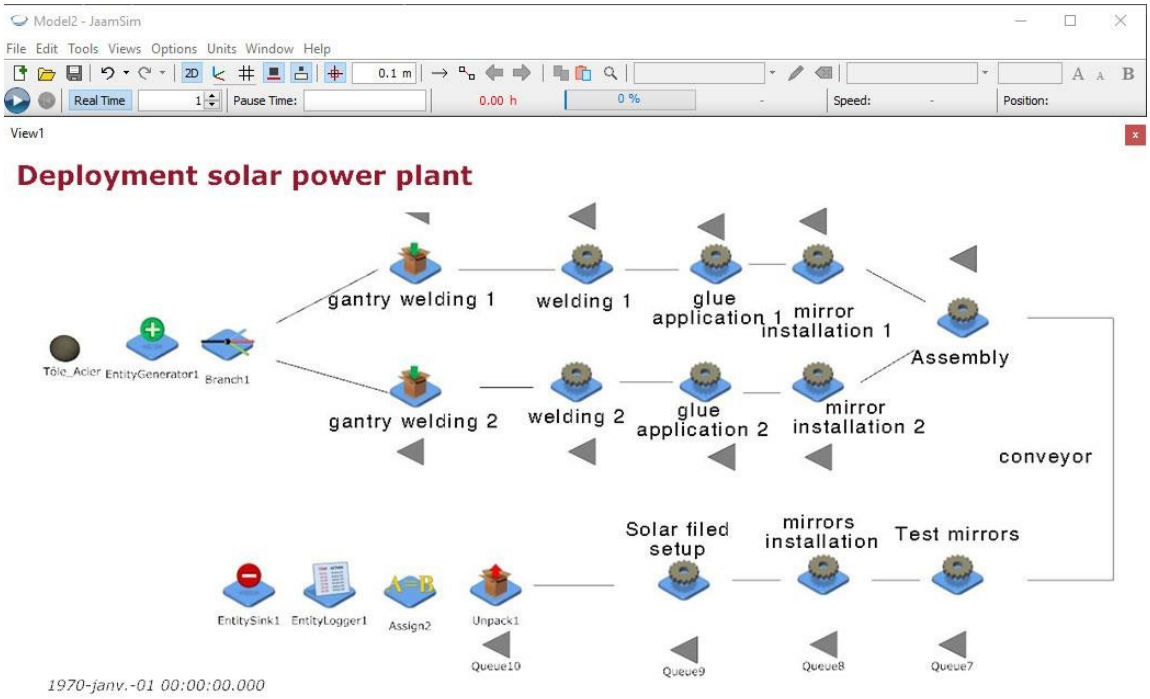

Figure 14. JaamSim deployment of the power plant.

As for the previous JaamSim simulation, all production lead times are input parameters of the simulation. Defined by default in the Papyrus configuration file, they are accessible for modification by the industrial risk-management tool. When the task is executed, Papyrus transmits the parameters through an HLA object and sends an execution order interaction.

Both JaamSim simulations were configured to trace two KPI's at simulation runtime:

- evolution of leadtime;
- evolution of work in progress (WIP).

As a demonstration of efficiency of our proposition, we ran the same simulation two times: first with the risk-management tool enabled and then without the risk-management tool. We can observe these results in Figure 15.

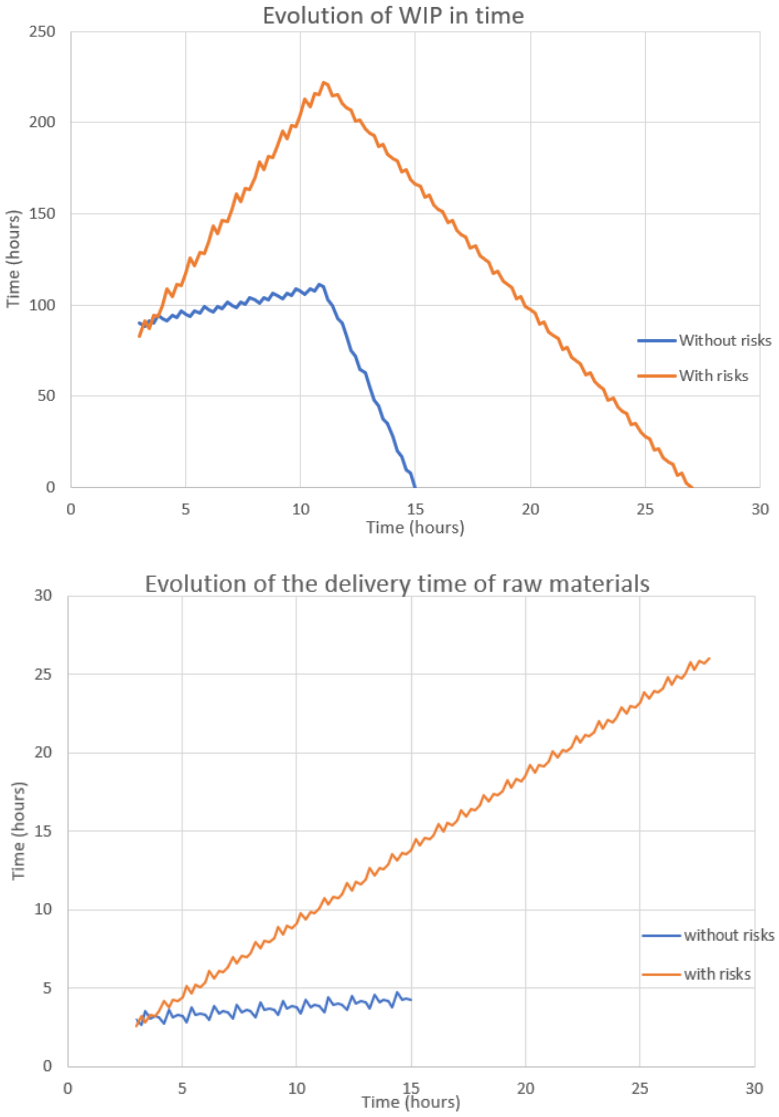

Figure 15. Evolution of JaamSim KPI's with and without risk management.

Papyrus is a modeling and simulation tool that can work autonomously. By default, the orchestration of Figure 11 can be run without the risk-management extension. Each of the tasks in the model (with the exception of the tasks referring to JaamSim simulations) has a nominal duration defined by the user. The set of these durations is determined by the engineers who estimate the average time of each steps. The uppers graphs of Figure 15 are results of each JaamSim simulations without risk management enabled. We can see that the delivery time of raw materials without risk management (bottom blue) is linear and is almost horizontal. Once risk management is activated (bottom red), we can see that WIP is degraded. The same observation can be done on the second JaamSim simulation KIP's (top graph of Figure 15). All Papyrus tasks have thus received a temporal degradation according to various factors:

- semirandom laws described according to the company's knowledge (reproducible with a seed generation system);
- simulation dates;
- environmental factors related to the project (location, planned date of the construction site, rainfall, sunshine, etc.);

8. Conclusions and Perspectives

The Papyrus tool is an efficient way to model and simulate an industrial process. We have shown in this article that the extension of Papyrus can be untied to other methods and technologies. The legacy effect that Papyrus extensions have between them allows us to cumulate contributions one after the other. Thus, we can add modeling and simulation components without impacting the basic model.

In this project, we assembled three Papyrus extensions into a single tool by first connecting a database on industrial risks to Papyrus, so that this information can impact the simulation results. We also open the tool to an external real-time communication standard (FMI 2.0). We demonstrated the use of a component capable of extracting data from the environment (through the FMU making queries to a weather web service). This example shows the possibilities of interconnection between a simulated system and its real environment. Finally, we demonstrated the tool's ability to communicate or direct other simulators through a distributed-simulation standard (HLA).

All these results are linked with 4.0 industry. Getting risk information from a global risk database is related to data-driven processes and big data. Opening the Papyrus tool to several distributed-simulation standards is a link with the concept of the internet of things. Orchestrating several distributed simulations can be associated to smart manufacturing and digital twins.

There are still prospects in the use of HLA and FMI distributed-simulation standards within Papyrus. The implementation of more complex time-management mechanisms between HLA and Papyrus would increase the exchange and communication capacities between the two tools. The next release of FMI 3.0 opens up interesting perspectives for our contribution. Indeed, the cosimulation standards announce clocks for synchronization of variables changes across FMUs, which could allow for a FMU to trigger its master component and send information without request from it. Furthermore, binary data type can now be exchange through FMI standard. This improvement leads to a reconciliation of data types between HLA and FMI standards: users will be able to build complex messages exchanged between components. Currently, five distributed tools have been developed for the application. An interesting objective would be to extend the cosimulation framework built to integrate other components orchestrated by the Papyrus model. We have demonstrated that it is possible to connect a Papyrus model to web service calls. Transposing our contributions to other domains (such as the service industry) is a source of interesting perspectives.

Author Contributions: Conceptualization, S.G., J.P., G.Z., N.P., and Y.D.; methodology, S.G., J.P., G.Z., N.P., and Y.D.; software, S.G., J.P., and G.Z.; validation, S.G., J.P., G.Z., N.P., and Y.D.; formal analysis, S.G., J.P., G.Z., and Y.D.; investigation, S.G., J.P., G.Z., and Y.D.; resources, S.G., J.P., G.Z., and Y.D.; data curation, S.G. and J.P.; writing—original draft preparation, S.G., J.P., G.Z., and Y.D.; writing—review and editing, S.G., J.P., G.Z., and Y.D.; visualization, S.G. and J.P.; supervision, G.Z. and Y.D.; project administration, G.Z.; funding acquisition, G.Z. and N.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by Region Nouvelle Aquitaine, grant number 2016-1R60102-00007447.

Acknowledgments: Authors acknowledge ALSOLENTECH and CEA TECH for their technical support and their help in using Papyrus and Moka.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Industry 4.0: Definition and Implementation to the Connected Plant. Visiativ Solutions. Available online: https://www.visiativ-solutions.fr/industrie-4-0/ (accessed on 25 September 2020).
2. Kagermann, H.; Lukas, W.D.; Wahlster, W. Industrie 4.0: Mit dem Internet der Dinge auf dem Weg zur 4. industriellen Revolution. *VDI Nachr.* **2011**, *13*, 2–3.

3. Adamik, A.; Nowicki, M. Preparedness of Companies for Digital Transformation and Creating A Competitive Advantage in the Age of Industry 4.0. In Proceedings of the International Conference on Business Excellence, Bucharest, Romania, 18–19 March 2018; Sciendo: Lodz, Poland, 2018; Volume 12, pp. 10–24.
4. Ericson, Å.; Lugnet, J.; Solvang, W.D.; Kaartinen, H.; Wenngren, J. Challenges of Industry 4.0 in SME businesses. In Proceedings of the 2020 3rd International Symposium on Small-Scale Intelligent Manufacturing Systems (SIMS), Kyoto, Japan, 10–12 June 2020; pp. 1–6. [CrossRef]
5. Ghadge, A.; Er Kara, M.; Moradlou, H.; Goswami, M. The impact of Industry 4.0 implementation on supply chains. *J. Manuf. Technol. Manag.* **2020**, *31*, 669–686. [CrossRef]
6. Madni, A.M.; Madni, C.C.; Lucero, S.D. Leveraging digital twin technology in model-based systems engineering. *Systems* **2019**, *7*, 7. [CrossRef]
7. Mangles, C. Gartner Hype Cycle 2018—Most Emerging Technologies Are 5–10 Years Away. Available online: https://www.smartinsights.com/managing-digital-marketing/managing-marketing-technology/gartner-hype-cycle-2018-most-emerging-technologies-are-5-10-years-away/2018 (accessed on 3 September 2016).
8. Taylor, S.J.E. Distributed simulation: State-of-the-art and potential for operational research. *Eur. J. Oper. Res.* **2019**, *273*, 1–19. [CrossRef]
9. Guermazi, S.; Tatibouet, J.; Cuccuru, A.; Dhouib, S.; Gérard, S. Executable Modeling with fUML and Alf in Papyrus: Tooling and Experiments. *EXE MoDELS* **2015**, *11*, 3–8.
10. Lee, D. Development of Mediator-Based Direct Wokrflow Simulation System and HLA/RTI-Based Collaborative BPMS Middleware. Ph.D. Thesis, Korea Advanced Institute of Science and Technology, Daejeon, Korea, 2010.
11. Falcone, A.; Garro, A.; D'Ambrogio, A.; Giglio, A. Engineering Systems by combining BPMN and HLA-based distributed simulation. In Proceedings of the 2017 IEEE International Systems Engineering Symposium, Vienna, Austria, 11–13 October 2017; pp. 1–6.
12. Vernadat, F.B. UEML: Towards a Unified Enterprise Modelling Language. In Proceedings of the Actes 3ème Conférence Francophone de Modélisation et Simulation—MOSIM'01, Troyes, France, 25–27 April 2001.
13. Possik, J.; d'Ambrogio, A.; Zacharewicz, G.; Amrani, A.; Vallespir, B. A BPMN/HLA-Based Methodology for Collaborative Distributed DES. In Proceedings of the 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Capri, Italy, 12–14 June 2019; IEEE: New York, NY, USA, 2019; pp. 118–123.
14. Taylor, S.J.E.; Turner, J.S.; Strassburger, S. Guidelines for Commercial Off-the-Shelf Simulation Package Interoperability. In Proceedings of the 2008 Winter Simulation Conference, Miami, FL, USA, 7–10 December 2008; pp. 193–204.
15. Bocciarelli, P.; Gianni, D.; Pieroni, A.; D'Ambrogio, A. A Model-Driven Method for Building Distributed Simulation Systems from Business Process Models. In Proceedings of the 2012 Winter Simulation Conference (WSC), Berlin, Germany, 9–12 December 2012; pp. 1–12.
16. Bazoun, H.; Zacharewicz, G.; Ducq, Y.; Boyé, H. SLMToolBox: An Implementation of MDSEA for Servitisation and Enterprise Interoperability. In *Enterprise Interoperability VI*; Mertins, K., Bénaben, F., Poler, R., Bourrières, J.-P., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 101–111.
17. Zacharewicz, G.; Frydman, C.; Giambiasi, N. G-DEVS/HLA environment for distributed simulations of workflows. *Simulation* **2008**, *84*, 197–213. [CrossRef]
18. Yilmaz, F.; Durak, U.; Taylan, K.; Oguztuzun, H. Adapting Functional Mockup Units for HLA-compliant Distributed Simulation. In Proceedings of the 10th International Modelica Conference, Lund, Sweden, 11–12 March 2014; pp. 247–257.
19. Falcone, A.; Garro, A. Distributed co-simulation of complex engineered systems by combining the high level architecture and functional mock-up interface. *Simul. Model. Pract. Theory* **2019**, *97*, 101967. [CrossRef]
20. Awais, M.U.; Palensky, P.; Mueller, W.; Widl, E.; Elsheikh, A. Distributed Hybrid Simulation Using the HLA and the Functional Mock-Up Interface. In Proceedings of the IECON 2013—39th Annual Conference of the IEEE Industrial Electronics Society, Vienna, Austria, 10–13 November 2013; pp. 7564–7569.
21. Obeng, E. *All Change!: The Project Leader's Secret Handbook*; FT/Prentice Hall: Upper Saddle River, NJ, USA, 1994.
22. Shachter, R.D. Evaluating influence diagrams. *Oper. Res.* **1986**, *34*, 871–882. [CrossRef]
23. Sonchan, P.; Ramingwong, S. ARMI 2.0: An Online Risk Management Simulation. In Proceedings of the 2015 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Hua Hai, Thailand, 24–27 June 2015; pp. 1–5.
24. Jürjens, J. *Secure Systems Development with UML*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2005.
25. Altuhhova, O.; Matulevicius, R.; Naved, A. An Extension of BPMN for Security risk management. *Int. J. Inf. Syst. Model. Des.* **2013**, *4*, 93–113. [CrossRef]
26. Laue, R.; Müller, C. The Business Process Simulation Standard (BPSIM): Chances and Limits. In Proceedings of the ECMS, Regensburg, Germany, 31 May–3 June 2016; pp. 413–418.
27. Aksyonov, K.; Bykov, E.; Aksyonova, O. Development and Application of Software Engineering Solution BPsim. SD. In Proceedings of the 2013 European Modelling Symposium, Lancaster, UK, 23–25 October 2013; IEEE: New York, NY, USA, 2013; pp. 321–325.
28. Fan, M.; Thongsri, T.; Axe, L.; Tyson, T.A. Using a probabilistic approach in an ecological risk assessment simulation tool: Test case for depleted uranium (DU). *Chemosphere* **2005**, *60*, 111–125. [CrossRef] [PubMed]

29. Lu, H.; Axe, L.; Tyson, T.A. Development and application of computer simulation tools for ecological risk assessment. *Environmental Model. Assess.* **2003**, *8*, 311–322. [CrossRef]
30. Bixio, D.; Parmentier, G.; Rousseau, D.; Verdonck, F.; Meirlaen, J.; Vanrolleghem, P.; Thoeye, C. A quantitative risk analysis tool for design/simulation of wastewater treatment plants. *Water Sci. Technol.* **2002**, *46*, 301–307. [CrossRef] [PubMed]
31. Benaben, F.; Montarnal, A.; Truptil, S.; Lauras, M.; Fertier, A.; Salatge, N.; Rebiere, S. A Conceptual Framework and A Suite of Tools to Support Crisis Management. In Proceedings of the HICSS 2017—50th Hawaii International Conference on System Sciences, Hilton Waikoloa Village, HI, USA, 4–7 January 2017; pp. 237–246.
32. Marmier, F.; Deniaud, I.F.; Gourc, D. Strategic decision-making in NPD projects according to risk: Application to satellites design projects. *Comput. Ind.* **2014**, *65*, 1107–1114. [CrossRef]
33. Fertier, A. Interprétation Automatique de Données Hétérogènes Pour la Modélisation de Situations Collaboratives: Application à la Gestion de Crise. Ph.D. Thesis, Ecole des Mines d'Albi-Carmaux, France, 2018.
34. Cleary, P.W.; Thomas, D.; Hetherton, L.; Bolger, M.; Hilton, J.E.; Watkins, D. Workspace: A workflow platform for supporting development and deployment of modelling and simulation. *Math. Comput. Simul.* **2020**, *175*, 25–61. [CrossRef]
35. Van Der Aalst, W.; Van Hee, K.M.; van Hee, K. *Workflow Management: Models, Methods, and Systems*; MIT Press: Cambridge, MA, USA, 2004.
36. IEEE Computer Society. *IEEE Standard 1516-2010 for M&S-HLA-Framework and Rules*; IEEE: New York, NY, USA, 2010.
37. IEEE Computer Society. *1516-2010—IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules*; IEEE: New York, NY, USA, 2010.
38. Borshchev, A.; Karpov, Y.; Kharitonov, V. Distributed simulation of hybrid systems with AnyLogic and HLA. *Future Gener. Comput. Syst.* **2002**, *18*, 829–839. [CrossRef]
39. About | Functional Mock-Up Interface. Available online: https://fmi-standard.org/about/ (accessed on 15 September 2020).
40. King, D.H.; Harrison, H.S. Open-source simulation software "JaamSim". In Proceedings of the 2013 Winter Simulations Conference (WSC), Washington, DC, USA, 8–11 December 2013; IEEE: New York, NY, USA, 2013; pp. 2163–2171.
41. *Semantics of a Foundational Subset for Executable UML Models (fUML)*; OMG Publication: Milford, MA, USA, 2018.
42. Possik, J.J.; Amrani, A.A.; Zacharewicz, G. Development of a co-simulation system as a decision-aid in Lean tools implementation. In Proceedings of the 50th Computer Simulation Conference, Calgary, AB, Canada, 30 July–3 August 2018; Society for Computer Simulation International: Vista, CA, USA, 2018; p. 21.
43. *Unified Modeling Language (UML)*; OMG Publication: Milford, MA, USA, 2017.