HLA-based time management and synchronization framework for lean manufacturing tools evaluation

Jalal Possik, Grégory Zacharewicz, Anne Zouggar, Bruno Vallespir

HLA-based time management and synchronization framework for lean manufacturing tools evaluation

Jalal Possik[1], Gregory Zacharewicz[2], Anne Zouggar[3] and Bruno Vallespir[3]

[1]FGES, Université Catholique de Lille, France
[2]Laboratory for the Science of Risks, IMT Mines Ales, France
[3]CNRS, IMS, UMR 5218, Université de Bordeaux, France

Abstract

Discrete event simulation (DES) is a method for digitally replicating the behavior and performance of real-world processes, systems, and facilities. DES is widely applied in manufacturing, logistics, healthcare, and military domains. In some cases, DES method is insufficient. On one hand, the simulation must be disassembled into subsystems and then distributed on a multiprocessing environment to enhance the simulation performance. On the other hand, to expand current functionality and prevent future application development, a set of interacting simulations is required. In this project, the IEEE high-level architecture (HLA) standard mechanisms are adopted to solve the interoperability problems between heterogeneous components. Time synchronization between federates is essential to have all DESs running in parallel. In this paper, we present a distributed simulation framework designed to assist decision-makers in making the best decisions and prioritizing the adoption of Lean tools and techniques.

Keywords

Modeling and simulation, distributed simulation, HLA, time synchronization, discrete event simulation, Lean manufacturing

1. Introduction

In the past 50 years, the advancements in simulation software and the computing field have helped DES (Discrete Event Simulation) to become one of the most widespread modeling and simulation techniques. The pioneering period dates to the late 1950s and continues to the 1960s. During this period, simulations were performed on first-generation computers. These simulations were designed and developed using machine code. Moreover, the 1960s witnessed drastic improvements in the field of DES. This is due to the development of programming languages and the increased reliability and power that computers could offer during that time. In addition, many simulation software were developed in this period such as GPSS and SIMSCRIPT.[1] The period of innovation started in the 1970s. It embodies persistent improvement and innovation. Simulation software continued to progress along with the advancements in the computing field. Numerous new programming languages appeared (i.e. SLAM, GPSS-H, etc.).[2] In addition, microcomputers were first introduced in the late 1970s. During that time, everything was being prepared for the ''revolution'' to occur in the 1980s where microcomputers became more commonly available in organizations and enterprises due to IBM's introduction to the market. During this period, powerful microcomputers became accessible by most organizations, and new simulation systems and languages appeared (HOCUS, SIMAN/CINEMA, and GENETIK) incorporating new features especially for manufacturing systems' modeling and simulation.[3] Thus, the manufacturing sector started adopting DES as a decision-aiding tool. Since 1990s, the world witnessed the success of personal computers, workstations, and servers, as well as a remarkable evolution of Windows, Mac, and Linux operating systems. These technologies helped the field of DES and enabled models to be executed at high-speed rates.[4] The evolution of DES covered many areas such as software integration, visual interactive modeling, and simulation optimization, and it is

Corresponding author:
Jalal Possik, FGES, Université Catholique de Lille, Lille F-59000, France.
Email: jalal.possik@univ-catholille.fr

widely applied in manufacturing, logistics, healthcare, and military domains.[5]

In some cases, DES alone is not an effective solution. The simulation system must be disassembled into subsystems or nodes to be parallelized or distributed on a multiprocessing environment for performance enhancements.[6] In other cases, a collection of interacting simulations is needed to form a prominent system that extends the existing one to offer additional functionalities while distributing the components on a network of processors.[7] For all the aforementioned scenarios, time management and synchronization mechanisms are necessary to avoid timing discrepancies and to ensure precise event interconnections and data communication between subsystems or simulations. In addition, companies are constantly looking for a solution to assure interoperability between diverse heterogeneous systems and applications. Interoperability becomes crucial for the modern economy and a key requirement for most companies. Nevertheless, interoperability is a difficult and complex task to achieve. To solve interoperability issues, a tremendous advancement was made through simulation integration resulting in the birth of distributed simulation (DS).[8] Different standards exist for DS. In this project, the IEEE high-level architecture (HLA) standard mechanisms are adopted for the DS implementation process.

Concurrently, managers and engineers are in continuous search for supported methodology and cross-analysis for effective Lean management implementation.[9,10] One of the major challenges that managers face is the difficulty to choose the Lean tools and techniques that best fit their company and lead toward better productivity and quality.[11–14] Managers are not taking into consideration the context in which Lean tools should be applied, they often rely on their own experience and subjective appreciation. However, there exists a relation between the context and Lean techniques. Consequently, different scenarios showed Lean implementation failures. This constitutes a hypothesis in this project. The research literature consolidates this research path to identify and analyze the possibility of contexts influencing the choice of the prior Lean techniques to implement. Indeed, the scientific context is targeting to test various configurations revealing the beginning of context influence (e.g., market fluctuation, demand diversification, or uncertainty of resources). Furthermore, using the right tools in a convenient context reflects the company's profitable or poor implementation of Lean.

The goal of this project is to guide decision-makers willing to implement Lean Manufacturing in their industries to choose the right Lean tools that suit their industry and economic contexts. For this purpose, a DS framework has been developed to simulate an industrial model in parallel and simultaneously with the model having Lean tools applied on (predefined tools are set in place for simulation target). A graphical interface has been developed for the users to choose the Lean tools to load, test, and experiment. The user can also fill in the information related to the market demand, the number of references, the setup time and processing time of each machine, the travel time between machines, the planned/unplanned downtime of each machine, the defects rate, and so on. Users are also able to start/pause/stop the simulations or change the simulation speed factor from this platform. Federates that represent the Lean tools can run on a network of processes, on different machines and different operating systems, which makes this framework powerful and independent from the computer resources.

By varying the data input and the industrial contexts during the simulation run, the user can easily compare the simulation results and choose the ultimate Lean tools that best fit the organization's production and financial targets. The remaining part of this paper provides a structured approach of DS framework developed to support the decision-makers in taking the right choices and priorities for the implementation of Lean tools and techniques.

2. Literature review

The simulation in manufacturing and supply chain fields became a widespread scientific approach because of the ability to reproduce a virtual system that simulates the real production system.[15] In addition to ''What If'' analyses to observe and understand the Supply Operations[16] and to forecast the impact of alternative configurations,[17] the DES, in particular, is becoming one of the preferred research topics nowadays especially in the production domain.[18] DES was often considered as a dynamic tool that allows the visualization and quantification of technological and operational changes in processes.[19] It is considered an effective tool for process improvement. In addition, it is a method used in different environments (manufacturing plants, queuing systems, distribution systems, inventory and delivery systems, healthcare, transportation networks, communication networks, and many others) to simulate a real system or process. DES is a frequently used tool for Production Planning and Control problems. It represents more than 45% of the simulation models in the studied samples.[20] Further studies have made the attempt of combined methods as DES and Agent technology for studying complex supply network[21] to be able to integrate micro-behaviors of individuals and macrosystem to guide the managers in their decision-making process. In a complex production environment with a complex demand evolution, many authors use DES to quantify the effect of Lean implementation on performance measures.[22,23]

However, the traditional DES approach becomes insufficient in complex simulation contexts, requiring interoperability between various components and resource-intensive calculations. Over the last two decades, hybrid

DS, defined as an approach that integrates multiple modeling methodologies (DES, system dynamics, agent-based simulation, etc.) has increased in popularity in the modeling and simulation field.[24,25] In this case, DS becomes necessary to solve the interoperability problems between multiple heterogeneous components. The majority of the early DS applications were developed in the defense sector to train the army at a minimal cost and without risk.[26] Several simulators, including flight simulators and computer-generated forces, were integrated to create a fully immersive virtual environment for hypothetical and real-world modeling and simulation scenarios.[27] Some hybrid, parallel, or distributed systems and applications have been developed outside of the military industry. To simulate a complicated transportation network, Bae et al.[28] created a hybrid and collaborative model that combines agent-based simulation, an auction mechanism, and optimization. Kim et al.[29] built a parallel simulation system to model and control air traffic operations. To address interoperability concerns and improve the reusability of discrete event systems, Cao[30] designed a distributed interactive simulation based on HLA for emergency material delivery. Different hybrid and distributed systems and simulations have been developed in the production and manufacturing domain to improve manufacturing processes, prevent failures, shorten lead and repair times, and optimize production systems.[31–33] However, to our knowledge, no simulation system has yet been created to compare the efficiency of Lean tools based on the industrial context of the company in order to avoid the adoption of tools that will not benefit the industry or that might lead to financial failures.

Nevertheless, in terms of both dynamism and heterogeneity, DS systems are becoming increasingly complex and difficult to implement. The level of difficulty is characterized by the interoperability barriers in terms of data, applications, and middleware heterogeneities.[34] In addition to the interoperability barriers, time management and synchronization between different running simulations are considered another challenge that may face a DS implementation.[35] There are two main DS standards: a European standard called functional mock-up interface (FMI), which was recently developed and managed as a Modelica Association Project (MAP), and an international standard called HLA, which was developed in the early 1990s by the US Department of Defense (DoD) and transformed into an open international IEEE standard in the year 2000.

FMI supports model exchange and co-simulation of dynamic models based on Extensible Markup Language (XML) files and compiled C code. This standard was developed during a project named MODELISAR. It is now developed and managed as an MAP.[36] FMI 1.0, published in 2010, is the first version of the FMI. This version was followed by FMI 2.0 issued in 2014. FMI was developed to improve the simulation models' exchange between the suppliers and the original equipment manufacturers. It is now supported by more than a hundred simulation tools mostly used in automotive industries.[37] HLA standard has three versions: HLA 1.3 published by the DoD in 1998, HLA IEEE 1516-2000 published in 2000 by IEEE, and HLA IEEE 1516-2010 known as HLA evolved. The next version of HLA (HLA 4) is currently under development. This version includes new object modeling capabilities as well as simulation security improvements.[38]

Due to certain significant limitations in the FMI standard, this work employed the HLA standard to communicate data among running models. FMI lacks the time management and synchronization features found in the HLA standard.[11] In FMI, programmers must create a master algorithm to coordinate the steps of the co-simulation. The master algorithm is responsible for the synchronization and the exchange of data between the simulation models. FMI does not have HLA mechanisms that enable the interaction with external heterogeneous DS components. Such interactions are made possible by HLA's publish/subscribe (p/s) and time management mechanisms. Furthermore, FMI does not allow time stamp events, making event-driven simulations difficult and time-consuming to conduct. Moreover, FMI is reliant on the master unit and operates as a single black box entity, which is not the case under the HLA standard.[39]

3. Materials and methods

The main goal of HLA standard is to enable the simulation interoperability and reuse of simulations distributed on local or wide area networks and implemented in different programming languages and operating systems. The connected simulation components form a federation. The components are called federates. Using the p/s mechanism of HLA, components can exchange data and collaborate information. The HLA is a centralized architecture, where all connected federates are connected to the central runtime infrastructure (RTI). The federation object model (FOM) in Figure 1 is an XML file that contains all shared objects/attributes and interactions/parameters between connected federates. The developed framework discussed in this paper is composed of a master federate developed on Java and different DES models. The master federate is responsible for the orchestration process of all running models. One of the models represents the current scenario of the simulated company and the others represent the Lean tools implemented in the company. The HLA-based DS framework is developed using the Java library of pitch technologies[40] and JaamSim DES for the models' development.[41]

Each of the linked federates begins by determining whether the federation exists in order to build or join the federation accordingly. Following the registration of object

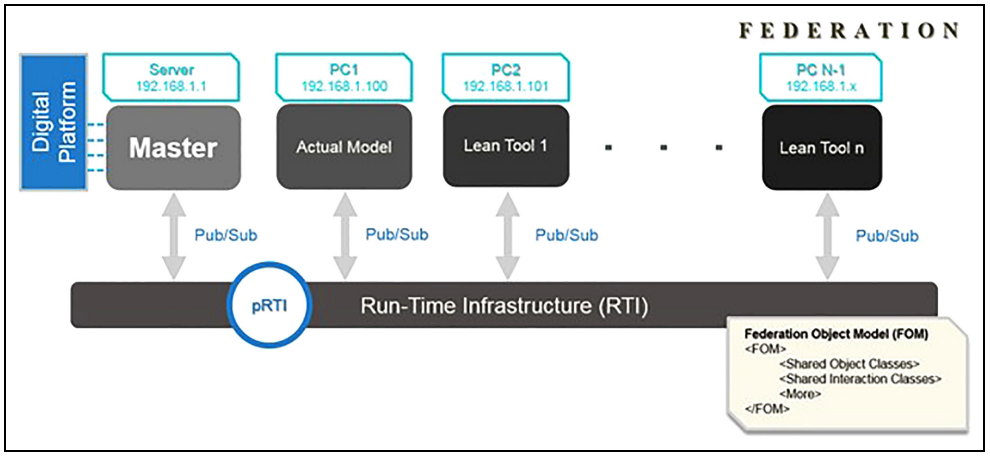

Figure 1. HLA federation.

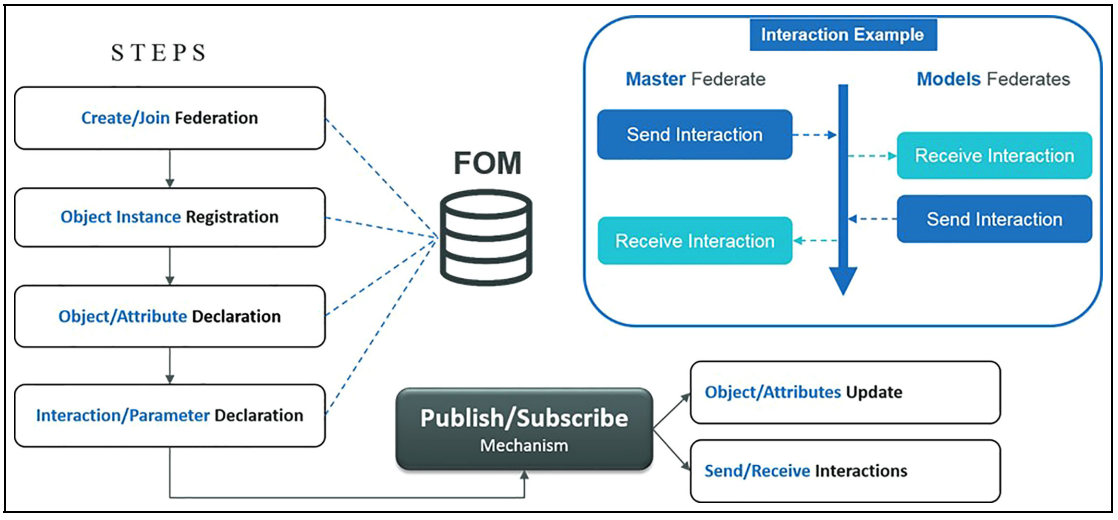

Figure 2. Publish/subscribe mechanism of HLA.

instances, each federate declares the Objects/Attributes and Interactions/Parameters that should be used for either the publishing or subscription processes (refer to Figure 2). The p/s mechanism of HLA is employed to collaborate data between federates. Data can be exchanged in two ways: as an interaction with its parameters or as an object with its attributes. When a model delivers an object or an interaction to the RTI throughout the simulation process, all subscribed models receive this object or interaction at the appropriate time.

The *updateAttributeValues()* method can be used by any of the linked federates to communicate the updated objects/attributes to the RTI. Similarly, using the *sendInteration()* function, a federate can publish an interaction class with its associated parameters. Figure 3 depicts a business process modeling and notation (BPMN) flowchart illustrating a tiny example of HLA-based communication between three connected federates. In this example, Federate 1 uses the function *updateAttributeValues()* to update its registered objects/attributes. Through the method *reflectAttributeValues()*, an RTI callback is issued to all associated federates at the appropriate time. Following this, Federate 2 sends an interaction to the RTI. Through the method *receiveInteraction()*, an RTI callback is issued to all associated federates at the appropriate time.

The primary objective of this project is to simulate the basic scenario of the company on JaamSim, a DES platform, in parallel with other DESs, each of which represents one or more Lean tools applied to the manufacturing model. All the DESs, as well as an external Java application, are linked to the same RTI. Using this platform, decision-makers can induce industrial context changes during a simulation run to see how Lean tools react to such a change or fluctuation. They can experiment the

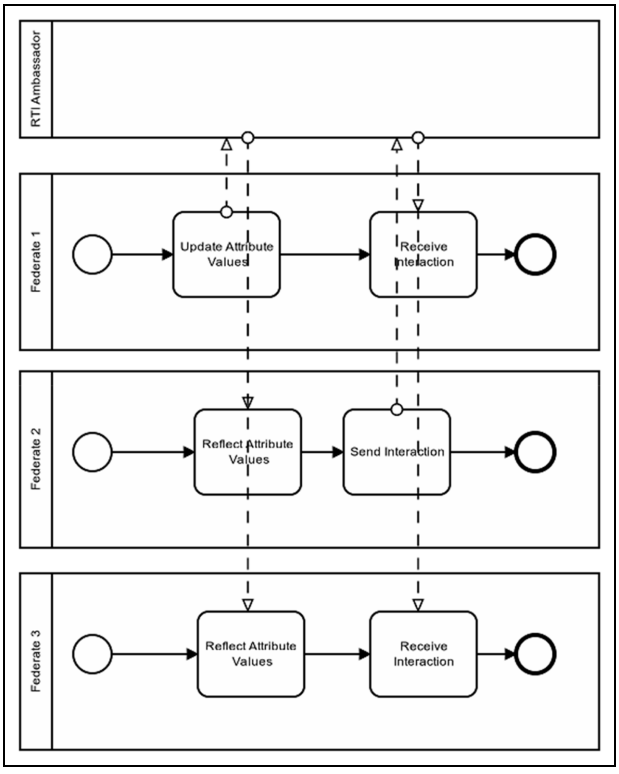

Figure 3. BPMN process to update attributes and interactions.

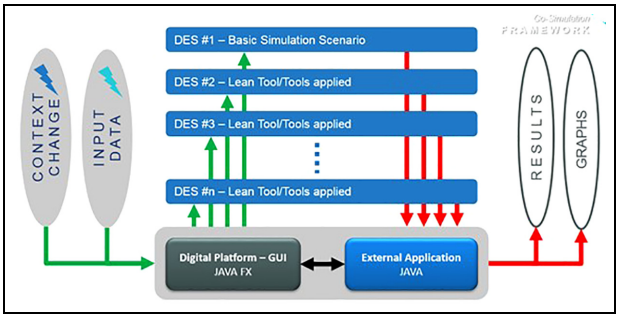

Figure 4. DS framework architecture.

tools response to a market fluctuation, demand diversification, uncertainty of resources, and others. Furthermore, during the simulation run, users can change the input data (machine processing and setup times, transfer time between workstations, planned/unplanned down time, etc.) to see how such a change might affect the production process and which tool would serve as a solution for such a change (refer to Figure 4).

The following sections discuss the time management implementation used to synchronize the operating DESs, the DES configuration, as well as the key performance indicators (KPIs) utilized to compare production results.

3.1. Time management and synchronization

The RTI offers optional time management capabilities to help federates organize the interchange of events. Events can be linked to a certain point in time, and the RTI can guarantee causal behavior. In a federation, one or more federates can completely disregard time. The RTI does not coordinate time among federates by default. One of the HLA's features is that it not only enables a diverse set of time management rules but it also forecasts interoperability between federates with different policies. Time constantly moves forward in a federation. The sense of the present time, however, may differ between connected federates. The time management mechanism of HLA is responsible for controlling the progression of each federate along the time axis of the federation. Time advances are coordinated to the object management services so that federates will get their information in a precise and causally ordered manner.

Federates can be assigned as regulating, constrained, or regulating/constrained. A regulating federate can control the logical time progress of constrained federates. The time regulating and time constrained services are initially 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 study, all DESs have the time regulating/constrained feature activated, allowing them to operate in parallel at almost the same simulation time. However, both time regulating/constrained mechanisms are disabled in the external master application because this federate receives time stamped data from each federate, making it independent from the federation time axis.

Different time advancement services, such as event-based, time-step, and optimistic, can be requested. Because this work is using event-based federates, it employs the event-based time advancement service. This service's objective is to process all events in time stamp order (TSO). In event-based federates, the method *nextEventRequest()* is called to request a logical time advancement. Each federate declares a positive value for the lookahead. The lookahead being the time delay that cannot be exceeded between simulations, it is essential to allow the processing of concurrent events having different time stamps. 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. The aforementioned HLA services are used to prevent messages from being delivered out of sequence.

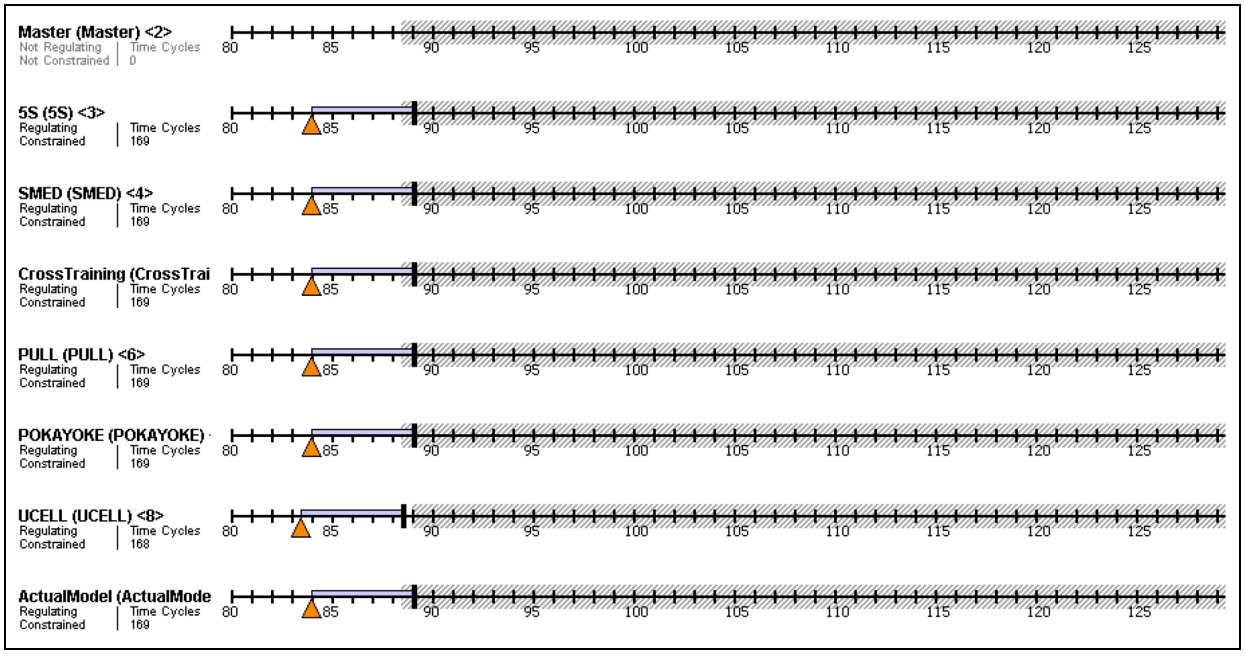

Figure 5. Time management and synchronization of connected Lean tools federates.

When a federate calls the *nextEventRequest()* method to request a time advancement and communicate new events, the RTI ensures that no messages with a TSO smaller than the lookahead time and the federate actual time combined are delivered.

The orange triangle in Figure 5 displays the current simulation time of each federate. Each federate's lookahead value is shown by the purple bar. During the simulation process, the lookahead for each federate is 5 s. The DS will be slower if the lookahead is too small. Figure 5 shows that lookahead 5 is picked as a compromise between instant parallel communication and simulation slowness.

The goal of this study is to run the DESs representing the Lean tools simultaneously in parallel and inject the same input data to all of them to test the reaction (output result) of each implemented Lean tool. Each DES simulator might have a different simulation time or running slower or faster than other simulators. This HLA configuration will pause/resume each DES numerous times during the DS run to maintain simulation parallelism. As a result, no federate's logical time can exceed the minimum existent logical time plus its lookahead. As seen in Figure 5, all simulations run at nearly the same logical time, and the lookahead is never exceeded.

3.2. DES tool

JaamSim, an open-source Java-based simulator, was utilized in this research as the DES tool. JaamSim has built-in objects for modeling and simulation. Users with Java programming skills can create new objects and modify current built-in objects. The JaamSim model may be automatically launched from the terminal or command line. To enhance performance, tags can be appended to the command to start the simulation without a graphical interface. JaamSim's graphical user interface (GUI) is split into six major components. The first component is the control panel window, which provides a variety of simulation control options. The second is the model builder, which allows the user to search for various items and choose the entities required to create the simulation model. Users can select from a variety of graphic objects, probability distributions, basic objects (FileToVector, FileToMatrix, ExpressionLogger, TimeSeries, etc.), process flow objects (EntityGenerator, Server, Queue, Branch, etc.), calculation objects (Controller, Polynomial, Integrator), and fluid objects in the model builder palette. This project was created using JaamSim version 2019-05. It is compatible with both Windows and Unix operating systems.

JaamSim is a black box simulator that does not communicate with other systems and is incompatible with DSs. A significant effort has been put into building an HLA interface for JaamSim, which will allow us to link JaamSim to external HLA compliant components in order to collaborate and exchange data with those components. This work introduced a function to all JaamSim objects that reads all attributes set to JaamSim objects; if an entity contains an attribute named ''waitRTIOrder,'' it will be regarded as an HLA entity capable of collaborating with other systems and exchanging data, as shown in Figure 6. This entity

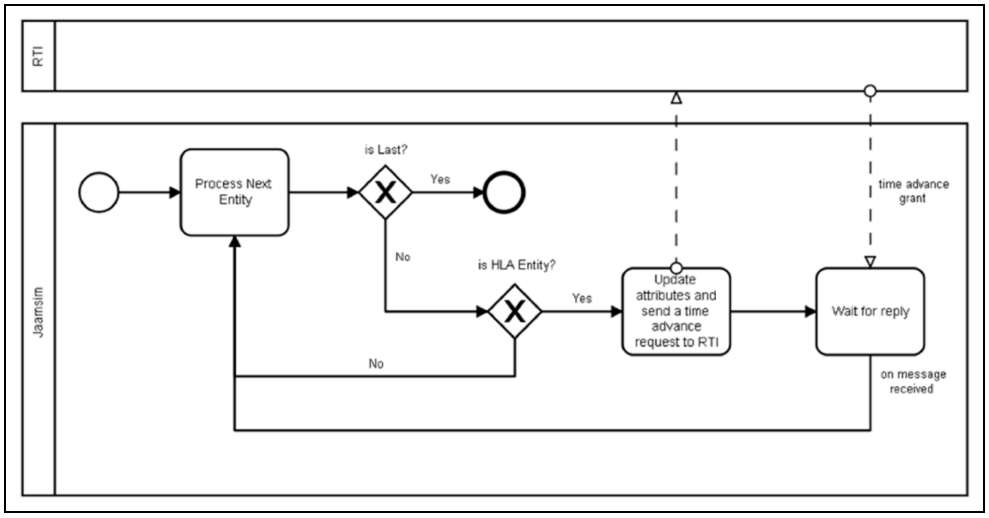

Figure 6. BPMN process of JaamSim developed HLA module.

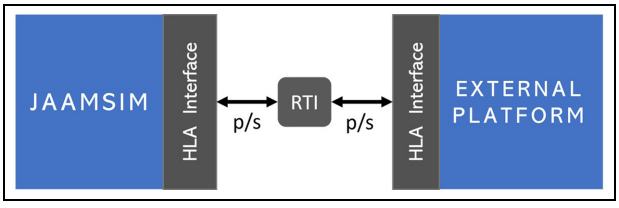

Figure 7. HLA interfaces.

seeks a time advance from the RTI during the simulation run. Following this, it will await the RTI answer before moving on to the next entity. If the next entity does not have a ''waitRTIOrder'' attribute, the following entity will be handled directly. This process is repeated throughout the simulation process until the simulation is completed.

An HLA interface is developed for the master external platform designed for users to visually define their suitable models as well as the Lean tools settings. As shown in Figure 7, both interfaces are linked to the same RTI in order to exchange messages (objects/attributes and interaction/parameters). Messages are transmitted in both directions: JaamSim gets the initial data input from the External platform before or during the simulation run, and then delivers the output result to the External platform, which draws them into real-time graphs.

3.3. *A quick overview of the case study and tested Lean tools*

The case study used in this work comes from the aeronautical industry sector and is named here ''Aerocomp'' for confidentiality.[42,43] The product manufactured by Aerocomp is an aeronautical fastener consisting of a metallic cylinder section with bearings on the right and left sides. The gears are then welded and fitted onto the back of the metallic cylinder. The metallic cylinder has a certain length and diameter given by the client in a specification document. Raw materials are first transported to the cutting shop based on the order book, where the metallic cylinder is cut to the exact specifications requested by the client. The goods in process are then transported to the treatment shop, where a layer of zinc is applied to the product. The product is subsequently sent to the assembly shop, where four workstations create the semi-finished axis, add the bearings, and then fix the gears. Finally, it is delivered to the machining shop, where two workstations insert the pins and send the finished aeronautical fastener to the warehouse for distribution (refer to Figure 8).

The Lean-free situation is represented by the current model (Scenario 0). The following describes briefly the Lean tools implemented in this project:

The *Pull technique* seeks to reduce and eliminate overproduction. In the Pull scenario, when a machine's work in progress (WIP) surpasses a preset number of units, it sends a signal to the upstream machine to cease supplying items in process.[22,12,44]

SMED is designed to reduce/eliminate waste caused by a shortage of materials, to keep tools and machines clean, and to manage the workshop space associated with setup/changeover procedures. SMED decreases the changeover time required to setup a machine.[45,46]

5S seeks to create a self-explanatory, organizing, and improving workplace. It is a collection of concepts that enhance the workplace environment, which improves the quality and efficiency of production.[47,48]

Cross-training seeks to develop workers' multiskills. This broadens the scope of the job and ensures that the operators' workloads are balanced.[49,50]

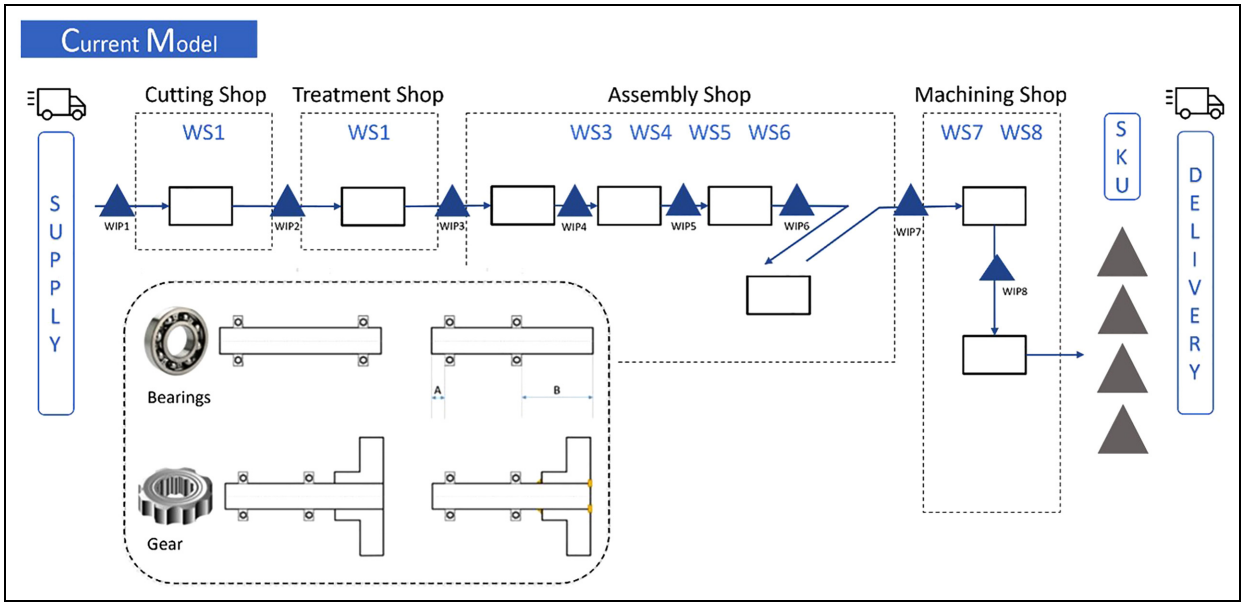

Figure 8. Case study JaamSim model.

Ucell is concerned with product flow. Machines are situated near to one another to reduce transfer time between them.[11,51]

Poka Yoke means "mistake-proofing." This is a straightforward tool that prevents defective goods in progress from being sent to the next step. The fundamental idea behind this method is to detect, remove, and repair mistakes at their source before they reach the consumer.[52,53]

3.4. Operating instructions for the platform (master application)

The master component builds the HLA federation after the platform is started. Following the establishment of the HLA federation, the master federate joins the federation and launches the other federates, which also join the federation. It is worth noting that the simulations have not yet been loaded; they have only joined the HLA federation. At this point, 5S, SMED, Poka Yoke, Cross-training, Pull, and Ucell federates are all linked to the federation's RTI, along with the master federate (external application). The communication between federates is based on the p/s mechanism of HLA. All configured objects/attributes and interactions/parameters can be found in the FOM file of Figure 9.

The user begins by running the master federate and choosing one or more Lean tools to load before beginning the simulation process. For instance, in Figure 10, all available tools are selected (5S, SMED, Poka Yoke, Cross-training, Pull, and Ucell). To load these DESs, the user hits the "LOAD" button. The Master federate publishes the interaction *ScenarioLoad* to the RTI, and all subscribing federates to the *ScenarioLoad* interaction load their DES accordingly. The federate that successfully loaded its simulation publishes the interaction *ScenarioLoaded* with the name of its federate (as a parameter—*FederateName*) to the RTI. When a federate encounter an error while loading its simulation, it publishes the interaction *ScenarioError* with both arguments, *FederateName* and *Error*. The master federate, which is the sole federate subscribed to *ScenarioLoaded* and *ScenarioError* interactions, receives all loaded scenarios with their federate names, as well as all scenarios that experienced issues (*FederateName* and the *Error* encountered).

After loading the DESs, the user has the option of changing the simulation speed factor on the master federate, which is set to "1.0" by default. The simulation speed may be doubled by hitting the "+" symbol button. Using the "−" symbol button, the user may slow down the simulation. Furthermore, the user may enter the simulation speed factor in the text box. A *SimulationControl* and *RealTimeFactor* interactions are configured to publish simulation speed data to all JaamSim federates.

The data entry procedure comes next. In this work, the master federate plays two major roles. The first role is to provide all running models with input data, and the second is to collect the outcomes of the DS from running federates in order to plot them and provide the user with graphical results information. Input/output data are configured as objects/attributes in the HLA federates. The FOM file in Figure 9 displays the input data that must be entered on the master federate, such as the processing time required on each machine, the setup time required to switch from one

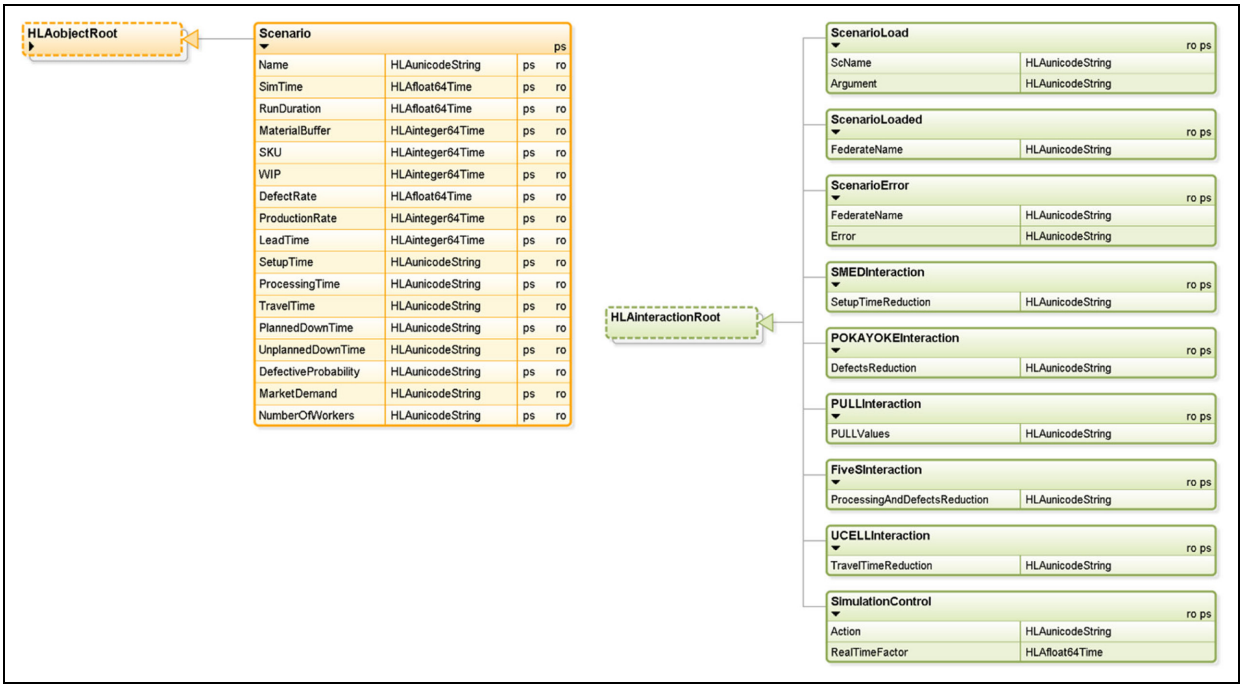

Figure 9. FOM file configuration.

Figure 10. Master platform home interface.

type of product to another, the travel time between workstations, the planned and unplanned downtime intervals for each workstation, the time required to fix the downtime on each machine, the number of workers in each shop, the defect rate, the Lean tools configuration, and others. In addition, Figure 9 displays the output variables, including lead time, WIP, production throughput, and defect rate, that the master federate uses to depict the co-simulation outcomes during the simulation run. All of the input/output variables mentioned above have been added to the FOM file and configured as objects/attributes in each federate's HLA interface. However, both in the FOM file and the HLA interfaces, all of the interactions that take place are set up as interactions/parameters. Any of the connected federates can use the *updateAttributeValues()* function to send the modified objects/attributes to the RTI. A federate can also publish an interaction class with its associated parameters by utilizing the *sendInteration()* method.

When the user clicks the ''Send'' button, the master federate publishes the objects/attributes to the RTI, and based on the p/s mechanism, the RTI then distributes them to all subscribed operational federates.

The user then clicks the green ''Start'' button to concurrently run all loaded simulations. The co-simulation is handled by the time management services and mechanism of HLA using both methods *nextMessageRequest()* and *timeAdvanceGrant()*. A federate must request the time advancement during the DS run and wait for the RTI callback function timeAdvanceGrant() in order to move forward in simulation time. This regulates how quickly each federate advances in time, enabling us to run the models concurrently. This process is detailed in section 3.1.

Simulations can be paused or terminated at any time. During the simulation run and based on HLA's p/s and time management mechanisms, if the user changes any of the input data, this change will be received by all ongoing simulations. Furthermore, the output data supplied to the master external application will alter accordingly. This enables us to compare the outcomes of Lean tools in response to any input or context change.

3.5 KPIs

Performance indicators are the instruments used by decision-makers and managers to assess, comprehend, and verify whether the business is on track to meet its goals or deviating. The establishment of KPIs should begin with the firm's plans and the goals that the organization wishes to achieve. It is critical to select KPIs that are tailored to the organization's specific requirements and conditions.[54] In this study, we picked four KPIs to measure the industry's four primary objectives (quality, flexibility, cost, and responsiveness). The developed platform is integrated into the master federate, which oversees receiving/sending data from/to other federates (Lean tools). KPI values are shown concurrently in real-time throughout the simulation process. The following are the KPIs that were used.

3.5.1. KP1—lead time. Lead time is the amount of time needed to fulfill the customer's request, from the time the order is received until the final product is delivered.[55] Companies strive to reduce lead times and meet client deadlines on a continuous basis. This section explains the average lead-time calculation, which is the total number of lead times divided by the total number of orders placed, because the lead time is most likely different for each order. The value of lead time on the lead-time graph is automatically updated for all ongoing DESs representing the different Lean tools retained in this study.

Knowing that:

- S_p: set of products produced by the company. Each reference is denoted by XRF_i, where $i = 1, \ldots, n$.
- $S_p = \{XRF_i | i = 1, \ldots, n\}$.
- (t): period of time over the global planning horizon.
- $D_{XRF_{it}}$: demand of the product XRF_i at a period of time (t).
- $\hat{D}_{XRF_{it}}$: demand fluctuation of the product XRF_i at a period of time (t).
- $\overline{D_{C_u}}$: variety of demand required by client (C_u).
- Q_{ti}: quantity of products of type (i) produced at period (t).

The designed KPI can be written as follows:

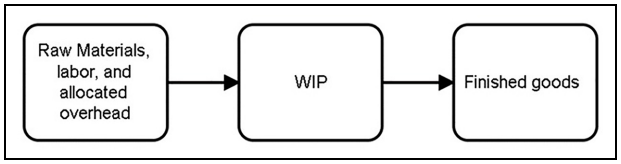

Figure 11. Manufacturing stages.

$\forall i = 1, \ldots, n,\ t \in \{0, \ldots, m\}$, and whatever are the assigned values to $D_{XRF_{it}}$, $\hat{D}_{XRF_{it}}$, $\overline{D_{C_u}}$, and Q_{ti}, we define the following elements:

Ω_{i0}: the simulation time where the product XRF_i started the production process;

Ω_{iF}: the simulation time of product XRF_i at the end of the production process.

$$KP1 : leadtime_{XRF_i} = \Omega_{iF} - \Omega_{i0}, \text{lead-time of the product } XRF_i$$

3.5.2. KP2—WIP. In this production case study, there are only stocks at the beginning (raw materials) and at the end (finished goods) of the production process; therefore, it will be considered that the WIP is directly linked to the cost of partially finished products in the manufacturing process.[56] As shown in Figure 11, the manufacturing process is divided into three stages: raw materials, WIP, and finished products. The cost of production can include the cost of raw material storage, the cost of WIP within production lines, the cost of finished goods in the warehouse, the cost of daily human labor, the cost of machines, and other costs. In all cases, the assumption is that the cost of daily human labor and the cost of machines are identical. They reflect the common stable costs, and the variations in scenarios will have no effect on this stable part but will influence the variable elements such as WIP. The costs of raw material storage and completed product storage are considered neglected since the supplier is regarded as a reliable partner that delivers the required components on time. The finished products are delivered as soon as they are produced. As a result, the cost of WIP inside the production line can be testimonies about the variation of production costs. Analysts can track the WIP inventory of the company to guarantee that costs are allocated properly.[57]

The major purpose of maintaining WIP as low as possible is to reduce the associated expenses with in-process products (in the machines and in the queue). WIP products, in fact, necessitate storage, as well as floor space and a variety of utilities to keep them running. Furthermore, warehouses may consume electricity, and labor costs are frequently required to preserve and safeguard WIP products. Furthermore, WIP in queue will block the production flow, resulting in decreased

production rates and, as a result, missed client deadlines. When raw materials are introduced into the manufacturing process, they become WIP. However, if they have not yet completed the entire production process, they are considered unfinished goods.

Regarding the case study and the steps through which the products are moving, we can write the following:

$\forall M_p, p = 1, \ldots, U$, $\forall i = 1, \ldots, n$, $t \in \{0, \ldots, m\}$, $D_{XRF_{it}}$, $\hat{D}_{XRF_{it}}$, $\overline{D_{C_u}}$, and Q_{ti}, we define the following elements:
$queue_{M_{pt}}$: the queue of machine M_p at time t;
$\varepsilon_{M_{pt}} \in \{0, 1\}$: takes the value 1 if M_p is in a working state, otherwise it is equal to 0.

$$KP2 : WIP_t = \sum_{p=1}^{U} (\varepsilon_{M_{pt}} + queue_{M_{pt}}), \text{WIP at time } t$$

3.5.3. KP3—production throughput. Production throughput refers to the quantity of products that can be produced/manufactured within a period of time.[58] The production throughput is calculated each day (eight working hours) in the developed DS framework and is updated during the simulation process. Product complexity and nature, machine setup times, defective goods, worker skills, and other factors can all have an impact on production throughput.

Production throughput can be expressed as follows:

$\forall i = 1, \ldots, n$, $t \in \{0, \ldots, m\}$, $\forall k = 0, \ldots, 241$, the number of working days over 1-year simulation horizon, each day being equivalent to eight working hours.

$$KP3 : Production\ throughput = \sum_{i=1}^{n} \sum_{t=8k+1}^{8k+8} Q_{ti}, \text{ production throughput per day}$$

3.5.4. KP4—defect rate. The defect rate is the most accurate predictor of product quality.[59] It is used to monitor and assess productivity, projects, services, programs, and processes. Companies strive to lower defect rates in order to improve product quality. In addition, lowering the failure rate enhances on-time delivery and production throughput.

Defect rate can be expressed as follows:

$\forall i = 1, \ldots, n$, $t \in \{0, \ldots, m\}$, $\forall k = 0, \ldots, 241$;
$\tau_{ti} \in \{0, 1\}$: takes the value 1 if the product XRF_i is defective, otherwise it is equal to 0.

$$KP4 : defect\ rate = \left(\frac{\sum_{i=1}^{n} \sum_{t=8k+1}^{8k+8} \tau_{ti}}{\sum_{i=1}^{n} \sum_{t=8k+1}^{8k+8} Q_{ti}} \right) \times 100, \text{ defect rate per day}$$

The aforementioned KPIs are used in this work to demonstrate the impact of each Lean tool on the manufacturing system and how it can effectively help a company achieve its key objectives. Figure 12 displays the general concept of the DS framework. During the simulation run, industrial context changes are induced to the seven models running in parallel. The first model represents the current situation of the company depicted in Figure 8. The other models correspond to the same model that has been modified with Lean tools' implementation. Attractive charts are set up to display the co-simulation results based on the KPIs indicated above, allowing the user to monitor the performance of each applied Lean tool. These KPIs stand in for the business's industrial goals. This paper emphasizes the simulation part of this work. Possik (2019) describes in full the experiments and findings that are relevant to production and industrial concerns.[11]

4. Results and discussions

The major goal of this project was to present a co-simulation framework that simulates several Lean tools along with the current model of an aeronautic company in parallel in order to investigate the outcomes and impacts of the tools on the production process. Decision-makers that need to research how lean techniques will affect their industrial goals will find this DS tool to be of great relevance. Users of this tool only need to alter one or more input variables (such as the processing and setup times of machines, the travel distances between workstations, the Lean configuration, unplanned and planned downtime, and others) or an industrial context (such as market fluctuations, demand diversification, resource uncertainty, and others) to see which Lean tool responds to the change more effectively. In the KPI graphs, each Lean tool/technique is represented by a line color (see Figure 13).

A 1-year simulation for each model takes around 8 min on a workstation with an Intel® Xeon® E5-1600 v4 Processor (up to eight cores, 3.7 GHz), 16 GB of RAM, an NVIDIA® Quadro® P6000 graphics card, and Linux operating system. Running the seven models sequentially would take roughly 56 min, without including the time required to compare the models (Lean tools) for each input alteration to identify which tool is performing better than the others. However, running all the models in parallel takes approximately 8.5 min. Given that each federate is operating on a different computer with the same specifications, the network communication and

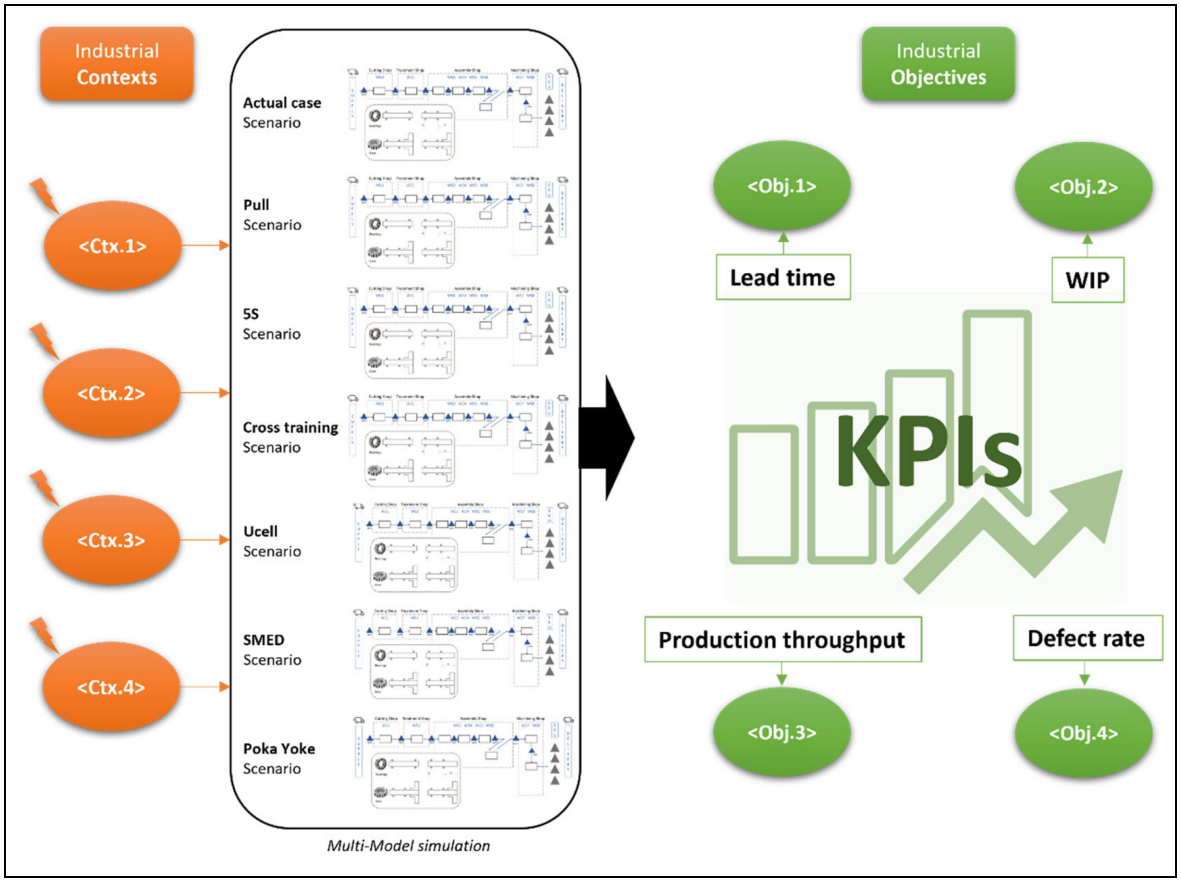

Figure 12. Overall concept.

time management/synchronization used to make the federates run in parallel account for the 0.5-min difference between running one tool and running all the tools simultaneously. This showed that the DS performs 84.8% faster than running the simulations serially. In addition, on the DS platform, the decision-makers have their KPI results plotted on a single window, with the outcomes of each tool shown in a distinct color. This makes it incredibly simple and obvious for users to select the tool that will respond to input/context changes the most effectively. The outcome comparison in serial simulation must be performed manually, which adds to the process' complexity and time requirements.

Figure 13 depicts the outcomes of three different scenarios run on the DS architecture to test the responsiveness of Lean tools. The red circles in the graphics indicate the location of the disturbance or context fluctuation. In the market fluctuation scenario, 5S demonstrated to be the best in class during these fluctuations. The WIP value was improved by SMED and Poka Yoke. However, at a strong demand rise (+30%), both were unable to limit WIP overcapacity, resulting in a high WIP growth. In a market fluctuation scenario, Ucell and Cross-training have no substantial improvements in production.

The production of 2, 4, 8, and 16 references has been tested in the demand diversification scenario. Lead-time value is reduced during this scenario with Poka Yoke and SMED tools. However, when the number of references reached a specific threshold, the aforementioned tools experienced overcapacity in their WIPs, resulting in an increase in lead times. When it comes to demand diversification, Pull and 5S are the best in class. However, when the number of product references was extended to 16, the lead time for 5S grew dramatically. Even though, Pull kept the lead-time consistent. During an operator disturbance, Cross-training was the only tool capable of keeping the production running.

The preceding findings are useful and should be considered when a company is confronted with any of the scenarios listed above. Actually, we highlighted the application of the developed DS framework for enterprises interested in experimenting the implementation of Lean tools in their manufacturing processes. This tool can be used with many settings and configurations to suit the needs of users' businesses or industries. They can test a variety of scenarios, adjustments, and input configurations.

Simulating each tool sequentially, waiting for findings, and storing them keep the user from performing

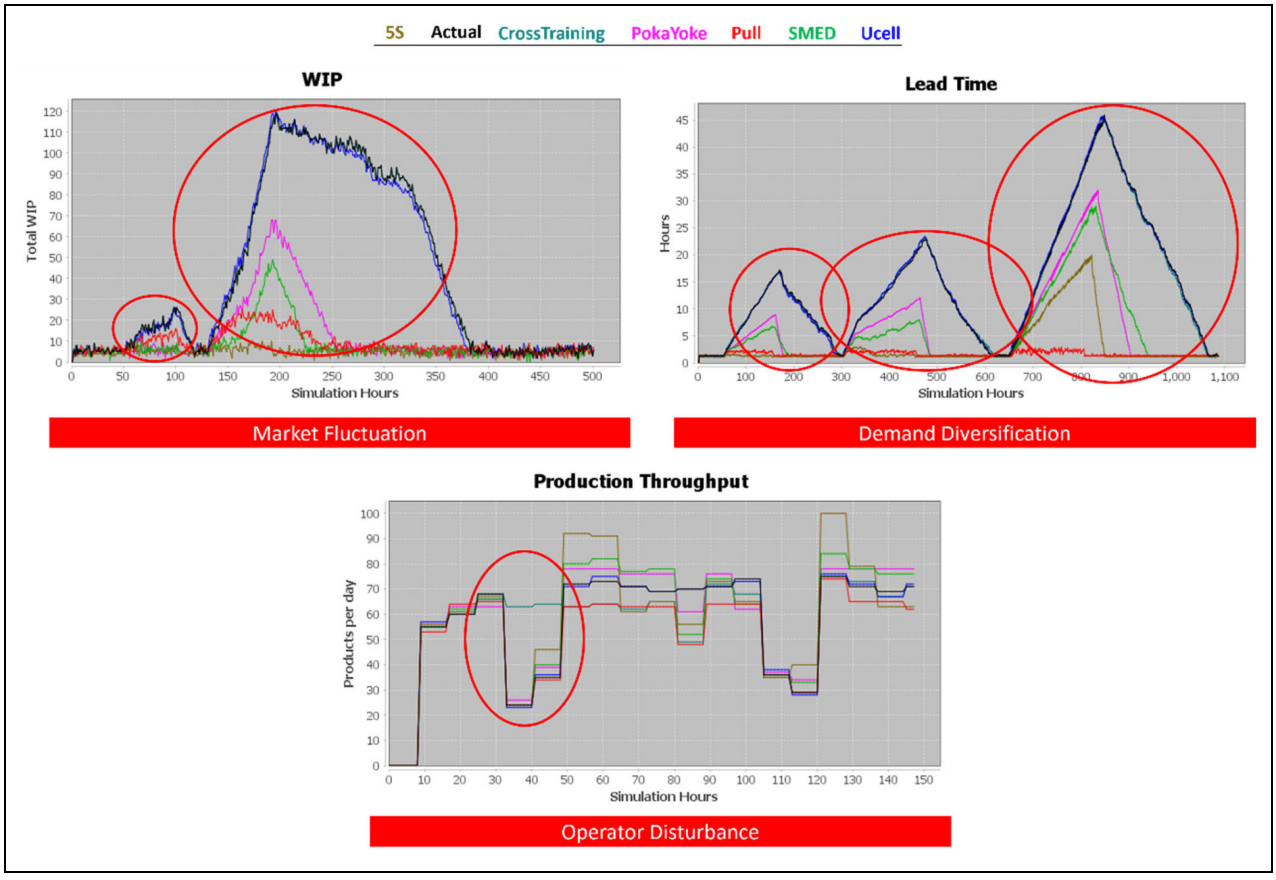

Figure 13. Lean tools response to disturbance and context changes.

immediate analysis and force him to cumulate the output results on each simulation run. The aim behind the new digitally produced platform is to allow for a common ''input introduction'' for several tools running concurrently. The developed DS framework constitutes a decision-aided framework for managers to help them in selecting Lean tools that best suit their organization production and financial targets. The parallel HLA-based simulation allows real-time monitoring of Lean tools response, which helps managers to choose one Lean solution versus another. This platform is simple to use, with the production line represented by modular components that the user may edit, combine, move, and remove to depict his own industrial system. Market condition updates, internal production line execution, and any type of disruption can be represented, modeled, parameterized, and simulated.

5. Conclusion

Leading manufacturers are progressively incorporating Lean principles into their manufacturing processes. Lean methods and approaches are becoming increasingly important in eliminating or reducing waste and non-value activities in the production process. However, Lean deployment necessitates an in-depth examination of the company's context to implement appropriate Lean practices and secure financial and quality improvements. Many manufacturing businesses are inefficiently implementing Lean tools, even though Lean offers benefits regardless of the form of the applied tools; most of these organizations are experiencing failure. This paper introduces a DS platform that enables interoperability between a master federate and other components representing the implemented Lean tools and techniques to experiment their response to disturbance and industrial context fluctuations. A significant effort was put into designing a new module for JaamSim DES tool to convert it into an HLA compatible DES tool capable of interfacing and exchanging data with external federates. These functionalities were critical in this research because they allowed us to run all the modeled Lean tools (federates) in parallel and impose input data changes throughout the simulation run to test the behavior and responsiveness of the Lean tools.

The use of HLA standard broadens the horizons and opens the door for the development of additional Lean tools. Six Lean tools are developed till now; the goal is to expand the built co-simulation framework to gradually

integrate additional Lean techniques. The co-simulation framework will enable us to create and run multiple Lean scenarios over a broad processors' network. Using this framework and digital platform, one can introduce modifications and disruptions in many variables from design to commercialization (market demand, travel time, processing time, setup time, planned/unplanned down time, defects, and others). Different hypothesis leading to different and diverse output results can be explored on this framework. As a future work, the reinforcement learning technique could be used to automatically produce input changes and identify the tools that are best suited to firms depending on context changes and companies' objectives.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

ORCID iDs

Jalal Possik https://orcid.org/0000-0002-5246-8102
Gregory Zacharewicz https://orcid.org/0000-0001-7726-1725

References

1. Goldsman D, Nance RE and Wilson JR. A brief history of simulation revisited. In: *Proceedings of the 2010 winter simulation conference*, Baltimore, MD, 5 December 2010, pp. 567–574. New York: IEEE.
2. Brunner DT and Crain RC. GPSS/H in the 1990s. In: *Winter simulation conference proceedings*, Phoenix, AZ, 8–11 December 1991. New York: IEEE, 1991.
3. Bell PC and O'keefe RM. Visual interactive simulation—history, recent developments, and major issues. *Simulation* 1987; 49: 109–116.
4. Hollocks BW. Forty years of discrete-event simulation—a personal reflection. *J Oper Res Soc* 2006; 57: 1383–1399.
5. Thorwarth M and Arisha A. *Application of discrete-event simulation in health care: a review*. Dublin: Technological University Dublin, 2009.
6. Possik J, D'Ambrogio A, Zacharewicz G, et al. A BPMN/HLA-based methodology for collaborative distributed DES. In: *2019 IEEE 28th international conference on enabling technologies: infrastructure for collaborative enterprises (WETICE)*. Capri, Italy, 12–14 June 2019, pp. 118–123. New York: IEE.
7. Gorecki S, Possik J, Zacharewicz G, et al. A multicomponent distributed framework for smart production system modeling and simulation. *Sustainability* 2020; 12: 6969.
8. Robinson S. Discrete-event simulation: from the pioneers to the present, what next? *J Oper Res Soc* 2005; 56: 619–629.
9. Dolgui A. Operations research techniques for design and analysis of lean manufacturing systems. *IFAC Proc Vol* 2007; 40: 11–19.
10. Rossini M, Costa F, Tortorella GL, et al. Lean production and industry 4.0 integration: how lean automation is emerging in manufacturing industry. *Int J Prod Res* 2021; 1–21.
11. Possik J. *Contribution to a methodology and a co-simulation framework assessing the impact of lean on manufacturing performance*. PhD Thesis, University of Bordeaux, Bordeaux, 2019.
12. Possik J, Zouggar-Amrani A, Vallespir B, et al. Lean techniques impact evaluation methodology based on a co-simulation framework for manufacturing systems. *Int J Comput Integr Manuf* 2022; 35: 91–111.
13. Possik J, Amrani A and Zacharewicz G. WIP: Co-simulation system serving the configuration of lean tools for a manufacturing assembly line. In: *Proceedings of the works in progress symposium*, Baltimore, MD, 15 November 2018.
14. Possik J, Amrani A and Zacharewicz G. Development of a co-simulation system as a decision-aid in Lean tools implementation. In: *2018 summer simulation multi-conference*, Bordeaux, 9–12 July 2018, pp. 1–12. San Diego, CA: Society for Modeling and Simulation International.
15. Long Q. Distributed supply chain network modelling and simulation: integration of agent-based distributed simulation and improved SCOR model. *Int J Prod Res* 2014; 52: 6899–6917.
16. Chatfield DC, Harrison TP and Hayya JC. SISCO: an object-oriented supply chain simulation system. *Decis Support Syst* 2006; 42: 422–434.
17. Neagoe M, Hvolby H-H, Taskhiri MS, et al. Using discrete-event simulation to compare congestion management initiatives at a port terminal. *Simul Model Pract Theory* 2021; 112: 102362.
18. Yoo T, Cho H and Yücesan E. Hybrid algorithm for discrete event simulation based supply chain optimization. *Expert Syst Appl* 2010; 37: 2354–2361.
19. Hong S-Y, Bal A, Badurdeen F, et al. Evaluation of bunker size for continuous/discrete flow systems by applying discrete event simulation: a case study in mining. *Simul Model Pract Theory* 2020; 105: 102155.
20. Jeon SM and Kim G. A survey of simulation modeling techniques in production planning and control (PPC). *Prod Plan Control* 2016; 27: 360–377.
21. Alavi-Moghaddam M, Forouzanfar R, Alamdari S, et al. Application of queuing analytic theory to decrease waiting times in emergency department: does it make sense? *Arch Trauma Res* 2012; 1: 101.
22. Abdulmalek FA and Rajgopal J. Analyzing the benefits of lean manufacturing and value stream mapping via simulation: a process sector case study. *Int J Prod Econ* 2007; 107: 223–236.
23. Zekhnini K, Cherrafi A, Bouhaddou I, et al. A model integrating lean and green practices for viable, sustainable, and digital supply chain performance. *Int J Prod Res* 2021; 1–27.
24. Brailsford SC, Eldabi T, Kunc M, et al. Hybrid simulation modelling in operational research: a state-of-the-art review. *Eur J Oper Res* 2019; 278: 721–737.
25. Li Y, Chen J, Hu Z, et al. Co-simulation of complex engineered systems enabled by a cognitive twin architecture. *Int J Prod Res* 2021; 1–22.
26. Fujimoto RM. *Parallel and distributed simulation systems*. Chichester: John Wiley, 2000.
27. Bocciarelli P, D'Ambrogio A, Falcone A, et al. A model-driven approach to enable the simulation of complex systems on distributed architectures. *Simulation* 2019; 95: 1185–1211.

28. Bae K-H, Mustafee N, Lazarova-Molnar S, et al. Hybrid modeling of collaborative freight transportation planning using agent-based simulation, auction-based mechanisms, and optimization. *Simulation* 2022; 98: 753–771.
29. Kim YJ, Mavris D and Fujimoto R. Time- and space-parallel simulation of air traffic networks. *Simulation* 2019; 95: 1213–1228.
30. Cao Q. Research on co-simulation of multi-resolution models based on HLA. *Simulation* 2022.
31. Mollajan A, Iranmanesh H, Khezri A, et al. Effect of applying independence axiom of Axiomatic Design theory on performance of an Integrated Manufacturing Information System: a computer simulation modeling approach. *Simulation* 2022; 98: 535–561.
32. Osman MFS. Modeling and simulation for inventory management of repairable items in maintenance systems. *Simulation* 2022; 98: 1013–1037.
33. Malega P, Gazda V and Rudy V. Optimization of production system in plant simulation. *Simulation* 2022; 98: 295–306.
34. Blair GS, Paolucci M, Grace P, et al. Interoperability in complex distributed systems. In: *International school on formal methods for the design of computer, communication and software systems*, Bertinoro, 13–18 June 2011, pp. 1–26. New York: Springer.
35. Zacharewicz G, Frydman C and Giambiasi N. G-DEVS/HLA environment for distributed simulations of workflows. *Simulation* 2008; 84: 197–213.
36. Sievert N. *Modelica models in a distributed environment using FMI and HLA*. Master Thesis, Linköping University, Linköping, 2016.
37. Neema H, Gohl J, Lattmann Z, et al. Model-based integration platform for FMI co-simulation and heterogeneous simulations of cyber-physical systems. In: *Proceedings of the 10th international modelica conference*, Lund, 10–12 March 2014, pp. 235–245. Linköping: Linköping University Electronic Press.
38. Möller B, Karlsson M, Herzog R, et al. *Security in simulation—new authorization opportunities in HLA 4*. Linköping: Pitch Technologies, 2021.
39. Garro A and Falcone A. On the integration of HLA and FMI for supporting interoperability and reusability in distributed simulation. In: *Proceedings of the symposium on theory of modeling simulation: DEVS integrative symposium*, Alexandria, VA, 12–15 April 2015, pp. 9–16. San Diego, CA: Society for Computer Simulation International.
40. Pitch Technologies, 2021, www.pitch.se.
41. King D and Harrison HS. Open-source simulation software ''JaamSim.'' In: *Winter Simulations Conference (WSC)*, Washington, DC, 8–11 December 2013, pp. 2163–2171. New York: IEEE.
42. Amrani A. *Lean and six sigma in aeronautic industry*. Internal Report Case Study. Bordeaux: University of Bordeaux, 2017.
43. Amrani A and Ducq Y. Lean practices implementation in aerospace based on sector characteristics: methodology and case study. *Prod Plan Control* 2020; 31: 1313–1335.
44. Nambiar AN. Modern manufacturing paradigms—a comparison. In: *world congress on engineering*, London, 4–6 July 2012, pp. 1662–1667. CiteseerX Princeton, NJ, USA.
45. Benjamin SJ, Murugaiah U and Marathamuthu MS. The use of SMED to eliminate small stops in a manufacturing firm. *J Manuf Technol Manag* 2013; 24: 792–807.
46. Ghobadian A, Talavera I, Bhattacharya A, et al. Examining legitimatisation of additive manufacturing in the interplay between innovation, lean manufacturing and sustainability. *Int J Prod Econ* 2020; 219: 457–468.
47. Ho SK. 5-S practice: the first step towards total quality management. *Total Qual Manag* 1999; 10: 345–356.
48. Ho SK, Cicmil S and Fung CK. The Japanese 5-S practice and TQM training. *Train Qual* 1995; 3: 19–24.
49. Abrams C and Berge Z. Workforce cross training: a re-emerging trend in tough times. *J Workplace Learn* 2010; 22: 522–529.
50. Tanaka H. Effects of cross-training. *Sports Med* 1994; 18: 330–339.
51. Chong MY, Prakash J, Ng SL, et al. Parallel Kanban-Conwip system for batch production in electronics assembly. *Int J Ind Eng* 2018; 20: 468–486.
52. Stewart DM and Grout JR. The human side of mistake-proofing. *Prod Oper Manag* 2001; 10: 440–459.
53. Bayers P. Using Poka Yoke (mistake proofing devices) to ensure quality. In: *Proceedings of 1994 IEEE applied power electronics conference and exposition-ASPEC'94*, Orlando, FL, 13–17 February 1994, pp. 201–204. New York: IEEE.
54. Marr B. *Key Performance Indicators (KPI): The 75 measures every manager needs to know*. London: Pearson, 2012.
55. Liao C-J and Shyu C-H. An analytical determination of lead time with normal demand. *Int J Oper Prod Manag* 1991; 11: 72–78.
56. Capkun V, Hameri A-P and Weiss LA. On the relationship between inventory and financial performance in manufacturing companies. *Int J Oper Prod Manag* 2009; 29: 789–806.
57. Belyh A. Understanding work-in-progress (WIP) when analyzing financial statements, 2019. https://www.cleverism.com/work-in-progress-wip/.
58. Alden JM, Burns LD, Costy T, et al. General Motors increases its production throughput. *Interfaces* 2006; 36: 6–25.
59. Westgard J. *Defect rates, quality and productivity*, 2019, https://www.westgard.com/guest5.htm