Chapter 1

Distributed Modeling of Discrete Event Systems

Giulia Pedrielli, Tullio Tolio, Walter Terkaj and Marco Sacco

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/50350

1. Introduction

Computer simulation is widely used to support the design of any kind of complex system and to create computer-generated "virtual worlds" where humans and/or physical devices are embedded (e.g. aircraft flight simulators [20]). However, both the generation of simulation models and the execution of simulations can be time and cost expensive. While there are already several ways to increase the speed of a simulation run, the scientific challenge for the simulation of complex systems still resides in the ability to model (simulate) those systems in a parallel/distributed way [35].

A computer simulation is a computation that emulates the behavior of some real or conceptual systems over time. There are three main simulation techniques [23]:

- *Continuous simulation*. Given the discrete nature of the key parameters of a digital computer, including the number of memory locations, the data structures, and the data representation, continuous simulation may be best approximated on a digital computer through time-based discrete simulation where the time steps are sufficiently small relative to the process being modeled.
- *Time-based discrete simulation*. In this case the universal time is organized into a discrete set of monotonically increasing timesteps where the choice of the duration of the timestep interval changes as a result of the external stimuli, any change between two subsequent timesteps must occur atomically within the corresponding timestep interval. Regardless of whether its state incurs and changes, a process and all its parameters may be examined at every time step.
- *Discrete event simulation* [5]. The difference between discrete event simulation and time-based simulation is twofold. Firstly, the process being modeled is understood to advance through events under discrete event conditions. Second, an event (i.e. an activity of the process as determined by the model developer) carries with it the potential for affecting the state of the model and is not necessarily related to the

progress of time. In this case, the executable model must necessarily be run corresponding to every event to accurately reflect the reality of the process.

Since continuous simulation is simply academic and cannot be reproduced on real computers, it is important to comment the difference between time-based simulation and discrete event simulation.

Under time-based simulation, the duration of the timestep interval is determined based on the nature of the specific activity or activities of the process that the model developer considers important and worth modeling and simulating. Similarly, under discrete event simulation, events for a given process are also identified on the basis of the activity or activities the model developer views as important. Whereas time-based simulation constitutes the logical choice for processes in which the activity is distributed over every timestep, discrete event simulation is more efficient when the activity of a process being modeled is sparsely distributed over time. The overhead in discrete event simulation, arising from the additional need to detect and record the events, is higher than in the simpler time-based technique and must be more than compensated by the savings not to have to execute the model at every time step.

A fundamental difference between time-based and discrete event simulations lies in their relationship to the principle of causality. In the time-based approach, while a cause may refer to a process state at a specific timestep, the fact that the state of the process is observed at every subsequent time step reflects the assumption that the effect of the cause is expected. Thus both the cause and the effect refer to the observed state of the process in time-based simulation. In discrete event simulation, both cause and effects refer to events. However, upon execution due to an event, a model may not generate an output event thus appearing to imply that a cause will not necessary be accompanied by corresponding observed facts.

Discrete Event Simulation (DES) has been widely adopted to support system analysis, education and training, organizational change [43] in a range of diverse areas such as commerce [13], manufacturing ([14],[38], [79]), supply chains [24], health services and bio-medicine ([3], [18]), simulation in environmental and ecological systems [6], city planning and engineering [45], aerospace vehicle and air traffic simulation [40], business administration and management [16], military applications [17].

All the aforementioned areas are usually characterized by the presence of complex systems. Indeed, a system represented by a simulation model is defined as complex when it is extremely large, i.e. a large number of components characterize it, or a large number of interactions describes the relationships between objects within the system, or it is geographically dispersed. In all cases the dynamics can be hard to describe. The complexity is reflected in the system simulation model that can be characterized according to the following concepts [23]:

1. Presence of entity elements that are dynamically created and moved during a simulation [62]
2. Asynchronous behavior of the entities

3. Asynchronous interactions between the entities
4. Entities which concur for the use of shared resources
5. Connectivity between the entities

The simulation of complex systems through the use of traditional simulation tools presents several drawbacks, e.g. the long time required to develop the unique monolithic simulation model, the computational effort required for running the simulation, the impossibility to run the simulation model on a set of geographically distributed computers, the absence of fault tolerance (i.e. the work done is lost if one processor goes down), the impossibility to realize a realistic model of the entire system in the case several subsystems are included and the owners of each subsystem do not want to share the information.

Most of the aforementioned problems can be effectively addressed by the distributed simulation (DS) approach which will be the focus of this chapter.

The chapter will be organized as follows: Section 2 presents the main concepts and definitions together with a literature review on applications and open issues related to distributed simulation. Section 3 delves into the High Level Architecture [1], i.e. the reference standard supporting the distributed simulation. Section 4 shows an application of distributed simulation on a real industrial case in the manufacturing domain [77]. Finally, Section 5 presents the conclusions and the main topics for future research in the field of distributed simulation.

2. Distributed simulation

Traditional stand alone simulation is based on a simulation clock and an event list. The interaction of the event list and the simulation clock generates the sequence of the events that have to be simulated.

The execution of any event might cause an update of the value of the state variables, a modification to the event list and (or) the collection of the statistics. Each event is executed based on the simulation time assigned to it, i.e. the simulation is sequential.

The idea underlying the distributed simulation is to minimize the sequential aspect of traditional simulation. Distributed simulation can be classified into two major categories: (1) parallel and distributed computing, and (2) distributed modeling.

Parallel and distributed computing refers to technologies that enable a simulation program to be executed on a computing system containing multiple processors, such as personal computers, interconnected by a communication network [20].

The main benefits resulting from the adoption of distributed computing technologies are [20]:

- *Reduced execution time*. By decomposing a large simulation computation into many sub-computations and executing the sub-computations concurrently across different processors, one can reduce the global execution time.

- *Geographical distribution*. Executing the simulation program on a set of geographically distributed computers enables one to create virtual worlds with multiple participants that are physically located at different sites.
- *Integration* of simulators that execute on machines from different manufacturers.
- *Fault tolerance*. If one processor goes down, it may be possible for other processors to pick up the work of the failed machine allowing the simulation to proceed despite the failure.

The definition of distributed modeling can be given by highlighting the differences compared to the concept of parallel and distributed computing as presented by Fujimoto [20]. If a single simulator is developed and the simulation is executed on multiple processors we talk about *parallel* and *distributed computing*. Whereas if several simulators are combined into a distributed architecture we talk about *distributed modeling*; in this case, the simulation execution requires the synchronization between the different simulators.

The *distributed computing* can be still applied to each simulator in a distributed simulation model [60], but the complexity related to the synchronization of the different models can be such that the performance of the simulation (in terms of speed) can be worse than when a single simulation model is developed. This drawback related to the decrease in the efficiency in terms of speed of simulation leads to the following question: "Why is it useful to develop a distributed simulation model?". The following benefits represent an answer to this question ([57], [77]):

- *Complexity management*. If the complexity of the system to be simulated grows and the modeling of each sub-system requires various and specific expertise, then the realization of a single monolithic simulation model is not feasible [65]. Under the distributed modeling approach the problem is decomposed in several sub-problems easier to cope with.
- *Overcoming the lack of shared information*. The developer of a simulation model can hardly access all the information characterizing the whole system to model, again hindering the feasibility of developing a unique and monolithic simulation model.
- *Reusability*. The development of a simulation model always represents a costly activity, thus the distributed modeling can be seen as a possibility to integrate pre-existing simulators and to avoid the realization of new models.

The feasibility of the distributed simulation concept was demonstrated by the SIMNET project (SIMulator NETworking [73]), which ran from 1983 to 1990. As consequence of this project, a set of protocols were developed for interconnecting simulations and the Distributed Interactive Simulation (DIS) standard was the first one. Afterwards, the High Level Architecture (HLA) standard ([1], [15], [27]) was developed by the U.S. Department of Defense (DoD) under the leadership of the Defense Modeling and Simulation Office (DMSO). The next sub-section presents a general overview of the HLA standard for distributed simulation, whereas Section 2.2 gives an overview of distributed simulation in civilian applications.

2.1. HLA-standard: An overview

HLA (IEEE standard 1516) is a software architecture designed to promote the use and interoperation of simulators. HLA was based on the premise that no single simulator could satisfy all uses and applications in the defense industry and it aimed at reducing the time and cost required to create a synthetic environment for a new purpose.

The HLA architecture (Figure1) defines a *Federation* as a collection of interacting simulators (*federates*), whose communication is orchestrated by a *Runtime Infrastructure* (RTI) and an interface. Federates can be either simulations, surrogates for live players, or tools for distributed simulation. They are defined as having a single point of attachment to the RTI and might consist of several processes, perhaps running on several computers.

HLA can combine the following types of simulators (following the taxonomy developed by the DoD):

- *Live* - real people operating real systems (e.g. a field test)
- *Virtual* - real people operating simulated systems (e.g. flight simulations)
- *Constructive* - simulated people operating simulated systems (e.g. a discrete event simulation)

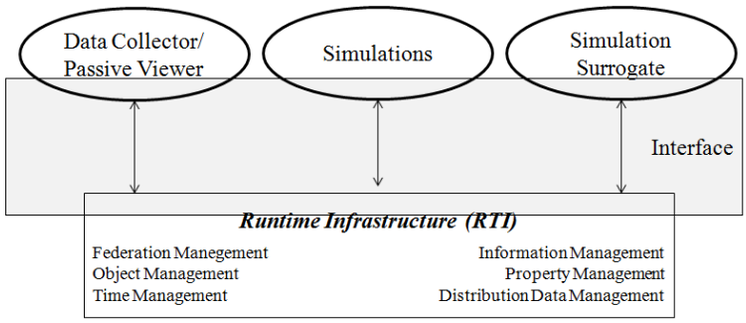

Figure 1. HLA Reference Architecture

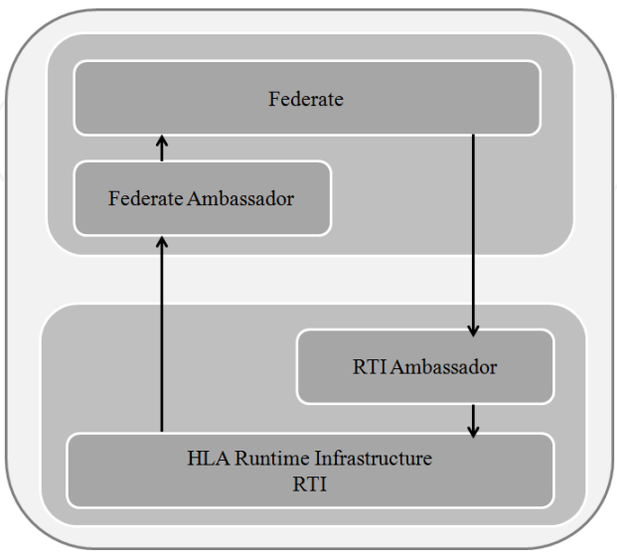

Figure 2. RTIAmbassador and FederateAmbassador

The HLA standard provides four main components for the realization and management of a federation:

- HLA rules (IEEE 1516.0, 2000) representing a set of 10 rules that the simulators (federates) have to follow in order to be defined HLA-compliant.
- Federate Interface Specification (FIS) (IEEE 1516.2, 2000) defining how simulators are supposed to interact with the RTI.
- Object Model Template (OMT) (IEEE 1516.1, 2000) specifying what kind of information is communicated between simulators and how simulations are documented. Following the OMT each federate defines the data that it is willing to share (publish) with other federates and the data it requires from other federates (subscribe). The resulting object models related to each federate are called simulation object models (SOMs). The federation object model (FOM) combines the federate SOMs into a single object model for the federation to define the overall data to be exchanged within the federation.
- Federate Development Process (FEDEP) (IEEE 1516.3, 2004) defining the recommended practice processes and procedures that should be followed by users of the HLA to develop and execute their federations.

The federates cannot directly exchange information throughout the federation, instead the RTI plays the role of the operating system of the distributed simulation, providing a set of general-purpose services for federation management and enabling the federates in carrying out federate-to-federate *interactions*. In particular interactions represent an explicit action taken by a federate that may have some effect on another federate within a federation execution, such action can be tied with a specific time defined as *interactionTime*, when the action takes place.

Each federate is endowed with an *RTIAmbassador* and a *FederateAmbassador* (Figure 2) to access the services offered by the RTI. Operations on the *RTIAmbassador* are called by the federate whenever it needs an RTI service (e.g. a request to advance simulation time). In the reverse direction, the RTI invokes an operation on the *FederateAmbassador* whenever it needs to pass data to the federate (e.g. to inform the federate that the request to advance simulation time has been granted). Six classes of services (Figure 1) have to be provided by the RTI to be defined HLA-compliant. These classes are specified within the FIS and they can be summarized as follows:

- Federation Management. These services allow federates to create and destroy federation execution and join or resign from an existing federation.
- Declaration Management. These services allow federates to publish federate data and subscribe to updated data produced by other federates.
- Object Management. These services allow federate to create and delete object instances, and produce and receive data.
- Ownership Management. These services allow federates to transfer the ownership of object data during the federation execution.
- Time Management. These services coordinate the advancement of simulation time of the federates.

- Data Distribution Management. These services can reduce unnecessary information transfer between federates by filtering out irrelevant data.

HLA overcame the shortcomings of the DIS standard by being simulation-domain neutral (it was not developed referred to any specific language, therefore HLA provides means to describe any data exchange format as required and specifying functionalities for time management and bandwidth control (see the FIS module).

HLA provides Application Programming Interfaces (APIs) for all the classes of services just mentioned, but the RTI software and algorithms are not defined by HLA. Also the operations in the *FederateAmbassador* need to be implemented at the federate level, as part of the federate code or some interface service (*adapter*).

These facts have caused the growth of multiple HLA-RTI implementations (e.g. [80], [81]) and the development of *ad-hoc* solutions for the adapters on the federate side [25]. In particular the last aspect represents one of the most relevant criticalities in applying HLA for distributed simulation: the lack of a standardized approach to adapt a simulator within an HLA-based distributed architecture, makes a distributed simulation project time expensive since a lot of implementation is required in addition to the effort to build the simulation model.

This consideration represents one of the leading arguments for the research community in the direction of the development of additional complementary standards (Section 3) to ease the creation and management of an HLA-based distributed simulation.

It is the objective of the next section to analyze the state of the art on the adoption and advancements in the use of HLA-based distributed simulation technique.

2.2. Distributed simulation in civilian applications

Herein the attention is focused on distributed modeling of complex systems in civilian domain.

HLA constitutes an enabler for implementing the distributed simulation. The standard, though, was conceived for military applications and several problems arise when trying to interoperate heterogeneous simulators in civilian applications (the terminology Commercial off-the-shelf discrete-event simulation packages CSPs [62] will be used to describe commercially available simulators for the analysis of Discrete Event Systems).

Boer [12] investigated the main benefits and criticalities related to the industrial application of HLA by interviewing the actors involved in the problem (e.g. simulation model developers, software houses, HLA experts, [9]-[11]). The results of the survey showed that CSPs vendors do not see direct benefits in using distributed simulation, whereas in industry HLA is considered troublesome because of the lack of experienced users and the complexity of the standard. In addition, as suggested in [49], although the approaches and general methods used in military and civilian simulation communities have similarities, the

terminology turns out to be completely different [36]. For instance, terms like live simulation and virtual emulator are rarely used in civilian applications although equivalent techniques are commonly applied.

The major difference between military and civilian domain resides in the way simulation models are developed and what are the goals to meet when starting a simulation development process. In the military community where time and budget constraints are not the key elements leading the building process of a simulation tool, languages such as C++ and Java are usually adopted because of their flexibility. On the other hand, in the civilian simulation community, the use of commercial simulation tools (e.g. Arena, Automod, Simio, ProModel, Simple++, SLX, etc.) is the common practice. These tools satisfy the need of rapidly and cost-effectively developing the simulation models.

The use of commercial simulation tools hinders the applicability of the HLA standard for the realization of a distributed simulation model, because the direct access to the HLA APIs (Section 2.1.) from the commercial simulation software tools is not usually possible. Therefore, the enhancement of HLA with additional complementary standards [51] and the definition of a standard language for CSPs represent relevant and not yet solved technical and scientific challenges ([25], [49], [50]). Recently, the COTS Simulation Package Interoperability-Product Development Group (CSPI-PDG), within the Simulation Interoperability Standards Organization (SISO), worked on the definition of the CSP interoperability problem (Interoperability Reference Models, IRMs) [74] and on a draft proposal for a standard to support the CSPs interoperability (Entity Transfer Specification, ETS) [61].

2.2.1. Literature review

The application of distributed simulation in the civilian domain has been studied by reviewing the available literature with the purpose to individuate which civilian domain distributed simulation is generally called, which motivations underlie the adoption of the distributed technique, which technical and scientific challenges have been faced and which solutions have been proposed so far. More than 100 papers have been analyzed and classified according to three criteria:

- *Domain of application*, i.e. the specific civilian sector where the distributed simulation was applied (e.g. manufacturing domain, health care, emergency, etc.).
- *Motivation* underlying the adoption of the distributed simulation, i.e. the main problem leading to the adoption of the distributed simulation architecture.
- *Technical issue* faced, i.e. the solutions to integration issue or enhancement to services of the HLA architecture proposed within the considered article.

Most of the articles can be classified according to more than one criterion and Figure 3 shows the percentage of articles falling in each category.

The bibliographic search was carried out by considering the following keywords: Distributed Simulation, Operations Research and Management, Commercial Simulation Packages, Interoperability Reference Models, High Level Architecture, Manufacturing

Systems, Discrete Event Simulation, Manufacturing Applications, Industrial Application and Civilian Applications. These keywords brought to the identification of 26 core papers based on the number of citations ([4], [12], [8], [11], [9], [10], [20], [28], [29], [30], [33], [74], [75] , [48], [50], [47], [49], [58], [53], [59], [56], [68], [70], [68], [73] and [71]). These papers can be considered as introductory to the topic of distributed simulation in civilian domain. Starting from these articles the bibliographic search followed the path of the citations, i.e. works cited by the core papers and papers citing the core ones were considered. This search brought to the selection of 83 further articles. The overall 109 papers were published mainly in the following journals and conference proceedings: Advanced Simulation Technologies Conference, European Simulation Interoperability Workshop, European Simulation Symposium, Information Sciences, International Journal of Production Research, Journal of the Operational Society, Journal of Simulation, Workshop on Principles of Advanced and Distributed Simulation and Winter Simulation Conference.

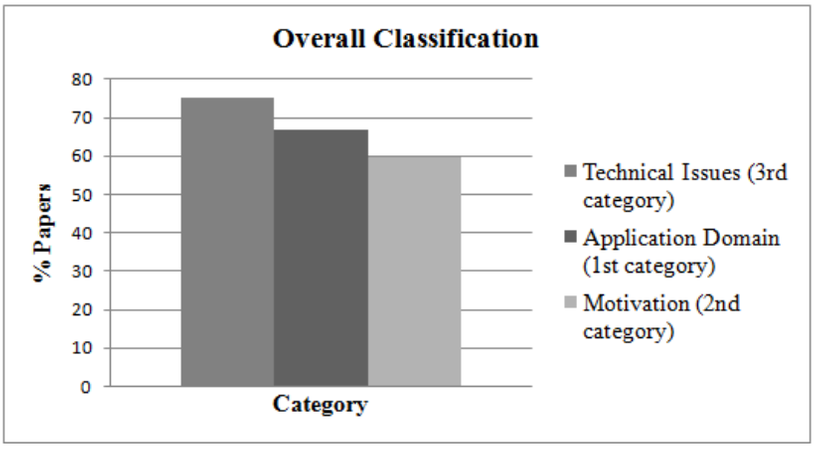

Figure 3. Overall Classification Criteria

2.2.2. Domain of application

More than 60% (Figure 3) of the analyzed papers propose an application in a specific field of the civilian domain (e.g. [72], [42]). As stated in [46], transportation and logistics are typical application areas of simulation and also the first areas where HLA has been tested by the civilian simulation community. Manufacturing and health care are acquiring increasing importance because of the growth of the extended enterprise and the increase in attention for bio-pharmaceutical supply chains respectively.

The main fields of application of DS (Figure 4) are Supply Chain Management (33% of the papers stating the domain of interest) (e.g. [64], [22], [42]), Manufacturing (29% of the papers) (e.g. [69], [77]), Health Care (e.g. [34]) and Production Scheduling & Maintenance (e.g. [72]), 17% of the articles are related to Health Care.

A further analysis was carried out by considering only the articles related to the manufacturing domain, aiming at evaluating whether the contributions addressed real industrial case applications or test cases applications. Only 22% of the articles address a real case, thus confirming the outcomes obtained by Boer [8] in the analysis of the adoption of

distributed simulation in the manufacturing environment. Although solutions have been developed for the manufacturing domain, this technique is still far from being adopted as an evaluation tool by industrial companies because the end-users perceive HLA and distributed simulation as an additional trouble rather than a promising approach [10]. As a consequence, a lot of effort is put in the development of decision support systems that hide the complexity of a distributed environment to the end user [41].

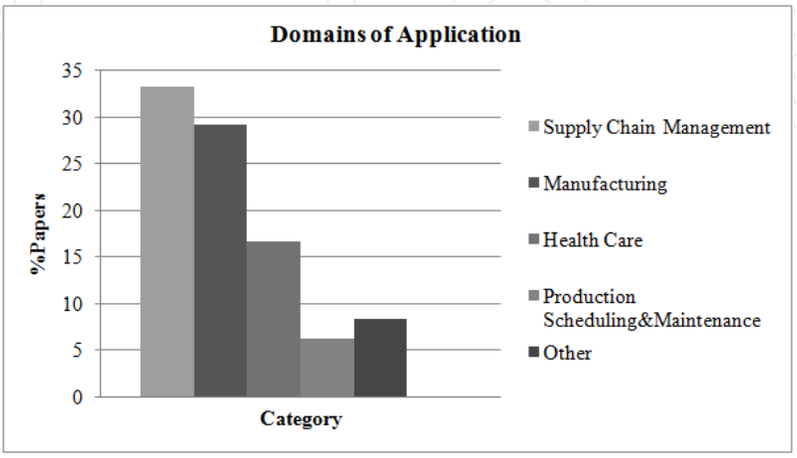

Figure 4. Distributed Simulation Main fields of application

2.2.3. Motivations underlying the adoption of the distributed simulation

As Boer stated in [8], if a problem can be solved by a monolithic simulation model created in a single COTS simulation package and a distributed approach is not explicitly required the simulation practitioner should certainly choose the monolithic solution in the selected CSP. Similarly, Strassburger [49] suggests that if a maintainable and reusable monolithic application can be built, then there is no point in building it in a distributed platform.

However, there are simulation projects where the distributed solution seems more advantageous and straightforward [38] because it enables to cope with:

1. Demand for *reusability* of the simulator output of the simulation project. Here the word reusability is adopted both in terms of the possibility to reuse simulators already developed and of building new simulators that can be readopted in the future.
2. *Lack of Shared information*. This is the case when no one modeler has enough information to develop the simulator. This condition holds when the whole system to be modeled is divided into subsystems owned by different actors that do not want to share data related to their subsystems.
3. *System complexity*. In this case no one modeler has enough knowledge to realize the whole simulation model.

All the papers stating a motivation for using DS mention the system complexity (e.g. [22], [72], [30], [32]), whereas 44% of the papers the demand for reuse [78]. The low percentage (around 5%) of papers using DS to cope with lack of shared information can be partially traced back to the lack of real industrial applications that still characterizes DS in civilian environment [76].

2.2.4. Technical issue faced

Over 70% of the articles deal with technical issues, thus showing that HLA and DS experts are putting a lot of effort in the enhancement and extensions of HLA-based DS to face civilian application problems. Indeed, the application of distributed simulation to civilian domain still presents several technical issues. In particular four main research areas can be identified:

1. *Integration of commercial discrete event simulators (CSP)*. Several CSPs are put together and synchronized by means of the services offered by the HLA infrastructure.
2. *Interoperability reference models and entity transfer*. The papers in this category work in the standardization of the communication between federates within an HLA-compliant federation (Section 3.).
3. *Time management enhancement*. The issues related to the time synchronization of federates are faced.
4. *RTI-services extension*. In this case the services listed in Section 2.1. are enhanced for specific applications [82].

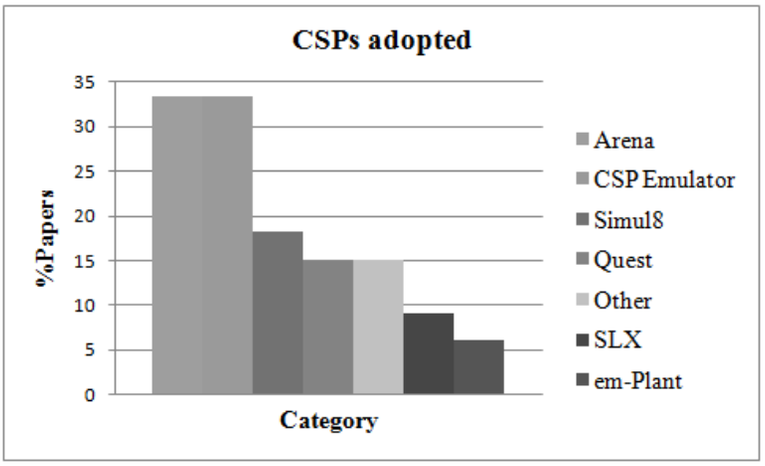

Figure 5. CSP adopted

The outcome of the review was that the integration of CSPs is the most addressed technical issue, (45% of the papers) nonetheless the integration of real CSPs (i.e. not general purpose programming languages) still represents a challenging topic. Figure 5 gives a picture of the main CSP solutions that have been adopted in the literature. In particular, the *y-axis* reports the percentage of articles that use one of the listed CSPs (e.g. [37], [21], [67]) within the papers that deal with the interoperation of simulators. It can be noticed that CSP Emulators (e.g. [68], [38]) are still one of the most used solutions because of the problems related to interoperating real CSPs. These problems are mainly caused by the lack of data and information mapping between simulators and the difficulty in interacting (e.g. send and receive information, share the internal event list) while the simulation is running.

The enhancement of the Run Time Infrastructure services is another key research topic ([2], [19]). In particular, the scientific articles deal with two open issues: (1) Time management (e.g. [39], [31]), (2) Data Distribution Management (e.g. [66], [73]). Time Management has

received more attention (91% of the papers dealing with enhancement of RTI services) because it strongly influences the computational performance of the distributed simulation.

The following conclusions can be drawn from the literature analysis:

- There is a lack of distributed simulation applications in real manufacturing environments.
- The interoperability of CSPs still represents a technical challenging problem.
- The HLA architecture components (in particular RTI services) must be extended and adapted to civilian applications.

The issues faced to model complex systems give raise to problems in the distributed simulation realization that are strongly dependent on the specific application case [57] and the solutions to those needs can be implemented through an RTI in many different (incompatible) ways. Each way can be promising in its own context, but the lack of a standardized approach means it is difficult for end users and CSP vendors to choose a solution thus slowing down the spreading of the distributed simulation technique.

3. A standard based approach for distributed simulation

The main contribution in standardization of distributed modeling has to be credited to Simulation Interoperability Standard Organization (SISO) and in particular to the High Level Architecture Simulation Package Interoperability Forum (HLA-CSPIF). HLA-CSPIF and, then, COTS Simulation Package Interoperability Product Development Group (CSPI-PDG) were created in an attempt to produce a generalizable solution to the problem of integrating distributed heterogeneous CSPs.

As highlighted at the end of Section 2.2., a standardized approach is fundamental to increase the use of distributed simulation in civilian applications. This led to formalize the problem of the interaction between simulators in civilian applications and to standardize the way data are exchanged between federates within the federation.

The main results of the standardization effort are the Interoperability Reference Models (IRMs) and the Entity Transfer Specification (ETS), that will be presented in Section 3.1 and 3.2, respectively. Section 3.3 shows the distributed simulation communication protocol presented in [77], based on IRMs and the extended ETS proposed in [38].

3.1. Interoperability reference model

An interoperability problem type is meant to capture a general class of interoperability problems, whereas an IRM is meant to capture a specific problem within that class.

The creation of the IRMs has proved to be a powerful tool in the development of standards in the distributed simulation research community, as it is now possible to create solutions for specific integration problems.

An initial set of interoperability problems identified by the CSPI-PDG have been divided into a series of problem types that are represented by IRMs. The purpose of an IRM can be to [54]:

- Clearly identify the CSP/model interoperability capabilities of an existing distributed simulation.
- Clearly specify the CSP/model interoperability requirements of a proposed distributed simulation.

There are four types of IRM:

- Type A - Entity Transfer (Section 3.1.1.)
- Type B - Shared Resource (Section 3.1.2.)
- Type C - Shared Events (Section 3.1.3.)
- Type D - Shared Data Structure (Section 3.1.4)

The literature review showed that around 21% of the articles dealing with technical issues (Section 2.2.1.) taken into consideration deal with IRMs (e.g., [39], [56], [51], [63], [55] and [34]).

3.1.1. IRm type A: Entity transfer

IRM Type A Entity Transfer represents interoperability problems that can occur when transferring an entity from one simulation model to another. This IRM type is the most formalized at the present moment, since the need to transfer entities between simulators has been the most popular feature requested from the distributed simulation users so far.

Figure 6 shows an illustrative example of the problem of Entity Transfer where an entity *e1* leaves activity A1 in model M1 at time T1 and arrives at queue Q2 in model M2 at time T2. For example, if M1 is a car production line and M2 is a paint shop, then the entity transfer happens when a car leaves M1 at T1 and arrives in a buffer in M2 at T2 to wait for painting.

There are three subtypes of IRM Type A:

- IRM Type A.1 General Entity Transfer is defined as the transfer of entities from one model to another such that an entity *e1* leaves model M1 at T1 from a given place and arrives at model M2 at T2 at a given place and $T1 \leq T2$ or $T1 < T2$. The place of departure and arrival will be a queue, workstation, etc.
- IRM Type A.2 Bounded Receiving Element. The IRM Type A.2 is defined as the relationship between an activity A in a model M1 and a bounded queue Q2 in a model M2 such that if an entity e is ready to leave activity A at T1 and attempts to arrive at the bounded queue Q2 at T2 then:
  - If the bounded queue Q2 is empty, the entity e can leave activity A at T1 and arrive at Q2 at T2
  - If the bounded queue Q2 is full, the entity e cannot leave activity A at T1; activity A may then block if appropriate and must not accept any more entities.
  - When the bounded queue Q2 becomes not full at T3, entity e must leave A at T3 and arrive at Q2 at T4, activity A becomes unblocked and may receive new entities at T3.

- T1 ≤ T2 and T3 ≤ T4.
- If activity A is blocked then the simulation of model M1 must continue.

- IRM Type A.3 Multiple Input Prioritization. As shown in Figure 7, the IRM Type A.3 Multiple Input Prioritization represents the case where a model element such as queue Q1 (or workstation) can receive entities from multiple places. Let us assume that there are two models M2 and M3 which are capable of sending entities to Q1 and that Q1 has a First-In-First-Out (FIFO) queuing discipline. If an entity *e1* is sent from M2 at T1 and arrives at Q1 at T2 and an entity *e2* is sent from M3 at T3 and arrives at Q1 at T4, then if T2 < T4 we would expect the order of entities in Q1 would be e1, e2. A problem arises when both entities arrive at the same time, i.e. when T2 = T4. Depending on implementation, the order of entities would either be *e1*, *e2* or *e2*, *e1*. In some modeling situations it is possible to specify the priority order if such a conflict arises, e.g. it can be specified that model M1 entities will always have a higher priority than model M2 (and therefore require the entity order *e1*, *e2* if T2 = T4). Furthermore, it is possible that this priority ordering is dynamic or specialized.

Note that the IRM sub-types are intended to be cumulative, i.e. a distributed simulation that correctly transfers entities from one model to a bounded buffer in another model should be compliant with both IRM Type A.1 General Entity Transfer and IRM Type A.2 Bounded Receiving Element.

The largest part of the papers analyzed in the literature review deal with the basic IRM that is the general entity transfer (85% of the papers dealing with IRMs), since most of the applications are related to Supply Chain Management and queues can often be modeled as infinite capacity as they represent inventory, production or distribution centers.

The situation is slightly different if the manufacturing domain is considered. Indeed, IRM Type A.2 (70% of the articles dealing with IRMs), is largely adopted when a production system is modeled, because the decoupling buffers between workstations must be usually represented as finite capacity queues.

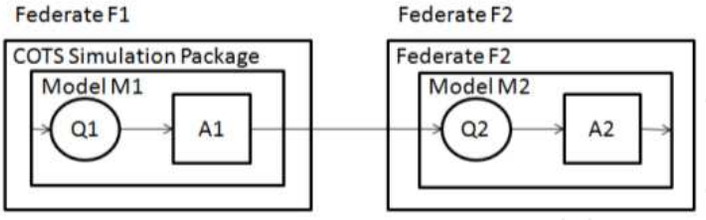

Figure 6. Typr A.1 IRM

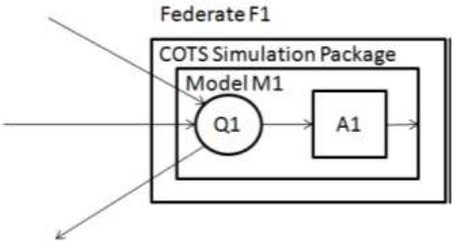

Figure 7. Type A.3 IRM

3.1.2. IRM type B: Shared resource

IRM Type B deals with the problem of sharing resources across two or more models in a distributed simulation. A modeler can specify if an activity requires a resource (such as machine operators, conveyor, pallets, etc.) of a particular type to begin. If an activity does require a resource, when an entity is ready to start that activity, it must therefore be determined if there is a resource available. If it is available then the resource is secured by the activity and held until the activity ends. A resource shared by two or more models leads to a problem of maintaining the consistency change the deleted part with: related to the status of the resource..

Currently there is only one IRM Type B subtype. The IRM Type B.1 General Shared Resources is defined as the maintenance of consistency of all copies of a shared resource such that:

- if a model M1 wishes to change its copy of the resource at time T1 then the value of all other copies of the same resource present in other models will be guaranteed to be the same at T1.
- if two or more models wish to change their copies of the resource at the same time T1, then all copies will be guaranteed to be the same at T1.

3.1.3. Type C: Shared event

IRM Type C deals with the problem of sharing events (such as an emergency signal, explosion, etc.) across two or more models in a distributed simulation.

There is currently one IRM Type C sub-type. The IRM Type C.1 General Shared Event is defined as the guaranteed execution of all local copies of a shared event E such that:

- if a model M1 wishes to schedule a shared event E at T1, then its local copies (i.e. the events scheduled within each simulator) will be guaranteed to be executed at the same time
- if two or more models wish to schedule shared events E1, E2, etc. at T1, then all local copies of all shared events will be guaranteed to be executed at the same time T1.

3.1.4. Type D: Shared data structures

IRM Type D Shared Data Structure deals with the sharing of variables and data structures across simulation models that are semantically different to resources (e.g. while referring to the supply chain environment, this is the case with bill of material information management or shared inventory). There is currently one IRM Type D sub-type.

3.2. Entity transfer specification

CSPI-PDG proposed the Entity Transfer Specification (ETS) Protocol ([51], [52], [62]) which refers to the architecture shown in Figure 8.

Each federate consists of a COTS simulation package (CSP), a model that is executed by the CSP, and the middleware that is a sort of adaptor interfacing the CSP with the Run Time

Infrastructure (RTI) (Figure 8). The relationship between CSP, the middleware and the RTI consists of two communication flows: (1) middleware-RTI, (2) CSP-middleware. The middleware translates the simulation information into a common format so that the RTI can share it with the federation. In addition, the middleware receives and sends information from/to the CSP. The CSP communicates with its middleware by means of Simulation Messages (Section 3.3.1.) [77]. The presence of Simulation Messages is the main difference between the reference architecture in Figure 8 and the architecture proposed by Taylor et al [51].

ETS defines the communication between the sending model and the receiving model (*ModelA* and *ModelB* in Figure 8, respectively) at RTI level. In particular the way the middleware of each federate and the RTI exchange information is formalized by means of a special hierarchy of interaction classes. An interaction class is defined as a template for a set of characteristics (parameters) that are shared by a group of interactions (refer to IEEE HLA standard, 2000). The middleware of the sending model instantiates a specific interaction class and sends it to the RTI whenever an entity has to be transferred.

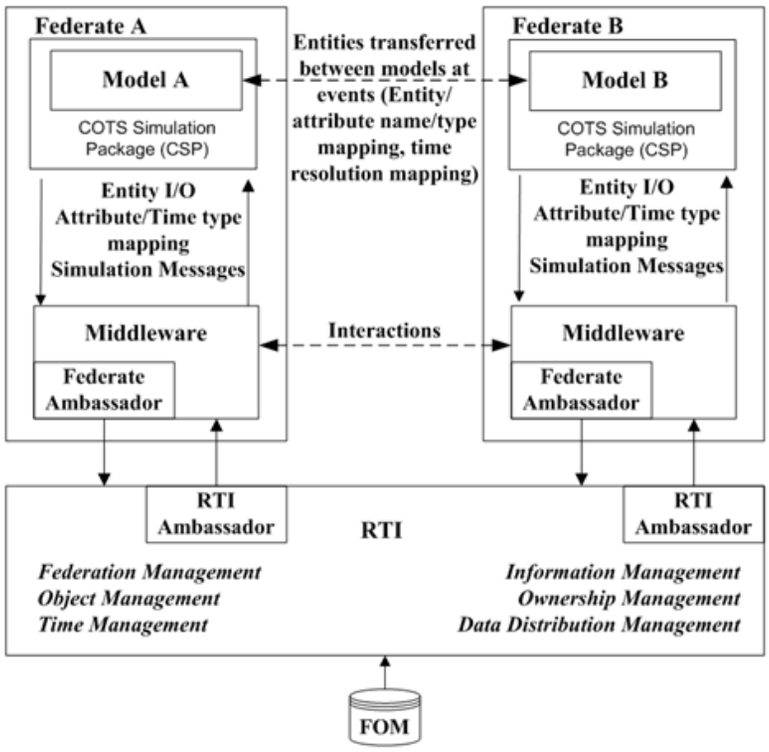

Figure 8. ETS refrence Architecture

Two main issues arise when the simulation information is translated for the RTI:

- A common time definition and resolution is necessary. For example, the time should be defined as being the time when an entity exits a source model and then instantaneously arrives at the destination model (i.e. the definition of time implies zero transit time) [62]. Alternatively, it should be defined including the notion of travel time and the entity would arrive at destination with a delay equal to the transfer time.

- The representation of an entity depends on how the simulation model is designed and implemented in a CSP. Indeed, the names that the modelers use to represent the same entity might be different. A similar problem can arise for the definition of simple datatypes. For example, some CSPs use 32-bit real numbers while others use 64-bit [62].

Straburger [50] highlighted some relevant drawbacks in the ETS standard proposal:

- It is not possible to differentiate multiple connections between any two models.
- ETS suggested interaction hierarchy does not work: a federate subscribing to the superclass will never receive the values transmitted in the interaction parameters.
- The specification of user defined attributes is placed into a complex datatype, this introduces new room for interoperability challenges as all participating federates have to be able to interpret all of the attributes.
- There are some possibilities for misinterpretation in the definition of Entity and EntityType introducing changes in FOMs whenever a new entity type is talked about.

Furthermore, the ETS was not designed to manage the Type A.2. IRM and the interaction class hierarchy refers to the entity transfer without taking into account any information on the state of the receiving buffer (e.g. Q2 in Figure 6).

One of the most recent contributions in ETS was presented by Pedrielli et al. [77] and consists in the proposal of a new class hierarchy. In particular, different subclasses of the transferEntity class were defined to enable the differentiation of multiple connections between models and the Type A.2. IRM management. After developing the interaction class hierarchy, following the HLA standard, the Simulation Object Model (SOM) and Federation Object Model (FOM) were developed to include the novel interactions and their parameters. In particular, extensions were proposed to the Interaction Class Table (part of the OMT, Section 2.1) to include the novel interaction classes and define them as publish and/or subscribe. The Parameter Table (part of the OMT, Section 2.1) was modified to include the proposed parameters for the interactions and the Datatype table was also modified.

The resulting class hierarchy consists of the following classes [38]:

- *transferEntity*, as already defined in the ETS protocol. This superclass allows the federate subscribing to all the instances of entity transfer. The instantiation of this class is related to visualization and monitoring tasks.
- *transferEntityFromFedSourceEx* is a novel subclass defined for every exit point, where FedSourceEx stands for the name or abbreviation of a specific exit point in the sending model. This class is useful to group the instances of the transferEntity that are related to the source federate, so that the FedSourceEx can subscribe to all these instances without explicitly naming them.
- *transferEntityFromFedSourceExToFedDestEn* is a novel subclass defined for each pair of exit point (Ex) of the source federate (FedSource) and entry point (En) of the receiving federate (FedDest). This class is instantiated both when a sending model needs to transfer a part to a specific entry point in the receiving model, and when a receiving model needs to share information about a buffer state or about the receipt of a part from

a specific exit point in a sending model. The models both publish and subscribe to this subclass that was designed to create a private communication channel between the sending and the receiving model. Therefore, if an entry point in the receiving model is connected with multiple federates/exit points, then the receiving federate has to inform about the state of the entry point by means of multiple interactions, each dedicated to a specific federate/exit point. This communication strategy is not the most efficient in a generic case, but it offers the possibility to deliver customized information and adopt different priorities for the various federates/exit points. This becomes fundamental in real industrial applications where information sharing among different subsystems is seen as a threat, thus rising the need to design a protocol that creates a one to one communication between each pair of exit/entry point inside the corresponding sending/receiving model.

The ETS Interaction class table was modified to represent the *transferEntityFromFedSourceEx* and *transferEntityFromFedSourceExToFedDestEn* subclasses. The Parameter Table was modified to include the parameters of the novel interaction class *transferEntity FromFedSourceExToFedDestEn*. The introduced parameters are presented below. The similarities with the parameters included in the ETS Parameter Table are highlighted where present.

- *Entity*. It is a parameter of complex datatype containing the *EntityName* that is used to communicate the type of the entity, and the *EntityNumber* that is used to communicate the number of entities to be transferred. The *EntityName* and *EntityNumber* play the role of the *EntityName* and *EntitySize* defined in ETS, respectively [51], [62].
- *ReceivedEntity*. It refers to the entity received by the receiving federate and has the same type of the parameter Entity.
- *Buffer_Availability*. It was designed to enable the communication about the buffer availability.
- *SourcePriority*. This parameter was designed to communicate the priority assigned to the entity source, so that the infrastructure can be further extended to manage Type A.3 IRM (Section 3.1)
- *EntityTransferTime*. It defines the simulation time when the entity is transferred to the destination point, i.e. the arrival time. Herein the entity leaves the source node and reaches the destination node at the same time, since it is assumed that the transferred entity instantaneously arrives at destination.

The resulting tables are shown in Tables 1, 2 and 3.

HLAinteractionRoot(N)	TransferEntity (N/S)	TransferEntityFrom FromFedSourceExA (N/S)	TransferEntity FromFedSourceExAToFedDestEnC(PS)
			TransferEntity FromFedSourceExBToFedDestEnC(PS)

Table 1. Table 1. Interaction Class Table

Interaction	Parameter	DataType	Transportation	Order
TransferEntityFromFedSourceExAtoFedDestEnC (P/S)	Entity	EntityType	HLAreliable	TimeStamp
	ReceivedEntity	EntityType	HLAreliable	TimeStamp
	Buffer_Availability	HLAInteger32BE	HLAreliable	TimeStamp
	SourcePriority	HLAInteger32BE	HLAreliable	TimeStamp
	EntityTransferTime	HLAFloat32BE	HLAreliable	TimeStamp

Table 2. Parameter Table

Record Name	Field		Encoding
	Name	Type	HLAfixedRecord
EntityType	EntityName	HLAASCIIString	
	EntityNumber	HLAInteger32BE	

Table 3. Fixed Record Datatype table

3.3. Communication within the HLA-based integration infrastructure

Pedrielli et al. [77] proposed a communication protocol (see Section 3.3.2) based on messages to manage the communication between a CSP and its middleware (or adapter). The communication protocol was conceived for the distributed simulation of network of Discrete Event Manufacturing Systems characterized by the transfer of parts in the presence of buffers with finite capacity, with the objective to minimize the use of zero-lookahead [62] for the synchronization of federates.

Before illustrating the communication protocol, Section 3.3.1. presents the concept and functioning of Simulation Messages created to support the communication between a CSP and the middleware. The communication protocol between federates is then explained in Section 3.3.2., whereas Section 3.3.3. defines the hypotheses needed to minimize the zero lookahead when applying the proposed protocol.

3.3.1. Simulation messages

The function of the simulation messages depends on the role played by the federate. The sending federate uses the message for communications concerning the need of sending an entity to another model (outgoing communication) and/or information on the availability of the target receiving federate (incoming communication). The receiving federate uses the message for communications concerning the buffer state and/or the acceptance of an entity (outgoing communication) and/or the receipt of an entity from other models (incoming communication). Simulation Messages are implemented as a class that is characterized by the following attributes:

- time referring to the simulation time when the message is sent to the middleware from the CSP. This attribute is used by the middleware to determine the TimeStamp of the interaction that will be sent to the RTI.

- BoundedBuffer containing the information about the availability of the bounded buffer in the receiving model.
- TransferEntityEvent representing the entity transfer event scheduled in the sending model event list and contains the information about the entity to be transferred and the scheduled time for the event.
- ExternalArrivalEvent representing the external arrival event that is scheduled in the receiving model. It contains the information about the entity to be received and the scheduled time for the event.
- ReceivedEntity representing the information about the entity that was eventually accepted by the receiving model.

3.3.2. Communication protocol

Herein, the behavior of the sending federate will be analyzed at first, then the receiving federate will be taken under consideration. Finally an example will be described to clarify how the protocol works.

Sending Federate. The CSP of the sending federate sends a message to its middleware whenever a *TransferEntityEvent* is scheduled, i.e. the departure event of an entity from the last workstation of the sending model is added to the simulation event list. Then, the middleware uses the attributes time and *TransferEntityEvent* to inform the RTI about the need of passing an entity, while the simulation keeps on running (the *TransferEntityEvent* time corresponds to the *EntityTransferTime* presented in Section 3.2.).

The request to advance to *EntityTransferTime* is sent by the middleware to the RTI as soon as all local events scheduled for that time instant have been simulated.

After the time has advanced, the middleware can inform the CSP of the sending model about the state of the receiving buffer in the receiving model. If the receiving buffer is not full, then the workstation can simulate the *TransferEntityEvent*, otherwise it becomes blocked. From the blocking instant until when the middleware informs the sending model that the receiving buffer is not full, the model keeps on sending requests for time advance at the lookahead value.

Receiving Federate. The CSP of the receiving federate sends a message to its middleware whenever a change in the buffer availability occurs. This message contains the updated value of the attribute *boundedBuffer* representing the availability of the buffer, i.e. the number of available slots. Then, the middleware communicates this information to the RTI via interactions. In particular the information on the availability of the buffer represents a field of the timestamped interaction *transferEntityFromFedSourceExToFedDestEn* (Section 3.2.).

If the change in the buffer availability is due to the arrival of an entity from another model, then the update of the information does not imply zero lookahead and the communication is characterized by defining the entity that has been accepted (i.e. the *ReceivedEntity* attribute). If the buffer state change is not related to an external arrival, then the update of the buffer information may imply a zero lookahead whenever it is not possible to determine an advisable a-priori lookahead for the federation (Section 3.3.3) [62]. After being informed by

the middleware that another federate needs to transfer an entity, the receiving model actually simulates the arrival of the entity only if the buffer is not full, otherwise the arrival is not simulated and the workstation in the sending model becomes blocked.

Example. The application of the Simulation Messages can be better appreciated by presenting an example (see Figure 9) that is characterized as follows: (1) the reference production system is represented in Figure 10, (2) the buffer Q2a at time t accommodates a number of parts that is greater than zero and less than the buffer capacity and an entity enters workstation W1a, (3) a departure event from workstation W1a is scheduled for time t' = t + p, where p represents the processing time of the leaving entity at station W1a, (4) during the time interval (t; t'), no event happening in the federate M2 (local event) influences the state of the buffer Q2a. Since W1a is the last machine in model M1, the departure event is also a *TransferEntityEvent*. Therefore, the CSP sends a message to its middleware containing time (t) and the *TransferEntityEvent* attributes. After receiving the message, the middleware of the sending model informs the RTI via interaction.

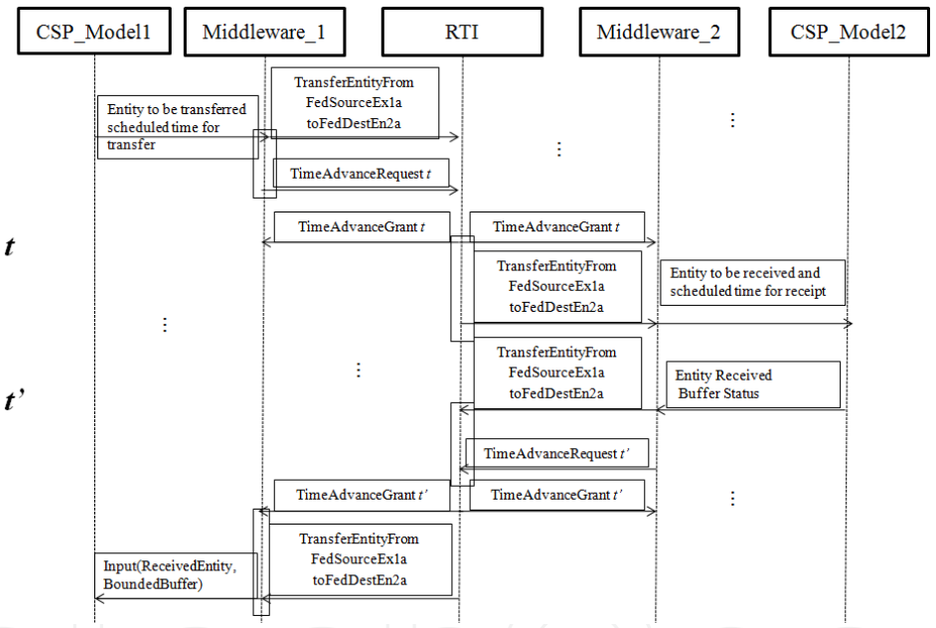

Figure 9. Communication Protocol

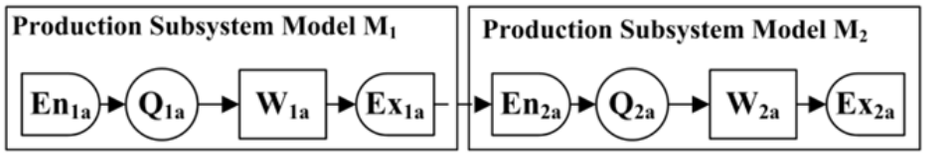

Figure 10. Reference Production System

Once the RTI time advances to time t, the middleware of the receiving model receives the information about the need of the sending model to transfer an entity at time t'. Then, the middleware sends to the receiving model a simulation message containing the *ExternalArrivalEvent*. The receiving model simulates the external arrival as soon as the simulation time advances to t' and all local events for that time have been simulated (since

the buffer Q2a is not full according to the example settings). A message is sent to the middleware of the receiving model containing the updated level of Q2a (attribute *BoundedBuffer*) together with the information concerning the recently accommodated part (attribute *ReceivedEntity*).

Afterwards, the middleware sends two interactions to the RTI: one is with a TimeStamp equal to t' and contains the updated state of the buffer Q2a and the receipt of the entity, the other contains the request of time advance to time t'. Once the RTI reaches time t', the middleware of the sending model receives the information regarding the state of Q2a and the received entity by means of the RTI. Since the entity has been delivered to the receiving model, the station W1a is not blocked by the middleware.

3.3.3. Formal characterization of the communication protocol

This section defines which hypotheses are needed to minimize the occurrence of zero lookahead if the communication protocol afore presented is adopted.

Let represent an external event scheduled in the i-th federate j-th exit (entry) point at simulation time t, where t can be, in general, smaller or equal to t' that represents the simulation time when the event is supposed to be simulated. An event scheduled into the event list of a simulator is defined as external if one of the three following conditions holds:

- The realization of the event depends on the state of a federate that is, in general, different from the one that scheduled the event. One example of external event is when the sending federate (model M1) wants to transfer a part to the receiving federate (model M2), the possibility for the leaving event to be simulated depends on the state of the queue of the receiving federate.
- The simulation of the event leads to changes into the state of other federates in the federation. This is the case when the downstream machine to the first buffer in the receiving model takes a part from the buffer thus changing its availability, this information must be delivered to sending models that are willing to transfer an entity, the state of the sending federate(s) will change depending on the information delivered (W1a can be idle or blocked).
- The event is not scheduled by the simulator that will simulate it, but is put into the simulation event list by the middleware associated with the simulator. This is the case of the External Arrival Event (Section 3.3.1.).

Herein three types of external events are taken into consideration:

- *Entity transfer event*, this event happens when a sending federate wants to transfer a part to a receiving federate.
- *Buffer_availability* change event, this is a departure event from the workstation downstream the buffer representing the entry point of the receiving model.
- *External Arrival event*, this event is scheduled by the middleware inside the simulation event list of the receiving federate every time a part has to be transferred.

If $t < t'$ it means that the simulation message can be sent by the sending (receiving) model and received by the target federate before the event contained in the message has to be

executed. When this happens it is possible to minimize the use of the zero lookahead for the communication between federates.

The federate sending the message can communicate with $t < t'$ under the following conditions:

- The Entity transfer event is scheduled when the part enters the machine in the sending model. In this case the event is put into the event list a number of time units before it must be simulated that is at least equal to the processing time of the workstation under analysis. In the case the event is scheduled when the part leaves the workstation, then the condition holds if there exists a transfer time between the sending and the receiving model that is larger than zero and no events affect the arrival of the part once the transfer has started. The conditions aforementioned are not unrealistic when a manufacturing plant is simulated: both in the case the event is scheduled before or after the processing activity, the time between the departure from the exit point and the arrival to the entry point is in general not negligible. Nonetheless, in both the aforementioned cases, it is required that no other external events are scheduled by the same exit point during the interval (t; t'). This can happen when, after a leaving event has been scheduled, a failure affects the machine. In this case the information related to the part to be transferred has already been delivered and cannot be updated. As a consequence an external arrival event will be scheduled in the receiving model although the sending model will not be able to deliver the part because of the machine failure. A solution to this issue is part of present research.
- It is possible to communicate in advance the Buffer availability change event if the workstation processing the part schedules the leave event in advance to its realization and no other events are scheduled by the same workstation during the interval (t; t'). However, the zero lookahead cannot be avoided by the sending federate which cannot be aware of the downstream buffer changes and then it will send update request at the lookahead value.
- The zero lookahead can be avoided if the middleware of the receiving model can schedule the External Arrival event in advance and then inform the target federate(s) on the availability of the buffer in advance. This condition can be satisfied based on the entity transfer event characteristics.

In the case one or more of the conditions aforementioned do not hold than the communication protocol shown in the Section 3.3.2. implies the use of zero lookahead. If the hypothesis that no additional external events must be scheduled by the same exit (entry) point in a federate (sending or receiving) within the time interval (t; t') is relaxed, then the middleware should be able to arrange incoming events in a queue and wait before delivering the information to the simulator until when the most updated information has been received. However it is quite straightforward to show that, in the worst case, the middleware should wait until when the simulation time reaches t', and therefore all the time advance requests would be performed at the zero lookahead. This relaxation is under analysis.

4. Distributed simulation in industry: A case study

This section presents an application of the architecture and communication protocol proposed in Section 3.3. The aim is to evaluate whether the use of distributed simulation can help to better analyze the dynamics of complex manufacturing systems, whereas the comparison between the HLA-based distributed simulation and a monolithic simulation in terms of computational efficiency is out of scope.

Herein the attention is focused on the industrial field represented by sheet metal production. In this industrial field, the production systems are characterized by the presence of at least two subsystems interacting with each other: the Roll Milling System and the Roll Shop. The Roll Milling System produces sheet metal using milling rolls that are subject to wearing out process. The rolls must be replaced when worn out to avoid defects in the laminates. Then the Roll Shop performs the grinding process to recondition the worn out rolls.

The following types of rolls have been considered in the case study:

- Intermediate Rolls (IMR) representing back-up rolls that are not in contact with the laminate. Depending on the size of the roll, the IMR roll will be referred to as IMR_1 (the bigger roll type) and IMR_2.
- Work Roll (WR) representing the rolls directly in contact with the laminate. Also in this case there are two subtypes that will be referred to as WR_1 (the bigger roll type) and WR_2.

If the attention is focused on the rolls, then the resulting production system is a closed loop: the Roll Milling System sends batches of worn out rolls to the Roll Shop following a given policy and receives reconditioned rolls back. Both the Roll Milling System and the Roll Shop have finite capacity buffers, therefore it is necessary to check whether the buffer in the system receiving the rolls has enough free slots. The deadlock in the closed loop is avoided because the number of rolls circulating in the system is less than the number of available slots (taking into account also the machines) and it is constant.

The two subsystems forming a closed loop are strongly related and their reciprocal influence should be considered to properly evaluate the performance of the whole factory by means of a comprehensive simulation model. However, both the Roll Shop designer and the Roll Milling System owner usually develop their own detailed simulator to evaluate the performance of their subsystem, because of the lack of shared information between the owner of the Roll Milling System and the Roll Shop designer. Indeed, the owner of the Roll Milling System usually provides the Roll Shop designer only with aggregated data about the yearly average demand of worn out rolls to be reconditioned. Moreover, the Roll Milling System works according to specific roll changing policies that are not shared with the Roll Shop designer even if they play a key role in the dynamics of the whole factory. For instance, when a roll is worn out, also the other rolls are checked and if their remaining duration is under a predefined threshold, then they are sent to the Roll Shop together with the completely worn out rolls. The presence of roll changing policies determines a relation

between different roll types, since a roll can be sent to the Roll Shop depending on the behavior of other roll types.

Even if separate simulators are developed, the Roll Shop designer still has to evaluate the performance of the whole system while taking into account the influence of the Roll Milling System related to (1) the arrival rate of worn out rolls from the Roll Milling System that is estimated from the yearly aggregate demand of reconditioned rolls and (2) the acceptance of the reconditioned rolls sent by the Roll Shop (closed loop model).

In addition to this the Roll Shop designer has to guarantee that the Roll Milling System the Roll Shop is being designed for, never waits for reconditioned rolls interrupting the production of sheet metal.

Hence, even if simulation models are available, usually the Roll Shop designer over-dimensions the number of rolls that have to populate the whole system to avoid any waiting time at the Roll Milling System. For this reason we focused our first analysis on the effect of the number of rolls over the performance of the whole system (Roll Milling System and Roll Shop).

Sections 4.1 and 4.2 will give the main details characterizing the simulation models, whereas Section 4.3 will present (Section 4.3.1 and Section 4.3.2) and compare (Section 4.3.3) two approaches for the system analysis.

4.1. The Roll Shop simulator

In this section the simulator of the Roll Shop developed for the case of interest will be explained in detail.

The Roll Shop simulator has been developed in C++ language using the object oriented paradigm. The C++ based simulator emulates a COTS, following the approach showed in Wang et al. [67], [68]. Figure 11 gives a pictorial representation of the simulation model for the Roll Shop under analysis.

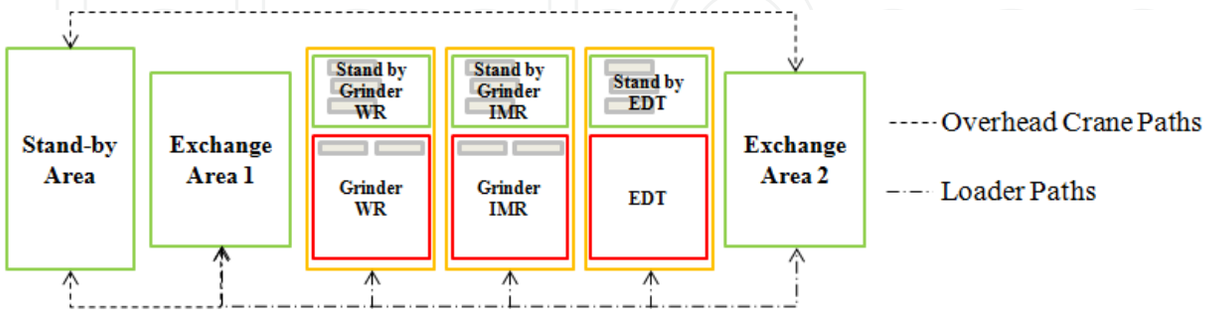

Figure 11. The Roll Shop System representation

The Roll Shop is composed by the following elements (Figure 11).

- *Buffer areas* where the rolls are kept while waiting to be transferred to the Roll Shop or the Milling system. The buffer areas can be:

    - *Stand-by Area*, represents the entry/exit point of the Roll Shop and only the overhead crane can access it. The batches of rolls coming from the Roll Milling System and the batches of grinded rolls to send back to the Roll Milling System are placed here.
    - *Exchange Area 1*, represents the interface between the part of the system managed by the overhead-crane and the grinding system served by the loader
    - *Exchange Area 2*, represents the interface between the exit of the grinding system and the exit from the roll shop system managed by the overhead-crane.
- *Workstations* where the rolls are reconditioned;
    - *Grinder Machine Work* (Grinder WR), i.e. grinding machine dedicated to *work roll* type.
    - *Grinder Machine Intermediate* (Grinder IMR), i.e. grinding machine dedicated to *intermediate roll* type.
    - *Electro-Discharge Texturing Machine* (EDT), i.e. machine executing a surface finishing process on the rolls.
- Two types of conveyors:
    - *Loader*, i.e. an automatic conveyor that transfers rolls to the grinding machines
    - *Overhead crane*, i.e. a semi-automatic handling system to transfer rolls from the arrival point in the roll shop (Stand by area) to the exchange areas of the system

All the workstations, but the EDT, have two buffer positions (grey rectangles in Figure 11) within the working area.

The main parameters needed to configure the simulator are:

- *Size* of the roll batches
- *Type* of rolls (e.g. WR_1 and WR_2, IMR_1 and IMR_2 as described in Section 4)
- *Process Sequence* for every roll, i.e. the process plan and the assignment of operations to the production resources in the Roll Shop. The process sequence depends on the roll type.
- *Processing time* of each operation. Those processing time have been considered deterministic for the experiments presented in this section (i.e. no failures affect the workstations).
- *Transfer time, i.e.* the time to move the roll within the Roll Shop. This is a deterministic quantity as a function of the path.
- *Number of workstations* of each type. We have dedicated machines, in particular we have one grinding machine dedicated to WR type (i.e. WR_1 and WR_2) and one grinding machine dedicated to IMR type roll (i.e. IMR_1 and IMR_2). We have one EDT machine (processing only WR type rolls). In case the WR type roll finds the dedicated grinding machine occupied and the IMR machine is idle, then it can be processed also on the IMR machine. However the time required for the process increases.

Table 4 summarizes the output statistics that can be gathered from the Roll Shop simulator. The minimum, maximum average values are supplied for every statistic, and the variance is computed as well.

Component	Statistics	Unit of Measurement
Roll	Number of rolls in the Roll Shop system for every roll type	-
	System Time for every roll type	[min/roll]
	Grinding time for every roll type	[min/roll]
	Transfer time for every roll type	[min/roll]
Machine	Utilization	[%]
	Processing Time	[min/roll]
	Number of processed rolls	-
	Number of rolls in the buffer area	-
Conveyor	Transfer time	[min/transfer]
	Idle time	[min]
	Utilization	[%]
Buffer	Number of rolls in the buffer	-
	Waiting time in the buffer	[min/roll]

Table 4. Roll Shop Statistics

4.2. The Roll Milling System simulator

In this section the simulator of the Roll Milling System developed for the case of interest will be explained in detail.

The Roll Milling System simulator has been developed in C++ language. The C++ based simulator emulates a COTS, following the approach showed by Wang et al. [67], [68].

A generic milling system can be represented as shown in Figure 12, whereas the simulation model realized for the case presented is graphically represented in Figure 13.

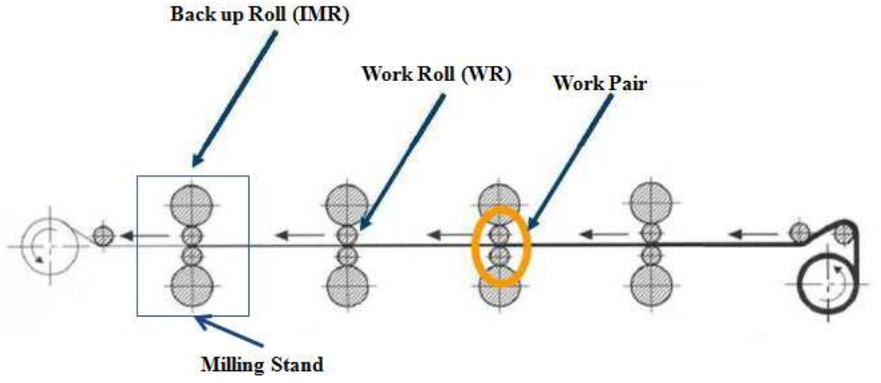

Figure 12. Roll Milling System [87]

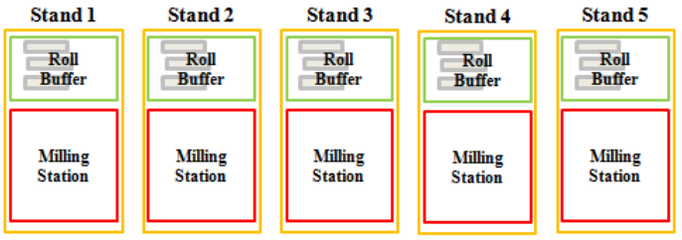

Figure 13. Milling System Simulation Model

The Roll Milling System considered for the industrial case is composed of 5 stands (refer to Figure 12 for the definition of *stand*) each characterized by a milling station and a buffer for rolls (Figure 13). It is important to highlight that more rolls are needed with respect to those in use to process the metal sheet in order to minimize the waiting time of the milling system when the rolls are changed.

Once the rolls worn out, the Roll Milling System interrupts the process and the rolls must be replaced. The rolls are then replaced with the rolls of the same type available at the rolls buffer (Figure 13) close to the station. In the case the rolls required are not available the Roll Milling System stops producing. The worn out rolls are sent to the Roll Shop System as soon as a batch of rolls is ready (the size of the batch is usually fixed and represents a parameter of the simulation model). The batches are then transferred to the Roll Shop by means of a special conveyor, the *Transfer Car*.

The interval between roll changes (*interchange time*) is the interval between two consecutive sending of the same roll to the Roll Shop. This interval is fundamental to the correct sizing of the Roll Shop Plant (i.e. the number of grinding machines for every type, the size of the buffer areas) and of the number of rolls, for every type, populating the system.

This interval is mainly related to the life duration and the roll changing policies adopted within the system. For the industrial case considered, these policies can be brought back to two main criteria:

1. If an IMR roll has to be changed, since this requires high setup time, also the WR rolls from the same station are sent to the Roll Shop even if they have not reached their end on life.
2. If more batches have almost reached their life duration, they are sent together to grinding process to avoid multiple sending.

The main parameters to configure the simulation model are:

- Number of stands
- Capacity of the buffers at every stand
- Number of rolls in the system
- Size of roll batches and types of rolls
- Life duration for each roll type

The Roll Milling System simulator supplies the following output statistics (Table 5):

Component	Statistics	Unit of Measurement
Roll	Number of rolls in the Roll Milling system for every roll type	-
	Interchange Time	[min]
Station	Utilization	[%]
	Waiting Time	[min]
	Busy Time	[min]
	Number of rolls in the buffer area	-
Buffer	Number of rolls in the buffer	-
	Waiting time in the buffer	[min/roll]
Service Level	Busy Time/Simulation Time	[%]

Table 5. Roll Milling System Simulator Statistics

The Service Level (SL) is the typical key performance indicator (KPI) for analyzing the Roll Milling System. It is defined as the time the Roll Milling System produces sheet metal over the time it is operative (the total simulation time in the case of the computer experiment). It must be highlighted that the system cannot produce if all the required rolls are not present at each station. For this reason every station will have the same "Busy Time", i.e. the same time period during which it produces. The service level of a system can be increased managing the plant in a way such that we always have an available batch of rolls to change the worn out ones that have to be sent to the Roll Shop.

It is then clear that the Service Level would be reduced if the Roll Milling System had to wait too long for reconditioned rolls coming from the Roll Shop.

4.3. Sheet metal production system analysis: Approach A, approach B

The Roll Shop designer may choose two possible approaches to estimate the effects of the Roll Shop design choices over the performance of the Roll Milling System:

- *Approach A*. The Roll Milling System is represented by a simplified model inside the detailed simulation model of the Roll Shop (Section 4.1). This simplified model roughly reproduces the Roll Milling System by generating the arrival of worn out rolls and accepting the reconditioned ones.
- *Approach B*. The performance of the whole factory is evaluated by adopting the HLA-based Infrastructure integrating the simulators of the Roll Milling System and of the Roll Shop.

4.3.1. Approach A

The simulator of the whole system is realized introducing within the detailed simulation model of the Roll Shop a simplified model of the Roll Milling System. Also this simulator is developed in C++ language.

More specifically, the Roll Milling System is modeled as an oracle sending (receiving) batches of worn out (reconditioned) rolls to the Roll Shop based on the information on the wearing out time of every roll type.

It must be stressed that the simulator of Approach A cannot be considered as a proper monolithic simulator of the whole factory, since the Roll Milling System is only poorly modeled.

The input parameters needed to initialize the oracle are:

- Number of rolls present in the system, when the simulation starts, for every type. In this case the number of rolls represents the total number of rolls in the Roll Milling System. The only initial condition we can set using this simulation model is that all the rolls at the simulation start are at the beginning of their life and are all in the Roll Milling System, whereas the Roll Shop is empty.
- Life duration of every roll type.
- Size of the batch of rolls for every roll type (the rolls are moved in batches as explained in Section 4.1 and Section 4.2).

Table 6 defines the life duration of the rolls (for every type) given as input for Approach A..

Roll Type	WR_1	WR_2	IMR_1	IMR_2
Batch Size [#Rolls/batch]	8	2	8	2
Duration [min/batch]	438	288	3228	1452

Table 6. Rolls parameters

Table 7 reports the results in terms of average intervals between rolls change, estimated running the simulation model of the Roll Milling System (Section 4.2) as standalone. The life duration given as input to the Roll Milling System simulator was the same given in Table 6. Although the life duration used is the same the resulting intervals between rolls change are different because of the effect of the roll changing policies which are not taken into account in the simplified model of the Roll Milling System (Section 4.2).

Time to change rolls batch [min/batch]	Detailed Model
WR_1	432
WR_2	288
IMR_1	3024.04
IMR_2	1440

Table 7. Rolls interarrival time estimated running the MS simulation model

In particular the intervals between rolls change result decreased because the WR_2 type, characterized by the shorter life duration, draws the change of all other roll types. This result represents a first motivation towards the distributed approach. Indeed it shows that the aggregated information related to the interchange time is not enough to represent the

dynamics of the Roll Milling System. A possible idea to increase the accuracy of Approach A could be to replace data in Table 6 with the estimated information in Table 7, thus taking into account, at least on average, the behavior of the Roll Milling System.

However, the missing feature of this approach is that in any case, even updating the life duration, we will not be able to reproduce the *dynamics* over time of the Roll Milling System which is what really affects the estimation of the performance of the whole system more than the average behavior captured by data in Table 7.

In addition, the only statistics of the Roll Milling System that can be gathered if Approach A is adopted are:

- Average Number of Rolls at every stand
- Average Waiting Time for grinded rolls
- Service Level. In particular, the utilization time (Section 4.2) is estimated as the difference between the total simulation time and the computed sum of time intervals during which the number of rolls for at least one type are equal to 0. If this condition is verified then the Roll Milling System cannot produce and has to wait for reconditioned rolls. As defined in Section 4.2, the ratio between this time and the total simulation time gives the estimate of the service level.

Summarizing, Approach A supplies an approximate estimate of the whole system performance:

- The real behavior of the Roll Milling System cannot be precisely modeled since it is reduced to a black box sending and receiving rolls (e.g. the roll changing policies are not modeled).
- The performance of the Roll Milling System cannot be evaluated in detail (e.g. mean starvation time for every station, mean level of roll buffers, etc).

4.3.2. Approach B

In Approach B, the detailed models of the two subsystems are directly adopted. Indeed the two simulators described in Section 4.1 and 4.2 are linked together thanks to the HLA-based developed infrastructure (Figure 15).

The HLA-based architecture was implemented as follows:

- MAK-RTI 3.3.2 (www.mak.com) was used as the RTI component implementation.
- The middleware was developed in C++ language following the specifications defined in Section 3.3 and was named *SimulationMiddleware*.

The *FederateAmbassador* and *RTIAmbassador* were provided by MAK-RTI as C++ classes and were linked to the *SimulationMiddleware*. Further extensions were needed to implement the proposed modification to ETS (Section 3.2) and the Simulation Messages (Section 3.3.1). The former required a modification to *FederateAmbassador* class, whereas the latter led to the development of a new C++ class. The *SimulationMiddleware* was implemented to manage the information contained in Simulation Messages.

The interaction tables (Section 3.2) developed for this case are shown in Tables 10,11,12 and 13.

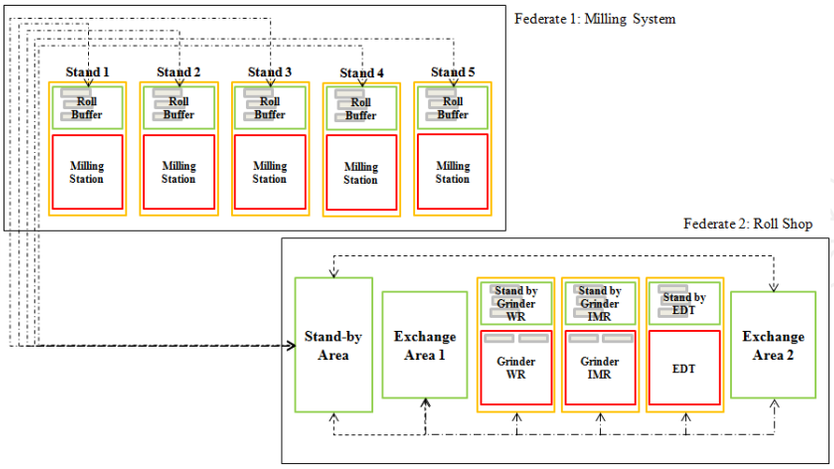

Figure 14. Distributed Simulation of Roll Shop and Roll Milling System

4.3.3. Comparison between approach A and approach B

The two approaches have been compared by designing a set of experiments characterized as follows:

- Three experimental conditions were designed with reference to the total number of rolls circulating in the whole system. These three conditions were defined as Low, Medium, High level.
- The simulation run length was set to six months (4 weeks of transitory period). The roll changing policy adopted for the Approach B simulator has been kept fixed throughout the experimentation.

The results of the experiments are shown in Table 8. Approach A and Approach B are compared in terms of the estimated Service Level (refer to Section 4.2 for the definition). The results show that the difference between the two approaches is larger for the High and Medium level conditions.

When the level of rolls is Low the roll changing policy does not affect the overall performance of the production system because the Roll Milling System is frequently starved and therefore the estimations are similar (consider also that the Low level condition has no industrial meaning, but was considered to study the service level response). In case of Medium and High level conditions the workload of the rolls in the Roll Shop can be strongly influenced by the roll changing policy, thus generating a higher difference in the estimation between the two approaches.

Approach B generates more accurate estimates of the whole system performance because the Roll Milling System is represented with high level of detail and the roll changing policies are modeled. In addition to this if Approach B is used, detailed information related to the Roll Milling System performance are available. Table 9 shows an example of output for the

average statistics related to the Roll Milling System from the simulation of the High level condition in Table 8.

Experimental Conditions	*Approach A*	*Approach B*	*Percentage difference*
High Level	0.995	0.872	12.3
Medium Level	0.946	0.682	27.7
Low Level	0.308	0.273	3.5

Table 8. Service Level Results

Output Statistic	**Value**
Busy Time	5.2[months]
Waiting Time	0.8[months]
Utilization	87.2[%]
Number of Rolls	Stand 1: 1.177 WR 1.470 IMR
	Stand 2: 1.177 WR 1.470 IMR
	Stand 3: 1.177 WR 1.470 IMR
	Stand 4: 1.177 WR 1.470 IMR
	Stand 5: 3.401 WR 1.740 IMR
Service Level	87.2[%]

Table 9. Roll Milling System simulator. Output Statistics

The number of rolls in the system together with the roll changing policy have a strong impact on the workload conditions of the Roll Shop. This aspect can only be taken into account under Approach B and the results show the dramatic difference in performance estimation due to this additional information that characterizes the model.

Approach A is overestimating the performance of the system, thus decreasing its effectiveness as supporting tool for decision making.

Based on the analysis carried out so far, it was decided to design further experiments to analyze the behavior of the whole system with different starting workload conditions, i.e.

the number of rolls that are present in the Roll Milling System when the simulation starts. These experiments can be useful to analyze the ramp-up period and select the roll changing policy that avoids the arising of critical workload conditions. These additional experiments can be carried out only adopting *Approach B*, since the starting workload conditions cannot be modeled with *Approach A* (Section 4.3.1). Indeed, the simplified model generates rolls for the Roll Shop independently from the starting workload conditions. Therefore, the simplified model would generate roll arrivals even if all the rolls are already located in the Roll Shop, thus incorrectly increasing the number of rolls in the whole system.

The second set of experiments was designed as follows:

- Two *types of roll* circulate in the factory (*WR* and *IMR*). The roll of type *IMR* has a longer roll life than *WR*
- For each type of roll three levels of the *Starting Workload* (i.e. number of rolls) in the Roll Milling System are considered
- Three simulation run lengths are considered, i.e. 1 week, 2 weeks and 4 weeks
- The roll changing policy is fixed for all experiments
- total number of rolls is equal to the *High* level of the previous experimentation and is fixed for all the experiments.

Fig. 15 shows the main effects plot for the *Service Level* evaluated by simulating the 27 resulting experimental conditions with *Approach B*. The plot suggests a significant influence of the factor *Starting Workload for WR*. This roll type assumes a key role because of its short roll life (Section 4.3.1). If the *Starting Workload For WR* is *Low*, the Roll Shop can hardly follow the frequent roll requests of *WR* from the Roll Milling System and low values of *SL* are observed. This phenomenon occurs in all conditions of the simulation lengths, however it mitigates when the simulation length increases. Indeed the *SL* tends to a stationary value that is independent from the starting conditions. Nonetheless this analysis can be useful for the Roll Milling System owner that can individuate critical conditions, thus designing roll changing policies that avoid the occurrence of these situations during the ramp-up period.

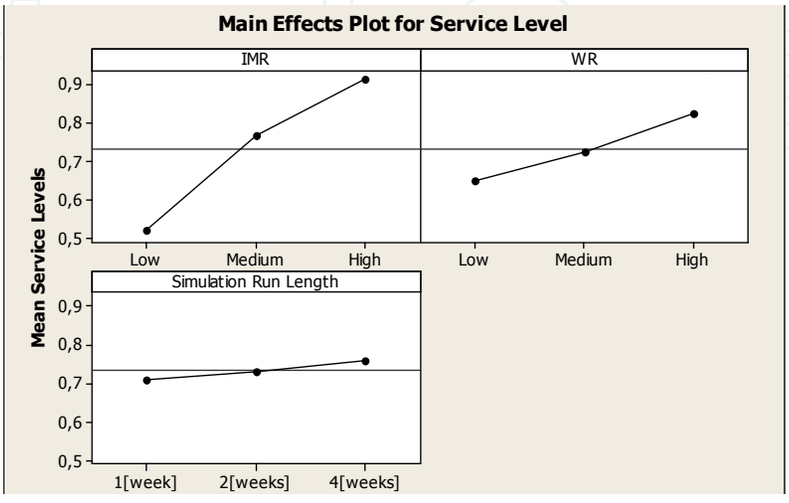

Figure 15. Main Effects Plot of Service Lovel, Approach B

HLAinteraction Root(N)	TransferEntity (N/S)	TransferEntityFromFedSourceMilling(N/S)	TransferEntityFromFedSourceExMillingtoFedDestEnRollShop(P/S)
	Entity (N/S)	TransferEntityFromFedSourceRollShop(N/S)	

Table 10. Interaction Class Table

Interaction	Parameter	DataType	Transportation	Order
TransferEntityFromFedSourceExMillingtoFedDestEnRollShop(P/S)	RollEntity	RollType	HLAreliable	TimeStamp
	ReceivedRollEntity	RollType	HLAreliable	TimeStamp
	RollShopBuffer	HLAInteger32BE	HLAreliable	TimeStamp
	SourcePriority	HLAInteger32BE	HLAreliable	TimeStamp
	EntityTransferTime	HLAFloat32BE	HLAreliable	TimeStamp

Table 11. Parameter Table for Roll Shop SOM

Interaction	Parameter	DataType	Transportation	Order
TransferEntityFromFedSourceExRollMillingtoFedDestEnRollShop (P/S)	RollEntity	RollType	HLAreliable	TimeStamp
	ReceivedRollEntity	RollType	HLAreliable	TimeStamp
	MillingBuffer	HLAInteger32BE	HLAreliable	TimeStamp
	SourcePriority	HLAInteger32BE	HLAreliable	TimeStamp
	EntityTransferTime	HLAFloat32BE	HLAreliable	TimeStamp

Table 12. Parameter Table for Milling System SOM

Record Name	Field		Encoding
	Name	Type	HLAfixedRecord
RollType	Roll	HLAASCIIString	
	BatchSize	HLAInteger32BE	

Table 13. Fixed Record Datatype Table (SOM) for Roll Shop and Milling Systems

5. Conclusions

This chapter has presented an overview of distributed simulation and the contemporary innovations in the use of distributed modeling to support the analysis of complex systems. The attention has been focused on CSP-based distributed simulation in civilian applications and especially in manufacturing domain. The literature review showed the need of a general standard solution to the distributed simulation of systems within the civilian domain to increase the use of distributed techniques for the analysis of complex systems. The Interoperability Reference Model standard released by SISO CSPI-PDG has been analyzed. Furthermore, the latest advancements in ETS standard proposal and Communication Protocol between federates within HLA-based distributed environment have been shown. In the end the industrial application of an HLA-based infrastructure proved the benefits of the distributed approach to effectively analyze the behavior of complex industrial systems.

The distribute simulation is acquiring more and more interest also because the need to interoperate several simulators leads to the need to improve the methodologies and the tools developed so far for the simulation of Discrete Event Systems. In other words the issues arising when trying to make several simulators interoperate are those issues the simulation community has been dealing with in the last years.

Further effort is needed in the formalization of the IRMs. In particular the presence of shared resources and the modeling of the control policies characterizing the system represent challenging issues not yet solved.

The communication protocols need to be enhanced and the zero-lookahead issue represents one of the main bottlenecks against the increase of efficiency of distributed simulations. Research effort is necessary to come up with new algorithms enabling the avoidance of zero-lookahead. In this area the research on protocols that do not force to send interaction at every time unit to communicate the state of the federates, but enable the interaction depending on the system state (Adaptive Communication Protocols) look promising.

The need of a shared data model ([26], [7]) and of a common definition of the objects which are input and output of the simulation and a common simulation language, are all scientific and technical challenging topics that make research in distributed simulation always up to date.

In particular the definition of a common reference model to describe information generated by each simulator while it is running will be a key factor for the success of the distributed simulation technique. A fundamental contribution in this field was given by the Core Manufacturing Simulation Data CMSD [7], but the interoperability between simulators is still far from being reached. The data modeling research topic is wider than what just stated and covers several areas and simulation is just one of those. For example proposals to standardize the information modeling for manufacturing systems have been done in [83-84].

The European project Virtual Factory Framework VFF ([38] [85-86]) represents one of the latest proposals from the research community in terms of framework supporting the

interoperability between digital tools. Future research will evaluate if the VFF approach can be exploited by distributed simulation applications.

Author details

Giulia Pedrielli and Tullio Tolio
Politecnico di Milano, Dipartimento di Ingegneria Meccanica, Milano (MI), Italy

Walter Terkaj and Marco Sacco
Istituto Tecnologie Industriali e Automazione (ITIA), Consiglio Nazionale delle Ricerche (CNR), Milano (MI) Italy

Acknowledgement

The research reported in this chapter has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No: NMP2 2010-228595, Virtual Factory Framework (VFF). Sections of this chapter are based on the paper: Pedrielli, G., Sacco, M., Terkaj, W., Tolio, T., Simulation of complex manufacturing systems via HLA-based infrastructure. *Journal Of Simulation, to be published*. Authors would like to acknowledge Professor Taylor S.J.E and Professor Starssburger S. for their ongoing support and their work the in developing current range of standards for distributed simulation. Authors acknowledge Tenova Pomini Company for providing the support to build the industrial case.

6. References

[1] Ieee standard for modeling and simulation (m amp;s) high level architecture (hla) - framework and rules. IEEE Std. 1516-2000, pages i-22, 2000.

[2] Khaldoon Al-Zoubi and Gabriel A. Wainer. Performing distributed simulation with restful web services approach.

[3] James G. Anderson. Simulation in the health services and biomedicine, pages 275{293. Kluwer Academic Publishers, Norwell, MA, USA, 2003.

[4] J. Banks, J.C. Hugan, P. Lendermann, C. McLean, E.H. Page, C.D. Pegden, O. Ulgen, and J.R. Wilson. The future of the simulation industry. In Simulation Conference, 2003. Proceedings of the 2003 Winter, volume 2, pages 2033-2043 vol.2, dec. 2003.

[5] J. Banks and B.L. Nelson. Discrete-event system simulation. Prentice Hall, 2010.

[6] Lee A. Belfore, II. Simulation in environmental and ecological systems, pages 295-314. Kluwer Academic Publishers, Norwell, MA, USA, 2003.

[7] S. Bergmann, S. Stelzer, and S Strassburger. Initialization of simulation models using cmsd. In Simulation Conference (WSC), Proceedings of the 2011 Winter, WSC '11, pages 594-602. IEEE Press, 2011.

[8] C.A. Boer. Distributed Simulation in Industry. PhD thesis, Erasmus University Rotterdam, Rotterdam, The Netherlands, October 2005.

[9] C.A. Boer, A. de Bruin, and A. Verbraeck. Distributed simulation in industry - a survey part 1 - the cots vendors. volume 0, pages 1053-1060, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[10] C.A. Boer, A. de Bruin, and A. Verbraeck. Distributed simulation in industry - a survey part 3 - the hla standard in industry. In Simulation Conference, 2008. WSC 2008. Winter, pages 1094-1102, dec. 2008.

[11] Csaba Attila Boer, Arie de Bruin, and Alexander Verbraeck. Distributed simulation in industry - a survey: part 2 - experts on distributed simulation. In Proceedings of the 38th conference on Winter simulation, WSC '06, pages 1061-1068. Winter Simulation Conference, 2006.

[12] Csaba Attila Boer and Alexander Verbraeck. Distributed simulation and manufacturing: distributed simulation with cots simulation packages. In Proceedings of the 35th conference on Winter simulation: driving innovation, WSC '03, pages 829-837. Winter Simulation Conference, 2003.

[13] Vesna Bosilj-Vuksic, Mojca Indihar Stemberger, Jurij Jaklic, and Andrej Kovacic. Assessment of e-business transformation using simulation modeling. Simulation.

[14] Agostino G. Bruzzone. Preface to modeling and simulation methodologies for logistics and manufacturing optimization. Simulation, 80(3):119-120, 2004.

[15] Judith S. Dahmann, Frederick Kuhl, and Richard Weatherly. Standards for simulation: As simple as possible but not simpler the high level architecture for simulation. SIMULATION, 71(6):378- 387, 1998.

[16] Wilhelm Dangelmaier and Bengt Mueck. Simulation in business administration and management, pages 391-406. Kluwer Academic Publishers, Norwell, MA, USA, 2003.

[17] Paul K. Davis. Military applications of simulation, pages 407-435. Kluwer Academic Publishers, Norwell, MA, USA, 2003.

[18] T. Eldabi and R.J. Paul. A proposed approach for modeling healthcare systems for understanding. In Simulation Conference, 2001. Proceedings of the Winter, volume 2, pages 1412 -1420 vol.2, 2001.

[19] Richard Fujimoto, Michael Hunter, Jason Sirichoke, Mahesh Palekar, Hoe Kim, and Wonho Suh. Ad hoc distributed simulations. In Proceedings of the 21st International Workshop on Principles of Advanced and Distributed Simulation, PADS '07, pages 15-24, Washington, DC, USA, 2007. IEEE Computer Society.

[20] R.M. Fujimoto. Parallel and distributed simulation systems. Wiley series on parallel and distributed computing. Wiley, 2000.

[21] Boon Ping Gan, Peter Lendermann, Malcolm Yoke Hean Low, Stephen J. Turner, Xiaoguang Wang, and Simon J. E. Taylor. Interoperating autosched ap using the high level architecture. In Proceedings of the 37th conference on Winter simulation, WSC '05, pages 394-401. Winter Simulation Conference, 2005.

[22] Boon Ping Gan, Peter Lendermann, Malcolm Yoke Hean Low, Stephen J. Turner, Xiaoguang Wang, and Simon J. E. Taylor. Architecture and performance of an hla-based

distributed decision support system for a semiconductor supply chain. SimTech technical reports, 7(4):220-226, 2007.

[23] Sumit Ghosh and Tony Lee. Modeling and Asynchronous Distributed Simulation Analyzing Complex Systems. Wiley-IEEE Press, 1st edition, 2000.

[24] Strong D.R. Richards N. Goel N.C. Goel, S. A simulation-based method for the process to allow continuous tracking of quality, cost, and time. Simulation, 78(5):330-337, 2001.

[25] Hironori Hibino, Yoshiyuki Yura, Yoshiro Fukuda, Keiji Mitsuyuki, and Kiyoshi Kaneda. Manufacturing modeling architectures: manufacturing adapter of distributed simulation systems using hla. In Proceedings of the 34th conference on Winter simulation: exploring new frontiers, WSC '02, pages 1099-1107. Winter Simulation Conference, 2002.

[26] Marcus Johansson, Bjorn Johansson, Anders Skoogh, Swee Leong, Frank Riddick, Y. Tina Lee, Guodong Shao, and Par Klingstam. A test implementation of the core manufacturing simulation data specification. In Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come, WSC '07, pages 1673-1681, Piscataway, NJ, USA, 2007. IEEE Press.

[27] Frederick Kuhl, Richard Weatherly, and Judith Dahmann. Creating computer simulation systems: an introduction to the high level architecture. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[28] P. Lendermann. About the need for distributed simulation technology for the resolution of realworld manufacturing and logistics problems. In Simulation Conference, 2006. WSC 06. Proceedings of the Winter, pages 1119 -1128, dec. 2006.

[29] P. Lendermann, M.U. Heinicke, L.F. McGinnis, C. McLean, S. Strassburger, and S.J.E. Taylor. Panel: distributed simulation in industry - a real-world necessity or ivory tower fancy? In Simulation Conference, 2007 Winter, pages 1053-1062, dec. 2007.

[30] Richard J. Linn, Chin-Sheng Chen, and Jorge A. Lozan. Manufacturing supply chain applications 2: development of distributed simulation model for the transporter entity in a supply chain process. In Proceedings of the 34th conference on Winter simulation: exploring new frontiers, WSC '02, pages 1319-1326. Winter Simulation Conference, 2002.

[31] A. W. Malik. An optimistic parallel simulation for cloud computing environments. SCS M&S Magazine, 6:1-9, 2010.

[32] Charles McLean and Swee Leong. The expanding role of simulation in future manufacturing. In Proceedings of the 33nd conference on Winter simulation, WSC '01, pages 1478-1486, Washington, DC, USA, 2001. IEEE Computer Society.

[33] Charles McLean, Swee Leong, Charley Harrell, Philomena M. Zimmerman, and Roberto F. Lu. Simulation standards: current status, needs, future directions, panel: simulation standards: current status, needs, and future directions. In Proceedings of the 35th conference on Winter simulation: driving innovation, WSC '03, pages 2019-2026. Winter Simulation Conference, 2003.

[34] Navonil Mustafee, Simon J.E. Taylor, Korina Katsaliaki, and Sally Brailsford. Facilitating the analysis of a uk national blood service supply chain using distributed simulation. SIMULATION, 85(2):113-128, 2009.

[35] Brian Unger, and David Jefferson. Distributed simulation, 1988 : proceedings of the SCS Multiconference on Distributed Simulation, 3-5 February, 1988, San Diego, California / edited by Brian Unger and David Jefierson. Society for Computer Simulation International, San Diego, Calif. 1988.

[36] Ernest H. Page and Roger Smith. Introduction to military training simulation: A guide for discrete event simulations, 1998.

[37] Jaebok Park, R. Moraga, L. Rabelo, J. Dawson, M.N. Marin, and J. Sepulveda. Addressing complexity using distributed simulation: a case study in spaceport modeling. In Simulation Conference, 2005 Proceedings of the Winter, page 9 pp., dec. 2005.

[38] G. Pedrielli, P. Scavardone, T. Tolio, M. Sacco, and W. Terkaj. Simulation of complex manufacturing systems via hla-based infrastructure. In Principles of Advanced and Distributed Simulation (PADS), 2011 IEEE Workshop on, pages 1 -9, june 2011.

[39] P. Peschlow and P. Martini. Eficient analysis of simultaneous events in distributed simulation. In Distributed Simulation and Real-Time Applications, 2007. DS-RT 2007. 11th IEEE International Symposium, pages 244-251, oct. 2007.

[40] A. R. Pritchett, M. M. van Paassen, F. P. Wieland, and E. N. Johnson. Aerospace vehicle and air traffic simulation, pages 365-389. Kluwer Academic Publishers, Norwell, MA, USA, 2003.

[41] M. Raab, S. Masik, and T. Schulze. Support system for distributed hla simulations in industrial applications. In Principles of Advanced and Distributed Simulation (PADS), 2011 IEEE Workshop on, pages 1 -7, june 2011.

[42] M. Rabe, F.W. Jkel, and H. Weinaug. Supply chain demonstrator based on federated models and hla application. 2006.

[43] Stewart Robinson. Distributed simulation and simulation practice. SIMULATION, 81(1):5-13, 2005.

[44] Marco Sacco, Giovanni Dal Maso, Ferdinando Milella, Paolo Pedrazzoli, Diego Rovere, and Walter Terkaj. Virtual Factory Manager, volume 6774 of Lecture Notes in Computer Science, pages 397- 406. Springer Berlin, Heidelberg, 2011.

[45] B. Sadoun. Simulation in city planning and engineering, pages 315{341. Kluwer Academic Publishers, Norwell, MA, USA, 2003.

[46] Thomas Schulze, Steffen Strassburger, and Ulrich Klein. Migration of hla into civil domains: Solutions and prototypes for transportation applications. SIMULATION, 73(5):296-303, 1999.

[47] S. Straburger. Overview about the high level architecture for modeling and simulation and recent developments. Simulation News Europe, 16(2):5-14, 2006.

[48] S. Straburger, G. Schmidgall, and S. Haasis. Distributed manufacturing simulation as an enabling technology for the digital factory. Journal of Advanced ManufacturingSystem, 2(1):111 -126, 2003.

[49] S. Strassburger. Distributed Simulation Based on the High Level Architecture in Civilian Application Domains. PhD thesis, Computer Science Faculty, University Otto von Guericke, Magdeburg, 2001.

[50] Steffen Strassburger. The road to cots-interoperability: from generic hla-interfaces towards plug and play capabilities. In Proceedings of the 38th conference on Winter simulation, WSC '06, pages 1111-1118. Winter Simulation Conference, 2006.

[51] Taylor, Strassburger, S.J. Turner, M.Y.H. Low, Xiaoguang Wang, and J. Ladbrook. Developing interoperability standards for distributed simulaton and cots simulation packages with the cspi pdg. In Simulation Conference, 2006. WSC 06. Proceedings of the Winter, pages 1101 -1110, dec. 2006.

[52] S J E Taylor and N Mustafee. An analysis of internal/external event ordering strategies for cots distributed simulation. pages 193-198. Proceedings of the 15th European Simulation Symposium (ESS2003), Delft, 2003.

[53] Simon J. E. Taylor, Navonil Mustafee, Steffen Strassburger, Stephen J. Turner, Malcolm Y. H. Low, and John Ladbrook. The siso cspi pdg standard for commercial off-the-shelf simulation package interoperability reference models. In Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come, WSC '07, pages 594-602, Piscataway, NJ, USA, 2007. IEEE Press.

[54] Simon J. E. Taylor, Stephen J. Turner, and Steffen Strassburger. Guidelines for commercial-off-the-shelf simulation package interoperability. In Proceedings of the 40th Conference on Winter Simulation, WSC '08, pages 193-204. Winter Simulation Conference, 2008.

[55] Simon J.E. Taylor. A proposal for an entity transfer specification standard for cots simulation package interoperation. In Proceedings of the 2004 European Simulation Interoperability Workshop, 2004.

[56] Simon J.E. Taylor. Distributed Modeling, pages 9:1-9:19. Chapman & Hall/CRC, 2007.

[57] S.J.E. Taylor. Realising parallel and distributed simulation in industry: A roadmap. In Principles of Advanced and Distributed Simulation (PADS), 2011 IEEE Workshop on, page 1, june 2011.

[58] S.J.E. Taylor, A. Bruzzone, R. Fujimoto, Boon Ping Gan, S. Strassburger, and R.J. Paul. Distributed simulation and industry: potentials and pitfalls. In Simulation Conference, 2002. Proceedings of the Winter, volume 1, pages 688- 694 vol.1, dec. 2002.

[59] S.J.E Taylor, M. Ghorbani, N. Mustafee, S. J. Turner, T. Kiss, D. Farkas, S. Kite, and S. Strassburger. Distributed computing and modeling & simulation: speeding up simulations and creating large models. In Proceedings of the 2011 Winter Simulation Conference, pages 1-15, December 2011.

[60] S.J.E. Taylor, M. Ghorbani, N. Mustafee, S.J. Turner, T. Kiss, D. Farkas, S. Kite, and S. Strassburger. Distributed computing and modeling amp; simulation: Speeding up

simulations and creating large models. In Simulation Conference (WSC), Proceedings of the 2011 Winter, pages 161-175, dec. 2011.

[61] S.J.E. Taylor, Xiaoguang Wang, S.J. Turner, and M.Y.H. Low. Integrating heterogeneous distributed cots discrete-event simulation packages: an emerging standards-based approach. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 36(1):109- 122, jan. 2006.

[62] S.J.E. Taylor, Xiaoguang Wang, S.J. Turner, and M.Y.H. Low. Integrating heterogeneous distributed cots discrete-event simulation packages: an emerging standards-based approach. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 36(1):109- 122, jan 2006.

[63] Sergio Terzi and Sergio Cavalieri. Simulation in the supply chain context: a survey. Computers in Industry, 53(1):3-16, 2004.

[64] J. Vancza, P. Egri, and L. Monostori. A coordination mechanism for rolling horizon planning in supply networks. CIRP Annals - Manufacturing Technology, 57(1):455- 458, 2008.

[65] Gabriel A. Wainer, Olivier Khaldoon, Al-Zoub Dalle, David R.C. Hill Hill, Saurabh Mittal, Jos L. Risco Martn, Hessam Sarjoughian, Luc Touraille, Mamadou K. Traor, and Zeigler Bernard P. Standardizing DEVS model representation. 2010.

[66] Gang Wang, Chun Jin, and Peng Gao. Adapting arena into hla: Approach and experiment. In Automation and Logistics, 2007 IEEE International Conference on, pages 1709-1713, aug. 2007.

[67] Xiaoguang Wang, Stephen John Turner, and Simon J. E. Taylor. Cots simulation package (csp) interoperability -a solution to synchronous entity passing. In Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation, PADS '06, pages 201-210, Washington, DC, USA, 2006. IEEE Computer Society.

[68] Xiaoguang Wang, Stephen John Turner, Simon J. E. Taylor, Malcolm Yoke Hean Low, and Boon Ping Gan. A cots simulation package emulator (cspe) for investigating cots simulation package interoperability. In Proceedings of the 37th conference on Winter simulation, WSC '05, pages 402-411. Winter Simulation Conference, 2005.

[69] Xiaoguang Wang, Sthephen John Turner, Malcolm Yoke Hean Low, and Boon Ping Gan. A generic architecture for the integration of cots packages with the hla. In Proceedings of the 2004 Operational Research Society Simulation Workshop, Special Interest Group for Simulation, pages 225-233. Association for Computing Machinery's, 2005.

[70] Gregory Zacharewicz, Claudia Frydman, and Norbert Giambiasi. G-devs/hla environment for distributed simulations of workows. Simulation, 84:197-213, May 2008.

[71] zer Uygun, Ercan ztemel, and Cemalettin Kubat. Scenario based distributed manufacturing simulation using hla technologies. Information Sciences, 179(10):1533-1541, 2009. <ce:title>Including Special Issue on Artificial Imune Systems</ce:title>.

[72] Y. Zhang and L. M. Zhang. The Viewable Distributed Simulation Linkage Development Tool Based on Factory Mechanism. Applied Mechanics and Materials, 58:1813-1818, June 2011.

[73] Cosby, L.N. SIMNET: An Insider's Perspective. Ida documenT D-1661, Institute for Defense Analyses 1801 N. Beauregard Street, Alexandria, Virginia 22311-1772. pp: 1-19. March 1995

[74] SISO CSPI-PDG www.sisostds.org.

[75] SISO COTS Simulation Package Interoperability Product Development Group, (2010). Standard for Commercial - off - the - shelf Simulation Package Interoperability Referenve Models. SISO-STD-006-2010.

[76] Kubat, C., Uygun O. (2007). HLA Based Distributed Simulation Model for Integrated Maintenance and Production Scheduling System in Textile Industry. *P.T. Pham, E.E. Eldukhri, A. Soroka (Eds.), Proceedings of 3rd I*PROMS Virtual International Conference, 2–13 July 2007*: 413–418.

[77] Pedrielli, G., Sacco, M, Terkaj, W., Tolio, T. (2012). Simulation of complex manufacturing systems via HLA-based infrastructure. *Journal Of Simulation*. To be published.

[78] Kewley, R., Cook, J., Goerger, N., Henderson, D., Teague, E., (2008). "Federated simulations for systems of systems integration," *Simulation Conference, 2008. WSC 2008. Winter* , vol., no., pp.1121-1129, 7-10 Dec. 2008.

[79] Bruccoleri M, Capello C, Costa A, Nucci F, Terkaj W, Valente A (2009) Testing. In: Tolio T (ed) Design of Flexible Production Systems. Springer: 239-293. ISBN 978-3-540-85413-5.

[80] http://www.pitch.se/

[81] MAK RTI, www.mak.com

[82] Ke Pan; Turner, S.J.; Wentong Cai; Zengxiang Li; , "Implementation of Data Distribution Management services in a Service Oriented HLA RTI," *Simulation Conference (WSC), Proceedings of the 2009 Winter* , vol., no., pp.1027-1038, 13-16 Dec. 2009 doi: 10.1109/WSC.2009.5429557

[83] Colledani M, Terkaj W, Tolio T, Tomasella M (2008) Development of a Conceptual Reference Framework to manage manufacturing knowledge related to Products, Processes and Production Systems. In Bernard A, Tichkiewitch S (eds) Methods and Tools for Effective Knowledge Life-Cycle-Management. Springer: 259-284. ISBN 978-3-540-78430-2.

[84] Colledani M, Terkaj W, Tolio T (2009) Product-Process-System Information Formalization. In: Tolio T (ed) Design of Flexible Production Systems. Springer: 63-86. ISBN 978-3-540-85413-5.

[85] Pedrazzoli, P, Sacco, M, Jönsson, A, Boër, C (2007) Virtual Factory Framework: Key Enabler For Future Manufacturing. In Cunha, PF, Maropoulos, PG (eds) Digital Enterprise Technology, Springer US, pp 83-90

[86] Sacco M, Pedrazzoli P, Terkaj W (2010) VFF: Virtual Factory Framework. Proceedings of 16th International Conference on Concurrent Enterprising, Lugano, Switzerland, 21-23 June 2010.

[87] Kalpakjian Serope; Schmid Steven R. Book title: Tecnologia Meccanica, Editor: Pearon Education Year: 2008 The Figure 12 picture was inspired by the book

IntechOpen