Journal of Simulation (2012) 6, 237–252

www.palgrave-journals.com/jos/

An HLA-based distributed simulation for networked manufacturing systems analysis

G Pedrielli[1]*, M Sacco[2], W Terkaj[2] and T Tolio[1]

[1]*Politecnico di Milano, Dipartimento di Ingegneria Meccanica, MI, Italy;* [2]*Istituto Tecnologie Industriali e Automazione (ITIA), Consiglio Nazionale delle Ricerche (CNR), MI, Italy*

Manufacturing systems can be thought as production networks nodes whose relations have a strong impact on design and analysis of each system. Commercial simulators are already adopted to analyse complex networked systems, but the development of a monolithic model can be too complex or infeasible when a detailed description of the nodes is not available outside the 'owner' of the node. Then the problem can be decomposed modelling complex systems with various simulators that interoperate in a synchronized manner. Herein, the integration of simulators is addressed by taking as a reference the High Level Architecture (HLA). This paper proposes modifications to Commercial-off-the-shelf Simulation Package Interoperability Product Development Group protocols and to use patterns of how HLA can be effectively adopted to support Commercial Simulation Package interoperability: a new solution for the synchronous entity passing problem and modifications to the Entity Transfer Specification are presented. The resulting infrastructure is validated and tested over an industrial case.
Journal of Simulation (2012) **6,** 237–252. doi:10.1057/jos.2012.6; published online 15 June 2012

Keywords: complex manufacturing systems; discrete event simualtion; distributed simulation; HLA; interoperability reference models

1. Introduction

Nowadays, manufacturing companies have to face an increasingly turbulent market characterized by demand variability, unstable requirements from the clients and short product lifecycles (Terkaj *et al*, 2009). In addition to the problems related to the market, the performance of a manufacturing system is deeply affected by its relations with other systems. Indeed, every production system is not a standalone unit, but a node in a production network characterized by complex dynamics that should be considered during the design and analysis phases (Wiendahl and Lutz, 2002). Both in the literature and in the industrial practice, Discrete Event Simulation is used to analyse production and logistics problems in various industrial domains (Law, 2007; Smith, 2003). This is also due to the spreading of Commercial-Off-The-Shelf (COTS) Simulation Packages (CSPs) that are available on the market and provide a wide range of functionalities (eg, visual building of the simulation model, simulation run support, animation, etc).

However, the complexity of a simulation model becomes hardly manageable when the relations between the nodes of a production network must be considered, since the modelling of a single node is already complex by itself and requires specific expertise and information (Vàncza *et al*, 2008). Moreover, in real practice the developer of a simulation model can hardly access all the information characterizing a production network because information sharing is seen as a threat by most of the companies, thus hindering the feasibility of developing a unique and monolithic simulation model to evaluate the performance of complex production networks.

A distributed simulation (DS) approach can be proposed to face the aforementioned criticalities, but several problems arise when trying to interoperate heterogeneous simulators in real industrial cases, even though a standard like High Level Architecture (HLA) (IEEE Standard a, b, c, 2000, Kuhl *et al*, 1999 and IEEE Standard 2003) has already been proposed to support the interoperability between simulators. Indeed, the enhancement of HLA with additional complementary standards (Taylor *et al*, 2006a, b) and the definition of a standard language for CSPs represent relevant and not yet solved scientific and technical challenges (Hibino *et al*, 2002; Straßburger, 2001; Straßburger, 2006b).

The realization of an effective integration between several COTS is hindered by the lack of a unique simulation language and a common standard for data modelling (Sacco *et al*, 2011). Indeed, as long as different COTS do not share a common language, the interaction between the HLA-Run Time Infrastructure (RTI) and the simulator has to be tackled in a dedicated way. An example is represented by the work of Mustafee and Taylor (2006), who propose the development of a manufacturing adapter to the simulation software Simul8. The connection between the simulator and the distributed environment is obtained, thanks to the CSP controller middleware. This controller performs two specific tasks: it communicates with Simul8 through its COM

**Correspondence:* *G Pedrielli, Politecnico di Milano, Dipartimento di Ingegneria Meccanica, Via Giuseppe La Masa, 1 - 20155 Milano, Italy.* E-mail: giulia.pedrielli@mail.polimi.it

interface and interacts with RTI using the HLA interface specification. In particular, the first task is performed by the Simul8 adapter. As highlighted in the aforementioned paper, the adapter is tailored on the specific simulator and the experience suggests that the development of general purpose adapters may be more difficult than it logically seems.

On the other hand, the definition of a data model describing the system under analysis would facilitate the exchange of information between disparate software applications. In the manufacturing domain a lot of interest is being given to standard information modelling (examples are Core Manufacturing Simulation Data—CMSD—Lee *et al*, 2006, and Simulation Data Exchange—SDX—Sly and Shreekanth, 2001).

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) and on a draft proposal for a standard to support the CSPs interoperability (Entity Transfer Specification, ETS). Nevertheless, an effective interoperability among CSPs is still far to be reached in industrial contexts.

Boer and Verbraeck (2003) investigated the main benefits and criticalities related to the industrial application of HLA by interviewing the actors involved in the problem (eg, simulation model developers, software houses, HLA experts). The results of the survey showed that CSPs vendors do not see direct benefits in using DS, whereas in industry HLA is considered troublesome because of the lack of experienced users and the complexity of the standard.

The remainder of this paper is structured as follows. Section 2 presents preliminary results obtained by the authors from the analysis of the literature on HLA applied in civil domain. Section 3 presents the Problem Statement, delving into the problem of CSP interoperability and the IRMs. Section 4 analyses the literature and highlights some of the open issues. Section 5 proposes a solution to the Type A.2 IRM, a modification to ETS, and a communication protocol between the CSP and its adapter taking as a reference the work already carried out by CSPI-PDG. Section 6 addresses the implementation of the proposed solution that is validated (Section 7) and then tested over a realistic industrial case (Section 8). Finally, conclusions and future developments are drawn in Section 9.

2. Literature review

Although HLA-based DS has received significant attention from the scientific research and is frequently adopted in military applications, several issues arise when DS is proposed outside the military environment (see Boer and Verbraeck, 2003; Boer, 2005; Boer *et al*, 2006a b c; Alvarado *et al*, 2008; Pedrielli *et al*, 2011). The literature related to the application of HLA-based DS technique in the civil domain and, in particular, the manufacturing domain was investigated aiming at:

- Individuating the specific fields in the civil domain where DS is currently more applied.
- Identifying the motivations that lead to the use of DS technique.
- Highlighting the main technical and scientific open issues that still represent a hurdle for the use of DS.

The bibliography 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 Application, Industrial Application and Civil Applications. The use of these keywords brought to identify 20 core papers based on the number of citations (Fujimoto, 1998; Straßburger, 2001; Linn and Chen, 2002; Taylor *et al*, 2002, 2003, 2004; Banks *et al*, 2003; Boer and Verbraeck, 2003; McLean *et al*, 2003; Straßburger *et al*, 2003; Wang *et al*, 2004, 2006; Boer, 2005; Boer *et al*, 2006a, b, c; Lendermann, 2006; Straßburger, 2006a; Lendermann 2007; Zacharewicz *et al*, 2008; Liang *et al*, 2009; SISO-STD-006-2010; Taylor, 2011; Yuan and Zhang, 2011). These papers can be considered as introductory to the topic of DS in civil domain. Starting from these papers the bibliographic search followed the path of the citations, that is, works cited by the core papers and papers citing the core ones were considered. This search brought to the selection of 83 further papers. The overall 103 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*.

The collected papers addressing the HLA-based DS in civil applications have been analysed and classified according to the following criteria:

1. *Specific field of application* of DS in the civil domain (eg, Supply Chain Management (SCM), Health Care).
2. *Motivation that justified the use of DS technique*.
3. *Technical issue that is addressed in the paper*, proposing a solution to an integration issue or enhancement to services of the HLA architecture components.

Most of the papers can be classified according to more than one criterion. In particular, over 70% of the papers 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 civil application problems. In addition, more than 60% of the papers propose the application of HLA-based DS in a specific field in the civil domain (eg, Rabe *et al*, 2006; Kubat *et al*, 2009).

Figure 1 shows which are the main fields of application of DS: SCM (eg, Terzi and Cavalieri, 2004; Gan *et al*, 2007), Manufacturing (eg, Taylor *et al*, 2005), Health Care (eg, Mustafee *et al*, 2009) and Production Scheduling and Maintenance (eg, Uygun, 2006). Figure 1 clearly shows that much attention is paid to SCM (Rabe *et al*, 2006) and Manufacturing, whereas Health Care is acquiring larger interest in the recent years.

A further analysis was carried out by considering only the papers related to the manufacturing domain, aiming at evaluating whether the contributions addressed real industrial case applications or test cases applications. This analysis was performed to have a measure of how the solutions proposed within the scientific community have then been implemented over real cases.

The results show that only 22% of the papers address a real case. This result confirms the outcomes obtained by Boer (2005) in the analysis of the adoption of DS in the manufacturing environment. Although solutions have been developed for manufacturing domains, this technique is still far from being adopted as an evaluation tool in industrial companies because the end-users perceive HLA and DS as an additional trouble rather than a promising approach (Boer *et al*, 2006 c). 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 (Raab *et al*, 2011).

Analyzing the contributor under the second criterion, the main motivations leading to adopt an HLA-based DS for civil applications were:

1. *Complexity*: The DS is used because a single developer is not able to reproduce all the dynamics of the system that consists of several complex subsystems.
2. *Re-use*: The DS is proposed to integrate pre-existing simulators.
3. *Industrial secrecy*: This motivation is similar to *complexity*, but in this case the main problem for realizing a unique simulation model consists in the *lack of shared information*.

Complexity is clearly the main reason for the use of DS technique, as highlighted in McLean and Riddick, 2000; Linn and Chen, 2002; Gan *et al*, 2007; Kubat and Uygun, 2007. On the other hand, the low percentage ($\approx 5.6\%$ of all papers analysed) of papers using DS to cope with industrial secrecy can be partially traced back to the lack of real industrial applications that still characterizes DS in civil environment.

Concerning the third criterion two main technical issues were identified analyzing the collected papers:

1. *Integration of discrete event simulators (CSP)*: Several discrete event simulators are put together and synchronized by means of the services offered by the HLA infrastructure.
2. *Enhancements to RTI services* (please refer to Section 4.1 for more details on HLA integration infrastructure).

The integration of CSPs is the most addressed technical issue, nonetheless the integration of real CSPs (ie, not general purpose programming languages) still represents a challenging topic. Figure 2 gives a picture of the main simulators that have been adopted (eg, Park *et al*, 2005; Gan *et al*, 2005; Wang *et al*, 2007). It can be noticed that CSP Emulators (eg, Wang *et al*, 2006; Pedrielli *et al*, 2011) 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 possibility to interact (eg, send and receive information, share the internal event list), while the simulation is running.

The enhancement of the RTI services is another key research topic (Fujimoto *et al*, 2007; Al-Zoubi and Wainer, 2009). In particular, the scientific papers deal mainly with two open issues: (1) Time management

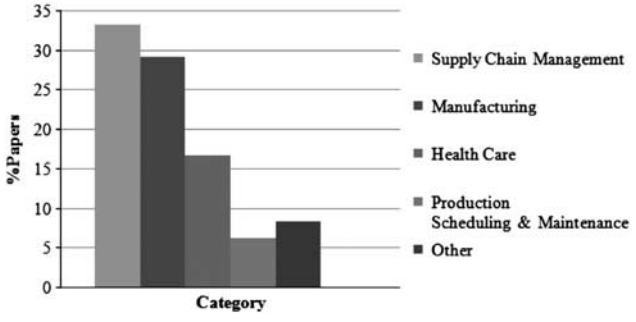

Figure 1 Domains of application.

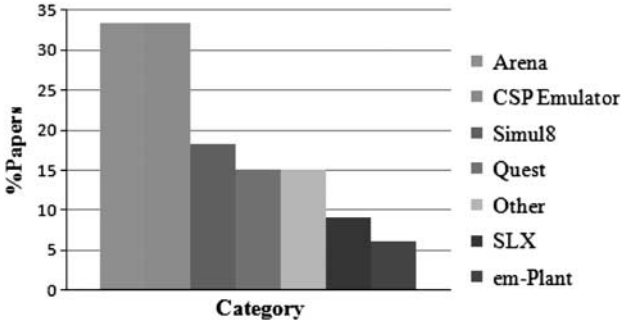

Figure 2 CSP adopted.

(eg, Peschlow and Martini, 2007; Malik, 2010); (2) Data distribution management (eg, Wainer *et al*, 2010; Yuan and Zhang, 2011). Time Management has received more attention because it strongly influences the computational performance of the DS.

The following conclusions can be drawn from the literature analysis, providing also the motivations of this paper:

- There is a lack of DS applications in the real manufacturing environment.
- The interoperability of CSPs still represents a technical challenging problem.
- The HLA architecture components (RTI services) must be extended and adapted to civil applications.

3. Problem statement

The analysis of networked manufacturing systems by means of distributed discrete event simulation is addressed by developing distinct simulators, each representing a single sub-system, and connecting them according to the relations characterizing the network. The work presented in this paper is aimed at supporting the design and analysis tasks (see Section 1) for the following classes of manufacturing systems (Colledani and Tolio, 2005):

- Assembly/disassembly systems, where assembly (disassembly) machines take different components (part) from one or more input buffer(s) and produce a single part (parts of different types), which is (are) placed in a downstream buffer (different output buffers).
- Split/merge systems, where different part types are managed. A split machine receives all parts from a single upstream buffer and then places the processed parts in different downstream buffers according to the part type and the adopted policy. A merging machine receives the parts of different type from more than one upstream buffer, but places all the processed parts in a single downstream buffer. The split/merge systems differ from the assembly/disassembly ones because parts cannot be modelled as components of the same product. For further details please refer to Colledani and Tolio (2005).
- Closed loop systems, where the last machine of the system is linked with the first machine and the number of parts in the system remains constant.
- General production lines with dedicated (flexible) machines performing only one or more operations.

The simulation of these classes of manufacturing systems via a distributed approach is strongly related to the representation of the part and information flow between the subsystems, thus rising the need to formalize this kind of transfer.

In literature (Taylor *et al*, 2004), the part transfer has been formalized through the definition of the *entity passing* problem where the term *entity* refers to elements that are dynamically created and moved during a simulation (Taylor *et al*, 2006b). The main result in the formalization of entity passing problem was presented by CSPI-PDG with the definition of Type A IRM, namely Entity Transfer. Figure 6 outlines the basic idea behind both Type A.1 and Type A.2 IRMs. The manufacturing system is decomposed into subsystems consisting of workstations and buffers, and each subsystem is associated with a different simulation model. M_i represents the model of the i-th production subsystem and En_{ij}/Ex_{ij} represent the j-th entry/exit point in M_i. Q_{ik} is associated with the k-th buffer in M_i, W_{ih} stands for the h-th workstation in M_i, whereas the arrows represent the flow of entities. In Figure 3, an entity can be transferred from workstation W_{1a} to buffer Q_{2a}.

Type A.1 IRM (named 'General Entity Transfer') is defined as the transfer of entities from one model to another, that is, an entity leaves from a given place in a sending model (eg, M_1) at time T_1 and arrives at a given place in a receiving model (eg, M_2) at time T_2 ($T_1 \leqslant T_2$). The departure and arrival places can be buffers, workstations and so on (eg, W_{1a} is a departure place and Q_{2a} is an arrival place). According to Type A.1 IRM the entity transfer is always feasible, thus no authorization is required before passing an entity from a model to another. The entity transfer represented in Figure 3 can be of Type A.1 IRM only if the capacity of Q_{2a} is unbounded.

Type A.2 IRM (named 'Bounded Receiving Element') is defined as the relation between a generic element in the sending model and a bounded element in receiving model, so that the feasibility of the entity transfer depends on the state of the receiving element. For instance, an entity transfer is blocked when the queue of the receiving element is full, even if the entity is ready to leave at time T_1 and would attempt to arrive at the bounded element at time T_2. Therefore, the information about the target element state is needed by the sending model, since it could stop the entity transfer. The entity transfer represented in Figure 3 is of Type A.2 IRM if the capacity of Q_{2a} is bounded.

Type A.3 IRM (named 'Multiple Input Prioritization') represents the case where an element of the receiving model can receive entities from more than one sending model. A problem arises if entities coming from different places arrive at the same time (ie, there are simultaneous events)

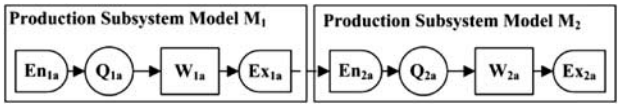

Figure 3 Entity transfer.

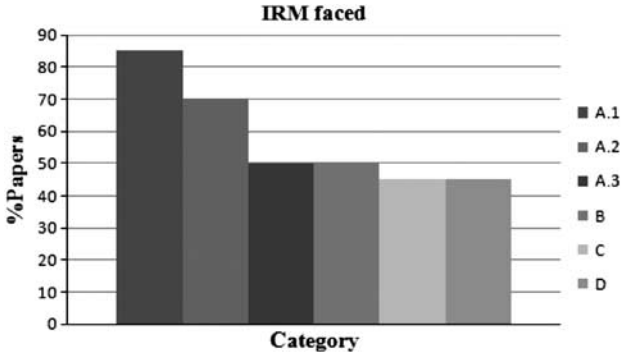

Figure 4 IRMs distribution.

and the receiving element is not able to receive all of them (Wang *et al*, 2006; Peschlow and Martini, 2007).

The literature review showed that around 21% of the papers taken into consideration dealt with IRMs. Figure 4 shows how the IRMs addressed are distributed within the sample of papers facing them (Taylor *et al*, 2004, 2006a, b, 2008, 2009; Peschlow and Martini, 2007).

The largest part of the papers deal with the basic IRM, that is, the general entity transfer, since most of the applications is related to SCM (ie, queues can often be modelled as infinite capacity as they represent inventory, production or distribution centres).

The situation is slightly different if the manufacturing domain is analysed. Indeed, IRM Type A.2 is largely adopted in the case of manufacturing applications. The reason is that, when a production system is modelled, the decoupling buffers between workstations must be usually represented as finite capacity queues.

Merge systems (see Figure 7), can be modelled using Type A IRM. In particular, the receiving model will start only when the batch of components is available, this can be done defining the entity in the receiving model as a batch. The communication can be represented by Type A.1 and A.2 based on the receiving model buffer capacity being infinite or finite. The case of split systems (see Figure 6) can be modelled by IRMs Type A.1 and A.2 depending on the receiving model queue capacity (finite or infinite).

In addition, in many applications, it is necessary to deal with management policies (eg, shared resources assignment, simultaneous events and entity priority) requiring a solution to IRM Type A.3 and IRM Type B. However, the management policies still represent a challenge both in terms of problem formalization and proposed solutions (Taylor, 2011).

These results motivate the research into the field of the formalization of the interoperability issues in manufacturing domain: the definition of interoperability issues is still an ongoing research topic and the standardization of a solution to these problems is far from being reached.

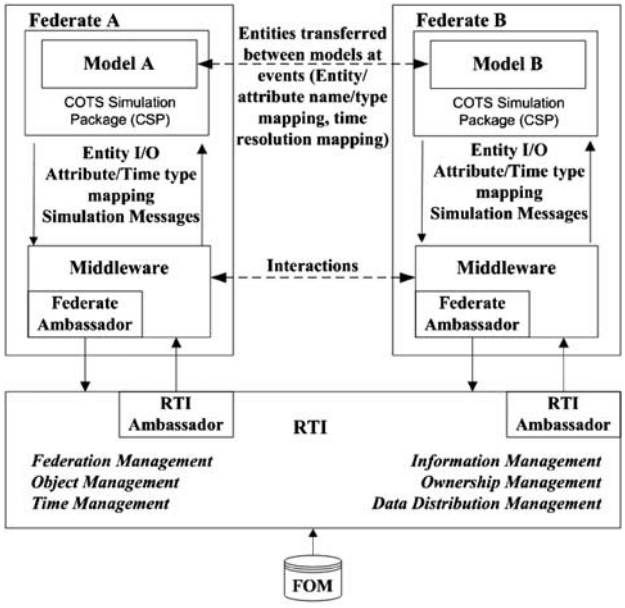

Figure 5 Reference architecture.

4. The problem of entity transfer

Past research contributions presented solutions to face both Type A.1 and Type A.2 IRMs (Section 3). In particular, a reference architecture (Figure 5) for the DS (Taylor *et al*, 2006b) and a protocol to manage the communication between the architecture components (Taylor *et al*, 2006a, b; Straßburger, 2006a) were proposed.

Section 4.1 presents the reference architecture adopted in this work and highlights the open issues, whereas Section 4.2 introduces the communication protocols and highlights their major drawbacks.

4.1 Reference architecture

The general architecture shown in Figure 5 is taken as a reference throughout this work and it is mainly based on architecture proposed by Taylor *et al* (2006b). A detailed description of the architecture components can be found in Taylor *et al* (2006a, b). Each federate consists of a COTS CSP, a model that is executed by the CSP, and the middleware that is a sort of adaptor interfacing the CSP with the RTI. 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 in 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.

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 (ie, the definition of time implies zero transit time) (Taylor *et al*, 2006a). Alternatively, it should be defined including the notion of travel time, in this case the entity would arrive to 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 modellers 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, whereas others use 64-bit (Taylor *et al*, 2006a).

Since the aforementioned issues are related to the adopted CSP and the decisions taken by the modelers, two simplifying hypotheses have been made in this paper: all the models have the same time definition and resolution (ie, there is a relationship between how time is represented in one CSP and another) and a mapping relationship exists between the entity representations in the various models. This paper addresses the transfer mechanism that is used to move an entity from one model to another thanks to the communications between middleware and RTI, and between CSP and middleware.

CSPI-PDG proposed the ETS Protocol (Taylor *et al*, 2006a, b) to manage communication at middleware-RTI level (see Section 'ETS protocol'). In this paper, a communication protocol based on Simulation Messages is proposed to manage the communication between a CSP and its middleware (or adapter). The communication protocol was conceived for the DS of network of Discrete Event Manufacturing Systems characterized by the transfer of parts in the presence of buffers with finite capacity. The presence of Simulation Messages is the main difference between the reference architecture in Figure 5 and the architecture proposed in Taylor *et al* (2006a).

4.2 ETS protocol

The ETS protocol proposed by CSPI-PDG defines the communication between the sending model and the receiving model (*ModelA* and *ModelB* in Figure 5, respectively) at RTI level 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 in common within 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.

After developing the interaction class hierarchy, following the HLA standard, the Simulation Object Model 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 to include the novel interaction classes and define them as publish and/or subscribe. The Parameter Table was modified to include the proposed parameters for the interactions and the Datatype table was also modified. For further details please refer to (Taylor *et al*, 2006b).

Straßburger (2006a, b) 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 (eg, Q_{2a} and Q_{2b} in Figure 6). The industrial cases defined in Section 3 can be modelled only if the first drawback of the list is properly addressed: the simulation of a manufacturing system in a distributed way may ask for the representation of multiple connections between the models, thus requiring multiple entry points in a receiving model (eg, *Model 2* in Figure 6) and/or multiple exit points in a sending model (eg, *Model 1* in Figure 7). The ETS draft standard should be modified to manage Type A.2, IRM as well.

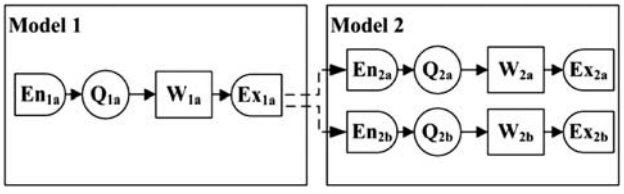

Figure 6 Multiple part type production system (eg, disassembly or split system)—*Case I*.

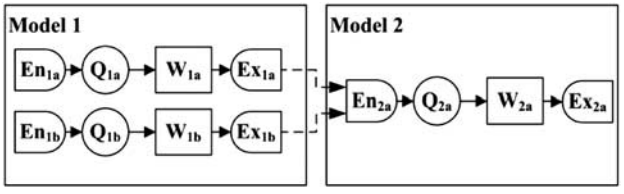

Figure 7 Multiple part type production system (eg, assembly or merge systems)—*Case II*.

5. A solution proposal for the type A.2 IRM

This section presents the solutions proposed to cope with the problems related to the Type A.2 IRM, as highlighted in Section 3. In particular, Section 5.1 proposes a modification to ETS, whereas Section 5.2 describes how information is sent (received) between the CSP and its middleware. Section 5.3 shows the new protocol aimed at managing the communication between a CSP and its middleware. Finally, the hypothesis underlying this protocol to minimize the use of *zero lookahead* is shown in Section 5.4.

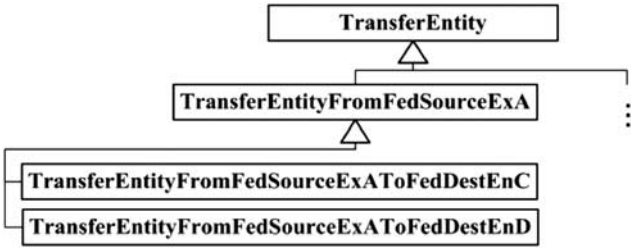

Figure 8 Interaction class hierarchy.

5.1 Proposal to modify ETS

The ETS standard proposal (Taylor *et al*, 2006a, 2006b) is modified by defining a new class hierarchy. In particular, different subclasses of the *transferEntity* class are defined to face the drawbacks of the ETS Protocol (Section 4.2). The resulting class hierarchy consists of the following classes (Figure 8):

- *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 *transferEntityFromFedSourceExToFedDestEn*. 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 (Taylor *et al*, 2006a b), respectively.
- *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).
- *EntityTransferTime*: It defines the simulation time when the entity is transferred to the destination point, that is the arrival time. In this work, 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 (Section 4.1).

5.2 Simulation messages

Simulation Messages are designed to support the communication between a CSP and its middleware (Section 4.1), that is, not on the HLA side. The function of the communication protocol depends on the role played by the

federate. The sending federate uses the protocol 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 protocol 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.

5.3 Communication protocol

This section presents the communication protocol between federates, whereas in Section 5.4 will define the hypotheses needed to minimize the *zero lookahead* when applying the proposed protocol.

Herein, the behaviour of the sending federate will be analysed 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* (Section 5.2) is scheduled, that is, 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 5.1).

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, that is, 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*.

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 (ie, 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* (Taylor *et al*, 2006b), whenever it is not possible to determine an advisable *a-priori* lookahead for the federation (Section 5.4). 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 3, (2) the buffer Q_{2a} at time t accommodates a number of parts, that is, greater than zero and less than the buffer capacity and an entity enters workstation W_{1a}, (3) a departure event from workstation W_{1a} is scheduled for time $t' = t + p$, where p represents the processing time of the leaving entity at station W_{1a}, (4) during the time interval (t, t'), no event happening in the federate M_2 (local event) influences the state of the buffer Q_{2a}.

Since W_{1a} is the last machine in model M_1, 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.

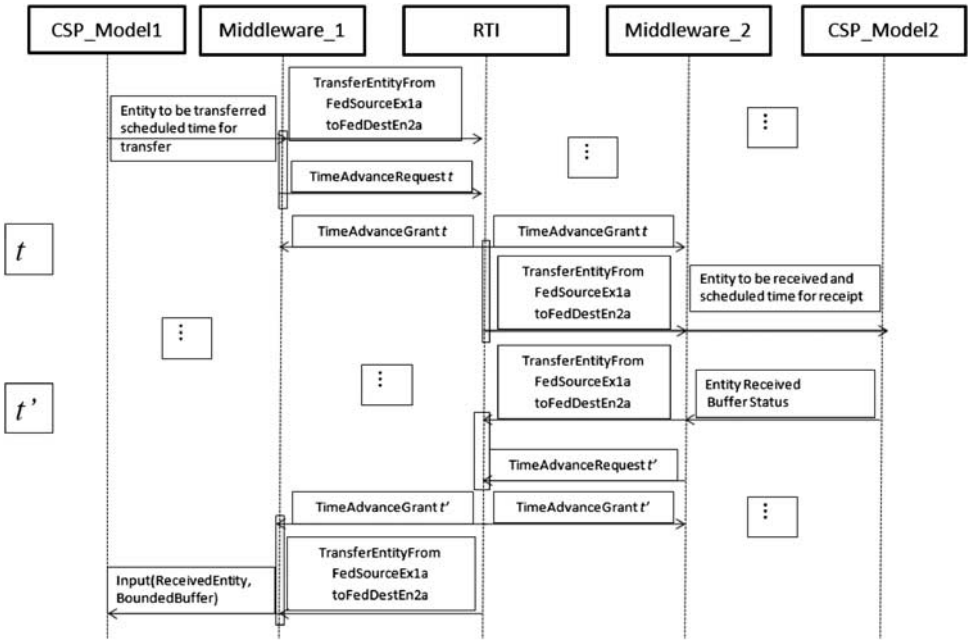

Figure 9 Communication protocol.

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 *External-ArrivalEvent*. 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 Q_{2a} is not full according to the example settings). A message is sent to the middleware of the receiving model containing the updated level of Q_{2a} (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 Q_{2a} 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 Q_{2a} and the received entity by means of the RTI. Since the entity has been delivered to the receiving model, the station W_{1a} is not blocked by the middleware.

5.4 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 $E_j^i(t, t')$ 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 has been addressed in Section 5.3: when the sending federate (model M_1) wants to transfer a part to the receiving federate (model M_2), 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 (W_{1a} 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*.

In this paper 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 the future developments of this work (Section 9).
- 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 federates 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 5.3 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 by the authors (Section 9).

6. HLA-based infrastructure implementation

The HLA-based architecture shown in Figure 5 was implemented as follows:

- MAK-RTI 3.3.2 (http://www.mak.com) was used as the RTI component implementation.
- The middleware was developed in C++ language following the specifications defined in A Solution Proposal for Type A.2 IRM section and was named *SimulationMiddleware*.
- The simulation models were developed using a CSP emulator, thus following the approach suggested in Wang *et al* (2006).

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 (see Section 'Proposal to modify ETS) and the Simulation Messages (see Section 'Simulation messages'). 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.

7. Manufacturing production system: a test case

The experiments presented in this section were designed to test the proposed entity passing solution (Section 5.1). The test case refers to a factory that produces two part types (*Part_A* and *Part_B*) and consists of two separated manufacturing systems (*Plant1* and *Plant2*). *Plant1* executes a set of manufacturing operations on both part types,

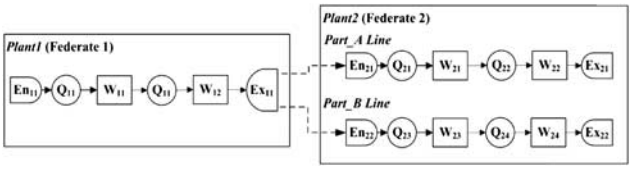

Figure 10 Factory test case.

whereas *Plant2* is divided into two production lines and each line is dedicated to process a single part type. If the performance of *Plant1* and *Plant2* is simulated via two separated simulators, then the whole production system can be simulated in a distributed way by representing *Plant1* and *Plant2* as federates within a federation (see Figure 10). This test case is characterized by Type A.2 IRM, because the workstation W_{12} has to transfer parts of *type A* (*type B*) to the bounded buffer Q_{21} (Q_{23}) in the *Part_A Line* (*Part_B Line*) of *Plant2*. For validation purposes the results from the DS were compared with a monolithic reference implementation. Two simulators representing *Plant1* and *Plant2* were developed using the CSP emulator mentioned in Section 6. These simulators were integrated by means of the HLA-based infrastructure (*DS approach*). Furthermore, a monolithic simulator was built using the CSP emulator to represent the overall factory (*SA approach*). Finally, a monolithic simulator was built using Rockwell Arena® 12 to validate the CSP emulator (results of this validation are out of scope, thus they are not reported). It is assumed that all simulation models do not contain any stochastic element. The experiments were designed as follows:

- Parts of type *A* arrive at *Plant1* every two time units, whereas parts of type *B* every time unit.
- The capacity of Q_{11} was set to a value large enough to guarantee that it never becomes completely full.
- Three conditions of maximum capacity (ie, 1, 5 and 50) were considered for the other buffers.
- Three conditions of processing times were considered for the workstations. Equal to 1 for all the workstations in the first condition, then to 7 for workstation W_{11} and to 1 for all the other workstations in the second condition, and finally to 7 for workstations W_{21} and W_{23} and to 1 for all the other workstations in the third condition.

Nine experimental conditions were obtained by combining the buffer capacity and processing time conditions. For each experimental condition the performance of the manufacturing system was simulated according to both the *SA* and the *DS* approach. A simulation length of 10 000 time units was set for all the experiments. A conservative synchronization approach was adopted: given the deterministic nature of the simulation, it was possible to evaluate the smallest interval elapsing between two consecutive events and the lookahead was set to that value (ie, 1 time unit) for all the experiments.

Table 1 Analysis of interactions

Type of interaction	*Total number of interactions*	
	Sending federate	*Receiving federate*
Time advance	9999	10 000
Entity passing (part type *A*)	715	1429
Entity passing (part type *B*)	1429	2857
Entity passing ($A + B$)	2144	4286

The experiments were run on a single machine Intel Core2 Duo Processor T7250, 2.00 GHz and 2046 MB-RAM. Both simulation approaches (*DS* and *SA*) returned identical simulation output values (not reported for reason of space) for each experiment, thus validating the interoperability offered by the proposed solution for Type A.2 IRM. However, the *DS* approach required a significantly higher (four times on average) computational time compared with the *SA* approach because of the overhead related to the services offered by RTI (Gan *et al*, 2000). The aforementioned result led to examine the number of interactions sent during the simulation experiments, thus delving into the overhead related to the RTI services.

Table 1 reports the number of interactions sent during the DS of the experiments characterized by the first condition of buffer capacity and the third condition of processing times. This experimental condition is the most critical in terms of the number of interactions needed to synchronize the two federates because the two slowest workstations are placed in *Plant 2* and therefore buffers Q_{21} and Q_{23} are frequently full while workstation W_{12} is frequently blocked. As a consequence, many interactions are needed to communicate when the parts can be actually transferred according to the state of buffers Q_{21} and Q_{23}.

The receiving federate sends more entity passing interactions than the sending federate because the receiving federate sends an interaction every time an entity is received and/or the availability of the receiving buffer changes, whereas the sending federate sends an interaction only when a departure event is scheduled. Each part arriving at the buffer, except the first that is directly assigned to the machine, causes two interactions: one to communicate the receipt of the entity and the updated buffer state, and another one to communicate the updated buffer availability when the entity leaves the buffer to enter the next workstation. A large number of time interactions is necessary because a request for time advance is generated at every time unit since the workstation W_{12} is almost always blocked and the time advances at the lookahead (conservative synchronization approach). This behaviour highlights how the solution proposed in Section 5 needs to be further improved.

8. Industrial case

This section aims at verifying if the proposed integration infrastructure presented in Section 5 can help to better evaluate the performance of complex manufacturing systems as described in Section 3. The goal does not consist in comparing the HLA-based DS with a monolithic simulation in terms of accuracy and computational efficiency as already done in the previous section. Herein, the attention is focused in the industrial field represented by sheet metal production, thanks to the collaboration offered by the company Tenova Pomini. In this industrial field, the production systems are characterized by the presence of at least two subsystems interacting with each other (Figure 11). The *Roll Milling System* produces sheet metal using milling rolls that are subject to wearing-out process; once the rolls wear out (ie, at the end of the *roll life*) there is the need to change them to avoid defects in the laminates. The *Roll Shop* performs the grinding process to recondition the worn out rolls. Tenova Pomini is a designer and provider of this kind of systems.

If the attention is focused on the rolls, then the resulting production system can be considered as a closed loop (Figure 11). The Roll Milling System sends batches of worn out rolls to the Roll Shop following a given policy and receives reconditioned rolls back. Both Roll Milling System and Roll Shop have finite capacity buffers. This implies that it is necessary to check whether the buffer in the system receiving the rolls has available slots. The deadlock in the closed loop is avoided because the number of rolls circulating in the system is less that the number of available slots (taking into account also the machines) and is constant.

The two subsystems forming a closed loop are strongly related and their reciprocal influences should be considered to properly evaluate the performance of the whole factory. However, the lack of shared information between the owner of the Milling System and the Roll Shop designer makes the realization of a monolithic simulation model hard to obtain or even infeasible. In particular, 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. Indeed, when a roll is worn out, the remaining life of the other rolls is checked and if the remaining life of a roll is under a predefined threshold, then it is sent to the grinding system together with the completely worn out rolls. The presence of a policy determines a relation between different roll types, since a roll can be sent to the grinding system depending on the behaviour of other roll types.

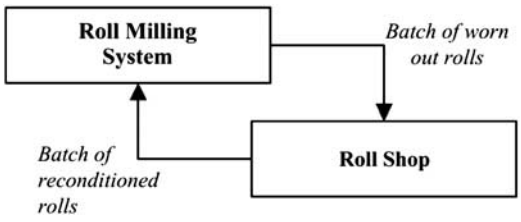

Figure 11 Industrial case representation.

When Tenova Pomini designs a Roll Shop, the owner of the Roll Milling System provides aggregated information about the yearly average demand of worn out rolls to be reconditioned. Then the Roll Shop designer develops a simulator for the roll grinding process with high level of detail.

The hypothesis is made that the Roll Shop designer has developed and validated a simulator using the CSP emulator (see Section 6). Similarly, the Roll Milling System owner has developed and validated a simulator using the CSP emulator, modelling the milling process and the roll changing policy with high level of detail; however, the model of the Roll Milling System simulator is not shared with the Roll Shop designer. The *Service Level* (*SL*) is the typical key performance indicator for evaluating the milling system and is defined as the ratio of the time during which the milling system is producing laminates over the total time when the milling system is available (ie, there is no failure). The *SL* would be reduced if the Roll Milling System had to wait for reconditioned rolls coming from the Roll Shop.

The Roll Shop designer has to evaluate the system performance 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).

The influence of the Roll Milling System can be represented by a mathematical model inside the detailed simulation model of the Roll Shop. This mathematical model roughly reproduces the Roll Milling System by generating the arrival of worn out rolls and accepting the reconditioned ones. The realization of a simulation model as described before will be referred to as *Approach A*. The main drawbacks of *Approach A* consist in:

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

These drawbacks lead to a potentially inaccurate evaluation of the factory performance. The Roll Shop designer could increase the level of detail of the mathematical model to improve the completeness of the simulation model and the

accuracy of the estimated *SL*. However, the lack of shared information hinders the feasibility of this enhancement. It must be stressed that simulator of Approach A cannot be considered as a proper monolithic simulator of the whole factory, since the Roll Milling System is only poorly modelled.

An alternative approach (*Approach B*) to evaluate the performance of the whole factory can be developed by adopting the proposed HLA-based Infrastructure to integrate the simulators of the Roll Milling System and of the Roll Shop. In this case, the mathematical model is removed from the simulation model of the Roll Shop, since the behaviour of the Roll Milling System is already modelled by its simulator. *Approach B* enables to evaluate the impact of the number of rolls on the system performance, so that the Roll Milling System owner can optimize the investment cost associated with the expensive rolls, whereas the Roll Shop designer can design a more effective and efficient Roll Shop thus better meeting the needs of the customer.

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

- Three experimental conditions are designed with reference to the total number of rolls circulating in the whole system. These three conditions are defined as *Low*, *Medium* and *High* level.
- The simulation run length was set to 6 months.
- 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 2.

Approach A and *Approach B* are compared in terms of the estimated *SL*. 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. 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.

On the basis of the analysis carried out so far, it was decided to design further experiments to analyse the behaviour of the system with different starting workload conditions, that is the number of rolls that are present in the Roll Milling System when the simulation starts. These experiments can be useful to analyse 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 modelled with *Approach A*. Indeed, the mathematical model generates rolls for the Roll Shop independently from the starting workload conditions. Therefore, the mathematical 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. This represents an additional criticality of the *Approach A* that can be solved only using *Approach B*.

Table 2 Service level results

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

The second set of experiments was designed as follows: (1) Two *types of roll* circulate in the factory (*RollType1* and *RollType2*). The roll of type *RollType2* has a longer roll life than *RollType1*; (2) For each type of roll three levels of the *Starting Workload* (ie, number of rolls) in the Roll Milling System are considered; (3) Three simulation run lengths are considered, that is, 1, 2 and 4 weeks; (4) The roll changing policy is fixed for all experiments; (5) the total number of rolls is equal to the *High* level of the previous experimentation and is fixed for all the experiments.

Figure 12 shows the main effects plot for the *SL* evaluated by simulating the 27 resulting experimental conditions with *Approach B*. The plot suggests a significant influence of the factor *Starting Workload for RollType1*. This roll type assumes a key role because of its short roll life. If the *Starting Workload For RollType1* is *Low*, the Roll Shop can hardly follow the frequent roll requests of *RollType1* from the Roll Milling System during the transient period and low values of *SL* are observed. This transient 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.

9. Conclusions and future works

The simulation of complex manufacturing systems led to the investigation of the use of DS technique in industrial environment. The first results of the literature review performed by the authors were presented. The sample of papers will be enlarged and further review will be performed to obtain a clear picture of DS in manufacturing. The need of simulating complex manufacturing systems led to investigate the integration of CSPs based on HLA and to propose a solution to the CSP interoperability problem by

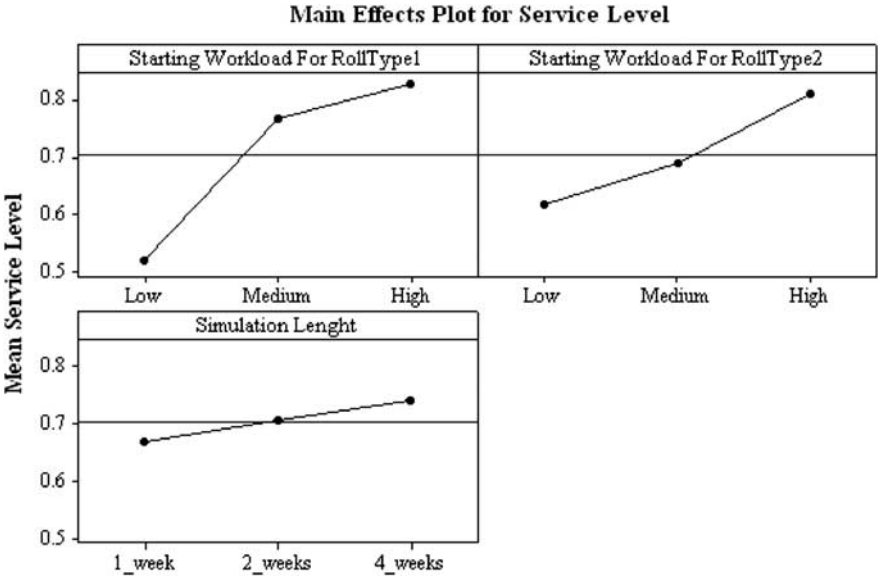

Figure 12 Main effect on *service level* with *Approach B*.

addressing the Type A.2 IRM and modifying the ETS protocol. In particular, Simulation Messages were designed to manage the communication between a CSP and its adaptor in case of Type A.2 IRM. Nonetheless, the implementation can be extended to the case of simultaneous events (Type A.3 IRM). The main hypotheses that have to be satisfied to minimize the *zero lookahead* have been described. Future work will be done to relax the hypothesis of the absence of additional events when the first simulation message has been sent. The experiments showed the feasibility of the use of the integrated simulators infrastructure. The last section showed the industrial benefits that can be reached by applying the HLA-based DS.

Further developments are needed to optimize the time advance management. The effect of different lookahead values should be investigated together with the use of optimistic synchronization approaches. The analysis of the number of interactions needed for the entity transfer (see Section 7) highlights that further research on the implementation of Type A.2 IRM is necessary as well. For instance, it would be interesting to investigate the design of 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).

Acknowledgements—The research reported in this paper has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No: NMP2 2010-228595, Virtual Factory Framework (VFF). The authors would like to thank Prof. S.J.E. Taylor for his on-going scientific support and Prof. S. Straßburger for the precious suggestions during the PADS workshop. Authors acknowledge Tenova Pomini for the definition of the industrial case.

References

Alvarado JR, Osuna RV and Tuokko R (2008). Distributed simulation in manufacturing using high level architecture. *International Federation for Information Processing* **260**(4): 121–126.

Al-Zoubi K and Wainer G (2009). Performing distributed simulation with restful web-services approach. In: Dunkin A, Ingalls R, Yücesan E and Rossetti M *et al* (eds). *Proceedings of the 2009 Winter Simulation Conference*, 13–16 December, Austin, TX, pp 1323–1334.

Banks J *et al* (2003). The future of simulation industry. In: Ferrin D, Morrice D, Sanchez P and Chick S (eds). *Proceedings of the 2003 Winter Simulation Conference*, 7–10 December, New Orleans, LA, pp 2033–2043.

Boer CA (2005). Distributed simulation in industry. PhD Thesis. Erasmus Research Institute of Management (ERIM), Erasmus University Rotterdam, the Netherlands.

Boer CA, De Bruin A and Verbraeck A (2006a). Distributed simulation in industry: A survey—Part 1 The COTS vendors. In: Nicol D, Fujimoto R, Lawson B and Liu J *et al* (eds). *Proceedings of the 2006 Winter Simulation Conference*, 3–6 December, Monterey, CA, pp 1094–1102.

Boer CA, De Bruin A and Verbraeck A (2006b). Distributed simulation in industry: A survey—Part 2 experts on distributed simulation. In: Nicol D, Fujimoto R, Lawson B and Liu J *et al* (eds). *Proceedings of the 2006 Winter Simulation Conference*, 3–6 December, Monterey, CA, pp 1061–1068.

Boer CA, De Bruin A and Verbraeck A (2006c). Distributed simulation in industry: A survey—Part 3 the HLA standard in industry. In: Nicol D, Fujimoto R, Lawson B and Liu J *et al* (eds). *Proceedings of the 2006 Winter Simulation Conference*, 3–6 December, Monterey, CA, pp 1061–1068.

Boer CA and Verbraeck A (2003). Distributed simulation and manufacturing: Distributed simulation with COTS simulation packages. In: Ferrin D, Morrice D, Sanchez P and Chick S (eds). *Proceedings of the 2003 Winter Simulation Conference*, 7–10 December, New Orleans, LA, pp 829–837.

Colledani M and Tolio T (2005). A decomposition method to support the configuration/reconfiguration of production systems. *CIRP Annals—Manufacturing Technology* **54**(1): 441–444.

Fujimoto RM (1998). Parallel and distributed simulation. In: Wiley (ed). *Handbook of Simulation*. University of Michigan, Wiley Online Library.

Fujimoto RM *et al* (2007). Ad Hoc distributed simulations. *Proceedings of the 21st International Workshop on Principles of Advanced and Distributed Simulation, IEEE Computer Society*, pp 15–24.

Gan BP *et al* (2000). Distributed supply chain simulation across enterprise boundaries. In: Fishwick P, Kang K, Joines J and Barton R (eds). *Proceedings of the 2000 Winter Simulation Conference*, 10–13 December, Orlando, FL, pp 1245–1251.

Gan BP *et al* (2005). Interoperating Autosched AP using the high level architecture. In: Armstrong F, Joines J, Steiger N and Kuhl N (eds). *Proceedings of the 2005 Winter Simulation Conference*, 4–7 December, Orlando, FL.

Gan BP *et al* (2007). Architecture and performance of an HLA-based distributed decision support system for a semiconductor supply chain. *SimTech Technical Reports* **7**(4): 220–226.

Granoweiter L and Watrous B (2004). *Challenges and Solutions for Large Scale HLA Federations*. Simulations Interoperamlity workshop, Cambridge, MA, pp 3–38.

Hibino H *et al* (2002). Manufacturing adapter of distributed simulation systems using HLA. In: Snowdon J, Charnes J, Yücesan E and Chen CH (eds). *Proceedings of the 2002 Winter Simulation Conference*, 8–11 December, San Diego, CA, pp 1099–1107.

Kubat C and Uygun O (2007). HLA based distributed simulation model for integrated maintenance and production scheduling system in textile industry. In: Pham PT, Eldukhri EE and Soroka A (eds). *Proceedings of 3rd I*PROMS Virtual International Conference*, 2–13 July, Cardiff, England, pp 413–418.

Kubat C, Uygun O and Ozmetel E (2009). Scenario based distributed manufacturing simulation using HLA technologies. *Information Sciences* **79**(10): 1533–1541.

Kuhl F, Dahmann J and Weatherly R (1999). *Creating Computer Simulation Systems: An introduction to the High Level Architecture*. Prentice Hall PTR: Upper Saddle River, NJ.

Law AM (2007). *Simulation modeling and Analysis* 3rd edn, McGraw-Hill: New York.

Lee YT, Leong S and Riddick F (2006). A core manufactur ing simulation data information model for manufacturing applications, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.161.6233. Manufacturing Systems Integration Division National Institute of Standards and Technology Gaithersburg, MD 20899-8260 USA. 301-975-5426, 301-975-3550, 301-975-3892.

Lendermann P (2006). About the need for distributed simulation technology for the resolution of real-world manufacturing and logistics problems. In: Nicol D, Fujimoto R, Lawson B and Liu J *et al* (eds). *Proceedings of the 2006 Winter Simulation Conference*, 3–6 December, Monterey, CA, pp 1119–1128.

Lendermann P (2007). Panel: Distributed simulation in industry—A real-world necessity or ivory tower fancy? In: Morse K, Perumalla K and Riley G (eds). *Proceedings of the 2007 Winter Simulation Conference*, 15–17 June, San Diego, CA, pp 1053–1062.

Liang Y, Cai W and Turner SJ (2009). *Interfacing RePast with HLA using a generic architecture for COTS simulation package interoperability*. Joint SISO/SCS Spring Simulation Interoperability Workshop, San Diego, CA.

Linn RL and Chen CS (2002). Development of distributed simulation model for the transporter entity in a supply chain process. In: Snowdon J, Charnes J, Yücesan E and Chen C-H (eds). *Proceedings of the 2002 Winter Simulation Conference*, 8–11 December, San Diego, CA, pp 1319–1326.

Malik AW (2010). An optimistic parallel simulation for cloud computing environments. *SCS M&S Magazine* **6**(4): 1–9.

MAK RTI. http://www.mak.com.

McLean C *et al* (2003). Simulation standards: Current status, needs, and future directions. In: Ferrin D, Morrice D, Sanchez P and Chick S (eds). *Proceedings of the 2003 Winter Simulation Conference*, 7–10 December, New Orleans, LA, pp 2019–2026.

McLean C and Riddick F (2000). Integration of manufacturing simulations using HLA. *Proceedings of the 2000 Advanced Simulation Technologies Conference (ASTC 2000)*, 14–17 October, Cleveland, OH.

Mustafee N and Taylor SJE (2006). Investigating distributed simulation with COTS simulation packages: Experiences with Simul8 and the HLA. *Proceedings of the 2006 Operational Research Society Simulation Workshop (SW06), Leamington Spa*, pp 33–42.

Mustafee N, Taylor SJE, Katsaliaki K and Brailsford S (2009). Facilitating the analysis of a UK national blood service supply chain using distributed simulation. *Simulation* **85**(2): 113–128.

Park J *et al* (2005). Addressing complexity using distributed simulation: A case study in spaceport modeling. In: Armstrong F, Joines J, Steiger N and Kuhl N (eds). *Proceedings of the 2005 Winter Simulation Conference*, 4–7 December, Orlando, FL.

Pedrielli G *et al* (2011). Simulation of complex manufacturing systems via HLA-based Infrastructure. *Proceedings of the 2011 Workshop on Principles of Advanced and Distributed Simulation, Nice*, 14–17 June, Nice, pp 78–89.

Peschlow P and Martini P (2007). Efficient analysis of simultaneous events in distributed simulation. *Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT '07)*, 22–24 October, Chania, Greece, pp 244–251.

Raab M, Masik S and Schulze T (2011). *Support system for distributed HLA simulations in industrial applications* Workshop on Parallel Advanced and Distributed Simulation 87–93.

Rabe M, Jaekel FW and Weinaug H (2006). Supply chain demonstrator based on federated models and HLA application. *Simulation und Visualisierung 2006*. European Publishing House: Ghent, Erlangen, San Diego, Delft, pp 329–338.

Sacco M *et al* (2011). VFF: Virtual Factory Manager. In: Shumaker Randall (ed) *Virtual and Mixed Reality—Systems and Application, Lecture Notes in Computer Science* Vol. 6774, Springer: Berlin Heidelberg, pp 397–406.

SISO CSPI-PDG. http://www.sisostds.org.

SISO COTS Simulation Package Interoperability Product Development Group (2010). Standard for Commercial-off-the-shelf Simulation Package Interoperability Reference Models. SISO-STD-006-2010.

Sly D and Shreekanth M (2001). Simulation Data Exchange (SDX) implementation and use. In: Peters BA, Smith JS, Medeiros DJ and Rohrer MW (eds). *Proceedings of the 2001 Winter Simulation Conference*, pp 1473–1477.

Smith JS (2003). Survey on the use of simulation for manufacturing system design and operation. *Journal of Manufacturing Systems* **22**(2): 157–171.

Straßburger S (2001). *Distributed simulation based on the high level architecture in civilian application domains*. PhD Thesis,

Computer Science Faculty, University Otto von Guericke, Magdeburg.

Straßburger S (2006a). The road to COTS-interoperability: from generic HLA-interfaces towards plug-And-play capabilities. In: Nicol D, Fujimoto R, Lawson B and Liu J *et al* (eds). *Proceedings of the 2006 Winter Simulation Conference*, 3–6 December, Monterey, CA.

Straßburger S (2006b). Overview about the high level architecture for modeling and simulation and recent developments. *Simulation News Europe* **16**(2): 5–14.

Straßburger S, Schmidgall G and Haasis S (2003). Distributed manufacturing simulation as an enabling technology for the digital factory. *Journal of Advanced Manufacturing System* **2**(1): 111–126.

Taylor SJE (2011). Realising parallel and distributed simulation in industry: A roadmap. *Proceedings of the 25th International Workshop on Principles of Advanced and Distributed Simulation*. IEEE Computer Society, 14–17 June, Nice, p 1.

Taylor SJE, Bohli L, Wang X and Turner SJ (2005). Investigating distributed simulation at the ford motor company. *Proceedings of the Ninth IEEE International Symposium on Distributed Simulation and Real-Time Applications*. IEEE Computer Society, pp 139–147.

Taylor SJE *et al* (2002). Distributed simulation in industry: Potentials and pitfalls. In: Snowdon J, Charnes J, Yücesan E and Chen C-H (eds). *Proceedings of the 2002 Winter Simulation Conference*, 8–11 December, San Diego, CA, pp 688–694.

Taylor SJE *et al* (2003). HLA-Cspif panel on commercial off-the-shelf distributed simulation. In: Ferrin D, Morrice D, Sanchez P and Chick S (eds). *Proceedings of the 2003 Winter Simulation Conference*, 7–10 December, New Orleans, LA, pp 881–887.

Taylor SJE, Turner S and Low M (2004). A proposal for an entity transfer specification standard for COTS simulation package interoperation *Proceedings of the 2004 European Simulation Interoperability Workshop*. 28 June–1 July, Edinburgh, Scotland.

Taylor S *et al* (2006a). Developing interoperability standards for distributed simulation and COTS simulation packages with the CSPI PDG. In: Perrone LF *et al* (eds). *Proceedings of the 2006 Winter Simulation Conference*. Institute of Electrical and Electronics Engineers: Piscataway, NJ, pp 1001–1110.

Taylor SJE, Wang X, Turner SJ and Low MYH (2006b). Integrating heterogeneous distributed COTS discrete-event simulation packages: An emerging standards-based approach. *IEEE Transactions on Systems, Man & Cybernetics* **36**(1): 109–122.

Taylor SJE, Turner SJ and Straßburger S (2008). Guidelines for commercial off-the-shelf simulation package interoperability. In: Mason SJ *et al* (eds). *Proceedings of the 2008 Winter Simulation Conference*. Association for Computing Machinery Press: New York, NY, pp 193–204.

Taylor SJE *et al* (2009). Commercial-off-the-shelf simulation package interoperability: Issues and futures. In: Tew J, Barton R, Henderson J and Biller B *et al* (eds). *Proceedings of the 2009 Winter Simulation Conference*, 13–16 December, Washington, DC, USA.

Terkaj W, Tolio T and Valente A (2009). Designing manufacturing flexibility in dynamic production contexts. *Design of Flexible Production Systems Methodologies and Tools*. Springer: Berlin Heidelberg, pp 1–18.

Terzi S and Cavalieri S (2004). Simulation in the supply chain context: A survey. *Computers in Industry* **53**(1): 3–16.

Uygun Ö (2006). HLA-based distributed simulation model for integrated maintenance and production scheduling system in textile industry. *International Symposium on Intelligent Manufacturing Systems*.

Vàncza J, Egri P and Monostory L (2008). A coordination mechanism for rolling horizon planning in supply networks. *CIRP Annals—Manufacturing Technology* **57**(1): 455–458.

Wainer G *et al* (2010). Standardizing DEVS model representation. In: *Discrete Event Modeling and Simulation: Theory and Applications*. CRC Press.

Wang X, Turner SJ, Low MYH and Gan BP (2004). A Generic Architecture for the Integration of COTS Packages with the HLA. In: Robinson S and Taylor SJE (eds). *Proceedings of the 2004 Operational Research Society Simulation Workshop*. Association for Computing Machinery's—Special Interest Group for Simulation: Birmingham, UK, pp 225–233.

Wang X, Turner SJ and Taylor SJE (2006). COTS simulation package (CSP) interoperability—A solution to synchronous entity passing. *Proceedings of the 20th International Workshop on Principles of Advanced and Distributed Simulation*. IEEE Computer Society, pp 201–210.

Wang G, Chun J and Peng G (2007). Adapting arena into HLA: Approach and experiment [A]. *Proceedings of the IEEE International Conference on Automation and Logistics [C]*. Vol. 8, IEEE: Jinan, China, pp 1709–1713.

Wiendahl HP and Lutz S (2002). Production in networks. *CIRP Annals—Manufacturing Technology* **51**(2): 573–586.

Yuan Z and Zhang LM (2011). The viewable distributed simulation linkage development tool based on factory mechanism. *Applied Mechanics and Materials* **58**(60): 1813–1818.

Zacharewicz G *et al* (2008). G-DEVS/HLA environment for distributed simulations of workflows. *SIMULATION: Transactions of the Society for Modeling and Simulation International* **84**(5): 197–213.

1516-2000 IEEE Standard for M&S—HLA—Framework and rules. *IEEE Computer Society*, 2000, pp 1–22.

1516-2000 IEEE Standard for M&S—HLA—Federate Interface Specification. *IEEE Computer Society*, 2000, pp 1–29.

1516-2000 IEEE Standard for M&S—HLA—Object Model Template (OMT). *IEEE Computer Society*, 2000, pp 1–30.

1516-2003 IEEE Recommended Practice for HLA FEDEP. *IEEE Computer Society*, 2003, pp 1–79.

Received 22 September 2011;
accepted 9 February 2012 after one revision