Bridging The Gap: A Standards-Based Approach to OR/MS Distributed Simulation

SIMON J. E. TAYLOR, Brunel University
STEPHEN J. TURNER, Nanyang Technological University
STEFFEN STRASSBURGER, Ilmenau University of Technology
NAVONIL MUSTAFEE, Swansea University

In Operations Research and Management Science (OR/MS), Discrete Event Simulation (DES) models are typically created using commercial off-the-shelf simulation packages (CSPs) such as AnyLogic™, Arena™, Flexsim™, Simul8™, SLX™, Witness™, and so on. A DES model represents the processes associated with a system of interest. Some models may be composed of submodels running in their own CSPs on different computers linked together over a communications network via distributed simulation software. The creation of a distributed simulation with CSPs is still complex and typically requires a partnership of problem owners, modelers, CSP vendors, and distributed simulation specialists. In an attempt to simplify this development and foster discussion between modelers and technologists, the SISO-STD-006-2010 Standard for COTS Simulation Package Interoperability Reference Models has been developed. The standard makes it possible to capture interoperability capabilities and requirements at a DES modeling level rather than a computing technical level. For example, it allows requirements for entity transfer between models to be clearly specified in DES terms (e.g. the relationship between departure and arrival simulation times and input element (queue, workstation, etc.), buffering rules, and entity priority, instead of using specialist technical terminology. This article explores the motivations for distributed simulation in this area, related work, and the rationale for the standard. The four Types of Interoperability Reference Model described in the standard are discussed and presented (A. Entity Transfer, B. Shared Resource, C. Shared Event, and D. Shared Data Structure). Case studies in healthcare and manufacturing are given to demonstrate how the standard is used in practice.

1. INTRODUCTION

A computer simulation is used to conduct experiments with models that represent systems of interest to enable decision makers to make better informed decisions

[Pidd 2004]. One such simulation technique is Discrete-Event Simulation (DES) and it is frequently applied across a range of industries such as manufacturing, travel, finance, healthcare, and supply chains [Hollocks 2006]. In the context of Operations Research and Management Science (OR/MS) in general and operations management in particular, DES models are typically created using specialist Commercial Off The-Shelf (COTS) Simulation Package (CSP) software such as AnyLogic^{TM} (XJ Technologies), Arena^{TM} (Rockwell Automation), Flexsim^{TM} (Flexsim Software Products Inc.), Simul8^{TM} (Simul8 Corporation), SLX^{TM} (Wolverine Software), Witness^{TM} (Lanner Group), and so on. OR/MS literature suggests that most of these models tend to be single DES models that represent the processes associated with particular systems of interest. However, in cases where the underlying system is large and/or logically divided, the system might be conceptualized as several subsystems. These subsystems may belong to multiple functional areas of an organization (e.g., a manufacturing facility may have several well-defined subprocesses such as "assembly line" and "paint shop"), they may be related to several organizations (e.g., distinct subprocesses such as "procurement", "transport", and "distribution" owned by individual organizations that form a part of a supply chain [Gan et al. 2000]), or they may indeed be a combination of subsystems belonging not only to several organizations but also to several distinct functional areas within each of these organizations (e.g., an organization may have two subsystems associated with "manufacturing" and "repair and maintenance", and it may depend on the "transport" and "distribution" subprocesses belonging to other organizations for shipping of both new and returned goods). In such cases, creating an all-encompassing DES model may be difficult for reasons such as, for example, concerns among intra- and inter-organizational stakeholders with regard to data/information sharing, security, and privacy of shared data/information [Mertins et al. 2005; Li et al. 2010]. Furthermore, issues such as model composability, data transfer/access problems [Taylor et al. 2011], and execution speed [Mustafee et al. 2009] may also make a single model approach problematic.

The alternative is to build separate DES models (or reuse existing models) representing the various intra- and/or inter-organizational subsystems that are constituent parts of the larger system. These separate models can be linked together over a computer network such as the Internet, using specialist networking software to create a distributed simulation of the system of interest. Although this approach of executing multiple DES models permits data-hiding and enables the simulation of the overall system, there are considerable technological barriers in implementing this solution; not least of which is the clear specification of the interoperability requirements between each distributed model. This is derived from the lack of common ground between distributed simulation developers and OR/MS simulation practitioners. It is extremely difficult for an OR/MS simulation practitioner to express how models should interoperate in technical terms. The opposite holds true as well; it is unlikely that a distributed simulation developer will fully understand model interoperability in valid OR/MS terms.

In an attempt to bridge this gap, this article reports on the outcome of an international standardization effort, the *SISO-STD-006-2010 Standard for COTS Simulation Package Interoperability References Models* (IRMs) [SISO 2010a]. The contribution this article is an approach to facilitating the capture of interoperability requirements at a modeling level rather than a technical level and enables OR/MS modelers and vendors to properly specify the interoperability requirements of a distributed simulation in their terms. This is the first time interoperability requirements have been specified (and standardized) from the point of view of OR/MS simulation. This is a clear benefit to this simulation community, as there is no other standardized mechanism that does this and will facilitate the development of distributed simulations in this area.

The rest of the article is organized as follows. The next section presents a discussion on distributed simulation in the context of OR/MS simulation practice. This is followed by an overview of related work. The standardized guidelines for CSP-based distributed simulation are then introduced. Two case studies that have applied the SISO-STD-006-2010 guidelines are presented; the first case study is on a healthcare, supply chain and the second case study relates to a manufacturing system. A conclusion then summarizes the article and outlines future work.

2. DISTRIBUTED SIMULATION AND OR/MS SIMULATION PRACTICE

Distributed simulation can be defined as the distribution of the execution of a simulation program across multiple processors [Fujimoto 2000]. Distributed simulation software (sometimes called *middleware*) is quite complex and implements well-known distributed simulation time management algorithms to achieve synchronization between individual running simulations [Fujimoto 1990]. The current standard to support this is IEEE 1516 High Level Architecture (HLA) [IEEE 2010] (first released in 2000 and updated in 2010). This came from the need of the US Department of Defense to make modeling and simulation cost-effective through reuse of computer simulations and to improve interoperability between its sometimes very heterogeneous simulation systems. In HLA terminology, a distributed simulation is called a *federation*, and each individual simulator is referred to as a *federate*. The HLA Federate Interface Specification (Federate I/F Spec) defines the various services provided by the *Runtime Infrastructure* (RTI). A distributed simulation is therefore a federation composed of many federates interacting over a communication network via RTI software [Fujimoto and Weatherly 1996]. The first reference implementation of the RTI was the DMSO RTI [US Department of Defense 1999] (now no longer available). There are several RTIs presently available, including Pitch pRTI [Karlsson and Olsson 2001], the Service-Oriented HLA RTI (SOHR) [Pan et al. 2008], the CERTI Open Source RTI [Noulard et al. 2009] and the Portico open source RTI [Malinga and Le Roux 2009]. There have been many significant advances in distributed simulation as applied to military problems (e.g. the Millennium Challenge 2000 and 2002 [GlobalSecurity.org 2002] and the Joint Live Virtual Constructive Architecture [Henninger et al. 2008]).

With regard to a CSP-based distributed simulation (which is the focus of this article), the combination of a CSP and its model is a federate. A *CSP-based federation* is comprised of multiple CSP-based federates that interact with the RTI using the services defined by the Federate I/F Spec. Thus, all communication between the CSP-based federates (e.g., transfer of finished goods from the production line to the warehouse) are represented as messages that are exchanged between them via the RTI over a communication network.

To illustrate how distributed simulation might benefit OR/MS simulation practice, consider a system that may be composed of several well-defined intraorganizational subsystems (e.g. a multinational organization that operates in several countries), or indeed, subsystems owned by different organizations (e.g. an interorganizational supply chain). Here there may be concerns regarding information security since each functional entity may not wish to reveal its data and internal processes to the other functional entities that it works with (in the remainder of this section, we refer to these functional entities as stakeholders; the stakeholders can belong to either the same organization or to different organizations). If this system, as a whole, was represented as a single model then these secrets would be revealed, as they would have to be explicitly specified in the model. In addition to privacy, further problems include the following.

—*Data transfer/access problems*. Stakeholders may be open to each other (happy to share data and internal processes). In such cases, although a single model may reside on a single computer in a particular place (say, the place associated with stakeholder X), that model will still need data drawn from processes associated with the other stakeholders. However, databases can be large and time consuming to copy. Also, arguably, data when copied is instantly out of date. Running a model using copies of stakeholders' data can therefore be time consuming and inaccurate.

—*Model composability problems*. Even if each of the stakeholders had previously developed models using the same CSP, these models cannot simply be cut and pasted into the same single model. Variable name clashes, global variables, and different validation assumptions are three examples of the many problems of this approach. Further, if a stakeholder needs to update its model, it has to update the single model. How do we make sure that every stakeholder has the correct version of the single model? What if the update causes problems in another part of the single model owned by another stakeholder? Additionally, models developed in different CSPs are usually not compatible. One cannot transfer a model developed in one CSP into another without significant effort.

—*Execution time*. Large models will most likely develop large event lists that must be processed and updated each time an event is executed. This can take a considerable amount of time. Worse, the processing capacity of even a high specification PC may not be enough to physically cope, as the actual CSP may have an upper limit on the event list size.

In these cases, an alternative approach is needed. Here we create separate DES models in separate CSPs for processes representative of each stakeholder. Linking the models together over a network such as the Internet, using distributed simulation software creates a distributed simulation. This allows the models to be executed separately and privately by the stakeholders to simulate both intra- and inter- organization-specific processes, while accessing local data and avoiding many model composability issues.

The key benefit of distributed simulation is the creation of large, distributed models that are private, access local up-to-date data, implement local changes efficiently, and share the processing load of the model across the computers of the organizations. The modular nature of the individual models also means that these can be potentially plugged into different distributed models that an organization might be part of as required [Lendermann 2006; Boer et al. 2009]. However, there are significant barriers to this.

There are two predominant barriers to implementing a distributed simulation solution in OR/MS. The first is that the present generation of CSPs is not capable of executing distributed models directly, as these are set up for developing single models. Thus, the CSPs have to be interfaced with existing distributed simulation software by potentially costly software experts (a technical solution not covered by a CSP license fee). The second is that there is a very steep learning curve associated with implementing a CSP-HLA (or other distributed simulation technology) integrated solution because it requires good familiarity with distributed simulation theory and software. Contemporary simulation vendors and consultancies rarely have this knowledge. Successful distributed simulations of this kind have been created via collaborations between OR/MS simulationists, CSP vendors, and distributed simulation specialists. These factors may be a reason why distributed simulation is widely and successfully used in the military but not in industry [Strassburger et al. 2008]. Several research groups have developed technologies specifically for this area. However, these often take different, incompatible approaches. The next section reviews these attempts and related areas.

3. RELATED WORK

There has been much work that has investigated distributed simulation in OR/MS, primarily motivated by the need for privacy across supply chains (a factor commonly cited in the papers in this review). Researchers have used different technologies to support distributed simulation. For example, the Common Object Request Broker Architecture (CORBA) [Zeigler et al. 1999], the Message Passing Interface using an Asynchronous Simulation Protocol (MPI-ASP) [Gan et al. 2000; Gan and Turner 2000], and the Generic Runtime Infrastructure for Distributed Simulation for Supply Chain Federation (GRIDS-SCF) [Taylor et al. 2002], the Distributed Component Object Model (DCOM) [Santos et al. 2008], Inter-Process Communication (IPC) [Bandinelli et al. 2006] and Web Services [Lee et al. 2008]. The drawback with these approaches is that they are highly individualized and are typically implemented by a research team and not developed for widespread sustainable commercial use. The existence of a common standard is, however, a major factor in developing a commonly available approach, as there is typically much documentation for the standard and its technologies and a very large community who frequently use it. Rather than using low-level technologies and standards that have been developed for other purposes, other researchers have developed approaches based on the HLA.

The problem of creating distributed simulations consisting of CSPs using the HLA was first addressed in Strassburger et al. [1998]. Individual research projects developed different, but incompatible approaches to the use of the HLA supporting distributed simulation with specific CSPs. These included AnyLogic^{TM} [Borshchev et al. 2002], AutoSched^{TM} [Gan et al. 2005; Lendermann et al. 2007], Witness^{TM} [Taylor et al. 2005a], SIMUL8^{TM} [Mustafee and Taylor 2006]; and simulation languages MODSIM III^{TM} [Johnson 1999], DEVS [Al-Zoubi and Wainer 2008] and SLX^{TM} [Strassburger et al. 2007]. A problem with these approaches is that they are also highly specific to the CSPs that they use. Attempts to generalize approaches have developed CSP adaptor software that can be arguably connected to any CSP. Research using this approach includes the supply chain work carried out in the MISSION Project [Rabe et al. 2006], experiments with QUEST^{TM}, SIMPLE++^{TM}, and GAROPS^{TM} [Hibino et al. 2002] and research into semiconductor supply chains [Lendermann et al. 2007]. Uygun et al. [2009] refine these approaches by attempting to generalize data models used to interface with the HLA. In earlier work, Gan et al. [2000] investigated general distributed simulation of supply chain issues and, following from this, Taylor et al. [2005b] investigated the performance of distributed simulation middleware (HLA RTI) over different supply chain topologies. Experiments were conducted by interfacing the middleware with a COTS Simulation Package Emulator [Mustafee 2004; Taylor et al. 2005b] intended to assist in the investigation of algorithmic approaches to distributed simulation in OR/MS. Jain et al. [2007] developed a distributed simulation of a supply chain based on the HLA, intended to support CSPs in testing interoperability protocols between organizational units.

While this research builds on a common standard, the approaches are all largely incompatible in terms of data exchange format, interoperability, and time management. In recent years attempts have been made to unify the approaches into a single standard based on the HLA standard [Taylor, et al. 2006]. Started in 2003, the CSP Interoperability Forum (CSPIF) held meetings at many simulation conferences around the world. The CSPIF had about 70 international members formed from a mix of distributed simulation researchers, simulation practitioners, and CSP vendors (and included leading members of the field). In 2005, discussions were held with the Simulation Interoperability Standards Organization (SISO) and the CSPIF was invited to submit a product nomination (the first step in becoming a recognized Product

Development Group with a standards development mandate). This resulted in the creation of the CSPI Product Development Group (CSPI PDG). The CSPI PDG continued work in aligning these CSP interoperability/distributed simulation approaches. The CSPI PDG standards (products) are intended to provide guidance on how specific requirements of HLA-based distributed simulation can be supported with CSPs. As part of this activity a set of four Interoperability Reference Models (IRMs) have been defined to create a common frame of reference to assess the capabilities of particular approaches and to help practitioners and vendors achieve solutions to complex interoperability problems [Taylor et al. 2007, 2008; SISO 2010a]. These were created by following SISO's Balloted Standards Development and Support Process (BPDSP), which involves several rounds of voting/balloting and refinement before a standard is formally recognized by SISO's Standards Activity Committee (SAC) and its Executive Committee (EXCOM). Examples of research based on IRMs include Wang et al. [2006], who studied possible implementations, Taylor et al. [2005a], who investigated the use of distributed simulation in engine manufacturing, Gan et al. [2005] and Lendermann et al. [2007], who investigated the use of distributed simulation in semiconductor manufacturing supply chains, and Mustafee et al. [2009], who applied distributed simulation to a healthcare supply chain. Many of these authors have participated in the development of the CSPI standards. Standardization of the adaptor approach is still ongoing but closely follows the technological discussion in the case study.

Another related standard is the Core Manufacturing Simulation Data (CMSD) Information Model (SISO-STD-008-2010) [SISO 2010b], which addresses the development of a common data exchange format in the area of manufacturing simulation. Although CMSD might to some extent also serve as a model exchange format for converting models between different CSPs, its main focus is on interoperability between simulation systems and other manufacturing IT systems (e.g., Manufacturing Execution Systems) [Strassburger and Taylor 2012]. CMSD is therefore not an alternative solution to linking CSPs using distributed simulation, rather it supports other aspects of simulation interoperability.

4. GUIDELINES FOR CSP-BASED DISTRIBUTED SIMULATION

State of the art CSPs are not easily used for distributed simulation. As presented earlier, this type of distributed simulation is typically developed by a partnership of problem owners, modelers, CSP, vendors and distributed simulation specialists. Modelers create the distributed simulation to the satisfaction of the problem owners. However, as CSPs do not really have direct support for distributed simulation (access to event lists, inclusion of external events, etc. [Strassburger et al. 1998]), modelers must express their needs to CSP vendors who then work with distributed simulation specialists who jointly develop the distributed simulation as needed. Often the result is not ideal, that is the technical implementation does not completely support the requirements of the modeler. The result is that the modeler has to make do with what technical implementation can provide and a distributed model that cannot be satisfactorily validated against the real system. Worse, it is possible that the distributed simulation specialists think that they have done a valid job and the modeler believes that this is the case, as neither has a common language to communicate expectations and solutions.

To illustrate this, consider the following. The owners of two factories want to find out how many products their factories can manufacture in a year. Both factories have been modeled separately using two CSPs. The models might interact by sending entities (possibly the delivery and return of some defective stock), by sharing resources (to reflect a shared set of machinists that can operate various workstations), by scheduling shared events of various kinds (such an emergency shutdown), or by sharing data (such as the current production volume). A CSP will simulate its model using a DES

algorithm. Each model/CSP represents a federate normally running on its own computer. In a distributed simulation, each model/CSP federate therefore exchanges data directly or via a runtime infrastructure (RTI) implemented over a network in a simulation time-synchronized manner. Federate F1 consists of the model M1 and the COTS Simulation Package CSP1 and federate F2 consists of the model M2 and COTS Simulation Package CSP2. In this case federate F1 publishes and sends information to the RTI in an agreed format and simulation time-synchronized manner and federate F2 must subscribe to and receive that information in the same agreed format and time-synchronized manner, that is both federates must agree on a common representation of data and both must use the RTI in a similar way. Further, the sending or transfer of entities and the sharing of resources require different distributed simulation protocols. In entity transfer, the departure of an entity from one model and the arrival of an entity at another can be represented through a time-stamped interaction message sent from one federate to another. The sharing of resources cannot, however, be handled in the same way. For example, when a resource is released or an entity arrives in a queue, a CSP executing the simulation will determine if a workstation can start processing the entity. If resources are shared, each time an appropriate resource changes state a time-stamped communication protocol is required to inform and update the changes of the shared resource state. Similar arguments can be made about shared events and shared data structures.

Trying to get some consensus between modeler, vendor, and distributed simulation specialist on these implementation details is extremely difficult; the degree of subtlety involved in even describing these problems can lead to lengthy discussions where the parties involved typically finish with no definitive understanding of the problems that must be solved.

To attempt to solve this, the SISO standardization group CSPI PDG has created a standardized set of Interoperability Reference Models or "interoperability design patterns" that attempt to capture these subtleties. It allows the capture of interoperability requirements at a modeling level rather than at a technical level, which enables OR/MS modelers and vendors to properly specify the interoperability requirements of a distributed simulation in their terms. These were formally recognized by SISO in 2010 after two rounds of balloting, as SISO-STD-006-2010 Standard for COTS Simulation Package Interoperability Reference Models [SISO 2010a]. These enable modelers, vendors, and solution developers to specify the interoperability problems that must be solved and implemented. The Interoperability Reference Models (IRMs) are therefore intended to be used as follows.

—Clearly identify the model/CSP interoperability capabilities of an existing distributed simulation, for example, the CSP-based distributed simulation is compliant with IRMs Type A.1, A.2 and B.1.
—Clearly specify the model/CSP interoperability requirements of a proposed distributed simulation, for example, the CSP-based distributed simulation must be compliant with IRMs Type A.1 and C.1.

An IRM is defined as the simplest representation of a problem within an identified interoperability problem type. Each IRM can be subdivided into different subcategories of problem. As IRMs are usually relevant to the boundary between two or more interoperating models, models specified in IRMs will be as simple as possible to capture the interoperability problem and to avoid possible confusion. These simulation models are intended to be representative of real model/CSPs but use a set of common model elements that can be mapped onto specific CSP elements. Where appropriate, IRMs specify time synchronization requirements and present alternatives that must be agreed upon. IRMs are intended to be cumulative (some problems may well consist of several IRMs).
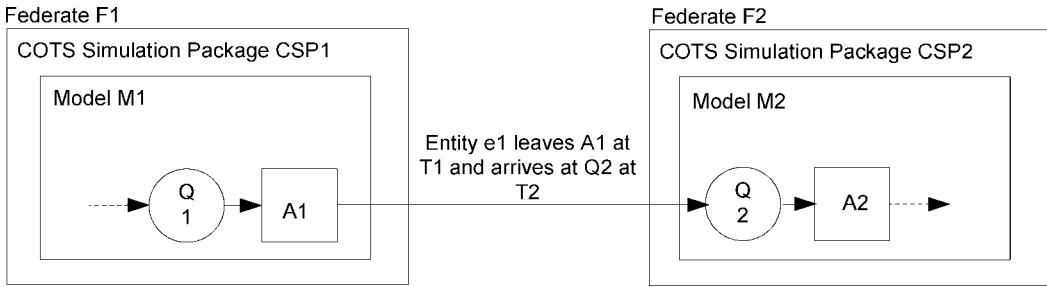
Fig. 1.   IRM Type A.1. General Entity Transfer.

Most importantly, IRMs are intended to be understandable by modelers, vendors, and distributed simulation specialists. Different representations have been used to specify the IRMs (e.g. UML). However, the diagrammatic form presented here was considered to be the most accessible by the different stakeholder groups represented by members of the CSPI PDG involved in the development of the standard. The IRMs themselves resulted from four years of consultation.

4.1. Interoperability Reference Model Types

There are four different types of IRM. These are the following.

Type A.   *Entity Transfer*
Type B.   *Shared Resource*
Type C.   *Shared Event*
Type D.   *Shared Data Structure*

Briefly, IRM Type A Entity Transfer deals with the requirement of transferring entities between simulation models, such as, an entity *Part* leaves one model and arrives at the next. IRM Type B Shared Resource refers to sharing of resources across simulation models. For example, a resource R might be common between two models and represents a pool of workers. In this scenario, when a machine in a model attempts to process an entity waiting in a queue it must also have a worker. If a worker is available in R then processing can take place. If not then work must be suspended until one is available. IRM Type C Shared Event deals with the sharing of events across simulation models. For example, when a variable within a model reaches a given threshold value (a quantity of production, an average machine utilization, etc.) it should be able to signal this fact to all models that have an interest in this fact (to throttle down throughput, route materials via a different path, etc.) IRM Type D Shared Data Structure deals with the sharing of variables and data structures across simulation models. Such data structures are semantically different to resources, for example a bill of materials or a common inventory. Note that during the development of the standard this classification previously appeared as Types I-VI [Taylor et al. 2006]. We now detail each IRM in turn.

4.2. Interoperability Reference Model Type A: Entity Transfer

IRM Type A Entity Transfer represents interoperability problems that can occur when transferring an entity from one model to another. Figure 1 shows a simple illustrative example of the problem of Entity Transfer where an entity e1 leaves activity A1 in model M1 at T1 and arrives at queue Q2 in model M2 at T2. For example, if M1 is a car production line and M2 is a paint shop, then this represents the system where a car leaves a finishing activity in M1 at T1 and arrives in a buffer in M2 at T2 to await painting. Note that the IRM subtypes are intended to be composable, that is 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.

There are currently three IRM Type A Subtypes

IRM Type A.1 General Entity Transfer
IRM Type A.2 Bounded Receiving Element
IRM Type A.3 Multiple Input Prioritization

4.3. IRM Type A.1 General Entity Transfer

IRM Type A.1 General Entity Transfer represents the case, as described in the preceding and shown in Figure 1, where an entity e1 leaves activity A1 in model M1 at T1 and arrives at queue Q2 in model M2 at T2 (see the previous example). This IRM is inclusive of cases where there are many models and many entity transfers (all transfers are instances of this IRM) but not where the receiving element is bounded (IRM Type A.2), and multiple inputs need to be prioritized (IRM Type A.3).

The 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 = < T2 or T1<T2. The place of departure and arrival will be a queue, workstation, and so on. Note that the time inequality must be specified and is a critical requirement. There is an important difference between these time requirements as, T1 = < T2 allows the possibility of an entity leaving M1 and arriving at M2 at the same time (the time at which the entity leaves activity A1 and arrives at queue Q2 is the same). The reason for this is that in the requirement T1 = < T2, T1 = T2 is possible. If this is the case, then the distributed simulation is capable of producing a zero time advance and is classed in distributed simulation terms as a "zero lookahead problem" [Bryant 1977; Chandy and Misra 1979; Fujimoto 1990]. Without going into the technical details (for which the reader may refer to the aforementioned references) it is sufficient to state that some distributed simulation implementations cannot support zero lookahead. Those would not be able to correctly implement scenarios requiring entity transfer where T1 = < T2. This further justifies the need for IRMs as this level of detail is often overlooked.

4.4. IRM Type A.2 Bounded Receiving Element

Consider a production line where a machine is just finishing working on a part. If the next element in the production process is a buffer in another model, the part will be transferred from the machine to the buffer. If, however, the next element is bounded, for example a buffer with limited space or another machine (no buffer space), then a check must be performed to see if there is space or the next machine is free. If there is no space, or the next machine is busy, then to correctly simulate the behavior of the production process, the current machine must hold onto the part and block, that is it cannot accept any new parts to process until it becomes unblocked (assuming that the machine can only process one part at a time). The consequences of this are quite subtle. This is the core problem of the IRM Type A.2. Figure 2 shows an illustrative example, where an entity e1 attempts to leave model M1 at T1 from activity A1 and to arrive at model M2 at T2 in bounded queue Q2. If A1 represents a machine then the following scenario is possible. When A1 finishes work on a part (an entity), it attempts to pass the part to queue Q2. If Q2 has spare capacity, then the part can be transferred. However, if Q2 is full, then A1 cannot release its part and must block. Parts in Q1 must now wait for A1 to become free before they can be machined. Further, when Q2 once again has space, A1 must be notified that it can release its part and transfer it to Q2. Finally, it is important to note that if A1 is blocked the rest of model M1 still functions as normal,
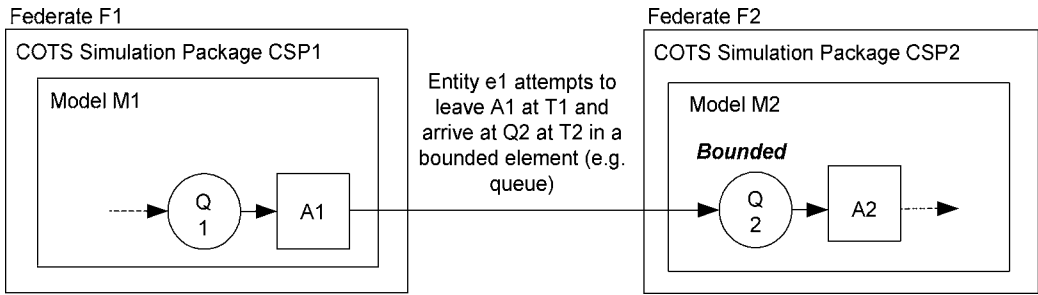
Federate F1

COTS Simulation Package CSP1

Model M1

Entity e1 attempts to
leave A1 at T1 and
arrive at Q2 at T2 in a
bounded element (e.g.
queue)

Federate F2

COTS Simulation Package CSP2

Model M2

Bounded

Fig. 2.   IRM Type A.2. Bounded Receiving Element.

Federate F1

COTS Simulation Package CSP1

Model M1

Entities arrive from different
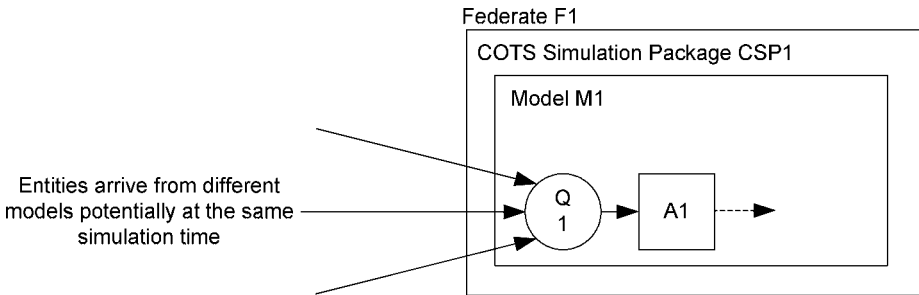models potentially at the same
simulation time

Fig. 3.   IRM Type A.3 Multiple Input Prioritization.

that is a correct solution to this problem must still allow the rest of the model to be
simulated (rather than just stopping the simulation of M1 until Q2 has unblocked).

This IRM is therefore inclusive of cases where the receiving element (queue, worksta-
tion, etc.) is bounded but not where there are multiple inputs that need to be prioritized
(IRM Type A.3). A solution to this IRM problem must also be able to transfer entities
(IRM Type A.1). The IRM Type A.2 is defined as the relationship between an element
O in a model M1 and a bounded element Ob in a model M2 such that if an entity e is
ready to leave element O at T1 and attempts to arrive at bounded element Ob at T2
then the following apply.

—If bounded element Ob is empty, the entity e can leave element O at T1 and arrive
   at Ob at T2, or if bounded element Ob is full, the entity e cannot leave element O at
   T1; element O may then block if appropriate and must not accept any more entities.
—When bounded element Ob becomes not full at T3, entity e must leave O at T3 and
   arrive at Ob at T4; element O becomes unblocked and may receive new entities at T3.
—T1 = <T2 and T3 = <T4.
—If element O is blocked then the simulation of model M1 must continue.
—Note: In some special cases, element O may represent some real world process that
   may not need to block. If T3<T4 then it may be possible for bounded element O to
   become full again during the interval if other inputs to Ob are allowed.

4.5. IRM Type A.3 Multiple Input Prioritization

As shown in Figure 3, 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
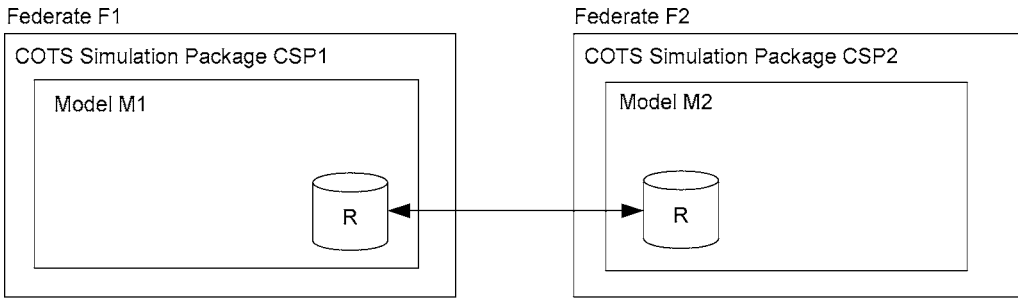
A shared resource R exists at two models M1 and M2. If shared resource R
changes at time T1 in model M1 then it must change at T1 in model M2
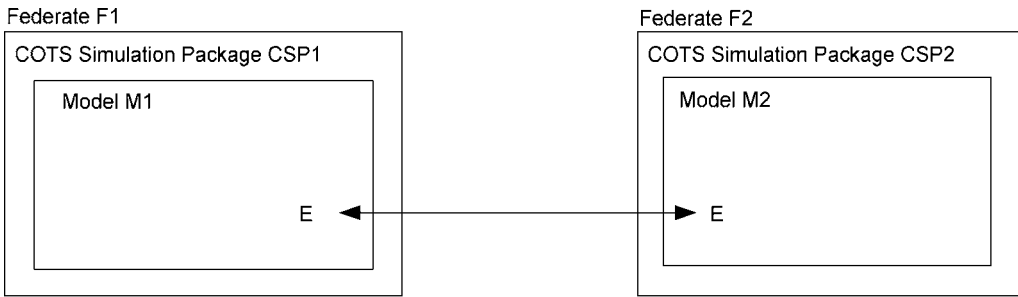
Fig. 4.   IRM Type B.1: General Shared Resource.

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, that is 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, for example 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). Further, it is possible that this priority ordering could be
dynamic or specialized. This IRM is therefore inclusive of cases where multiple inputs
need to be prioritized. This IRM does not include cases where the receiving element is
bounded (IRM Type A.2). A solution to this IRM problem must also be able to transfer
entities (IRM Type A.1).

The IRM Type A.3 Multiple Input Prioritization is defined as the preservation of the
priority relationship between a set of models that can send entities to a model with
receiving queue Q, such that priority ordering is observed if two or more entities arrive
at the same time. Note that the priority rules must be specified and that priority rules
may change during a simulation if required for the real system being simulated.

4.6. Interoperability Reference Model 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, doctors, runways, 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 there is, then the resource
is secured by the activity and held until the activity ends. A resource shared by two or
more models therefore becomes a problem of maintaining the consistency of the state
of that resource in a distributed simulation. Note that this is similar to the problem of
shared data. However, in CSPs resources are semantically different from data and we
therefore preserve the distinction in this standard. There is currently one IRM Type B
subtype, the IRM Type B.1 General Shared Resource.

The IRM Type B.1 General Shared Resource represents the case, as previously out-
lined and shown in Figure 4, where the state of a resource R shared across two or more
models must be consistent. In a model M1 that shares resource R with model M2, M1
will have a copy RM1, and M2 will have a copy RM2. When M1 attempts to change
the state of RM1 at T1, then it must be guaranteed that the state of RM2 in M2 at
T1 will also be the same. Additionally, it must be guaranteed that both M1 and M2
can attempt to change their copies of R at the same simulation time, as it cannot be
guaranteed that this simultaneous behavior will not occur.

Federate F1

COTS Simulation Package CSP1

Model M1

E

Federate F2

COTS Simulation Package CSP2

Model M2

E

A shared event E takes place in two models M1 and M2 at T1.

Fig. 5.   IRM Type C.1: General Shared Event.

The IRM Type B.1 General Shared Resources is defined as the maintenance of consistency of all copies of a shared resource R such that if a model M1 wishes to change its copy of R (RM1) at T1 then the state of all other copies of R will be guaranteed to be the same at T1, and if two or more models wish to change their copies of R at the same time T1, then all copies of R will be guaranteed to be the same at T1.

## 4.7. Interoperability Reference Model 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 subtype, the IRM Type C.1 General Shared Event. This represents the case, as shown in Figure 5, where an event E is shared across two or more models. In a model M1 that shares an event E with model M2 at T1, then we are effectively scheduling two local events EM1 at M1 at T1 and EM2 at M2 at T1. We must therefore guarantee that both copies of the event take place. Care must also be taken to guarantee that if two shared events E1 and E2 are instigated at the same time by different models, then both will occur.

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 the local copies EM1, EM2, and so on. will be guaranteed to be executed at the same time T1, and if two or more models wish to schedule shared events E1, E2, and so on. at T1, then all local copies of all shared events will be guaranteed to be executed at the same time T1.

4.8. Interoperability Reference Model Type D: Shared Data Structure

IRM Type D deals with the problem of sharing data across two or more models in a distributed simulation (such as a production schedule, a global variable, etc.) A shared data structure that is shared by two or more models therefore becomes a problem of maintaining the consistency of the state of that data structure in a distributed simulation. Note that this is similar to the problem of shared resources. However, in CSPs, resources are semantically different from data and we therefore preserve the distinction in this standard. Note also that we consider the sharing of a single data item such as an integer, as being covered by this IRM. There is currently one IRM Type D Subtype, the IRM Type D.1 General Shared Data Structure.

IRM Type D.1 General Data Structure represents the case, as outlined in the preceding and shown in Figure 6, where a data structure D shared across two or more models must be consistent. In a model M1 that shares a data structure D with model M2, M1 will have a copy DM1 and M2 will have a copy DM2. When M1 attempts to change the value of DM1 at T1, then it must be guaranteed that the value of DM2 in
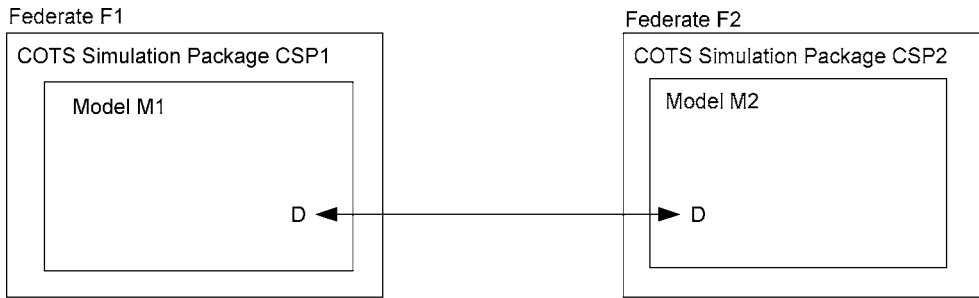
A shared data item D exists at two models M1 and M2. If shared data item D changes at time T1 in model M1 then it must change at T1 in model M2

Fig. 6.   IRM Type D.1: Shared Data.

M2 at T1 will also be the same. Additionally, it must be guaranteed that both M1 and M2 can attempt to change their copies of D at the same simulation time as it cannot be guaranteed that this simultaneous behavior will not occur. The IRM Type D.1 General Shared Data Structure is defined as the maintenance of consistency of all copies of a shared data structure D such that if a model M1 wishes to change its copy of D, DM1 at T1, then the value of all other copies of D will be guaranteed to be the same at T1, and if two or more models wish to change their copies of D at the same time T1, then all copies of D will be guaranteed to be the same at T1.

We now present two OR/MS distributed case studies, one in healthcare and the other in manufacturing, that have been guided by the SISO-STD-006-2010 standard [SISO 2010a]. Other examples of IRM use can be found in Strassburger et al. [2007], Raab et al. [2008], Taylor et al. [2009], Garg et al. [2009], and Pedrielli et al. [2011].

5. THE NATIONAL BLOOD SERVICE CASE STUDY: A DISTRIBUTED SUPPLY CHAIN SIMULATION

The first case study discusses the UK National Blood Service (NBS) supply chain and its realization as a distributed simulation using the CSP Simul8™ and the DMSO HLA-RTI distributed simulation middleware. The National Blood Service (NBS) is a part of the UK National Health Service Blood and Transplant (NHSBT) organization. The NBS infrastructure consists of 15 Process, Testing, and Issuing (PTI) centers, which together serve 316 hospitals across England and North Wales. Each PTI Center thus serves around 20 hospitals. The NBS is responsible for collecting blood through voluntary donations, classifying it by ABO and Rhesus grouping, testing the blood for infectious diseases such as HIV, and processing the blood into around 115 different products (the main ones being Red Blood Cells (RBC), platelets and plasma). The NBS stores the stockpile, transfers excess stock between different NBS centers, and issues the different blood products to the hospitals as per their demand. Each hospital has different ordering policies, and local strategies vary regarding optimum local stock levels. Although this initially appears to be a simple supply chain, issues of multiple blood products, individual ordering strategies, blood unit assignment, and shelf life (RBC usually has a shelf life of only 35 days) make this an extremely complex one.

To investigate this system, the NBS Supply Chain simulation was created by Katsaliaki and Brailsford [2007] and was modeled with inputs from the Southampton PTI Center. This included only RBC and platelets, which together comprise 85% of issues and are the chief source of wastage and shortages. We refer to this model as the conventional NBS Supply Chain model. We have selected this case study for our
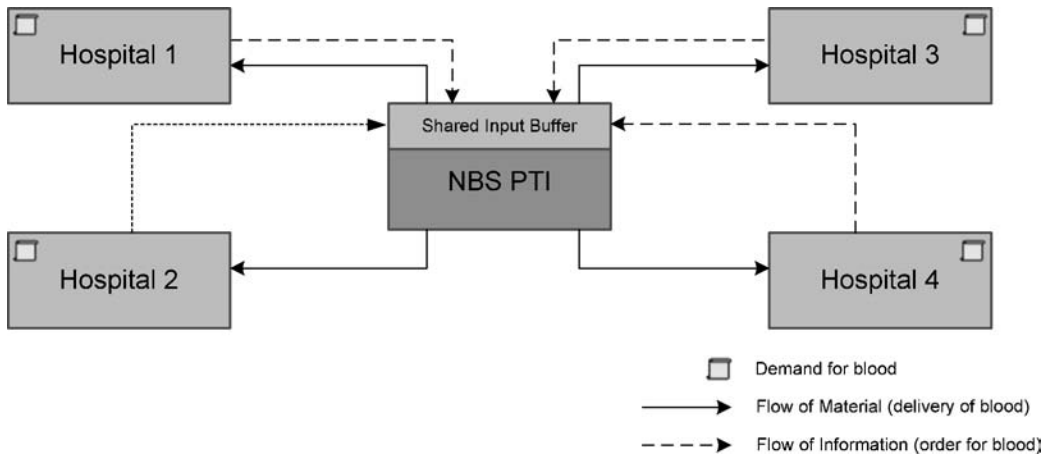
Fig. 7. Information and material flow among the models of the distributed supply chain simulation.

distributed simulation implementation since multiple stakeholders are involved in this supply chain (e.g., the NBS PTI and the different hospitals) and it illustrates privacy and data access issues, and also because all the elements of the model were developed in CSP Simul8. The conventional model contains the processes of the NBS PTI Center, from the collection of blood to the delivery of blood products, and the processes of a hospital. The model captures physicians' requests for blood and the processes whereby the hospital blood bank checks its stock levels and places orders. The order entities and item entities are represented as information flow (hospital orders) and material flow (blood products), respectively.

The problem with the nondistributed approach is threefold. First, the data used is private and sensitive, as it involves information related to clinical practice. Most data in healthcare systems cannot just be taken on demand and is subject to stringent and lengthy data protection checks. Admittedly in the UK, data-sharing does take place between hospitals in Primary Care Trusts and, to some extent, Strategic Health Authorities. However, the NHSBT and the NHS hospitals are effectively separate organizations and it cannot be assumed that data is freely shared. Further, to generalize the original work by Katsaliaki and Brailsford, that is a supply chain analysis tool usable by the many developed countries that have equivalent blood supply chains, then one cannot assume that privacy issues will be any different.

Second, the data is fixed. The data sources are private and not easily moved. There is much data in the model that is used to model the demand for blood, the availability of blood products and the current stock of the blood units in the supply chain. These are updated frequently, and centralizing the data in a single model would make it difficult to ensure the data is up to date. Finally, the execution time is extremely poor. A single year took 14 minutes to run with a single supply center and single hospital, 78 minutes with two hospitals, 17.5 hours with three hospitals and 35.8 with four (1.7GHz processor desktop PC with 1GB RAM). Note that in terms of execution time it may be possible to simplify the modeling approach to increase the speed of the simulation (such as by sacrificing the detail at which blood product orders are placed and/or the shelf-life of blood products). However, the goal here is to understand wastage and ordering patterns; a sacrifice in detail for the sake of performance may produce results faster but not at the required level of detail.

Information and material flows in the distributed version of the NBS model are shown in Figure 7. It is at this point where the guidelines became extremely useful as
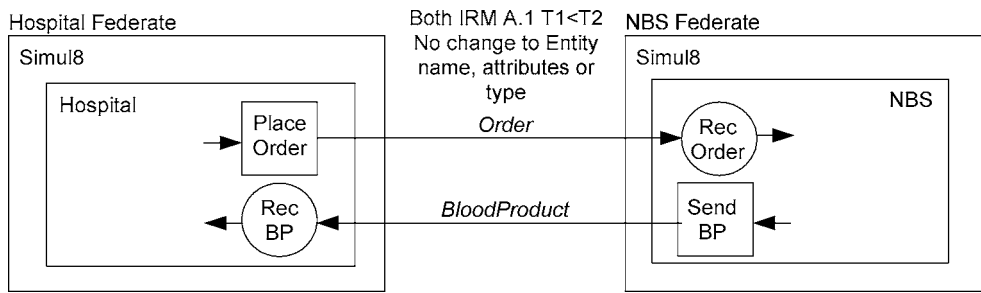
Fig. 8.   IRM for National Blood Service supply chain simulation.

it allowed the modeling team and the distributed simulation team to use the simple
set of diagrams to actively engage in specifying how the models interacted. The main
interaction between the supply center and the hospitals is by sending and receiving
entities that represent orders and deliveries of blood units and products. There are no
shared resources, shared data structures or shared events. IRM Type A Entity Transfer
is therefore the main guideline for this distributed simulation.

In this model there are essentially two entity transfers: (a) orders from a hospital
to the NBS center, and (b) blood units from the NBS center to a hospital. IRM Type
A.1 General Entity Transfer was therefore appropriate for this work. It was discussed
whether IRM Type A.2 Bounded Receiving Element was also appropriate. However, it is
assumed in the model there will always be space for blood units. Similarly, the queuing
of order entities did not pose a problem. IRM Type A.3 Multiple Input Prioritization
suggested that there could have been problems with the queuing of orders received
by the supply center. It was decided that if two or more orders arrived at the same
time then these would be processed in an arbitrary order, as orders are taken within
a certain timeframe. IRM Type A.1 General Entity Transfer requires the inequality
T1 = < T2 or T1<T2 to be specified. It was decided that as the transfer of entities
would effectively represent the scheduling of a future event (the placing and receiving
of an order, the sending and receiving of a blood product), then T1<T2 would always
hold (Figure 8). The IRMs gave significant clarity to the development of the distributed
simulation as it allowed the developers and the modelers to discuss interoperability in
a clear and consistent manner.

The distributed NBS models representing the supply center and hospitals are ex-
ecuted on different computers. Each CSP simulates one element of the supply chain
and is connected with the other models via a computer network. The NBS federation
is composed of one PTI federate and several hospital federates interacting via an RTI
and specially developed adapter software (more details given below). The technical
implementation of the NBS case study follows approaches developed in Taylor et al.
[2006], Mustafee and Taylor [2006] and Mustafee et al. [2009]. More specifically, to link
a CSP to an RTI, an adaptor called the *CSP Controller Middleware (CCM)* has been
developed (as modifying a CSP or RTI directly is often not possible). The CCM links the
CSP and the RTI; it supports two RTI time advance mechanisms: the Time Advance
Request and the Next Event Request; it implements entity transfer as user-defined
HLA interactions. The NBS distributed simulation federation is shown in Figure 9.
Appendix 1 (online) presents the Federation Execution Data (FED) file associated with
the case study implementation; the syntax of the FED file is part of the Federate I/F
Spec. Although, unlike the Federation Object Model (FOM), the FED does not have
types associated with attributes and parameters (e.g., integer, strings), it does contain
the extract of the FOM that is necessary for the RTI to function (Kuhl et al. 1999) and is
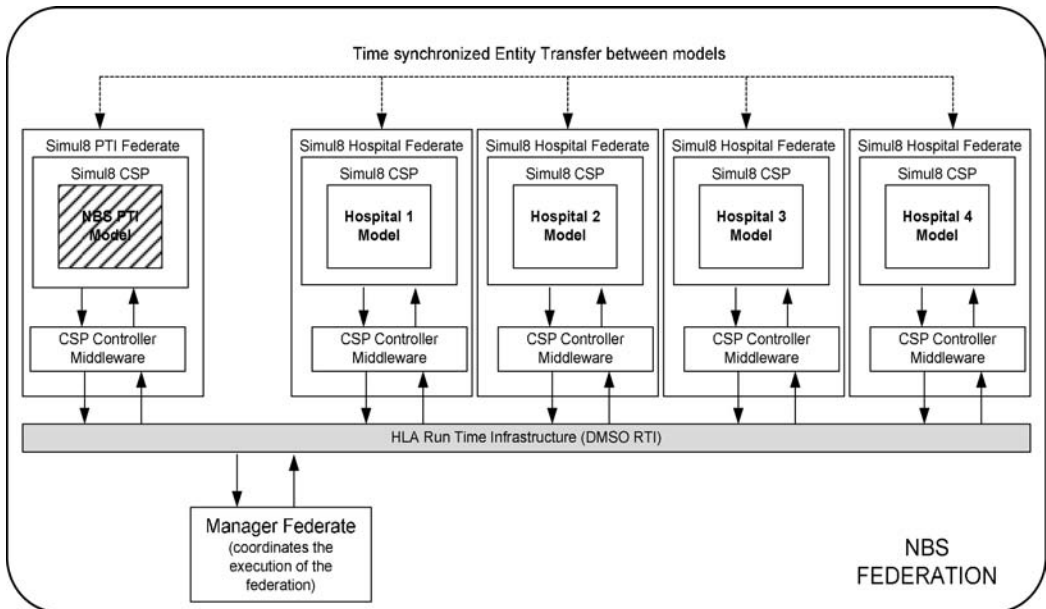
Fig. 9.   The NBS Distributed Simulation Federation comprised of one NBS-PTI model, four NBS-Hospital models, and one Manager Federate.

useful in understanding the HLA user-defined object and interaction classes that have been used in the implementation of a distributed simulation. For example, Appendix 1 shows that the NBS federation uses a total of eight user-defined interaction classes to represent the information flow and the material flow between the PTI and four hospitals (e.g., *"Hospital1Orders"*, *"Hospital2Orders"*, *"NBSDeliveryHospital1"*, *"NBS-DeliveryHospital2"*). Further, it shows that each interaction has between five and eight parameters, for example, product type (RBC or platelets), blood group, delivery type (scheduled delivery or emergency delivery), time when the blood unit expires, and so on.

6. THE TRACTOR FACTORY CASE STUDY: A DISTRIBUTED SIMULATION IN THE CONTEXT OF MANUFACTURING

Within the planning, design, and redesign of new production facilities at Deere & Co., a leading manufacturer of agricultural, forestry, and construction equipment, DES models are routinely used to simulate the behavior of the systems under investigation [Taylor et al. 2009]. Dedicated solution sets exist for the simulation of certain types of production systems, like assembly lines or paint systems. These models are used for planning new factories as well as for supporting ongoing factory operations. The CSP used in the presented case study was SLX^{TM} [Henriksen 1997].

   The basis of the case study is a planned tractor factory in South America [Strass-burger et al. 2007]. With a target production of 40 tractors per day, Monday through Friday, the production system under investigation consists of seven components, as shown in Figure 10; in total four prepaint asynchronous assembly lines for chassis, transmissions, and front a xle assembly, two post-paint asynchronous assembly lines for cabs and tractors, and a wet-on-wet paint system. Each component of the production system consists of multiple manned work stations. The information and material flow of the overall production system is shown in Figure 10.

   In the traditional application of simulation in the company, each of the production sections would be simulated using a separate simulation model using established
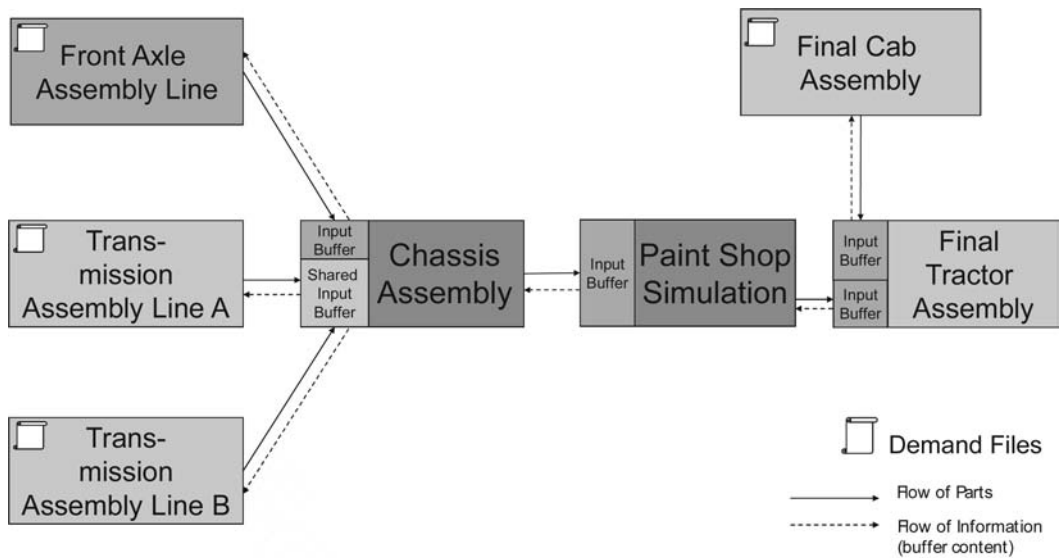
Fig. 10. Information and material flow among the models of the distributed manufacturing simulation.

simulation frameworks. This yields good results when each section is investigated separately. It does, however, imply the use of simplified assumptions about the input and output behavior of the different sections. In order to simulate and take into account interdependencies between the different production sections (like different shift regimes and the size of input buffers) correctly, an overall simulation of all relevant production sections was needed.

The motivation for using distributed simulation in this example was therefore the integration of independently developed existing models that cannot easily be combined within a single CSP for common execution. The IRMs discussed in this article were used to identify the interoperability requirements in the feasibility analysis conducted between the simulation experts at Deere & Co. as the domain experts and the institution performing the integration of the models into a distributed simulation. In this analysis it was determined that the main focus of the combination of the models was the correct implementation of the material flow. There are several types of entities that must be transferred between the models, all relating to parts that are the output of a certain system. Entity transfer had to take into account bounded input buffers in all of the production systems receiving parts. There is a travel time for parts between each of the sections. With one exception there is a unique input buffer for each of the sending models.

With the help of the IRMs it was therefore possible to define that the interoperability solution for the distributed manufacturing simulation had to be capable of implementing an IRM Type A.2 entity transfer with T1<T2 and T3<T4 in all cases. In one case there was also a shared input buffer into which two models delivered entities. This input buffer was a candidate for IRM A.3, but as the scenario did not require input prioritization, this fact had no implications for the required implementation.

The distributed simulation that was developed in this case study uses the open source CERTI [Noulard et al. 2009]. Like the NBS case study, the implementation of the actual entity transfer was performed using HLA interaction messages. However, unlike the NBS case study, there was also a need to exchange information about buffer content of the several input buffers. This was achieved using HLA object instances
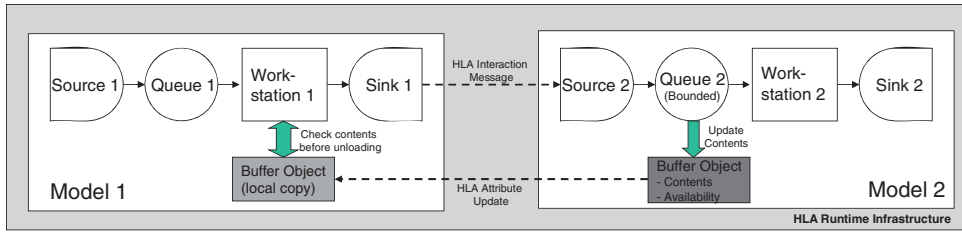
Fig. 11.   Implementation scheme for IRM A.2 in the manufacturing case study.

generated for each relevant input buffer (Figure 11). Each model sending into a certain input buffer would have to subscribe to the class of that buffer and would subsequently be informed about changes in the buffer content. The sending model is thus enabled to decide about the need to delay an entity transfer, depending on the buffer content.

Appendix 2 (online) presents the abridged FED file associated with the tractor factory case study; the FED file lists the user-defined object classes and interaction classes that have been used. As can be seen from the appendix, the sender-receiver relationship between the federates is represented by building a hierarchy of interaction classes and subclasses. As the root user-defined class, an interaction class called *"Transfer-Entity"* is specified. It has subclasses that correspond to all potential recipient models, for example *"TransferEntityToPaintShop", "TransferEntityToChassisAssembly"*, and so on. In addition to this subclass identifying the target federate, individual interaction classes are introduced for each connection between a sending model and a target model. In our example (Figure 10), there would be a single subclass to *"TransferEntityToPaintShop"* called *"TransferEntityChassisAssemblyToPaintShop"*. This interaction class is published by the sending model (Chassis Assembly) and is subscribed by the receiving model (Paint Shop). With regard to our second example (subclass *"TransferEntityToChassisAssembly"*), there are three interaction subclasses that are published by the sending models (Front Axle Assembly Line, Transmission Assembly Line A, Transmission Assembly Line B) and which are subscribed by the receiving model (Chassis Assembly). These three interaction subclasses are *"TransferEntityTransmissionAssemblyAToChassisAssembly", "TransferEntityTransmissionAssemblyBToChassisAssembly", and "TransferEntityFrontAxleAssemblyToChassisAssembly"*. For transmitting the state of a transferred entity, the interaction classes have a single parameter named *"Entity"*. The type of this parameter is a complex data type (record) identifying the name of the entity, identifiers for the source and destination (which identify the sink of the sending model and the targeted source of the receiving model), and other simulation dependent attributes (e.g., a unique sequence number of the entity and a set of option codes). With regard to IRM Type A.2 (Bounded Receiving Element), some mechanism for checking if there is sufficient space in the input queue/buffer of the receiving model had to be implemented. In our implementation the input queue/buffer is modeled as a persistent object class instance in the HLA sense (Figure 11). Furthermore, a hierarchical definition of these object classes has been used; this is similar to the one for the user-defined interaction classes. Thus, we have defined the HLA object class *"UpdateObject"* as the root of the user-defined object classes; there are subclasses *"UpdateObjectChassisAssembly", "UpdateObjectPaintShop"*, and so on. under the root class; these classes further encapsulate the subclass *"InputBuffer"* (online Appendix 2).

The implemented interoperability solution makes use of the SLX-HLA-Interface (Figure 12). This interface is implemented as a dynamic-link-library and provides functions to SLX models that allow direct access to HLA functionality [Strassburger et al. 1998]. This functionality is further encapsulated by introducing
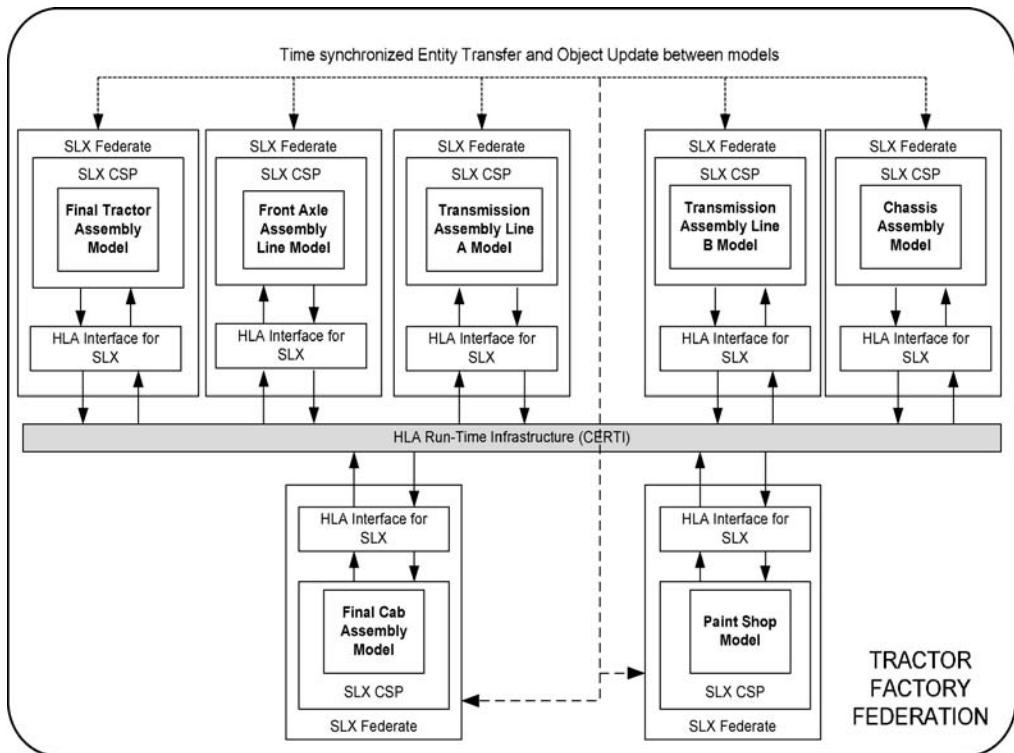
Fig. 12.   The Distributed Simulation of the tractor factory, including seven simulation models, all implemented in the CSP SLX^TM, but in partially different simulation frameworks. Distributed Simulation offered the only way to integrate these models.

easy-to-use statements into the SLX language [Strassburger 2006]. All models are executed on a single multicore PC. Runtimes of all models, simulating a week's production range between 30 seconds and 8 minutes, depending on different lookahead values. This illustrates that in this case study it was not speed-up or memory constraints that led to the use of distributed simulation, but the need to integrate different, heterogeneous simulation models. Further implementation details can be found in Strassburger et al. [2007].

7. CONCLUSION

This article has argued that the development of a single all-encompassing simulation model that represents well-defined subsystems belonging to multiple functional areas (associated with either one organization, or indeed, with several organizations) can bring with it issues of data/information sharing, data transfer/access, model composability, and execution time. Distributed simulation has been introduced as a possible solution for these issues, as it addresses concerns among intra- and inter-organizational stakeholders with regard to data/information sharing and/or security and privacy of shared data/information. The article has also argued that a set of standardized Interoperability Reference Models (IRMs) can assist in development of CSP-based distributed simulation as it clarifies communication issues between the modelers and distributed simulation specialists. The case studies have demonstrated how distributed simulation in OR/MS can be realized and its benefits and barriers. It has shown how data privacy is maintained and how problems of data transfer/access can be avoided. It has

illustrated how issues of model composability can be avoided by keeping the models separate. Although influenced by work described in the related work section, the technological implementation took a great deal of time and required expert knowledge, developers with appropriate skills, and close collaboration with the vendor. The IRMs helped reduce the time needed to understand and validate how models interacted.

In reality how realistic is this? Our case studies have demonstrated that if concerns relating to privacy, data, and model composition are an issue then distributed simulation gives a potential solution for simulation both within the confines an organization (e.g. simulation of well-defined subprocesses such as "procurement" and "manufacturing" within an organization) and across organizations (e.g. supply chain simulation without complex nondisclosure agreements and data access policies). However, a significant barrier is the skill set needed to achieve an implementation. Generally in the military simulation community, the skills needed to implement a distributed simulation are arguably widespread. However, in the OR/MS community of users and vendors represented by CSPs it is not. We argue that the way forward is standardization. Standardization activities bring together stakeholders to debate appropriate solutions to problems. The CSPI PDG has produced the first of several standards targeted at distributed simulation in OR/MS. These IRMs are an important step forward, as these are intended to simplify issues in distributed simulation as well as forming the basis for relevant stakeholders to engage over technical solutions. Garg et al. [2009] and Pedrielli et al. [2011] have investigated generic implementations of the IRMs for the HLA using a similar approach to the *CSP Controller Middleware* mentioned in this article (Section 5). Now that the HLA Evolved [IEEE 2010] and the Distributed Simulation Engineering and Execution Process (DSEEP) [IEEE 2011] standards have been finalized, work continues to standardize HLA representations of the IRMs by developing guideline Federation Object Model and Simulation Object Model specifications and RTI implementations. The first step towards this is the creation of the CSPI specification schema that formally represents the interoperability relationships between models. Investigations are also being carried out to produce IRM versions compatible with the Base Object Model standard [SISO 2006] to better link in IRM research with contemporary HLA development. Similarly, the Levels of Conceptual Interoperability Model (LCIM) [Tolk et al. 2007] is being studied to determine at which level the IRMs should be included.

ELECTRONIC APPENDIX

The electronic appendix to this article is available in the ACM Digital Library.

ACKNOWLEDGMENTS

REFERENCES

AL-ZOUBI, K. AND WAINER, G. 2008. Interfacing and coordination for a DEVS simulation protocol standard. In *Proceedings of the 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications (DS-RT)*. IEEE Computer Society, Los Alamitos, CA 300–307.

BANDINELLI, R., RAPACCINI, M., TUCCI, M., AND VISINTIN, F. 2006. Using simulation for supply chain analysis: Reviewing and proposing distributed simulation frameworks. *Prod. Plann. Contr, 17*, 2, 167–175.

BOER, C. A., DE BRUIN, A. AND VERBRAECK, A. 2009. A survey on distributed simulation in industry. *J. Simul. 3*, 1, 3–16.

BORSHCHEV, A., KARPOV, Y., AND KHARITONOV, V. 2002. Distributed simulation of hybrid systems with AnyLogic and HLA. *Future Gen. Comput. Syst. 18*, 6, 829–839.

BRYANT, R. E. 1977. Simulation of packet communication architecture computer systems, Tech. rep. MIT/LCS/TR-188, Massachusetts Institute of Technology, Cambridge, MA.

CHANDY, K. M. AND MISRA, J. 1979. Distributed simulation: A case study in design and verification of distributed programs. *IEEE Trans. Softw. Eng. 5*, 5, 440–452.

FUJIMOTO, R. M. 2000. *Parallel and Distributed Simulation Systems*. John Wiley & Sons, New York, NY.

FUJIMOTO, R. M. 1990. Parallel discrete event simulation. *Comm. ACM 33*, 10, 30–53.

FUJIMOTO, R. M. AND WEATHERLY, R. M. 1996. Time management in the DoD High Level Architecture. In *Proceedings of the 10th Workshop on Parallel and Distributed Simulation Workshop*. IEEE Computer Society, Los Alamitos, CA, 60–67.

GAN, B. P . AND TURNER, S. J. 2000. An asynchronous protocol for virtual factory simulation on shared memory multiprocessor systems. *J. Oper. Res. Soc. 51*, 4, 413–422.

GAN, B. P., LENDERMANN, P., LOW, M. Y. H., TURNER, S. J., WANG, X., AND TAYLOR, S. J. E. 2005. Interoperating Autosched AP using the High Level Architecture. In *Proceedings of the 37th Winter Simulation Conference*. ACM Press, New York, NY, 394–401.

GAN, B. P., LIU, L., JAIN, S., TUMER, S. J., CAI, W., AND HSU, W. 2000. Distributed supply chain simulation across enterprise boundaries. In *Proceedings of the 32nd Winter Simulation Conference*. ACM Press, New York, NY, 1245–1251.

GARG, A. K., VENKATESWARAN, J., AND SON, Y.-J. 2009. Generic interface specifications for integrating distributed discrete-event simulation models. *J. Simul. 3*, 114–128.

GLOBALSECURITY.ORG. 2002. Millennium Challenge. http://www.globalsecurity.org/military/ops/millennium-challenge.htm

HENNINGER, A. E., CUTTS, D., LOPER, M., LUTZ, R., RICHBOURG, R., SAUNDERS, R., AND SWENSON, S. 2008. Live virtual constructive architecture roadmap (LVCAR) final report, Tech. rep., M&S CO Project No. 06OC-TR-001.

HENRIKSEN, J. O. 1997. An Introduction to SLX^{TM}. In *Proceedings of the Winter Simulation Conference*. 559–566.

HIBINO, H., FUKUDA, Y., YURA, Y., MITSUYUKI, K., AND KANEDA, K. 2002. Manufacturing adapter of distributed simulation systems using HLA. In *Proceedings of the 34th Winter Simulation Conference*. ACM Press, New York, NY, 1099–1107.

HOLLOCKS, B. W. 2006. Forty years of discrete-event simulation—a personal reflection. *J. Oper. Res. Soc. 57*, 12, 1383–1399.

IEEE. 2010. IEEE 1516-2011 IEEE Standard For Modeling And Simulation (M&S) High Level Architecture (HLA). Institute of Electrical and Electronics Engineers, New York, NY.

IEEE. 2011. IEEE 1730-2011 IEEE Standard For Distributed Simulation Engineering And Execution Process (DSEEP). Institute of Electrical and Electronics Engineers, New York, NY.

JAIN, S., RIDDICK, F., CRAENS, A., AND KIBIRA, D. 2007. Distributed simulation for interoperability testing along the supply chain. In *Proceedings of the 39th Winter Simulation Conference*. ACM Press, New York, NY, 1044–1052.

JOHNSON, G. D. 1999. Networked simulation with HLA and MODSIM III. In *Proceedings of the 31st Winter Simulation Conference*. ACM Press, New York, NY, 1065–1070.

KARLSSON, M. AND OLSSON, L. 2001. pRTI 1516—Rationale and design. In *Proceedings of the Fall Simulation Interoperability Workshop* (*SIW*).

KATSALIAKI, K. AND BRAILSFORD, S. 2007. Using simulation to improve the blood supply chain. *J. Oper. Res. Soc. 58*, 2, 219–227.

KUHL, F., WEATHERLY, R., AND DAHMANN, J. 1999. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*. Prentice Hall, Upper Saddle River, NJ.

LEE, S., ZHAO, X., SHENDARKAR, A., VASUDEVAN, K., AND SON, Y.-J. 2008. Fully dynamic epoch time synchronisation method for distributed supply chain simulation. *Int. J. Comput. Appl. Technol. 31*, 3-4, 249–262.

LENDERMANN, P. 2006. About the need for distributed simulation technology for the resolution of real-world manufacturing and logistics problems. In *Proceedings of the 38th Winter Simulation Conference*. ACM Press, New York, NY, 1119–1128.

LENDERMANN, P., TURNER, S. J., LOW, M. Y. H., GAN, B. P., JULKA, N., CHAN, L. P., CAI, W. T., LEE, L. H., CHEW, E. P., TENG, S. Y., AND MCGINNIS, L. F. 2007. An Integrated and adaptive decision-support framework for high-tech manufacturing and service networks. *J. Simul. 1*, 2, 69–79.

LI, J., YUAN, A., AND WU, Q. 2010. A framework of simulation for cluster supply chain collaboration. In *Proceedings of the International Conference on Internet Technology and Applications* (*ITAP*). Art. no. 5566100.

MALINGA, L. AND LEROUX, W. H. 2009. HLA RTI performance evaluation. In *Proceedings of the European Simulation Interoperability Workshop*. 1–6.

MERTINS, K., RABE, M., AND JÄKEL, F-W. 2005. Distributed modeling and simulation of supply chains. *Int. J. Comput. Integr. Manuf. 18*, 5, 342–349.

MUSTAFEE, N. 2004. Performance evaluation of interoperability methods for distributed simulation. MSc. Dissertation, Department of Information Systems, Computing and Mathematics, Brunel University, UK.

MUSTAFEE, N. AND TAYLOR, S. J. E. 2006. Investigating distributed simulation with COTS simulation packages: experiences with Simul8 and the HLA. In *Proceedings of the Operational Research Society Simulation Workshop*. Operational Research Society, Birmingham, UK, 33–42.

MUSTAFEE, N., TAYLOR, S. J. E., KATSALIAKI, K., AND BRAILSFORD, S. 2009. Facilitating the analysis of a UK National Blood Service supply chain using distributed simulation. *Trans. Soc. Model. Simul. Intern. 85*, 2, 113–128.

MUSTAFEE, N., TAYLOR, S. J. E., KATSALIAKI, K., AND BRAILSFORD, S. 2006. Distributed simulation with COTS simulation packages: A case study in health care supply chain simulation. In *Proceedings of the 38th Winter Simulation Conference*. ACM Press, New York, NY, 1136–1142.

NOULARD, E., ROUSSELOT, J-Y., AND SIRON, P. 2009. CERTI, an Open Source RTI, Why and How. In *Proceedings of the Spring Simulation Interoperability Workshop*. 23–27.

PAN, K., TURNER, S. J., CAI, W., AND LI. Z. 2008. Design and performance evaluation of a service oriented HLA RTI on the grid. In *Grid Computing: Infrastructure, Service, and Application*. Taylor and Francis, London.

PEDRIELLI, G., TERKAJ, W., SACCO, M., AND TOLIO, T. 2011. Simulation of complex manufactuing systems via HLA-based infrastructure. In *Proceedings of the 25th Workshop on Principles of Advanced and Distributed Simulation*. IEEE Computer Society, Los Alamitos, CA.

PIDD, M. 2004. *Computer Simulation in Management Science* 5th Ed. John Wiley & Sons, Chichester, UK.

RAAB, M., SCHULZE, T., AND STRASSBURGER, S. 2008. Management of HLA-based distributed legacy SLX-Models. In *Proceedings of the Winter Simulation Conference*. 1086–1093.

RABE, M., JAEKEL, F.-W., AND WEINAUG, H. 2006. Reference models for supply chain design and configuration. In *Proceedings of the 38th Winter Simulation Conference*. ACM Press, New York, NY, 1143–1150.

SANTOS, R. A., NORMEY-RICO, J. E., GOMEZ, A. M., ARCONADA, L. F. A., AND DE PRADA MORAGA, C. 2008. Distributed continuous process simulation: An industrial case study. *Comput. Chem. Eng. 32*, 6, 1195–1205.

SISO. 2006. SISO-STD-003-2006 Base Object Model (BOM) template specification. Simulation Interoperability Standards Organization, Orlando, FL.

SISO. 2010a. SISO-STD-006-2010 Standard for COTS Simulation Package Interoperability Reference Models. Simulation Interoperability Standards Organization, Orlando, FL.

SISO. 2010b. SISO-STD-008-2010 Standard for Core Manufacturing Simulation Data: UML Model. Simulation Interoperability Standards Organization, Orlando, FL.

STRASSBURGER, S. 2006. The road to COTS-interoperability: From generic HLA-interfaces towards plug-and-play capabilities. In *Proceedings of the Winter Simulation Conference*. ACM Press, New York, NY, 1111–1118.

STRASSBURGER, S., SCHULZE, T., AND FUJIMOTO, R. M. 2008. Future trends in distributed simulation and distributed virtual environments—Results of a peer study. In *Proceedings of the Winter Simulation Conference*. ACM Press, New York, NY, 777–785.

STRASSBURGER, S., SCHULZE, T., KLEIN, U., AND HENRIKSEN, J. O. 1998. Internet-based simulation using off-the-shelf simulation tools and HLA. In *Proceedings of the 30th Winter Simulation Conference*. ACM Press, New York, NY, 1669–1676.

STRASSBURGER, S., SCHULZE, T., AND LEMESSI, M. 2007. Applying CSPI reference models for factory planning. In *Proceedings of the 39th Winter Simulation Conference*. ACM Press, New York, NY, 603–609.

STRASSBURGER, S. AND TAYLOR, S. J. E. 2012. A comparison of the CSPI and CMSD standards. In *Proceedings of the Spring Simulation Interoperability Workshop*.

TAYLOR, S. J. E., SUDRA, R., JANAHAN, T., TAN, G., AND LADBROOK, J. 2002. GRIDS-SCF: An infrastructure for distributed supply chain simulation. *Trans. Soc. Model. Simul. Intern. 78*, 5, 312–320.

TAYLOR, S. J. E., BOHLI, L., WANG, X., TURNER, S. J., AND LADBROOK, J. 2005a. Investigating distributed simulation at the Ford Motor Company. In *Proceedings of the 9th International Symposium on Distributed Simulation and Real-Time Applications (DSRT)*. IEEE Computer Society, Los Alamitos, CA, 139–147.

TAYLOR, S. J. E., TURNER, S. J., MUSTAFEE, N., AHLANDER, H., AND AYANI, R. 2005b. COTS distributed simulation: A comparison of CMB and HLA interoperability approaches to type I interoperability reference model problems. *Trans. Soc. Model. Sim. Intern. 81*, 1, 33–43.

TAYLOR, S. J. E., WANG, X., TURNER, S. J., AND LOW, M. Y. H. 2006. Integrating heterogeneous distributed COTS discrete-event simulation packages: An emerging standards-based approach. *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans 36*, 1, 109–122.

TAYLOR, S. J. E., MUSTAFEE, N., STRASSBURGER, S., TURNER, S. J., LOW, M. Y. H., AND LADBROOK, J. 2007. The SISO CSPI PDG standard for commercial off-the-shelf simulation package interoperability reference models. In *Proceedings of the Winter Simulation Conference*. ACM Press, New York, NY, 594–602.

TAYLOR, S. J. E., TURNER, S. J., AND STRASSBURGER, S. 2008. Guidelines for commercial-off-the-shelf interoperability. In *Proceedings of the 40th Winter Simulation Conference*. ACM Press, New York, NY, 193–204.

TAYLOR, S. J. E., TURNER, S. J., STRASSBURGER, S., MUSTAFEE, N., AND PAN, K. 2009. Commercial-off-the-shelf simulation package interoperability: Issues and futures. In *Proceedings of the 41st Winter Simulation Conference*. ACM Press, New York, NY, 203–215.

TAYLOR, S. J. E., GHORBANI, M., KISS, T., FARKAS, D., MUSTAFEE, N., KITE, S., TURNER, S. J., AND STRASSBURGER, S. 2011. Distributed computing and modeling and simulation: Speeding-up simulations and creating large models. In *Proceedings of the Winter Simulation Conference*. ACM Press, New York, NY, 161–175.

TOLK, A., DIALLO, S. Y., AND TURNITSA, C. D. 2007. Applying the levels of the conceptual interoperability model in support of integratability, interoperability, and composability for system-of-systems engineering. *J. Systemics, Cybern. Inform. 5*, 5, 65–74.

US DEPARTMENT OF DEFENSE. 1999. *High Level Architecture Run-Time Infrastructure RTI 1.3-Next Generation Programmer's Guide*. US Department of Defense Modeling and Simulation Office.

UYGUN, O., ÖZTEMEL, E., AND KUBAT, C. 2009. Scenario based distributed manufacturing simulation using HLA technologies. *Inform. Sci. 179*, 10, 1533–1541.

WANG, X., TURNER, S. J., LOW, M. Y. H AND TAYLOR, S. J. E. 2006. COTS simulation package (CSP) interoperability—A solution to synchronous entity passing. In *Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation*. IEEE Computer Society, Los Alamitos, CA, 201–210.

ZEIGLER. B. P., KIM, D., AND BUCKLEY, S. J. 1999. Distributed supply chain simulation in a DEVS/CORBA execution environment. In *Proceedings of the 31st Winter Simulation Conference*, 1333–1340.