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 simulation packages such as Simul8™ and SLX™. A DES model represents the processes associated with a system of interest; but, in cases where the underlying system is large and/or logically divided, the system may be conceptualized as several sub-systems. These sub-systems may belong to multiple stakeholders, and creating an all-encompassing DES model may be difficult for reasons such as, concerns among the intra- and inter-organizational stakeholders with regard to data/information sharing (e.g., security and privacy). Furthermore, issues such as model composability, data transfer/access problems and execution speed may also make a single model approach problematic. A potential solution could be to create/reuse well-defined DES models, each modeling the processes associated with one sub-system, and using distributed simulation technique to execute the models as a unified whole. Although this approach holds great promise, there are technical barriers. One such barrier is the lack of common ground between distributed simulation developers and simulation practitioners. In an attempt to bridge this gap, this paper reports on the outcome of an international standardization effort, the SISO-STD-006-2010 Standard for Commercial-Off-The-Shelf Simulation Package Interoperability References Models (IRMs). This facilitates the capture of interoperability requirements at a modeling level rather than a technical level and enables simulation practitioners and vendors to properly specify the interoperability requirements of a distributed simulation in their terms. Two distributed simulation examples are given to illustrate the use of IRMs.

Categories and Subject Descriptors: **I.6.8 [Simulation and Modeling]**: Distributed Simulation

General Terms: Standards

Additional Key Words and Phrases: Commercial-off-the-shelf Simulation Packages, Healthcare, Manufacturing

ACM Reference Format:

…

DOI = …

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

Authors' addresses: S.J.E. Taylor, ICT Innovation Group, Department of Information Systems and Computing, Brunel University, Stephen J. Turner, Parallel and Distributed Computing Center, Nanyang Technological University, Steffen Strassburger, School of Economic Sciences, Ilmenau University of Technology, Navonil Mustafee, School of Business and Economics, Swansea University.

frequently applied across a range of industries such as manufacturing, travel, finance, healthcare and supply chains [Hollocks 2006]. In the context of Operations Research/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 like Witness™ (Lanner Group), Simul8™ (Simul8 Corporation), SLX™ (Wolverine Software), AnyLogic™ (XJ Technologies) and Arena™ (Rockwell Automation), etc. Extant OR/MS literature suggests that a vast majority 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 sub-systems. These sub-systems may belong to multiple functional areas of an organization (e.g., a manufacturing facility may have several well-defined sub-processes such as "assembly line" and "paint shop") or they may be related to several organizations (e.g., distinct sub-processes like "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 sub-systems 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 sub-systems associated with "manufacturing" and "repair and maintenance", and it may depend on the "transport" and "distribution" sub-processes 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, concerns among the intra- and inter-organizational stakeholders with regard to data/information sharing and/or 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 sub-systems 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 underlying system of interest. Although this approach of executing multiple DES models permits data-hiding and yet enables the concurrent 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 paper 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 of the paper is an approach to facilitating the capture of interoperability requirements at a modelling level rather than a technical level and enables OR/MS modellers 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 paper 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 paper 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 the IEEE 1516 High Level Architecture [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 *Run-time 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]; however the DMSO RTI middleware is no longer available. There are several RTIs presently available including the Pitch pRTI [Karlsson and Olsson 2001], the Service-Oriented HLA RTI (SOHR) [Pan et al. 2008], CERTI Open Source RTI [Noulard et al. 2009] and 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 paper), the combination of a CSP and its model is a *federate*; a *CSP-based federation* comprises 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 the communication network.

To illustrate how distributed simulation might benefit OR/MS simulation practice, consider a system that may be composed of several well-defined intra-organizational sub-systems (e.g., a multi-national organization that operates in several countries), or indeed, sub-systems *owned* by different organizations (e.g., an inter-organizational 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 specified explicitly in the model. In addition to *privacy*, further problems include:

— *Data transfer/access problems*. Stakeholders may be 'open' to each other (i.e. 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 (even when accessed over the Intranet). 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 the above 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 Intranet or the Internet using *distributed simulation* technologies 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 benefits 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 CSPs are 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 (i.e. a technical solution is not covered by the CSP license fee). The second is that there is a very steep learning curve associated with implementing a CSP-HLA integration solution because it requires familiarity with distributed simulation theory, the HLA standard and HLA-based technology. 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 researcher 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 emerging technologies to support distributed simulation. For example, Common Object Request Broker Architecture (CORBA) [Zeigler et al. 1999], Message Passing Interface for Shared-memory MultiProcessor (MPI-SMP) [Gan and Turner 2000], and Generic Runtime Infrastructure for Distributed Simulation for Supply Chain Federation (GRIDS-SCF) [Taylor et al. 2002]. More recently researchers have used technological standards such as 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, 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: AnyLogic™ [Borshchev et al. 2002], AutoSched™ [Gan et al. 2005; Lendermann et al. 2007], Witness™ [Taylor et al. 2005a], SIMUL8™ [Mustafee and Taylor 2006]; and simulation languages MODSIM III™ [Johnson 1999], DEVS [Al-Zoubi and Wainer 2008] and SLX™ [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™, SIMPLE++™ and GAROPS™ [Hibino et al. 2002] and research into semi-conductor supply chains [Lendermann et al. 2007]. Uygun, Öztemel and Kubat [2009] refine the above 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 on from this, Taylor et al. [2005b] investigated the performance of distributed simulation middleware HLA-RTI over different supply chain topologies (pipeline topology, local feedback topology and fully interconnected topology). 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 the above 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 above approaches into a single standard that is 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 around 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; Taylor et al. 2008; SISO 2010a]. These were created by following SISO's Balloted Standards Development and Support Process (BPDSP) which involved 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 study possible implementations; Taylor et al. [2005a] who investigate the use of distributed simulation in engine manufacturing; Gan et al. [2005] and Lendermann et al. [2007] who investigate the use of distributed simulation in semiconductor manufacturing supply chains; Mustafee et al. [2009] who apply distributed simulation in the context of a healthcare supply chain. Many of the authors listed above have participated in the development of the CSPI standards. Standardization of the 'adaptor' approach is still on-going 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 extend 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 no alternative solution to coupling CSPs using distributed simulation; rather it supports other aspects of simulation interoperability.

4. GUIDELINES FOR CSP-BASED DISTRIBUTED SIMULATION

The range of state of the art CSPs are not easily used for distributed simulation. As presented earlier, this type of distributed simulation is developed typically 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 with the technical implementation not completely supporting 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 perfect job and the modeler believes them 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).

Reiterating the overview given earlier, implementing this as a distributed supply chain simulation or federation composes of a set of CSPs and their models. 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, i.e. 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 an 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.

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 long, 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 a technical level that 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 are effectively a set of simulation patterns or templates that enable modelers, vendors and solution developers to specify the interoperability problems that must be solved. The Interoperability Reference Models (IRMs) are intended to be used as follows:

— To clearly *identify* the model/CSP interoperability *capabilities* of an *existing* distributed simulation, e.g. the CSP-based distributed simulation *is compliant* with IRMs Type A.1, A.2 and B.1.

— To clearly *specify* the model/CSP interoperability *requirements* of a *proposed* distributed simulation, e.g. 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

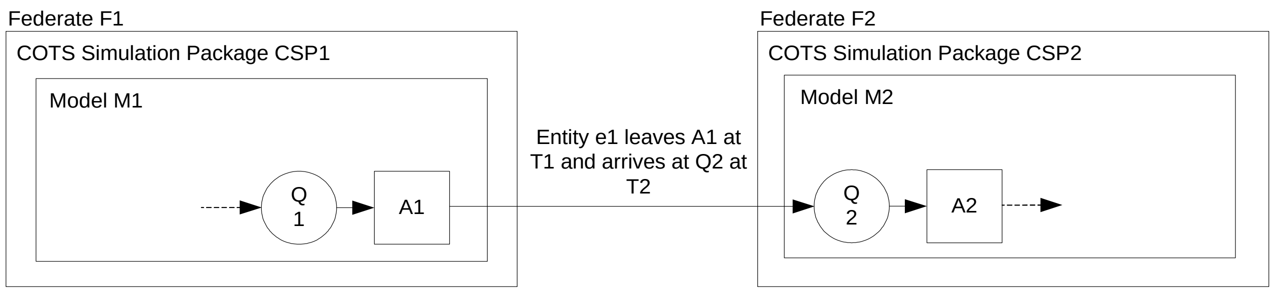

Federate F1
COTS Simulation Package CSP1
Model M1
Q 1
A1
Entity e1 leaves A1 at T1 and arrives at Q2 at T2
Federate F2
COTS Simulation Package CSP2
Model M2
Q 2
A2

Fig. 1. IRM Type A.1: General Entity Transfer.

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 (i.e. some problems may well consist of several IRMs). Most importantly, IRMs are intended to be understandable by modelers, vendors and distributed simulation specialists. Other forms of the models have been developed using a CSPI schema. The diagrammatic form presented here resulted from four years of consultation with stakeholder groups represented by members of the CSPI PDG.

4.1 Interoperability Reference Model Types

There are four different types of IRM. These are:

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 its 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 the above 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

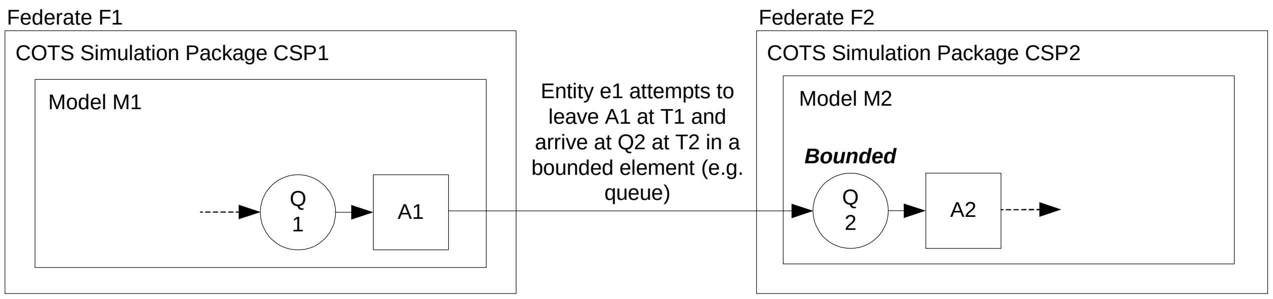

Federate F1
COTS Simulation Package CSP1
Model M1
Q 1
A1
Entity e1 attempts to leave A1 at T1 and arrive at Q2 at T2 in a bounded element (e.g. queue)
Federate F2
COTS Simulation Package CSP2
Model M2
Bounded
Q 2
A2

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

composable, i.e. a distributed simulation that correctly transfers entities from one model to a bounded buffer in another model should be can 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 Sub-types

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 above 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 above for an 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, etc. Note that this inequality must be specified. This is a critical requirement as, if T1 <= T2 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 and therefore will not be able to correctly implement this requirement. In such cases the requirement T1 < T2 must be fulfilled. This sort of detail justifies the need for IRMs as often this level of detail is 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 (i.e. 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***, i.e. 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

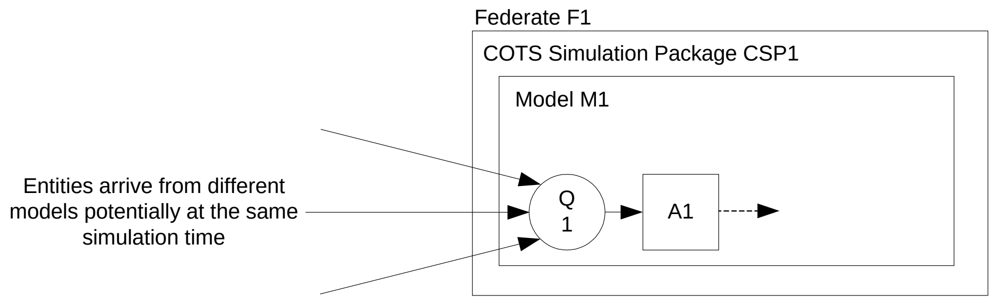

Federate F1
COTS Simulation Package CSP1
Model M1
Entities arrive from different models potentially at the same simulation time
Q 1
A1

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

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 the fact that if A1 is blocked the rest of model M1 still functions as normal, i.e. 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, workstation, 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:

— 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

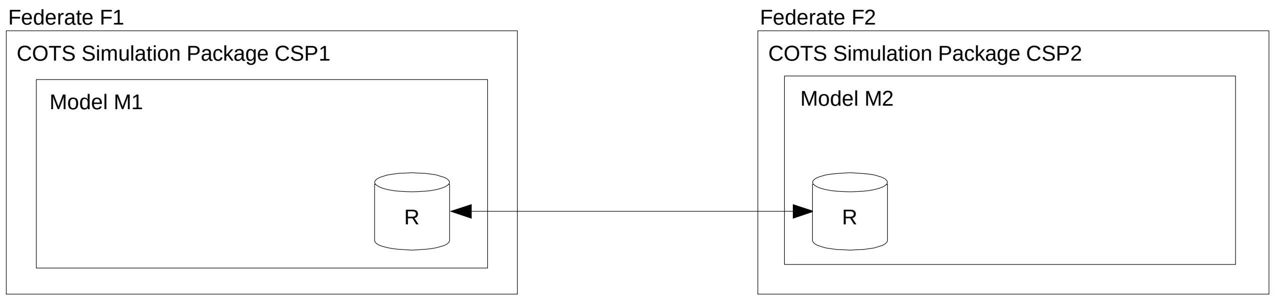
As shown in Figure 3, the IRM Type A.3 Multiple Input Prioritization represents the case where a model element such as queue Q1 (or workstation) can receive entities from multiple places. Let us assume that there are two models M2 and M3 which are capable of sending entities to Q1 and that Q1 has a First-In-First-Out (FIFO) queuing discipline. If an entity e1 is sent from M2 at T1 and arrives at Q1 at T2 and an entity e2 is sent from M3 at T3 and arrives at Q1 at T4, then if T2<T4 we would expect the order of entities in Q1 would be e1, e2. A problem arises when both

entities arrive at the same time, i.e. when T2=T4. Depending on implementation, the order of entities would either be e1, e2 or e2, e1. In some modeling situations it is possible to specify the *priority* order if such a conflict arises, e.g. it can be specified that model M1 entities will always have a higher priority than model M2 (and therefore require the entity order e1, e2 if T2=T4). 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.

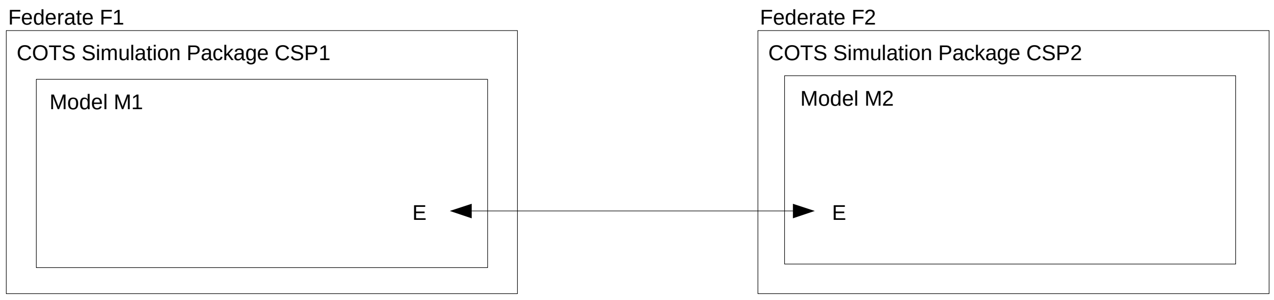
4.6 Interoperability Reference Model Type B: Shared Resource

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 to data and we therefore preserve the distinction in this standard. There is currently one IRM Type B Sub-type, the IRM Type B.1 General Shared Resource.



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.



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

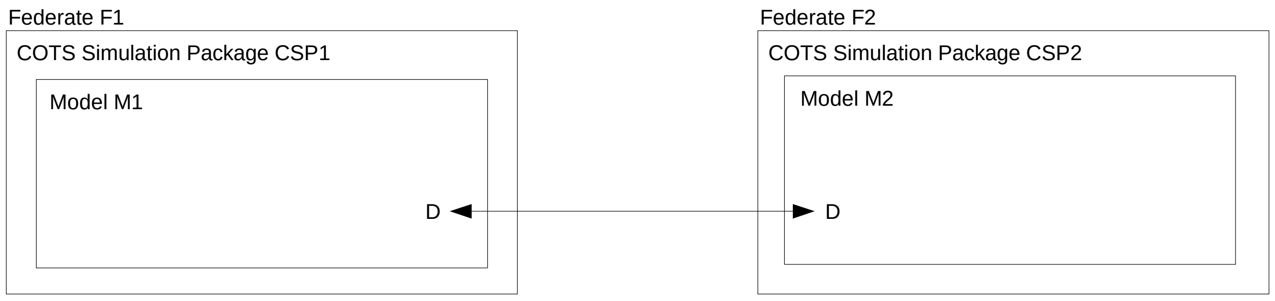
Fig. 5. IRM Type C.1: General Shared Event.

The IRM Type B.1 General Shared Resource represents the case, as outlined above 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.

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 sub-type, 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 if two shared events E1 and E2 are instigated at the same time by different models, then both will occur.



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.

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

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 to 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 Sub-type, the IRM Type D.1 General Shared Data Structure.

IRM Type D.1 General Data Structure represents the case, as outlined above 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 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 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 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 the 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 non-distributed approach is threefold. Firstly, the data used is private and sensitive as it involves information related to clinical practice. Most data in healthcare systems data cannot just be taken on demand and is subject to stringent and length 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 (although these have recently been disbanded). 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, i.e. 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.

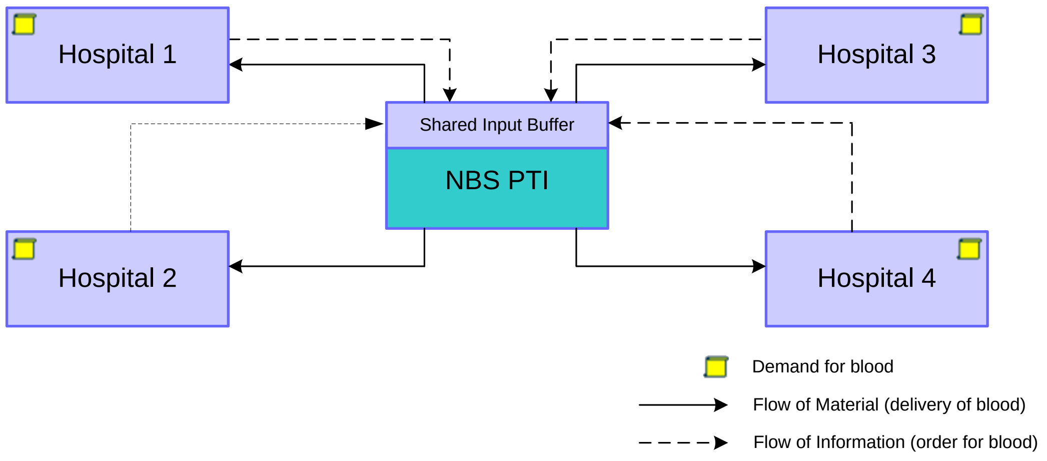


Fig. 7. Information and material flow among the models of the distributed supply chain simulation

Secondly, *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 and 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 is shown in Figure 7. It is at this point where the guidelines became extremely useful as 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.

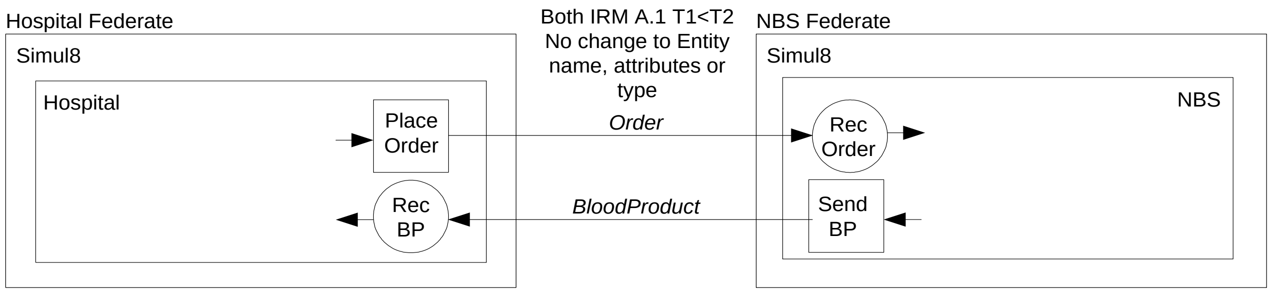


Fig. 8. IRM for National Blood Service Supply Chain Simulation.

In this model there are essentially two entity transfers: (a) orders from a hospital to

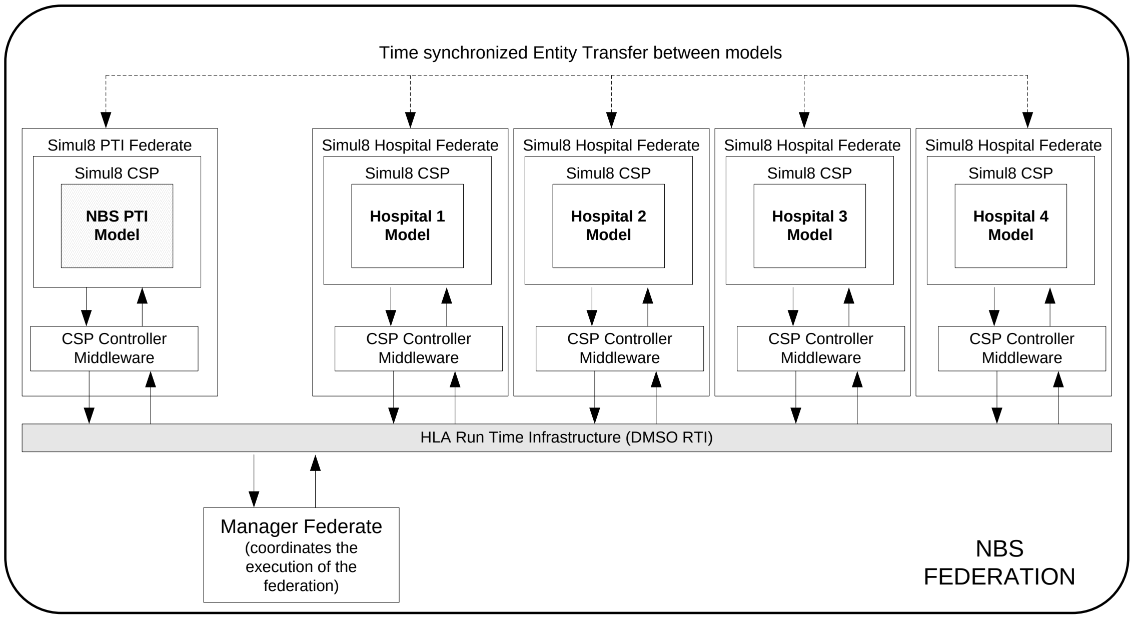

Time synchronized Entity Transfer between models
Simul8 PTI Federate
Simul8 CSP
NBS PTI Model
CSP Controller Middleware
Simul8 Hospital Federate
Simul8 CSP
Hospital 1 Model
CSP Controller Middleware
Simul8 Hospital Federate
Simul8 CSP
Hospital 2 Model
CSP Controller Middleware
Simul8 Hospital Federate
Simul8 CSP
Hospital 3 Model
CSP Controller Middleware
Simul8 Hospital Federate
Simul8 CSP
Hospital 4 Model
CSP Controller Middleware
HLA Run Time Infrastructure (DMSO RTI)
Manager Federate (coordinates the execution of the federation)
NBS FEDERATION

Fig. 9. The NBS Distributed Simulation Federation comprising of one NBS-PTI model, four NBS-Hospital models and one Manager Federate. The CSP Controller Middleware is the interface between the CSP Simul8™ and HLA

the NBS centre, and (b) blood units from the NBS centre to a hospital. IRM Type A.1 General Entity Transfer was therefore appropriate for this work. It was discussed if 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 be held (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 executed 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 we have developed an approach to using an adaptor called the *CSP Controller Middleware (CCM)* (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 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 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"*, *"NBSDeliveryHospital2"*). Further, it shows that each interaction has between five to eight parameters, e.g., product type (RBC or platelets, blood group, delivery type (scheduled delivery or emergency delivery), time when the blood unit expires, etc.

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

Within the planning, design and redesigning of new production facilities at Deere & Co., a leading manufacturer for 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 the planning of new factories as well as for supporting ongoing factory operations. The CSP used in the presented case study was SLX™ [HENRIKSEN 1997].

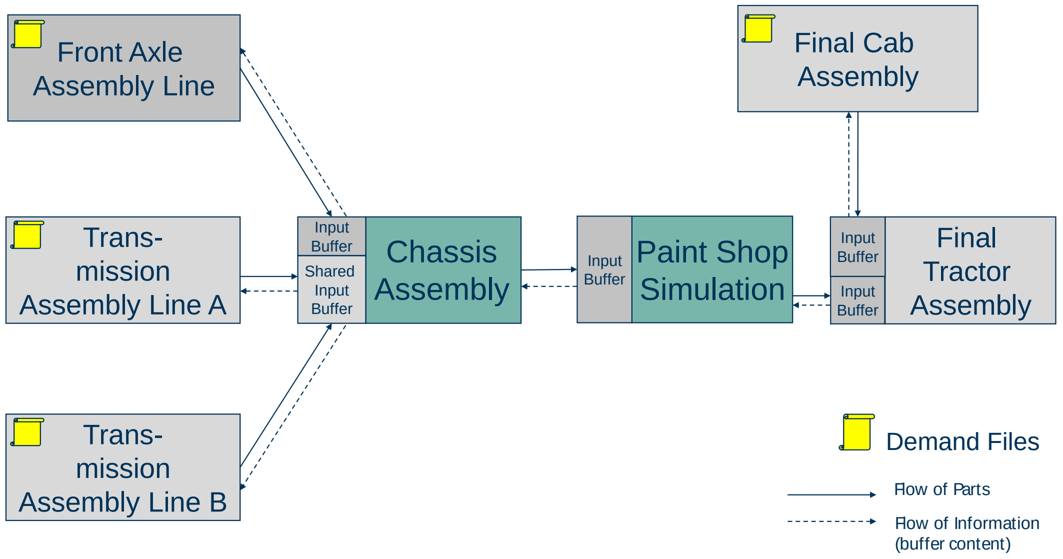


Fig. 10. Information and material flow among the models of the distributed manufacturing simulation

The basis of the case study is a planned tractor factory in South America [Strassburger 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 pre-paint asynchronous assembly lines for chassis, transmissions and front axle 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 simulation frameworks. This yields good results when each section is investigated separately. It does, however, imply the usage 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 which cannot easily be combined within a single CSP for common execution. The IRMs discussed in this paper 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 which must be transferred between the models, all relating to parts which 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. In one case, there was a shared bounded input buffer into which two models deliver parts, but without any prioritization.

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 a IRM Type A.2 entity transfer with $T1<T2$ and $T3 < T4$ in all cases. In one case there was also shared input buffer into which two models deliver entities. This input buffer also qualifies as an IRM A.3 model, but as the scenario does not required input prioritization, this fact had no implications on the required implementation.

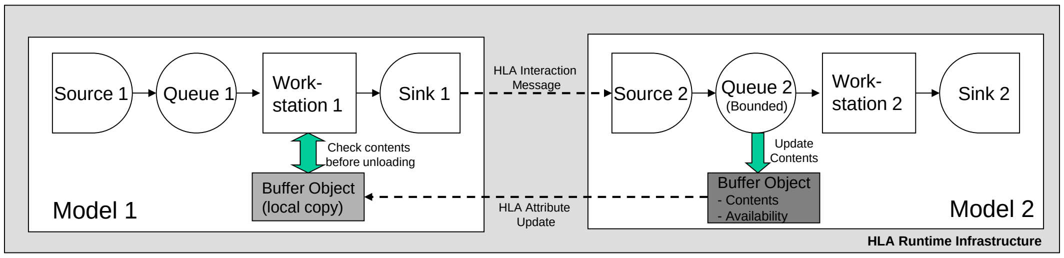
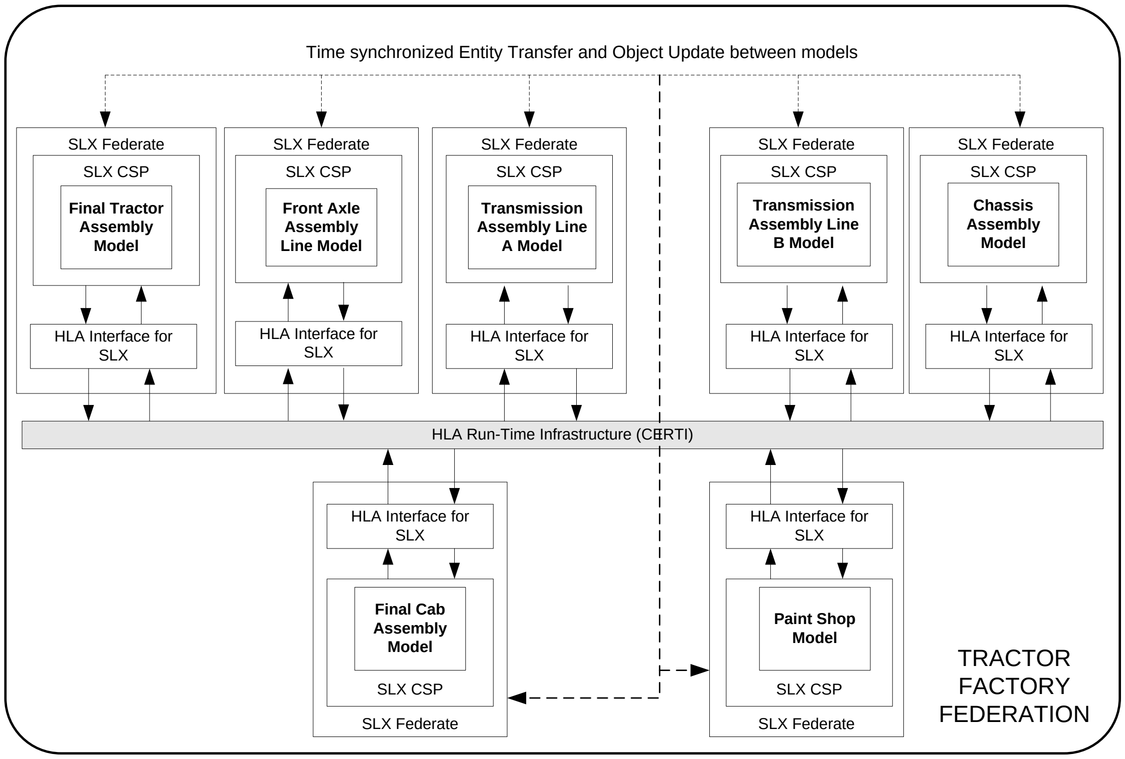


Figure 11: Implementation scheme for IRM A.2 in the manufacturing case study

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 generated for each relevant input buffer (compare 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 necessity to delay an entity transfer depending on the buffer content.

Appendix 2 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 sub-classes. As the root user-defined class, an interaction class called *"TransferEntity"* is specified. It has subclasses which correspond to all potential recipient models, e.g. *"TransferEntityToPaintShop", "TransferEntityToChassisAssembly"*, etc. Further 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 (compare 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 used for the user-defined interaction classes. Thus, we have defined the HLA object class *"UpdateObject"* as the root of user-defined object classes; there are sub-classes "*UpdateObjectChassisAssembly*", "*UpdateObjectPaintShop*", etc. under the root class; these classes further encapsulate the sub-class *"InputBuffer"* (please refer to 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 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 setting illustrates that in this case study it was not speed up or memory constraints that led to the usage of distributed simulation, but the need to integrate different, heterogeneous simulation models. Further implementation details can be found in [Strassburger et al. 2007].



Time synchronized Entity Transfer and Object Update between models
SLX Federate
SLX CSP
Final Tractor Assembly Model
HLA Interface for SLX
Front Axle Assembly Line Model
Transmission Assembly Line A Model
Transmission Assembly Line B Model
Chassis Assembly Model
HLA Run-Time Infrastructure (CERTI)
Final Cab Assembly Model
Paint Shop Model
TRACTOR FACTORY FEDERATION

Fig. 12. The Distributed Simulation of the tractor factory, including seven simulation models all implemented in the CSP SLX™, but in partially different simulation frameworks. Distributed Simulation offered the only way to integrate these models.

7. CONCLUSION

This paper has argued that the development of a single all-encompassing simulation model that represents well-defined sub-systems belonging to multiple functional areas (associated with either one organization, or indeed, related 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 to the above as it addresses concerns among the intra- and inter-organizational stakeholders with regard to data/information sharing and/or security and privacy of shared data/information. The paper 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. Using the IRMs helped reduce the time needed to understand 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 affords a potential solution to concurrent simulation execution both within the confines an organization (e.g., simulation of well-defined sub-processes such as “procurement” and “manufacturing” within an organization) and across organizations (e.g., supply chain simulation without complex non-disclosure agreements and data access policies). However, a significant barrier is the skill set needed to achieving an implementation. Generally in the military simulation community the skills needed to implement a distributed simulation is 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 have 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 paper (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 which 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 research. 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.

ACKNOWLEDGEMENTS

The authors would like to thank the Simulation Interoperability Standards Organization (SISO), the members of the CSPI PDG and the involved CSP vendors for their on-going support of this work.

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-RT2008).* IEEE Computer Society Washington, DC, 300-307.

BANDINELLI, R., RAPACCINI, M., TUCCI, M. AND VISINTIN, F. 2006. Using simulation for supply chain analysis: reviewing and proposing distributed simulation frameworks. *Production Planning & Control*, 17, 2, 167-175.

BOER, C.A., DE BRUIN, A. AND VERBRAECK, A. 2009. A survey on distributed simulation in industry. *Journal of Simulation*, 3, 1, 3-16.

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

BRYANT, R. E. 1977. Simulation of packet communication architecture computer systems, Technical Report MIT/LCS/TR-188, *Massachusetts Institute of Technology, Cambridge, Massachusetts*.

CHANDY, K. M. AND MISRA, J. 1979. Distributed simulation: A case study in design and verification of distributed programs. IEEE Transactions on Software Engineering, 5, 5, 440-452.

FUJIMOTO, R.M. 2000. *Parallel and distributed simulation systems*. New York, NY: John Wiley & Sons.

FUJIMOTO, R.M. 1990. Parallel discrete event simulation. *Communications of the 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, Washington, DC, 60-67.

GAN, B.P. AND TURNER, S.J. 2000. An asynchronous protocol for virtual factory simulation on shared memory multiprocessor systems. *Journal of Operational Research Society,* 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. *Journal of Simulation*, 3: 114-128.

GLOBALSECURITY.ORG. 2002. Millennium Challenge. *Available online http://www.globalsecurity.org/military/ops/millennium-challenge.htm [last accessed 26 February, 2012].*

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. *M&S CO Project No. 06OC-TR-001.*

HENRIKSEN, J.O. 1997. An Introduction to SLX™. In *Proceedings of the 1997 Winter Simulation Conference*, SCS, Atlanta, pp.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. *Journal of the Operational Research Society*, 57, 12, 1383-1399.

IEEE. 2010. IEEE 1516-2011 *IEEE standard for modeling and simulation (M&S) high level architecture (HLA).* New York, NY: Institute of Electrical and Electronics Engineers.

IEEE. 2011. IEEE 1730-2011 *IEEE standard for distributed simulation engineering and execution process (DSEEP).* New York, NY: Institute of Electrical and Electronics Engineers.

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 2001 Fall Simulation Interoperability Workshop*. Simulation Interoperability Standards Organization, Orlando, Florida, USA, 01F-SIW-038.

KATSALIAKI, K. AND BRAILSFORD, S. 2007. Using simulation to improve the blood supply chain. *Journal of the Operational Research Society*, 58, 2, 219-227.

KUHL F., WEATHERLY R. AND DAHMANN J. 1999. *Creating computer simulation systems: an introduction to the high level architecture*. Upper Saddle River, NJ: Prentice Hall PTR.

LEE, S., ZHAO, X., SHENDARKAR, A., VASUDEVAN, K. AND SON, Y.-J. 2008. Fully dynamic epoch time synchronisation method for distributed supply chain simulation. *International Journal of Computer Applications in Technology,* 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, pp. 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. *Journal of Simulation*, 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 2010*, art. no. 5566100.

MALINGA, L. AND LE ROUX, W. H. 2009. HLA RTI performance evaluation. In *Proceeding of the European Simulation Interoperability Workshop, Istanbul, Turkey*, pp. 1-6.

MERTINS, K. , RABE, M. AND JÄKEL, F-W. 2005. Distributed modeling and simulation of supply chains. *International Journal of Computer Integrated Manufacturing*, 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 2006 Operational Research Society Simulation Workshop (SW06),* Operational Research Society, Birmingham, UK, pp. 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. *SIMULATION: Transactions of the Society of Modeling and Simulation International*. Volume 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, pp. 1136-1142.

NOULARD, E., ROUSSELOT, J-Y. AND SIRON, P. 2009. CERTI, an Open Source RTI, Why and How. In *Proceeding of the Spring Simulation Interoperability Workshop, 23-27 March 2009, San Diego, United States*, 09S-SIW-015.

PAN, K., S.J. TURNER, W. CAI, AND Z. LI. 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, Washington DC.

PIDD, M. (2004). *Computer simulation in management science (5th edition).* Chichester, UK: John Wiley & Sons.

RAAB, M., SCHULZE, T., AND STRASSBURGER, S. 2008. Management of HLA-based distributed legacy SLX-Models. In: *Proceedings of the 2008 Winter Simulation Conference*, ACM Press, New York, NY, pp. 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, pp. 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. *Computers and Chemical Engineering*, 32, 6, 1195–1205.

SISO. 2006. SISO-STD-003-2006 Base Object Model (BOM) Template Specification.Simulation Interoperability Standards Organization, Orlando, Florida.

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

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

STRASSBURGER, S. The Road to COTS-Interoperability: From Generic HLA-Interfaces Towards Plug-And-Play Capabilities. In *Proceedings of the 2006 Winter Simulation Conference*, ACM Press, New York, NY, pp. 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 2008 Winter Simulation Conference*, ACM Press, New York, NY, pp, 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, pp. 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, pp. 603-609.

STRASSBURGER, S. AND TAYLOR, S. J. E. 2012. A comparison of the CSPI and CMSD standards. In *Proceedings of the 2012 Spring Simulation Interoperability Workshop. Orlando, USA,* March 26-30, 2012.

TAYLOR, S. J. E., SUDRA, R., JANAHAN, T., TAN, G. AND LADBROOK, J. 2002. GRIDS-SCF: an infrastructure for distributed supply chain simulation. *Simulation: Transactions of the Society of Modeling and Simulation International*, 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 2005)*, IEEE Computer Society, Washington, DC, 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. *Simulation: Transactions of the Society of Modeling and Simulation International,* 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 Transactions on Systems, Man and Cybernetics: Part A*, 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 2007 Winter Simulation Conference*, ACM Press, New York, NY, pp. 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, pp. 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 2011 Winter Simulation Conference, ACM Press, New York, NY, pp. 161-175.

TOLK, A., DIALLO, S. Y. AND TURNITSA, C. D. 2007. Applying the levels of conceptual interoperability model in support of integratability, interoperability, and composability for system-of-systems engineering. Journal on Systemics, Cybernetics and Informatics, 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,USA.*

UYGUN, O., ÖZTEMEL, E. AND KUBAT, C. 2009. Scenario based distributed manufacturing simulation using HLA technologies. *Information Sciences*, 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, Washington, DC, USA, pp. 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*, pp. 1333-1340.

APPENDIX 1 - FED FILE FOR THE NATIONAL BLOOD SERVICE CASE STUDY

```
(FED
  (Federation NBS_SupplyChainSimulaion)
  (FEDversion v1.3)

  ;; no routing spaces are defined
  (spaces)

  ;; OBJECT CLASSES
  ;; Class ObjectRoot and its two sub-classes RTIprivate and Manager are required.
  ;; Parameters associated with these classes/sub-classes are not shown for brevity
  (objects
    (class ObjectRoot
      (attribute privilegeToDelete reliable timestamp)
      (class RTIprivate)
      (class Manager)

    ;; WE DO NOT DEFINE ANY USER-DEFINED OBJECT CLASSES SINCE WE USE ONLY
INTERACTIONS

    ;; End of ObjectRoot
    )
  ;; End of objects
  )


  ;; INTERACTION CLASSES
  ;; Class InteractionRoot and its two sub-classes RTIprivate and Manager are required.
  ;; Parameters associated with these classes/sub-classes are not shown for brevity
  (interactions
    (class InteractionRoot reliable receive
      (class RTIprivate reliable receive)
      (class Manager reliable receive)

  ;;  USER-DEFINED INTERACTION CLASSES
      (class Hospital1Orders reliable timestamp
        (parameter Hospital1Product)
                (parameter Hospital1Group)
                (parameter Hospital1Deltype)
                (parameter Hospital1Wheretogo)
                (parameter Hospital1Usage)
      )
      (class Hospital2Orders reliable timestamp
        (parameter Hospital2Product)
                (parameter Hospital2Group)
                (parameter Hospital2Deltype)
                (parameter Hospital2Wheretogo)
                (parameter Hospital2Usage)
      )
      (class Hospital3Orders reliable timestamp
        (parameter Hospital3Product)
                (parameter Hospital3Group)
                (parameter Hospital3Deltype)
                (parameter Hospital3Wheretogo)
                (parameter Hospital3Usage)
      )
      (class Hospital4Orders reliable timestamp
        (parameter Hospital4Product)
                (parameter Hospital4Group)
                (parameter Hospital4Deltype)
                (parameter Hospital4Wheretogo)
                (parameter Hospital4Usage)
      )
      (class NBSDeliveryHospital1 reliable timestamp
        (parameter NBSProductH1)
                (parameter NBSGroupH1)
```

```
            (parameter NBSDeltypeH1)
            (parameter NBSWheretogoH1)
            (parameter NBSUsageH1)
            (parameter NBSTimeenteredsystemH1)
            (parameter NBSExpirationtimeH1)
            (parameter NBSExpirationrulehospH1)
      )
      (class NBSDeliveryHospital2 reliable timestamp
        (parameter NBSProductH2)
            (parameter NBSGroupH2)
            (parameter NBSDeltypeH2)
            (parameter NBSWheretogoH2)
            (parameter NBSUsageH2)
            (parameter NBSTimeenteredsystemH2)
            (parameter NBSExpirationtimeH2)
            (parameter NBSExpirationrulehospH2)
      )
      (class NBSDeliveryHospital3 reliable timestamp
        (parameter NBSProductH3)
            (parameter NBSGroupH3)
            (parameter NBSDeltypeH3)
            (parameter NBSWheretogoH3)
            (parameter NBSUsageH3)
            (parameter NBSTimeenteredsystemH3)
            (parameter NBSExpirationtimeH3)
            (parameter NBSExpirationrulehospH3)
      )
      (class NBSDeliveryHospital4 reliable timestamp
        (parameter NBSProductH4)
            (parameter NBSGroupH4)
            (parameter NBSDeltypeH4)
            (parameter NBSWheretogoH4)
            (parameter NBSUsageH4)
            (parameter NBSTimeenteredsystemH4)
            (parameter NBSExpirationtimeH4)
            (parameter NBSExpirationrulehospH4)
      )

    ;; End of InteractionRoot
    )
  ;; End of Interactions
  )
;; End of FED file
)
```

APPENDIX 2 - FED FILE FOR THE TRACTOR FACTORY CASE STUDY

```
(FED
 (Federation TractorFactorySimulaion)
 (FEDversion v1.3)

 ;; no routing spaces are defined
 (spaces)

 ;; OBJECT CLASSES
 ;; Class ObjectRoot and its two sub-classes RTIprivate and Manager are required.
 ;; Parameters associated with these classes/sub-classes are not shown for brevity
 (objects
   (class ObjectRoot
     (attribute privilegeToDelete reliable timestamp)
     (class RTIprivate)
     (class Manager)

   ;; USER-DEFINED OBJECT CLASSES
     (class UpdateObject
       (class UpdateObjectChassisAssembly
         (class InputBuffer
           (attribute BufferName reliable timestamp)
           (attribute ModelName reliable timestamp)
           (attribute Content reliable timestamp)
           (attribute Available reliable timestamp)
         )
        )
       (class UpdateObjectPaintShop
         ;; Attributes for sub-class Input Buffer not repeated
         (class InputBuffer …. )
        )
       (class UpdateObjectFinalTractorAssembly
         (class InputBuffer …. )
       )
    ;; End of UpdateObject
    )
   ;; End of ObjectRoot
   )
 ;; End of objects
 )

 ;; INTERACTION CLASSES
 ;; Class InteractionRoot and its two sub-classes RTIprivate and Manager are required.
 ;; Parameters associated with these classes/sub-classes are not shown for brevity
 (interactions
   (class InteractionRoot reliable receive
     (class RTIprivate reliable receive)
     (class Manager reliable receive)

 ;;  USER-DEFINED INTERACTION CLASSES
    (class TransferEntity reliable timestamp
      (class TransferEntityToChassisAssembly reliable timestamp)
        (class TransferEntityTransmissionAssemblyAToChassisAssembly reliable timestamp
          (parameter Entity)
        )
        (class TransferEntityTransmissionAssemblyBToChassisAssembly reliable timestamp
          (parameter Entity)
        )
        (class TransferEntityFrontAxleAssemblyToChassisAssembly reliable timestamp
          (parameter Entity)
        )
      )
      (class TransferEntityToPaintShop reliable timestamp)
        (class TransferEntityChassisAssemblyToPaintShop reliable timestamp
          (parameter Entity)
        )
        )
        (class TransferEntityToFinalTractorAssembly reliable timestamp)
          (class TransferEntityFinalCabAssemblyToFinalTractorAssembly reliable timestamp
            (parameter Entity)
          )
          (class TransferEntityPaintShopToFinalTractorAssembly reliable timestamp
            (parameter Entity)
          )
        )
    ;; End of TransferEntity
    )
   ;; End of InteractionRoot
   )
  ;; End of Interactions
  )
;; End of FED file
)
```