Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, eds.

COMMERCIAL-OFF-THE-SHELF SIMULATION PACKAGE INTEROPERABILITY: ISSUES AND FUTURES

Simon J. E. Taylor
Navonil Mustafee

Centre for Applied Simulation Modelling
School of Information Systems, Computing & Maths
Brunel University
Uxbridge, Middlesex, UB8 3PH, UK

Stephen J. Turner
Ke Pan

Parallel & Distributed Computing Centre
School of Computer Engineering
Nanyang Technological University
Singapore 639798, SINGAPORE

Steffen Strassburger

School of Economic Sciences
Technical University of Ilmenau
Helmholtzplatz 3
98693 Ilmenau, GERMANY

ABSTRACT

Commercial-Off-The-Shelf Simulation Packages (CSPs) are widely used in industry to simulate discrete-event models. Interoperability of CSPs requires the use of distributed simulation techniques. Literature presents us with many examples of achieving CSP interoperability using bespoke solutions. However, for the wider adoption of CSP-based distributed simulation it is essential that, first and foremost, a standard for CSP interoperability be created, and secondly, these standards are adhered to by the CSP vendors. This advanced tutorial is on an emerging standard relating to CSP interoperability. It gives an overview of this standard and presents case studies that implement some of the proposed standards. Furthermore, interoperability is discussed in relation to large and complex models developed using CSPs that require large amount of computing resources. It is hoped that this tutorial will inform the simulation community of the issues associated with CSP interoperability, the importance of these standards and its future.

1. INTRODUCTION

In Operational Research & Management Science (ORMS), Discrete-Event Simulation (DES) has been used to analyze production and logistics problems in many areas such as commerce, defense, health, manufacturing and logistics for many years (Law, 2007). The first DES languages appeared in the late 1950s. These evolved during the 1960s and 1970s. In the context of simulation practice in industry, although programming languages may be used to build simulations in certain circumstances, models are generally created using commercially available simulation packages (Robinson 2005). With the arrival of the IBM PC, the 1980s saw the rise of Visual Interactive Modeling Systems (VIMS) that allowed simulation modelers to visually create and simulate discrete-event models. VIMS enable users to create models in a graphical environment through an interactive "click-and-drag" selection of pre-defined simulation objects (entry points, queues, workstations, resources, etc.) and link them together to represent the underlying logical interactions between the entities they represent (Pidd 2004). These have matured into the Commercial-off-the-shelf (COTS) Simulation Packages (CSPs) that are very familiar to ORMS practitioners today. They include Arena™ (Rockwell automation), Anylogic™ (XJ technologies), Flexsim™ (Flexsim Software Products, Inc.), Simul8™ (Simul8 corporation), Witness™ (Lanner group), etc. Each has a wide range of functionality including visual model building, simulation run support, animation, optimization and virtual reality. Some have their own dedicated programming language and all are able to be linked to other COTS software (such as Microsoft Excel). However, the vast majority of these packages do not support CSP package interoperability.

In the context of the CSPs, interoperability can be seen from two perspectives – *homogenous CSP interoperability* (interoperability between models developed using identical CSPs) and *heterogeneous CSP interoperability* (interoperability between models created using different CSPs). There are approximately 33 DES CSPs that are available for use by the

ORMS researchers (Mustafee 2007). In order to realise both *homogenous and heterogeneous CSP interoperability* it is essential that the CSP vendors that implement these products adhere to some standard. This advanced tutorial will present an emerging standard on CSP interoperability and will discusses the issues and the futures.

The rest of the paper is organised as follows. In section 2 we discuss the enabling simulation theory (distributed simulation) and the enabling technology (distributed simulation middleware) for CSP interoperability and outline some motivations for distributed simulation of the CSPs. The emerging standard for interoperating CSP models is discussed next (section 3), followed by two case studies that present the implementation of some of these standards in sections 4.1 and 4.2 respectively. Large and complex models created using CSPs can potentially benefit from large amounts of computing resources, and this is discussed in relation to CSP interoperability and Grid computing in section 5. A case study is also presented (section 5.2) that extends an earlier case study (section 4.1) with an objective to execute a CSP-based distributed simulation, conferring to the emerging CSP interoperability standards, over Grid resources. Section 6 then discusses the future of the standard and section 7 makes some concluding remarks and draws the paper to a close.

2. DISTRIBUTED SIMULATION AND CSP INTEROPERABILITY

In a distributed simulation, a large computer model is typically executed over several processors. These processors can be a part of a multiprocessor computer or may belong to multiple PCs that are connected over a network. Parallel Discrete Event Simulation (PDES) usually refers to the execution of such distributed DES on parallel and distributed machines (Page and Nance 1994). Fujimoto (2003) uses the term distributed simulation to refer to both the parallel and distributed variants of PDES. The rationale presented is that, although historically, the terms “distributed simulation” and “parallel simulation” referred to geographically distributed simulations and simulations on tightly coupled parallel computers respectively, new distributed computing paradigms like clusters of workstations and Grid computing has made this distinction less obvious. This advanced tutorial takes a similar view and therefore does not distinguish between the parallel and distributed variants of PDES. We will therefore use “*distributed simulation*” to refer to the execution of simulations on both multiprocessor machines and over a network of PCs.

Some of the motivations for using distributed simulation are as follows (Fujimoto 1999; Fujimoto 2003) - (a) Distributed simulation can facilitate model reuse by “hooking together” existing simulations into a single simulation environment, thereby potentially alleviating the cost of creating new models; (b) a large simulation may have memory and processing requirements that cannot be provided by a single system and, thus, distributing the simulation execution across multiple machines may allow the memory and processors of many computer systems to be utilized; (c) executing simulations on a set of geographically distributed computers facilitates wider user participation in the simulation experiments. This also alleviates the cost and time that is normally associated with bringing participants to one physical place in, for example a joint training exercise or decision making in a supply chain. The motivations for enabling *homogenous* and *heterogeneous* CSP interoperability using distributed simulation are similar.

Distributed simulation requires the use of a distributed simulation middleware. A distributed simulation middleware is a software component that implements the distributed simulation algorithms to achieve synchronisation between individual running simulations. Furthermore, the middleware enables the various executing models to exchange information between them. Middleware such as HLA-RTI (IEEE 2000), FAMAS (Boer 2005), GRIDS (Taylor et al. 2002) and CSPE-CMB (Mustafee 2004) can be used to facilitate distributed execution of CSP-based simulations.

The simple act of linking together, or interoperating, two or more CSPs and their models, can be extremely complex. This is due to time synchronization requirements and the complexity of distributed simulation algorithms and/or software used to create the link (Fujimoto 2000). This complexity can often hide the precise nature of what is being shared between these interoperating CSPs. There are several published examples of how distributed simulations have been created with CSPs. The first major work in the area was done by Straßburger (2001). Various strategies, implementations and applications have been investigated since then to create ORMS distributed simulations with CSPs, for example, McLean and Riddick (2000); Mertins, Rabe, and Jäkel (2000); Rabe and Jäkel (2001); Borshchev, Karpov, and Kharitonov (2002); Hibino et al. (2002); Revetria, Blomjous, and Van Houten (2003); Gan et al. (2005); Lendermann et al. (2005); Taylor et al. (2005); Mustafee and Taylor (2006); Strassburger (2006); Taylor et al. (2006ab); Katsaliaki et al. (2009); Mustafee et al. (2009). This work, while contributing to the development of CSP interoperability, presents a problem. There is a variety of different approaches to distributed simulation with CSPs. As argued in Taylor (2003), a common approach is needed. To standardise this approach, an international standardisation group, Simulation Interoperability Standards Organization’s COTS Simulation Package Interoperability Product Development Group (CSPI PDG) <http://www.sisostds.org/>, have produced a set of draft standards in this area and are described initially in Taylor et al. (2006b). The distributed approach discussed in this advanced tutorial is based on the standards being developed by the CSPI PDG and can be found in full in Taylor, Turner, and Strassburger (2008).

3. INTEROPERABILITY REFERENCE MODELS

To simplify the process of generating a distributed simulation, the CSPI PDG has created a standardized set of Interoperability Reference Models or “interoperability design patterns” that attempt to capture these subtleties. 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 distributed supply chain 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 distributed hospital 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 are 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 particular CSP elements. Where appropriate, IRMs specify time synchronization requirements and present alternatives. 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 *simulation developers, CSP vendors and technology solution providers.*

3.1 Interoperability Reference Model Types

There are currently 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* leaving one model and arriving 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. We now introduce the most commonly used IRM – The IRM Type A Entity Transfer. For the purposes of this advanced tutorial, we discuss IRM Type A in sufficient detail that would facilitate the understanding of the case studies presented in section 4.

IRM Type A Entity Transfer represents interoperability problems that can occur when transferring an entity from one model to another. There are currently three IRM Type A Sub-types. These are:

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

For the purposes of this advanced tutorial, discussions pertaining to IRM Type A.1 and IRM Type A.2 will suffice. These are discussed next.

3.2 IRM Type A.1 General Entity Transfer

IRM Type A.1 General Entity Transfer represents the case, 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. This IRM is inclusive of cases where there are many models and many entity transfers (all transfers are instances of this IRM). This IRM does not include cases where (a) the receiving element is bounded (IRM Type A.2), and (b) 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.

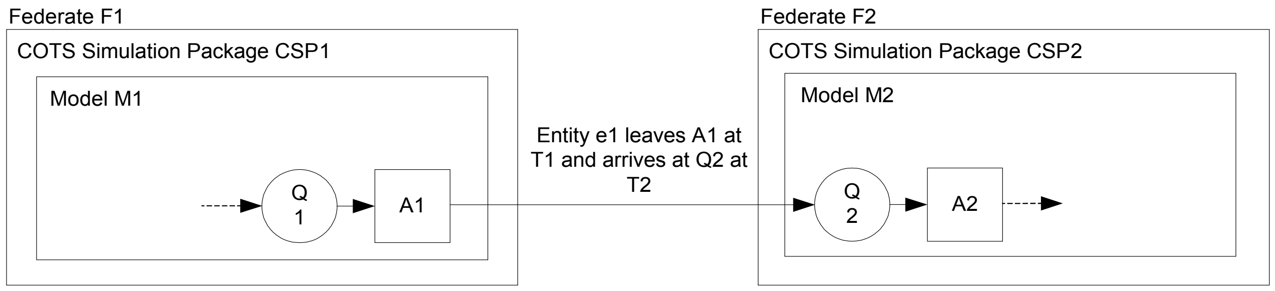
Federate F1
COTS Simulation Package CSP1
Model M1
Q 1
A1
Entity e1 leaves A1 at T1 and arrives at Q2 at T2
Federate F2
COTS Simulation Package CSP2
Model M2
Q 2
A2

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

3.3 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 shows an illustrative example, where an entity e1 attempts to leave model M1 at T1 from activity A1, 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. This IRM does not include cases where multiple inputs need to be prioritized (IRM Type A.3 – this is not discussed in this tutorial). Finally, 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 Ob to become full again during the interval if other inputs to Ob are allowed.

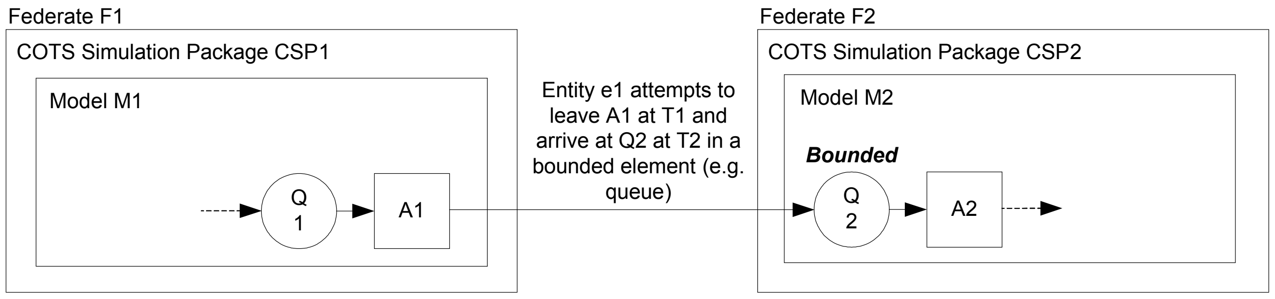
Federate F1
COTS Simulation Package CSP1
Model M1
Q 1
A1
Entity e1 attempts to leave A1 at T1 and arrive at Q2 at T2 in a bounded element (e.g. queue)
Federate F2
COTS Simulation Package CSP2
Model M2
Bounded
Q 2
A2

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

4. CASE STUDIES

Two case studies are presented in this section - the *National Blood Service Simulation* (section 4.1) and the *Distributed Manufacturing Simulation* (section 4.2) - the former an implementation of the *IRM Type A.1 General Entity Transfer* and the latter implementing *IRM Type A.2 Bounded Receiving Element*.

4.1 National Blood Service (NBS) Simulation

Katsaliaki and Brailsford (2007) describe experiences with the use of DES to investigate the supply chain of blood from an UK National Blood Service (NBS) Processing, Testing and Issuing (PTI) centre to hospitals in the Southampton area of the UK. The CSP used in this research was Simul8™. The purpose of the study was to achieve improved inventory control, ordering and distribution policies, and to bring less wastage, less shortage and provide better quality of service. This was done by reconfiguring the processes and parameters of the blood supply chain. A problem identified in this work was that as the system being modeled grew in size and complexity, the time taken to perform one simulation run increased to a point that made the use of simulation infeasible. Before we present the experimental results of the execution of the NBS PTI model, we would like to highlight the reason for the increase in the complexity of the model.

The size and the complexity increased with addition of more hospitals into the model. Addition of more hospitals was necessary since each NBS PTI center in the UK has the responsibility to cater to the blood requirements of multiple hospitals under its jurisdiction. For example, the NBS PTI centre serving the Southampton area has the responsibility of serving 16 hospitals. Thus, the Simul8™ model had to grow in size to model the blood supply chain of the Southampton area. Our experiments showed that the conventional model (model executed using CSP Simul8™ on one computer) with one hospital took approximately 14 minutes to run for a whole simulated year. The run time rose to 78 minutes when the model ran with two hospitals and to approximately 17.5 hours with three hospitals. The addition of the fourth hospital increased the execution time to 35.8 hours. Thus, it was decided to restructure the conventional NBS simulation model (Figure 3), such that different logical parts of the model could be executed over several computers (each model being simulated over a local copy of Simul8™) using DMSO RTI (DMSO 1998) middleware for distributed simulation (Figure 4). The natural "division" in the model was between the hospitals and the NBS PTI center. Furthermore, the interoperability requirements of the NBS distributed simulation are compliant with IRM Type A.1 with T1<T2 in all cases.

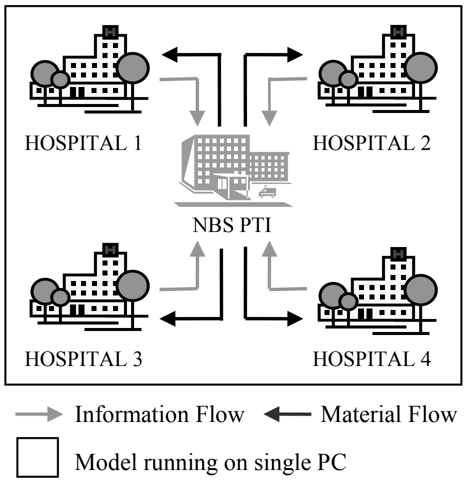

Figure 3: NBS Conventional Model with the NBS PTI and Four Hospitals

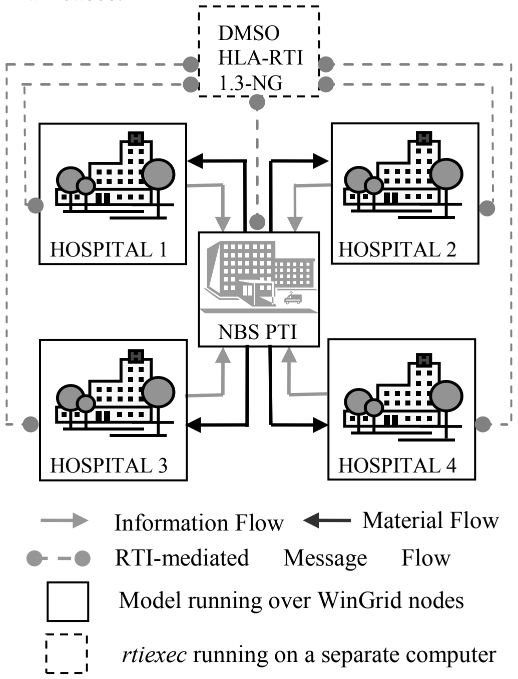
DMSO
HLA-RTI
1.3-NG
HOSPITAL 1
HOSPITAL 2
NBS PTI
HOSPITAL 3
HOSPITAL 4
Information Flow
Material Flow
RTI-mediated Message Flow
Model running over WinGrid nodes
rtiexec running on a separate computer

Figure 4: NBS Distributed Model with NBS PTI and Four Hospitals

Dividing the conventional NBS PTI model resulted in individual federates, each simulating a part of the model in a locally installed copy of Simul8™. Together these federates formed a federation that interacted by time stamped messages that represent the interaction of one model part with another (e.g., when an entity leaves one part of a model and arrives at another). Entities are represented using the CSPI Entity Transfer Specification (Taylor et al. 2006a) and exchanged using

HLA interactions. Our distributed model with one NBS supply centre and one hospital ran in approximately 8.5 hours, with two hospitals in 9.8 hours, with three hospitals in 12.7 hours and with four hospitals in 16.5 hours. Comparing the results of the standalone and the distributed NBS model we see that that the distributed model performs better when the model grows in size and complexity. The opposite of this is also true (the standalone models executed many times faster in case of one and two hospitals). Our work has shown the feasibility of using distributed simulation when the models are large and complex. Furthermore it has shown the utility of the IRM Type A.1 General Entity Transfer. Further information on this work and the implementation of the NBS distributed simulation can be found in Mustafee et al. (2009) and Katsaliaki et al. (2009).

4.2 Distributed Manufacturing Simulation

Deere & Company (founded in 1837 and collectively called John Deere) is one of the world's leading manufacturers for agricultural, forestry, and construction equipment. It does business around the world and employs approximately 56,000 people. Within the planning, design and redesigning of new production facilities, DES models are routinely used to simulate the behaviour of the systems under investigation. 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™.

The basis of the case study is a planned tractor factory in South America (Strassburger, Schulze, and Lemessi 2007; Raab, Schulze, and Strassburger 2008). 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 5: 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 material flow of the overall production system is shown in Figure 5.

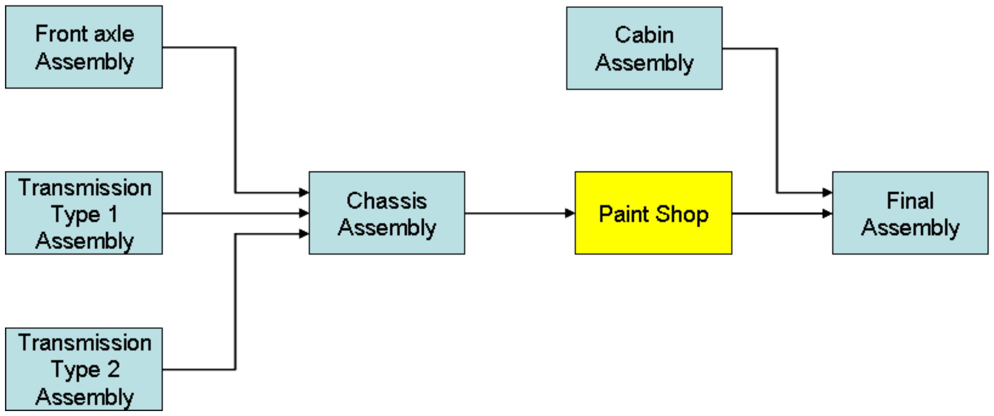

Figure 5: Schematic View of the Production System

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 suggested method to combine the different models was therefore the usage of distributed simulation. The motivation for using distributed simulation in this example is to integrate different independently developed *existing* models which cannot easily be combined within a single CSP for common execution. This differs from the motivation given in the UK National Blood Service (NBS) example, where the motivation for using distributed simulation was to gain speed-up (Mustafee et al. 2009).

The IRMs described in this paper helped to identify the interoperability requirements of the scenario in a common “language”. 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. There is a unique input buffer for each of the sending models. We can therefore quite clearly state: The interoperability requirements of the distributed manufacturing simulation are compliant with IRM Type A.2 with T1<T2 and T3<T4 in all cases.

In the implementation of the distributed simulation, a modified version of Entity Transfer Specification also under discussion in the CSPI PDG (Taylor et al. 2006a) was used. We applied the suggested interaction class hierarchy for transferring entities along the different possible routes between the models. In addition, we also had to model input buffers of the receiving models to decide whether a sending model could actually transfer an entity or whether it had to block. Input buffers were modeled by the receiving models using HLA object instances. Sending models had to subscribe to the input buffer relevant to them and create a local "copy" reflecting its current state. Please note that a correct and generally valid implementation of a IRM Type A.2 problem comes with a zero lookahead requirement, as changes of the input buffer contents have to be taken into account immediately in the subscribing models. Only under certain circumstances which are not discussed here this requirement can be relaxed (see Strassburger, Schulze, and Lemessi 2007). Figure 6 shows a snapshot of the visualization of the overall distributed manufacturing simulation including the color-coded state of the input buffers.

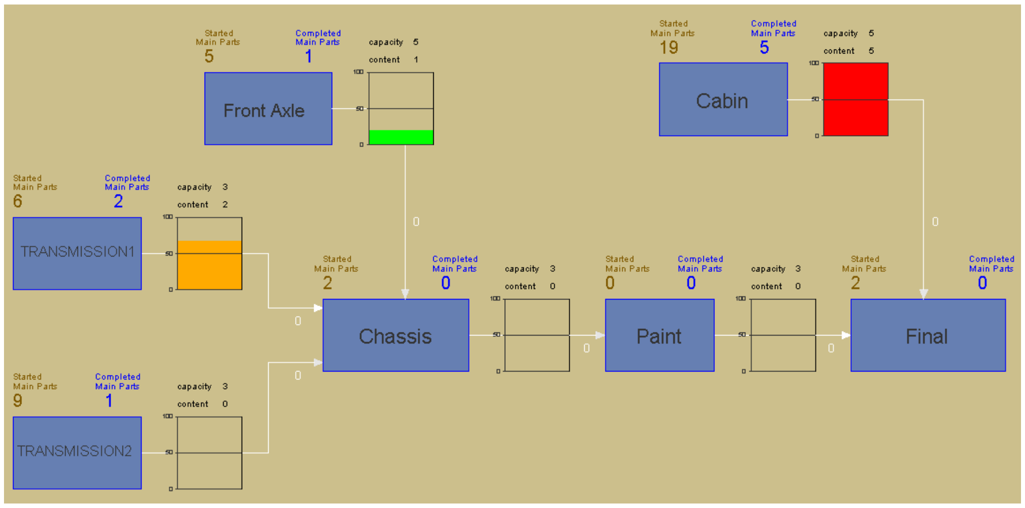

Figure 6: Visualization of the distributed manufacturing simulation including input buffer fill levels

Experience with this case study has shown that the formal classification of typical interoperability problems as it is done in the IRMs has significantly simplified the discussion between project members and helped to capture the specific interoperability requirements of the application scenario.

5. CSP INTEROPERABILITY OVER THE GRID

Distributed simulations constructed from components developed using COTS Simulation Packages (CSPs) often require large amounts of computing resources. Moreover, the data sets used by such simulations may be geographically distributed. In order to cater for the increasing complexity of such systems, it is often convenient to harness distributed resources on a Local Area Network or even over the Internet. The emergence of Grid technologies provides exciting new opportunities for both the development and execution of large scale distributed simulations, enabling the use of distributed computing resources, while also facilitating access to geographically distributed data sets.

Distributed simulation over the Grid also supports information hiding. For example, in recent years, supply chain modeling has evolved from a single enterprise with multiple facilities to one that crosses enterprise boundaries. Applying simulation in designing, evaluating, and optimizing the supply chain becomes more difficult since the participating companies may not be willing to share their simulation models with partners. Distributed simulation over the Grid allows each participating company to run their own simulation model (typically developed using a CSP) at their own site. Detailed model information is encapsulated within the company itself and the participating companies only need to define essential information flows from one model to another.

A number of approaches to interoperating CSPs have been based on the High Level Architecture (HLA), an IEEE standard for distributed simulation (IEEE 2000). Traditionally, HLA-based distributed simulations are conducted using a vendor-specific RTI software and federates with different RTI versions cannot cooperate with each other. To run a distributed simulation over the Internet, the required software and hardware resource arrangements and security settings must be made before the actual simulation execution. Because of this inflexibility, it is not easy to run HLA-based distributed simulations across administrative domains. To address these inflexibility issues and leverage globally pervasive resources for distributed simulations, the Grid is naturally considered as a solution.

Grid computing was proposed by Foster as flexible, secure and coordinated resource sharing among dynamic collections of individuals, institutions and resources (Foster, Kesselman, and Tuecke 2001). Among the various available Grid middlewares, Globus Toolkit (Globus 2008) is the de facto standard middleware for Grid computing. Its latest version GT4 contains five components, namely Common Runtime, Security, Data Management, Information Services and Execution Management, to facilitate heterogeneous resource sharing.

Three approaches can be defined for HLA-based distributed simulation over the Grid (Pan et al. 2007), namely a Grid-facilitated approach, a Grid-enabled approach and a Grid-oriented approach. In the Grid-facilitated approach, Grid services are defined to facilitate the execution of HLA-based distributed simulations while the actual simulation communications are through a vendor-specific RTI. An example of this approach is the Grid HLA Management System (G-HLAM) proposed by Rycerz (2006) for efficient execution of HLA-based distributed simulations on the Grid.

In the Grid-enabled approach, Grid (or web) service interfaces are provided to enable HLA-based distributed simulations to be conducted in a Grid (or web) environment. A client federate communicates with a federate server using Grid (or web) service communications and the federate server representing the client federate joins an HLA-based distributed simulation using a vendor-specific RTI. An example of this approach is the work done by the XMSF group (Pullen et al. 2005) to integrate simulations with other applications using web services.

In the Grid-oriented approach, the RTI is implemented using Grid services according to the HLA specification. All communications are through Grid service invocations. This approach was raised in Fox's keynote at DSRT 2005 (Fox 2005).

5.1 A Service Oriented HLA RTI

Nanyang Technological University (Singapore) has developed a Service Oriented HLA RTI (SOHR) framework which implements an HLA RTI entirely using Grid services following the Grid-oriented approach. The various Grid services of SOHR cooperate with each other to provide the functionalities of an RTI as services. Federates participate in federations by invoking specific Grid services (through an HLA service interface) without the installation of a heavy-weight vendor-specific RTI software at the local site. Since Grid services are used for communications, firewalls can be overridden and distributed simulations across administrative domains can be conveniently conducted using SOHR. Moreover, the various Grid service components can be dynamically deployed, discovered and undeployed on demand. All these features of SOHR enable scalable execution of HLA-based distributed simulations on the Grid.

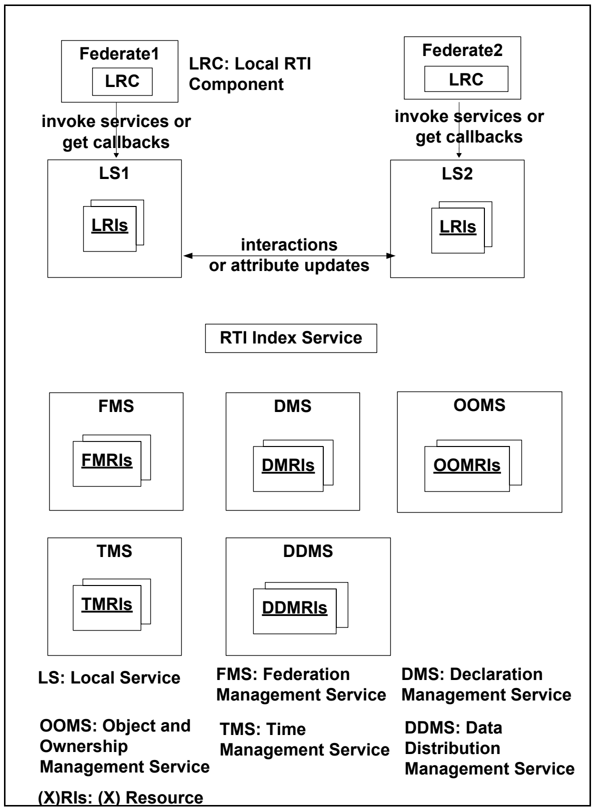
Federate1
LRC
LRC: Local RTI Component
Federate2
LRC
invoke services or get callbacks
invoke services or get callbacks
LS1
LRIs
LS2
LRIs
interactions or attribute updates
RTI Index Service
FMS
FMRIs
DMS
DMRIs
OOMS
OOMRIs
TMS
TMRIs
DDMS
DDMRIs
LS: Local Service
FMS: Federation Management Service
DMS: Declaration Management Service
OOMS: Object and Ownership Management Service
TMS: Time Management Service
DDMS: Data Distribution Management Service
(X)RIs: (X) Resource

Figure 7: Framework overview

The architecture of the SOHR framework is illustrated in Figure 7. It consists of seven key Grid services, namely the RTI Index Service, the LS (Local Service) and five management services, all of which are implemented based on GT4. The RTI Index Service provides a system-level registry so that all other services are able to register their EPRs (End Point References) here and dynamically discover each other. It also provides services to create and destroy federations in the system.

The five management services correspond to the six HLA service groups (the implementation of the HLA Object Management and Ownership Management Service groups are combined in a single Object and Ownership Management Service (OOMS) for the convenience of maintaining object instances and their respective attribute ownership information). Each of them contains multiple Resource Instances, with one Resource Instance for each federation. For example, the FMS provides functionalities of the HLA Federation Management Service group as a Grid service and contains multiple Resource Instances FMRIs, with one FMRI for each federation. The LS (Local Service) is used as a messaging broker of federates and contains multiple Resource Instances LRIs, with one LRI for each federate. A federate communicates with the outside world through its LRI by invoking services and getting callbacks. The LRI is structured into six modules – Callback Module, FM Module, DM Module, OOM Module, TM Module, DDM Module. Each of the modules except the Callback Module corresponds to one of the management services. The Callback Module is used to buffer callbacks for a federate.

The objective of separating the HLA service groups into different Grid services and using a modular structure for the LRI is to create a plug-and-play paradigm so as to build an extensible SOHR framework. There may be multiple algorithms for the implementation of an HLA service group, and these can generally be classified as centralized algorithms or distributed algorithms. In a centralized algorithm, the major processing is done by the corresponding management service while the corresponding module in the LRI simply keeps some necessary information related to its federate. In a distributed algorithm, the major processing is done by the corresponding module in the LRI while the corresponding management service simply keeps relevant centralized information of the federation. A particular module in the LRI and its corresponding management service cooperate to provide the services of the HLA service group. Based on different algorithms for the HLA service group, multiple combinations of a particular module of the LRI and its corresponding management service can be implemented and plugged into SOHR.

It may be thought that the separation of the HLA service groups into the various Grid services may impact the overall performance of SOHR. However, this is not the case as most of the communications, which are generally attribute updates and interactions, are directly conducted between LRIs in a peer-to-peer manner.

The LRC (Local RTI Component) is a federate's local library that implements the HLA service interfaces and does the translation between HLA service interfaces and the corresponding Grid service invocations. A decoupled design is used between a federate and its LRI so that the LRC passes all requests by the federate to the underlying LRI and the LRI buffers callbacks for the federate in its Callback Module. The LRC allows legacy HLA-based federates to work in SOHR without any code modification. Both the HLA 1.3 specification (DMSO 1998) and the IEEE 1516 standard (IEEE 2000) are provided as alternative LRC libraries. Since a standard interface is supported, CSP interoperability solutions that follow the HLA-CSPI standards may readily be supported by SOHR. This allows the possibility of executing federations of CSP models in a Grid environment.

A complete description of the SOHR framework, together with performance results, may be found in Pan et al. (2009). Some of the major benefits of the SOHR framework are summarized as follows:

- The functionalities of an RTI are provided entirely as Grid services so that distributed simulations can be conducted without any vendor-specific RTI software.
- Grid services, which can pierce firewalls, are used as the communication infrastructure so that distributed simulations can be conveniently conducted across administrative domains over the Internet.
- The various Grid services can be deployed and undeployed on demand. The RTI Index Service operates as a system-wide registry so that the other services can dynamically discover each other.
- The decoupled design between a federate and its LRI makes the LRC light weight and simplifies federate migration. As the major local RTI processing is offloaded to the LRI, the federate has more computation resources for executing the simulation.
- Both HLA 1.3 and 1516 interfaces are supported in two different LRC libraries, so that legacy HLA-based federates work well in SOHR.
- Due to the modular design for SOHR, alternative combinations of a particular module of the LRI and its corresponding management service can be implemented based on different algorithms for an HLA service group and plugged in SOHR, which makes SOHR an extensible framework.
- Based on the Grid, SOHR can utilize the underlying Grid infrastructure, such as the Grid Security Infrastructure (GSI), to evolve to a secure, scalable and coordinated large scale distributed simulation environment.

5.2 Extending the NBS Case Study to use SOHR

In the UK the National Health Service Connecting for Health initiative <www.connectingforhealth.nhs.uk> aims to improve health care by enabling the access to patient information. It is a massive IT program and represents an excellent opportunity to investigate advanced decision support systems for health. As Grid technology is stabilizing and is widely used in large scientific experimentation, to begin an investigation in this area we combined our NBS case study with the SOHR framework.

The NBS case study, presented in section 4.1, was extended in order to experiment with CSP interoperability over the Grid. The SOHR framework was used for this. The distribution of the individual NBS models was similar to that illustrated in Figure 4. Thus, we had separate federates simulating the NBS PTI Centre and the various hospitals. The middleware used in this extended case study was SOHR which replaced the DMSO HLA RTI 1.3-NG as shown in Figure 4). Our experiments enabled us to compare the performance (in terms of execution speed) of NBS simulation with two different middleware - DMSO HLA RTI 1.4-NG and SOHR. The performance graph is shown in Figure 8.

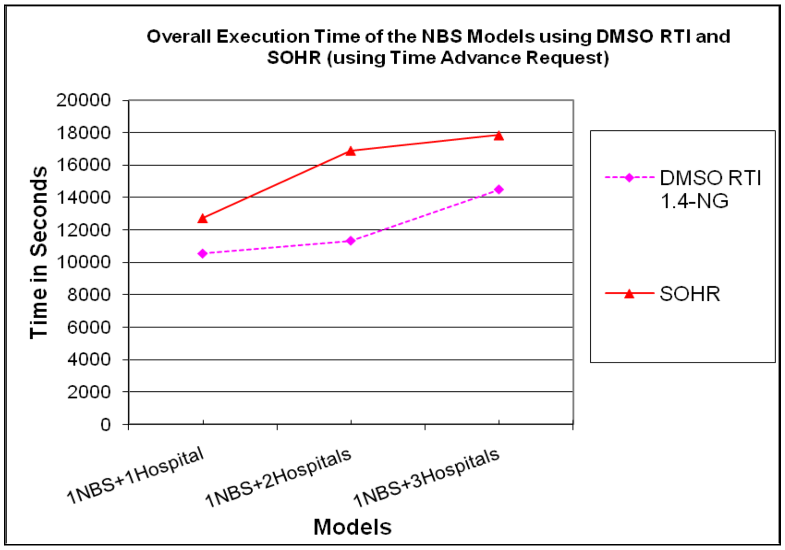

Figure 8: Performance of NBS distributed simulation using two different HLA RTI – DMSO and SOHR

The performance results are based on the execution of three different NBS distributed simulations (*1NBS+1Hospital*, *1NBS+2Hospitals* and *1NBS+3Hospitals*), each model being executed for 5 simulated months. The results show that the performance of the NBS simulations using DMSO RTI 1.3-NG middleware is better when compared to SOHR. However, the difference in performance is relatively small and therefore acceptable as SOHR is based on Web-Services and the latter have overheads associated with them (as has any approach using Web Services). Furthermore, the version of Globus middleware used was not the recommended version (with persistent sockets) and this affected the performance further. However, this case study does show the feasibility of executing SOHR-based distributed simulations created using CSPs over Grid resources. We hope to pursue this further to establish limitations on distributed simulations such as these.

6. CSPI FUTURES

What of the future of CSPI? It seems the future of interoperability and Grid computing are linked. Following other fields of distributed computing, the key to move forward is a parallel development of technology and standards. Technology always moves faster than standards as standards often represent the distilled "best practice" represented by a community. The standards described in this paper are the first in a series of standards required for common development. In some ways the IRMs represent the definition of a problem that needs to be solved effectively. However, alongside these data representation and methodologies must also be standardised.

There have been various debates over recent years as to the ultimate usefulness of these technologies and techniques. To simulation modellers the technology is complex and sometimes difficult to implement. To technology enablers the subtleties of valid implementations can sometimes be missed. To software vendors, the cost of realisation within a CSP can be costly and can require additional development skills that can be costly to acquire. What is certain is that the technology is no longer a barrier and a future where packages and models can be connected together in a "plug & play" manner in such a way to exploit the resources of a Grid is now entirely feasible. To reiterate the examples described in Taylor, Turner and Strassburger (2008), CSPI will make possible the following exciting modeling and simulation scenarios:

- A supply chain distributes equipment to front line troops. A model is built to represent the different supply centers and transportation links to various battlefronts. Experimentation investigates the reliability of the supply chain under different threat conditions.
- An automotive company is planning to build a new factory. The manufacturing line is modeled and simulated using a CSP. Experimentation investigates how many engines can be produced in one year against different levels of resources (machines, buffers, workers, etc.)
- A regional health authority needs to plan the best way of distributing blood to different hospitals. A model is built using a CSP. Experimentation is carried out to investigate different supply policies against "normal" and emergency supply situations.
- A police authority needs to determine how many officers need to be on patrol and how many need to be in the different police stations that it manages. A model is built and experiments are carried out to investigate staffing against different scenarios (football matches, terrorist attacks, etc.)
- A bank sells different financial products. When a new product is planned, managers need to determine the resource impact against current financial services. Using existing business process models (in BPMN for example), a new model is built and simulated. Experiments investigate different resource levels in different departments.

7. CONCLUSIONS

This paper has presented and overview of CSPI and some future directions. Progress in this area is typically made by effective inter-disciplinary collaborations between operations researchers, computer scientists and package vendors. These collaborations tend to be exciting and fruitful. The authors strongly encourage participation in this fascinating area.

REFERENCES

Boer, C. A. 2005. Distributed simulation in industry. *PhD thesis*. Erasmus Research Institute of Management (ERIM), Erasmus University Rotterdam, The Netherlands.

Borshchev, A., Y. Karpov, and V. Kharitonov. 2002. Distributed simulation of hybrid systems with AnyLogic and HLA. *Future Generation Computer Systems*, 18(6):829–839.

DMSO. 1998. Defense Modeling and Simulation Office. High Level Architecture 1.3 – Rules, Interface Specification and Object Model Template.

Foster, I., C. Kesselman, and S. Tuecke. 2001. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3):200-222.

Fox, G., A. Ho, S. Pallickara, M. Pierce, and W. Wu. 2005. Grids for the GiG and Real Time Simulations. In *Proceedings of 9th IEEE International Symposium on Distributed Simulation and Real Time Applications*, 129-138.

Fujimoto, R. M. 1999. Parallel and distributed simulation. In *Proceedings of the 1999 Winter Simulation Conference*, ed. P.A. Farrington, H.B. Nembhard, D.T. Sturrock , G.W. Evans, 122– 131. ACM Press, New York, NY, USA.

Fujimoto, R.M. 2000. *Parallel and Distributed Simulation Systems*. John Wiley and Sons Inc. New York NY, USA.

Fujimoto, R.M. 2003. Distributed simulation systems. In *Proceedings of the 2003Winter Simulation Conference*, eds. S. E. Chick, P. J. Sanchez, D. M. Ferrin, D. J. Morrice, 124-134. Winter Simulation Conference, USA.

Gan, B.P., P. Lendermann, M.Y.H. Low, S.J. Turner, X. Wang, and S.J.E. Taylor. 2005. Interoperating Autosched AP using the High Level Architecture. In *Proceedings of the 2005 Winter Simulation Conference*, eds. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 394-401. Association for Computing Machinery Press, New York, NY.

Globus. 2008. Globus Toolkit Version 4. Available via <http://www.globus.org/> [accessed July 3, 2008].

Hibino, H., Y. Fukuda, Y. Yura, K. Mitsuyuki, and K. Kaneda. 2002. Manufacturing adapter of distributed simulation systems using HLA. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C. H. Chen, J. L. Snowdon, J. M. Charnes, 1099-1107. Association for Computing Machinery Press, New York, NY.

IEEE. 2000. IEEE Standard 1516 (HLA Rules), 1516.1 (Interface Specification) and 1516.2 (Object Model Template).

Katsaliaki, K., and Brailsford, S.C. 2007. Using Simulation to Improve the Blood Supply Chain. *Journal of the Operational Research Society*, 58(2):219-227.

Katsaliaki, K., N. Mustafee, S.J.E. Taylor, and S. Brailsford. (2009). Comparing Conventional and Distributed Approaches to Simulation in Complex Supply-Chain Health Systems. *Journal of the Operation Research Society*, 60(1): 43-51.

Law, A.M. 2007. *Simulation Modeling and Analysis (4th Ed)*. McGraw-Hill.

Lendermann, P., M.Y.H. Low, B.P. Gan, N. Julka, C.L. Peng, S.J. Turner, W. Cai, X. Wang, L.H. Lee, T. Hung, S.J.E. Taylor, and L.F. McGinnis. 2005. An Integrated and Adaptive Decision-Support Framework for High-Tech Manufacturing and Service Networks. 2005. In *Proceedings of the 2005 Winter Simulation Conference*, eds. M. E. Kuhl,

N. M. Steiger, F. B. Armstrong, and J. A. Joines, 2052-2062. Association for Computing Machinery Press, New York, NY.

McLean, C., and F. Riddick. 2000. The IMS MISSION architecture for distributed manufacturing simulation. In *Proceedings of the 2000 Winter Simulation Conference*, eds. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 1539-1548. Association for Computing Machinery Press, New York, NY.

Mertins, K., M. Rabe, and F.W. Jäkel. 2000. Neutral Template Libraries for Efficient Distributed Simulation within a Manufacturing System Engineering Platform. In *Proceedings of the 2000 Winter Simulation Conference*, eds. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 1549-1557. Association for Computing Machinery Press, New York, NY.

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

Mustafee, N. 2007. A Grid Computing Framework for Commercial Simulation Packages. *PhD thesis*. School of Information Systems, Computing and Mathematics, Brunel University, UK.

Mustafee, N., and S.J.E Taylor. 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, UK)*, 33-42.

Mustafee, N., S.J.E. Taylor, K. Katsaliaki, and S. Brailsford. 2009. Facilitating the Analysis of a UK NBS Chain Using the HLA. *SIMULATION: Transactions of the Society of Modelling and Simulation International*, 85(2): 113-128.

Page, E.H., and R.E. Nance. 1994. Parallel discrete event simulation: a modelling methodological perspective. In *Proceedings of the 8th Workshop on Parallel and Distributed Simulation*, 88-93. ACM Press, New York, NY, USA.

Pan, K., S.J. Turner, W. Cai, and Z. Li. 2007. A Service Oriented HLA RTI on the Grid. In *Proceedings of the 2007 International Conference on Web Services*, 984-992.

Pan, K., S.J. Turner, W. Cai, and Z. Li. 2009. Design and Performance Evaluation of a Service Oriented HLA RTI on the Grid. In *Grid Computing: Infrastructure, Service, and Application*, ed. L. Wang, W. Jie, and J. Chen, 459-482. Taylor and Francis.

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

Pullen, J.M., R. Brunton, D. Brutzman, D. Drake, M. Hieb, K. L. Morse, and A. Tolk. 2005. Using Web Services to Integrate Heterogeneous Simulations in a Grid Environment. *Future Generation Computer Systems*, 21(1):97-106.

Raab, M., T. Schulze, and S. Strassburger. 2008. Management of HLA-Based Distributed Legacy SLX-Models. In: *Proceedings of the 2008 Winter Simulation Conference*, eds. S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler, 1086-1093. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Rabe, M., and F.W. Jäkel. 2001. Non military use of HLA within distributed manufacturing scenarios. In *Proceedings of Simulation und Visualisierung*, 141-150.

Revetria, R., P.E.J.N. Blomjous, and S.P.A. Van Houten. 2003. An HLA Federation for Evaluating Multi-Drop Strategies in Logistics. In *Proceedings of the 15th European Simulation Symposium and Exhibition Conference*, 450-455.

Robinson, S. 2005. Distributed simulation and simulation practice. *Simulation*, 81(5): 5-13.

Rycerz, K. 2006. Grid-Based HLA Simulation Support. *PhD thesis*, University van Amsterdam and AGH Krakow.

Straßburger, S. 2001. Distributed Simulation Based on the High Level Architecture in Civilian Application Domains. *PhD thesis*, University of Magdeburg, Germany.

Strassburger, S. 2006. The Road to COTS-Interoperability: From Generic HLA-Interfaces Towards Plug-and-Play Capabilities. In *Proceedings of the 2006 Winter Simulation Conference*, eds. L. R. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 1111-1118. Association for Computing Machinery Press, New York, NY.

Strassburger, S., T. Schulze, and M. Lemessi. 2007. Applying CSPI Reference Models for Factory Planning. In *Proceedings of the 2007 Winter Simulation Conference*, eds. S. G. Henderson, B. Biller, M.-H Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 603-609. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Taylor, S.J.E. 2003. The High Level Architecture – COTS Simulation Package Interoperation Forum (HLA-CSPIF). In *Proceedings of the European Simulation Interoperability Workshop 2003*. Simulation Interoperability Standards Organisation, Institute for Simulation and Training. Florida. 03E-SIW-129.

Taylor, S.J.E., R. Sudra, T. Janahan, G. Tan, and J. Ladbrook. 2002. GRIDS-SCF: An infrastructure for distributed supply chain simulation. *Simulation*, 78(5):312–320.

Taylor, S.J.E., L. Bohli, X. Wang, S.J. Turner, and J. Ladbrook. 2005. Investigating Distributed Simulation at the Ford Motor Company. 2005. In *Proceedings of the 9th International Symposium on Distributed Simulation and Real-Time Applications*, 139-147.

Taylor, S.J.E., S.J. Turner, M.Y.H. Low, X. Wang, S. Strassburger, and J. Ladbrook. 2006a. Developing Interoperability Standards for Distributed Simulation and COTS Simulation Packages with the CSPI PDG. In *Proceedings of the 2006*

Winter Simulation Conference, eds. L. R. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 1101-1110. Association for Computing Machinery Press, New York, NY.

Taylor, S.J.E., X. Wang, S.J. Turner, and M.Y.H Low. 2006b. Integrating Heterogeneous Distributed COTS Discrete-Event Simulation Packages: An Emerging Standards-Based Approach. *IEEE Transactions on Systems, Man & Cybernetics: Part A*, 36(1):109-122.

Taylor, S.J.E., S.J. Turner, and S. Strassburger. 2008. Guidelines for Commercial Off-the-shelf Simulation Package Interoperability. In *Proceedings of the 2008 Winter Simulation Conference*, eds. S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler, 193-204. Association for Computing Machinery Press, New York, NY.

AUTHOR BIOGRAPHIES

SIMON J E TAYLOR is the Founder and Chair of the COTS Simulation Package Interoperability Product Development Group (CSPI-PDG) under the Simulation Interoperability Standards Organization. He is the co-founding Editor-in-Chief of the UK Operational Research Society's (ORS) Journal of Simulation and the Simulation Workshop series. He was Chair of ACM's Special Interest Group on Simulation (SIGSIM) (2005-2008). He is a Reader in the School of Information Systems, Computing and Mathematics at Brunel and has published over 100 articles in modeling and simulation. His recent work has focused on the development of standards for distributed simulation in industry. His email address is <Simon.Taylor@brunel.ac.uk>.

NAVONIL MUSTAFEE is a Research Fellow in Grid Computing and Simulation in the School of Information Systems, Computing and Mathematics, Brunel University (UK). He received his M.Sc. in Distributed Information Systems and Ph.D. in Information Systems and Computing from Brunel University (UK). His research interests are in Grid Computing, Parallel and Distributed Simulation, Healthcare Simulation and Information Systems. He is a member of the drafting group of the COTS Simulation Package Interoperability Product Development Group (CSPI-PDG) under the Simulation Interoperability Standards Organization. His email address is <Navonil.Mustafee@brunel.ac.uk>.

STEPHEN JOHN TURNER is Professor of Computer Science and Head of the Computer Science Division in the School of Computer Engineering at Nanyang Technological University (Singapore). He received his MA in Mathematics and Computer Science from Cambridge University (UK) and his MSc and PhD in Computer Science from Manchester University (UK). His current research interests include: Parallel and Distributed Simulation, Grid Computing, High Performance Computing and Multi-Agent Systems. He is also Secretary of SISO's COTS Simulation Package Interoperability PDG. His email address is <Steve@pmail.ntu.edu.sg>.

KE PAN is a Ph.D. student in the School of Computer Engineering (SCE) at Nanyang Technological University (NTU), Singapore. He received his B.Eng. in the same school with a 1st Class Honors degree. His email address is <Pank0001@ntu.edu.sg>

STEFFEN STRASSBUGER is a professor at the Ilmenau University of Technology in the School of Economic Sciences. In previous positions he was working as head of the "Virtual Development" department at the Fraunhofer Institute in Magdeburg, Germany and as a researcher at the DaimlerChrysler Research Center in Ulm, Germany. He is a member of the editorial board of the Journal of Simulation. His research interests include simulation and distributed simulation as well as general interoperability topics within the digital factory context. He is also the Vice Chair of SISO's COTS Simulation Package Interoperability Product Development Group. Email: <Steffen.Strassburger@tu-ilmenau.de>.