Simulation Interoperability Standards Organization (SISO)

Standard for Commercial-off-the-shelf Simulation Package Interoperability Reference Models

SISO-STD-006-2010

9 March 2010

Prepared by:

SISO COTS Simulation Package Interoperability Product Development Group

SISO Inc. Board of Directors
P.O. Box 781238
Orlando, FL 32878-1238, USA

Acknowledgements

This document was created as a community effort by the COTS Simulation Package Interoperability Product Development Group (CSPI PDG). This PDG was chartered by the Simulation Interoperability Standards Organization (SISO) Standards Activity Committee (SAC) on 10 October 2005. This document would not have been possible without the hard work and dedicated efforts of the following individuals:

Product Development Group Officers
Simon J E Taylor (Chair)
Steffen Strassburger (Vice-Chair)
Stephen J Turner (Secretary)
Katherine Morse/Grant Bailey (SAC TAD)

Drafting Group Team
Additional Document Contributions
Malcolm Low (Lead Editor)
Navonil Mustafee (Assigned Reviewer)
John Ladbrook (Assigned Reviewer)

Balloting Group

Emmet Beeker	The MITRE Corporation
David Bell	Brunel University (Fluidity Group)
Sergio de Cesare	Brunel University (Fluidity Group)
Don Cox	Raytheon Company, Missile Systems
Em Delahostria	Rockwell Automation
Mark Elder	SIMUL8 Corporation
Boon Ping Gan	D-SIMLAB Technologies Pte Ltd
Chris Gaughan	US Army RDECOM Edgewood Chemical Biological Center
Frank Hill	no-affiliation
Shane Kite	Saker Solutions
Jason Lightfoot	Saker Solutions
Malcolm Low	Nanyang Technological University
James McCall	Air Force Research Laboratory / General Dynamics I
Charles McLean	National Institute of Standards and Technology
Navonil Mustafee	University of Swansea
Markus Rabe	Fraunhofer IPK
Frank Riddick	NIST
Jon Saville	Nohorizon
Marco Schumann	Fraunhofer IFF
Steffen Strassburger	Technical University of Ilmenau
David Sturrock	Simio LLC
Simon JETaylor	Brunel University (Distributed Systems Research Group)
Stephen Turner	Nanyang Technological University
Anthony Waller	Lanner Group

Table of Contents

1. Introduction

The Simulation Interoperability Standards Organization (SISO) focuses on facilitating simulation interoperability across government and non-government applications worldwide. SISO's interests include methods that support and promote reuse of simulation components; agile, rapid, and efficient development and maintenance of models; as well as integration of models into operational systems or embedding real world systems into virtual environments.

Discrete-event simulation represents a range of techniques that model a system over time as it changes from event to event. For many years discrete-event simulation has been used to analyze production and logistics problems in commerce, defense, health and many areas of manufacturing. In the early 1980s, visual interactive modeling environments were created that supported the development, experimentation and visualization of simulation models. These Commercial-off-the-shelf (COTS) productivity tools have common facilities including drag and drop model development environments, animation, experimentation support and statistical tools. Examples of these tools include Arena, Anylogic, Flexsim, Simul8 and Witness. Collectively, these tools are referred to as COTS Simulation Packages (CSPs).

Distributed simulation can be defined as the distribution of the execution of a single run of a simulation program across multiple processors. There are various motivations for this. In terms of CSPs, reasons to use distributed simulation include the integration of discrete-event simulations across virtual organizations, extended enterprises and supply chains; reducing the execution time of the simulation by using several computers; reducing the cost of model development by enabling the reuse of distributed model components; and the protection of intellectual property (information hiding in distributed models).

There are many problems when a distributed simulation consisting of CSPs and their models is created. A single CSP is used to create a single model. A distributed simulation therefore involves the interoperation of a model and its CSP and other models and their CSPs. There have been many different approaches used to create such distributed simulations. The differences between approaches and their implementations is extremely difficult to capture. Indeed it is often extremely difficult to assess the capabilities of a particular approach. Further, simulation practitioners that use CSPs tend not to be technically minded and vendors of CSPs often find simulation interoperability techniques inaccessible. It has been the work of SISO's COTS Simulation Package Interoperability Product Development Group (CSPI PDG) to establish common approaches to these problems. As part of this activity a set of "Interoperability Reference Models" has 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. This document specifies the current set of Interoperability Reference Models for CSPs.

In standardization it is important to establish history and credibility. The identification of common CSP interoperability problems and the need for standardization was first recognised during the UK EPSRC[1] project GROUPSIM. This laid the foundation for the predecessor to the CSPI PDG, the COTS Simulation Package Interoperability Forum (CSPIF) that consisted of a representative set of practitioners, CSP vendors and researchers. This ran in parallel with a virtual study group with SISO. Over one year and five workshops the CSPIF laid the foundation for the standardization of CSP interoperation. After considerable published research and presentations at relevant forums such as the Winter Simulation Conference, the UK ORS Simulation Workshop, the ASIM Dedicated Conference on Production and Logistics and the Germany HLA Forum (see section 2 References), as well as further meetings with practitioners, CSP vendors and researchers, the initial outcome of this work has led to this set of Interoperability Reference Models (IRMs). It must be noted that substantial effort has been made to include and consult with CSP vendors, either directly or indirectly via researchers who have established links with CSP vendors, and well-known practitioners. Additionally, the CSPI PDG team are well-known in the general discrete-event simulation community. For example, Taylor, the founder of the CSPI PDG, jointly ran the UK ORS Simulation Study Group, has jointly founded the UK ORS Simulation Workshop Series and is joint founder and co-editor-in-chief of the Journal of Simulation; Strassburger was one of the first researchers to address the problem of CSP interoperability and continues to do so; Turner is a leading researcher in parallel and distributed

[1] Equivalent body to the US NSF.

simulation, is steering committee chair of the ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS) and has served as general or program chair for a number of simulation conferences; and Ladbrook is a well-known simulation practitioner and leader of a large simulation team at The Ford Motor Company. Taylor, Strassburger and Turner have strong links with industry and have studied the problem of CSP interoperability within an industrial context. In summary, the CSPI PDG team has made every effort to ensure that any standards that it develops are both *relevant* and *credible* to the communities that it serves.

1.1 Purpose

This "Standard for COTS Simulation Package Interoperability Reference Models" introduces a set of templates or patterns intended to classify specific interoperability problems encountered in the development of distributed simulations consisting of CSPs and their models. The purpose is to form a common ground where the capabilities of current approaches to CSP interoperability/distributed simulation can be assessed and compared and where practitioners, CSP vendors and solution developers can identify candidate solutions to their interoperability/distributed simulation problems. 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

1.2 Scope

The "Standard for COTS Simulation Package Interoperability Reference Models" is intended to simplify the complexity of assessing capabilities of current approaches to CSP interoperability/distributed simulation and to assist practitioners, CSP vendors and solution developers to identify candidate solutions to their interoperability/distributed simulation problems.

Where is the complexity? Consider the following. As an example, 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. As shown in Figure 1-1, the (extremely simplistic) factories, modeled as models M1 and M2, are simulated in their own CSPs running on their own separate computers. Queues, activities and resources are represented as Q, A and R respectively. The models interact, in this example, as denoted by the thin arrows connecting the models (possibly the delivery and return of some defective stock). Further, the models might share resources (to reflect a shared set of machinists that can operate various workstations), events of various kind (such as the end of a shift) or data (such as the current production volume). The question is, how do we implement this distributed simulation?

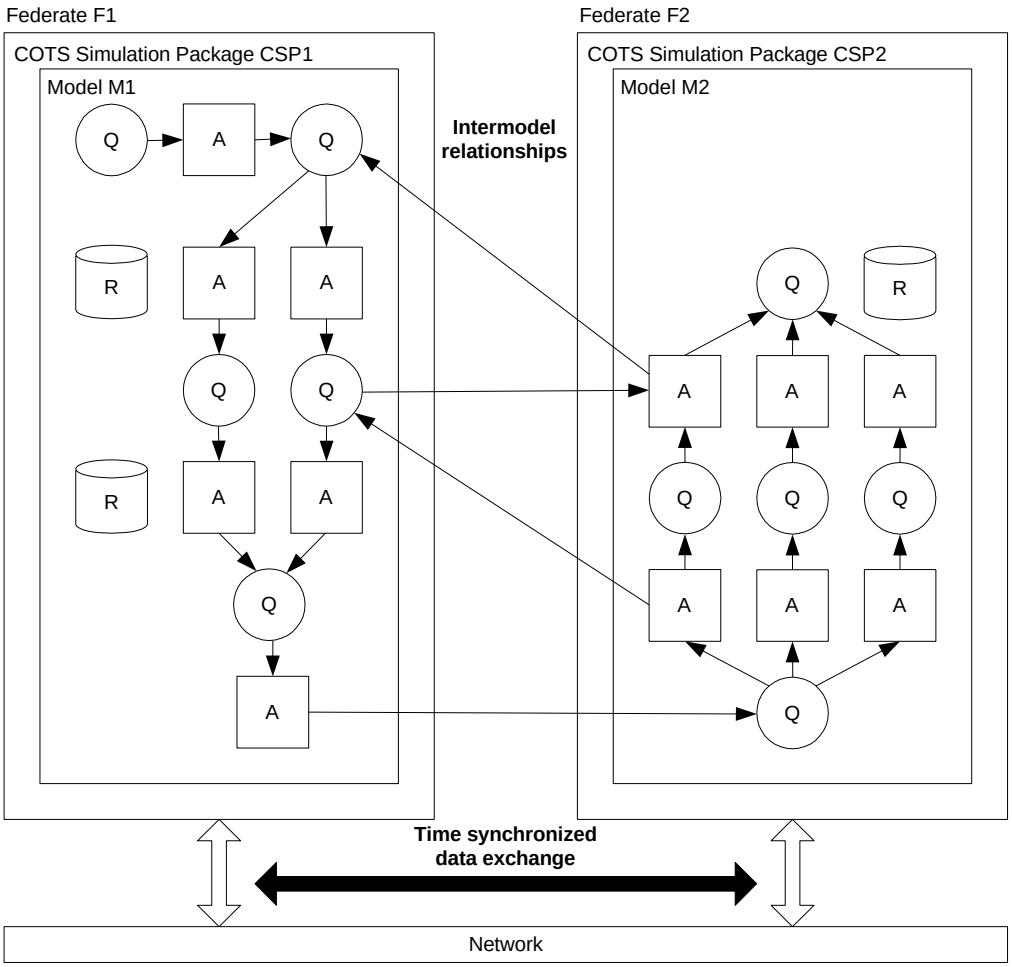


Figure 1-1: The COTS Simulation Package Interoperability Problem

Although a distributed simulation or federation may be composed of a mix of federates using different types of products (e.g, CSP, GOTS and custom-built, etc.), this standard currently only identifies interoperability reference models for the case where all federates have CSP products. A distributed simulation or federation is therefore composed of a set of CSPs and their models. Within this document, a CSP will simulate its model using a discrete-event simulation algorithm. Each model/CSP represents a federate normally running on its own computer. In a distributed simulation, each model/CSP federate therefore exchanges data over a network in a time synchronized manner (as denoted by the thick double-headed arrow). 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 sends and receives information in an agreed format and time synchronized manner and federate F2 must send 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 communicate in the same way. Further, the "passing" of entities and the sharing of resources require different distributed simulation protocols. In entity passing, the departure of an entity from one model and the arrival of an entity at another can be the same scheduled event in the two models – most distributed simulations represent this as a timestamped event message sent from one federate to another. The sharing of resources cannot 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 timestamped communication protocol is required to inform and update the changes of the shared resource state. Further problems arise when we begin to "dig" further into the subtleties of interoperability. It is the purpose of our Interoperability Reference Models to try to simplify this complexity.

1.3 Objectives

The objectives of this standard are therefore:

- to specify a range of interoperability reference models (IRMs) that can be used to clearly identify the capabilities or requirements of an existing or proposed CSP distributed simulation, and
- to establish a mechanism whereby the capabilities or requirements of an existing or proposed CSP distributed simulation can be formally recognised.

1.4 Intended Audience

This document is intended for individuals and organizations in the Modeling and Simulation (M&S) (or simulation modeling) community who are interested in using and/or developing distributed simulations consisting of COTS Simulation Packages that use discrete-event simulation. Potential consumers of this specification include those who analyze production and logistics problems in commerce, defense, health and many areas of manufacturing and any potential user of discrete-event simulation COTS Simulation Packages, COTS Simulation Package vendors and solution providers/developers *across the world.* Note that although a distributed simulation or federation may be composed of a mix of federates using different types of products (e.g, CSP, GOTS and custom-built, etc.), this standard currently only identifies interoperability reference models for the case where all federates have CSP products.

2 References

This document is intended to be a reference for other documents. The Interoperability Reference Models are also intended to be presented without direct reference to the IEEE 1516 High Level Architecture and, as such, broadens the scope of this document. The following SISO reference documents were used in preparing this SISO product. A complete list of approved SISO products that may be referenced are available by linking to "Products" through the SISO web site at http://www.sisostds.org/.

2.1 SISO References

	Document Number	Title
1	SISO-ADM-001-2001	SISO Numbering Procedure
2	SISO-ADM-002-2008	SISO Policies and Procedures (P&P)
3	SISO-ADM-003-2008	SISO Balloted Products Development and Support Process (BPDSP)
4	SISO-ADM-005-2008	The Style and Format of SISO Documents

2.2 Bibliography

	DOCUMENT NUMBER	TITLE
1	IEEE 100	*The Authoritative Dictionary of IEEE Standard Terms,* Seventh Edition, New York, Institute of Electrical and Electronics Engineers, Inc.

3 Definitions

The definitions identified Table 3-1 are common terms used in this document. Given that the meaning of some of these terms differs among domains of interest, these definitions are provided to identify the meaning of the terms in the scope of this document. The *Authoritative Dictionary of IEEE Standards Terms, Seventh Edition* [IEEE 100] should be referred to for terms not defined in this section.

Table 3-1 Common Terms

Model (M)	A model represents a real world system and is executed by a CSP.
Federate (F)	A federate consists of a model, its CSP and associated software interfaces running on a single computer.
Event (E)	An instantaneous state change at a time T. For example, an event E at a time T marks the transfer of an entity from a queue to an activity or an activity to a queue or a change to a data structure.
Time (T)	A value representing a specific instant of simulation time in a model. As time units are relative, these have no specific units in this standard.
Entity (e)	An entity represents something that is processed, i.e. an entity passes through a series of queues and activities. For example, an entity could represent a patient, an engine part, a task, etc. Entities belong to a class and are differentiated by attributes.
Queue (Q)	A queue of entities. We assume that a queue will have some queuing discipline (FIFO, LIFO, etc.)
Activity (A)	An activity represents some time consuming action in a system with a known duration. The start of an activity is marked by an event and the end of an activity is marked by another event. For example, a workstation in a factory will process an entity for a given time. Typically, an activity will begin if there is an entity in its preceding queue. Optionally, an activity will begin if there is an entity in its preceding queue and a resource available (for example a machine that needs an operator).
Resource (R)	A resource represents something that is needed by an activity for it to begin. For example, the operator of a machine.
Data Structure (D)	This is similar to a resource but semantically different in terms of modeling and is therefore separately included. For example, a data structure could be an inventory record. In this standard a single variable is considered to be data structure as it has the same interoperability requirements.

4 Acronyms and Abbreviations

The following acronyms and abbreviations are used in this document.

API	Application Program Interface
CSP	COTS Simulation Package
CSPI	COTS Simulation Package Interoperability
EXCOM	Executive Committee
HLA	High Level Architecture
IEEE	Institute of Electrical and Electronics Engineers, Inc.
IRM	Interoperability Reference Model
M&S	Modeling and Simulation (alternative Simulation Modeling)
PDG	Product Development Group
POC	Point of Contact
RTI	Run-time Infrastructure
SAC	Standards Activity Committee
SISO	Simulation Interoperability Standards Organization
TAD	Technical Area Director
UML	Unified Modeling Language

5 Interoperability Reference Model Rationale

To reiterate the outline problem stated in Section 1.2 Purpose, the complexity of developing, or indeed understanding, a distributed simulation consisting of interoperating CSPs and their models requires some form of simplification to verify if a distributed simulation correctly implements what it claims to interoperate. Interoperability Reference Models (IRMs) is one of the enabling standardized constructs provided by the CSPI PDG.

The identified interoperability problems have been divided into a set of problem *types* that are represented by IRMs. An interoperability problem *type* is meant to capture a general class of interoperability problem, while an *IRM* is meant to capture a specific problem within that class.

5.1 Interoperability Reference Model Definition

An Interoperability Reference Model (IRM) is defined as the simplest representation of problem within an identified interoperability problem type. Each IRM can be subdivided into different subcategories of problem. As IRMs are usually relevant to the boundary between two or more interoperating models, models specified in IRMs will be as simple as possible to "capture" the interoperability problem and to avoid possible confusion. These simulation models are intended to be representative of real model/CSPs but use a set of "common" model elements that can be mapped onto specific CSP elements (see 5.2 Clarification of Terms). Where appropriate, IRMs will specify time synchronization requirements and will present alternatives. IRMs are intended to be composable (i.e. some problems may well consist of several IRMs). Most importantly, IRMs are intended to be understandable by *CSP vendors, simulation users and technology solution providers.*

5.2 Clarification of Terms

As indicated above, an IRM will typically focus on the boundary between interoperating models. To describe an interoperability problem we therefore need to use model elements that are as general as possible. Generally, CSPs using discrete-event simulation model systems that change state at *events*. Rather than providing a set of APIs to directly program discrete-event simulations, these CSPs use a visual interface that allows modelers to build models using a set of objects. These models are typically composed of networks of alternating *queues* and *activities* that represent, for example, a series of buffers and operations composing a manufacturing system. *Entities*, consisting of sets of typed variables termed *attributes*, represent the elements of the manufacturing system undergoing machining. Entities are transformed as they pass through these networks and may enter and exit the model at specific points. Additionally, activities may compete for *resources* that represent, for example, the operators of the machines. To simulate a model a CSP will typically have a simulation executive, an event list, a clock, a simulation state and a number of event routines. The simulation state and event routines are derived from the simulation model. The simulation executive is the main program that (generally) simulates the model by first advancing the simulation clock to the time of the next event and then performing all possible actions at that simulation time. For example, this may change the simulation state (for example ending a machining activity and placing an entity in a queue) and/or schedule new events (for example a new entity arriving in the simulation). This cycle carries on until some terminating condition is met (such as running until a given time or a number of units are made).

A problem is, however, that virtually every CSP has a different variant of the above. CSPs also have widely differing terminology, representation and behavior. For example, without reference to a specific CSP, in one CSP an entity as described above may be termed an *item* and in another *object*. In the first CSP the data types might be limited to integer and string, while in the other the data types might be the same as those in any object-oriented programming language. The same observations are true for the other model elements such as queue, activity and resource. Behavior is also important as the set of rules that govern the behavior of a network of queues and activities subtly differ between CSPs (for example the rules that govern behavior when an entity leaves a machine to go to a buffer). Indeed even the representation of *time* can differ. This is also further complicated by variations in model elements over and above the "basic" set (e.g. entry/exit points, transporters, conveyors, flexible manufacturing cells, robots, etc.) See section 3 for specific definitions of terms.

5.3 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 shared inventory.

Note that the above classification previously appeared as:

Type I: Asynchronous Entity Passing
Type II: Synchronous Entity Passing (Bounded Buffer)
Type III: Shared Resources
Type IV: Shared Events
Type V: Shared Data Structures
Type VI: Shared Conveyor

This has been rationalised to the Type A-D classification to “group” IRM problems (essentially new Entity Transfer problems were identified). Note that the “Shared Conveyor” IRM has been deleted as it was felt by the PDG that this would usually be represented as a separate model and therefore fall into the other IRM Types.

5.4 Interoperability Reference Model Use

Note that the IRM types and sub-types are intended to be composable, i.e. a distributed simulation that correctly transfers entities from one model to a bounded buffer in another model as well as sharing a production schedule (shared data) should be can be compliant with both IRM Type A.1 General Entity Transfer, IRM Type A.2 Bounded Receiving Element and IRM Type D.1 General Shared Data (see the specific *compliancy* sections in each IRM).

6 Interoperability Reference Model Type A: Entity Transfer

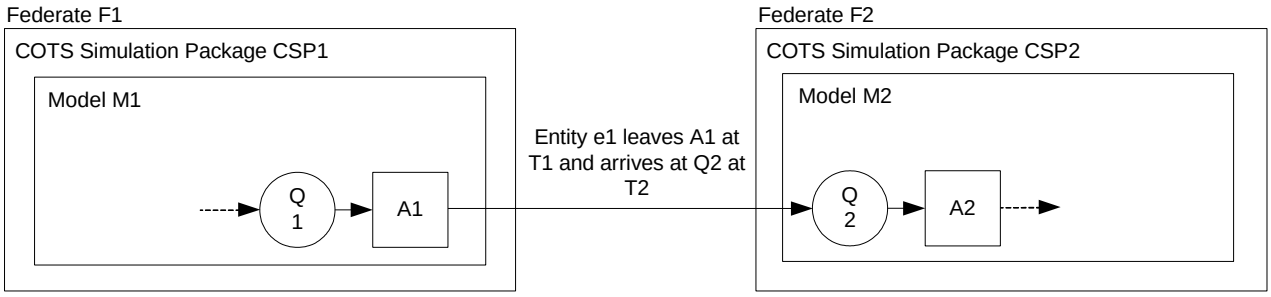


Figure 6-1: IRM Type A.1: General Entity Transfer

6.1 Overview

IRM Type A Entity Transfer represents interoperability problems that can occur when transferring an entity from one model to another. Figure 6-1 shows an 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 sub-types are intended to be composable, i.e. a distributed simulation that correctly transfers entities from one model to a bounded buffer in another model will be compliant with both IRM Type A.1 General Entity Transfer and IRM Type A.2 Bounded Receiving Element (see the *compliancy* sections for both IRMs for more details).

6.2 Interoperability Reference Model Type A Sub-types

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

6.3 IRM Type A.1 General Entity Transfer

6.3.1 Overview

IRM Type A.1 General Entity Transfer represents the case, as described above and shown in Figure 6-1, 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).

This IRM does not include cases where

- the receiving element is bounded (IRM Type A.2), and
- multiple inputs need to be prioritized (IRM Type A.3).

6.3.2 Definition

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 in some instances a CSP distributed simulation need only satisfy the condition T1<T2.

6.3.3 Compliancy

A distributed simulation shall be deemed compliant with the IRM Type A.1 General Entity Transfer only if it

- meets functionality as defined in 6.3.2, and
- specifies which condition T1=<T2 or T1<T2 is met.

6.4 IRM Type A.2 Bounded Receiving Element

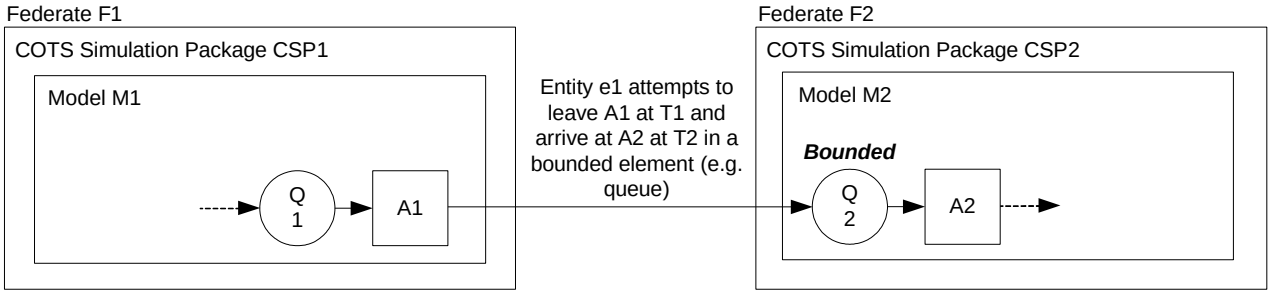


Figure 6-2: IRM Type A.2: Bounded Receiving Element

6.4.1 Overview

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 6-2 shows an illustrative example, where an entity e1 attempts to leave model M1 at T1 from activity A1 and to arrive at model M2 at T2 in ***bounded*** queue Q2. If A1 represents a machine then the following scenario is possible. When A1 finishes work on a part (an entity), it attempts to pass the part to queue Q2. If Q2 has spare capacity, then the part can be transferred. However, if Q2 is full then A1 cannot release its part and must block. Parts in Q1 must now wait for A1 to become free before they can be machined. Further, when Q2 once again has space, A1 must be notified that it can release its part and transfer it to Q2. Finally, it is important to note 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).

A solution to this IRM problem must also

- be able to transfer entities (IRM Type A.1).

6.4.2 Definition

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.

6.4.3 Compliancy

A distributed simulation shall be deemed compliant with the IRM Type A.2 only if it

- meets the functionality as defined in 6.4.2,
- specifies the blocking policy of the equivalent of element O, and
- specifies if the case

  > "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"

  is possible and how this is dealt with in the distributed simulation

6.5 IRM Type A.3 Multiple Input Prioritization

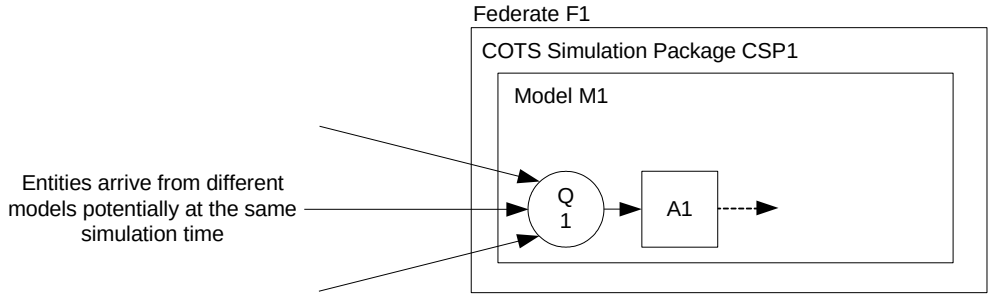


Figure 6-3: IRM Type A.3 Multiple Input Prioritization

6.5.1 Overview

As shown in Figure 6-3, 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 specialised.

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).

6.5.2 Definition

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:

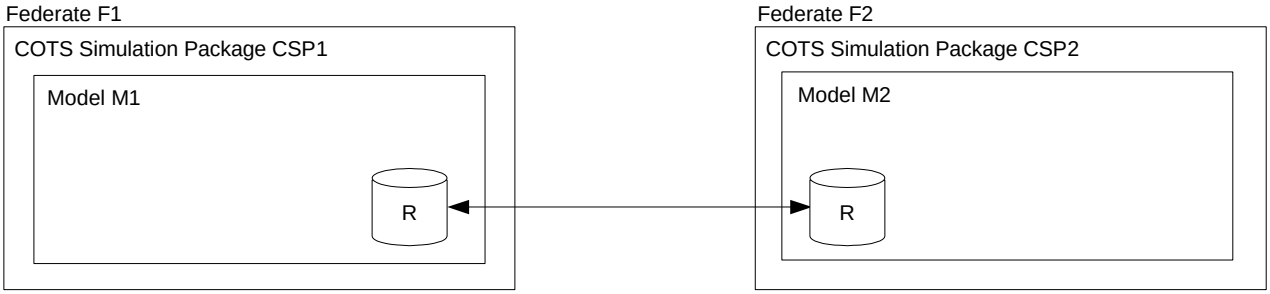
- The priority rules must be specified.
- Priority rules may change during a simulation if required for the real system being simulated.

6.5.3 Compliancy

A distributed simulation shall be deemed to be compliant with the IRM Type A.3 Multiple Input Prioritization only if it

- meets functionality as defined in 6.5.2,
- specifies the priority rules that must be implemented, and
- specifies if such priority rules change during a simulation, then it must define how these changes are implemented.

7 Interoperability Reference Model Type B: 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

Figure 7-1: IRM Type B.1: General Shared Resource

7.1 Overview

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.

7.2 Interoperability Reference Model Type B Sub-types

There is currently one IRM Type B Sub-types

- IRM Type B.1 General Shared Resource

7.3 IRM Type B.1 General Shared Resource

7.3.1 Overview

IRM Type B.1 General Shared Resource represents the case, as outlined above and shown in Figure 7-1, 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.

7.3.2 Definition

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.

7.3.3 Compliancy

A distributed simulation shall be deemed compliant with the IRM Type B.1 General Shared Resource only if it meets functionality as defined in 7.3.2.

8 Interoperability Reference Model Type C: Shared Event

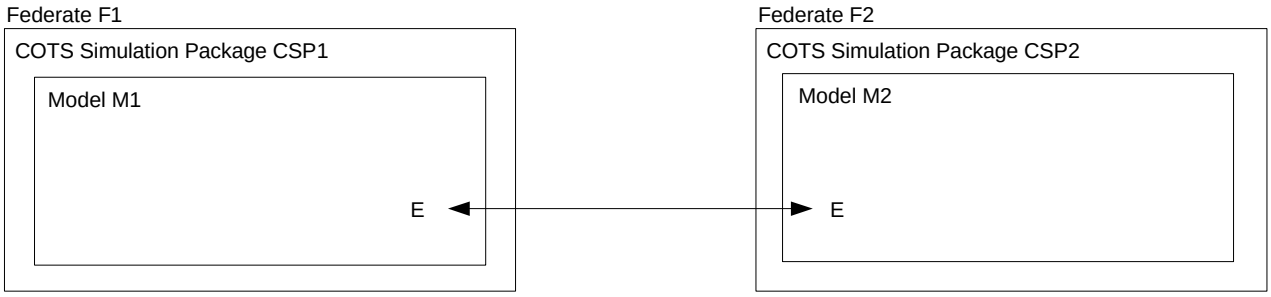


Figure 8-1: IRM Type C.1: General Shared Event

8.1 Overview

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.

8.2 Interoperability Reference Model Type C Sub-types

There is currently one IRM Type C sub-type

- IRM Type C.1 General Shared Event

8.3 IRM Type C.1 General Shared Event

8.3.1 Overview

IRM Type C.1 General Shared Event represents the case, as shown in Figure 7-1, 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.

8.3.2 Definition

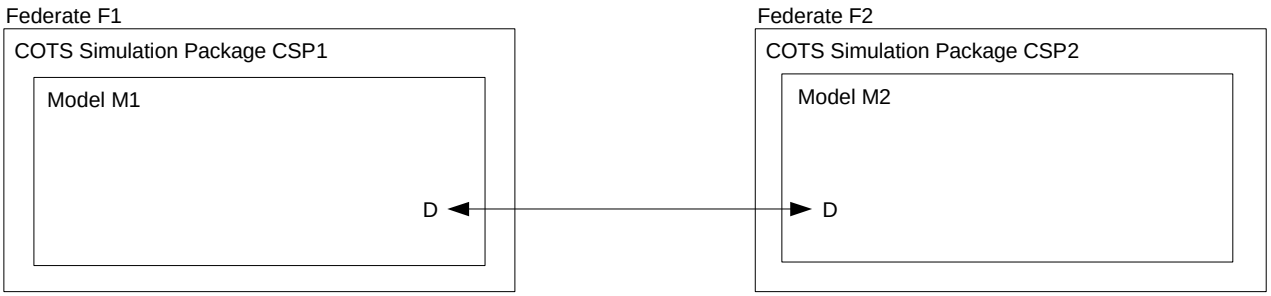
The IRM Type C.1 General Shared Event is defined as the guaranteed execution of all local copies of a shared event E such that

- 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.

8.3.3 Compliancy

A distributed simulation shall be deemed compliant with the IRM Type C.1 General Shared Event only if it meets functionality as defined in 8.3.2.

9 Interoperability Reference Model Type D: Shared Data Structure



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

Figure 9-1: IRM Type D.1: Shared Data

9.1 Overview

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.

9.2 Interoperability Reference Model Type D Sub-types

There is currently one IRM Type D Sub-type.

- IRM Type D.1 General Shared Data Structure

9.3 IRM Type D.1 General Shared Data Structure

9.3.1 Overview

IRM Type D.1 General Data Structure represents the case, as outlined above and shown in Figure 9-1, 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.

9.3.2 Definition

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.

9.3.3 Compliancy

A distributed simulation shall be deemed compliant with the IRM Type D.1 General Shared Data Structure if it meets functionality as defined in 9.3.2.

Annex A Related Work

The following may provide further background information on the development of this standard and examples of its use.

[B1] Brailsford, S., Katsaliaki, K., Mustafee, N. and Taylor, S.J.E. (2006) Modeling Very Large Complex Systems Using Distributed Simulation: A Pilot Study in a Healthcare Setting. In *Proceedings of the 2006 UK ORS Simulation Study Group Workshop*. UK Operational Research Society, Birmingham, UK. 257-262.

[B2] 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 2005 Winter Simulation Conference*. Association for Computing Machinery Press, New York, NY. 394-401.

[B3] Gan, B.P., Low, M.Y.H., Turner, S.J. and Wang, X. (2005). Using Manufacturing Process Flow for Time Synchronization in HLA-Based Simulation. In *Proceedings of the 9th IEEE International Symposium on Distributed Simulation and Real Time Applications*, IEEE Computer Society. 148-157.

[B4] Lendermann, P., Turner, S.J., Low, M.Y.H., Gan, B.P., Julka, N., Chan, L.P., Cai, W., 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.

[B5] Lendermann, P., Low, M.Y.H., Gan, B.P., Julka, N., Peng, C.L., Turner, S.J., Cai, W., Wang, X. Lee, L.H., Hung, T., Taylor, S.J.E. and McGinnis, L.F. (2005) An Integrated and Adaptive Decision-Support Framework for High-Tech Manufacturing and Service Networks. In *Proceedings of the 2005 Winter Simulation Conference*. Association for Computing Machinery Press, New York, NY. 2052-2062.

[B6] 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 UK ORS Simulation Study Group Workshop*. UK Operational Research Society, Birmingham, UK. 33-41.

[B7] 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 2006 Winter Simulation Conference*. Association for Computing Machinery Press, New York, NY. 1136-1142.

[B8] Ryde, M. and Taylor S.J.E. (2004). Designing Interoperating COTS Simulation Packages from the User's Perspective. In *Proceedings of the 2004 UK ORS Simulation Study Group Workshop*. UK Operational Research Society, Birmingham, UK. 31-38.

[B9] Ryde, M.D. and Taylor, S.J.E. (2003). Issues in Using COTS Simulation Packages for the Interoperation of Models. In *Proceedings of the 2003 Winter Simulation Conference.* Association for Computing Machinery Press, New York, NY. 772-777.

[B10] 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.* Association for Computing Machinery Press, New York, NY. 1111-1118.

[B11] Taylor, S.J.E. (2003). HLA-CSPIF: The High Level Architecture – COTS Simulation Package Interoperation Forum. In *Proceedings of the Fall 2003 Simulation Interoperability Workshop*. Simulation Interoperability Standards Organisation, Institute for Simulation and Training. Florida. 03F-SIW-126.

[B12] 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 Organization, Institute for Simulation and Training. Florida. 03E-SIW-129.

[B13] Taylor, S.J.E. (2004). Distributed Simulation: Status and Perspectives. In *Experiences from the Future: New Methods and Applications in Simulation for Production and Logistics*. Mertins, K. and Rabe, M. (eds.) Fraunhofer IRB Verlag, Stuttgart, Germany, 23-42.

[B14] Taylor, S.J.E., Bohli, L., Wang, X., Turner, S.J. and Ladbrook, J. (2005). Investigating Distributed Simulation at the Ford Motor Company. In *Proceedings of the Ninth IEEE International Symposium on Distributed Simulation and Real-Time Applications*. IEEE Computer Society. 139-147.

[B15] Taylor, S.J.E., Gan, B.P., Strassburger, S., Verbraeck, A. (2003). HLA-CSPIF Technical Panel on Distributed Simulation. In *Proceedings of the 2003 Winter Simulation Conference.* Association for Computing Machinery Press, New York, NY. 881-887.

[B16] Taylor, S.J.E., Strassburger, S. Turner, S.J., Low, M.Y.H., Wang, X. and Ladbrook, J. (2006). Developing Interoperability Standards for Distributed Simulation and COTS Simulation Packages with the CSPI PDG. In *Proceedings of the 2006 Winter Simulation Conference*. Association for Computing Machinery Press, New York, NY. 1101-1110.

[B17] Taylor, S.J.E., Turner, S.J. and Low, M. Y.-H. (2004) A Proposal for an Entity Transfer Specification for COTS Simulation Package Interoperation. In *Proceedings of the 2004 European Simulation Interoperability Workshop*. Simulation Interoperability Standards Organization, Institute for Simulation and Training. Florida. 04E-SIW-081.

[B18] Taylor, S.J.E., Turner, S.J. and Low, M.Y.H. (2005) The COTS Simulation Interoperability Product Development Group. In *Proceedings of the 2005 European Interoperability Workshop*. Simulation Interoperability Standards Organization, Institute for Simulation and Training, Florida, 05E-SIW-056.

[B19] 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.

[B20] 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 Twentieth ACM/IEEE.SCS Workshop on Principles of Advanced and Distributed Simulation*. IEEE Computer Society. 201-210.

[B21] Wang, X., Turner, S.J., Taylor, S.J.E., Low, M.Y.H. and Gan, B.P. (2005). A COTS Simulation Package Emulator for Investigating COTS Simulation Package Interoperability. In *Proceedings of the 2005 Winter Simulation Conference*. Association for Computing Machinery Press, New York, NY. 402-411.

[B22] Wang, X., Turner, S.J., Low, M.Y.H. and Gan, B.P. (2004). A Generic Architecture for the Integration of COTS Packages with the HLA. In *Proceedings 2004 UK ORS Simulation Workshop*. Operational Research Society, Birmingham, UK. 225-233.

For a background on discrete-event simulation and CSPs, please see the following references.

[B23] Law, A.M. (2006). *Simulation Modeling and Analysis (4th ed.)*. McGraw-Hill, New York, NY.

[B24] Pidd, M. (2004). *Computer Simulation in Management Science (5th ed.)*. Wiley, Chichester, UK.

[B25] Robinson, S. *Simulation: The Practice of Model Development and Use*, Wiley, Chichester, UK.

[B26] Schriber , T.J. and Brunner, D.T. (2006). Inside Discrete-Event Simulation Software: How It Works and Why It Matters. In *Proceedings of the 2006 Winter Simulation Conference*. Association for Computing Machinery Press, New York, NY. 118-128.

[B27] Swain, J.J. (2007). INFORMS Simulation Software Survey. A publication of OR/MS Today. Web site http://www.lionhrtpub.com/orms/surveys/Simulation/Simulation.html.