Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

ISSUES USING COTS SIMULATION SOFTWARE PACKAGES FOR THE INTEROPERATION OF MODELS

Michael D. Ryde
Simon J. E. Taylor

Centre for Applied Simulation Modelling
Department of Information Systems and Computing
Brunel University
Uxbridge, UB8 3PH, UNITED KINGDOM

ABSTRACT

This paper intends to examine the interoperation of simulation models from the viewpoint of a simulation engineer who uses standard tools and methods to create these models. The paper will look at the models in the context of COTS (Commercially available Off-The Shelf) simulation packages with a view to applying Distributed Simulation (DS) theory to the subject. By studying current methods employed which enable COTS simulation packages to interoperate, this paper will discuss the tools currently used and examine their appropriateness. The paper will also suggest how an example COTS simulation package could be modified to provide the necessary functions and interoperability required to allow full distributed simulation.

1 INTRODUCTION

Distributed Simulation has provided many opportunities for simulation models to be run together over a computer network such as the Internet. The High Level Architecture (HLA) is one major contribution made by this field. The standard (IEEE 1516) provides a framework for distributed simulation. Each model, or federate, interacts with each other (interoperates) to accomplish the simulation exercise. The combined set of interoperating federates is referred to as a federation. The HLA gives standards for data representation (needed so that the communicating federates can "talk" the same language – the format of data exchanged between models) and middleware (to allow communicating parties to "talk" – this is the federate interface specification, the implementation of which is called a run time infrastructure, RTI).

Distributed simulation enabled by the HLA has been used extensively in military systems (see previous Winter Simulation Conferences and SISO's Simulation Interoperability Workshops for many examples). There have been relatively few examples of this in industry. This is not for the lack of opportunity. See Strassburger (2001) for an in-depth discussion on how the HLA could be used outside of the defense arena. Another use of the HLA outside the defense arena was put forward as part of the Intelligent Manufacturing Systems (IMS) mission project, see McLean and Riddick (2000). Interestingly, an observation made during this research was that the current RTIs (developed by different sources) did not interoperate with each other thus all models in a distributed simulation would need to use the same RTI.

Another RTI based development includes GRIDS, which provides a generic run-time infrastructure for the execution of distributed simulations. GRIDS provides basic simulation services to connect simulation models (federates) cooperating to perform a distributed simulation (federation), and extensible simulation services to provide performance enhancement, time-management, mobile entities, as required. Sudra et al. (2000).

The benefits of distributed simulation could include:

- Model reuse. If components or parts of models can be reused within a larger model then this could save development time (a major motivation for the development of the HLA).
- Inter-enterprise simulation. The modelling of global enterprises across geographical boundaries, which could normally be prohibitive because of distance and/or 'working hours' issues.
- Commercial sensitivity, non-disclosure, protection of intellectual copyrights (IPRs) and privacy. In a supply chain, where confidentiality may have prevented organizations sharing information (since it is likely that model developers would need access to each others models and hence potentially sensitive information) the creation of models that work together over a network that are private but share information, could enable the modelling of supply chains where information must be secure.

- Concurrent Development. Models can be built in relative isolation thereby enabling concurrent development in the same way that many large software packages are developed. Indeed, in some instances, a model can be as complex and as time consuming to write as a software package.
- Large model development. Whatever the arguments are to the contrary, some simulationists build models of parts of production lines. "Cutting and pasting" these models together to run in a single environment is sometimes not possible. This is another opportunity for distributed simulation.

In the simulation modelling community there are extreme views as to the ultimate future of distributed simulation in industry, not least of which were expressed in Taylor et al. (2002). To contribute to this debate, this paper reviews some of the issues of the distributed simulation, or interoperation, of simulation models in industry. To do this, the paper is structured as follows. In section 2 we review contemporary simulation modelling practice in industry from the perspective of commercial off-the-shelf (COTS) simulation packages. Section 3 then discusses the general requirements that interoperation makes of COTS simulation packages. Section 4 suggested some enhancements to a typical COTS simulation package by way of discussing functional and user interface requirements. Section 5 provides some evaluation of the issues raised and Section 6 concludes the paper with a short discussion of some possible 'ways forward'.

2 COTS SIMULATION PACKAGES

A typical COTS simulation package, for the purposes of this paper, is considered to be an application in which simulation models can be constructed, saved and reused. The model would normally be constructed from objects, some of which would be standardized between models. Further, it is also expected that the package would have some form of representation for entities (items of work) which would be used within the model. Typically these packages would include definitions for entity distributions and methods by which various objects within the model could be linked or ordered. COTS simulation packages can be, and often are, used by various sized organizations but are easily accessible to even the smallest of businesses because of their low cost. Thus the diversity of model that the packages are expected to deal with is fairly broad.

Often organizations have their own 'favorite' or site-wide license for a particular COTS simulation package. Many packages have adopted different implementations and model object definitions making it difficult, even if the development resource was available, to enable model interoperability. There is a selection of COTS simulation packages available. A brief review carried out during March 2003 revealed the following (although not exhaustive) list:

1. ARENA (Rockwell Software)
2. AUTOMOD (Brooks Automation AutoSimulations Division)
3. Awe Sim (Frontstep, Inc.)
4. EXTEND (Imagine That, Inc.)
5. GPSS for Windows (Minuteman Software)
6. GPSS/H/Proof Animation/SLX (Wolverine Software Corporation)
7. iGraphx Process 2000 (Micrografx, Inc.)
8. microGPSS/webGPSS (Ingolf Stahl)
9. ProModel (Production Modelling Corporation)
10. QUEST (DELMIA Corporation)
11. SIGMA (Custom Simulation)
12. SIMPROCESS/SIMSCRIPT II.5 (CACI Products Company)
13. SIMUL8 (SIMUL8 Corporation)
14. Taylor Enterprise Dynamics (F & H Simulations)
15. Visual Simulation Environment (Orca Computer, Inc.)
16. WITNESS (Lanner Group, Inc.)

2.1 Pseudo Distributed Simulation: The 'Spreadsheet Approach'

Currently there are no known products that have the ability to support and natively allow multiple models to interoperate without at-least the use of some basic middleware component. However, there are methods used to emulate the interoperation of models.

2.1.1 Passing Data, Storing Results – Sequentially

Usually simulation models require, as a minimum, input in the form of a distribution of entities. The entity distribution for a model could be taken from existing models by executing a number of experimental runs to determine the required spread and frequency. This information could then be passed directly into a model via a spreadsheet. Many COTS simulation packages provide functionality to write out to and read variables from a spreadsheet package in order to provide a way of passing information between models. In many cases this provides little more than the passing of information sequentially from one model to another.

2.1.2 Passing Data, Storing Results – In a DS Way

To apply the same method to many models passing information (entities) to one another one must consider the synchronization if causality issues are to be avoided. It is likely that if multiple models were running and passing information to each other then these models could be running at different speeds; i.e. the simulation clocks would be different. Thus Model A when receiving an event from Model B and Model C, would need to determine which event to process first. Using a spreadsheet package to facilitate the passing of entities may provide some limited mechanism for reading/writing timestamped information, event list information and even synchronization logic (time-management). However it is suggested that a spreadsheet, using basic functions would be grossly inadequate and such a mechanism would require some further middleware logic (program instructions) to give the required functionality. It can then be argued that the spreadsheet package is no longer acting as a simple data passing mechanism, more as a fully-fledged time-management component. Is a spreadsheet package really the best tool for the job in this case?

It has long been suggested that the distribution and interoperation of simulation models can be achieved through the use of a 'Spreadsheet' some evidence of this can be found in Clarke (1993). This we term as the *'Spreadsheet Approach', which*, it is postulated, is inappropriate for all but the simplest interoperations.

As suggested earlier it can be seen that using this method for distributed simulation cannot work without some layer of intermediate code to deal with the time-management functionality. It can therefore be assumed that programming skills would also be required by the simulation engineer in order to create this middleware.

2.2 Distributed Simulation

There are some emerging developments which may support the interoperation of COTS packages in the future such as FAMAS-HLA Bridge, Boer et al. (2002). However, the design of the system requires that the packages used to build the models are HLA compliant which, unfortunately not all are. The middleware used within this system uses a standard version of the HLA RTI (Run Time Infrastructure).

By examining various COTS simulation packages it is evident that their implementations may also make it difficult to pass entities (due to the definition of an entity) and information due to non-standardization of model interfaces.

3 REQUIREMENTS FOR DISTRIBUTED SIMULATION

COTS simulation packages can be made to interoperate through the use of bespoke middleware which may include functionality to control the overall model, pass and convert entity information, set and control global parameters, collect results and statistics and provide mechanisms for different entry and exit points into each model. Where interoperating models are tightly coupled the complexity of such middleware increases. Further issues regarding time-management must also be addressed for successful implementations.

If a COTS Simulation package provides full external control and the ability to externalize data (using a spreadsheet package), then with the use of middleware it is believed that the package could be made to support a distributed simulation model. However, the success will largely depend on how well these functions have been implemented and to what level of granularity the facilities have been provided. In a package like SIMUL8 the use of middleware would be relatively straightforward.

The middleware in such a distributed model would primarily be responsible for message passing and synchronization, and in the case of heterogeneous COTS simulation package interoperability, translation of message/entity information to a common standard format.

Without middleware it is unlikely that any of the current COTS simulation package designs could support distributed simulation.

4 SUGGESTED ENHANCEMENTS TO A COTS SIMULATION PACKAGE

It is believed that due to the way many COTS Simulation packages are designed adding interoperability could be relatively straightforward. For the purposes of this paper we restrict ourselves to one package, SIMUL8 (SIMUL8 Corporation). This package has a Visual Interactive Modeling (VIM) interface, uses event lists and defined entities. SIMUL8 is an accessible package for many organizations due to its costing structure and is available on the Microsoft Windows platform. The VIM provides a high level of control to many of the technical features and functions available to the simulation engineer and the package is believed to be an appropriate candidate for our suggested enhancements. An attempt has been made to suggest new or modified functions and even a possible user interface using SIMUL8 as an example.

We have also decided for the purposes of this case study not to address heterogeneous COTS simulation package interoperability.

4.1 Functions

Table 1 gives examples of functions that could be made available in COTS simulation packages such as SIMUL8. The authors of this paper have no knowledge of the internal mechanisms or software design that SIMUL8 uses and so these functions serve merely as general software design suggestions.

At the current time the main body of work has focused on run control and entity exchange. The functions suggested would allow a model to use external objects and variables and also enable the model to share it's own objects and variables. Further, the distribution of an 'input' could also be defined as an external function, providing an alternate method of distribution. A final function is provided to enable a se-

Table 1: Run Control and Entity Exchange Functions

Function	**Description**
Handle ExternObj(Object)	Externalization of objects for external access. Returns handle to object.
Handle Extern(Variable)	Externalization of variables for external access. Returns handle to variable.
SetMaster(Boolean)	Set Master Model - Allows a specific model to be set as a master to stop and start the entire simulation.
Entity GetExternDist(Model, FromObject, ToObject)	Get external distribution - Modify existing routine to interrogate objects within separate SIMUL8 models for distribution patterns. Returns Entity.
Boolean LinkExternal(Model A, Object, Model B, Object)	Links object in model A to an external object in model B. Returns True if successful.

lected model to become 'the master' for ease of control and synchronization of the 'global' model.

The functions in table 1 serve merely as example functions which could exist in an API (Application Program Interface) and are not intended to represent a complete list. However they do serve to highlight some important mechanisms which are required to provide external control and entity exchange with the COTS simulation package.

SIMUL8 supports the notion of Plugins which enable specific software modules to be integrated in to the package. A possible use for this could be for time-management algorithms. This could allow different synchronization protocols to be used when models have been distributed. The Plugins could include Conservative (lookahead, lookback and null message protocols) and Optimistic (Time Warp) algorithms. The integration detail is expected to be more complex for these software components however, the mechanism could provide a neat and elegant solution to the problem.

4.2 SIMUL8 Application Programming Interfaces

Although strictly not relevant to the simulation engineers (due to the requirement of software development skill), the APIs provide the first steps towards interoperability. Once the necessary native functions have been introduced to the application it is not unreasonable to expect separate organizations and even users with software development experience to develop standardized middleware to be used by general simulation engineers in order to allow model interoperability. Currently SIMUL8 supports API's at a number of different levels, i.e. OLE Automation, COM and ActiveX interface. There are also some direct linking facilities, using the user interface, which can enable the user to link to Microsoft Excel or Visual Basic (although these probably use the facilities provided in the API).

4.3 SIMUL8 Interface Suggestions

Modifications to the SIMUL8 interface will be required to enable the Simulation Engineer to design interoperating models. Below are suggested interface enhancements to provide access to the interoperability functionality, primarily focusing on model selection, object linking and setting the master control.

4.3.1 Selecting External Models

The current object linking box in SIMUL8 version 9 provides a mechanism to link various objects within the same simulation, see figure 1. Figure 2 suggests a modification to this dialog box to allow links to be made to external objects by first selecting the model in which the object resides.

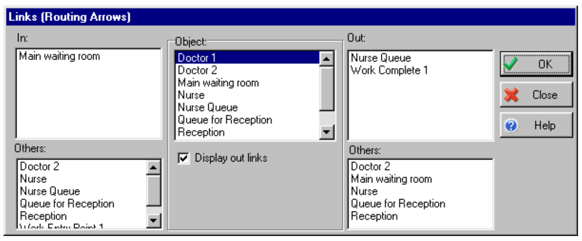


Figure 1: Current Object Linking Dialog

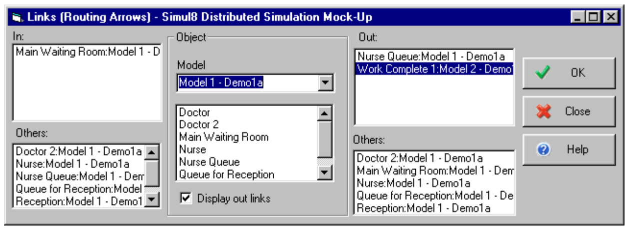
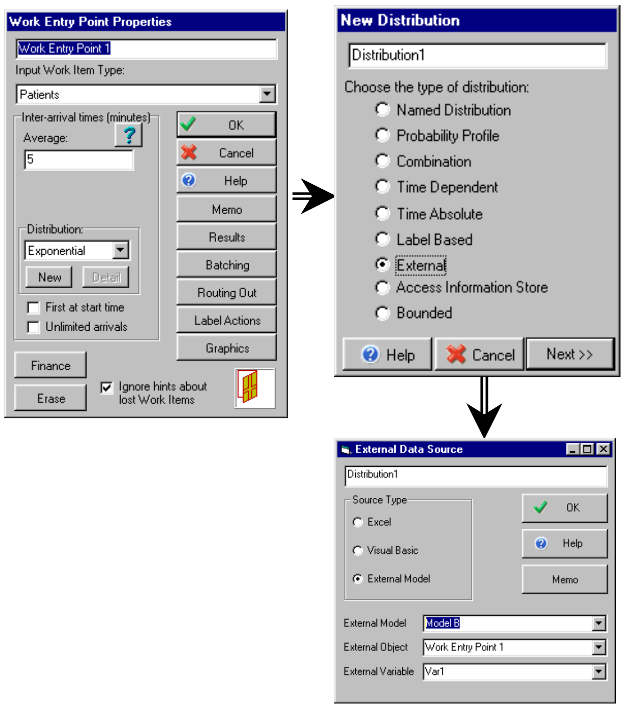

Links (Routing Arrows) - Simul8 Distributed Simulation Mock-Up
In:
Main Waiting Room:Model 1 - D
Others:
Doctor 2:Model 1 - Demo1a
Nurse:Model 1 - Demo1a
Nurse Queue:Model 1 - Dem
Queue for Reception:Model
Reception:Model 1 - Demo1
Object
Model
Model 1 - Demo1a
Doctor
Doctor 2
Main Waiting Room
Nurse
Nurse Queue
Queue for Reception
Display out links
Out:
Nurse Queue:Model 1 - Demo1a
Work Complete 1:Model 2 - Demo
Others:
Doctor 2:Model 1 - Demo1a
Main Waiting Room:Model 1 - Dem
Nurse:Model 1 - Demo1a
Queue for Reception:Model 1 - De
Reception:Model 1 - Demo1a
OK
Close
Help

Figure 2: Modified Object Linking Dialog

4.3.2 Linking External Objects

Once a model has been selected, external objects could then be used for specific distributions. Alternatively, an externalized variable from the model such as a published 're-sults' variable could be used to provide the input. Figure 3 shows an example of the dialog boxes to enable external distribution selection.

The main purpose of creating external distributions is to replace the commonly used stochastic distributions and provide 'real' input in the form of entity occurrences (as opposed to a statistically derived distribution). The input captured for interoperating models could then be used to define, after a number of experimentations, a distribution which could be used within the model. Further implemen-



Work Entry Point 1
Input Work Item Type: Patients
Average: 5
Distribution: Exponential
[ ] First at start time
[ ] Unlimited arrivals
[✓] Ignore hints about lost Work Items

Distribution1
Choose the type of distribution: [✓ External]

Distribution1
Source Type: [✓ External Model]
External Model: Model B
External Object: Work Entry Point 1
External Variable: Var1

Figure 3: Modified External Distribution Dialog

tation could be considered to integrate the process with the 'optimisers' which are often provided in COTS simulation packages. This could provide a mechanism by which experiments could be automated from which a set of distributions could be derived from interoperating models.

4.3.3 Setting Model to be the Master

The modified user interface shown in figure 4 reveals an additional menu option to set the current model to be the master controller for all linked models. This functionality could provide 'central' control for all interoperating models, such as synchronized start and stop.

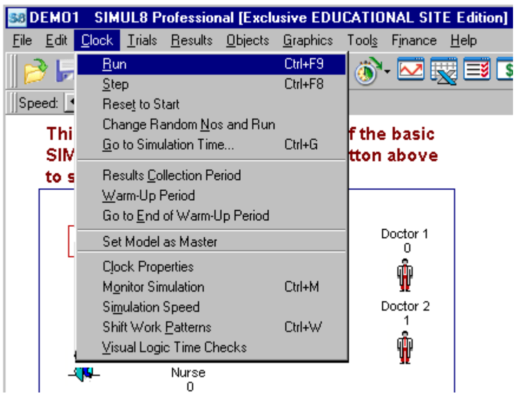

DEMO1 SIMUL8 Professional [Exclusive EDUCATIONAL SITE Edition]
File
Edit
Clock
Trials
Results
Objects
Graphics
Tools
Finance
Help
Run Ctrl+F9
Step Ctrl+F8
Reset to Start
Change Random Nos and Run
Go to Simulation Time... Ctrl+G
Results Collection Period
Warm-Up Period
Go to End of Warm-Up Period
Set Model as Master
Clock Properties
Monitor Simulation Ctrl+M
Simulation Speed
Shift Work Patterns Ctrl+W
Visual Logic Time Checks
Speed:
Doctor 1
0
Doctor 2
1
Nurse
0

Figure 4: Modified Clock Menu

5 EVALUATION

The suggested enhancements demonstrate some of the simple modifications that could possibly be made to SIMUL8, at functional and user interface levels, to provide some degree of interoperability between models. To implement any of the suggested interoperability functionality it is accepted that a significant amount of development work would be required by the vendors of the COTS simulation packages.

The cost justification for such developments are arguable but not clearly defined as suggested in the following 'chicken and egg' scenario. A significant issue that should not be overlooked is that currently simulation modelers cannot create interoperating models unless they have the skills which enable them to write some kind of middleware. Interoperability functionality between simulation packages is rarely if ever native to a simulation package; chicken and egg scenario – many COTS simulation package manufactures do not see the need to provide DS functionality since they believe there is not a demand from their user-base. Users (simulation engineers) will not attempt to build interoperating models unless they have simple tools in which they can be created.

Perhaps we should expect simulation engineers to be au fait with the skills demanded of a software development engineer, after all designing and implementing a large simulation model is often not a trivial task. Often it is a requirement that the engineer be well versed and skilled with not only simulation practices but also the subject area and the intricacies of the simulation tools used. However, this still leaves the issue of the time wasted in developing middleware for potentially each interoperating model and the unlikelihood of the re-usability of the middleware. In reality, simulation engineers do not necessarily have the software expertise to create the required middleware to enable disparate COTS simulation packages to interoperate.

It is also suggested that through COTS simulation package interoperability the compromising of privacy and IPRs based issues may be avoided. External model knowledge would not be required other than required inputs and outputs of the interoperating models.

6 CONCLUSIONS AND SUGGESTED AREAS FOR FURTHER RESEARCH

Only those who have the ability to write middleware, usually programmers, will be able to use COTS simulation packages in a distributed way. It is suggested that it is unreasonable to expect simulation experts to also be experts in programming. It is further suggested that with the inclusion of interoperability (i.e. DS) functionality within mainstream COTS simulation packages that simulation engineers may well be able to harness and make use of work currently accessible only to academics in the DS field.

However, very little exists in terms of tools or methodologies for the simulation engineer to develop large models within a team. Concurrent development of a simulation model would require a tool set and methodologies similar to that used by software engineers. i.e. source code control (or model control) and version control. Further, the paradigm could be extended to include specific development tools for the simulation modeller, for example, determining the best partition points within a simulation – this could be calculated through experimentation, possibly an extension to simulation optimising tools currently available. It also believed that the paradigm could include specific methodologies and practices used in large model development in much the same way that project management and systems management methodologies are used in large IT developments such as PRINCE or SSADM. Extensions to existing software development tools such as UML (the Unified Modelling Language) to include a standardised set of development stages and model definition. If DS/Model interoperability does 'make it' into mainstream products, it is believed that many of the above areas would require further investigation and research.

The main generic processes that are required for SIMUL8 to interoperate with other models (created in the same package) can be summarized as follows:

- To link objects in different models and use their entity distributions where required.
- To pass entity data between objects in disparate models.
- Provision of access to control the starting and stopping of a model externally.
- The implementation of time-management algorithms for model synchronization.
- The ability to interrogate the event list in order to examine the next event before it is executed
- Separate control for re-running C-Phase of operation as specified in the three-phase simulation methodology, see Brooks and Robinson (2001).

REFERENCES

Boer, C.A. and Verbraeck, A. 2002. Connecting High level Distributed Simulation Architectures: An Approach for a FAMAS-HLA Bridge. In *Proceedings of the 14th European Simulation Symposium*. Society for Computer Simulation Publishing House, Erlangen, Germany. 398–405.

Brooks, R.J. and Robinson, S. 2001. *Simulation*. Palgrave, Hampshire, UK 32-35.

Clarke, R. 1993. Module interconnection frameworks for a real-time spreadsheet. Computer Abstracts International Database, reference: 39_1890.

McLean, C. and Riddick, F. 2000. The IMS Mission Architecture for Distributed Manufacturing Simulation. In *Proceedings of the 2000 Winter Simulation Conference*, J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds. Piscataway, New Jersey: Institute for Electrical and Electronics Engineers. 1539-1548.

Strassburger, S. 2001. Distributed Simulation Based on the High Level Architecture in Civilian Application Domains. Society for Computer Simulation Publishing House, Erlangen, Germany.

Sudra, R., Taylor, S. J. E. and Tharumasegaram, J. 2000. Distributed Supply Chain Simulation in GRIDS. In *Proceedings of the 2000 Winter Simulation Conference*, J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds. Piscataway, New Jersey: Institute for Electrical and Electronics Engineers. 356-361.

Taylor, S. J. E., Bruzzone, A., Fujimoto, R., Boon Ping Gan, Straßburger, S. and Paul, R. J. 2002. Distributed Simulation and Industry: Potentials and Pitfalls. *Proceedings of the 2002 Winter Simulation Conference, E. Yücesan, C. H. Chen, J. L. Snowdon, and J. M. Charnes, eds.* Piscataway, New Jersey: Institute for Electrical and Electronics Engineers. 688-694.

AUTHOR BIOGRAPHIES

MICHAEL D. RYDE is a Ph.D. student at the Department of Information Systems and Computing, Brunel University in the United Kingdom. He also received his M.Sc. at Brunel University in 2000 and is a member of the university's Centre for Applied Simulation Modelling (CASM).

SIMON J.E. TAYLOR is the Chair of the Simulation Study Group of the UK Operational Research Society and the collaborative simulation-modelling forum, the GROUPSIM Network (www.groupsim.com). He is a Senior Lecturer in the Department of Information Systems and Computing and is a member of the Centre for Applied Simulation Modelling, both at Brunel University, UK. With Dr Gary Tan of the School of Computing, National University of Singapore he is joint leader of the UK (EPSRC)/Singapore (DSTR)-funded BRUNUSIM distributed simulation research programme. He has an undergraduate degree in Industrial Studies (Sheffield Hallam), a M.Sc. in Computing Studies (Sheffield Hallam) and a Ph.D. in Parallel and Distributed Simulation (Leeds Metropolitan). His main research interest is collaborative simulation modelling. He is also a member of the London-based Purple Theatre Company.