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

HLA-CSPIF PANEL ON COMMERCIAL OFF-THE-SHELF DISTRIBUTED SIMULATION

Simon J. E. Taylor

Centre for Applied Simulation Modelling
Department of Information Systems and Computing
Brunel University
Uxbridge, Middlesex UB8 3PH, U.K.

Boon Ping Gan

Singapore Institute of Manufacturing Technology
71 Nanyang Drive
638075, SINGAPORE

Steffen Straßburger

Fraunhofer Institute for Factory
Operation and Automation
Department Virtual Development and Training
Sandtorstraße 22
39106 Magdeburg, GERMANY

Alexander Verbraeck

Technology, Policy and Management,
Systems Engineering Group
TU Delft
Jaffalaan 5
NL-2628BX Delft, THE NETHERLANDS

ABSTRACT

Commercial-off-the-shelf (COTS) simulation packages are widely used in many areas of industry. Several research groups are attempting to integrate distributed simulation principles and techniques with these packages to potentially give us *COTS distributed simulation.* The High Level Architecture – COTS Simulation Package Interoperation Forum (HLA-CSPIF) is a group of researchers and practitioners that are studying methodological and technological issues in this area. This panel paper presents the views of four members of this forum on the technical problems that must be overcome for this emerging field to be realized.

1 INTRODUCTION

Distributed simulation is an application of distributed systems technology that enables models to be linked together over computer networks so that they work together (or interoperate) during a simulation run. The High Level Architecture (IEEE 1516.2000) is a standard that defines the distributed system technology to make this possible. A Commercial-off-the-shelf (COTS) simulation package is a term used to refer to software used by many simulationists to build and experiment with models. Swain (2001) reviews many of these. There have been various attempts to interoperate models and the COTS simulation packages in which they have been developed. See Boer, et al. (2002a; 2002b), Gan and Turner (2000), Hibino, et al. (2002), Lendermann, et al. (2001), Sudra, et al. (2000), and Taylor et al. (2002b) for examples of these and associated issues. Taylor et al. (2002a) discuss the problems and pitfalls of distributed simulation across a wide range of domains.

Currently many approaches are not compatible. Even for those using the HLA, there is no real standard "use" pattern for the High Level Architecture within the context of this application area. In attempt to create this and to unify research and development activities in this area the High Level Architecture - COTS Simulation Package Interoperability Forum (HLA-CSPIF `www.cspif.com`) was created in August 2002. The ultimate goal of the Forum is to inform and create standards through SISO that will facilitate the interoperation of COTS simulation packages and thus make available to users of such packages the benefits of distributed simulation enjoyed by other modelling and simulation communities. This panel has been convened from four members of the HLA-CSPIF to consider the major problems that are present in the application of distributed simulation principles and techniques to COTS simulation packages.

This paper is structured as follows. First an overview of COTS simulation packages and the HLA-CSPIF is given. The views of each panelist are then presented. The paper finishes with a summary.

2 COTS SIMULATION PACKAGES

COTS simulation packages are used by simulationists mostly for model building, experimentation, animation, visualization and reporting. They have evolved from attempts to build computer environments that support modeling and simulation practice in dynamic process environ-

ments that are found in domains such as business, health, and manufacturing. They are typically based on some variant of the discrete event simulation paradigm, i.e. models change state at discrete points in time by scheduled or conditional events and typically represent entities or objects (documents, patients, parts, trains, etc.) in some form that pass through networks of queues and workstations (work queuing at a desk in an office, patients waiting to see a doctor, parts buffered for machining, trains waiting at a station, etc.) Generally, each package has a range of basic model elements (queue, workstation, resource, source, sink, etc.) that are used to build a model via a drag and drop visual interface. Each model element can be modified as is required, either by a menu system or by a package programming language, to better represent the system being studied (for example the queuing logic of a queue or the behavior of a resource). Entities can be represented and differentiated by attributes. Terminology between packages differs as there is no internationally recognized naming convention.

A COTS simulation packages typically possesses a simulation executive, an event list, a clock, a simulation state and a number of event routines (this is a gross simplification as these packages have many variants of this). The simulation state and the event routines are derived from the model that is implemented in the package. Initializing the simulation, events are placed on the event list (typically modelling entities arriving in the model). If we assume that the simulation executive uses some form of the three phase approach world view (TPA) (other world views exist – see Schriber and Brunner (2002) for examples of others), the simulation first advances clock time to the time of the next event (the A Phase) and then executes that event (the B Phase) according to the event's routine (the event code). This may result in a change in the simulation state and the scheduling of new events on the event list. The simulation executive then determines if the changed state has enabled any conditional events (the C Phase). If any have been enabled, these events are executed in some priority order and any changes to the simulation state or event list are made. The simulation executive continues to make a new cycle of the three phases until some terminating condition is met.

A COTS simulation packages is therefore a support environment in which models are built and experimented on. Interoperating two or more models therefore involves sharing information between the models and the COTS simulation packages in which the models are built. In the next section we introduce the forum dedicated to the interoperation of COTS simulation packages and their models and the creation of widespread COTS distributed simulation.

3 THE HLA-CSPIF

The HLA-CSPIF was created to unify research and developments in the interoperation of COTS Simulation Packages, the .realization of COTS Distributed Simulation. The aim of the HLA-CSPIF is to create a standardized approach to distributed simulation using the IEEE 1516-2000 *High Level Architecture* to support the interoperation of discrete event models created in COTS simulation packages. The objectives of HLA-CSPIF are therefore to

- Involve representative researchers, users, and vendors in the standardization process,
- Create standard *reference model(s)* that can be used to communicate concepts and problems between researchers, users, and vendors in support of the HLA-CSPIF aim,
- Develop a standard data exchange representation (OMT),
- Develop a standard data exchange mechanism (RTI/Federate Ambassadors),
- Develop a standard specification for a distributed simulation co-ordination tool (federate), and
- To report to bodies (such as SISO) that influence the evolution of HLA standards (productization) on the findings of this forum through this publications, workshops, a web site (www.cspif.com) and a SISO reflector (HLA-CSPIF).

As stated, one of the objectives is to create standard reference model(s) that can be used to communicate concepts and problems between researchers, users, and vendors in support of the HLA-CSPIF aim. We now present the views of four members of the Forum on the realization of COTS Distributed Simulation.

4 POSITION STATEMENT OF BOON PING GAN

Simulation is widely used to study the dynamic evolution of manufacturing and supply chain systems arising from their high variability and stochastic uncertainty. Many commercial off-the-shelf (COTS) simulation packages, such as Automod, AutoSched AP, Simple++, Witness, etc, are available to model these systems. But a major drawback of these COTS packages is their inability to interoperate with each other or with the same package when such need arises. For example, in a supply chain simulation that crosses the enterprise boundary, different partners use different COTS packages to build their simulation model. Integrating these models to form a single large supply chain model is a major challenge, especially when distributed simulation technology is not an embedded part of the COTS packages. Having said that, integrating models built on the same package through distributed simulation technology seamlessly also poses a challenging problem. Various technical issues such as time management, event transformation, and object model creation need to be addressed. These issues are discussed here in the context of an industry problem that requires models that are built on the same

COTS package to be integrated together using the High Level Architecture (HLA) standard.

We are talking about a company with several factories that are capable of sharing their capacity. A simulation model for each of these factories has been built using a COTS simulation package. Currently, these simulation models are primarily used for improving the operation of the individual factories. To realize the concept of a borderless factory that allows factories to share capacity seamlessly through movement of lots from one to another, simulation is used as the decision support tool to analyze the impact of the lots movement on the performance of the factories. As the simulation model of each individual factory is already available, distributed simulation using HLA was identified as a natural solution to integrate these models. The primary requirement by the company is that the modelers does not have to know the technical details of realizing distributed simulation. They should be allowed to build their model as if the model is being used for a single node simulation. A middleware approach is used to ensure this transparency. It integrates the simulation engine with the Runtime Infrastructure (RTI) of the HLA, through subscription to system events of the simulation engine.

There are many technical issues that need to be resolved in order to make the COTS package's simulation models talk to each other. Only two of the most critical issues will be discussed here, namely time management and event transformation. Time management addresses the issue of synchronizing multiple copies of the simulation such that the causality constraint is not violated. Event transformation addresses the issue of transparent transformation of an internal event to an interaction when the event needs to be sent to a remote simulation. Also, an interaction needs to be transformed back to an internal event transparently when it reaches the destination. Relating to this issue, some means of allowing the modeler to define the object model needs to be incorporated into the modeling tool of the COTS package.

The HLA standard provides two basic time advancing services: *nextEventRequest* (*NER*) and *timeAdvanceRequest* (*TAR*). The most appropriate service to use for time management is *NER* as each running simulation program needs to synchronize the processing of both external and internal events. An *NER* has to be issued to get the permission from the RTI to make the time progress. The argument to this service is the earliest timestamped event that is currently sitting in the local event list. This event is not supposed to be processed until the RTI grants the time request. Hence, it is mandatory that the COTS package provides some means of obtaining the earliest timestamped event in its local event list, and some means of preventing the simulation engine from progressing its time prior to obtaining the time grant from the RTI. This is easily realized by subscribing to a notification event of the COTS package that is triggered at the end of each simulation event processing. When the notification event is triggered, a *NER* is issued with the earliest timestamp event of the local event list. During the process of waiting for the time grant, an interaction with a smaller timestamp might be received. This interaction is transformed to a local event and inserted to the local event list. When the control returns to the simulation engine, the event with the smallest timestamp is processed. In this way, the causality of the simulation is preserved.

Simulation models in the COTS package are built through several input files, that define the recipes, resources, and probability distribution of the stochastic events. Lots are the basic entity that are moving from one step to another in the simulation. Hence, to facilitate the movement of lots from one model to another, the concept of virtual step is introduced to the modeling tool. The modeler defines the circumstances/conditions at which lots need to be routed to another factory. The lot is then rerouted to the virtual step when the condition arises, that is then transformed to a timestamp order interaction to be sent out through the RTI. This process of transformation is hidden from the modeler. At the receiving end, the interaction is transformed to an internal event, and is inserted to the local event list. The COTS package provides all the basic methods of event creation and insertion that makes this process much simpler.

Resolving the two issues discussed here is not enough to make the COTS package fully functional as a distributed simulation enabled package. Critical issues such as object model creation need to be addressed as well. The discussion here only provides some insights into what are the basic functionalities that a COTS package needs to provide before it can be incorporated with distributed simulation technology to interoperate with itself. The issues will become even more complicated when COTS packages from different vendors need to be integrated. In any case, embedding distributed simulation technology to COTS packages brings tremendous benefits to the industry. In the given example, it enables the use of simulation as a tool to access the concept of borderless factory to improve factory utilization without rebuilding one single large simulation model that is running at a single site. The benefits of model reusability is fully exploited here.

5 POSITION STATEMENT OF STEFFEN STRAßBURGER

With the objective of a Digital Factory, interoperability between simulation models becomes more and more important. A company wide simulation backbone which provides connectivity between all types of simulation applications becomes an appealing idea. The IEEE standard HLA is currently the most advanced standard in the area of distributed simulation and is therefore the state-of-art with which every solution in the area of simulation integration has to compete.

Technical solutions for enabling COTS to talk to each other via HLA exist for some time now (see Straßburger 2001 and Johnson 1999 for examples). They differ in the effort which has been taken to provide a general and user-friendly solution. For material flow simulations as an example, very pragmatic and easy to use solutions exist. They are often based on a simple source-sink principle, extending the source and sink concept of many existing tools for enabling the transfer of moving parts to other models via connected sink-source pairs. The advantage of keeping it that simple is usability. A user simply has to define the connections between the models and is ready to go, assuming some underlying data model has been agreed upon before. Such a solution has been implemented for the simulation systems QUEST and IGRIP based on a Universal Federate Adapter concept (Straßburger, et al. 2003).

Reference FOMs for certain simulation domains could provide a basis for communication here (e.g. by defining the parts which can be transferred in a Manufacturing Reference FOMs). It is recommended to stay with standard HLA mechanisms for implementing the part transfer. A method which has proven successful is to register HLA object instances for moving parts and transfer them with HLA Ownership Management services (Straßburger, et al. 2002). On the first glance this seems to be a heavy-weighted solution, because it would also be sufficient to send a simple interaction message from a sink to the appropriate source and then create the indicated part locally in the target simulation. The disadvantage of the latter is that it introduces a non-standard (i.e., proprietary) solution, which would lead to non-interoperable simulations.

Another technical solution for HLA-enabling simulation systems is to provide a wrapped HLA API for access in the respective simulation system, e.g., as it has been done for the simulation system SLX (Straßburger, et al. 1998). This is a solution which provides the greatest flexibility but also burdens the simulation developers with all the overhead of accessing the HLA interface themselves.

Whichever method for HLA-enabling a COTS is applied, care has to be taken that no proprietary shortcuts are taken. This is a danger if everybody adapts their own way of using HLA. It is advisable to stick to the core functionality which HLA suggests. Sometimes this is not unambiguously possible, e.g., if there are multiple ways of achieving the same task, or if HLA interface services invite proprietary solutions. Ideally, simulation system vendors should therefore sit together and define a standard way of integrating HLA into their systems. One groups which has started work in this area is the HLA-CSPIF group (see `www.cspif.com`).

As said before, interoperability between simulation systems on a technical level is solvable. It can be done and is already applied, e.g., in pilot projects at DaimlerChrysler. The only real major technical difficulty which still needs improvement is providing efficient **and** easy to use synchronization mechanisms. Existing mechanisms which are easy to use (like time-stepped execution or conservative synchronization of all internal events based on HLA's *nextEventRequest*) have the major disadvantage of being rather inefficient. More advanced mechanisms, like identifying the externally relevant events and only synchronizing them are rather complicated to implement and highly depend on the simulation model. Support for applying optimistic synchronization techniques based on rollback mechanisms are, to the authors knowledge, only supported by the simulation system SLX (Henriksen 1997), which offers efficient state saving capabilities.

In summary, technical solutions for applying HLA for integrating simulation models developed in different commercial off-the-shelf simulation systems exist and can already be applied. It is, of course, advisable to be very careful in order to partition the models in an appropriate way. Interfaces between models should (for reasons of performance and simplicity) be kept as simple and lean as possible. Keeping that in mind HLA-based simulation integration can become a reality today.

6 POSITION STATEMENT OF SIMON J. E. TAYLOR

The range of approaches and solutions to COTS distributed simulation appear to increase each year. In WSC 2003 there are examples of the use of this technique with several different COTS simulation packages and middleware including (but not limited to) the High Level Architecture. As understanding of what COTS distributed simulation is, demand (slowly) increases. However, current solutions are isolated and only allow interoperability between COTS simulation packages within the same group *even* between solutions based on the HLA.

Specifically, the interoperation problem is this. Irrespective of translation mechanisms (such as ambassadors or adaptors, etc.) one federate composed of a COTS simulation package and its model must publish information to an RTI in a certain format and manner. Another COTS simulation package/model federate must receive that information in a certain format and manner, i.e. both federates must agree on a common representation of data and both must use the RTI in a similar way. If information format and RTI use are at all different then interoperation cannot happen. For example, if one federate exchanges information concerning the transfer of an entity *Part* to the other federate, we have the option of representing this as a published object or interaction. If we represent the entity as an object then we might use

- Object Class Structure table
- Attribute table
- Fixed Record Datatype table
- Enumerated Datatype table

If, however, we were to use interactions, then we might use

- Interaction Class Structure table
- Parameter table
- Fixed Record Datatype table
- Enumerated Datatype table

i.e. a different set of tables. If one distributed simulation solution provide was to use objects and another then our COTS distributed simulation could not exchange entities between federates. The moral of this story is therefore that in order to develop COTS distributed simulation where any COTS simulation package can potentially exchange information with any other, there must be agreement on the format of information and the methods by which it is exchanged. Additionally, this integration must also be harmonized in terms of data types, names, and language (colour vs color!)

How can we make progress in this area? One solution might be to identify the kind of information that is shared between models as trying to solve everything at once is difficult! Additionally, it is important to develop common understanding as to *what* is shared – there are many different concepts in simulation modelling and a common understanding of these do not necessarily exist! So how can we develop the types of information that can be shared and a common basis for understanding? The current solution has been the development of *reference models* (RMs). These are the first deliverable from the HLA-CSPIF. Each RM is intended to represent a problem *type*, a general class of interoperability problem. The RMs identified so far are:

- Type I Reference Model (Asynchronous Entity Passing)
- Type II Reference Model (Synchronous Entity Passing) (Bounded Buffer)
- Type III Reference Model (Shared Resources)
- Type IV Reference Model (Shared Events)
- Type V Reference Model (Shared Data Structure)
- Type VI Reference Model (Shared Conveyor)

For further details on these, visit the HLA-CSPIF web site at `www.cspif.com`. It is hoped that these will form the basis for true international progress in COTS distributed simulation.

7 POSITION STATEMENT OF ALEXANDER VERBRAECK

The incorporation of COTS discrete event simulation models into a federation of models is not an easy task. Most COTS simulation languages are not able to use protocols such as HLA, and in spite of possibilities to link externally written code in C++, VBA, or Java to the simulation models, many of the internal functions and variables needed to create a distributed model are not exposed for the externally linked modules.

But even when this technical problem has been solved – and it has been shown in several projects that it is possible to link models developed in different COTS simulation languages (Boer, et al. 2002a; Boer, et al. 2002b, and other examples from the CSPIF initiative) – an even more serious challenge surfaces. When models written in different simulation languages share or exchange entities, the entities have to be mapped from the formalism of one COTS simulation language onto the formalism of the other language. Consider the following situation. When two simulation models interact they might need to transfer a simulation entity from one model to the other one. Say that the entity being transferred is a *truck*, then we might represent the abstract truck entity type as $\widetilde{E}_{truck}$. At simulation time t the sender creates a truck entity instance $E_i \in \widetilde{E}_{truck}$ that is transferred to the second model. Due to the fact that the truck entity is transferred to the second model, both models should provide the possibility to instantiate the $\widetilde{E}_{truck}$ abstract entity. Unfortunately, this solution is not supported in most of the cases.

During the transfer, the entity instance is always reduced to the state information that it contains, i.e. its attributes and a description of the types of the attributes. The sending COTS simulation model serializes the entity in some way, and the receiving model deserializes it. During the transfer, any communication mechanism such as RMI, Corba, or even plain TCP/IP can be used to transfer the serialized entity from one model to the other. If the simulation packages are the same and if they work with a similar set of entities, both the sender and receiver can instantiate the same type of entities (e.g. a truck entity). However, if the sets of entities are different, the instantiation is very difficult if not impossible. The same holds when the two models are coded in different COTS simulation packages. Hand-coded wrappers in external code might help to do the trick, but this is quite labor-intensive. In some of the cases the problem can be solved using syntactical analyzers by checking the definitions of the entities. For example, in the sender model the abstract entity of the transferred entity is a truck $\widetilde{E}_{truck}$. The receiver model might miss the truck entity, but might contain an abstract lorry entity $\widetilde{E}_{lorry}$. The truck and lorry entities are basically the same (described by the same attributes), but they are defined by a different name. The aim of a syntactical analyzers is to find syntactical errors and to discover the possible matching between different types of entities (e.g. $\widetilde{E}_{truck}$ and $\widetilde{E}_{lorry}$). When the receiver model cannot instantiate any transportation type entity (e.g. $\widetilde{E}_{truck}$ and $\widetilde{E}_{lorry}$) then the transfer of this kind of entities cannot occur. Basically in this situation the receiver is not allowed to subscribe for any trans-

portation entity transfer. Further, in some of the cases the abstract entity names are the same but they define different attribute sets. For example, both the first and second models can instantiate an $\widetilde{E}_{truck}$ abstract entity, but in the second model the $\widetilde{E}_{truck}$ does not include the definition of the information about its shipment. Semantic analyzers might be applied in order to tackle this problem, but they need access to additional descriptive information of the entity definitions. Moreover, if the set of entities and the set of attributes are the same we might still be confronted with the problem that the type of the attributes differs. The wrapper might be able to solve this. The first simulation model transfers an entity to the second simulation model. The second model is able to instantiate the same abstract entity but some of the attribute types of the instantiated transferred entity differ. Therefore the wrapper of the second model creates a temporal entity instance table and maps the original attributes of the transferred entity with those that the second model supports. What the analyzers cannot solve, however, is a difference in meaning or use of the entity within the models.

Although wrappers are able to solve the syntactical differences between similar entities in a federation, the semantics remain a problem that can, in my opinion, at this moment only be solved by *human* analysis. A good, formalized, and standardized description of the semantics of simulation entities and simulation variables would be a big help to overcome this hurdle.

8 SUMMARY

This paper has presented four views on COTS distributed simulation from members of the HLA-CSPIF. It is hoped that this will foster further discussion that will lead to making distributed simulation generically available to uses of COTS simulation packages.

REFERENCES

Boer, C. A., A. Verbraeck, and H.P.M. Veeke. 2002a. The Possible Role of a Backbone Architecture in Real-Time Control and Emulation. In *Proceedings of the 2002 Winter Simulation Conference,* E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes (eds.) Association for Computing Machinery Press, New York, NY. 1675-1682.

Boer, C.A., A. Verbraeck and H.P.M. Veeke. 2002b. Distributed Simulation of Complex Systems: Application in Container Handling. In *Proceedings of SISO European Simulation Interoperability Workshop.* Simulation Interoperability Standards Organisation, Orlando, Florida.

Gan, B.P., and S.J. Turner. 2000. An asynchronous protocol for virtual factory simulation on shared memory multiprocessor systems. *Journal of Operational Research Society,* 51: 413-422.

Henriksen, J.O. 1997 *An Introduction to SLX.* In *Proceedings of the 1997 Winter Simulation Conference,* S. Andradóttir, K.J. Healy, D.H. Withers, B.L. Nelson (eds.) Association for Computing Machinery Press, New York, NY. 559-566.

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,* E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes (eds.) Association for Computing Machinery Press, New York, NY. 1099-1107.

Johnson, G. 1999. *Networked Simulation with HLA and MODSIM III.* In *Proceedings of the 1999 Winter Simulation Conference,* P. Farrington, H. Nembhard, D. Sturrock. and G. Evans (eds.) Association for Computing Machinery Press, New York, NY. 1065-1070.

Lendermann, P., B.P. Gan and L.F. McGinnis. 2001. Distributed Simulation with Incorporated APS Procedures for High-Fidelity Supply Chain Optimization. In *Proceedings of the 2001 Winter Simulation Conference,* B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, (eds.) Association for Computing Machinery Press, New York, NY. 1138-1145.

Schriber, T.J. and D.T. Brunner. 2002. Inside discrete-event simulation software: How it works and why it matters. In *Proceedings of the 2002 Winter Simulation Conference,* E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes (eds.) Association for Computing Machinery Press, New York, NY. 97-107.

Straßburger, S. 2001. *Distributed Simulation Based on the High Level Architecture in Civilian Application Domains.* Ghent: Society for Computer Simulation International, ISBN 1-56555-218-0.

Straßburger, S., A. Hamm, G. Schmidgall, and S. Haasis. 2002. *Using HLA Ownership Management in Distributed Material Flow Simulations.* In *Proceedings of the 2002 European Simulation Interoperability Workshop.* Simulation Interoperability Standards Organisation, Orlando, Florida.

Straßburger, S., G. Schmidgall and S. Haasis. 2003. *Distributed Manufacturing Simulation as an Enabling Technology for the Digital Factory.* Journal of Advanced Manufacturing Systems (JAMS). 2:1, 111-126.

Straßburger, S., T. Schulze, U. Klein and J.O. Henriksen. 1998. Internet-based Simulation using off-the-shelf Simulation Tools and HLA. In *Proceedings of the 1998 Winter Simulation Conference,* Medeiros, D.J. and E. Watson (eds.) Association for Computing Machinery Press, New York, NY.

Sudra R., S.J.E Taylor and T. Janahan. 2000. Distributed Supply Chain Management in GRIDS. In *Proceedings of the 2000 Winter Simulation Conference,* J. A.

Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds. Association for Computing Machinery Press, New York, NY. 356-361. Association for Computing Machinery Press, New York, NY.

Swain, J. J. 2001. Power tools for visualization and decision making: 2001 simulation software survey. *OR/MS Today* 28(1): 52-63.

Taylor, S.J.E., A. Bruzzone, R. Fujimoto, B.P. Gan, S. Strassburger and R.J. Paul. 2002a. Distributed Simulation and Industry: Potentials and Pitfalls. In *Proceedings of the 2002 Winter Simulation Conference*, E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes (eds.) 688-694. Association for Computing Machinery Press, New York, NY.

Taylor, S.J.E., R. Sudra, T. Janahan, G. Tan and J. Ladbrook. 2002b. GRIDS-SCS: An Infrastructure for Distributed Supply Chain Simulation. *SIMULATION.* 78(5): 312-320.

AUTHOR BIOGRAPHIES

SIMON J.E. TAYLOR is the convener of this panel. He is also Information Director of ACM SIGSIM, ACM SIGSIM PADS Liaison Officer and Chair of the Simulation Study Group of the UK Operational Research Society. He is a steering committee member of PADS, DS-RT and general co-chair of the UK Simulation Workshop Series. He currently leads the international COTS simulation package interoperability forum (HLA-CSPIF) through SISO (`www.cspif.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. He was previously part of the Centre for Parallel Computing at the University of Westminster. 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 interests are distributed simulation and applications of simulation health care. He is also a member of the Purple Theatre Company. His email is <`simon.taylor@brunel.ac.uk`>.

BOON PING GAN is a Research Engineer with the Production and Logistics Planning Group at Singapore Institute of Manufacturing Technology (formerly known as Gintic Institute of Manufacturing Technology). He is currently leading a research project that attempts to apply distributed simulation technology for supply chain simulation. He received a Bachelor of Applied Science in Computer Engineering and Master of Applied Science from Nanyang Technological University of Singapore in 1995 and 1998 respectively. His research interests are parallel and distributed simulation, parallel programs scheduling, and application of genetic algorithms. His email address is <`bpgan@SIMTech.a-star.edu.sg`>.

STEFFEN STRASSBURGER is manager of the department "Virtual Development" at the Fraunhofer Institute for Factory Operation and Automation in Magdeburg, Germany. Previous professional experience includes two years as researcher at the DaimlerChrysler Research Center in Ulm, Germany, where he was involved with research topics in the Digital Factory and Digital Engineering context, esp. in the area of simulation integration and distributed simulation. He holds a PhD and a Master's degree in Computer Science from the Otto-von-Guericke University in Magdeburg, Germany. His international experience includes a one-year stay at the University of Wisconsin, Stevens Point and a stay at the Georgia Institute of Technology, Atlanta. He actively participates in several international conferences. His main research interests lie in distributed and web-based simulation, virtual reality techniques, and middleware technologies like the High Level Architecture, CORBA, DCOM, and Web Services.

ALEXANDER VERBRAECK is an Associate Professor in the Systems Engineering Group of the Faculty of Technology, Policy and Management of Delft University of Technology, and part-time research professor in supply chain management at the R.H. Smith School of Business of the University of Maryland. He is a specialist in discrete event simulation both real-time analysis and control of complex transportation systems and for modeling business systems. His current research focus is on the development of generic libraries of distributed object oriented simulation building blocks and the development of Java-based simulation languages for Web-services . His email address is <`a.verbraeck@tbm.tudelft.nl`>.