Proceedings of the 1999 Winter Simulation Conference.
P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

ON-LINE DATA PROCESSING IN SIMULATION MODELS: NEW APPROACHES AND POSSIBILITIES THROUGH HLA

Thomas Schulze
Steffen Straßburger

Department of Computer Science
Otto-von-Guericke University Magdeburg
Universitätsplatz 2
39106 Magdeburg, GERMANY

Ulrich Klein

Fraunhofer Institute for
Factory Operation and Automation
Sandtorstrasse 22
39106 Magdeburg, GERMANY

ABSTRACT

The United States Department of Defense's High Level Architecture for Modeling and Simulation (HLA) provides a standardized interface for distributed simulations. The recent advent of HLA has greatly increased interest in the use of distributed, interoperable simulation model components. This paper focuses on how on-line data (i.e. data from real-time dependent processes) can be used in analytical simulation models and how the use of HLA based components can facilitate the integration of this kind of data into simulations. The paper also discusses the issue of cloning federates and federations and introduces some potential applications of cloning for a public transportation prototype example.

1 INTRODUCTION / MOTIVATION

The need for flexibility and cost effectiveness of software systems over a period of time and changing demands is imminent, which also holds true for simulation based applications.

Therefore, modular and flexible approaches, like component based systems and architectures, such as CORBA, gain more and more importance.

Moreover, the on-line connection to the information generating processes of the "real" world allows simulations to enter on-line/real-time application areas. If the integration of (process) information can be designed transparently (i.e. the simulation uses on-line and off-line sources the same way), the applicability is further enhanced.

2 HLA

The High Level Architecture for Modeling and Simulation (HLA) has been developed since 1996 by the U.S. Department of Defense (DoD) Defense Simulation and Modeling Office (DMSO 1997). HLA is scheduled to become the IEEE standard 1516 by the end of 1999.

Furthermore the HLA interface specification was adopted as the CORBA Facility for Distributed Simulation Systems in November 1998.

The HLA follows a framework approach and is defined by three major elements:

- *Rules* govern the behavior of the overall distributed simulation (Federation) and their members (Federates).
- an *Interface Specification* prescribes the interface between each federate and *the Runtime Infrastructure* (RTI), which provides communication and coordination services to the federates.
- an *Object Model Template* (OMT) which defines the way in which federations and federates have to be documented (using the Federation Object Model FOM and the Simulation Object Model SOM, respectively). Federations can be viewed as a contract between federates on how a common federation execution is intended to run.

Federates are only described by their SOM and, therefore, act as "black boxes". This feature can be exploited for dynamic configuration and composition of federations which will be dealt with in the following section.

3 THE COMPOSITION PRINCIPLE

In general, software can be seen as a tool for solving problems or for performing tasks. In the context of simulation, certain simulation models can be used for the estimation of performance parameters of planned or existing systems. In other contexts, text processors are applied for writing letters, papers, etc. The conclusion is

that there is no general tool that can be used for all possible tasks, and that there will not be such a tool in the future.

If different tools are needed, there are two main cases which can be distinguished:

A) The tool already exists and the tool will be reused.
B) A new tool has to be constructed.

The case A can be divided into two sub-cases:

A1) The selected existing tool will completely fit the problem.
A2) The selected existing tool must be adopted for fitting the problem.

Case A1 usually applies if known problems are to be solved. Existing tools can be applied easily for known tasks. This is very straightforward, of course. On the other side new problems appear all the time. Existing tools may totally fit the problems. However, it is not possible to recognize all possible use cases of a tool in the future during the initial tool development process. The conclusion is that existing tools will be adopted. If no existing tool can be modified to fit a user's needs there is no other way than to build a new tool. This leads to case B.

Case A2 is characterized by adopting existing tools. The functionality of the tool will be changed. There are two possibilities which can be distinguished:

A2a) The tool will be expanded with new functions.

The use of add-ins is an example for this case. This approach often has the disadvantage that the new tool includes functions that will not be used.

A2b) Necessary functions can be selected during the configuration process.

This approach overcomes the special disadvantage of case A2a. Only available functions and functions which are actually needed will be combined. The freedom for combining different functions is often restricted. The general disadvantage of case A2 is that only available, homogeneous and compatible functions can be added or combined.

For constructing a new tool as characterized in case B) there are two different approaches:

B1) All parts of the tool will be developed from scratch.
B2) Components will be combined to form a new tool.

A component is a constituent element of a system. All or some components will be combined to form a new tool. This approach is called the *composition approach*. The components must be interoperable for being usable in such a composition. It is not significant whether the components already exist or whether the components have yet to be developed. The combining of components to form a new tool will be called composition.

The composition principle is a well known principle in computer science. For example, the bottom-up approach is based on this principle. One significant characteristic of the composition principle is the interoperability between components. If the components are homogenous, e.g. if they run on the same processor, use the identical operating system or are implemented in the same language, then the components have an intrinsic interoperability.

If the composition principle will be applied to heterogeneous components, then the components must be interoperable and a suitable environment for information exchange must exist. CORBA is one of such environments available.

If simulation models are to be used as components there are additional issues which must be dealt with, e.g. the synchronization of the time advancement in the simulation models. The causality in event processing must be guaranteed. HLA offers such services in its interface specification and its RTI-software.

One of the original aims of HLA is to support the interoperability and synchronization between heterogeneous simulation models. Our work tries to expand these aims in order to build problem solving tools based on the composition principle. The use of HLA enables the integration of components that have their own local time like simulation models and animations.

The application of an composition approach in an public transportation prototype is elaborated on in section 6. The next section deals with the integration of on-line data into simulation models and will also be used for describing different possibilities offered by the composition principle.

4 INTEGRATION OF ON-LINE/REAL-TIME DATA

Prior to the description of the approach taken to integrate on-line/real-time information into HLA-based systems, the following definitions for on-line and real-time data used by the authors shall be given:

- *On-line* differs from *off-line* in that no intermediate storage is used to store input data. Classical analytical simulation models usually use off-line data.
- Various definitions of *real-time* exist with timing requirements ranging from microseconds

to minutes, depending on the application area. Common in all definitions is that incoming information is processed under timing constraints associated with costs violating them (soft deadlines refer to additional costs that occur in case of violation, while violating hard deadlines is seen as costs approaching infinity).

An integration approach which intends to transparently integrate on-line/real-time information must take this variety into account.

Under HLA, this integration can be realized by an on-line federate which acts as provider for the rest of the federates within a given federation.

The possible structure of an on-line federate is shown in Figure 1.

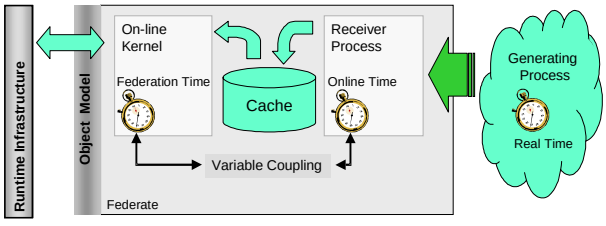

Figure 1: Suggested structure of an on-line federate

The on-line federate has two major tasks:

1) to receive the real-time/on-line data from a generating process
2) to possibly buffer this data and send updates about it into the federation at the according federation time.

These two major tasks should ideally be implemented using two distinct operating system threads. This provides more flexibility regarding the time advancement mechanisms the federate can use and also prevents the federation from possible deadlocks if the on-line source terminates unexpectedly.

Generally, there can be different alternatives regarding the time advancement of the on-line federate. In the context of HLA a federate has two properties regarding its synchronization: constrained and regulating. Both properties can be set to true or false, resulting in four major types of federates:

1) Not constrained, not regulating

   Federates with these properties are not constrained by other federates in their local time advancement and do not act regulating on other federates. An on-line federate with these properties could simply send the on-line data it receives as receive-order-events into the federation, i.e. updates or interactions are sent when they arrive, without any buffering in between.

2) Not constrained, regulating

   For an on-line federate these settings seem to be the most appropriate ones: The federate acts regulating on other federates (because it generates events with time stamps which the other federates have to take for granted), but is not constrained by other federates. This seems to be most obvious: The time stamp of the on-line data is fixed and independent from the federation time, it cannot easily subordinate itself to the federation time. Being constrained could result in sending events with time stamps less than the federation time, which is clearly not allowed.

   In this alternative the on-line federate could act as a pacemaker for the entire federation.

3) Constrained, regulating

   Under certain conditions it may be desirable to have an on-line federate which is constrained as well as regulating. In this case incoming on-line data needs to be buffered. Time advance requests have to be made for the next time stamp of an on-line data update. If no such event is yet received, the federate will block all other constrained federates until it receives the next on-line event.

4) Constrained, not regulating

   The last combination can be used if the on-line federate has to subordinate itself to the federation time advancement, but is not allowed to slow down other federates. In this case it is again not possible for the federate to send the updates as time stamp ordered events, but only as receive-order-events.

With respect to the HLA software structure and the properties discussed above the following figure (Figure 2) gives a detailed view on the possible structure of an on-line federate.

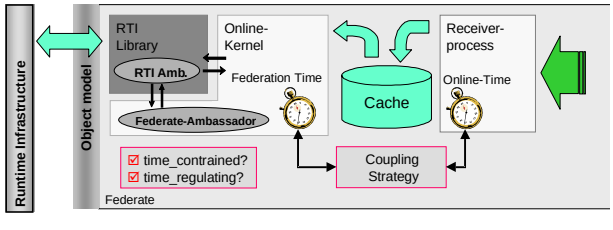

Figure 2: Detailed structure of an on-line federate with respect to HLA

The application of this concept for an on-line federate in a public transportation prototype is elaborated on in section 6. The next section deals with the possibilities which arise from cloning federates, which will also be described using the transportation prototype example.

5 CLONING HLA-BASED DISTRIBUTED SIMULATIONS

The Merriam-Webster Dictionary explains the following meanings of the word "clone":

> Main Entry: **clone**, Function: *verb*,
> Inflected Form(s): cloned; clon·ing, Date: circa 1948
> transitive senses:
> 1 : to propagate a clone from; 2 : to make a copy of.

5.1 Overview

Cloning in relation to HLA refers to the process of making a copy of federates or federations. To clone a federate means to create a federate with identical SOM and, if copied during runtime of a federation, with identical current status regarding objects, time management, etc.

To clone a federation means to create a federation with identical FOM and populate it with federates with SOMs identical to the original ones. Additionally, the federates and the RTI need to have an identical current status regarding objects, time management, etc. (this could be relaxed depending on the application).

Two main cases could be identified:

1) if the copy of one or more federates is to operate on the same time axis (regarding point of time and time management), then the copy could either be allowed to reside within the same federation or could form a separate one.
2) if the copy of a federate or a federation needs to use a different time axis in comparison to the original federate or federation (e.g. for as-fast-as-possible forecasts), then a separate federation has to be created and joined by the federates during the cloning operation.

The first approach is somewhat similar to the approach described in (Hybinette and Fujimoto 1997). In this approach logical processes (LPs) of a parallel simulation are cloned and kept as *virtual* logical processes, as long as (to keep it simple) they do not behave differently from the original ones (i.e. as long as they do not send or receive messages different from the original LP).

One could think of the same possibility of distinguishing between "virtual" and "real" copies or clones in the context of HLA federates as it is done for logical processes in this approach.

In order to distinguish messages from real federates and cloned ones, additional information needs to be inserted into the FOM and SOMs. With this additional information, messages could be distinguished to be relevant to the real federates or to the copies. It has to be noted, though, that this approach requires additional functionality inside the federates, whereas in (Hybinette and Fujimoto 1997) the logical processes are not concerned with who the sender / recipient of messages is.

With the second approach mentioned above, it would become possible (by fully exploiting the services offered by HLA) to achieve a more dynamic possibility of cloning federates. Considering a fully initialized training or simulation scenario, possibly reflecting a real-word situation by using on-line or real-time data, it would be desirable to have the possibility to clone either one or more federates or even the entire federation into a *new* federation.

This federation could then be used to test certain decisions a operator might take for their effects on the simulation. The major difference in the approach suggested by us in comparison to the approach suggested by Hybinette/Fujimoto lies in the fact that the separate copy can run independently regarding the advancement of logical time.

It has to be noted that the motivation for the approach taken by Hybinette/Fujimoto was not to have independent clones *per se*, but rather to have as few as possible copies of logical processes running at the same time due to performance and resource issues. This is not the primary motivation for our approach.

5.2 Copying a Federate During Runtime

To make a copy of a federate might be element of a runtime federate exchange procedure, a setup process of a parallel federation, or just the transfer of status to an additional federate within the current federation.

In order to have a generic approach to federate cloning which can be applied in most of the situations, several interface services which are part of the federation management group can be used. This especially applies to the services for synchronization of federates and the save/restore functionality.

As each of the federates is responsible for the internals of the "black box" (from the viewpoint of the HLA), the details on how each federate stores its internal data is beyond the scope of this general approach. Federates with identical object models that are foreseen to be exchanged during runtime must be able to store and read the

information in a format that could be read or written by the other federates.

A general procedure might be as follows:

1. Synchronize the federates in the federation (pause function)
2. Save the status of one/some/all federate(s)
3. Create the cloned federate
4. Initialize the cloned federate with stored status
5. Let the cloned federate join the appropriate federation
6. Continue the federation.

Depending on the details of the operation to be performed, the procedure might be relaxed and some steps omitted.

How Step 3 is realized depends on the individual federate. In most cases it will be necessary to start up a new process. In some cases the federate might also be able to create more than one instance of the federate/RTI ambassadors objects, thus becoming member of multiple federation executions. Such a federate might even be able to manage its internal state using different time management environments at the same time, if the clone will reside in an independent federation execution.

5.3 Copying a Federation During Runtime

The cloning of a federation can take two forms depending on whether the cloned federation uses an own time management and, therefore, needs to reside within an own federation execution.

If the cloned federation is to use a separate federation execution, then a federation-wide synchronization and saving process is needed to build the consistent status information from all the federates and the RTI.

Then a separate federation execution must be created and joined by the newly created and initialized cloned federates.

This process has to be automated if it is to be of practical use. This implies the existence of a managing "federate" which, e.g. by using the MOM, is able to start processes (federates) and create/manage federation executions. Such a managing federate would also be in charge of providing results and information from the cloned federation to the original federation (in case of an as-fast-as-possible forecast clone).

5.4 Applications

Applications can be identified in various fields where real-time systems (like air traffic management systems or command and control systems) should be able to do forecasts based on the actual situation. Another application is the evaluation of alternatives.

To copy a federate might also be seen as elementary step for run-time exchange of federates.

Also, copies of federates on separate machines may be used for load balancing.

6 PUBLIC TRANSPORT PROTOTYPE

The prototype described in this section was implemented to:

- show the applicability of the composition approach based on HLA federates as described in section 3
- demonstrate the integration of on-line data into HLA federations using the on-line federate approach described in section 4
- prepare a platform for cloning procedures and show possible applications of the cloning concepts described in section 5.

The federation is a public traffic management scenario in the city of Magdeburg, Germany, and was developed with the support of the Magdeburg traffic company (MVB).

The following section describes which components were generally needed for a federation of the kind that was to be developed.

6.1 Composing the Public Traffic Management Tool

Based on the composition principle, suitable combinations of components allow the building of qualified tools to perform different tasks. The spectrum includes tasks for planning, developing and introducing new strategies and staff training in existing systems. We identify the following components in the area of traffic management for urban streetcars and busses:

Simulation: Monolithic or distributed simulation models for the simulation of streetcar and bus traffic processes in the urban area. Different existing simulation models can be used for busses and streetcars. The simulation model must be able to process external data (like the current position of the busses or cars) to initialize or update the model state. Additionally the models have to support the process of different kinds of scenarios for training and planning applications.

Animation: The animation component has to visualize the simulated or real processes. The component must be able to show the animation in different animation speeds. For the use in training applications the animation component has to allow interaction with the user and real-time capability. Animation components can work at different locations and with varying tasks. The management needs to see a different animation than the passengers in passenger information system.

On-line data: The on-line data component sends the external on-line process data to the other components. This component prepares the collected data from the real processes. Information about the position of streetcars or busses is a typical example.

Off-line data: The off-line data components contain additional constant data for initialization of the simulation and animation components. These components ought to allow access to data base systems, file systems or geographical information systems (GIS).

A possible structure of the federation is shown in Figure 3.

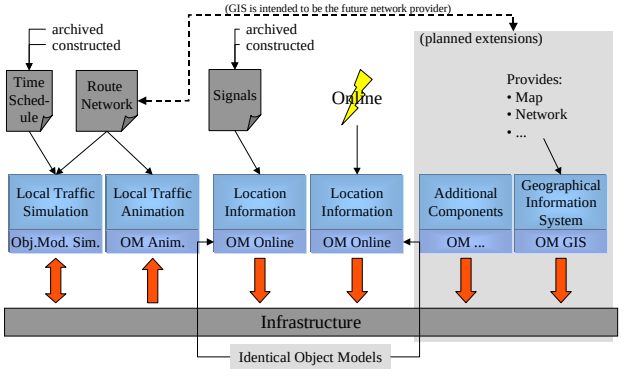

Figure 3: Setup of the traffic management prototype based on the composition principle

6.2 Implementing the Prototype for the City of Magdeburg

The following sections describe each of the federates used for the public transport prototype.

6.2.1 The Simulation Federate

A simulation model which performs a schedule based simulation of the public transportation system in Magdeburg (i.e. streetcars, busses) has been developed using the simulation system SLX (Henriksen 1997).

The simulation model uses the SLX-HLA-Interface (Straßburger et al. 1998) to be able to receive time-stamp-ordered events containing position updates of the simulated objects and to synchronize its local simulation clock with other federates. The simulation model itself can be seen as a classical analytical simulation model using logical simulation time.

6.2.2 The Animation Federates

Two different tools can be used for producing on-line animations of the federation.

The first tool is the web-based animation system Skopeo. Skopeo is a general animation system which provides platform-independent system animation anywhere in the WWW (Dorwarth et al. 1997), (Lorenz and Ritter 1997).

The second tool which has been used in this federation is Proof Animation for Windows. Proof Animation provides on-line animation on Windows Platforms and can be used by a wide variety of programs. In order to produce on-line animation with Proof, a program to drive Proof (in its Windows Dynamic Link Library version) is needed. Since Proof is tightly integrated with SLX and an HLA-Interface for SLX was already available, it was decided to use SLX to drive Proof Animation.

There were two general alternatives considered to use the combination Proof/SLX in the traffic management prototype:

1) Use the existing SLX-simulation model to also produce the on-line animation.
2) Use a different SLX-program acting as a federate which only receives the position updates and produces an on-line animation on its own.

Both alternatives have certain advantages. Alternative 1 is very easy to implement, since everything runs in one federate. Also, this alternative saves on the bandwidth, since position updates do not necessarily have to be sent into the federation.

Alternative 2 is more general, since it allows for more than one animation to take place. Different views on the animation can be selected at the same time by different users.

The default animation for the streetcar prototype uses alternative 1. Additional animations can be obtained at the same time (*non-exclusively*) by Skopeo and potentially by a Proof/SLX-federate based on approach 2.

Figure 4 shows a screenshot of the system obtained with Proof Animation.

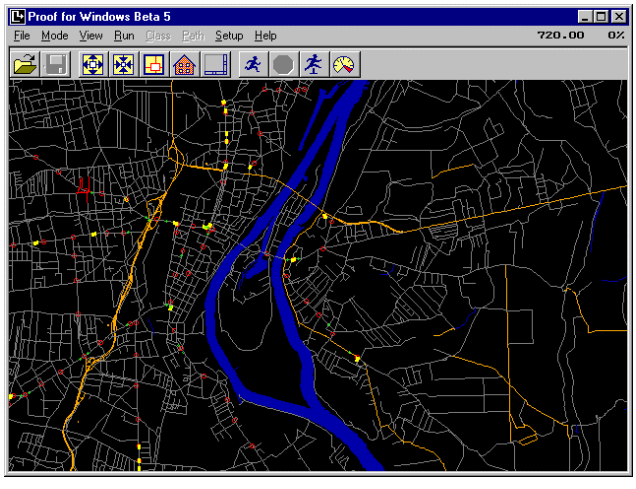

Figure 4: Screenshot of the streetcar federation obtained with Proof Animation for Windows

6.2.3 The On-line Federate

An on-line federate based on the concept outlined in section 4 has been developed based on SLX and the SLX-HLA-Interface.

The on-line federate starts a receiver process implemented as a separate thread to connect to the command and control computer of the Magdeburg traffic company. From there it receives position updates which are obtained from the "real-life" streetcars using infrared senders which each streetcar carries.

The receiver process buffers the position updates in an internal cache which can be read by the actual federate thread asynchronously. The federate thread is time-regulating in the HLA-sense. Since position updates are sometimes received with large time intervals, the federate thread also runs real-time synchronized via the system clock and announces its time advancement in fixed intervals. In our case the "time constrained" condition can be neglected, since the on-line federate is the only driving force in the federation and, also, does not depend from any other federate.

6.2.4 The "Off-line" Federate

An off-line federate which has the same object model as the on-line federate can be used to substitute the on-line federate. This can be useful for testing certain scenarios, e.g. operator training, or to replay a situation for further analysis. The off-line federate uses pre-recorded data about position updates which are read from a file.

6.3 Future Work

Our future work concentrates on the two areas of cloning of federates and integrating a geographical information system into HLA-federations.

6.3.1 Cloning

We are currently researching mechanisms to test some of the concepts discussed in section 5 using our traffic management federation. In order to do so, a way to save and restore the state of our simulation has to be implemented first. In order to have a solution which is generally valid, the simulation tool SLX needed to provide such kind of an option. Alternately, we could develop a model specific solution to save the state of this special simulation model.

The general potential of cloning in our prototype lies in the fact that the simulation model shows the *current* state of the *real* system (because of the existence of an on-line federate) which then can be cloned.

The clone can then be used to evaluate different alternatives for decisions, e.g. sending some streetcars and busses on a detour to by-pass a traffic jam, etc.

6.3.2 Development of a GIS Federate

Another direction of future work will be the integration of a geographical information system (GIS) as a federate into HLA federations. The GIS should be used for dynamically retrieving the street and route network, which the simulation components are based on, at runtime of a federation execution. This provides for a great flexibility, since no networks had to be hard-wired into the simulation model. A problem that may arise in this approach is to find a suitable GIS which has

a) access/interoperability to all the information the simulation model needs (including route networks of public transportation, all stops, terminal stations, distances, etc.)
b) good extensibility mechanisms to facilitate the development of an HLA interface.

7 CONCLUSIONS

Our work shows that the simulation community could make very good use of approaches for composing simulations from modular, re-usable components. The U.S. DoD's High Level Architecture can provide a suitable infrastructure for constructing simulation federations in this manner.

HLA also provides new possibilities for the variable use of on-line data in simulation models. A certain federate does not need to know whether the data it receives is actually obtained on-line from a generating process (using an on-line federate) or if the data is produced from a playback federate. Using the composition approach, one can simply plug-in a different federate instead of the on-line federate, which has the same Simulation Object Model (SOM) and then produces simulated "on-line" data.

Our future work will consider the possibility of using geographical information systems (GIS) as information provider for HLA federates. We intend to dynamically retrieve the street network from the GIS at *runtime* of the simulation.

We are also working on a possibility to clone federates which are based on commercial simulation tools (e.g. SLX). In order to do so, the simulation tool would have to provide the option to save and restore the state of the simulation. Additional mechanisms have to be implemented to co-ordinate the cloning process with the entire federation execution.

REFERENCES

Defense Modeling and Simulation Office (DMSO). 1997. The High Level Architecture Homepage. URL http://hla.dmso.mil/.

Dorwarth, H., P. Lorenz, K. C. Ritter, and T. J. Schriber. 1997. Towards a Simulation and Animation Environment for the Web. In *Proceedings of the 1997 Winter Simulation Conference*, eds. Andradóttir, S., K. Healy, D. Withers, and B. Nelson, pp. 1338-1344, SCS, Atlanta.

Henriksen, J.O. 1997. An Introduction to SLX™. In *Proceedings of the 1997 Winter Simulation Conference*, eds. Andradóttir, S., K. Healy, D. Withers, and B. Nelson, pp. 559-566, SCS, Atlanta.

Hybinette, M., R. Fujimoto. 1997. Cloning: A Novel Method for Interactive Parallel Simulation. In *Proceedings of the 1997 Winter Simulation Conference*, eds. Andradottir, S., K. Healy, D. Withers, and B. Nelson, pp. 444-451, SCS, Atlanta.

Klein, U., T. Schulze, S. Straßburger, and H.-P. Menzler. 1998. Traffic Simulation Based on the High Level Architecture. In *Proceedings of the 1998 Winter Simulation Conference*, eds. Medeiros, D., E. Watson, J. Carson, and M. Manivannan, pp. 1095-1103. SCS, Washington.

Klein, U., T. Schulze, S. Straßburger, and H.-P. Menzler. 1998a. Distributed Traffic Simulation based on the High Level Architecture. In *Proceedings of the Simulation Interoperability Workshop* Fall 1998, Orlando.

Lorenz, P. and K. C. Ritter. 1997. Skopeo: Platform-Independent System Animation for the W3. In Deussen, O. and P. Lorenz (Eds.), *Proceedings of the Simulation und Animation Conference Magdeburg*, March 6-7, 1997. SCS European Publishing House San Diego/Erlangen/Ghent/Budapest 1997, pp. 12-23.

Straßburger, S., T. Schulze, U. Klein, J.O. Henriksen. 1998. Internet-based Simulation using off-the-shelf Simulation Tools and HLA. In *Proceedings of the 1998 Winter Simulation Conference*, eds. Medeiros, D., E. Watson, J. Carson, and M. Manivannan, pp. 1669-1676. SCS, Washington.

Merriam-Webster Dictionary. Available on-line at http://www.m-w.com/.

AUTHOR BIOGRAPHIES

THOMAS SCHULZE is an Associate Professor in the Department of Computer Science at the Otto-von-Guericke-University in Magdeburg. His research interests include modeling methodology, public systems modeling, traffic simulation, and distributed simulation with HLA. He is an active member in the ASIM, the German organization of simulation.

STEFFEN STRASSBURGER holds a Master's degree in Computer Science from the Otto-von-Guericke University, Magdeburg. He is currently working towards his PhD degree at the Institute for Simulation and Graphics at the same university. His experience with inter-networking and simulation includes a one-year-stay at the University of Wisconsin, Stevens Point. His main research interests lie in distributed simulation and the High Level Architecture.

ULRICH KLEIN is a project manager at the Fraunhofer Institute for Factory Operation and Automation IFF in Magdeburg, Germany. He holds a Master's degree in Industrial Engineering from the University of Karlsruhe and has been involved in Emergency Management since 1992. He has a two-years experience as project manager for Command, Control and Communication Systems for Public Safety and Security in Europe. His research topics include Emergency Management, Urban Infrastructure Management and Logistics, Geographic Information Systems and the High Level Architecture.