ON THE HLA-BASED COUPLING OF SIMULATION TOOLS

Steffen Straßburger
Institute for Simulation and Graphics
Department of Computer Science
Otto-von-Guericke University
Universitätsplatz 2
D-39106 Magdeburg
email: strassbu@isg.cs.uni-magdeburg.de

ABSTRACT

The recent development of the High Level Architecture for Modeling and Simulation (HLA) has stimulated interest in the use of distributed, interoperable simulation models. This paper deals with the application of HLA-based coupling of simulation tools. Concepts for the extension of simulation tools with HLA capabilities are outlined and their realization is discussed. Some prototypic federations of commercial simulations tools are discussed.

KEYWORDS

Discrete Event Simulation, Distributed Simulation, High Level Architecture (HLA), AutoMod, Pro Model, Simplex 3, SLX.

INTRODUCTION

The recent advent of the High Level Architecture for Modeling and Simulation (HLA) has greatly increased interest in the use of distributed, interoperable simulation model components. To date, most models using HLA have been developed in conventional high-level languages (primarily C++). This paper discusses concepts by which HLA can be used to interconnect models developed using commercially available, off-the-shelf simulation software. General approaches for adapting such software for use with HLA are presented. Two generalized solutions for the simulators SLX and SIMPLEX 3 are discussed, as well as problems with developing solutions for other tools like Pro Model and Automod.

The High Level Architecture is defined by:

1. rules which govern the behavior of the overall distributed simulation (*Federation*) and its members (*Federates*).
2. an interface specification, which defines the interface between each federate and the Runtime Infrastructure (*RTI*), a supporting software component that is responsible for providing communication and coordination services to the federates.
3. an *Object Model Template (OMT)* which defines how federations and federates have to be documented.

Although HLA seems to have some similarities with CORBA, HLA offers more than CORBA can do for simulations tools. HLA has integrated mechanisms for the synchronization of simulation tools regarding time and data exchange as well as intelligent data distribution mechanisms.

HLA INTERFACE OF SIMULATION TOOLS

HLA defines a two-part interface which federates are required to use for communicating with the Runtime Infrastructure (RTI) (DMSO 1998). This interface is based on the ambassador paradigm. A federate communicates with the RTI using its RTI ambassador. Conversely, the RTI communicates with a federate via the federate's ambassador. From the federate programmer's point of view these ambassadors are objects and the communication between the participants is performed by calling methods of these objects.

HLA Integration Concepts

In order to enable simulation tools to access the HLA Application Programming Interface (API) it is inevitable to perform low level programming in a typical programming language like C++, Java, or ADA. It is highly desirable for simulation developers to only have to perform this task once for a variety of models. Therefore simulation model independent solutions are needed. This kind of approach can be classified as a "tool enhancement approach". Ideally this task would be performed by the tool developers themselves. Since only very few simulation tool developers actually see the necessity to build an HLA interface into their tools, model developers quite often are confronted with

building it on their own.

Most often the restrictions of time and money dictate rather "quick and dirty" solutions, which only provide a limited set of HLA functionality to the simulation tool. Often some model specifics are also hard wired into the HLA extension of the tool. This type of solution can be classified as a "model or application enhancement approach". Since the capabilities offered by the tool enhancement approach supercede those of the application enhancement approach, the tool enhancement approach is recommended.

The next two sections give two distinct perspectives on how HLA interfaces for simulation tools could look like: the first one is the programmer's perspective (the person developing the HLA interface and doing the low-level programming), the second one is the user's perspective (the person developing HLA based models).

The Programmer's Point of View

For the actual task of accessing the HLA API by doing low level programming the following general strategies have been introduced in (Straßburger et al. 1998).

1. Re-Implementation of the tool with HLA-extensions

If the source code of a tool is available this is most likely the straightforward solution. Skopeo (Lorenz and Ritter 1997), a web-based animation tool developed at the University of Magdeburg, and SIMPLEX 3, a simulation tool developed at the University of Passau, are examples for tools for which HLA-compliance has been accomplished in this manner.

2. Extension of intermediate code

Some simulation tools translate model descriptions written in a tool-dependent modeling language into another programming language (e.g. C++). This intermediate code is then compiled to an executable file. It is possible to modify this code to realize the HLA extensions. Since this code is compiler *generated*, an automated solution is desirable. (Klein et al. 1998)

3. Usage of an external programming interface

This solution is well suited for tools that offer an open and extensible architecture. The tool should offer a library interface (in Windows: a DLL interface) with the ability to call arbitrary functions or methods in these libraries. Some tools provide a somewhat similar interface by providing the possibility to include C or FORTRAN subroutines.

4. Coupling via a gateway program

The last solution for tools which can not be connected to the RTI by any of the prior methods is the development of a gateway program. The gateway program could communicate with the simulation tool via appropriate means (e.g. files, pipes, ports, network) depending on the capabilities of the simulation tool. Ongoing activities of some German Fraunhofer Institutes working together in the DZ-SIMPROLOG initiative use an approach similar to this gateway approach. The intention is to build a so-called "RTI-enabler" which will be universal for a variety of simulation tools (e.g. Simple, MOSYS), but limited to the application domain of logistical simulations. The simulation tools to be connected using this tool have to incorporate an adapter support into the tool (Mertins et al. 1998, Mertins et al. 1998a).

The four strategies discussed above address the question of where and how to perform the low level access to the HLA interface from the programmer's point of view. Nothing is said yet about in which form the user (i.e. the model developer) will be confronted with the HLA API and how the access to the HLA interface could look like.

The User's Point of View

There are two general ways of providing HLA functionality to a model developer. The first alternative is to provide a simulation language or simulation tool specific mapping between the HLA API (e.g. the C++ or JAVA API) and a tool specific API. The second alternative is to totally "hide" the HLA functionality from the model developer. Both alternatives are being elaborated on next.

Mapping from HLA API into a tool specific API

This solution is well suited for cases, where a library interface of the simulation tool is used to provide HLA functionality. In this case the library provides the users with functions that they have to call from within their models. The functions that the model can call should correspond with the RTI-ambassador methods defined in the HLA Interface specification. With that a mapping solution for the first part (the information flow from the simulation tool to the RTI) of the two-part HLA interface has been found.

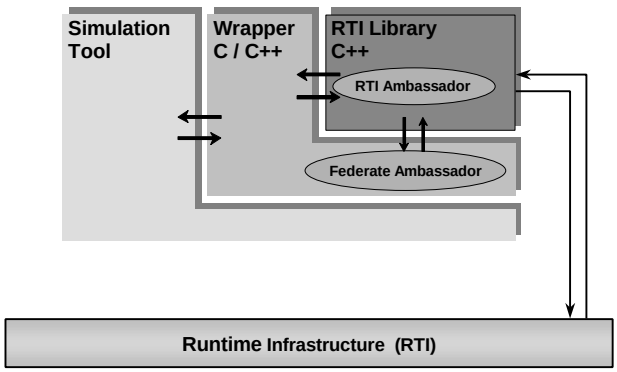

Figure 1: Using Wrapper Libraries as HLA-Interface for Simulation Tools

To build a solution for the second part of the HLA API (the information flow from the RTI back into the simulation tool) is somewhat more difficult, since most simulation tools do not provide the possibility to define functions that can be called from the outside ("callback"-functions). Therefore the wrapper library has the additional task of implementing the federate ambassador and receiving all incoming information (Figure 1). The federate ambassador should be implemented in a model independent manner, so that incoming data is stored in queue and buffer areas or is delivered directly to the simulation model.

It is then the task of the wrapper library to reflect the information which it has received into the simulation tool.

For this task several alternatives can be considered:

- Shared memory areas: If the simulation tool has the capability to pass pointers about its internal data structures the wrapper library can directly access the simulated objects.
- Using dynamic data exchange techniques: Some tools offer "copy and paste"-like capabilities (e.g. object linking and embedding (OLE) under MS Windows) that can be used for transferring data into the model. Care has to be taken that the tool offers this capability not only during the preparation phase of the model, but also at runtime, as some tools (e.g. Pro Model) do not.
- Pulling mechanisms: A last alternative that should only be applied if no other way to transfer the data from the wrapper library to the tool can be found is to apply pull mechanisms. In this approach the tool constantly issues queries to the library if new data has been received.

In addition to the technical approach for transferring the information into the simulation tool it is also necessary to adopt the simulation model to properly react on the change of state variables from outside. This also constitutes the slight disadvantage of the mapping approach: The simulation model has to be made aware of the fact that it is not a stand-alone model. Given a powerful modeling language like SLX with its *control variable* mechanisms this can be relatively straight-forward, though.

The main advantage of the mapping approach is that the model developer can influence every little detail with respect to the HLA functionality. He / she can influence the synchronization mechanisms, data exchange, lookahead issues, etc., but does not have to care about the RTI-tick mechanisms.

The "Below-the-Surface" Approach

For users who do not wish to get very deep into HLA programming and would rather like to have an automated solution, an approach where all HLA functionality is hidden from the user is desirable.

In addition to the normal simulation model the user ideally would only have to establish an Simulation Object Model (SOM) as required by HLA and the simulation tool would perform all HLA-housekeeping tasks below the surface.

Some of the tasks that needed to be done by the simulation tool below the surface are stated in the following:

- Synchronization with other federates: A zero lookahead approach would ensure universal validity, although it is (due to performance issues) generally desirable to operate with larger lookahead values. The tool would automatically synchronize with other federates via the standard HLA mechanisms. Special care has to be taken that only relevant events are synchronized if combined or pure continuous models shall be used.
- Automatic publishing and subscribing of HLA object and interaction classes.
- Automatic generation of updates / interactions if model variables that are reflected in HLA objects or interactions change.
- Automatic ghosting of objects and mapping onto appropriate model variables, special care has to be taken on multiple object instances of the same class.
- Conversion between tool specific data types and data types as defined in the Federation Object Model (FOM). This also requires conversion of different endian-types of different hardware platforms.

Such a below-the-surface approach can theoretically be applied, if either the source code of the simulation tool is available or an automated solution for extending the simulation model can be developed. The prototype of the HLA interface for SIMPLEX 3 introduced in the next section is an example were this approach was taken.

EXPERIENCES & PROTOTYPES

This section introduces some prototypical HLA extensions for various tools and discusses how the theoretical concepts discussed in the previous sections where applied in the implementation. Table 1 gives an overview about the various prototypes and which approach for an HLA Interface has been chosen.

Tool	Programmer's Point of View				User's Point of View		Model Independent?
	Re-Imple-mentation	External Program-ming Interface	Gateway Program	Intermediate Code	Mapping	Hidden	
SLX		X			X		X
Pro Model		X			X		
Automod		X			X		
Simplex 3	X					X	X
Modsim III	X				X		X

Table 1: Categorization of the HLA interfaces for the simulation tools discussed in this section

SLX

SLX is a new discrete event simulation tool for the Windows 95/98/NT operating systems (Henriksen 1996). SLX has a library interface which allows to call functions in any standard Windows DLL. Therefore SLX qualifies for the general possibility of applying a wrapper library for creating a HLA interface (Figure 2).

In the solution for SLX (Straßburger and Klein 1998) the HLA API is mapped into a tool specific API. The model developers have to enhance their models by certain API calls. Since SLX offers the possibility to pass pointers when calling libraries a shared memory approach is used for transferring data back into the simulation tool: Incoming updates or interactions are automatically stored in the associated SLX objects.

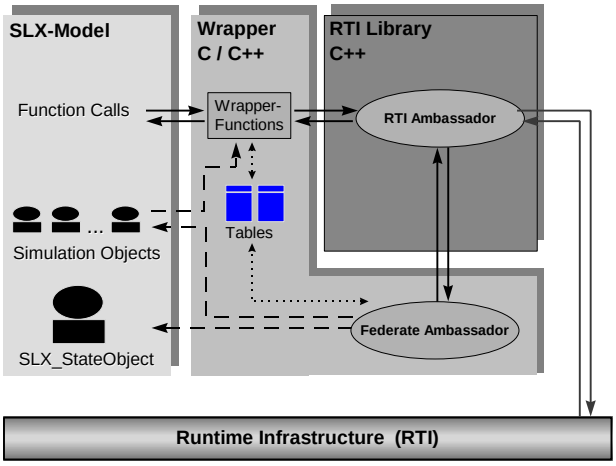

Figure 2: The HLA Interface for SLX

The main advantage of the solution for SLX is that the user can use the entire flexibility of HLA (by providing the possibility of calling almost every HLA API function), but at a much more comfortable level as if dealing with the raw HLA API.

Several existing stand-alone SLX models have successfully been extended with HLA functionality by using the HLA interface for SLX with rather few modifications (Klein et al. 1998a).

The HLA Interface for SLX in its current versions for RTI 1.0.3 and RTI 1.3 and also future versions will be available as an add-on product for SLX.

Pro Model 4.0

In a cooperative effort of the Fraunhofer Institute "Produktionstechnik und Automatisierung (IPA)" in Stuttgart and the University of Magdeburg it was tested if the HLA extension for SLX could be transferred into an HLA extension for Pro Model 4.0.

Pro Model 4.0 also offers a library interface for extending the tool by using DLL's. The functionality by this library interface is rather limited:

- It is not possible to pass pointers to Pro Model entities
- Only double-values can be returned by a DLL function
- The format of the parameters passed to the DLL has to be known at compile time

Although these are relatively strict limitations it was generally feasible to apply the SLX approach to Pro Model. Limitations apply regarding the synchronization and the data transfer back into the model.

These limitation result from the fact that Pro Model

a) does not have a built-in function for determining the time stamp on the next event. Therefore only time stepped synchronization approaches can be applied.

b) the transfer of received data back into Pro Model is not possible using shared memory areas. In a first approach Pro Model's OLE interface was tested for its suitability. Soon it was learned that this interface can only be used in the preparation phase of the model. Therefore in a first prototype pulling mechanisms were tested and applied for querying for received data.

Automod

In an independent development at the Institute for Machine Tools and Factory Management of the Technical University of Berlin a model dependent HLA extension for the simulation tool Automod was created (Seliger et al. 1999). Like the two above mentioned solutions for SLX and Pro Model this solution is also based on a wrapper library which provides a tool specific mapping of the HLA interface to the model. Since in this solution also some model specifics are hard-coded into the C++ sources the solution is also model specific. In the development of the DLL similar problems as experienced in the Pro Model interface were encountered (no built in functions for determination of the time stamp of the next event, difficulties transferring data back into the model).

SIMPLEX 3

To verify whether the theoretical approach of hiding all HLA functionality from the model developer is practicable at all, a cooperative effort between the Universities of Passau, Dresden, and Magdeburg to implement an HLA interface for SIMPLEX 3 was started (Lantzsch et al. 1999). Simplex 3, which has been developed at the University of Passau, is a simulation system which allows the creation of discreet, continuous, and combined models (Schmidt 1995).

In SIMPLEX 3 several model components (called Basic Components) can be connected by using a so-called "High Level Component" which defines the connections between the Basic Components. Basic Components can run independently from each other. It is not relevant for them whether they are connected to other components or not. This fact results from the so-called Glass-Box principle of SIMPLEX, which states that each component can read variables from other components.

The only change in using Basic Components in different combinations relates to the functionality of the entire model. The same Basic Components connected by different High Level Components will usually form a different model.

This property of SIMPLEX was used for the HLA extensions. A new item, called "HLA component" has been introduced which connects variables of a SIMPLEX model with HLA objects and interactions. With this solution it is not necessary to change or extend the actual modeling language of SIMPLEX (Simplex Model Description Language, MDL). The only extension from the model developer's point of view lies in the fact that he / she has to specify an HLA component for the mapping between SIMPLEX data structures and HLA data structures.

All synchronization and data exchange is handled internally by the HLA extensions to the SIMPLEX runtime system.

This has the main advantage of being very comfortable for the user, but also some disadvantages regarding the impossibility to use all flexibility that HLA could offer. For a general validity of the below-the-surface approach a conservative synchronization with zero lookahead has to be assumed. Depending on the model higher lookahead values would be possible and useful. Further investigation is scheduled in order to add more flexibility.

MODSIM III

MODSIM III recently has been equipped with an HLA support by its developing company. MODSIM's object library support for the HLA is integrated with other parts of the MODSIM III system such as SimGraphics. The raw HLA API is packaged into MODSIM objects that support all HLA areas. MODSIM's HLA support provides automatic handling of message pumping, exceptions, and callbacks.

MODSIM III also provides universal datavalue representation so that data can be transported "through the wire" between different computers and operating environments.

With this the solution for MODSIM provides a somewhat similar approach as the solution for SLX: A tool specific mapping of the HLA API into a tool specific API has been performed. The main difference lies in the implementation: While it was possible in the MODSIM approach to actually access and modify the source code of the tool this has not been done for SLX. Using the advanced statement concept of SLX, it is possible to make the SLX-HLA-Interface behave as if it was built-in into SLX, though.

REFERENCE FEDERATIONS

For testing the HLA-interfaces of the simulation tools described above, several reference federations have been implemented. Some of them are described in the following sections.

The Distributed Driving Federation

In this federation a microscopic urban traffic model written in SLX (Klein et al. 1998a) is connected with a real-time driving simulator for SGI-workstations. The federation successfully demonstrated the cooperation of two HLA-federates with different time-advancement mechanisms (logical time in SLX vs. real-time in the driving simulator) and the cooperation between different platforms (Intel PC vs. SGI). The latter was achieved by

using the capability of the SLX-HLA-Interface to automatically convert different endian types.

This federation was the first large-scale test for the SLX-HLA-Interface.

The Simplex 3 – SLX Federation

This federation was developed in a co-operation between the HLA team in Magdeburg and the University in Passau and is the first federation featuring a federate developed with the simulation system Simplex 3. The federate developed with Simplex 3 simulates a barrel filling station. The filling of barrels is modeled as a continuous process described by a differential equation. The second federate is a SLX model which simulates the logistical processes in a transport agency. Orders for barrels are generated from different locations throughout Germany and passed to the barrel filling station. The SLX federate also performs an online visualization of the federation with Proof Animation™ for Windows (Figure 3).

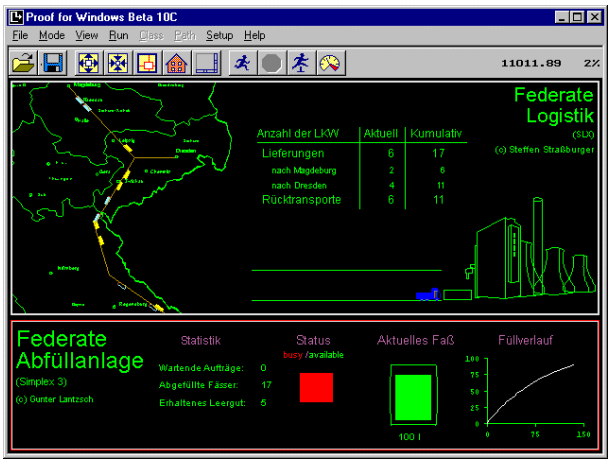

Figure 3: Screenshot of the Simplex-SLX Federation

The major goal of this federation is to demonstrate the cooperation of a discrete simulation model written in SLX and a combined model developed with Simplex 3.

The Streetcar Federation

In this federation online data from the central computer of Magdeburg's local traffic company is used in an analytical simulation model. Different setups exist for this federation. The typical setup consists of two federates: The first federate performs a schedule based simulation of the streetcar system in Magdeburg. The second federate connects online to the central computer of the streetcar system and sends position updates into the federation. The first federate can react to these updates and adjust the current state of the simulation accordingly. Both federates have been implemented using SLX and the SLX-HLA-Interface. The first federate also uses the new Proof for Windows to perform online animation of the system. A third federate, which is based on the Web-based animation tool Skopeo, can be used in addition to the online animation with Proof to provide visualization anywhere in the WWW. Figure 4 shows a screenshot of the system obtained with Proof Animation.

The federation was implemented to demonstrate an increased flexibility by using interoperable components (federates). The streetcar federate can be combined either with the online-federate or with a surrogate, which simulates online-date from a playback file to analyze different scenarios. No modifications have to be performed at the streetcar federate for this purpose.

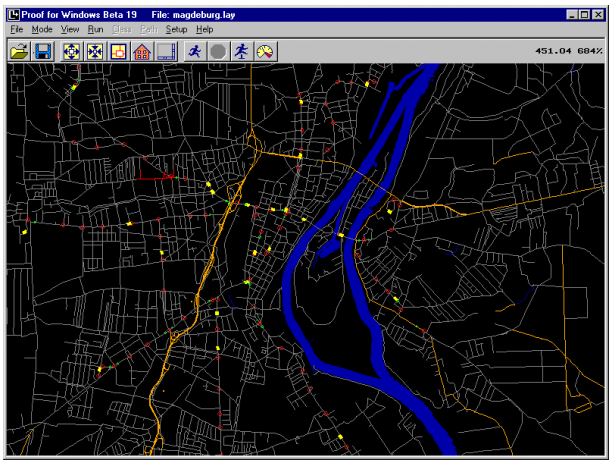

Figure 4: Screenshot of the Streetcar Federation

Further Prototypes

- The SLX - Pro Model Federation

  In this federation the interoperability between SLX models and a simulation model written in Pro Model 4.0 has been demonstrated. The federation models a production chain where different stages of a product's life cycle are modeled either by SLX or by Pro Model.

- The SLX - Automod Federation

  This federation demonstrates a cooperation between SLX and Automod in the area of production networks (Seliger et al. 1999) and is thus similar to the one described in the previous section.

CONCLUSION AND OUTLOOK

HLA offers a new simulation interoperability standard that will influence the non-military simulation market in a similar way it currently influences the area of military training simulators.

While in the military sector most applications are developed using C++ and thus automatically qualify for the usage of HLA the situation in the civil sector is different. The use of commercial simulation tools is very common. Although there is a need for simulator interoperability in the civil sector, too, the price for developing a tool specific HLA interface is relatively high.

Therefore it will take more time and also more insight on the side of the tool developers until HLA compatible simulation tools will become (commercially) available. SLX and MODSIM III are the first widely used commercial simulation tools which already provide HLA interfaces to date. More tools are hopefully to come soon.

Our research shows that HLA interfaces do not necessarily have to be built by the tool developers themselves. Depending on the capability of the simulation tool third party developers can implement HLA add-on packages for simulation tools.

The ultimate goal for simulator interoperability should be a built-in and transparent HLA interface for every simulation tool as demonstrated in the Simplex 3 approach. In order to achieve this goal modifications of the runtime system of a simulation system seem to be inevitable.

With that kind of HLA interface interoperability between different model components implemented in different simulation languages in a “plug-and-play” manner could come true.

The latest information about ongoing HLA-integration efforts can be found at http://isgsim.cs.uni-magdeburg.de/hla.

REFERENCES

Department of Defense (US). 1997. High Level Architecture Interface Specification, Version 1.3. Available online at the HLA Homepage: URL http://hla.dmso.mil/.

Henriksen, J.O. 1996. An Introduction to SLX. In *Proceedings of the 1996 Winter Simulation Conference*, eds. J.M. Charnes, D.M. Morrice, D.T. Brunner, J.J. Swain, pp. 468-475. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Klein, U., S. Straßburger, J. Beikirch. 1998. Distributed Simulation with JavaGPSS based on the High Level Architecture. *International Conference on Web-based Modeling and Simulation*, Jan. 11-14, 1998, San Diego.

Klein, U., Th. Schulze, S. Straßburger and H.-P. Menzler. 1998a. Traffic Simulation Based on the High Level Architecture. In *Proceedings of the 1998 Winter Simulation Conference*, eds. Medeiros, D.J. and Ed Watson, SCS, Washington.

Lantzsch, G., S. Straßburger, C. Urban. HLA-basierte Kopplung der Simulationssysteme Simplex III und SLX. In Deussen, O., V. Hinz, P. Lorenz (Ed.), *Tagung Simulation und Visualisierung*. March 4.-5. 1999, Magdeburg.

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

Mertins, K., M. Rabe, P. Rieger. 1998. Taking Advantage of Process Oriented Reference Models for Setting Up Federations for Distributed Simulation in HLA Environments. In Zobel, R. and D. Moeller (Ed.), *Proceedings of the 12th European Simulation Multiconference*. June 16-19, 1998. pp. 259-263.

Mertins, K., M. Rabe, P. Rieger. 1998a. Einsatz von Simulations-Referenzmodellen für eine effiziente Erstellung von Simulationsverbunden auf Basis von HLA. In Engeli, M. and V. Hrdliczka (Ed.), *Proceedings of the 12th Simulation Symposium ASIM 98*. Sept. 15-18, 1998. pp. 299-305.

Schmidt, B. 1995. *Simplex II - Benutzerhandbuch*. SCS Publications, San Diego 1995.

Seliger G., D. Krützfeldt, P. Lorenz, S. Straßburger. 1999. On the HLA- and Internet-based Coupling of Commercial Simulation Tools for Production Networks. *International Conference on Web-based Modeling and Simulation*. Jan. 17-20, 1999, San Francisco.

Straßburger, S., Klein, U. 1998. Integration des Simulators SLX in die High Level Architecture. In Lorenz, P., Preim, B. (eds.), *Tagung Simulation und Visualisierung 1998 Magdeburg.* SCS Europe Publishing House, Delft, Erlangen, Ghent, San Diego. pp. 32-40.

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.J. and E. Watson, Washington D.C.

AUTHOR BIOGRAPHY

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