TECHNICAL ARTICLE

SIMULATION 73:5, 296-303

ISSN 0037-5497/99
Printed in the United States of America

Migration of HLA into Civil Domains: Solutions and Prototypes for Transportation Applications

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

The United States Department of Defense's (DoD) High Level Architecture for Modeling and Simulation (HLA) is a mandatory standard for military simulations. The situation in the civil simulation community is different: simulator interoperability is desirable and even required, but there is no driving force to mandate the use of a certain standard. This article addresses the problems that a simulator interoperability standard in the civil world faces and discusses how HLA can possibly become the standard that is needed. Several solutions for connecting civil simulation tools using HLA are introduced and some prototypical applications focusing on the area of transportation are demonstrated.

Keywords: HLA, civil application, discrete event simulation, interoperability, transportation

1. Introduction

The DoD's High Level Architecture for Modeling and Simulation (HLA) can certainly be regarded as the state of the art in distributed simulation. This holds true for the military simulation community *per se*, since this is the origin of HLA.

A comparison of the military with the civil simulation world shows that, on the one hand, it is mostly the terminology that differs. Although the approaches and general methods which are used in both simulation communities are very similar, the terminology is entirely different [1]. Terms such as *dead-reckoning* and *constructive/virtual/live* simulation are relatively seldom used in the civil world, although equivalent techniques exist and are commonly applied.

On the other hand, there is a major difference in how simulations are developed. While in the military community, simulations are often developed in languages such as C++, ADA, or even Java, in the civil simulation community, the use of commercial simulation tools (e.g., Arena, GPSS/H, MODSIM, Pro Model, Simple++, SLX, etc.) is commonplace. These simulation tools satisfy the need to develop models rapidly and cost-effectively. It is rather unusual to develop simulations in programming languages such as C++. There are several well-known reasons for this:

1. C++ is difficult to learn.
2. It is too easy to make mistakes which have disastrous consequences, but are difficult to find, e.g., faulty use of pointers.
3. It lacks an inherent mechanism for describing parallelism.
4. Its debugging tools are simulation-unaware, i.e., they operate at a level far below that which would be convenient for most simulationists.

These different approaches for developing simulations also apply to HLA. While it is rather straightforward to use HLA in a simulation developed in C++, alternative means have to be used when developing a simulation in a simulation language. Although there is a need for interoperability, the access to the HLA Application Programming Interface (API) from the tools used in the civil world is usually not directly possible.

Therefore, this article discusses requirements and solutions for applying HLA in the civil world (Section 2) and introduces several sample federations focusing on transportation applications (Section 3).

2. HLA Interfaces for Simulation Tools

In order to enable simulation tools to access the HLA API and its interoperability software, called Runtime Infrastructure (RTI), performing low-level programming in a typical programming language such as C++, Java, or ADA is inevitable. 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.

HLA interfaces for simulation tools can be classified from two distinct perspectives: 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) [2].

2.1 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 [3].

- **Re-Implementation of the Tool**

If the source code of a tool is available, this is most likely a straightforward solution. Simplex 3, a simulation tool developed at the University of Passau, Germany, has been enhanced for HLA functionality in this manner.

- **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 intermediate code to realize the HLA extensions. Since this code is *compiler-generated*, an automated solution is desirable.

- **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. The HLA interface for the simulation tool SLX [4] has been implemented using this approach.

- **Coupling via a Gateway Program**

The last solution for tools which cannot access the HLA API 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. An example for a gateway solution is the DIS-HLA gateway [5, 6].

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. In the next section, we will discuss in which form the user (i.e., the model developer) can be confronted with the HLA API and what the access to the HLA interface could look like.

2.2 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. This approach is called the *explicit approach*, since the simulation model has to explicitly use HLA functionality (i.e., RTI function calls).

The second alternative is to totally "hide" the HLA functionality from the model developer. This approach is called the *implicit approach*, since the simulation system handles all HLA communication internally. The simulation developer is not concerned with HLA functionality, because all RTI calls are performed "implicitly" when they are appropriate (e.g., when an object attribute changes).

- **The Explicit Approach**

This solution is well suited for cases in which 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 to the methods defined in the HLA Interface Specification.

Since most simulation tools do not provide the possibility to define callback functions (i.e., functions which can be called from other software modules), alternative means have to be used for transferring data back to the tool. This is necessary because the HLA programming paradigm requires the implementation of callback functions which receive incoming data via callbacks from the RTI.

The SLX-HLA interface uses this approach to provide HLA functionality to the model developer.

- **The Implicit Approach**

In this approach all HLA functionality is hidden from the model developer.

A solution which uses this approach needs to perform the following tasks automatically:

- Synchronization with other federates: a conservative approach operating with zero lookahead would ensure universal validity, although it is (due to performance issues) generally desirable to operate with larger lookahead values [7]. The tool would automatically synchronize with other federates via the standard HLA mechanisms.
- Automatic generation of updates/interactions (if model variables change) that are reflected in HLA objects or interactions.
- Automatic ghosting of objects and mapping onto appropriate model variables. Ghosting is a method of creating local copies of remote objects for receiving and storing updates about them.
- 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 usually be applied only if the source code of the simulation tool is available. The prototype of the HLA interface for Simplex 3 is an example where this approach was taken.

2.3 Enhanced Simulation Tools

The main focus of the research presented in this article has been to test and validate the different approaches presented in the previous section with different simulation systems and to develop some prototypical applications in the civil sector. In two separate projects at the Universities of Magdeburg and Passau, HLA interfaces for the simulation systems SLX and Simplex 3 have been developed.

The HLA interface for SLX uses the Dynamic Link Library (DLL) interface of SLX to access a wrapper library (i.e., the solution uses the explicit approach from the user's point of view). The wrapper library is implemented as a Windows DLL and provides access to the HLA API. SLX models have to explicitly call HLA API functions for synchronization and data exchange.

The HLA interface for Simplex 3 uses the implicit approach outlined above. Therefore, the source code of the runtime system of Simplex 3 has been extended with HLA functionality [8]. The model description of

Table 1. Categorization of the HLA interfaces for simulation tools

Tool	Programmer's Point of View				User's Point of View		Status	Developed By
	Re-Implementation	External Programming Interface	Gateway Program	Intermediate Code	Explicit	Implicit		
SLX		X			X		Released; model-independent	University of Magdeburg
Pro Model		X			X		Experimental; model-dependent	Fraunhofer IPA
Automod		X			X		Experimental; model-dependent	TU Berlin
Simplex 3	X					X	Experimental; model-independent	University of Passau
Modsim III	X*				X		Released; model-independent*	CACI Corp.
Matlab		X			X		Experimental; model-independent	Universities of Rostock and Wismar

* Contact manufacturer for detailed information.

Table 2. Federates participating in the Distributed Driving Federation

Federate	Time Advancement	Hardware/Operating System		Language
Driving Simulator	Real-Time	SGI	IRIX	C++
Traffic Simulator	Event-Oriented	PC	Windows NT	SLX
Observer	Scaled Real-Time	Anywhere (Java Applet)		Java

a Simplex model does not include any HLA specifics. The only way to connect a Simplex model to other federates is by providing a special component which specifies the mapping of Simplex variables to HLA data structures. This has the advantage of an increased reusability of Simplex models.

Other ongoing efforts to develop HLA interfaces are found in both the academic and business communities. Table 1 gives an overview of some of the simulation tools which already have an HLA interface to date and which approach was used for them.

3. Applications of HLA In Transportation Prototypes

Transportation and logistics are classical application areas of simulation and also the first areas to which HLA has been applied in the civil simulation community. This application area covers the detailed modeling of logistical processes in production facilities as well as traffic management systems in cities and larger areas. This also includes the modeling of general transportation systems such as AGVs (automatically guided vehicles), cars, trucks, ships, etc.

Since it is not possible to introduce prototypes for the entire spectrum of transportation applications, the following sections introduce three prototypes which were developed with the direct participation of the authors. The prototypes stem from different application domains within transportation and logistics. Some specific interoperability aspects of HLA were tested with each prototype.

3.1 The Distributed Driving Federation

The distributed driving simulation was developed as a joint effort of the Institute for Simulation and Graphics at the University of Magdeburg and the Competence Center Informatik GmbH [9, 10]. The federation consists of federates that were brought into the project by both partners and independently extended for HLA compatibility. The simulation federates are based on existing simulation applications (legacy applications) of the partners.

Table 2 gives an overview of the participating federates and the environments in which they have been developed.

The federates participating in this federation are:

- **Federate Driving Simulator**: The basis for this federate is a real-time driving simulator for off-road driving within a synthetic environment. In the original simulation there was no street traffic modeled. The simulator is a C++ program for SGI machines and offers a 3D visualization based on the SGI's Performer software.
- **Federate Traffic Simulator**: This federate is based on a discrete event traffic simulation model for urban street traffic which focuses on the psychophysical behavior of street traffic flow. The simulation has been implemented with SLX. This federate models the street traffic and had to be extended to be able to react to the vehicle modeled by the driving simulator.
- **Federate Observer:** This federate is based on the Skopeo animation system which is discussed in Section 3.2.2. The federate was used to obtain an additional visualization of the federation.

The main focus in this federation was to test the interoperability capabilities offered by HLA under certain aspects. The federation successfully demonstrated that HLA is capable of supporting the following aspects:

- Interoperability among federates using different time-advance mechanisms (real time versus event oriented versus scaled real time)

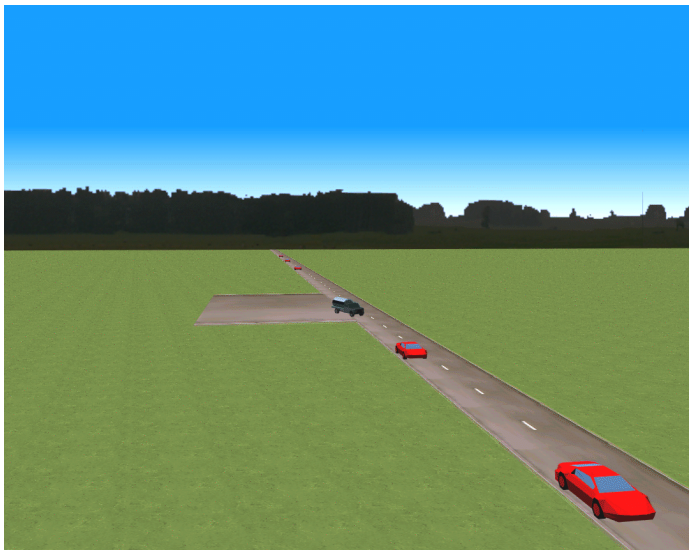

Figure 1. Screen shot of the interacting vehicles within the synthetic environment

- Interoperability among heterogeneous hardware platforms (SGI versus Windows PC versus platform-independent Java applications)
- Interoperability among different implementation languages (C++ versus SLX versus Java)
- Interoperability among different network environments (LAN versus ISDN link versus Internet)

In the federation, the Traffic Simulator provided the vehicles for the synthetic environment of the Driving Simulator (Figure 1).

The vehicles were correctly displayed, including collision detection, accelerated movements, etc. On the other hand, the Traffic Simulator received the position updates from the virtual vehicle modeled by the Driving Simulator and adjusted the movement of its vehicles accordingly (e.g., by reducing speed if the virtual vehicle came into closer range).

3.2 The Streetcar Federation

The Streetcar Federation models a public traffic management scenario in the city of Magdeburg, Germany, and was developed with the support of the Magdeburg Transportation Company (MVB).

The main purpose of the federation was to develop an application which demonstrates how HLA facilitates the transparent integration of real-time-dependent data into simulation models [11].

3.2.1 The Simulation Federate

The "heart" of the federation is a simulation federate which performs a schedule-based simulation of the public transportation system in Magdeburg (i.e., streetcars). This federate has been developed using the simulation system SLX.

The actual simulation model behind it is a classical analytical (or *constructive*) simulation using logical simulation time. The most interesting interoperability aspect is that the model can:

1. Run as a pure model in the absence of input from the outside world, or
2. Accept real-time data as corrections to predicted vehicle positions when coupled to the outside world.

Communication between the simulation and the online data source, as well as with other federates, is based on the HLA interface.

The simulation model uses the SLX-HLA Interface to acquire timestamp-ordered events containing position updates of the simulated objects and to synchronize its local simulation clock with other federates.

The simulation model has to compare the current real-life positions of the streetcars with the "simulated" positions. In case the positions do not match, the running simulation is brought up to date, i.e., the real-time data influences the simulation model. With this approach it is possible to have a simulation model which always represents the state of the real system in very good approximation.

The possible setup alternatives of the federation are shown in Figure 2.

3.2.2 The Animation Federates

Two different tools have been used for producing online animations of the federation.

The first tool is the Web-based animation system Skopeo [12]. Skopeo is a general animation system which provides platform-independent system animation anywhere on the World Wide Web. Skopeo has a prototypical HLA interface which is based on the beta version of the Java RTI 1.0 from DMSO. The Java RTI 1.0 has (in later RTI versions) been replaced by Java bindings to the C++ RTI.

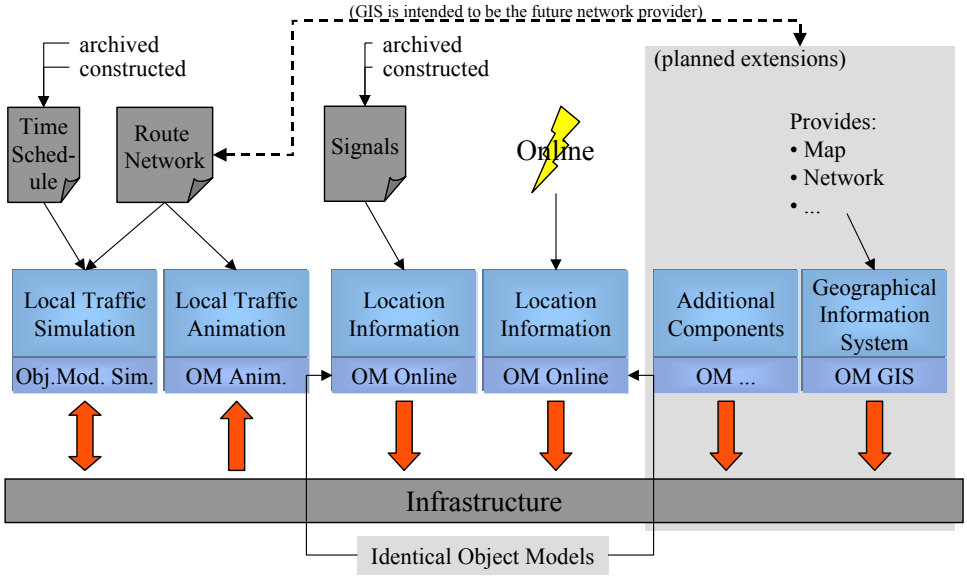

Figure 2. Setup of the traffic management prototype based on the composition principle

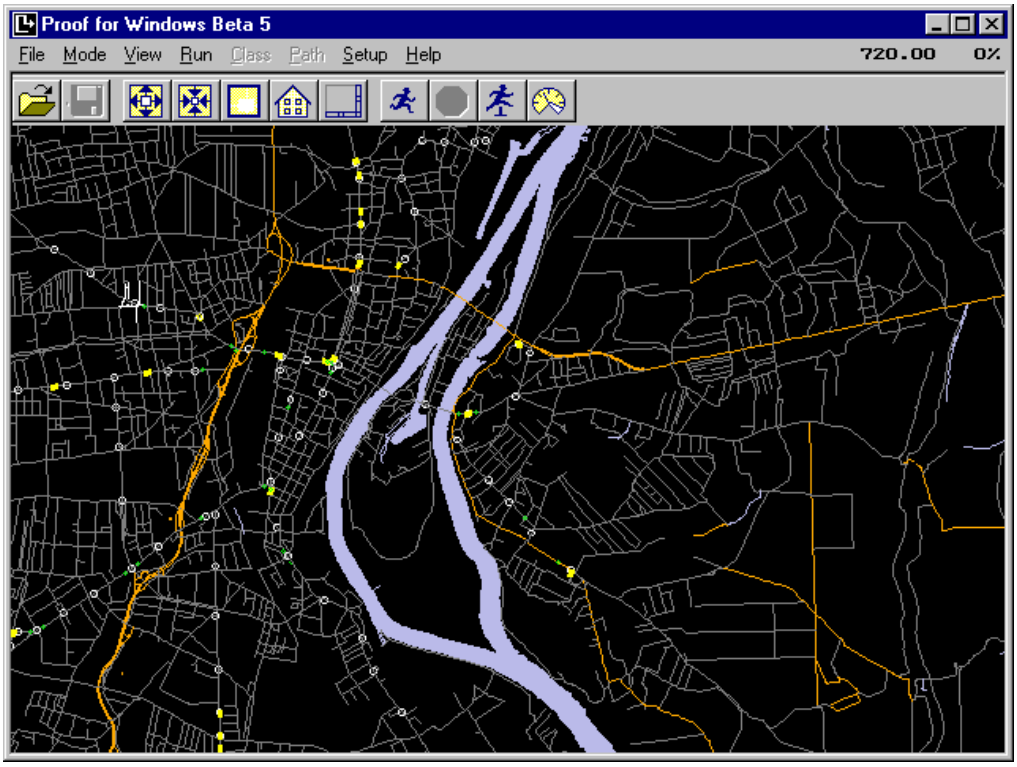

Figure 3. Screen shot of the streetcar federation obtained with Proof Animation for Windows

The second tool which has been used in this federation is Proof Animation™ for Windows [13]. Proof Animation provides online animation on Windows platforms and can be used by a wide variety of programs. In order to produce online animation with Proof, a program to drive Proof is needed. Since a library (Windows DLL) version of Proof and an SLX interface to the Proof DLL were available, it was decided to use SLX to drive Proof Animation (Figure 3).

3.2.3 The Real-Time Data Federates

To provide real-time data for the simulation federate, two different alternatives were developed:

- **Usage of Online Data**

In this alternative an online federate is used to provide position updates from the streetcar vehicles for the other federates.

The online federate starts a receiver process to connect to the command and control computer of the Magdeburg transportation company. From there it receives position updates which are obtained from the "real-life" streetcars using infrared transmitters which each streetcar carries. It should be noted that the position updates are only delivered each time a streetcar passes certain sensors located along the route network.

These position updates are then sent into the federation and received by the simulation federate using interactions. In the time interval between two updates the simulation federate has to interpolate the current position based on the streetcar schedule.

- **Usage of Offline Data**

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

3.2.4 Applications for the Streetcar Prototype

There are two major applications for the Streetcar Federation:

- **Visualization of the Current State of the Real-Life System**

It seems obvious that visualization without simulation could show a streetcar only at the time it passes a sensor located along the route network. This is clearly not a sufficient representation of the system. Therefore the simulation model interpolates the current positions of a streetcar between two sensors and visualizes the streetcars accordingly. By being able to separate the online visualization from the simulation (i.e., by using separate federates), it is possible to obtain a location-independent animation. This has the advantage of being able to view the visualization at different (and multiple) locations inside the intranet of the transportation company. This approach could also be transferred to the fleet management of other logistics companies.

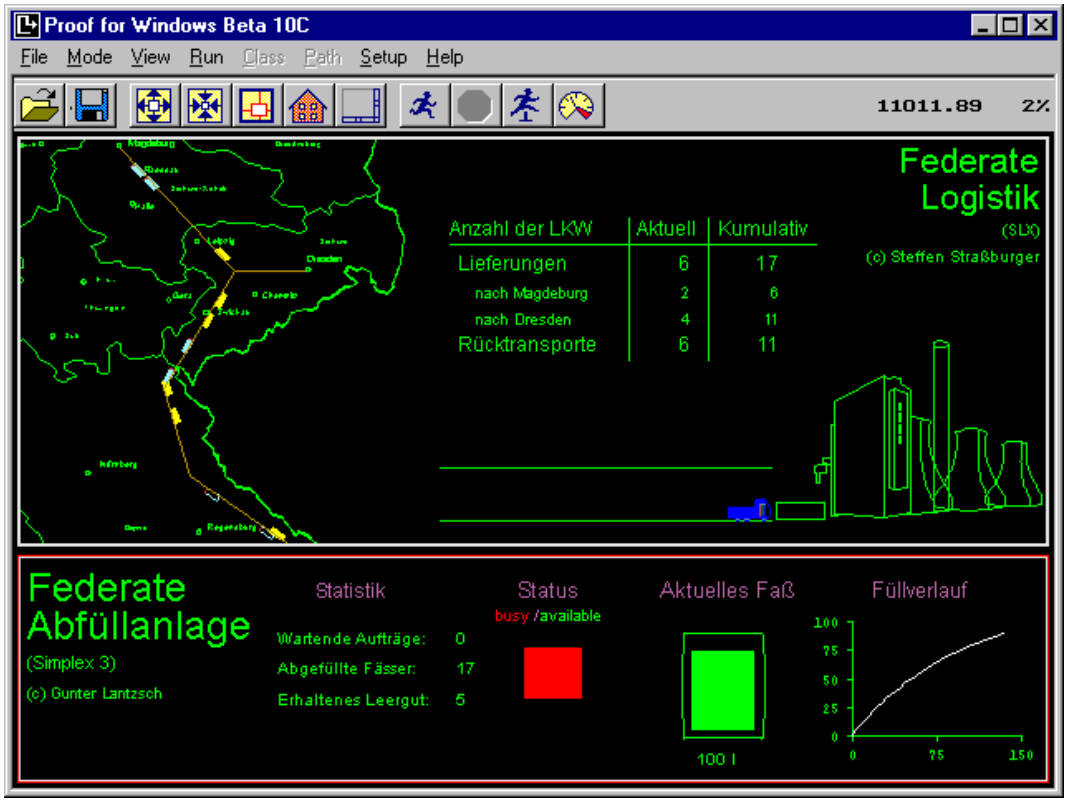

Figure 4. Screenshot of the Simplex-SLX federation

- **Dispatcher Training**

For this application scenario the real-time data is obtained from the offline federate. The offline federate provides certain scenarios in which the dispatcher must react to system conditions. The dispatcher does not need to leave his normal operating environment and does not need to know if he is operating in a real situation or in a constructed scenario.

3.3 The Barrel Filling Federation

This federation was developed in cooperation between the Universities in Magdeburg and Passau and is the first federation featuring a federate modeled with the simulation system Simplex 3. Simplex 3 is a simulation system developed at the University of Passau which can be used for discrete and continuous systems.

The main target in this federation was to combine the advantages of both the simulation systems used in this federation. SLX offers very good capabilities for modeling logistical processes because of its process-oriented world view. Simplex 3 has the ability to develop continuous models in a comfortable fashion and offers several procedures for the numerical integration. It was most desirable to combine these advantages of both systems to form a new (distributed) model consisting of two "sub-models," or federates, in the HLA sense.

The federate developed with Simplex 3 simulates a chemical barrel filling station. The filling of barrels is modeled as a continuous process described by differential equations.

The second federate is an 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 4).

The federation successfully demonstrated the cooperation of a discrete simulation model written in SLX and a combined model developed with Simplex 3.

Further applications exist in any simulation project which consists of models of which sub-components would ideally be modeled with different modeling tools.

4. Summary

Our work shows that the civil 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.

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, one should not underestimate the effort necessary for developing a tool-specific HLA interface.

The simulation systems SLX and Simplex 3 are examples of systems that have been successfully equipped with HLA interfaces. Although they use completely different approaches, federations between these two are now possible. The future will show whether civil industry partners have an actual interest in using the possibilities offered by HLA.

5. References

[1] Page, E.H. and Smith, R. "Introduction to Military Training Simulation: A Guide for Discrete Event Simulationists." In *Proceedings of the 1998 Winter Simulation Conference*, D. Medeiros, E. Watson, J. Carson, and M. Manivannan (eds.), pp 53-60, SCS, Washington, 1998.

[2] Straßburger, S. "On the HLA-based Coupling of Simulation Tools." In *Proceedings of the 1999 European Simulation Multiconference*, H. Szczerbicka (ed.), Vol. 1, pp 45-51, SCS, Warsaw, Poland, 1999.

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

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

[5] Cox, A., Wood, D.D., Petty, M.D. and Juge, P.K.A. "Integrating DIS and SIMNET into HLA with a Gateway." In *Defense Modeling and Simulation Office (DMSO): Proceedings of the 15th DIS Workshop*, Orlando, FL, Sept. 1996.

[6] Wood, D.D., Petty, M.D., Cox, A., Hofer, R. and Harkrider, S. "HLA Gateway Status and Future Plans." In *Simulation Interoperability Standards Organization (SISO): Proceedings of the Simulation Interoperability Workshop Spring 1997*, March 3-7, 1997, Orlando, FL. Paper 97S-SIW-125, SISO, 1997.

[7] Fujimoto, R. "Zero Lookahead and Repeatability in the High Level Architecture." In *Simulation Interoperability Standards Organization (SISO): Proceedings of the Simulation Interoperability Workshop Spring 1997*, March 3-7, Orlando, FL, SISO, 1997.

[8] Lantzsch, G. "HLA Interface for Simplex III." Master's Thesis, Technical University Dresden (German language).

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

[10] Klein, U., Schulze, T., Straßburger, S. and Menzler, H.-P. "Distributed Traffic Simulation Based on the High Level Architecture." In *Proceedings of the Simulation Interoperability Workshop*, Orlando, FL, 1998.

[11] Schulze, T., Straßburger, S., Klein, U. "On-line Data Processing in Simulation Models: New Approaches and Possibilities Through HLA." In *Proceedings of the 1999 Winter Simulation Conference*. In press.

[12] Lorenz, P. and Ritter, K.C. "Skopeo: Platform-Independent System Animation for the W3." In *Proceedings of the Simulation and Animation Conference*, O. Deussen and P. Lorenz (eds.), Magdeburg, March 6-7, 1997. SCS European Publishing House, pp 12-23, 1997.

[13] Henriksen, J.O. "Windows-Based Animation with Proof™." In *Proceedings of the 1998 Winter Simulation Conference*, D. Medeiros, E. Watson, J. Carson, and M. Manivannan (eds.), pp 241-247, SCS, Washington.

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

Steffen Straβburger holds a Master's degree in Computer Science from the Otto-von-Guericke University in Magdeburg, Germany. He is currently working towards his PhD at the Institute for Simulation and Graphics at the same university. His experience with internetworking 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 two years of experience as Project Manager for Command, Control and Communication Systems for Public Safety and Security in Europe. His research topics include emergency management, geographic information systems and distributed simulation-based systems.