On-Line Data Processing in a Civil Transportation Federation

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

Keywords:
HLA, on-line data, discrete event simulation, civil application.

ABSTRACT: *While HLA originates in the military simulation community, civil application areas are slowly evolving. This paper introduces some civil application areas for HLA and presents a civil transportation application. The application demonstrates how the civil simulation community can make use of the benefits that HLA offers: namely re-usability, interoperability, and increased flexibility of simulation components.*
This paper also 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.

1. Introduction / Motivation

Interoperability and reusability are key topics for the military and civil simulation community. In order to evaluate the potential of HLA in the civil application domain, the University of Magdeburg has developed several prototypes which cover different application areas. Several simulation and animation tools have been tested for HLA compliance. HLA interfaces for some of these tools (SLX, Simplex, Proof Animation, Skopeo) have been developed.

The prototypical federation described in this paper demonstrates how HLA addresses the need for flexibility and cost effectiveness of software systems in the civil simulation community and how simulation and animation tools used in the civil world can work together under HLA.

2. Public Transport Prototype

The main application described in this paper is a simulation of the public transportation system in the city of Magdeburg [1]. The actual simulation model behind it is a classical analytical (or *constructive*) simulation. The most interesting interoperability aspect is that an real-time data source is used in addition to the model. Communication between the simulation and the on-line data source, as well as with other federates, is based on the HLA interface.
There are different combinations of federates that can be used for the federation depending on the specific goal of the federation execution (Figure 1).

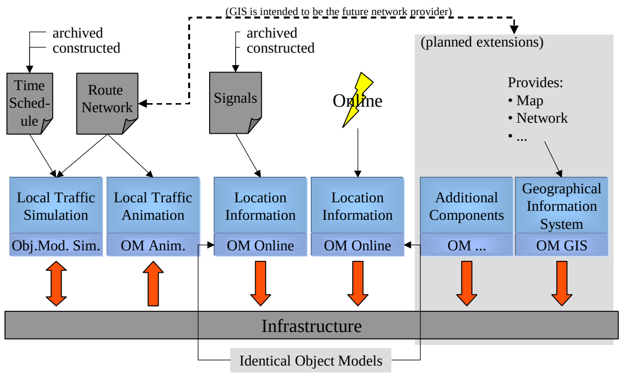

Figure 1: Setup of the traffic management prototype

Typical goals of the federation would be to use an up-to-date simulation model (i.e. a simulation which has the real-life positions of the streetcars) to do an as-fast-as-possible forecast to reveal certain bottlenecks so the control center manager could conduct evasive actions. In this configuration one would use the simulation federate and the on-line federate. To do the forecast one would have to disconnect the on-line federate or to clone the simulation federate. Another interesting alternative is to use constructed off-line position updates to test certain schedule alternatives.

The following sections describe each of the federates that have been developed and their main purpose.

2.1 The Simulation Federate

The simulation federate is the "heart" of the federation. It is a simulation model which performs a schedule based simulation of the public transportation system in Magdeburg (i.e. streetcars) and has been developed using the simulation system SLX [2].

The simulation model uses the SLX-HLA-Interface [3] 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 is a classical analytical simulation model using logical simulation time.

2.2 The Animation Federates

Two different tools have been used for producing on-line animations of the federation.

The first tool is the web-based animation system Skopeo [4]. Skopeo is a general animation system which provides platform-independent system animation anywhere in the WWW. Skopeo has a prototypical HLA interface which is based on the beta version of the Java RTI 1.0 from DMSO which is no longer supported.

The second tool which has been used in this federation is Proof Animation for Windows [5]. 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 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.

2.3 The On-line Federate

An on-line federate is used to provide position updates from the streetcar vehicles for the other federates.

The on-line federate starts a receiver process 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.

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.

3. Integration of On-line/Real-Time Data

As can be seen from the description of the federation, it was necessary to integrate real-time dependent data, which is generated by a real-life process totally independently from the actual federation run, into the federation. The real-time data was transferred on-line to the site where the federation was run; therefore we refer to it as on-line/real-time data.

For our federation an approach which transparently integrates this on-line/real-time into the federation was needed.

It was chosen to use a separate HLA federate (the on-line federate), which would act as provider for the rest of the federates within a given federation.

The possible structure of an on-line federate is shown in Figure 2.

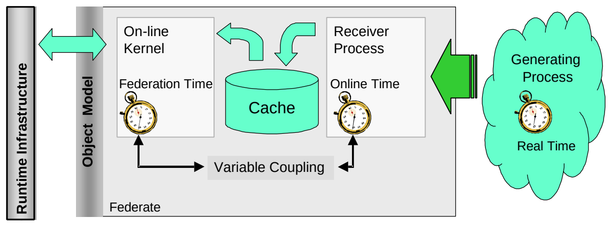

Figure 2: Suggested structure of an on-line federate

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, i.e. it sets the an upper limit on the time to which all other federates are allowed to advance.

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 (because it simply does not constrain itself to HLA time management). 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.

Only after a time advance grant is received, the buffered on-line update will be sent into the federation.

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 3) gives a detailed view on the possible structure of an on-line federate.

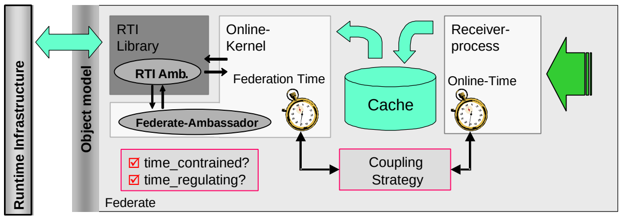

Figure 3: Detailed structure of an on-line federate with respect to HLA

4. Animation Systems under HLA

In the civil simulation community it is frequently the case to have separated simulation and animation tools. In a monolithical analytical simulation the simulation model would produce some kind of trace file which would then be read by an animation tool (e.g. Proof Animation™) which would produce a post-run animation of the simulation. The layout file usually needs to be produced separately.

Under HLA post-run animations play a less important role. Although they may prove useful for certain mission rehearsals and replay scenarios, on-line animations of single federates or the entire federation are usually desirable.

The same applies for civil HLA federations where usually on-line animations are required. Therefore we adopted two general animation systems to be capable to perform on-line animations under HLA. The tools and how they were integrated into the federation are described in the next sections.

4.1 Proof Animation for WindowsÔ

Proof Animation™ provides on-line and post-run animation on Windows Platforms (i.e. Windows 95/98/NT) and can be used by a wide variety of programs. While the post-run version of Proof is a stand-alone executable, the on-line version is provided as a Windows library (DLL). In order to produce on-line animation, a program to drive Proof (i.e. make calls to the DLL) is needed. Since Proof is tightly integrated with SLX and an HLA-Interface for SLX was already available, it was chosen to use SLX to drive Proof Animation.

Several general alternatives can be considered to use the combination Proof/SLX in HLA federations:

1) Use Proof animation for visualization of a single federate. In this case the SLX federate would ideally drive Proof Animation directly.

2) Use Proof Animation for visualization of the entire federation. One existing SLX federate has the secondary task to also produce the on-line animation of the entire federation. This alternative is only possible, if an animation is only needed at one location.

3) Use a separate SLX-program acting as a passive viewer federate which only receives the position updates and produces an on-line animation on its own. This has the advantage that a variable number of animations can be produced at different locations.

All three 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 for animation purposes only.

Alternative 2 provides an easy alternative for providing an animation possibility for the entire federation.

Alternative 3 is the most 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 2. Additional animations can be obtained at the same time (*non-exclusively*) by Skopeo and potentially by a Proof/SLX-federate based on approach 3. Figure 4 shows a screenshot of the system obtained with Proof Animation.

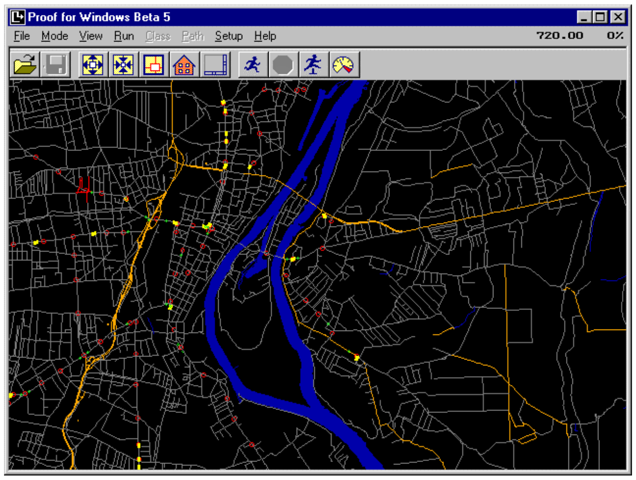

Figure 4: Screenshot of the streetcar federation obtained with Proof Animation for Windows

4.2 Skopeo Animation

Skopeo is a general animation system which provides platform-independent system animation anywhere in the WWW [4,6].

Skopeo also exists in a post-run and in an on-line version. The HLA compliant Skopeo is based on the Beta-Release of the Java-RTI 1.0 from DMSO.

Although this RTI release had some intrinsic problems and never reached production stage (support and development were terminated after the Beta-Test program), it was possible to use it for Skopeo. The Skopeo applet itself uses Corba mechanisms to communicate with its host server.

Figure 5 shows a screenshot of the transportation federation obtained with Skopeo Animation.

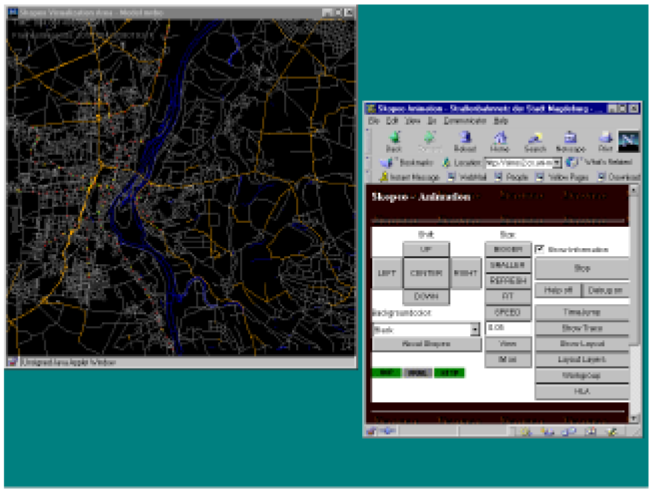

Figure 5: Screenshot of the streetcar federation obtained with Skopeo Animation

5. Simulation Systems under HLA

There are some general differences regarding the development of simulations in the military and in the civil simulation community. In the civil community it is very rarely usual to develop a simulation in a programming language like C++ or Java, even though several packages for supporting the development of simulations in these languages (e.g. Silk for Java) exist. There are a couple of reasons for that; the most important one is probably related to the very comfortable way one can develop simulation models with sophisticated and specialized simulation tools (e.g. Arena, Automod, GPSS, Modsim, Pro Model, SLX, etc).

Another reason is that the typical application areas are different: one would never think of developing a training simulator with the tools mentioned above, because this may prove to be very difficult, if not impossible. For analytical simulations, the situation might be different, because this is the actual strength of these tools.

In order to use the tools discussed above, concepts for HLA interfaces need to developed and implemented. The University of Magdeburg has developed several concepts for doing this [7].

Generally, HLA Interfaces for these tools can be classified by two aspects:

- the interface developer's point of view, and
- the model developer's point of view.

The interface developer's point of view relates to the question, how and where the actual access to the HLA API is performed. For this task the following alternatives can be considered (see [3]):

1. Re-Implementation of the tool with HLA-extensions
2. Extension of intermediate code
3. Usage of an external programming interface
4. Coupling via a gateway program.

The model developer's point of view relates to the issue, how the access to the HLA interface from the model looks like. The two general alternatives which we've shown to be feasible are:

1. Explicit access
2. Implicit access.

In 1) the model has to issue certain functions calls to the HLA interface actively, i.e. the HLA interface has to be called from inside the model. Also, the model has to take care of receiving data or at least, of processing incoming data.

In 2) the model does not have to be enriched with HLA functionality. The user only specifies a mapping between internal data (e.g. variables, object classes, attributes) and the data of its SOM. The actual access to the HLA interface is done from the runtime kernel of the simulation tool. This alternative has the advantage that the model description is independent from the fact of whether you are developing a federate or a stand-alone simulation.

6. 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. Some applications in the area of transportation and logistics have already been developed [8,9,10].

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.

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) internally or externally to the federation. This can be especially useful for our federation to provide as-fast-as-possible analyses without interruption a running federation.

7. References

[1] HLA at the University of Magdeburg. The Streetcar Federation. URL http://isgsim.cs.uni-magdeburg.de/hla/fed-streetcar.html.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

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

[8] 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.

[9] 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.

[10] Schumann, M., E. Bluemel, T. Schulze, S. Straßburger, K.-C. Ritter, Using HLA for Factory Simulation. In: *Proceedings of the Simulation Interoperability Workshop* Fall 1998, Orlando.

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, Geographic Information Systems and distributed simulation-based systems.