Using HLA for factory simulation

Marco Schumann, Eberhard Bluemel
Fraunhofer Institute for Factory Operation and Automation
Sandtorstrasse 22
39106 Magdeburg, Germany
+49-391-4090-158, +49-391-4090-110
schuma@iff.fhg.de, bluemel@iff.fhg.de

Thomas Schulze, Steffen Strassburger, Klaus-Christoph Ritter
Department of Simulation and Graphics in the Department of Computer Science
„Otto-von-Guericke“ University, Magdeburg
Universitätsplatz 2
39106 Magdeburg, Germany
+49-391-67-12017
tom@isg.cs.uni-magdeburg.de, strassbu@isg.cs.uni-magdeburg.de, kcritter@isg.cs.uni-magdeburg.de

Keywords:
Factory simulation, SLX, Simulation tools, Visualization

ABSTRACT: *Starting with a short analysis of the current situation in the field of factory simulation and an overview of current tendencies in the manufacturing area, the article introduces the work carried out in order to integrate HLA and existing simulation tools. It presents the simulation tool SLX and the visualization tool Skopeo, which were both utilized to perform a prototype federation of a manufacturing plant. The article is concluded by looking at further chances for HLA to also become a standard in the field of factory simulation.*

1 Current situation in factory simulation

Simulation has become an important tool for planning the complex process structure of a company's business. Whereas during the last few years simulation was mainly focused on a few key activities of a company’s overall business it recently became more important to consequently involve simulation in all parts of a factory’s life cycle.

In the design stage, simulation can be used to determine capacities of certain production facilities as well as to estimate efficiency and throughput of the future factory. The same simulation model might be used to evaluate different control strategies in subsequent stages. Also, the simulation model can support the training of personnel before the factory is even finished.

During the last few decades a variety of simulation tools has been developed. Most of them include model elements specialized in a certain area of application. Each simulation system usually has its own paradigm of how to map the real world's objects into the simulation model. Each product comes with its own version of a model editor and has its own way of presenting the simulation results ranging from simple text-based output to sophisticated three-dimensional graphical scenes.

Nevertheless, the great amount of time spent on acquiring input data, programming the simulation model, and the considerable effort of adequately visualizing the simulation results have been often criticized in the past. They conflict with current tendencies in the manufacturing industry. It is therefore necessary to research new concepts of factory simulation.

2 Tendencies within manufacturing

The shortening of a product's life cycle, the rising variety of customized products, as well as increased competition in the market force companies to transfer new technologies into production more rapidly in order to gain advantages over their competitors. This requires simulation models not only to be created cheaper and with less time but also to be easier to adapt. These improvements can be accomplished by increasing the reusability of simulation models, or at least parts of it.

Often plants of a certain manufacturing branch consists of similar components. For example, there are several car manufacturing plants in Germany but only very few manufacturer of paint shop equipment. Therefore, the

same hardware might even be installed in factories that produce cars of different brands. Suppliers of such equipment could take advantage of that fact and could already offer simulation models for their components. These models could be used by the customers to be included in their overall factory simulation model. That opens up a new market for the supplier and is also an advantage for the customer since it decreases development time for simulations.

Another tendency in the manufacturing industry is the globalization of the market. Complex products are very seldom manufactured by only one company. Usually components are produced by different companies and assembled at a different location. Capacities of ware-houses are reduced and often parts are delivered "just-in-time" as needed, which increases the dependency between companies. Therefore, it will become more important in the future for a simulation study to also include the simulation models of the company's suppliers. This raises the need for interoperability of simulation models to solve logistical problems.

Interoperability is also important because it increases flexibility of the simulation model. In case of an alteration of co-operation between companies, it is easier to adapt the simulation model to the new circumstances. If all simulation models adhere to the same standard, the component that simulates the old partner can be replaced by a new one. Interoperability is especially important because different industry partners use different simulation tools according to which tool is most feasible for their purpose. Reusability of their models can only be achieved by setting up a standardized interface through which interoperability is granted.

Due to the increased graphical capabilities of today's computers, there is also a new field of application for computer simulation: user training. Although many complex devices are computer-controlled, some of them still need to be supervised by humans. Incorrect operation can often cause substantial damage. For this reason, simulation tools are already in use, for instance, to qualify railroad engineers to operate Germany's high speed trains or for training drivers to operate harvesting machines. These tools, however, have the restriction that the teacher and trainee have to physically be at the same location. Also, collaborative work is not simulated adequately. Here, distributed simulation will become more important.

A very powerful application of distributed simulation is also the training of personnel for coordinating control stations within a factory. For training purposes, the trainee might use a one-by-one replica of the control panel whereas the real machines might be transparently replaced with a simulation model running on a different computer. Once the trainee is experienced enough, he/she will not actually control a simulation model but instead the real machine. This provides a high level of reality within the simulation.

To conclude the section above, it can be said that the current problems that simulation faces can be improved by consequently applying the concepts of reusability, interoperability, and distributed simulation to factory simulation. Hence, HLA has a promising basis to become an accepted standard within the field of factory simulation, too.

3 Towards an HLA based factory simulation

As mentioned in the first section, most factory simulation developers prefer to use specialized simulation tools rather than programming languages like C++. By being convinced to move to C++, developers could use the advantages of HLA immediately, but it would also imply the loss of an often highly specialized simulation and modeling environment. In most cases, this is not acceptable, therefore, propagation of HLA within the field of factory simulation will only be successful, if there are appropriate simulation tools available. Currently, this is not the case.

For this reason, research work focused on opportunities for the integration of classical stand-alone simulation tools into the High Level Architecture. In general, adapting a simulation tool to be HLA-compliant will require modifications of the source code as well as inside knowledge of the software. In particular, it is necessary to have access to the internal event list and detailed knowledge of the internal representation of data types within the simulation tool. For this reason, it can only be accomplished through a close co-operation with the software provider.

The simulation tool SLX has been selected as a platform to gain first experiences with HLA-based factory simulation. Reasons for this decision were the availability of the required inside information and the availability of a library for factory simulation. As a first step, SLX needed to be integrated into HLA. This was accomplished by Steffen Strassburger [1]. The results of his work will be described in chapter 4.

Another part of the research focused on adapting a visualization tool. For this purpose the Java-based tool Skopeo developed by K.C. Ritter [2] has been chosen.

The major advantage of this tool is the availability of its source code since it was developed at the University of Magdeburg. A description of HLA-Skopeo is given in chapter 5.

4 SLX

4.1 What is SLX?

SLX is a discrete event simulation tool for Windows 95/NT operating systems developed by the Wolverine Software Corporation. [3] SLX is a classical simulation language-oriented stand-alone tool that includes a programming language with a C-like syntax. The program has an open software architecture. It has a library interface which allows to call functions in any standard Windows DLL from SLX. This is one of the basic prerequisites for connecting a commercial tool like SLX to the RTI without modifying the actual source code of the tool. In addition to this interface, SLX also offers the extraordinary feature of writing DLL header files. This feature allows the user to take a look into the inside of SLX. The DLL header file discloses the internal structure of SLX objects with their attributes and therefore, facilitates the external modification of SLX objects (e.g. from a DLL).

Although the facts mentioned above identify SLX as a very open tool, not all of the requirements that the HLA interface specification imposes can be fulfilled at the first view. It is not possible to directly create instances of C++-objects inside SLX, which means it is not possible to call the methods of the RTI ambassador object directly from SLX.
It is furthermore not possible to directly implement the federate ambassador object with its callback methods inside SLX.

Another fact to consider is the difference between SLX and C/C++ data types. Although SLX has a C-like syntax, it has a different data type concept. This concept is meant to prevent SLX users from some of the pitfalls of C (e.g. freeing memory although it is still being referenced, exceeding the boundaries of arrays, strict type checking, ...). The entire string concept is completely different from the C approach: SLX internally keeps track of the current and maximum length of strings and does not use the zero-termination of C.

4.2 Integrating SLX into HLA

4.2.1 The Wrapper Solution

The solution to all problems mentioned above is very straight forward: A wrapper library (Figure 1) can be used to provide access to the RTI ambassador methods from SLX. This is accomplished by creating a normal C-function for each function that needs to be accessed from SLX.

The library itself has been developed using Microsoft Visual C++ 5.0 and the RTI libraries from the RTI 1.0.3 distribution for Windows NT 4.0.

These wrapper functions perform all necessary type conversions between SLX data types and C/C++ type.

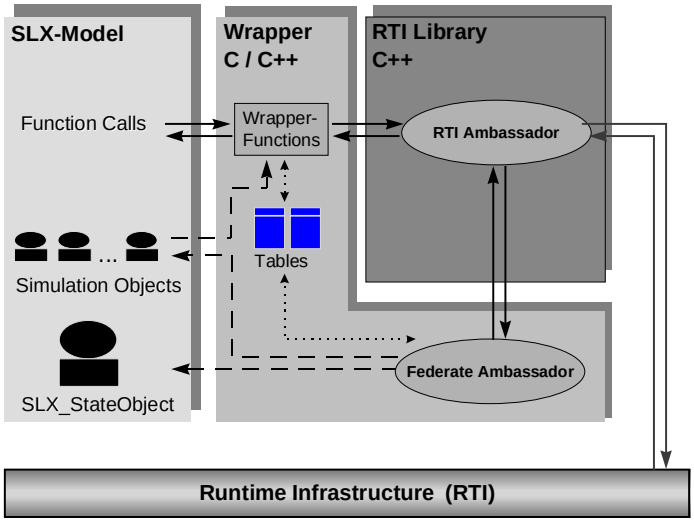

Figure 1: Wrapper library for SLX

The functions also simplify the whole process of programming with the RTI: the entire RTI-ID handling (creating attribute- and parameter-handle-pair-value sets, obtaining RTI-handles for class and attribute names, etc.) is handled internally.

The wrapper functions have direct access to the objects simulated in SLX. They internally keep track of the mapping between SLX objects and HLA objects and interactions using mapping tables. It is necessary to pass a pointer only once to the memory areas associated with the SLX objects during runtime. The functions can automatically detect the data types of SLX object attributes. Therefore the process of generating updates or sending interactions is very simple for the end-user: He/she only issues an „RTI_UpdateAttributeValue" function call and passes the object ID and all names of all parameters that have to be updated.

The second major task of the wrapper library, in addition to providing access to the RTI ambassador methods, is to implement the federate ambassador. The federate ambassador is responsible for receiving all kind of data from the RTI via callback invocations. This reception process is also handled internally by the wrapper library. Since the wrapper library cannot notify SLX that something has happened (SLX does not allow any user-defined callback functions), a mailbox-principle is used: Certain state information that the federate ambassador receives is stored in an SLX object with a fixed structure. The SLX model can then access this object to check the things that have changed on the RTI side. This is quite a suitable solution and not a very unusual approach.

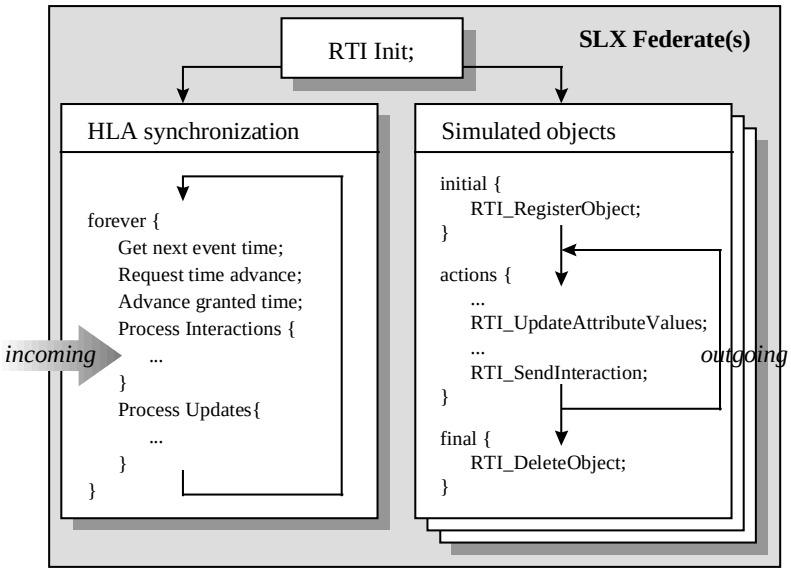

Figure 2: The User's Point of View

In addition to this static object our solution also introduces dynamic structures for the actual objects that are being modeled. If a simulation has subscribed to certain object classes and interactions, a mapping onto associated SLX objects is performed. If an update is received, it can be written directly to the appropriate SLX memory area. The same mechanism applies for interactions.

The problem with this approach is that the wrapper library does not know the structure of these SLX objects at compilation time. The DLL has to therefore calculate the address of each object attribute at runtime. Certain conventions have to be considered in order to do so.

With this dynamic approach we were thus able to build a model-independent wrapper library. This means that even if the simulation model inside SLX changes, no changes to the wrapper library are necessary.

4.2.2 Synchronization Issues

The HLA programming paradigm expects a federate to tick the RTI to trigger the reception of any callback invocations by calling the tick-method of the RTI ambassador. This should usually be done between requesting a time advance and waiting for the according time advance grant. As a simplification for the SLX user this is handled internally by the wrapper library. The SLX user simply requests to advance to the next logical event time and then (after a while) receives a time advance grant. If an external event has to be processed, the grant time may be smaller than the actual time advance that had been requested. The user is then expected to check the static and dynamic SLX objects for any data or events that need to be processed. After that the normal simulation execution can proceed; possibly by issuing the same time advance request again.

To ensure the behavior outlined above, the according functions of the wrapper library for requesting the time advances work synchronously, i.e. the wrapper library keeps control until a time advance grant is received. This is different from the original RTI ambassador functions which work asynchronously.

4.2.3 The Users Point of View

Everything stated up to now has been related to the implementation side of the solution for SLX. Since the primary goal of our solution was to provide the end-user with an easy access to the HLA functionality, this section elaborates on the user's point of view.

The actual modeling paradigm under SLX is not effected by being part of a distributed simulation. Some minor things have to be taken care of.

The most important change probably relates to the synchronization with other federates: a special synchronization thread has to be added to the model (see Figure 2). The synchronization thread should have the lowest priority in the model to ensure that at a certain logical simulation time all actual „simulation" events

have been processed before the time stamp of the next event is determined. This way no event can be scheduled with a time stamp lower than the one for which the request is issued.

The other change to the modeling paradigm relates to the classification of SLX objects. The following types can be identified:

- *Normal SLX objects:*

They are of no interest to the federation.

- *Public objects:*

They are of interest to other federates and need to be published and updated.

- *Local copies of remote objects:*

They are updated from the outside. The user has to take care to integrate them into his/her model.

In the second case the initial and final properties (which can be compared to constructors and destructors in C++) of the SLX objects have to be enhanced by only two function calls (see Figure 2). The only major change applies to their action property: Each time a change to an attribute is made, other federates have to be informed by sending an update.

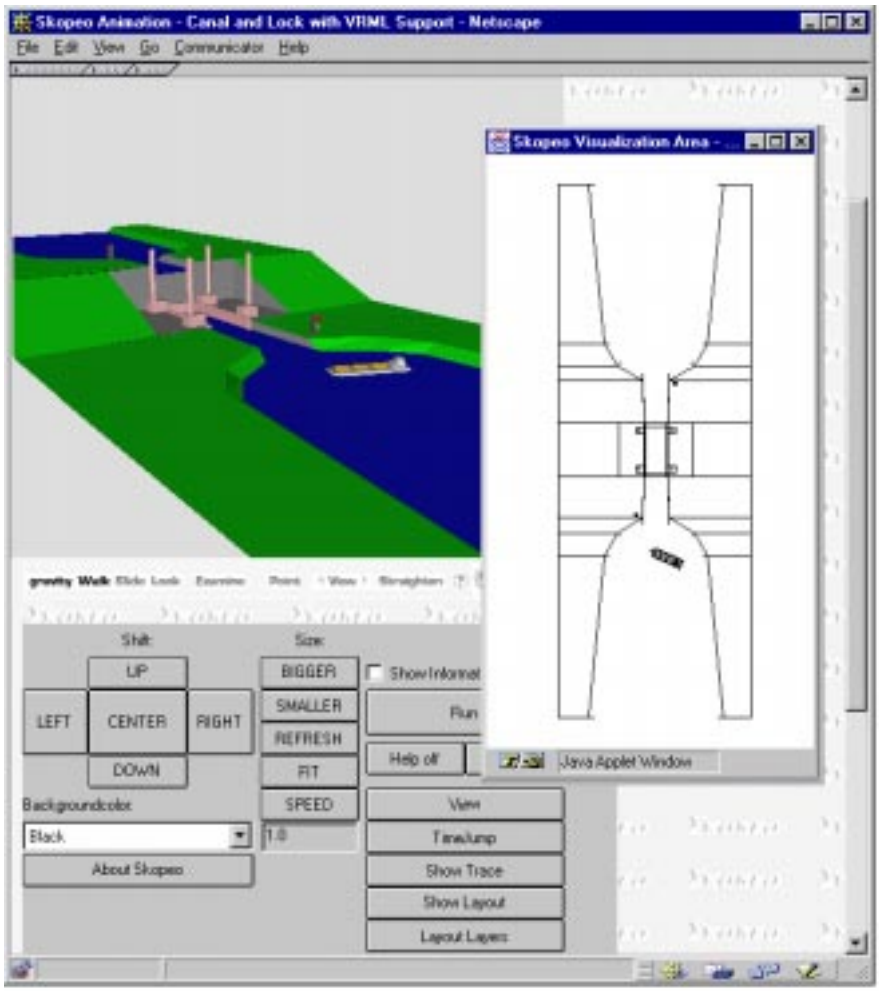

Figure 3: Screenshot of a 3D- and 2-Skopeo visualization

The third scenario is slightly more complicated: each time an object which matches the federate's subscription interests is detected, the federate ambassador notifies SLX about the existence of an object. The SLX model then has to create a local copy of that object (a ghost) and register it with the wrapper library. All subsequent updates are stored in this object instance. The SLX model has to react appropriately to every change. This can be done by using the SLX mechanism of control variables.

4.2.4 Possible Improvements

The current version of the wrapper library represents a simulation model independent HLA-connection for the simulation tool SLX. The solution provides access to a basic set of HLA functionality from SLX. It comes with an easy to use interface for utilizing HLA without having to care about the complicated programming work with the RTI. Certain higher techniques of HLA still need to be implemented. This mostly relates to the services of the HLA data distribution management, which had not been implemented in the current versions of the RTI provided by the DMSO. RTI support for this service group is announced for later this year and will then be incorporated into future versions of the wrapper library for SLX [4].

The next step in our development efforts will be to implement the concept of ownership transfer into SLX by providing access to the methods of the Ownership Management service group of the HLA Interface Specification.

Other areas of research are the investigation of different time management regimes. The current version uses a conservative approach with the possibility of using zero lookahead.

5 Visualizing the simulation results

As mentioned above, most factory simulations are currently developed with specialized simulation tools that usually provide functionality for representing the simulation results. Executing distributed simulations, however, requires a tool that can not only visualize its own data but also objects controlled by other simulations.

In general, there are two approaches of computer animation: active animation and passive animation. In a passive animation, the visualization tool only collects data and ensures events to be visualized in the proper order. There is no user interaction possible. In an active animation, the user can interact with the visualized objects. The latter is much more complicated to implement and to handle, therefore, the current implementation of the visualized tool uses passive animation.

During the research for an appropriate visualization tool, it turned out that the easiest approach was to adapt the already exiting Java-based tool Skopeo to comply with the HLA standard. Skopeo was developed by K.C. Ritter - a Ph.D. student at the University of Magdeburg. It had the best premises because its source code was available, and its Java-based implementation already allowed for a utilization in a networking environment widely independent from a given hardware platform.

The program runs as a stand-alone application as well as an applet within a web browser. Once the required data for a post-run scenario is downloaded from the Skopeo server, the animation completely runs on the client's machine. The Skopeo kernel does not distinguish between real-time data, such as needed for an online visualization, and trace data, such as needed for a post-run animation. Both kinds of data are inserted into the kernel's data basis by a thread synchronized with the real time clock.

By default, Skopeo displays the data as a two-dimensional graph within a window outside of the actual HTML page. The window size may be changed according to the computer's performance. In addition to the default display, Skopeo can communicate with other applications which are part of the same HTML page. This feature has been used to also allow the data to be displayed with a VRML browser (Figure 3).

In order to allow for a communication between all federates a workgroup concept has been established. A deamon running on the Skopeo host server receives data from all connected web browsers and can also sent data to each web browser, if desired, regarding the safety restriction implied by Java. This deamon has been modified to act as an HLA federate.

A major point to discuss is the time management. Basically, there would have been three alternatives:

- *always up to date:*

Within discrete event simulation time progresses in discrete steps. An animation trying to update the scene will immediately appear jerky in most cases.

- *floating mean value:*

An average time advance between two time steps is dynamically calculated. The disadvantage of this technique is the varying speed of animation.

- *buffered:*

Any data to be visualized is recorded in a buffer. The animation federate reads from the buffer at a speed determined by the user. In case the user sets the speed too high, i.e. faster than the simulation clock progresses, no data is lost, but the animation is slowed down and becomes jerky again.

For the implementation of Skopeo presented here, buffered animation has been selected. It allows for a time continuous animation. In addition, all displayed data can be recorded, and the speed of the animation can be changed dynamically by the user without slowing down the entire federation.

After joining the federation, Skopeo subscribes to all object attributes and interactions that are of interest for the visualization. Currently, Skopeo does not publish any attributes or interactions. In order to display the simulation objects at the proper locations, at least logical locations have to be defined in the SOM. In order to minimize the traffic between federates the detailed geometric representation of the simulation objects has been defined separately in 2D- and 3D-geometric data file. When the federation is executed, only the ID's of locations are transferred.

The next chapter introduces a small example federation that uses SLX and Skopeo federates. Another federation using SLX and Skopeo is introduced in [5].

6 Example Federation

A small sample federation has been established in order to test both tools. The federation is designed to consist of three federates maximum.

The simulation federate simulates a material flow through a small manufacturing area. Parts enter the virtual factory through the source and have to be processed sequentially by machines of three different

types. There is one machine of type 1 and 3, but two machines of type 2 so that parts may be handled alternatively by either machine 2.1 or 2.2. All machines are connected through conveyor belts. Parts leave the system after they have been processed by machine 3 (see Figure 4). The simulation federate is time-constrained as well as time-regulating.

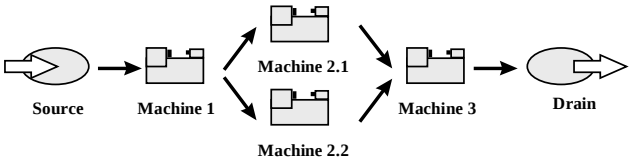

Figure 4: Material flow to be simulated

A second federate is used to simulate a control station which is intended to serve for training purposes. From time to time, a machine simulated by the first federate may fail. This is noticed by the control station federate which brings up a message to the user. In a training scenario, the user must then make arrangements for the machine's repair. He has to confirm when the machine is repaired and ready for operation. This will cause an interaction to be sent which is evaluated by the simulation federate that will then include that machine in the simulation again. The control station federate runs real-time proportional and it is time-regulating. The simulation and the control station federate, both are implemented using the SLX-RTI library.

The third federate is the visualization federate, Skopeo. It is used to visualize the simulation (Figure 5).

Although this federation is quite simple, it demonstrates that existing simulation tools can be adapted to be HLA compliant. Thus, it also demonstrated that HLA can be used in the field of factory simulation.

7 Conclusion and Outlook

As pointed out at the beginning of this article, current trends within the field of manufacturing will require simulation models to be developed more rapidly, and the market globalization will raise a need for increased interoperability of simulation models. HLA is a feasible concept to also be applied in the field of factory simulation.

Whereas simulations in the US military are often developed using C/C++, this is not the case in the field of factory simulation. Thus, a successful introduction of HLA in the field of factory simulation is closely related to the availability of HLA-compliant factory simulation tools. As demonstrated in this article and examined further in detail in [5], existing simulation tools can be extended to support HLA. However, this requires a substantial initial amount of work as well as inside knowledge of the simulation tool. In most cases it can only be accomplished by the supplier of the software.

Designing HLA-compliant simulations requires a certain amount of additional work, which is rewarded by the advantages of interoperability and reusability. These advantages, however, only apply when there already is a wide basis of simulation modules developed that use HLA. In contrast to military demands, there are no regulations to utilize the HLA in the field of factory simulation. This is the reason why the introductory time for HLA will be much longer than it is in the field of military simulation.

Also, it can be expected that some companies have no interest in developing interoperable simulation

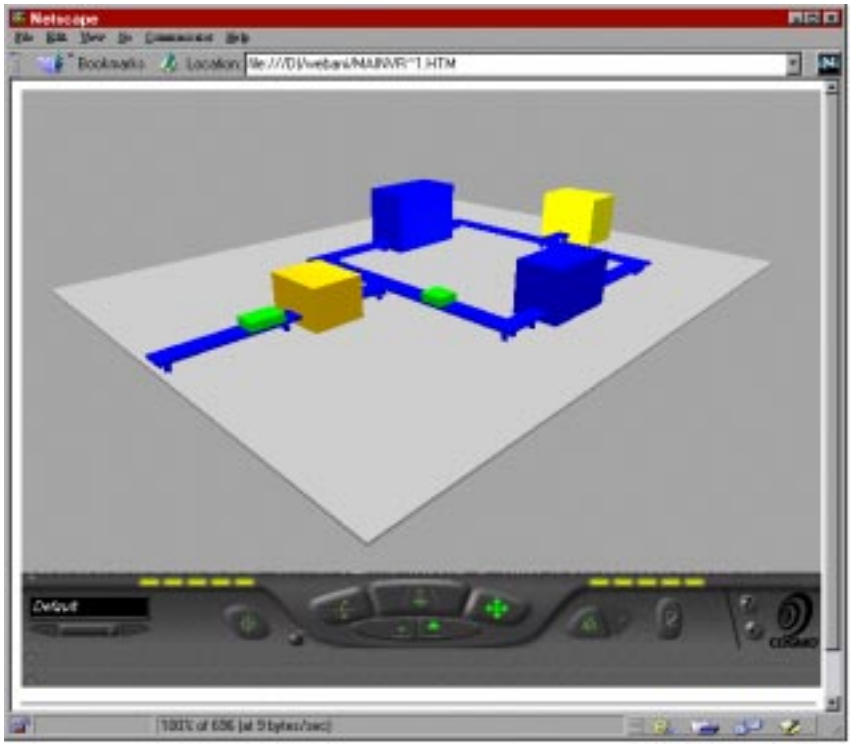

Figure 5: A screen shot from the Skopeo visualization

components. Big companies often develop their own software packages and require their customers to use their software. This creates a certain level of dependence which is needed to bind a customer to that company. To summarize, it can be stated that the HLA can be usefully applied in the field of factory simulation, but because of the different background and regulations, it will require more time.

Despite these facts the Fraunhofer Institute and the University of Magdeburg will continue to research the advantages of the HLA architecture. Research at the Fraunhofer institute will concentrate on developing new concepts and software that exploits the advantages of HLA for the training of multiple users at multiple sides. Here HLA opens new possibilities of distributed user training – certainly a worthy topic of another paper.

References

[1] Strassburger, S.; Klein, U.: "Integration des Simulators SLX in die High Level Architecture" , In: [8] pp. 32-40

[2] Ritter, K. C.: „Skopeo - Animation", http://simos2.cs.uni-magdeburg.de/skopeo/

[3] Henriksen, J. O.: "An introduction to SLX". In: "Proceedings of the 1997 Winter Simulation Conference", eds. Andradoittir, S; Heally, K. J.; D.H. Withers, and B.L. Nelson, pp.559-566

[4] Strassburger, S.: The SLX with HLA Homepage. Available online under http://www.cs.uni-magdeburg.de/~strassbu/SLX_with_HLA.html

[5] Menzler, H.-P.; Klein, U.: „Distributed Traffic Simulation based on the High Level Architecture", Paper 98F-SIW-016 presented at the Simulation Interoperability Workshop Fall 1998, Orlando, Florida, 1998.

[6] Schulze, T.; Klein, U.; Strassburger, S.; Ritter, K.C.; Bluemel, E.; Schumann, M.: „HLA basierte verteilte Simulationsmodelle fuer die Fertigung", In: [8], pp. 19-31.

[7] Schumann, M.: „Adaptive Generierung von Simulationsmodellen auf Grundlage von SLX", Master's thesis, department of computer science, University of Magdeburg, 1998.

[8] Lorenz, P.; Preim, B.: „Simulation und Visualisierung '98", Society for Computer Simulation International, Ghent/Belgium, 1998.

Author Biographies

EBERHARD BLUEMEL is the Department manager of the Department for Planning and Visualization Techniques at the Fraunhofer Institute Magdeburg. He holds a Ph.D. in natural science from the University of Magdeburg. His research fields include discrete optimization, operations research, logistics, simulation, and Virtual Reality.

KLAUS-CHRISTOPH RITTER holds a diploma in computer science and works as a Ph.D. student at the Institute for Simulation and Graphics within the Faculty for Computer Science of the Otto-von-Guericke University of Magdeburg. He is using Java and other W3 technologies in combination with simulation environments to visualize simulation results.

THOMAS SCHULZE is an Associate Professor at the Department for Computer Sciences of the Otto-von-Guericke-University 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, an organization for computer simulation in Gemany.

MARCO SCHUMANN is an employee at the Fraunhofer Institute Magdeburg. He holds a Master's degree in Computer Science from the Otto-von-Guericke-University Magdeburg. His experiences in developing simulations and applications for the Internet include a one-year-stay at the University of Wisconsin, Stevens Point. His main research interest lies in application of simulation methods in the field of factory planning and optimization.

STEFFEN STRASSBURGER is a Ph.D. student at the Department for Computer Science of the Otto-von-Guericke-University Magdeburg. He holds a Master's degree in computer science from 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 interest lies in distributed simulation and High Level Architecture.