DISTRIBUTED SIMULATION WITH JAVAGPSS BASED ON THE HIGH LEVEL ARCHITECTURE

Ulrich Klein, Steffen Straßburger, Jürgen Beikirch
Institute for Simulation and Graphics (ISG), Faculty of Computer Science
Otto-von-Guericke University Magdeburg
Universitaetsplatz 2
D-39106 Magdeburg
email: {uklein@isg, strassbu@sunpool, beikirch@sunpool}.cs.uni-magdeburg.de

KEYWORDS

Discrete Simulation, High Level Architecture, GPSS/H, Interactive Simulation, Java, Internet.

ABSTRACT

This paper focuses on the High Level Architecture for reusable and interoperable distributed simulation, platform independence, and how classical simulation tools can fit into this new arena. The current integration efforts focus on GPSS/H and SLX; in this paper the GPSS/H track is described. The experiences using the standard GPSS/H version and the reasons for a Java implementation of GPSS/H, JavaGPSS, are sketched. The introduction of JavaGPSS and the incorporation of distributed simulation mechanisms based on HLA into this tool build the main issue of the remainder of this paper. New application areas are mentioned as an outlook on this new dynamic technology.

1 DISTRIBUTED SIMULATION

In the past few years, the field of distributed simulation, driven by the rapid expansion and acceptance of the Internet and its multimedia front-end, the WWW, has become one of the most promising and challenging technologies in modeling and simulation. The large number of different approaches to distribute and interoperate simulations (e.g. *Aggregate Level Simulation Protocol ALSP*, *Distributed Interactive Simulation DIS*) has now lead to a unifying approach, the *High Level Architecture* (*HLA*), which has the challenging vision to support its predecessors and different time regimes.

2 HIGH LEVEL ARCHITECTURE

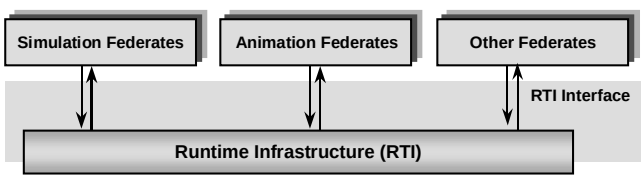
The High Level Architecture is a simulation interoperability standard currently being developed by the US Department of Defense (DMSO 1997).

The architecture is defined by :

1. rules which govern the behavior of the overall distributed simulation (*Federation*) and their members (*Federates*) (DoD 1996).
2. an interface specification, which prescribes the interface between each federate and the Runtime Infrastructure (*RTI*), which provides communication and coordination services to the federates. Federation communication only takes place between each federate and the RTI, not between federates themselves. The RTI as the central coordination software component as well as the federates can be located on any networked computer on an Intranet or the Internet (DoD 1997).
3. an *Object Model Template* (*OMT*) which defines the way how federations and federates have to be documented (using the *Federation Object Model*, *FOM* and the *Simulation Object Model*, *SOM*, resp.). The OMT uses a tabular approach which is well suited for automated tools and conversion into the *OMT data interchange format* (*OMT DIF*). OMTs promote the reuse of single federates or federations as a whole. Federations can be viewed as a contract between federates on how a common federation execution is intended to be run (DoD 1997a).

The time management services provided by HLA allow the transparent running of federates under different time regimes (e.g. real time, time stepped, event driven) (DMSO 1997a).

Even though it originates from military application surrounding, the architecture seems to be very well suited for civilian applications, too. Together with web-enabled simulation and animation components, an exciting new variety of applications and collaboration modes could be developed (Picture 1).



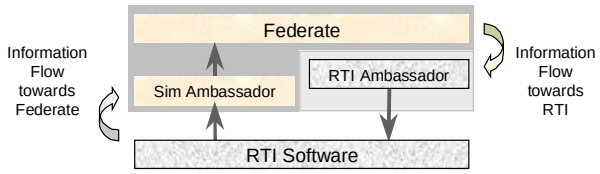
Picture 1 HLA-based Simulation and Animation

REQUIREMENTS

The requirements imposed on the simulation and animation tools by the HLA are considerable, which may be one of the main reasons for the fact that most of the example simulations released up to now were written in C++. Therefore, a closer look at the connectivity between classical simulation tools and the High Level Architecture seems to be necessary.

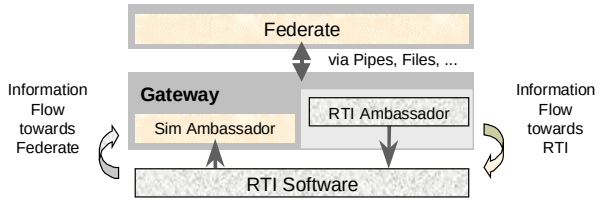
Typical stand-alone applications like GPSS/H (Schriber 1991) and SLX (Henriksen 1997) as simulation tools and Proof Animation (Wolverine 1992) as animation software only provide very limited means for simulation execution co-ordination or communication. Additionally, Proof Animation, as a post-run trace file animation tool, lacks dynamic and/or real-time capabilities.

In order to participate in an HLA-based federation as a full-featured federate, a tool should be able to co-operate with the Runtime Infrastructure via a two-part interface based on an ambassador metaphor (Picture 2). The *RTI ambassador* is a software library to be linked to the tool or model; it is included in the HLA distribution and contains methods called by the federate. Vice versa, the *federate ambassador* provides methods called by the RTI and has to be developed and implemented as part of the modeling process.



Picture 2 Federate/RTI Ambassadors (HLA Approach)

The necessity to either include or implement these ambassadors imposes some major problems on several classical simulation tools. If some of the requirements could be relaxed depending on the application domain, a loose coupling to the RTI (e.g. by gateways, see Picture 3) would be feasible, too.



Picture 3 Federate/RTI Gateway approach

The possibility to connect Proof and GPSS/H with the RTI via Gateways (e.g. by the use of files, pipes and programs translating between RTI calls and appropriate file entries that can be read by GPSS/H or Proof) is a solution that does not use all the flexibility that the HLA concept has to offer. Ways of coupling these standard software tools to the RTI software in a more sophisticated manner will be further investigated. A solution for SLX, which uses a C-wrapper-library for translating between SLX and the RTI, has been successfully implemented in a related project.

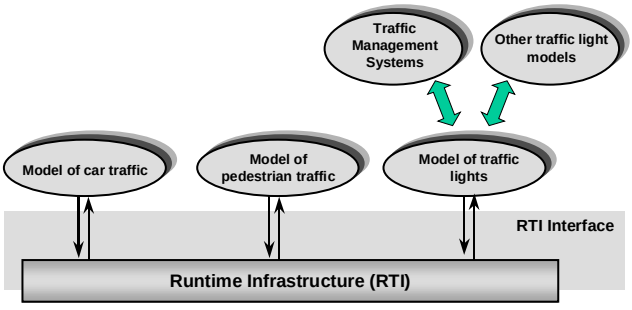
3 TOOLS AND THEIR NEW VERSIONS

In order to provide the functionality needed and, at the same time, promote the use of internet capabilities, simulation and animation tools are developed based on Java. One of these tools is Skopeo, an animation tool which reads trace files in a format compatible with Proof Animation. It was developed at the University of Magdeburg and runs within any Java-capable web browser (Ritter 1996; Lorenz and Ritter 1997; Dorwarth et. al. 1997). It is currently being extended to accept real-time input and to act as an HLA animation federate.

On the simulation side, a Java implementation of GPSS, JavaGPSS, was developed at the ISG. It accepts input in GPSS syntax which is then converted to Java source code. Compiled to Java bytecode, the simulation model could be run on any platform which a Java virtual machine can be run on. Furthermore, the performance can be improved considerably by on-site platform dependent just-in-time (JIT) compilation. Here, too, JavaGPSS will be extended to provide HLA functionality to the GPSS programmer as described below.

4 PROTOTYPES

A prototype traffic simulation model was taken as an example of how a model, built in a monolithic fashion, could be transitioned into a set of interacting submodels (Picture 4). This step-by-step process was documented and serves as an example for lectures on distributed simulation. The main aspect of this project was to examine different classical simulation tools for their suitability for distributed simulation based on HLA. While some tools have been assessed as well suited for this purpose (e.g. SLX), other tools have shown general problems (Proof, GPSS/H). Given these problems and the availability of implementations of compatible tools in Java, it is our intention to rather use these versions to integrate them into HLA.



Picture 4 Traffic Simulation Federation Example

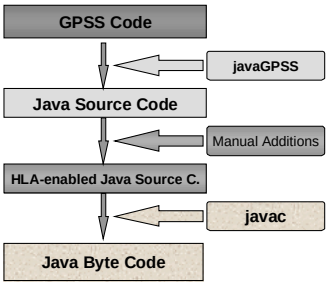
The final step of our project (the HLA compliant version of the traffic simulation) was started with the availability of the RTI software in late May 1997 using the tools mentioned above. At the time of writing, a version featuring 3 SLX models as federates has been completed. The SLX models can be distributed on any PC in the Internet running Windows 95 / NT (the current SLX version only supports these operating systems). The SLX simulator uses a wrapper library which not only translates between SLX and RTI function calls, but also simplifies the process of programming with the RTI. This is mainly done by hiding the complicated use of object handles and attribute handles / handle sets from the SLX programmer, since this can be quite cumbersome to control and deal with by a simulation specialist.

Given the topic of the conference, the remainder of this paper will focus only on the Internet / Java related parts of our project to bring HLA to classical simulation tools. The same model as described above is being used as a reference example for the JavaGPSS project.

5 BRINGING HLA TO GPSS

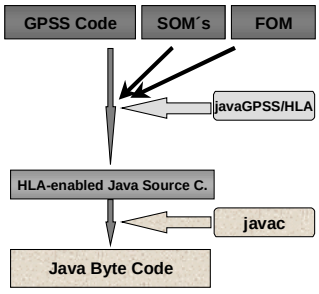
The transition of GPSS to JavaGPSS is described in the next two sections. Given the Java implementation, the incorporation of HLA capabilities into JavaGPSS was scheduled to have two stages:

1. Manual transformation of the standard JavaGPSS output to make it HLA-compliant (Picture 5)



Picture 5 First stage of HLA integration

2. Modification of the JavaGPSS-compiler to automatically generate HLA-compliant Java source code (with the use of additional information; see Picture 6).



Picture 6 Second stage of HLA integration

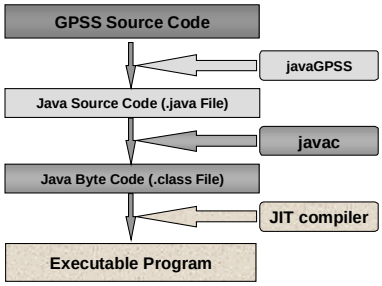
Since the Java-API and the release of the Java package to the RTI has been announced for late September 1997, we are currently concentrating on stage 1. The rest of this paper will focus on both the necessary enhancements that need to be implemented in this stage and on our concept for modifications of the actual JavaGPSS compiler.

5.1 Stand-alone GPSS/H

GPSS/H is a fast, stand-alone, discrete event simulation tool with a long tradition and is available on a variety of platforms (UNIX, OS/2, MS-DOS, etc.) (Schriber 1991). In GPSS/H, the dynamic elements (called transactions) are routed through blocks which describe static objects (e.g. logic switches, storages, facilities). Besides its simple file manipulation functionality, the coordination with external programs or simulations is quite limited. It is possible to include external functions and to call external programs; this does not allow the inclusion of the RTI library, though (Klein and Strassburger 1997).

5.2 JavaGPSS

The JavaGPSS compiler is a simulation tool which was designed for the Internet. The objective was to create a GPSS implementation which could truly be run as an applet in any Internet browser. Previous solutions developed at the University of Magdeburg had to use CGI scripts to transfer GPSS source code entered in an HTML form to a server. This server would than run the simulation with a commercial GPSS/H implementation and deliver the results back to the client. With the existence of JavaGPSS an interesting alternative has been developed. The JavaGPSS compiler is a Java program that translates GPSS source files to Java source code (Picture 7).



Picture 7 Code generation with JavaGPSS

Since Java applets cannot write files, the current version of JavaGPSS itself cannot be run as an applet. The output of JavaGPSS (the actual simulation) can be run in any Java-capable Internet browser, though.

The syntax of JavaGPSS is compatible with GPSS/H up to a very high percentage. There are only little modifications regarding obsolete block and control statements.

Future versions of JavaGPSS may consider the possibility of running the actual JavaGPSS compiler as an applet, possibly by rather interpreting the GPSS source code than generating new Java code.

5.3 HLA-enabled JavaGPSS

Our goal was to draft and implement a cross-compiler that transforms GPSS source code to HLA compliant Java source code. We intended to have as few modifications to the GPSS syntax as possible. This is done for the ease of the end-user and simulation developer, who has to be provided with an easy-to-use simulation language (as GPSS) and the possibility to "do" distributed simulation with a minimum of necessary knowledge of HLA, but without having to care about details in terms of synchronization, data exchange, federate ambassador, etc. One, though, has to care about some components that are required by HLA. One of these components, and thereby an integral part of simulation models participating in an HLA federation, is the Simulation Object Model (SOM) in conjunction with the Federation Object Model (FOM), which both document the Object Model in a way defined by the HLA OMT.

We were planning to use the existing JavaGPSS and enhance it in order to deliver HLA-compliant Java source code. To achieve this, the compiler was to use additional information specified in the object models mentioned above.

6 KEY ISSUES OF THE HLA-ENABLED JAVAGPSS

Generally speaking, there are two major areas to consider for the concept of the Java-GPSS cross-compiler to produce HLA-compliant output, which are the time management and the object and data distribution management.

The time management aspect concerns itself with the necessity of distributed simulations to coordinate the simulation clocks of the participating simulations. Our approach is going to use a conservative synchronization protocol with lookaheads.

The object and data distribution management has to establish rules for the representation of external data inside federates („ghosting“). It also has to organize publishing and subscribing to object classes and object attributes. Furthermore, the aspect of how to deal with interactions, a construct that does not exist in GPSS, has to be considered.

6.1 RTI Initialization

Additional code for the initialization of the RTI (create RTI ambassador) and for the implementation of the federate ambassador has to be added. This part has to also cover the setting of the initial time management parameters (time regulation, time constrained) and the publishing and subscribing of the object and attribute classes to be modeled. In order to have as few additions to the GPSS code as possible, it is necessary to derive the information about which object classes to publish and which to subscribe for each federate from the Simulation Object Model (SOM).

However, at this point in time, this information does not need to be documented to have a running HLA federation (although it must be specified in order to say that the federation is HLA compliant). The only thing needed for a running HLA federation is an .FED file specifying the federation execution data which represents, as a matter of fact, only the information in the Federation Object Model (FOM) and some information about transportation mechanisms.

To have subscription and publication services automated, we currently need an additional data source which will be implemented using a file. It is intended to use the DIF format which is the standard format for documenting the several object models required by HLA.

6.2 Time Management

There is a central point in the Java source code produced by the JavaGPSS compiler where the advancement of the simulator clock is performed. This routine is being enhanced to use the „NextEventRequest“- method of the RTI ambassador.

At the time being only a conservative time management protocol with lookahead is allowed. For a classical discrete event simulation language like GPSS a method for circumventing the necessity to specify a lookahead value strictly larger than zero would be desirable. However, this would require changes in the HLA concept/interface specification. (Fujimoto 1997). Further versions of the HLA specification may take these additions into account.

6.3 Data and Object Management

Objects that can be published and subscribed to in our first version will include a subset of the standard GPSS blocks (storage, facility, logic switch, amper variables). Transactions and certain other GPSS entities will be considered in a future version.

The registration of “HLA”-objects (the objects relevant to the federation execution and listed in the SOM) with

the RTI will be performed when instances of their JavaGPSS-counterparts are created (e.g. when the first definition/use of a logic switch occurs, it will be registered with the RTI). Any subsequent modifications of the object will be announced using the "UpdateAttributeValue" service of the RTI.

Once a certain object is registered with the RTI, and the first attribute update has been sent, any federate interested in objects of this class will receive a "Discover Object" notification. The receiving federate is going to create a local instance ("ghost") of this object (which will be a standard GPSS object) and store any subsequent attribute updates for this object instance in this object. Again, additional code has to be added to implement this concept.

For interactions and interaction classes, a construction which does not exist in GPSS, two different approaches are being considered. Taking into account that (at least for our reference examples) interactions are mainly used for telling remote objects to change an attribute value, the first and more simple approach is to have the JavaGPSS compiler generate an interaction when a federate tries to change an attribute value of an object it does not own (e.g. an object that it has subscribed to). The interaction is directed towards the owner object and contains the changed attribute value as a parameter. Although this does not use the whole flexibility of the interaction concept, it is very well suited for standard discrete event simulation purposes.

The second approach, which again will be considered by a future version, will offer all the flexibility of interactions. This advantage is coupled with the necessity to enhance the GPSS syntax, though.

7 CONCLUSION AND OUTLOOK

The main advantage of the introduced concept is that it brings classical simulation tools to HLA, whereas up to this point in time, most of the example simulations released by the U.S. DoD are written in C++, which may not be the most comfortable simulation tool for a simulation specialist.

Another advantage is that with the use of Java, the integration of simulation (in particular, distributed simulation) into the web can be lifted onto a new stage. However, the demonstrated approach can only be seen as one step towards a Web-Based Simulation Environment.

The approach outlined in this paper has been developed based on the provisional Java-API to the HLA RTI. Since this API and the necessary binaries to actually "code" a simulation are not officially released by the DMSO yet, it is at this point in time not possible to say when the actual implementation of this concept will be finished.

Further steps might include the consideration of CORBA as a possibility to bring a broader range of simulations and architectures to the Web (OMG 1997). This approach could use the strong correlation of Java and CORBA to produce synergy effects.

Application areas in urban traffic and emergency management (Williams 1996; Burmester 1996) will be addressed by forthcoming prototypes developed with web-enabled HLA compatible tools.

REFERENCES

Burmester, J. 1996. Plowshare Project. Available from http://www.stricom.army.mil/PRODUCTS/PLOWSHARES/.

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

Defense Modeling and Simulation Office (DMSO). 1997a. HLA Time Management Design Document, *Version 1.0, dated 15 August 1996*. Available online at the HLA Homepage (DMSO 1997).

Department of Defense (US). 1996. High Level Architecture Rules, Version 1.0, dated 15 August 1996. Available online at the HLA Homepage.

Department of Defense (US). 1997. High Level Architecture Interface Specification, Version 1.2 Draft 6, dated 1 August 1997. Available online at the HLA Homepage.

Department of Defense (US). 1997a. High Level Architecture Object Model Template, Version 1.1, dated 12 March 1997. Available online at the HLA Homepage.

Dorwarth, H.; P. Lorenz; K. C. Ritter; T. J. Schriber. 1997. Towards a Simulation and Animation Environment for the Web. Winter Simulation Conference WSC 1997. In preparation.

Fujimoto, R. M. 1997. Zero Lookahead and Repeatability in the High Level Architecture. Proceedings of the Spring 1997 Simulation Interoperability Workshop, March 3-7, Orlando. Paper No. SIW97S-046, available online at http://www.dmso.mil/projects/hla/papers/.

Henriksen, J. O. 1997. SLX and Proof Animation: Improved Integration of Simulation and Animation. In Deussen, O. and P. Lorenz (Ed.), Proceedings of the Simulation und Animation Conference Magdeburg, March 6.-7., 1997. SCS European Publishing House San Diego / Erlangen / Ghent / Budapest 1997, pp. 287-294.

Klein, U. and S. Straßburger. 1997. Die High Level Architecture (HLA): Anforderungen an interoperable und wiederverwendbare Simulationen am Beispiel von Verkehrs- und Infrastruktursimulationen (The High Level Architecture: Requirements of interoperable and reusable simulations by example of traffic and infrastructure simulations). Proceedings of the 11th Simulation Symposium ASIM 97 (Nov. 11-14), Dortmund, Germany. In preparation.

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 und Animation Conference Magdeburg, March 6-7, 1997. SCS European Publishing House San Diego / Erlangen / Ghent / Budapest 1997, pp. 12-23.

Object Management Group (OMG). 1997. Common Object Request Broker Architecture (CORBA) Homepage. URL http://www.omg.org.

Ritter, K. C. 1996. The Skopeo Animation System. Available online at http://simos2.cs.uni-magdeburg.de/Skopeo/Ani.html.

Williams, R. J. 1996. An Emergency Management Demonstrator Using The High Level Architecture. Proceedings of the European Simulation Symposium EES 1996. The Society for Computer Simulation International, October, 24-26 1996. Genoa, Italy.

Wolverine Software Corporation. 1992. Using Proof Animation.Wolverine Software Corporation.

Schriber, T. J. 1991. An Introduction to Simulation Using GPSS/H. John Wiley & Sons, New York.

AUTHOR BIOGRAPHIES

ULRICH KLEIN is a PhD candidate at the University of 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, Urban Infrastructure Management and Logistics, Geographic Information Systems, and the High Level Architecture.

STEFFEN STRASSBURGER is a Master's student at the Department of Simulation and Graphics, Faculty of Computer Science, Otto-von-Guericke University, Magdeburg. 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 the High Level Architecture.

JÜRGEN BEIKIRCH is a Masters student at the Department of Simulation and Graphics, Faculty of Computer Science, Otto-von-Guericke University, Magdeburg.