Proceedings of the 1998 Winter Simulation Conference
D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, eds.

TRAFFIC SIMULATION BASED ON THE HIGH LEVEL ARCHITECTURE

Ulrich Klein
Thomas Schulze
Steffen Straßburger

Department for Computer Science
Otto-von-Guericke-University
Universitätsplatz 2, D-39106 Magdeburg, GERMANY

ABSTRACT

Interoperability and reusability are features supported by the new High Level Architecture for Modeling and Simulation (HLA). While the traditional approach of monolithic traffic simulation modeling has proven to be successful, distributed traffic simulations gain more attention. The first part of the paper describes our work with distributed traffic simulation based on the High Level Architecture and the lessons learned from enhancing classic simulation and animation tools for HLA and our first HLA prototypes. The second part elaborates on the additional flexibility that architectures for distributed simulation offer, focussing on the dynamic integration of information relevant to the overall simulation into the dynamic event space. A promising outlook concludes the paper.

1 INTRODUCTION

Traffic simulation models are traditionally developed as monolithic, more or less stand-alone systems well suited for the actual needs. But what happens if these needs change, posing new requirements or new connectivity demands to other systems?

Interoperability and reusability are the catchwords for open systems which provide new flexibility in a modular way. The High Level Architecture is a promising, upcoming standard in distributed simulation. The first part of this paper deals with experiences with traffic simulation based on the HLA. Using the HLA as basis for distributed traffic simulation, new synergetic effects could be used in order to gain even more flexibility – the second part of the paper elaborates on that.

2 CLASSIFICATION ASPECTS

In order to illustrate the path to distributed traffic simulation, we use two property pairs:

- *Monolithic* vs. *distributed*: a *monolithic* model keeps all relevant information within one single model, whereas a *distributed* model is broken up into several submodels, each containing a part of the entire model. How a model is divided into submodels depends on several factors and is a major design decision; in the area of traffic simulation, models could be split up depending on geographic regions or traffic types.
- *Static* vs. *dynamic*: static simulation model elements do not change during the simulation model execution (refer to Figure 1) and are therefore constant. They may be hard-coded into the simulation model or read in during the initialization phase of the simulation model, but remain the same during the model execution. Dynamic simulation model elements have the potential to change their properties or attributes during the model execution. In models with static input parameters, the values for the input parameters are defined prior to runtime. The values of the infrastructures (e.g. street parameters) and traffic characteristics (e.g. intensity of traffic, schedules for traffic lights etc.) are mainly either hard-wired into the model or will be set during the initial phase of the simulation. Models with dynamic input parameters receive this information dynamically during runtime (from other sources or federates). One form of simulation based on dynamic simulation elements is interactive simulation where the user can make modifications during runtime.

Additionally, we consider two areas of information in which the data is prepared, collected, and managed for the simulation task:

- *Primary* information is the information simulations deal with at runtime, e.g. the simulation variables or

objects that are modeled by the simulation (the core). Primary information is of dynamic nature.

- *Secondary* information is information outside the simulation core, which is necessary to support the successful simulation and animation. Animation layouts or visualization information are examples for secondary data. This data, although not static by nature, is commonly used in a static manner and held in form of input files or simulation model code (hard-coded, e.g. the street network).

Note that the pairs *static/dynamic* and *primary/secondary* are similar but not equal: whereas static/dynamic denotes if, when, and how often a piece of information changes, the primary/secondary areas indicate to what extend the information belongs to the core (the second part of the paper is about making secondary data dynamic).

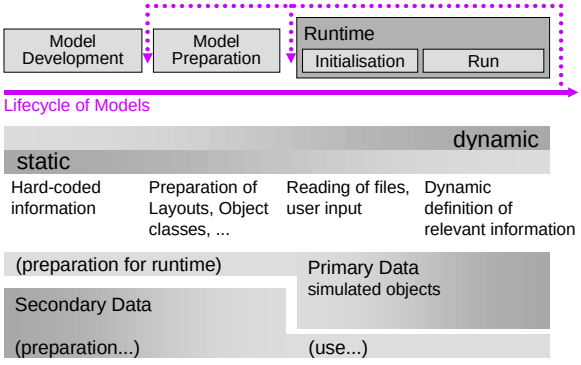

Figure 1: The Static / Dynamic Property of Primary and Secondary Simulation Model Elements During the Simulation Model Lifecycle

Given those properties, we could draw a time line:

- In the past, traditional traffic simulation modeling could be characterized as *monolithic* and *static* (refer to section 4);
- At present, distributed static models which focus on modularization of systems and reusability or interoperability are developed (refer to section 5);
- The next step in the future is the design of *distributed dynamic* traffic simulations that exploit the new flexibility given by the HLA (refer to section 6).

Two shifts are involved (Figure 2):

- from monolithic to distributed
- from static to dynamic (interactive) with the main example of shifting information from the secondary to the primary area

The High Level Architecture for Modeling and Simulation (HLA) supports the development of distributed, interoperable, and reusable simulations (shift 1) and is also used to evaluate the second shift. The next section, therefore, introduces this new architecture as a predecessor to three sections each characterizing the three steps mentioned above.

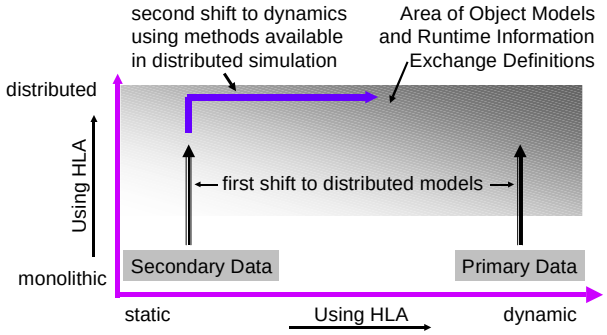

Figure 2: Paradigm Shift Towards Dynamics and Flexibility

Throughout the rest of the paper we illustrate those shifts using the following **traffic simulation example**:

Consider a microscopic traffic simulation model for an urban area. On a given street network with traffic lights etc., we focus on traffic patterns of individual car traffic and public transportation during rush hours.

We therefore model cars, buses, and pedestrians, which will be the active elements in our simulation (the primary data). In addition, we need logical, geometric, and visual information about the street network, the traffic lights, locations of sources and sinks (secondary data).

3 THE HIGH LEVEL ARCHITECTURE

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 unified approach, the *High Level Architecture* (*HLA*), which has the challenging vision to support its predecessors and different time regimes. The High Level Architecture is a forthcoming IEEE simulation interoperability standard currently being developed by the US Department of Defense (DMSO 1998).

The architecture is defined by :

- Rules which govern the behavior of the overall distributed simulation (*Federation*) and their members (*Federates*).
- 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 takes place only 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.
- An *Object Model Template* (*OMT*) that 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 HLA uses an object view on the "items" being simulated. It differs from standard OO nomenclature by treating dynamic, non-persistant object behavior (otherwise called methods) separately as interactions; while objects can have attributes, interactions can have parameters. 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).

In view of a completely defined federation development process (FEDEP), HLA is only the first of three parts of a common technical framework (CTF) so far established by DMSO. The second part consists of the data standardization to ease the exchange and the reusability of HLA object models. The third part covers the common semantics and syntax according to a domain of interest which is called the conceptual model of mission space (military terms), or CMMS.

Even though it originates from a military application surrounding, the architecture seems to be very well suited for civilian applications, too. The coupling of geographically, organizationally or otherwise distributed systems, simulations, viewers, information systems, etc. is a feature especially interesting for traffic management applications.

4 TRAFFIC SIMULATION – CLASSIC STYLE

Monolithic simulation models with *static* input parameters and *predefined* secondary data, like street networks, are commonly used in the area of traffic simulation. Referring to the classification above, these traffic simulation models can be classified with the labels *monolithic* and *static*: secondary data is prepared prior to runtime and is "read-only" for the simulation / animation. Models with these attributes are called classic style models.

Especially microscopic traffic simulation models are characterized by a high degree of detail. The monolithic models contain a lot of static components of the infrastructure (like lanes, intersections, traffic lights, etc.), of decision rules, and of dynamic entities like vehicles, pedestrians, and bicycles. The number of components and the number of relations between components lead to large scale models. Often microscopic traffic simulation models become too large, unwieldy, and difficult to maintain and adopt. These circumstances lead to:

- long run times (hours) for simulation runs
- long times for developing and testing of such monolithic models
- huge human effort for maintenance of models and for adapting the models for other purposes
- low flexibility and reusability

One solution is the employment of more powerful hardware. Another way is breaking up the model into a distributed set of submodels as described in section 5.

4.1 Monolithic Models

The use of new and higher powered hardware and the separation of large scale models into smaller submodels are possible solutions. Reducing the run times is a traditional goal for distributed traffic simulation (Mabry and Gaudiot 1994). Currently, there are new goals for distributed simulation: interoperability and reusability. In this paper it will be shown, that these new goals can be reached. It is time to disband the large old battleship, the monolithic models, into smaller active units. The aspects of interoperability and reusability become the main goals in using distributed simulation.

The next section focuses on distributed traffic simulation (see also section 5.1 for the discussion of distributed models).

4.2 Static Simulation Elements

The values of all input parameters for classic traffic simulation models are often hard-wired in the model code. The strong static for the input parameter can be weakened by using data driven models. The values for the input parameters will be set during the initialization phase of the simulation model. The Layout Based Model Generation (LBMG) method (Lorenz and Schulze 1995) uses this approach. The street network for the traffic models is created during the initialization phase, but the parameters are constant during the whole simulation. Changes in the infrastructure are predefined in the scenario parameters, e.g. close road A at time t1 for time t2. See also section 5.2 for the discussion of dynamic simulation elements.

Referring back to the **traffic simulation example** introduced at the end of section 2, here's how it could be implemented, classic style:

The microscopic traffic simulation model for an urban area is of monolithic type with hard-coded or constant street network. Input parameters would be defined in the initialization phase of the model. All additional geometric and visual information is prepared in a way suited for the animation tool used. The simulation model produces a trace file with animation commands. All but the simulated objects are of static nature during the simulation.

5 DISTRIBUTED TRAFFIC SIMULATION WITH THE HIGH LEVEL ARCHITECTURE

Distributed traffic simulation models are currently developed using the High Level Architecture. A set of interoperable federates cooperate and communicate via the HLA Run Time Infrastructure (RTI).

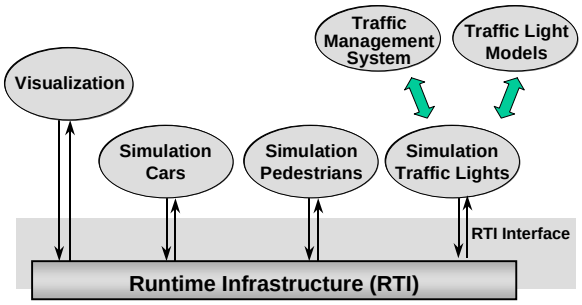

Figure 3: A Possible Structure for a Federation Designed with Strategy A

The question is how a monolithic traffic simulation task can be divided into different submodels. Different approaches exist depending on the system to be modeled and the tools already present; the two main approaches are

A) Every federate describes a different functional subsystem of the entire traffic system to be modeled. A traffic simulation could consist of federates for vehicle traffic, traffic light control, and railroad traffic. Figure 3 shows the possible structure of a federate for urban street traffic systems. An additional animation component is added.
B) Every federate describes a complex independent model for a small geographic region.

The development of distributed traffic simulation models based on HLA started in 1997 at the University of Magdeburg. Two prototypic federations will be described in section 5.4, which show different aspects of the goals reusability and interoperability.

5.1 Distributed Models

Moving away from monolithic simulation models (refer to section 4.1), the distribution of simulation and other model components may be focused on run time minimization, but also on interoperability and reusability. The High Level Architecture defines a framework that supports the development of interoperable and reusable simulation components.

Modular simulation models are easier to develop, test, and maintain than their monolithic counterparts and offer a high degree of flexibility when models have to be adopted, extended, or operated in a new environment.

5.2 Dynamic Simulation Elements

Within the HLA, each of the federates as well as the federation has to have an object model (SOMs and FOM, resp.) describing the objects and interactions the federates are able to talk about. Interactions are messages sent at a certain point of time, whereas objects are created and potentially modified (the object models describe this as update condition with values of static, conditional or periodic).

Those object models describe the dynamic content, which exists or is brought into the federation at runtime (during the federation execution). Nevertheless, the information which was coined secondary data in section 2 may remain outside the object models and is therefore only accessible for federates, which use it explicitly outside the HLA mechanisms of data exchange.

The main step in making simulation elements dynamic (depicted as shift 2 in figure 2) is described in section 6.

5.3 HLA-enabled Simulation and Animation Tools

In the military domain, the origin of the HLA technology, simulation models are commonly developed using languages like C++. In order to be HLA-compliant, potential federates need to link an RTI software library, which is a standard procedure if you use C++.

In the civil simulation domain, we face a different situation: simulation and animation tools and environments are commonplace, built for autonomous stand-alone use.

As a first step we therefore evaluated the process of integrating simulation and animation tools with the HLA and selected two tools, which were then extended for HLA compatibility:

- The simulation tool SLX from Wolverine Software Corporation (Henriksen 1997) is based on an open architecture. The used version 1.0 was not designed for distributed simulation. The use of the wrapper library (Straßburger 1998) enhanced the standard SLX for operating in the HLA community.
- The animation tool Skopeo (Ritter 1998) started as a java-based Proof Animation compatible 2D animation tool and was recently enhanced for 3D visualization using VRML 2.0. It was successfully extended as an HLA federate and is now able to visualize online what happens in federation executions. Figure 4 shows a scene of the urban traffic federation prototype described in section 5.4.1.

5.4 Prototypes

The following prototypes have successfully shown different types of reusability and interoperability.

5.4.1 Prototype simple urban traffic

This federation consists of four different independent federates. Every federate can run under stand-alone condition. There are three simulation federates and one observation federate. The structure of this federation is shown in figure 3.

The traffic model was divided into three simulation federates: car traffic, pedestrian traffic and traffic light

control. Every simulation federate was implemented in the SLX simulation language using the library for the SLX-HLA interface (Straßburger 1998). The car traffic federate simulates cars driving along a road; a pedestrian traffic light is installed in the middle of the road. The traffic light is used by pedestrians who want to cross the road; they will only cross the street if the traffic light shows green for them. The pedestrian stream and the pedestrian behavior are modeled in the federate pedestrian traffic. If a pedestrian arrives on the path and the traffic light is red, the pedestrian will push the traffic light button for the green light, triggering the next traffic light cycle. The traffic light control is simulated by the traffic light control federate. The animation of the parallel traffic processes is realized by the observer federate. The observer is implemented with Skopeo. Figure 4 shows a screen shot of the Skopeo animation system showing a traffic federation execution status.

HLA allows the use of a transparent time management between the federates. The switches with which the federate time management behavior is characterized are

- the capability to react on time constraints from other federates (*time constrained*) and
- the capability to regulate the time of other federates (*time regulating*).

The parameters of the switches are shown in Table 1. All simulations have to synchronize their local time advances in order to assure causality, so the *time constrained* and the *time regulating* switches are set.

Table 1: Overview of Federates and Their Time Management Characteristics

Federate	Time constrained	Time regulating	Time advance
Car traffic	Yes	Yes	Event oriented
Pedestrian	Yes	Yes	Event oriented
Traffic light control	Yes	Yes	Event oriented
Observer	Yes	Yes / No	Scaled real time

A passive observer federate has to listen to the simulation federates. It can make time jumps in the visualization depending on the time advances in the simulation federates. The updates and interactions have to be processed in causal order; since the observer federate doesn't generate events on its own, it has no influence on the time advance of the simulation will be set. federates. Therefore only the *time constrained* switch will be set. As the visualization federate operates in passive mode (it does not send messages back to the federation), the *time regulating* switch can be switched off.

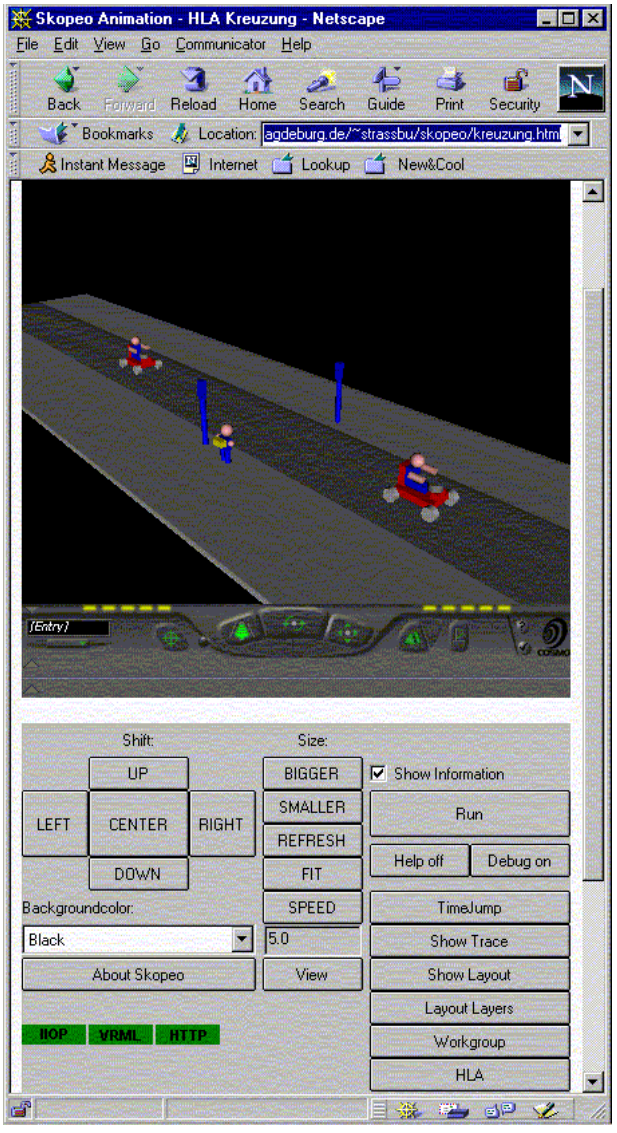

Figure 4: Skopeo 3D Online Visualization of the Current Federation Status via VRML 2.0

Depending on the visualization requirements, the observer federate can operate in an active manner. If the observer works as a scaled real-time federate, the observer influences the federation time advancement.

Table 2 shows the object classes of the federation object model (FOM). In addition to the standard FOM representation, the table shows which federate will publish the values for a given object class and which federates will subscribe to attribute changes for this class.

Table 2: Objects and Their Attributes, Publishers (Owners) and Subscribers

Object Class	Attribute	Published by	Subscribed by
TrafficLight	PedLight	Traffic light control	Pedestrian Observer
	CarLight		Car Traffic Observer
Car	...	Car traffic	Observer
Pedestrians	...	Pedestrian	Observer

If an arriving pedestrian has pushed the traffic light button, an instance of an interaction class will be sent by the pedestrian traffic federate. Only the traffic light control federate receives this interaction

The following goals have been determined for the development of this federation prototype:

- evaluation of the interoperability between event oriented simulation federates
- evaluation of the interoperability between simulation and animation federates
- evaluation of the RTI time management services for event-oriented simulation federates, using conservative time advancement approaches
- evaluation of the RTI time management services for the synchronization of local time advances between event-oriented and scaled real-time federations
- provision of a discrete event-oriented test bed for the HLA enabled Skopeo animation system

Referring again to the **traffic simulation example** introduced at the end of section 2, here's how it could be implemented as a distributed HLA federation:

The microscopic urban traffic simulation model could be split up like the traffic simulation prototype described above. Using the HLA object models, object updates and interactions are communicated during runtime of the federation. User interaction could be involved by including a human-in-the-loop federate. Input parameters would still be defined in the initialization phase of the model, and all additional geometric and visual information is prepared in a way suited for the animation tool used.

5.4.2 Distributed Driving Simulation

In order to evaluate the interoperability and federation development aspect further, a distributed driving simulation project was designed as a joint effort of the Institute for Simulation and Graphics (ISG) at the University of Magdeburg and the Competence Center Informatik GmbH (CCI). 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) from the partners. Both underlying simulation models had to be extended for:

- communication with the RTI,
- interoperability between the models,
- and synchronization of local time advancement.

The participating federates are:

- Federate **driving simulator**: The basis for this federate is a real-time driving simulator for off-road driving within a synthetic environment. There was no street traffic modeled. The simulator is a C++ program for UNIX machines and offers a 3D visualization. In the original military context it is used for tank driving simulation. This federate was developed by CCI.
- Federate **traffic simulator**: This federate is based on a discrete event traffic simulation model for urban street traffic which focuses on the psycho-physical behavior of street traffic flow (Schulze and Fliess 1997). The simulation is implemented with the SLX simulation environment running on Win32 standard PCs. This federate models the street traffic. The simulated cars of this federate have to react to the vehicle modeled by the driving simulator. The federate traffic simulator was developed by the ISG.
- Federate **observer**: This viewer federate is based on the 2D / 3D java-based visualization system Skopeo. It runs as an applet in any java-capable web-browser. This federate was developed by the ISG.

Figure 5 illustrates the federation and the common event space in which the federates operate.

The two simulation federates had to expand their old functionality. The driving simulator federate now has to visualize the moving cars on the road. Figure 6 shows a view from above the off-road driving truck. The human driver only has to react to the road cars if he influences the street traffic. At the same time, the traffic simulation models have to react to the external human driver. Both federates are operating in new common event space. They have to coordinate their behavior in space and time. The federates can be networked based on LAN, dial-up ISDN links, or the Internet.

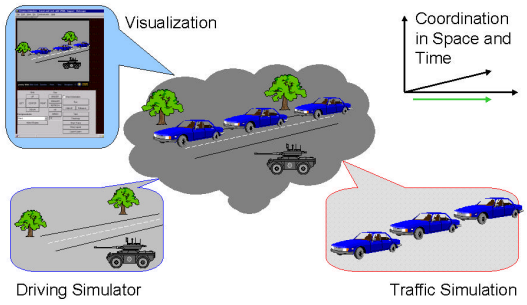

Figure 5: The Objects Each of the Federates is Modeling and How They are Combined in the Common Event Space of the Federation Execution

After definition of the underlying federation object model (the *Distributed Driving FOM*), the federates were developed completely independently from each other at the ISG and CCI sites. The time management characteristics of the Distributed Driving FOM are depicted in table 3; see also table 1 for the corresponding characteristics of the simple urban traffic simulation.

Table 3: Time Management Characteristics

Federate	Time constrained	Time regulating	Time advance
Driving Simulator	No	Yes	Real-time
Traffic Simulation	Yes	Yes	Event oriented
Observer	Yes	No	Scaled real-time

The FOM, as a first stage in the federation development process (FEDEP), equals the SOM for both the traffic and the virtual driver. Only a small number of functions from the RTI 1.0-3 have been used so far to support the SOM, which itself is based on the well known Real-time Platform Reference-FOM (RPR FOM 1998). These functions stem from the essential object-declaration, object-management, and time-management services of the RTI. Additionally, an interaction class, Collision, has been introduced.

It is worth noting that the driver software-architecture divides the application into two almost independent domains, i. e. the virtual simulator (driver) and its HLA interface. An object-oriented design promotes highly flexible structures to keep even the first prototypes as a base for future extensions and/or improvements. The method of choice is to introduce an
ntra-SOM-layer (ISOM) separating the interface from the simulator and, at the same time, being the internal representation of the SOM (without referring to any HLA interface functions of the RTI). This allows both the interface and the simulator to access a shared memory portion of the ISOM-layer based on commonly agreed ISOM classes (Menzler 1998):

The virtual driver was developed in C++, based on SGI-Performer 2.2. The ISOM-layer consists mainly of object references, which mimic the above-mentioned read and write memory shared by the HLA-interface and the driver. Several timeout events, which were used to “tick” the RTI or to dead reckon entities, use the X- Window’s event mechanisms. They are merged together with the Performer’s high frequency frame-rate events for updating the virtual environment and the driver’s state.

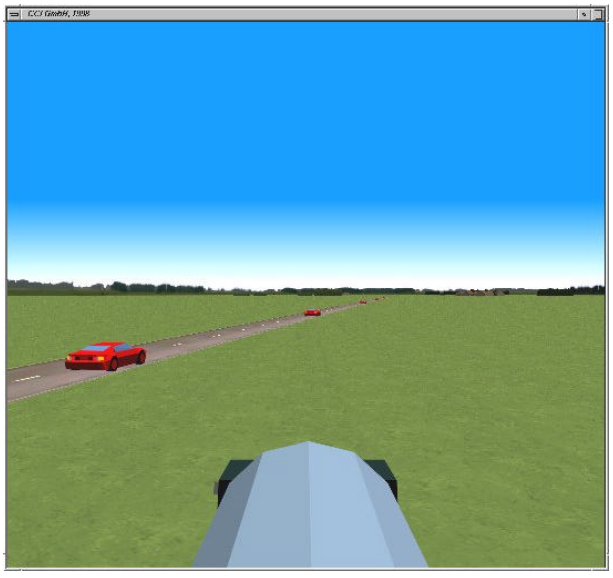

Figure 6: View From Above the Truck Driving Off-road; the Red Ferraris are Modeled by the Traffic Simulation. Visualization by the Driving Simulator

The Driving Simulation was to show that independent development of federates is possible. In addition, the following highlights underpin the successful approach of HLA to interoperability:

- time management interoperability (real-time vs. discrete, event-oriented vs. scaled real-time)
- platform interoperability (SGI vs. Windows PCs vs. platform-independent Java application)
- language interoperability (C++ vs. SLX vs. Java)
- independent federate development based on a common FOM definition
- network interoperability (LAN vs. ISDN-Link vs. Internet)
- real-time testbed for the HLA-enabled Skopeo animation system

6 DISTRIBUTED TRAFFIC SIMULATION USING DYNAMIC ENVIRONMENTS

Distributed simulation models with *dynamic environments* are currently designed in the area of traffic simulation using the High Level Architecture. Based on the object model technique of HLA, a method for the description of the common event space in which the federates act, is now available.

Standalone applications used static secondary data together with dynamic primary data to execute the simulation. Now that simulation is a cooperative effort under HLA, it has to be evaluated which (formerly static) secondary information has to be provided to the other

federates. This can be done via the integration into the object models:

- information can be provided to other federates in the form of objects and interactions
- this information can be static (as it was before) and dynamic (new possibilities)
- objects are able to interact

For a first evaluation, the following scenarios illustrate the new possibilities:

- traffic simulations can integrate their traffic networks by themselves or specialized federates
- the traffic network itself is now dynamic (e.g. dynamic road closure or road attrition)
- by adding visualization attributes to objects visualization federates can obtain their information dynamically.

The same applies to other information of the (synthetic) environment.

6.1 Making Secondary Information Dynamic

The advantages of putting secondary data into the dynamic area, under HLA represented by the Federation Object Model, are obvious.

A disadvantage is a longer initialization phase of a federation; all communication is now done via the RTI mechanisms. Additional communication caused by updates on previously static data has to be excluded.

Using the distributed driving simulation as an example, an interaction between an entity - such as the driver - and the virtual terrain is a highly sophisticated problem for interoperability compared to an entity-entity interaction, which itself is a well defined class in the HLA OMT. Dynamic terrain requires concepts beyond today's scope of HLA. The reason behind this is that in HLA, entities must be published across the federation by using the RTI, whereas environmental data stays private for a federate. Even though SEDRIS (SEDRIS 1998) is an upcoming solution for a DIF on static environmental data, it will be usable only for an interchange of that data between two different federate's visual systems, prior to the running federation.

In order to manage the problem - e. g. to let a driver collide with a traffic light - one has to go further. Currently, we are looking for a solution based on VRML, the virtual reality modeling language, to allow an exchange of environmental information. Any locally detectable terrain element has to have a VRML-representation string identifier, which in fact is possible in SGI's Performer extension Open-Inventor. In order to keep the environmental states consistent across a federation, a detecting federate sends an entity/VRML-interaction to the federation. From this, each VRML-compliant federate is able to keep track of environmental changes.

6.2 Geographic Information Federate

Traffic is a phenomenon based on physical (or otherwise) movement in time between geographically distinct locations. The infrastructures on which traffic takes place are themselves geographically referenced as they have a location, dimensions, etc.

In order to deal with these and other geographical information in an efficient and powerful way, geographical information systems (GIS) do exist. Why don't we transfer the task of geographic information management to a specialized HLA federate?

That's why are we evaluating the potential role of GIS federates in HLA federations in general, and for the traffic simulation domain in particular.

- **Passive Role**
  The GIS prototype currently under development provides the other federates with the necessary geographic street network and visualization information. The GIS Federate shows a map-based visualization of the current federation status.
- **Active Role**
  Additionally, the GIS federate can play an active role: the user is able to manage objects published by the it online by creating new or deleting objects or by altering their state using the functionality provided by the GIS.

7 OUTLOOK

The High Level Architecture is the currently most promising technology for interoperable, reusable networked simulation and other software. Based on HLA, we have successfully developed distributed-traffic simulations, which offer a new flexibility and advantages over common monolithic simulations. Federates operate in a common event space with coordinated time advance; by putting more, previously static information into this space, additional flexibility can be exploited as depicted in the second part of the paper. The next steps will be the integration of street network information and visualization information into the federation object model and the development of an HLA-enabled geographic information system. Furthermore, dynamic configuration of federation executions during runtime like federate substitution or federation cloning will be evaluated.

REFERENCES

Defense Modeling and Simulation Office (DMSO). 1998. The High Level Architecture Homepage. Online available at http://hla.dmso.mil.

Henriksen, J. O. 1997. An Introduction to SLX. In *Proceedings of the 1997 Winter Simulation Conference*, ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, pages 559-566.

Lorenz, P. and Th. Schulze, 1995. Layout Based Model Generation. In *Proceedings of the Winter Simulation Conference* 1995 (ed.) C. Alexopoulus, K. Kang, R. Lilegdon, and D. Goldsman, pp. 728-735

Mabry, Susan L., Jean-Luc Gaudiot. Distributed parallel object-oriented environment for traffic simulation (POETS). In: *Proceedings of the 1994 Winter Simulation Conference*, ed. J. D. Tew, S. Manivannan, D. A. Sodowski, and A. F. Seila, pages 1093-1100.

Menzler, P. 1998. Merging HLA with a Virtual Simulator. Paper to be presented at the I/ITSEC 1998.

Real-time Platform Reference Federation Object Model (RPR FOM). 1998. Documentation available online at http://siso.sc.ist.ucf.edu/docref/rpr-fom/.

Ritter, K. C. 1998. The Skopeo Animation System. Available online at http://simos2.cs.uni-magdeburg.de/Skopeo/Ani.html.

Ritter, K. C., U. Klein, S. Straßburger and M. Diessner. 1998. Web-based animation of distributed simulations using the High Level Architecture (german). In *Proceedings of the Simulation & Visualization Conference 1998* Magdeburg, ed. P. Lorenz, and B. Preim, March 5-6, 1998. SCS Europe, pages 41-52.

Schulze, T. and T. Fliess. Urban Traffic Simulation with psycho-physical vehicle-following models. In *Proceedings of the 1997 Winter Simulation Conference*, ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, pages 1222-1229.

Straßburger, S. 1998. Integration of classic simulation tools into the High Level Architecture. Master's Thesis, Otto-von-Guericke-University Magdeburg.

Synthetic Environment Data Representation & Interchange Specification (SEDRIS) Homepage. 1998. Available online at http://www.sedris.org.

AUTHOR BIOGRAPHIES

ULRICH KLEIN is a Ph.D. 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.

HANS-PETER MENZLER studied theoretical physics at the University of Osnabrück, Germany, and received his Ph.D. in mathematical and numerical studies on optical waveguides. He worked as a scientist at the Max-Planck-Institute for Plasmaphysics, Garching and became a system engineer at Competence Center Informatik GmbH (CCI), Meppen, Germany, in 1992. Since 1996 he has been working in the training and simulation department of CCI. His work focuses on the object-oriented development of interactive graphical user interfaces and planning algorithms in manufacturing and emergency management areas.

THOMAS SCHULZE is an Associate Professor at the Otto-von-Guericke-University Magdeburg in Department of Computer Science. 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 simulation in Germany.

STEFFEN STRASSBURGER is a Ph.D. student at the Simulation and Graphics Department in the Department of Computer Science, 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 interests lie in distributed simulation and the High Level Architecture.