CONNECTING HIGH LEVEL DISTRIBUTED SIMULATION ARCHITECTURES: AN APPROACH FOR A FAMAS-HLA BRIDGE

Csaba Attila Boer
Erasmus University Rotterdam
Faculty of Economics
Department of Computer Science
P.O. Box 1738, 3000 DR Rotterdam
The Netherlands
acboer@few.eur.nl

Alexander Verbraeck
Delft University of Technology
Faculty of Technology, Policy and Management
System Engineering Department
P.O. Box 5105, 2600 GA Delft
The Netherlands
a.verbraeck@tbm.tudelft.nl

KEYWORDS

Distributed Simulation, Architecture, Hierachical, HLA.

ABSTRACT

The distributed simulation architectures support the execution of simulation process in a distributed way by connecting different distributed simulation components of various functional areas. These simulation components collaborate and communicate in order to realize the functionality of the system as a whole. One of the well-known architectures is the High Level Architecture (HLA) which is a standard for modeling and simulation activities in the Department of Defense in the United States. Due to the fact that we faced with some inconveniences by using HLA for logistics area, a new distributed simulation architecture, called FAMAS Backbone Architecture, was developed. The FAMAS Backbone Architecture aims to model complex harbor simulation models in a distributed way. Although, the first version of FAMAS Backbone Architecture does not allow to connect the HLA compliant models, the next version should allow HLA compliance in order to exploit maximally the reusability feature. In order to enable to couple HLA models to FAMAS Backbone Architecture we have to combine these two architectures. The combination of these two architectures leads to a multi-level hierarchical design and development technology. The article introduces a new concept, called HLA-FAMAS bridge, for coupling these two distributed architectures.

1 INTRODUCTION

Distributed simulation refers to technologies that enable a simulation program to be executed on multiple geographically distributed computing systems (set of computers), interconnected by a communication network (Fujimoto 2000). Executing a simulation program in a distributed way leads to the following benefits:

1. *Reduced execution time*. By subdividing a large simulation computation into many sub-computations, and executing the sub computations concurrently across different computers, it can reduce the computation time up to a factor of ten (Fujimoto 2000).
2. *Integrating simulators that execute on machines from different manufacturers*. In a distributed system several components can be designed and developed simultaneously by different groups (manufacturers) in different simulation packages. Rather than porting these program to a single computer, it may be more cost effective to "hook together" the existing components (simulators), each executing on a different computer, to create a new virtual environment (Fujimoto 2000). What needs to be solved is the interaction between different components.
3. *Package independence*. In a distributed simulation execution the components which are part of the whole simulation study can be developed in any simulation environment.
4. *Reusability of existing models*. If a component that solves a problem already exists, we do not need to build it again, as it can be connected to the new system regardless of the environment in which it was developed.
5. *Flexibility in extension and maintenance*. As all components are responsible for a well-defined task, when making modifications it is enough to consider the structure and mechanism of the correspondent component. If multi level hierarchy is achieved components could be easier refined and improved.
6. *Fault tolerance*. Another benefit of applying distributed simulation is to increase the tolerance of failures. If one component goes down, it may be possible to other components to pick up the work of the failed machine, allowing of the simulation computation to proceed despite the failure. By contrast, if the simulation is developed as a single model, failure of that model means the entire simulation must stop (Fujimoto 2000).

In order to achieve the interoperability between the distributed simulation components different architec-

tures were proposed. One of the well-known distributed simulation architecture, called High Level Architecture (HLA), was developed by the Department of Defense in the United States. HLA was developed in order to support the simulation models in the military sector.

Another distributed simulation architecture is the FAMAS Simulation Backbone Architecture, which was developed within the FAMAS (First All Modes All Sizes) research program (FAMAS MV2 Backbone Project 2001). This Backbone Architecture supports the interoperability of different simulation models. Next to provide tools for designing the container terminals of the future for the Port of Rotterdam, the aim of this project is to be as generic as possible (especially for logistics problems). The functionalities that must be provided by the architecture end up in a generic framework that may be considered for the future generation of distributed simulation architecture (Boer et al. 2002b). The required functionalities of the FAMAS Simulation Backbone Architecture are: distributed execution, optimal communication, multiple testing possibilities, package independency, structure transparency and a hierarchical structure. Although, HLA is not fully suitable to achieve all these functionalities, it is still an IEEE/DoD standard for distributed simulation in the military field and a standard with growing importance in the civil domain (Schulze et al. 1999). Therefore we expect some simulation languages to become HLA compliant. The aim of this paper is to compare the HLA and FAMAS Backbone architectures and to give a possible method to couple HLA compliant models to the FAMAS Simulation Backbone Architecture by offering a so-called FAMAS-HLA bridge.

The rest of this paper is organized as follows. Section 2 briefly describes the structure of the High Level Architecture and FAMAS Simulation Backbone Architecture. Section 3 provides a comprehensive comparison of FAMAS Backbone Architecture and High Level Architecture. In section 4, an approach is introduced to couple HLA compliant models to the FAMAS architecture. Concluding remarks and directions for further research are given in section 5.

2 BRIEF INTRODUCTION OF HLA AND FAMAS BACKBONE ARCHITECTURES

2.1 High Level Architecture

The earliest researches and works on distributed simulation can be found in military applications. Geographically distributed simulators from different types of forces were connected to form full battle situations. In the early ’80ies the SIMNET (SIMulator NETworking) project had started in order to develop distributed simulation for virtual environments for the military sector. This project was sponsored by DARPA (Defense Advanced Research Projects Agency) in the United States. Training exercises have carried out in order to demonstrate the feasibility of interconnection of autonomous simulators. Later the SIMNET was replaced by DIS (Distributed Interactive Simulation) (DIS Steering Committee 1994) where standards were defined to support interoperability among autonomous training simulators in a geographically distributed simulation environment (Fujimoto 2000). The last evolution on this direction is the HLA (High Level Architecture) that became a standard architecture for modeling and simulation activities in the Department of Defense in the United States (Defense Modeling and Simulation Office 1996).

There are two important concepts introduced in HLA, namely the *federate* and the *federation*. Whereas the federate is an individual simulator, the federation is the collection of federates (in other words the federation is the whole distributed simulation in HLA). The federate can be not only a computer simulation but also an instrumented physical device or a passive data viewer. (Fujimoto 2000) (see Figure 1).

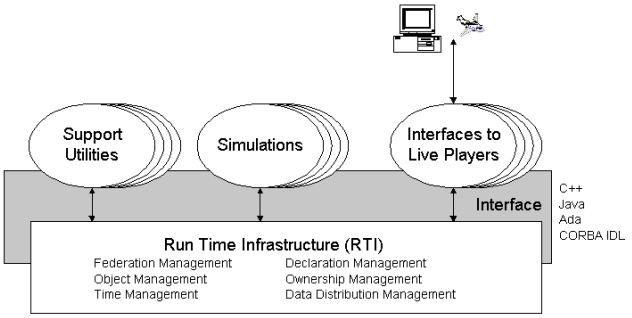

Figure 1: The structure of the High Level Architecture (Defense Modeling and Simulation Office 1996)

The HLA is defined by three components (Defense Modeling and Simulation Office 1996):

- *Federation Rules* ensures proper interaction of simulations in federation and describes the simulation and federate responsibilities.
- *HLA Interface Specification* defines Run-Time Infrastructure (RTI) services and identifies “callback” functions that each federate must provide.
- *Object Model Template (OMT)* provides a common method for recording information and establishes the format of key models (Federation Object Model, Simulation Object Model and Management Object Model)

2.2 The FAMAS Simulation Backbone Architecture

The FAMAS Simulation Backbone Architecture is represented by technical and functional components. Whereas the functional components represent the simulation models themselves, the technical components provide common tasks used by the functional components. The functional components can be simulation models, control programs, real equipments (e.g. Automated Guided Vehicle (AGV)), etc. (Boer et al. 2002b).

In Figure 2 we give a clear picture of the separately defined functional and technical components. There are five well defined subsystems, namely the Run Control Subsystem, the Backbone Time Manager Subsystem, the Logging Subsystem and the Visualization Subsystem (Boer et al. 2002a), (Veeke et al. 2002). The overall system consisting of all technical and functional subsystem is sometimes called a *federation*, where the subsystems that connect to the backbone are the *federates.*

The functionalities of the technical subsystems are the followings:

- *Run Control* for overall control of experiments; it starts, stops and periodically monitors the simulation process
- *Backbone Time Manager (BBTM)* for synchronizing the simulation time among different simulation subsystems
- *Logging* for collecting logging information from the distributed functional and technical components into a central database
- *Visualization* for providing separated or common visualization views for the subsystems or the entire simulation
- *Scenario* for completely defining a simulation run of a distributed model

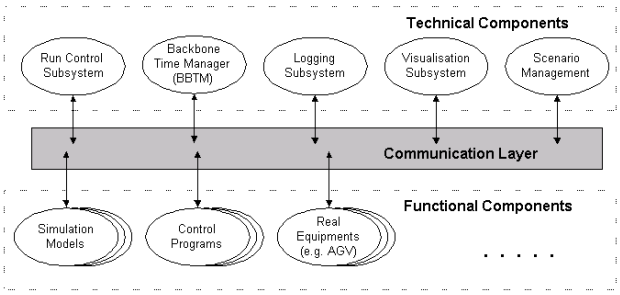

Figure 2: The structure of the FAMAS Simulation Backbone Architecture

3 A COMPARISON OF FAMAS AND HLA ARCHITECTURES

In order to give a comparative study of the FAMAS Backbone Architecture and HLA (see Table 1) we take into account two important aspects: the flexibility and the compatibility of the architectures. The flexibility refers to the way how easy is to initialize the architecture, to couple components to the architecture, the communication of components, to extend and to maintain the simulation components, the reusability of the components, etc.

Table 1: Comparison of HLA and FAMAS Backbone

	HLA	FAMAS Backbone
Separate simulation and communication	✔	✔
Facilitates construction and destruction of federations	✔	✔
Provides efficient communications	✔	✔
Supports an easy initialization	✔	✔
Support conservative and real time synchronization	✔	✔
Support object declaration and management between federates	✔	≈
Support optimistic time synchronization	✔	✘
Uses standardized RTI	✔	✘
Support built-in logging possibilities	✘	✔
Support built-in animation possibilities	✘	✔
Compatibility and Support for Civil Domain	✔	✔

The *flexibility* of the architecture is a required functionality during the entire simulation experiment. Therefore we examine the grade of the flexibility of the architectures in different phases and from different aspects, such as: in the initialization phase of the architecture, the interoperability of the architecture during the simulation run, the time management mechanism implemented in the architecture and some extended capabilities offered by the architecture.

Compatibility of an architecture refers to its ability to communicate and collaborate with other systems or

architectures. In the design and development phase is important to define the difficulty of developing simulation model in order to be HLA or FAMAS compliant. The architectures are required to support the coupling to different simulation packages, to various programming languages or even to other simulation architectures. Coupling may be possible both before and during the simulation run.

3.1 Separate Simulation and Communication

Both the HLA and the FAMAS Backbone architecture separate the communication and the simulation. Whereas for the communication the HLA uses the standard RTI, the FAMAS uses its backbone architecture.

3.2 Provides Efficient Communications

In the FAMAS backbone system all federates (technical and functional) that form a federation communicate by means of *messages*. The messages are sent and received by the standard socket mechanism with the TCP/IP protocol. The backbone works as a multiple client-server model; each subsystem is a client of all the other subsystems, but each subsystem is also a server, which can be addressed by all the other subsystems. In this way direct communication between subsystems is supported which minimizes communication over the backbone compared to sending all information through a central component (e.g. Run Control), which would act in this case as a gigantic switching board (Boer et al. 2002b).

The HLA Interface Specification is implemented by the Run-Time Infrastructure (RTI). This identifies how federates will interact with the federation and with each other. RTI provides services in a manner that is comparable to the way a distributed operating system provides services to applications. The communication between a federate and the RTI is realized using the RTI ambassador paradigm. The ambassador is nothing else just an object and the communication is performed by calling methods of these objects. Therefore the services defined in the interface specification can belong either to RTI ambassador or federate ambassador. HLA Interface Specification defines six service categories: Federation Management, Declaration Management, Object Management, Ownership Management, Time Management and Data Distribution Management (Defense Modeling and Simulation Office 1996).

3.3 Supports an Easy Initialization

The initialization of the architectures are quite simple both in FAMAS and in HLA. In FAMAS simulation backbone architecture there is a Run Control subsystem which is one of the core elements of this architecture (Boer et al. 2002a). It is the component that starts, monitors and stops the simulation run. It is also responsible to join and resign the simulation components (federates). In order to start a simulation run we must start the Run Control subsystem and indicate which scenario we will run through a Scenario Object. The Scenario Object contains the values of important parameters of individual simulations, or common parameters that are shared among different subsystems. The Run Control checks periodically all the subsystems in order to run the simulation in the correct way. The Run Control is globally achievable by all the subsystems that plan to join the simulation.

In HLA the RTI, which is comprised by RTI Executive Process (RtiExec), the Federation Executive Process (FedExec) and the libRTI library, manages the creation and destruction of federation executions (Defense Modeling and Simulation Office 1996). Like the Run Control in FAMAS backbone, the RtiExec is also a globally known process. Each application communicates with RtiExec to initialize RTI components. The RtiExec's primary purpose is to manage the creation and destruction of FedExecs. RtiExec ensures that each FedExec has a unique name. The FedExec manages a federation. It allows federates to join and resign, and facilitates data exchange between participating federates. It is always created by the first federate that joined the federation. The libRTI library extends RTI services to federate developers (Defense Modeling and Simulation Office 1996).

The initialization of the FAMAS Backbone Architecture slightly differ from the initialization of HLA. Due to the FAMAS Backbone Architecture is based on low-level communication (WinSock) messages the centralized Run Control subsystem is characterized by an IP address and a port number. These extra information that are necessary for the FAMAS Backbone Architecture as a difference compared to HLA is infinitesimal. This can not be considered as a constraint. On the other hand the Run Control offers a well designed user interface where some capabilities for logging the connections and the frequency of communication with subsystems that communicates with Run Control are displayed.

3.4 Support Object Declaration and Management Between Federates

An important aspect of the interoperability is the specification of the data that will be shared (transferred) from one federate to another. Furthermore, it is also refers to the way communications function between different federates.

In order to define a simulation run the FAMAS system uses scenarios. The description of the parts of scenario objects, namely Scenario Data, Initialization Script and Scenario Script, can be found in (Boer et al. 2002a). The scenario data defines the values of parameters for the simulation run, the initialization script defines the set-up of a simulation run and the scenario script defines how the simulation process is executed. The Run Control subsys-

tem interprets and executes these scenario objects. The Scenario Creator is a separate tool that enables to create, store and retrieve simulation scenarios.

In HLA a kind of object-oriented description (only attributes, without methods) is used in order to specify the data that the federate is going to share. A federate can use its object classes in order to describe its attributes and can use its interaction classes to describe the relation between different object classes. All the objects and interactions managed by a federate, and visible outside the federate, are described according to the standard Object Model Template (OMT) (Defense Modeling and Simulation Office 1996). The Simulation Object Model (SOM) of a federate defines the type of data is expected from and provided to other federates. The Federation Object Model (FOM) specifies the common objects used by simulators participating in a federation execution. The HLA offers different existing tools (e.g. Object Model Development Tool by AEgis) for the purpose of developing object models. Using these tools it is possible to generate Federation Execution Data files required by the RTI.

There are some significant differences between the OMT offered by HLA and the Scenario Object offered by the FAMAS Backbone Architecture. In FAMAS Scenario Object we can specify the technical and functional subsystems and their local and global (that can be used by many subsystems) variables. Furthermore the variables can be defined as static or dynamic. A static variable can not be changed during the simulation run, while a dynamic variable can be updated during the run, allowing other models to be informed about the new value. Taking a closer look to OMT we can state as a well defined description for information sharing. It follows a kind of object oriented description, where we can specify the shared objects and the interactions as well. Although compared to HLA the FAMAS does not specify any interactions between the subsystems, there is a special space called Scenario Script section, where we can specify some processes that must be executed during the simulation run. In the Scenario Script we can specify some events in simulation time that the Run Control subsystem must execute.

3.5 The Time Management Mechanism of the Architectures

The Time Management plays an important role in the simulation study. It aims to synchronize the simulation time among different simulation subsystems.

At the moment the FAMAS Backbone Architecture supports only discrete event based simulation. There is a special technical subsystem called Backbone Time Manager that implements the conservative and real-time synchronization. Using the conservative synchronization only one model is considered as “current” at any moment. This can be optimized to minimize communication by offering each model a time horizon and conditions under which it can act autonomously without consulting the Time Manager. The real-time synchronization is interval based and supports the experimenting with real equipment.

The time management mechanism of the HLA is more flexible and advanced because accommodates a variety of time management policies. The RTI provides an optimal time management service to coordinate the exchange of events between federates. There are possibilities for conservative, optimistic, time-stepped or real-time synchronization. In a federation, time always moves forward. However, the perception of the current time may differ among participating federates. Time management is concerned with the mechanism for controlling the advancement of each federate along the federation time axis (Defense Modeling and Simulation Office 1996). Regarding the time in HLA we can distinguished two important federates, namely *regulating* and *constrained*. Regulating federates regulate the progress in time of federates that are considered as constrained.

3.6 Extended Capabilities Offered by the Architectures (built-in logging and animation possibilities)

There are capabilities or functionalities offered by simulation packages that are not necessarily needed for a simulation run. The functionalities in the simulation packages are increasing based on the requirements and feedbacks of simulation modelers. Recently the most successful packages contain complex 2D and 3D animations, various analyzers, a lot of logging possibilities and so on. The architectures that support distribution among simulation models should contain some functionalities regarding the request of modelers or should enable an easy coupling technique to existing components that offer all the required functionalities. In FAMAS there are two technical subsystems that offer extended functionalities, namely the Logging subsystem and the Visualization subsystem.

The Logging subsystem provides a well-defined possibility to log the collected data in a relational database. The relational database management system helps us to select and categorize the stored information by easily writing SQL commands. In (Boer et al. 2002b) are defined mechanisms that can be used in Logging System, such as: log everything, log only relevant data and log at the end. There is also a separate tool, called Logging Viewer, which enables the modeler to analyze the logging information in the database.

The Visualization subsystem of the FAMAS backbone system makes the simulation state visible and makes it possible to present the simulation results to third parties during or after the run. The visualization may include snapshot screens, animation, statistics of various performance indicators and status views of equipment and processes. For the presentation purposes a 3D animation

subsystem has been developed. The subsystem can start various instances by showing different views of parts of the connected models and an overview of the whole system.

Although the HLA does not include any visualization subsystems, it provides possibilities to couple existing animation systems as animation federates. In (Straßburger 2001) there are two different animation tools discussed that produce online animation for the federation. The first tool is a general animation system, called Skopeo, which provides platform independent system animation anywhere in the Word Wide Web. The second tool, called Proof Animation for Windows, provides online animation on Windows Platforms.

3.7 Compatibility and Support for Civil Domain

As we mentioned before in the FAMAS Backbone Architecture the functional components are the properly so-called simulation models. The technical components provide the common tasks that can be used by the functional components in order to exchange information and to synchronize their simulation clock. A functional simulation model may be implemented either in a programming language (Java, C++, Delphi, etc.) or in any commercially available simulation packages (Arena, eM-Plant, Enterprise Dynamics, etc.). One of the big efforts made in the FAMAS research is to enable coupling of the functional subsystems to the backbone architecture with minimal effort. This means a good communication between the packages and the FAMAS architecture. In order to achieve this several interfaces were created for the subsystems that connect the simulation package or programming languages to the backbone. These wrappers (generally written in DLL's) are important part of the backbone system and they must run on the same computer as the corresponding simulation package (Veeke et al. 2002).

Although the architectural design of HLA achieves reusability and interoperability, HLA is still a technology which was developed within and for the world of military simulations (Defense Modeling and Simulation Office 1996). This can be seen in the way HLA supports interoperability: HLA defines interfaces in programming languages like C++, Java, or ADA, since these are the languages in which many military simulation models have been developed. In contrast to the military field, in the civil world simulations are most often developed using more user-friendly simulation packages which are commercially available, such as ARENA, Automod, Taylor ED, eM-Plant, SLX. This is a major difference between the military and the civil simulation communities (Straßburger 2001).

Although HLA is completely oriented towards military fields (Davis and Moeller 1999), there are some approaches that examine the possibility of how the civil domain could benefit from the HLA approach. In (Schulze et al. 1999) it is discussed how HLA can possibly become the standard of interoperability of civil simulation. There are some applications that develop HLA interfaces for existing commercial simulation packages. In the approach offered by (Straßburger 2001) there is a HLA interface developed for the SLX (Henriksen 1997) simulation language.

One of the projects that shows the applicability of HLA in civil domain is described in (Gan et al. 2000) that analyzes the performance of a distributed supply chain simulation model. Here a distributed simulation technique is presented using the HLA in order to allow corporations to construct a cross enterprise simulation while hiding the construction and functional details.

4 FAMAS - HLA BRIDGE

In order to be able to support the different project groups participating in the FAMAS.MV2 research program, the simulation environment must be flexible, easy to use and must achieve the following objectives:

- *Transparent structure* helps the modeler to couple the simulation models effortlessly. In the FAMAS architecture the common tasks for the support of a distributed simulation study are implemented in the structure itself, thereby avoiding that the simulation model developer needs to do a lot of programming to couple their models to the backbone structure.

- *Reusability* stands for reusing already modeled simulation models, resources and controls.

- *Hierarchical modeling,* which means that the models for the FAMAS project will be at different level of detail, and rough models can be implemented in more detail in later stage. The FAMAS Backbone Architecture supports the combination of global and detailed models.

- *Interoperability* means the use if different simulation and programming environments working together in a distributed way. Therefore each project is free to chose the platform it prefers, without loosing the ability to communicate with or use already defined models.

By satisfying all these objectives and functionalities mentioned in (Boer et al. 2002b) the FAMAS Backbone Architecture achieves their final aim.

The HLA is developed for the military domain, but recently the number of HLA compliant models in the civil domain are increasing. Therefore another objective of the FAMAS Backbone Architecture is to enable to couple

HLA compliant models to FAMAS Backbone Architecture.

In this section we describe the FAMAS-HLA bridge, which serves to connect these two distributed simulation architectures. (see Figure 3).

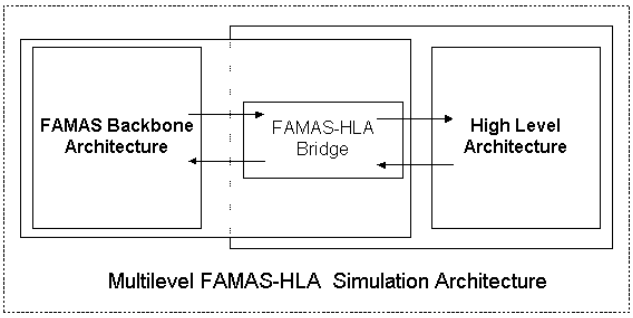

Figure 3: Connecting FAMAS and HLA applying a FAMAS-HLA bridge

The FAMAS-HLA bridge can participate in multiple distributed simulation architectures. Therefore, we can consider FAMAS-HLA bridge from two different point of view. From FAMAS Backbone Architecture point of view this bridge is a functional subsystem that is a participant in the FAMAS Backbone Architecture, while from HLA point of view it is a federate that is part of the HLA federation.

Considering the previously mentioned two views, the interfacing mechanism must be well defined in both directions. Therefore the FAMAS-HLA bridge defines two interfaces (see figure 4):

- There is a message oriented interface developed for FAMAS Backbone Architecture. It corresponds to the multi client server architecture design that sends and receives well-specified low level FAMAS WinSock messages.

- Furthermore, there is a message oriented interface developed for HLA, which corresponds to the RTI ambassador paradigm.

In spite of the fact that the FAMAS Backbone Architecture is hidden from the HLA and HLA is hidden from the FAMAS Backbone Architecture, the FAMAS-HLA bridge allows time synchronization and information sharing and exchange between the two architectures. The FAMAS-HLA Bridge Manager is the only one, which is able to map information (attributes, parameters, interactions, etc.) of two architectures using the HLA-INFO and FAMAS-INFO information sources. For example, an arbitrary FAMAS component changes an attribute X in a certain time. Due to the fact that attribute X has a correspondent in the HLA architecture, the Virtual FAMAS component is interested in any change of this attribute. Therefore, the Virtual FAMAS component will be informed about the changing of attribute X. The Virtual FAMAS component immediately informs the FAMAS-HLA Bridge Manager, which tries to find the correspondence between the FAMAS and the HLA attribute, using the FAMAS-INFO and HLA-INFO. If there exists an attribute Y in HLA that corresponds to the attribute X in FAMAS, the FAMAS-HLA Bridge Manager will immediately update the new value of Y.

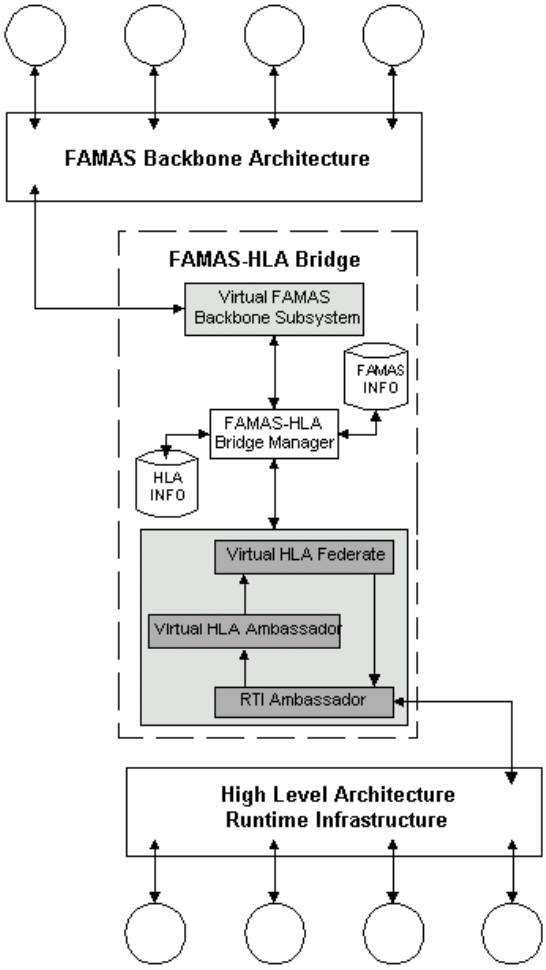

Figure 4. Detailed description of FAMAS-HLA bridge

Besides information sharing, the FAMAS-HLA Bridge Manager must coordinate the time synchronization between two distributed architectures. Coupling different architectures in the way that one architecture might be the part of the another one leads to a *multi level hierarchical distributed structure*.

5 CONCLUSIONS

In spite of the fact that we did not find the HLA suitable to achieve our final purpose (Boer et al. 2002b), it is a de-facto standard for distributed simulation in the military field and is growing its importance in the civil domain. Therefore we expect that a number of future simulation models will be HLA compliant. Although some commercial simulation packages can be coupled to HLA (Straßburger 2001), we can not expect the modelers to have enough HLA knowledge to make the models HLA compliant. In some manners the FAMAS Backbone Architecture is not so sophisticated as HLA, but it offers a more simple distributed execution and a more flexible possibility of coupling commercially available simulation packages.

The FAMAS Backbone Architecture might achieve its next objective, namely the coupling of HLA compliant models to the architecture. This coupling mechanism can be realized by a so-called FAMAS-HLA bridge By coupling FAMAS with HLA compliant models can achieve an important functionality of distributed simulation, multi hierarchical level design and development, that is not present at all in HLA. Coupling two different architectures leads to interoperability problems, especially information sharing, that can be resolved by the concept of FAMAS-HLA bridge.

REFERENCES

Boer, C.A., Y.A. Saanen, H.P.M. Veeke, and A. Verbraeck. 2002. "Final Report Simulation Backbone FAMAS MV2. Project 0.2 Technical Design." Research report to Connekt, 34 pages, Delft, The Netherlands. (March).

Boer, C. A., A. Verbraeck, and H.P.M. Veeke. 2002. "Distributed Simulation of Complex Systems: Application in Container Handling." In *Proceedings of the 2002 European Simulation Interoperability Workshop* (Harrow, Middlesex, UK, June 24-26). SISO, 134-142.

Davis, W. J. and G. L. Moeller 1999. "The High Level Architecture: Is There a Better Way?" In *Proceedings of the 1999 Winter Simulation Conference (P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.)*. IEEE, Phoenix, USA, 1595-1601.

Defense Modeling and Simulation Office. 1996. *HLA Specification.* Washington DC, USA. Available online via https://www.dmso.mil/public/transition/hla/, [accessed September 24, 2002].

DIS Steering Committee. 1994. "The DIS Vision, A map to the Future of Distributed Simulation." *Technical Report IST-SP-94-01*, Institute for Simulation and Training, Orlando Florida, USA.

FAMAS MV2 Backbone Project. 2001. Research Program FAMAS Maasvlakte II Project 0.2 - Simulation Backbone. Delft, The Netherlands. Available online via http://www.famas.tudelft.nl [accessed September 24, 2002].

Fujimoto, R. M. 2000. *Parallel and Distributed Simulation Systems.* John Wiley & Sons,Inc., New York.

Gan, B. P., L. Liu, S. Jain, S. J. Turner, W. Cai, and W. Hsu. 2000. "Distributed Supply Chain Simulation across Enterprise Boundaries." In *Proceedings of the 2000 Winter Simulation Conference (J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.)*. IEEE, Orlando, USA, pp. 1245-1251.

Henriksen, J.O. 1997. "An Introduction to SLX." In *Proceedings of the 1997 Winter Simulation Conference (S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, eds.)*. IEEE, New Jersey, USA, pp. 559-566.

Schulze, T., S. Straßburger, and U. Klein. 1999. "Migration of HLA into Civil Domains: Solutions and Prototypes for Transportation Applications." *Simulation*, Vol. 73, No. 5, November 1999. pp. 296-303.

Straßburger, S. 2001. *Distributed Simulation Based on the High Level Architecture in Civilian Application Domains*. Ghent : Society for Computer Simulation International, Magdeburg, Germany.

Veeke H.P.M., Y.A. Saanen, W. Rengelink, A. Verbraeck. 2002. "Final Report Simulation Backbone FAMAS MV2. Project 0.2 Functional Design." Research report to Connekt, 20 pages, Delft, The Netherlands. (April).

AUTHOR BIOGRAPHIES

CSABA ATTILA BOER is a Ph.D. student at the Department of Computer Science of the Faculty of Economics at Erasmus University Rotterdam, The Netherlands. He received his M.Sc. degree in Computer Science at the Babes Bolyai University, Cluj Napoca, Romania. Since April 2001 he has been involved in the FAMAS MV2 Simulation Backbone project. His research focuses on multi-level distributed simulation of complex systems. His email address is <acboer@few.eur.nl>.

ALEXANDER VERBRAECK is an associate professor in the Systems Engineering Group of the Faculty of Technology, Policy and Management of Delft University of Technology, and part-time research professor in supply chain management at the R.H. Smith School of Business of the University of Maryland. He is a specialist in discrete event simulation, both for real-time analysis and control of complex transportation systems and for modeling business systems. His current research focus is on the development of generic libraries of distributed object oriented simulation building blocks. His email address is <a.verbraeck@tbm.tudelft.nl>.