Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

THE POSSIBLE ROLE OF A BACKBONE ARCHITECTURE IN REAL-TIME CONTROL AND EMULATION

Csaba Attila Boer

Erasmus University Rotterdam
Faculty of Economics
Dept. of Computer Science
P.O. Box 1738, 3000 DR Rotterdam
THE NETHERLANDS

Alexander Verbraeck

Delft University of Technology
Faculty of Technology, Policy and Management
Systems Engineering Department
P.O. Box 5105, 2600 GA Delft
THE NETHERLANDS

Hans P.M. Veeke

Delft University of Technology
Faculty of Design, Engineering and Production
Transport Technology Group
Mekelweg 2, 2628 CD Delft
THE NETHERLANDS

ABSTRACT

The complexity of technical systems that need to be designed and researched is gradually increasing. In order to be competitive and to satisfy the demands of the market, thorough analyses and early risk assessment of the systems are required. System investigation might involve some changes that might entail the redesigning or even redeveloping of some parts of the system. Modifications are usually expensive, time consuming and risky. Consequently, there is a need for methods that examine the possible effects of the modifications before investing in the alteration of the system. There are four methods that are used to study the behavior of a system: simulation, emulation, real-time control and prototyping. In this paper we discuss an approach for a distributed architecture that supports these four testing methods and the interaction between them.

1 INTRODUCTION

One of the aims of a simulation study (Zeigler et al 2000) is to provide performance indicators about real systems before investing in their development or adaptation. In this paper, we look at simulation studies for technical systems with complex control components. Both during simulation and in reality, the execution process must be controlled and monitored in order to prevent and solve problems that might occur. Therefore, next to the *equipment* (resources) that provides the services, additional *control systems* are needed that control the equipment. In order to fulfill their role, the equipment and the control systems should collaborate and communicate in a well-defined and well-designed manner.

The development of a complex real system which is controlled by a separate control system may include one or more of the following phases, which aim to test the system during different design stages (Auinger et al. 1999). These four design stages are also illustrated in figure 1:

1. Full simulation: includes the simulation of both the equipment and the control systems;
2. Real-time control: uses real equipment and simulates the control systems (Verbraeck et al. 2000);
3. Emulation: simulates the equipment and uses real control systems (Mueller 2001);
4. Prototyping: involves tests with real equipment and real control systems.

Full prototyping seems the most realistic testing possibility, although it is quite expensive to build and experiment with the whole prototype system, especially because it involves the risk of failures if the possibilities of its design are not tested thoroughly beforehand. Full simulation, on the other hand, does not involve so high costs, however, it may disregard some phenomena that are present in the real system or contain additional factors that might influence the outcomes. Emulation and real-time control have the advantage that they can be carried out in a cheaper way

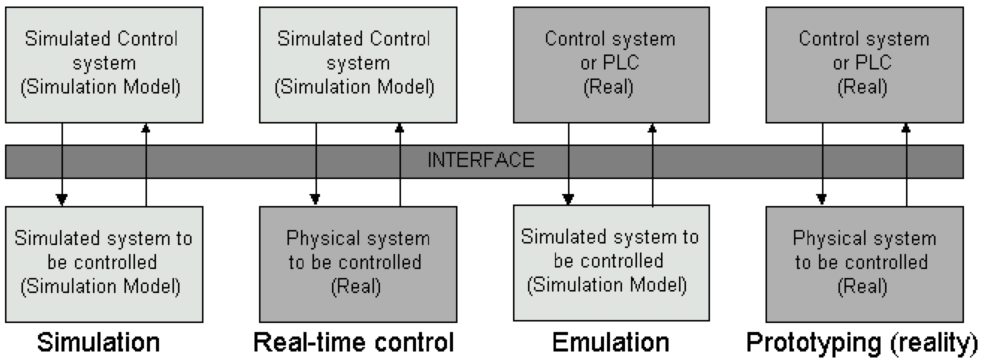
Simulated Control system (Simulation Model)
Simulated Control system (Simulation Model)
Control system or PLC (Real)
Control system or PLC (Real)
INTERFACE
Simulated system to be controlled (Simulation Model)
Physical system to be controlled (Real)
Simulated system to be controlled (Simulation Model)
Physical system to be controlled (Real)
Simulation
Real-time control
Emulation
Prototyping (reality)

Figure 1: The Four Possible Ways of Testing Systems

than full prototyping, and stay closer to reality and are therefore less time-consuming than full simulation. It is important to note that both the development of the real system and the development of the software control system are very expensive. Most organizations are reluctant to invest when the system is still in its design stage.

The testing stage (or experimentation in full simulation studies), where the interaction between the control system and the equipment is studied, is meant to measure the performance indicators of the system in order to be able to make changes in the equipment part or the control part of the system. Testing systems must reflect real situations as much as possible. For this reason different requirements, regarding the components of the system and the communication between them, must be satisfied. First of all the equipment and the control systems should behave in the same way as in reality – although it is difficult to indicate on beforehand on which aspects the similarity should be maximal. Next, the communication between various software and real components must achieve the same performance independently of the complexity of the system. The scalability of the system and the bandwidth of the communication should influence the performance of the whole system in a similar way as in reality. In addition, interrupting the execution of the process might entail some difficulties. A well-defined recovery mechanism should be offered in order to prevent from loosing the information, just like in reality. Furthermore, the robustness of the system must enable the continuation of the execution (after interruption) without any problem and with minimal user interference. All these requirements must be satisfied in order to achieve an effective communication, and to have a valid representation of a ‘real’ technical system.

The main problem we have to deal with when linking different simulation and real components is to find an appropriate method for connecting these elements together. The protocols, speed, and recovery mechanisms used in reality might differ from those that are provided by simulation languages, leading to different results in each of the testing situations of figure 1. In this paper we introduce a taxonomy of connecting components to the system through a backbone architecture, and we analyze different possibilities for attaching models and real components. We also discuss how (new) communication protocols could be used to solve the communication between various components developed in different environments.

The paper is structured as follows. In section 2 we introduce the basic concepts and the requirements of the FAMAS Simulation Backbone project. Section 3 describes the logical and technical structure of the backbone architecture. In section 4 we introduce an approach for backbone for connecting real-time control and emulation systems. Conclusions can be found in section 5.

2 THE FAMAS PROJECT

The FAMAS (First All Modes All Sizes) project provides tools for designing the container terminals for the future Port of Rotterdam (FAMAS 2001). Several simulation groups develop different functional areas of the new part of the port that will host the container terminals. Various new parts will be designed and developed in different simulation environments based on the experience of the modeling teams. The aim of the FAMAS Simulation Backbone is to provide a flexible architecture for the interoperability among various distributed simulation models (Boer et al. 2002). The success of the performance of container handling in the container terminals depends very much on the performance and functionality of the control strategies and algorithms. Therefore we have to carry out tests of a complex system, that is composed of a variety of interacting real and control subsystems, in a comprehensive way.

2.1 The Basic Concepts

This section aims to introduce some indispensable concepts of the harbor processes related to container handling. We distinguish three important basic processes that use the container: Storage, Transport and Transfer.

All the processes are defined as functions, which convert the actual state of the system into a new one. The *Storage* function (ΔS, Δt) executes a change that leads the physical container into a new state *S* during a storage time *Δt*. The *Transport* function (ΔP, ΔS, Δt), next to the modification of the container state and time moment, involves the change of an additional state, namely the position *P* of the container. Finally, the *Transfer* function (ΔM, ΔP, ΔS, Δt) is an extension of Transport function that entails the transformation of the involved modalities *M* (ships, trucks, trains, AGV's, cranes, etc.) as well.

The basic processes can be carried out by several *resources (R)*. Resources refer to personnel, equipment or space. For example, a quay crane performs a transfer function, an AGV a transport function and a carrier a sequence of transfer / transport / transfer functions. At the individual level (one resource) a *resource control* might be needed.

More instances of the same resource form a so-called *resource system (RS)*. When more resources are connected, a *resource control* is needed that regulates the collaboration of similar resources. Different resource systems can be combined into *multiple resource systems (MRS)*, with a coordinating and/or hierarchical control function. A special type of a multiple resource system is the *terminal (T)*, where the system is a geographically bound, autonomous organizational unit with connections to external transport functions.

When we combine a set of terminals to a co-operating complex of organizational units, we have a *terminal complex (TC)*. In contrast to single terminals TCs require the presence of some inter-terminal transport functions (ITT) and usually an overall terminal complex management function. The ITT system combines at least two terminals, therefore it is located on the same hierarchical level as the terminals. Figure 2 sketches the hierarchy of system components for the port project.

Resources can be exchanged at any hierarchical level, either between resource systems or between terminals. One resource belongs to one and only one resource system at a time. When the resource – e.g. an AGV or a quay crane – moves from one system into another it performs a transfer from one resource system into the other.

Information that arrives and that is sent by a component is defined as a flow. We distinguish two different types of flows that are handled by the system: *control flow* and *physical flow*. Control flows are commands that regulate the functioning of the system and are processed by control algorithms. Physical flows refer to – information on – real objects being transferred between two subsystems (Figure 3). In reality these transfers of real objects or state changes of components will trigger sensors, which lead to information being sent to the control algorithms.

2.2 Requirements of the FAMAS Project

The FAMAS Backbone Architecture is designed in a way as to satisfy the following requirements (Boer et al. 2002):

- *Distributed execution*: this can be achieved by a well-defined interoperability between different simulation components. The interoperability in the FAMAS Simulation Backbone is provided by a low-level message passing mechanism.
- *Optimal communication*: effort is required to attain an effective communication speed.
- *Stand-alone and distributed testing:* refers to the possibility to test distributed simulation models developed by different parties as in standalone as in distributed environment.
- *Package independence*: this requirement focuses on combining simulation models implemented in different simulation packages (e.g. Arena, eM-Plant, Enterprise Dynamics) and programming languages (C++, Java, Delphi, etc.). The characteristics mentioned so far reflect the grade of flexibility of the architecture and reusability of the simulation models.
- *Structure transparency*: aims to give some insights into the architecture for the groups who intend to develop models or support subsystems for

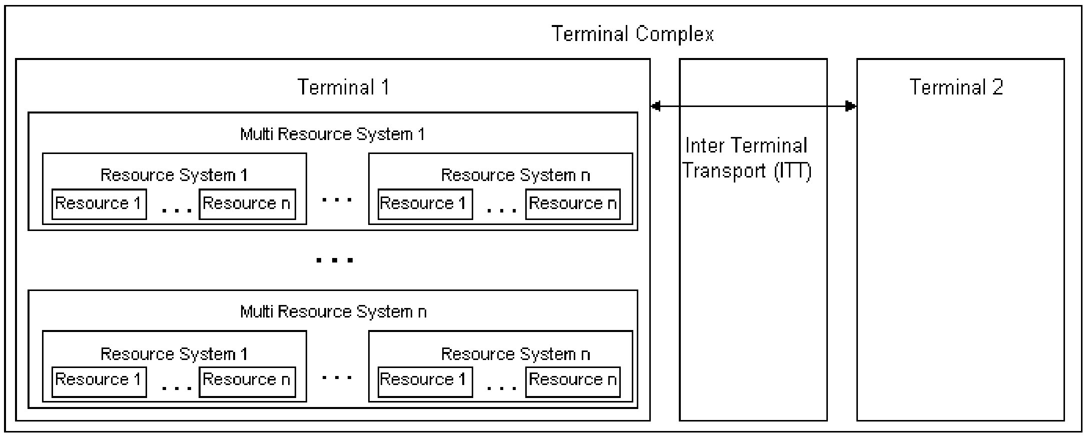

Figure 2: Hierarchical Concepts of the Port Processes

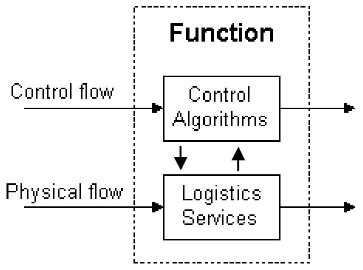

Figure 3: The Control and Physical Flows

it, in order to provide interoperability. The transparency helps the modeler to couple the simulation models effortlessly.

- *Hierarchical structure* allows for modeling, design, and development in a hierarchical manner. This feature is essential in the FAMAS project as the models might be developed at different levels of detail.

3 CONNECTING COMPONENTS TO THE FAMAS BACKBONE

This section introduces a taxonomy of connecting different components to the system. This classification helps to distinguish between different methods that can be used to attach real systems and simulation components to the backbone.

3.1 Connection at Logical Level

During simulation several pieces of equipment and controllers are attached to the system. Equipment (resources) and controllers can be joined separately to the backbone. A resource can be linked together with its controller or, at a higher level equipment and controllers can form a federation and can be attached to the backbone as a single, compound component. Based on these possibilities we distinguish between three methods that can be used to attach simulation models to the FAMAS Backbone architecture:

- control-based connection;
- function-based connection;
- system-based connection.

In the control based connection (Figure 4), every single control function (system controller) and execution function (resources or equipment) is separately connected to the backbone. Consequently, the backbone supports the communication between a resource (system) and its control. The grade of reusability of the resources is maximal in this case as each individual execution and control component can be reached separately via the backbone. This method has however the disadvantage that it overloads the backbone. Moreover, high demands are put on the developer, as he/she is required to take care of the connection of every single component to the backbone. Furthermore, this joining process might cause difficulties or even impede the hierarchical structuring (Figure 2), as all components must be developed as completely separated simulation models, which is usually not what the modelers want. A quay model, for instance, includes the quay cranes, while in this case, each quay crane and each control of a quay crane would be a separate model or system connected to the backbone.

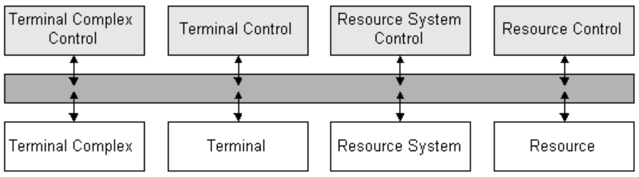

Figure 4: The Control Based Connection Structure

The *function based connection* (Figure 5) supports the attachment of complete systems (resources together with their controllers) to the backbone at any level of aggregation. Control and execution functions (the control system and the controlled system) are considered as a single subsystem (federation) that needs to be attached to the backbone. The details regarding the connection and communication protocol between the equipment and controller are left to the modeler's decision. Controlled and control elements might function on the same computer or in a distributed way. The communication between them is solved in a direct way, without making use of the backbone channels, consequently, the extraneous use of the backbone is handled by this approach. The hierarchical structure, however is still not realizable.

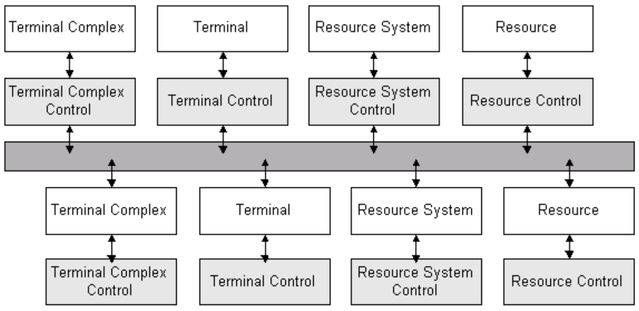

Figure 5: The Function Based Connection Structure

System based connection (Figure 6) supports the combination of subsystem in a hierarchical way. It is in fact a restricted version of the function based connection, which allows for the attachment of systems at the terminal or terminal complex level. During system based connection sets (federations) of resources and controllers are attached to the system as one single element. The modeler focusing on the backbone therefore does not have to bother with the details regarding the (hierarchical) realization of the subsys-

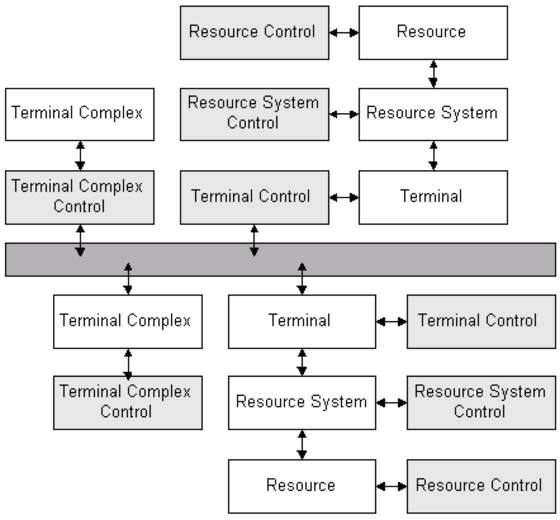

Figure 6: The System Based Connection Structure

tems that are attached to the backbone. In this way standard, reusable components can be defined and reused in a simple manner. The disadvantage of this method is that resources (e.g. AGV's) or controllers that communicate with a lot of other elements, might have to be included in more then one subsystem or model, because otherwise the structure of the subsystem might became too intricate.

The high variety of the questions that needs to be answered by the FAMAS port project requires a very flexible and transparent architecture. As individual projects might use models at any level (resource or terminal), the FAMAS Backbone should support the combination of all three connection structures mentioned above. Although the control based connection structure provides maximal reusability, the drawback of this structure design lies in the overload of communication through the backbone architecture with real-time testing. Therefore, for the real equipment that need very frequent communications with other components, the function based connection is more advisable.

3.2 The Technical Structure of the FAMAS Backbone

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.

In Figure 7 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. 2002). The overall system consisting of both technical and functional subsystem is sometimes called a *federation*, where the subsystems that connect to the backbone are the *federates.*

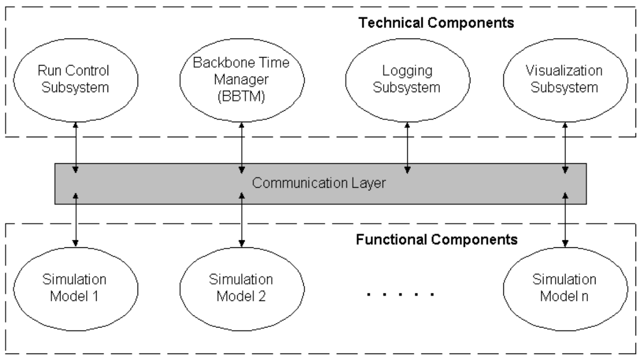

Figure 7: The Structure of the FAMAS Simulation Backbone Architecture

The technical subsystems provide the following functionality

- *Run Control* controls the experiments: it starts, stops and periodically monitors the simulation process;
- *Backbone Time Manager (BBTM)* synchronizes the simulation time among different simulation subsystems' (Fujimoto 2000);
- *Logging* aims to collect logging information from the distributed functional and technical components into a central database;
- *Visualization* provides separate or common visualization views for the different subsystems or the entire simulation.

4 THE EXTENDED FAMAS BACKBONE ARCHITECTURE

So far the FAMAS backbone architecture was focused to connect simulation components. This section aims to extend the backbone as to accept the attachment of real time-control and emulation systems as well.

4.1 Connecting Simulation and Real Components in a Distributed Way

In a complex system, such as a container port, there are thousands of pieces of equipment and controllers. Testing of complex systems like a port system might entail several difficulties, which, beside the general communication problems discussed in the first section, concern the variety of simulation environments, variety in the real equipment and differences between communication protocols.

The simulation models or simulation components are usually developed by different modelers, using different concepts and different simulation environments. Thus, the communication between various environments should be enabled in order to provide collaboration. Equipment and simulation models support different communication protocols, therefore, different models can communicate only if a common protocol is worked out or several interfaces are developed, that allows for communication between any two of them. Due to the complexity of the system the elaboration of several interfaces might cause an *explosion* of the number of model-model or model-equipment protocols.

Although the primary aim of the FAMAS Simulation Backbone is to provide a flexible architecture for the interoperability between various distributed *simulation models*, an extended version of this architecture should support the interaction with real components as well. The functional components of the advanced system consist of real control systems (e.g. PLC's), and real equipment (e.g. Automated Guided Vehicle (AGV)), next to the simulation models and control programs.

Figure 8 depicts the new architecture of the FAMAS Backbone System, which enables all the testing possibilities illustrated in Figure 1, namely: simulation, emulation, real-time control and prototyping. The logical structure of the architecture should also support all the three structure designs (*control based connection*, *function based connection* and *system based connection*) as described in section 3.1, in order to couple simulation and real systems at any abstraction level.

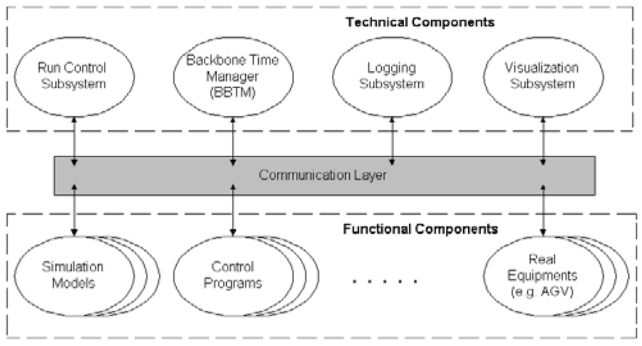

Figure 8: Extended FAMAS Backbone Structure Containing Real Equipment and Controls

4.2 Defining a Final Protocol for Effective Communication

Communication between several different types of components can be solved in two different ways. As a first approach one single, common protocol could be developed, that supports the communication between all the components, indifferent of their original communication protocol. Another solution is to elaborate new protocols between any two types of components.

Let us take the set of the protocols supported by the simulation packages (P_{SIM}), real equipment (P_{REA}) and real control system (P_{CONT}).

We distinguish four situations, regarding the communication protocols, as illustrated in figure 9:

1. $\exists\, p \in P_{SIM}, P_{REA}, P_{CONT}$ then take p as a common protocol,
2. $\exists\, p \in P_{SIM}, P_{REA} \wedge p \notin P_{CONT}$ then
   - Wrap P_{CONT} in order to support protocol p
   - Wrap P_{SIM} and P_{REA} in order to talk a protocol $p' \in P_{CONT}$
3. $\exists\, p \in P_{SIM}, P_{CONT} \wedge p \notin P_{REA}$ then
   - Wrap P_{REA} in order to support protocol p
   - Wrap P_{SIM} and P_{CONT} in order to talk a protocol $p' \in P_{REA}$
4. $\exists\, p \in P_{REA}, P_{CONT} \wedge p \notin P_{SIM}$ then
   - Wrap P_{SIM} in order to support protocol p
   - Wrap P_{REA} and P_{CONT} in order to talk a protocol $p' \in P_{SIM}$

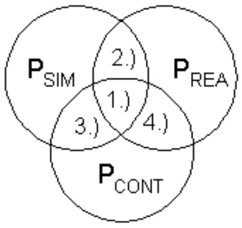

Figure 9: The Set of Simulation, Emulation and Real-Time Protocols

The first situation with the current existing simulation environment will never occur. Our experience shows that no standardized communication protocol exists that can be easily used by any simulation language without wrapping. In other words we can state that: $\bigcap_{i=1}^{n} p_i = \varnothing$, where p_i represents the supported protocol set of the ith simulation package (figure 10).

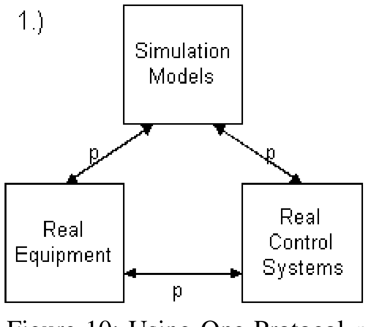

Figure 10: Using One Protocol *p* Without Wrapping

Usually we are faced with the last three cases when we need to wrap either the simulation or the real components. There are a lot of situations when the real control systems and the real equipment can communicate with each other but the COTS simulation packages can not communicate with them using the same protocol (Figure 11). In this case we prefer to choose the first solution of the fourth case, because we need to wrap only the simulation models. We strive to design and develop a *neutral protocol* that is supported by all the participants (both simulated and real) by minimizing the wrapping procedures.

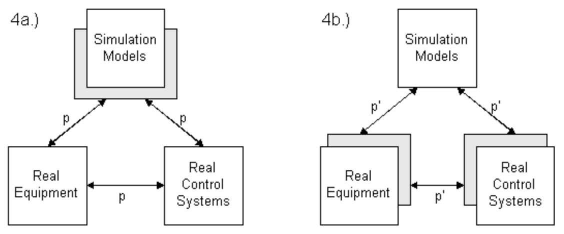

Figure 11: The Simulation Models do not Support the Protocol of Real Components

In the FAMAS project we distinguish between two kinds of protocols, a *technical* and a *functional protocol*. The technical protocol refers to the lower OSI layers, where the communication between components is realized. For this purpose the FAMAS project uses low level Winsock messages, as most of the simulation packages support the communication at this level.

The functional protocol is on a higher level than the technical protocol but it is based on the low level protocol. The functional protocol refers to information sharing between simulation models, such as variable exchange, object sharing, etc. Currently the commercially available simulation packages does not support at all the communication and collaboration at this level. In FAMAS we introduced a message protocol that tries to support high-level information exchange among simulation components.

In an advanced version of the FAMAS Backbone structure we aim to reanalyze the protocols that can be supported by the simulation and real components. Based on the results the wrappers can be designed and developed. The technical layer does not need modification, as it is designed at a low level and is supported by most of the simulation and real components. The functional layer is used by the simulation models and it needs to be extended for the real controls and equipment.

5 CONCLUSION

In order to improve the performance of a complex technical system, which consist of a set of subsystems and control systems, comprehensive experiments and tests are needed. Currently we can distinguish four types of analyzing and testing methods, namely simulation, emulation, real-time control and full prototyping. The FAMAS Backbone architecture is an approach that enables for the analyses and testing of distributed simulation models. Performance testing might be more advantageous in some cases, to test the effects of real control systems and control strategies or to test the effects of control strategies on real components. To enable the interaction between simulation models and real components we described the extension of the FAMAS backbone architecture in order to support this combination. We combine therefore all four testing methods: simulation, emulation, real-time control and prototyping, using a logically and technically distributed architecture. This paper discusses the logical structure of such an architecture and the problem of communication between several COTS simulation models and real components. We introduced a taxonomy that helps to distinguish between different connection possibilities of the new components to the backbone. It is also discussed how the communication problem between different types of components can be handled. By combining the control based, function based, and system based connection structures, a very flexible and multi-protocol interface can be built, in which the simulation models communicate using a neutral protocol, and the real components can communicate as much as possible using their 'normal' protocols.

ACKNOWLEDGMENTS

The authors would like to thank the Connekt organization and the TRAIL research school for supporting and funding the FAMAS Backbone project. They also acknowledge the large amount of work of the other researchers in the FAMAS Simulation Backbone project that made it possible to reach the results.

REFERENCES

Auinger, F., M. Vorderwinkler and G. Buchtela. 1999. Interface driven domain-independent modelling architecture for "soft-commissioning" and "reality in the loop". In: P.A. Farrington, H.B. Nembhard, D.T. Sturrock, and G.W. Evans (eds.). *Proceedings of the 1999 Winter Simulation Conference,* IEEE, pp. 798-805.

Boer, C. A., A. Verbraeck, and H.P.M. Veeke. 2002. Distributed Simulation of Complex Systems: Application in Container Handling. *Proceedings of SISO European Simulation Interoperability Workshop*, Harrow, Middlesex, UK, June 24-27.

FAMAS MV2 Backbone Project. 2001. Research Program FAMAS Maasvlakte II Project 0.2 - Simulation Backbone., Delft, The Netherlands. Available online via <www.famas.tudelft.nl> [accessed March 30, 2002].

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

Mueller, G. 2001. Using emulation to reduce commisioning costs on a high speed bottling line. In: B.A. Peters, J.S. Smith, D.J. Medeiros, and M.W. Rohrer, eds.). *Proceedings of the 2001 Winter Simulation Conference,* IEEE.

Verbraeck, A., E. Valentin, and Y.A. Saanen, 2000. Simulation as a Real-time Logistic Control System: AGV Control with Simple++. *The New Simulation in Production and Logistics – Prospects, Views and Attitudes.* pp. 245-255, Berlin, Germany.

Zeigler, B.P., H. Praehofer, and T.G. Kim. 2000. *Theory of Modeling and Simulation.* Academic Press, San Diego.

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 FAMAS MV2 Simulation Backbone project. His research focuses on Multi-Level Distributed Simulation of Complex System. 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 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>.

HANS P.M. VEEKE is an assistant professor at the Department Design and Manufacturing Technology of the Faculty of Design, Engineering and Production of Delft University of Technology, The Netherlands. He received his M.Sc. degree in Operation Research at Delft University in 1983. He was involved in the development of simulation languages (PROSIM, Must, TOMAS) and participated in advanced technological and organizational projects in industry. Since April 2001 he was involved in the FAMAS MV2 Backbone project. His current research concentrates on the integration of system approach and simulation modeling. His email address is <H.P.M.Veeke@wbmt.tudelft.nl>.