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

THE EXTENDED USE OF SIMULATION IN EVALUATING REAL-TIME CONTROL SYSTEMS OF AGVS AND AUTOMATED MATERIAL HANDLING SYSTEMS

Corné Versteegt
Alexander Verbraeck

Systems Engineering Group
Faculty of Technology, Policy and Management
Delft University of Technology
P.O. Box 5015
2600GA Delft, THE NETHERLANDS

ABSTRACT

Control systems for logistic and transport systems are among the most complex control systems in existence. Currently control systems are only fully tested at the shop floor after commissioning. This means a lot of costly failures occur at the startup stages of control systems. The goal of this paper is to describe the extended role that simulation can play in evaluating of fully automated logistic systems and their control systems before commissioning. We followed a three-step approach in evaluating both logistic and logistic control systems. A simulated control system was used to control simulated, emulated, and real prototypes of logistic resources. Three different simulation packages have been used; Simple++, AutoMod, Arena. The control system was implemented in all three simulation packages to control logistic resources at the Connekt TestSite. The TestSite is a special laboratory for testing new technologies in logistic automation.

1 CONTROL SYSTEMS

In this paper we use following definition of control (Aken 1978):

'Control is the use of control actions, or interventions, by a control system to promote the preferred behavior of a system-being-controlled.'

A clear distinction is made between control system and system-being-controlled, as can be seen in Figure 1. Control actions are the efforts the control system uses to influence the state of the system-being-controlled. For control actions the plural form is used to indicate that control is considered as being a continuous process, rather than a single action (Aken 1978). The controller promotes the preferred behavior of the system-being-controlled. This does not mean that the controller completely determines the behavior of the system. The control system influences the system-being-controlled, but the control actions do not have to be successful.

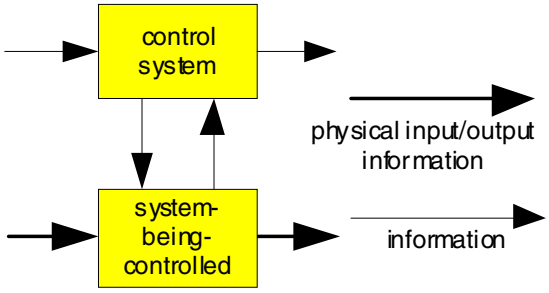

Figure 1: Control System and System-Being-Controlled

In this paper the systems-being-controlled are logistic systems that use highly automated logistic resources, like Automatic Guided Vehicles, abbreviated to AGV, and automated material handling systems. The control systems range from individual control systems to guide single AGVs to automated managers that control large sets of AGVs (Verbraeck and Versteegt 2001). Control systems for logistic and transport systems are among the most complex control systems that are in existence (Pyle et al. 1993). Such control systems have to control many concurrent processes, have to react to input within strict time windows, have a distributed nature, and have to work with large sets of heterogeneous data.

In this paper we show *an approach that designers of control systems for highly automated logistic systems can use to evaluate designed control systems before commissioning*.

2 TESTING CONTROL SYSTEMS

At this moment control systems are often only fully tested after commissioning at the shop floor. Auinger et al. (1999)

state that it is vital to test control systems before implementing them. They suggest using a combination of reality and simulation to test control systems. Four possible approaches to test control systems can be distinguished, based on the possible combinations between reality and simulation, as can be seen in Figure 2 (Auinger et al. 1999):

1. The *traditional way* to test control systems. A combination of a control system and logistic system both in reality. The control system is tested after commissioning.
2. *Soft commissioning*. A combination of a control system in reality and a simulated logistic system. This step is also called emulation (Schiess 2001, Mueller 2001).
3. *Reality in the loop*. A combination of a simulated control system and a real logistic system.
4. *Off-line simulation*. A combination of both a simulated control system and a simulated logistic system.

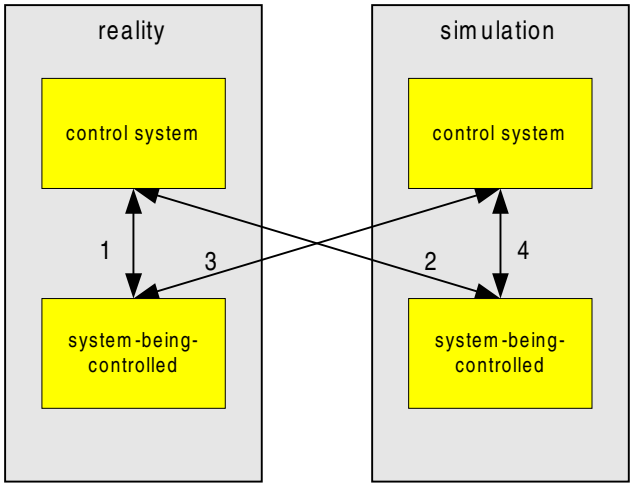

Figure 2: Approaches for Testing Control Systems

At this moment control systems are mostly only fully tested after commissioning at the shop floor, combination 1 in Figure 2. It is difficult to test or pre-commission a control system before implementing and coupling control systems with the real system-being-controlled. The testing takes place during the startup phase of the system-being-controlled. This is an expensive, risky and error-prone way of developing control systems.

Within communities of simulation and control systems emulation has been developed as a new improved way of testing control systems (Mueller 2001). Within emulation the real control system is connected to a simulation model that imitates the machines or production systems (Schiess 2001). Emulation can reduce the developing time of control systems and thus shorten the time-to-market. Emulation allows testing of control systems faster than real-time and under safe conditions. The conditions under which the tests are carried out can be better controlled. This allows us to study different scenarios with which the control system has to deal. The effects of worst-case scenarios, and machine break-downs can easily studied by simulating them. Finally, emulation can be used to train process operators in an easy and safe environment.

Although emulation and the combinations that Auinger et al. (2001) offer can be very useful for testing different kinds of control systems, in our research we need a extended approach. In this paper we advocate a new approach of evaluating control systems for highly automated logistic systems. We applied this approach to test control systems and automated logistics resources for the Underground Logistic System Schiphol, abbreviated to OLS Schiphol. The OLS Schiphol is a highly automated underground logistic system that will transport cargo between Amsterdam Airport Schiphol, logistics centers at Schiphol, the Flower Auction Aalsmeer, and a future Rail Terminal near Schiphol. The OLS Schiphol will use up to 400 Automated Guided Vehicles (AGVs) and 40 automated material handling systems (Verbraeck and Versteegt 2001). Both the control system and system-being-controlled for the OLS Schiphol do not yet exist. Furthermore, there is little experience available for controlling large-scale fully automated underground logistic systems (Versteegt et al. 2001). So there are still a lot of technological uncertainties that have to be solved. Working with only a simulation model of the logistic resources is not enough. Simulation will provide us detailed information on the behavior of the logistic resources, but technical aspects cannot be studied.

Our approach uses the methods that were developed for emulation (Schiess 2001, Mueller 2001) and by Auinger at al. (1999) as starting points. The approach consists of four phases:

1. **Testing in a fully simulated environment or off-line simulation**. In the first phase simulation models are constructed of both control system and logistic system. This phase is the same as the off-line simulation, combination 4 in Figure 2.
2. **Emulation of logistic resources**. In this phase we use highly detailed simulation models of the logistic resources and control systems. Although these models are still simulation models, they much more closely represent the real physical systems. The emulation models that were developed of AGVs contain the real software that will be used in real AGVs.
3. **Combining reality in the loop, emulation, and simulation**. In this phase we combine the simulation models that were developed in phase one and the emulation models that were developed in the second phase. Furthermore, we use prototypes and scale models of the logistic resources. All are con-

trolled by a simulated control system. This can be seen as using the combinations 3 and 4 in Figure 2 simultaneously. We use scale models and prototypes since the real logistic system is not yet available.

4. **Implementation of both control and system-being-controlled in reality**. In this phase the real control system and system-being-controlled are implemented in reality.

The main idea behind our approach is the development of interchangeable simulated, emulated and prototype components of the control systems and the systems-being-controlled. Interchangeable means that components can be changed during experiments without making changes to the control systems. This can be seen as 'plug-and-play' of hardware in Windows, see Figure 3. There are three control layers and one layer that contains the simulated, emulated and real prototypes of the logistic resources.

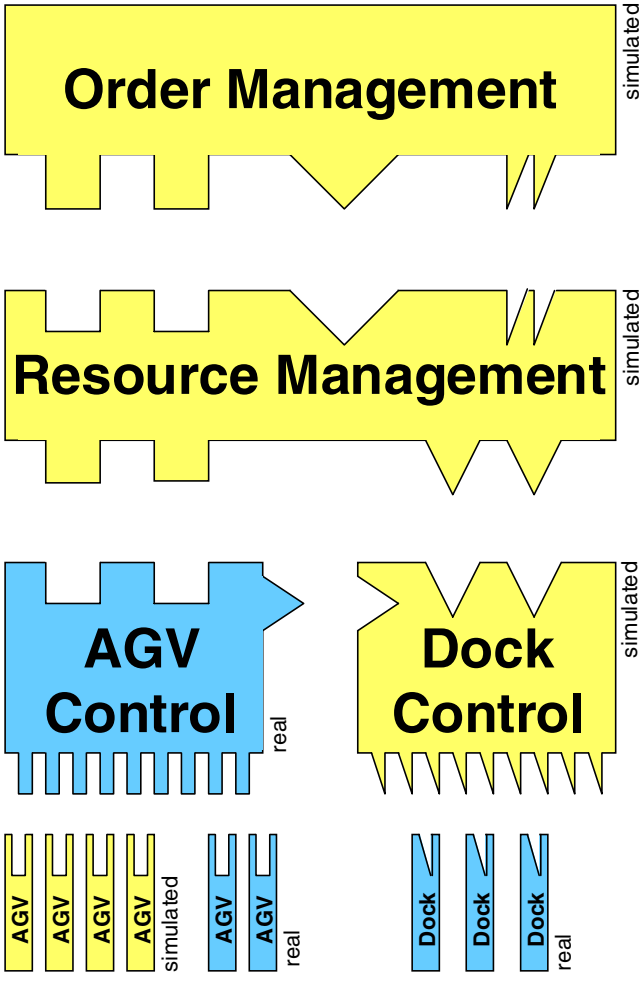

Figure 3: Interchangeable Components

The strategy in our four-step approach is to solve as many of the technical uncertainties at the first stages and delaying the investments in expensive control software and physical logistic resources to later stages. In a fully simulated environment problems can easily and quickly be detected and possible solution can be evaluated for their effectiveness. In later phases the high investments in control software are made, only when the uncertainties and problems are solved. When the uncertainties are solved in the beginning of the project, the chances of investing in wrong technologies is minimized.

3 INTERCHANGEABLE SIMULATION COMPONENTS

We used simulation packages to model the control systems. Literature provides us with several criteria that can be used to select simulation software (Law and Kelton 1991). In our research we used such criteria and two extra case dependent criteria (Verbraeck and Versteegt 2001):

- *Complex control structures*. The simulation package should allow the modeler to model complicated control structures. This makes it possible to implement complex logistic rules and control algorithms. Packages that offer a programming language interface have a clear advantage over more graphical oriented packages.
- *Open architecture*. The simulation package should have an open architecture. It should be easy to cooperate and communicate with other software packages and real systems. The package should be able to deal with both standard communication protocols and user-defined communication protocols. Types of interfaces that can be used are for instance DDE (Dynamic Data Exchange), DLL (Dynamic Link Library), TCP/IP socket connections, ActiveX, OPC (OLE for Process Control), DCOM (Distributed Components Object Model). When needed the user should also be able to construct custom made interfaces.

Based on these criteria we selected three simulation packages; Simple++ version 6.0 (Verbraeck and Versteegt 2000, Aesop 1999), Arena version 4.0 RT (Kelton, Sadowski and Sadowski 1998, Verbraeck et al. 2000), and AutoMod version 9.1 (Verbraeck et al. 2001, Banks 2000). We chose to implement the control system in three different simulation packages. We wanted to show that our approach and concepts for control are independent of any software platform. Furthermore, we wanted to gain detailed insight in the different possibilities simulation packages offer for real-time control and emulation.

The project started with the translation of control system into the simulation package. In Figure 4 our approach

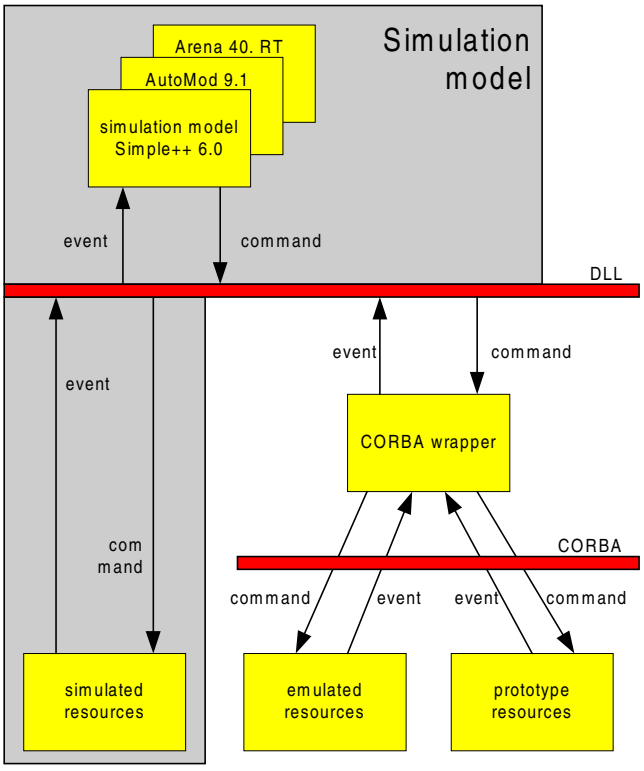

Figure 4: Approach for Testing Logistic Control

is sketched (Verbraeck et al. 2001). The simulation models of the control systems are located on the top. The simulation and emulation models and prototypes are located at the bottom.

The first phase the logistic control system is used to control simulated logistic resources. In the second phase the control systems is used to control emulated logistic resources. In the third phase the simulated logistic control is used to control physical prototypes of the logistic resources. In the end the three different models of the logistic resources could be tested simultaneously. So one simulated control systems controls a combined fleet of simulated, emulated, and prototype AGVs.

To make this possible interchangeable components were constructed, see Figure 3. The interfaces between the components were defined right at the beginning of the project. Communication between control system and logistic system were explicitly modeled even in the earliest simulation models (Verbraeck and Versteegt 2001). Later models and logistic resources had to comply to these interfaces. This meant that sometimes components had to be 'wrapped'. The interfaces between the control system and the systems-being-controlled are identical for all three different types of models of the logistic resources. The simulation model of the control system does not see any difference between simulated, emulated, and prototypes of the logistic resources. The control system sends the same commands and receives the same event messages back.

This approach allows us to change components without any problems. This makes it possible for researchers to independently work on parts of control and logistic systems. When components are finished they cooperate with each other without any problems, since they comply to the interface. The internal structure of components is a black box for other components, only the interfaces are known. All interfaces between the sub-systems were implemented with CORBA similarly to how they can (or will) be realized in the final system. The same interfaces are used for the prototypes, emulated models and simulation models.

An important choice to prepare the simulations and the interfaces of the subsystems for real-world applications was to work with *asynchronous messaging*. In reality, delays occur when exchanging information between system components that are coupled using a network. These delays can play an important role in the success or failure of the resulting control system and control strategies. When synchronous communication is used, the effect of a communication delay might be that the control system blocks until the information exchange has taken place. This might be fatal for other actions that have to be coordinated at exactly the right points in time. Asynchronous communication can help to reduce this problem. The application thereby does not make itself dependent on the immediate answer after sending a message.

Another point to take into consideration when implementing the off-line simulations that have to prepare for the interfacing to real systems is *the single-threaded character* of most simulation languages. All three simulation packages used are single-threaded. When making a complex calculation, or during the drawing of the animation, it might be impossible for the simulation language to handle incoming messages fast enough. The DLLs that were coupled to the simulation and that provide the interfaces to the outside world, were therefore implemented as multi-threaded DLLs. Several 'server threads' are responsible for interfacing to the external components of the system. The DLL buffers the incoming messages, and wait for the simulation model to import and handle the state changes that it received. In our case, we implemented the information exchange between the simulation model and the DLL with a polling mechanism that is triggered by an event in the simulation model. The other possibility, pushing the information into the simulation model during the run, turned out to make most simulation environments unstable. Our approach asked for frequent polling of the DLL information from the simulation model. The method to do the polling is scheduled 10 or 100 times per second.

Finally, the simulation *clock synchronization* with the wall clock needs to be taken care of. This is not as trivial as it seems, and several implementations offered by simulation vendors do not work properly. The usual implementation of wall clock synchronization is to jump to the next event on the event list, to check whether the time of this

event is such that it can be allowed to take place, and if not, delay the simulation environment until the event is allowed to take place. The problem here is that *external* events can come in before the next event time, while the simulation clock has already been advanced to that next time. The external events from the real world are not present on the event list, and therefore the simulation model cannot take these into account when advancing the clock. The event polling mechanism described before, also provided the solution for proper clock synchronization. The external events are transferred to the simulation model at fixed points in time, e.g. 10 or 100 times per second, the simulation clock cannot advance more than 10 or 100 milliseconds for each event, and the information associated with these events is transferred with a minimum time delay into the simulation model.

4 REALITY IN THE LOOP SIMULATIONS

After the first two phases described in section 2, the third phase of the approach aims at testing the control systems and control strategies in cooperation real systems. At Delft University of Technology a special laboratory has been constructed to evaluate new technologies in logistic automation and control systems (Verbraeck and Versteegt 2001). This laboratory, called the TestSite, is a special area of 1600 m^2 equipped with scale models (1:3) of logistic resources, AGVs and material handling systems, that will be used for the OLS Schiphol. Furthermore, prototypes (scale 1:1) of the AGVs and material handling systems are also available, as can be seen in Figure 5.

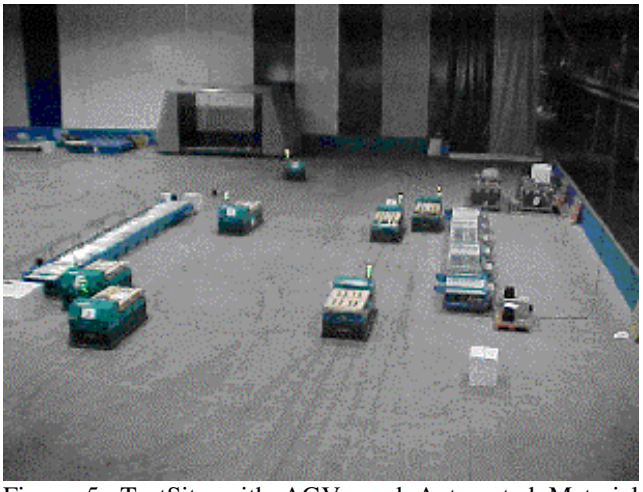

Figure 5: TestSite with AGVs and Automated Material Handling Systems

5 LEARNING POINTS: SYNCHRONIZATION

Synchronization is very important aspect in combing simulation, emulation, and prototypes. Two types of synchronization are distinguished; time and place. The synchronization of time is aimed at synchronizing the simulation clock of the simulated control system to the internal clocks of the prototypes and emulated AGVs and material handling systems. Arena 4.0 RT and Simple++ 6.0 offer standard built-in features for real-time time progress in simulation models, that work very well with the polling and synchronization method described in section 3. For AutoMod 9.1 we constructed a 'wall-clock peeker'. Every fixed time unit, e.g. every tenth of a second, the wall-clock peeker synchronizes the simulation clock with to the internal clock of the computer. This was implemented in a user written C++ function in a DLL. In all three implementations the simulated control system had sometime to 'catch-up' with the wall clock. This was especially the case when control algorithms had to be executed for AGVs to safely pass complicated crossings. These are calculation intensive algorithms. The simulation model then lagged behind the wall clock and had to catch-up with the wall clock. Two solutions were implemented to solve this. Firstly, the calculation-intensive algorithms were transferred from the simulation models into C++ code. The calculations can be executed faster in C++ than in simulation software. Secondly, the asynchronous communication described in section 3 helped a lot. Within synchronous communication, processes have to synchronize and react immediately when information is exchanged. So when one process is not yet ready, the other process has to wait. A more flexible solution is asynchronous communication (Ben-Ari 1990). In asynchronous communication a buffer, or queue for messages, is used. This allows processes to send messages without waiting for an immediate answer (Verbraeck and Versteegt 2001). A process sends messages to the mailbox and continues to operate as normally, without having to wait for the other process. In most cases, however, the simulation clock had to be slowed down to synchronize with the real system.

The *synchronization of place* proved to be more difficult. The positions and orientations of AGVs in the simulation models have to be synchronized with the actual positions and orientations of the physical AGVs at the TestSite. The AGVs have, at this moment, no absolute system to decide their position. The AGVs are equipped with odometers to keep track of their position. For calibration purposes the AGVs use a magnetic grid in the floor. These are, however, only relative calibrations. The odometers of the AGVs prove to work very accurate and are able to keep track of the actual positions of the AGVs. The AGVs are free-ranging, their steering is not guided by external control systems, both mechanical or electronic. Their movements are, however, limited by virtual tracks. The virtual tracks are given to the AGVs by the simulation model.

To synchronize their positions the AGVs send a so-called event notification to the simulation model when they have reached certain positions on a virtual track (Versteegt and Verbraeck 2001). Four different event notifications can be used for synchronization of place; on-event, positioned-

event, passed-event, and near-event, as can be seen in Figure 6. An on-event is generated when the front of an AGV has reached the beginning of a new track. The passed event is the opposite, it is generated when the back of an AGV has completely left an track. The positioned-event is generated when the front of an AGV has reached the end of a track. The near-event is a very special event notification, it is a position in time/space domain, instead of only being a physical position. The position of the near-event is equal to the breaking distance of the AGV towards the end of the track. The exact time and position of a near-event is therefore dependent on the actual speed of the AGV.

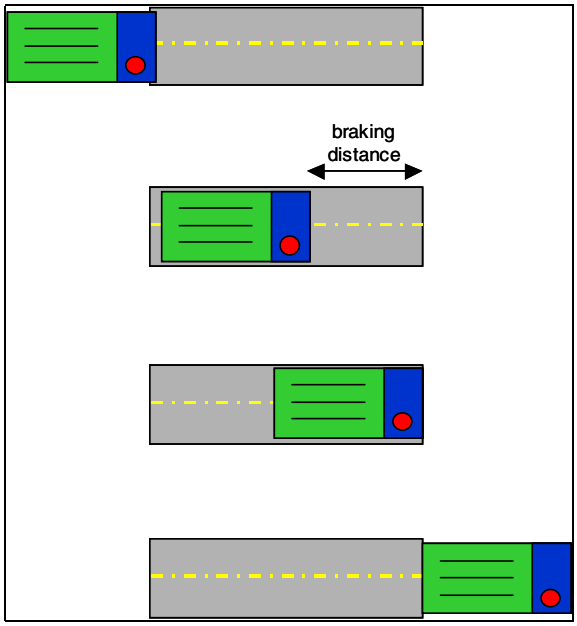

Figure 6: On-, Near-, Positioned-, and Passed-Event (Top-to-Bottom)

When one of the four events has been generated by an AGV the position and orientation of the physical and simulated AGVs are synchronized.

The synchronization of place proved to be even more difficult, because several representations of the AGV exist. We used five different representations of AGVs (Verbraeck et al. 2001). First of all, there are the actual positions of the prototypes of the AGVs. Secondly, there are emulated AGVs. The third and fourth representations are located in the simulation models, there is the logic representation and the animation representation. Finally, the CORBAWrapper has its own representation of the AGVs. All five 'different' AGVs have to be synchronized. Off course, the synchronization between the representation in the simulation logic and real prototype was the most vital. When these two representations differ too much from each other, crashes between physical AGVs are bound to happen. The animation was allowed to run ahead or behind from the other representations.

Another problem were the different *start-up sequences* of the simulated control system and emulated and prototypes systems. The start-up sequence can be seen as the initial synchronization. Again the initial synchronization in place proved to be more difficult than the initial synchronization of time. The simulation model starts empty without any resources. The real physical systems, however, starts with AGVs, docks, and loads. These are located at certain locations and have certain characteristics, e.g. an AGV has a position, orientation, and a possible load. To solve this we developed a special initialization protocol and startup procedure was developed for the TestSite. A web-based interface to the simulation model was used to enter AGVs into the simulated control system.

The last problem in synchronization was joining AGVs to, or removing AGVs from, the simulated control systems. In normal simulation models AGV are added in the model and remain there till the simulation experiments are finished. When controlling real prototype AGVs it is necessary to remove AGVS from the simulated control system and add them at a later stage. When large-scale automated logistic systems are operational they use several control systems in a distributed setting to control the AGVs. Each geographical area has its own control systems. This makes the control system scalable (Verbraeck and Versteegt 2001). This means however that AGVs will leave one control systems and enter another control system. In our approach we used a web-based interface that the operator can use to join or remove AGVs and loads.

6 EVALUATION OF SIMULATION PACKAGES

The logistic control was implemented in three different simulation packages. Only in Simple++ a full implementation was made. In Arena and AutoMod simplified implementations were modeled, mainly for testing of the communication protocols.

AutoMod and Simple++ had a clear advantage over Arena, because of the programming style interfaces. Complex logistic control rules could easily be implemented. All three simulation packages have an open structure and cooperation with other software packages and real prototypes of the logistic resources could easily be made.

Simple++ had one large advantage over AutoMod and Arena. Simple++ offers possibilities to construct object-oriented building blocks (Verbraeck and Versteegt 2000). In Simple++ we started with the construction of a library of components for both control and logistic system. This closely fits to the idea of interchangeable components.

AutoMod has a very open structure. The Model Communication Module offers many possibilities for AutoMod to cooperate with other software modules (Verbraeck et al. 2001). Furthermore, the standard built-in features that AutoMod offers for logistic control are very powerful.

The major disadvantage of all three simulation packages is that control system and system-being-controlled are strongly interwoven in the simulation languages. Real progress in real-time control and emulation can only be achieved when simulation packages implement a clear separation between control system and system-being-controlled. In the current versions this separation is not made, both are strongly interwoven. Clear well-defined interfaces between control systems and systems-being-controlled should also be provided.

The main advantages that simulation offers are the safe and fast testing of changes off-line. During the first experiments the AGVs often created deadlock situations (Versteegt and Verbraeck 2001). By studying the AGVs faster than real-time many deadlock situations could be quickly identified. The solutions for deadlocks could be safely be tested in a fully simulated environment. When the solutions proved to work adequately, they could easily be transferred to the emulated and prototype components.

7 CONCLUSIONS AND FUTURE RESEARCH

The extended use of simulation offered a number of advantages. The main advantages of out approach are based on the flexibility simulation offers. Changes in both control system and system-being-controlled could firstly be evaluated in a fully simulation environment. Here we could speed-up the time and study the effects of changes is a save environment. When the changes proved to work to well, they could easily be transferred to the real physical systems, because the interfaces are the same for both simulation and real systems.

The changeable components allowed us to test different aspect of the control system and logistic system in combined experiments, i.e. mechanical engineers could study the behavior of the physical scale models of AGV, while logistic expert could use simulated AGVs to evaluate control strategies. Changes between real components and simulated or emulated components were every easy to implement because of the clear interfaces that were defined.

The simulated control system proved to be able to work as a control system for the real-time control of physical logistic resources. A number of important questions still have to be answered. The first question is the scalability of the simulated control systems. At this moment we are able to control 10 AGVs and two material handling systems with our simulated control system. When the OLS Schiphol will implemented it will use up to 400 AGVs and 40 material handling systems. Can we still control such a large number of logistic resources with the simulated control systems, and still be able to meet the strict time-constrains.

Other experiments will focus on disturbances. It is very important that the simulated control systems are able to deal with disturbances. The simulated control system them has to detect disturbances and decide on a strategy to solve the identified disturbances. Within simulation models disturbances can easily be modeled. The simulation model simple 'turns off' a resource for some time. In controlling real prototypes the disturbances are, however, not initiated in the simulation models. Disturbances occur in reality, the simulated control system has to detect the disturbances, rather than initiating them. In simulation models disturbances are 'automatically' fixed after a while. In reality an adequate strategy has to be found to solve the disturbances.

ACKNOWLEDGMENTS

The authors would like to thank Henk Sol, Yvo Saanen, and Edwin Valentin of the Systems Engineering. Valuable contributions were given to us by: Jerry Banks, Ian McGregor, Joop Evers, Ben-Jaap Pielage, Gerrit-Jan Kleute, and Ruben van Miert. Furthermore, we acknowledge both financial and managerial support of Connekt for the TestSite.

REFERENCES

AESOP. 1999. *SiMPLE++ Reference Manual version 6.0*, Aesop Corporation, Stuttgart, Germany.

Aken, J.E. van. 1978. *On the control of complex industrial organizations*, doctoral dissertation, Martinus Nijhoff Social Sciences Division, Leiden.

Auinger, F., M. Vorderwinkler, and G. Buchtela. 1999. Interface driven domain-independent modelling architecture for "soft-commissioning" and "reality in the loop", *1999 Winter Simulation Conference*, IEEE.

Banks, J. 2000. *Getting started with AutoMod*, AutoSimulation, Bountiful, Utah.

Ben-Ari, M. 1990. *Principles of Concurrent and Distributed Programming*, Prentice Hall.

Kelton, W.D., R.P. Sadowski, and D.A. Sadowski. 1998. *Simulation with Arena*, McGraw-Hill.

Law, A.M. and W.D. Kelton. 1991. *Simulation modeling and analysis*, McGraw-Hill, New York.

Mueller, G. 2001. Using emulation to reduce commissioning costs on a high speed bottling line, *2001 Winter Simulation Conference*, IEEE.

Pyle, I., P. Hruschka, and M. Lissandre. 1993. *Real-time systems; investigating industrial practice*, John Wiley.

Schiess, C. 2001. Emulation: debug it in the lab-not on the floor, *2001 Winter Simulation Conference*, IEEE.

Verbraeck, A., P.L. Schroeder, and P.P. van Soest. 2000. Real-time simulation and control of AGVs in Arena; The design of a model in Arena 4.1 RT which can control multiple Automated Guided Vehicles on the Connekt Testsite in Delft, Report Faculty of Technology, Policy and Management, Delft University of Technology, Delft.

Verbraeck, A. and C. Versteegt. 2000. A bridge between the design and implementation of complex transporta-

tion systems; linking simulation models and physical models, *12th European Simulation Symposium*, pp. 238-243, Hamburg, Germany.

Verbraeck, A. and C. Versteegt. 2001. Logistic for fully automated large-scale freight transport systems, *2001 IEEE Intelligent Transportation Systems Proceedings*, pp. 774-779, Oakland (CA), USA.

Verbraeck, A., C. Versteegt, and G.J. Kleute. 2001. Real-time control of real AGVs using AutoMod, *Proceedings Brooks Automation European Simulation Symposium* 2001, Gent, Belgium.

Versteegt, C. and A. Verbraeck. 2001. Concepts for safety control for the concurrent use of infrastructure in intelligent transport systems using AGVS, *8th World Congress on Intelligent Transport Systems*, Sydney, Australia.

Versteegt, C., A. Verbraeck, and S. Geerdes. 2001. Simulation as a supporting tool for multidisciplinary design of underground freight transport systems, in: E. Taniguchi, R.G., Thompson (eds.), *City Logistics II, The Second International Conference on City Logistics*, p. 83-97, Okinawa, Japan.

AUTHOR BIOGRAPHIES

CORNÉ VERSTEEGT is a researcher in the Systems Engineering group of the Faculty of Technology, Policy and Management of Delft University of Technology. He specializes in logistics and logistic control systems. Currently, he is finishing this dissertation on controlling large-scale automated underground logistic systems. The research described in this paper is part of his dissertation research. His email and web-address are <cornev@tbm.tudelft.nl> and <www.tbm.tudelft.nl/webstaf/cornev>.

ALEXANDER VERBRAECK is an associate professor in the Systems Engineering Group of the Faculty of Technology, Policy and Management of Delft University of Technology. He is also a part-time professor at the University of Maryland. Alexander specializes in discrete event simulation, both for real-time analysis and control of complex logistic systems. Email and web-address are <alexandv@tbm.tudelft.nl> and <www.tbm.tudelft.nl/webstaf/alexandv>.