Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

SOFT-COMMISSIONING: HARDWARE-IN-THE-LOOP-BASED VERIFICATION OF CONTROLLER SOFTWARE

Harald Schludermann
Thomas Kirchmair
Markus Vorderwinkler

PROFACTOR Produktionsforschungs GmbH
Wehrgrabengasse 1-5, A-4400 Steyr, AUSTRIA

ABSTRACT

The basic idea of Soft-Commissioning (SoftCom) is to test industrial control software by connecting a controller, e. g. a PLC (Programmable Logic Controller) to a commercial discrete event simulator (DES), which provides system reactions and sensor signals similar to the behavior of real hardware, e. g. an industrial manufacturing line. In order to establish a connection between simulator and PLC, a modular architecture was developed. The basis of this modular system is a communication protocol common to all members. The two basic modules are the I/O Devices Driver (IODD), which is used to interface between the I/O hardware and the SoftCom protocol, and the Simulator to real World Interface (SWI). The SWI is used to link the simulator to the SoftCom system.

1 INTRODUCTION

Testing the behavior of a PLC, which controls a device being part of a more complex system, is usually done by connecting the controller to a 'stand-alone' version of the device, a so called mock-up. This method of verifying and validating the controller's software is expensive, and test conditions are hard to reproduce. The tester also has to put up with the fact that such tests are incomplete since the interaction of this device with the other parts of the system is simply ignored. Therefore a large part of testing and debugging is still carried out on-site. Both approaches rise additional costs and might lead to the destruction of the device or even endanger human life when testing the controller's reaction to critical situations.

Soft-Commissioning is a 'hardware-in-the-loop' (HIL)-based approach to solve such problems. HIL means that the inputs and outputs of a controller are connected to a simulation (emulation) of the part to be controlled. This enables better reproduction of test conditions and even allows the tester to reproduce the interaction of the various parts of the complete system. Whorter et al. (1997) indicate that using the same simulation model for system development and staff training can reduce costs and plant set-up times.

The main difference between Soft-Commissioning and other HIL systems are the required system reaction times: Soft-Commissioning is intended to operate with round trip times smaller than 100ms, while e. g. systems used in automotive industry need reactions in less than 1ms (Kiffmeier et al. 1997). Thus different hardware requirements apply: Soft-Commissioning shall execute on standard office Workstations, while most other HIL systems use fast 'Digital Signal Processor' (DSP) – boards to achieve the required signal response time (for an example see Hanselmann (1996)). Another difference is that Soft-Commissioning interacts with 'Discrete Event Simulators' (DES), while many HIL systems are based on continuous real-time simulation (an example is given by Boot et al. (1999)).

In section 2 this paper outlines the advantages and disadvantages of using 'off the shelf' I/O devices, simulators and operating systems. Section 3 gives the detailed description of the various elements of the SoftCom architecture. The paper closes with conclusions and gives an outlook on future research.

2 THE ENVIRONMENT

The SoftCom System was developed to interact with commercial simulators and conventional I/O hardware. Most of these programs and I/O devices run on 'general-purpose' operating systems such as Windows NT or UNIX/Linux. It is well known that these operating systems only provide soft-real-time capabilities, which means that the time delay, until an event is serviced, is not as deterministic as on hard real-time systems (Microsoft 1995). Overviews and tests of the NT real-time capabilities can be found in Ramamritham (1998), Mattern (1998), Cota-Robles (1999). Jones & Regehr (1999) depict that the overall system reaction time becomes even worse when certain hardware drivers are in use.

Real-time extensions such as INtime for Windows NT or RT-Linux are separate real-time operating systems where the actual NT or Linux OS run in one of the system's threads (Obenland et al. 1999, Heursch & Mächtel 2000). Thus a program like the simulator (or the I/O hardware device driver), which runs on NT or Linux, would not benefit from the real-time extension. (An interesting approach, where RT-Linux is used for data acquisition and repetition while the simulation runs on a separate real-time simulation computer, is described by (Ptak & Foundy 1998).)

Summing up, the simulator, being the most essential part of the SoftCom system, runs on a 'general purpose' operating system. This leads to the fact that the overall reaction time to PLC actions will be dominated by the operating system's latency time. This time not only depends on the Computer's hardware, but also on the current load and system configuration. Windows NT latency times on a single processor Pentium III 400MHz machine typically vary between 1ms to 100ms with runaways in the area of 3s coming up when the system starts to swap. Recent measurements on a dual Pentium III 300MHz machine with Windows 2000 Professional resulted in latency times of 1ms and below even under heavy load.

3 THE ARCHITECTURE

The SoftCom system was developed with the intention to provide a modular and scalable HIL architecture based on commercial discrete event simulators. Thus a special communication protocol was defined to make flexible SoftCom internal data exchange possible.

With this protocol as basis the first two components were developed. They are needed to establish the connection between PLC and simulator (Figure 1):

- The I/O Devices Driver (IODD) provides an interface to the I/O hardware to which the PLC is connected.
- The Simulator to Real World Interface (SWI) is used to enable communication between the simulator and other components of the SoftCom system.

Two additional programs were considered as useful as shown in Figure 2:

- The Virtual I/O system (VIOS), which can be switched into the signal path in order to provide simple signal pre-processing.
- The SoftCom Manager (SCM), which is responsible for system configuration and runtime control.

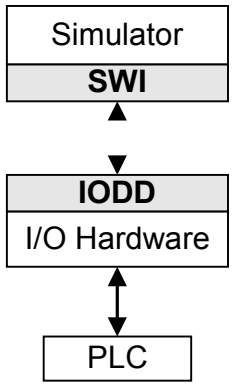

Figure 1: The SoftCom System is Based on Commercial Simulators and widely available I/O hardware

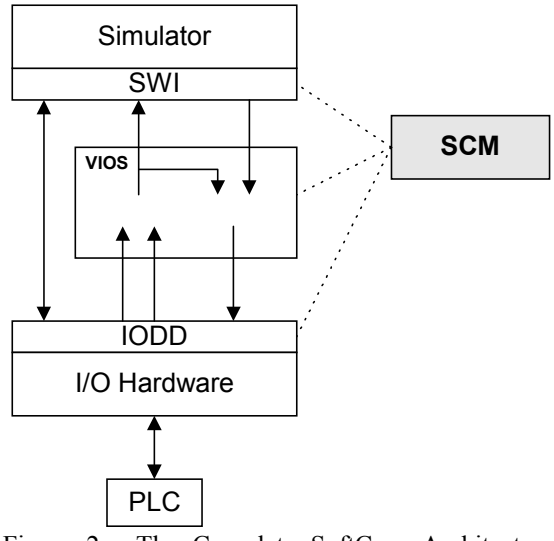

Figure 2: The Complete SoftCom Architecture with VIOS and SCM

The direct link between IODD and SWI in Figure 2 indicates that signals not being preprocessed may bypass the VIOS. The SCM maintains a link to all elements of the SoftCom system for configuration and runtime control.

3.1 The Communication Protocol

Data exchange between members of the SoftCom system is based on a special protocol set on top of TCP/IP. Using the TCP/IP standard allows the various parts of the SoftCom system to run on different computers and even on different operating systems such as Windows NT on computer A and B, and Unix on computer C as shown in Figure 3. If two programs run on the same computer (like on computer A), memory pipes are used to speed up communication.

Inter program connections are called 'Communication Channels' and are used for message exchange. Appending additional information to the messages enables the channels' endpoints to carry out redundancy checks. These

checks permit the detection of request-timeouts and transmission failures. The channel endpoints also bundle single messages to larger packages in order to reduce the overall network traffic (Figure 4).

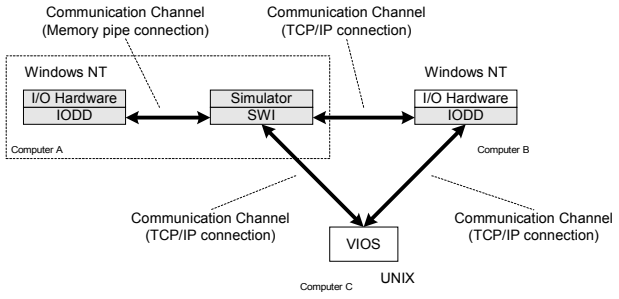

Figure 3: SoftCom is designed as Distributed Simulation And Test Environment

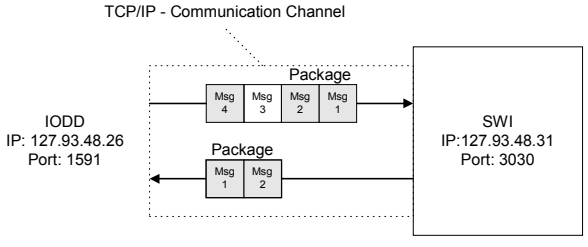

Figure 4: Two Elements of the SoftCom System Exchange Messages Bundled Together to Packages via TCP/IP

3.2 The I/O Devices Driver (IODD)

The inputs and outputs of the PLC-under-test are connected to the computer using standard I/O cards like 'Parallel I/O' or 'Field Bus I/O emulation'. These cards usually provide only simple and non-standardized access functions like read and write. Thus a hardware independent I/O Devices Driver (IODD) was developed to provide unified and versatile I/O access to the PLC.

The IODD internal link to the I/O cards is defined by a library interface, e. g. a Dynamically Linked Library (DLL) in the Windows world. The implementation of this interface depends on the I/O hardware in use. Thus the IODD must support connections to more than one library at the same time to be able to establish links to different I/O cards. Figure 5 is an example where two I/O cards (Parallel I/O card and Field Bus I/O emulation card) are connected to the IODD.

The IODD internal representation of a PLC input or output is called 'Pin'. A 'Pin Object' permits the definition of more complex access functions like triggered reading/ writing, sequence writing and active readout (Figure 6). Sequence writing means that a signal sequence is first stored in a buffer from which one sequence element after the other is triggered to the input line of the PLC. Active readout means that a Pin monitors its state and informs enlisted programs whenever changes are detected.

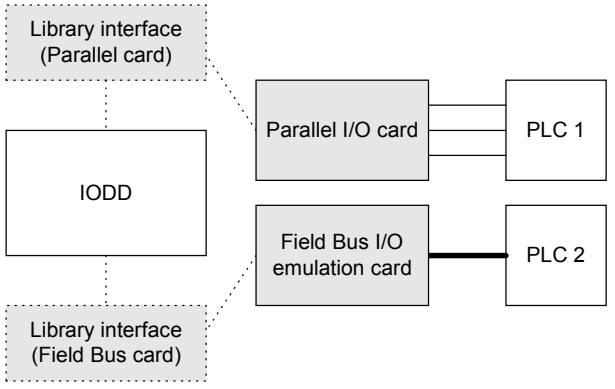

Figure 5: IODD connecting to two different I/O devices by making use of two Different Implementations of the Library Interface

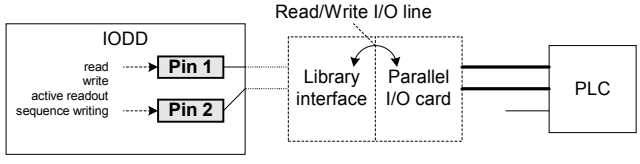

Figure 6: The IODD uses Pin Objects for the Internal Representation of PLC Input or Output Lines

3.3 The Simulator To Real World Interface (SWI)

The purpose of the SWI is to provide a communication interface between the simulator and the SoftCom system. The implementation of the SWI depends on the simulator's approach of providing access to its variables and objects (Straßburger 1999). Our prototype is based on the simulation environment Arena from Systems Modeling.

3.3.1 The Arena SWI

Arena provides two mechanisms for external programs to interact with the simulation. These mechanisms are the Visual Basic for Applications (VBA) module on the one hand, and a Dynamically Linked Library (DLL) interface on the other hand. Although the VBA module provides closer coupling to Arena, the DLL interface was chosen as the link between the SWI and the simulator. The reason for this decision is that the DLL can be implement in C++, which provides more flexibility in programming.

The DLL interface defines routines to interact with the simulation: One type gives access to simulator variables and the event calendar. The other type enables the simulation to execute user code when appropriate events are triggered. By forcing the simulation to call into the 'user event' DLL-routine periodically, a certain 'update' time step is achieved. This time step is needed to synchronize both simulation data and simulation time with the SoftCom system.

Simulation data is stored in two arrays, the input array (holding data sent from the SWI to the simulator) and the

output array (holding data to be sent from the simulator to the SWI at the next time step). SWI data is stored in objects called 'output pins' (for holding the data sent by the simulator) and 'input pins' (for the data to be sent to the simulator at the next time step).

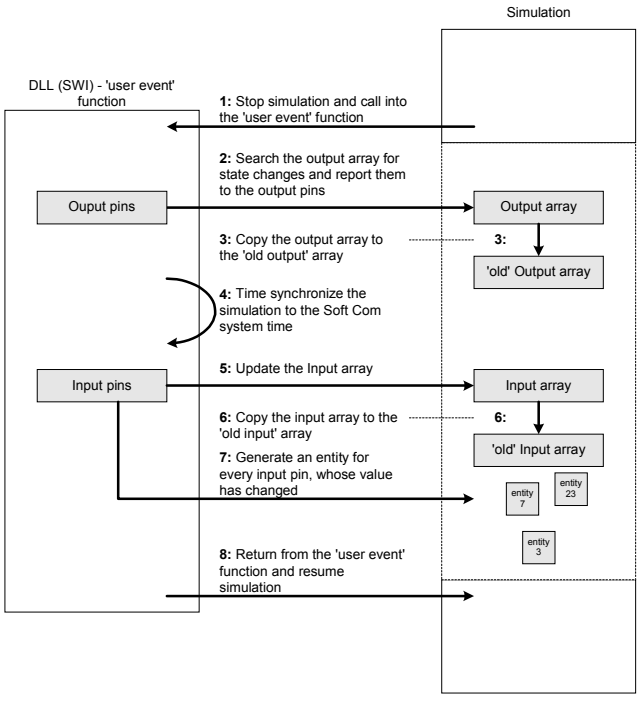

Figure 7: Order of events for Data And Time Synchronization between the Simulator and the SWI

The synchronization of data and time between SoftCom system and the simulator has the following order of events: Arena stops the simulation and jumps into the 'user event' function. This user-coded routine starts with searching the output array for state changes in order to report them to the 'output pins'. Next, the data of the output array is copied to the 'old output' storage to give the simulation access to values valid one time step before the actual one. Now the simulator is time-synchronized to the SoftCom system-time. This is done by forcing the simulator to sleep for the remaining time of this simulation time-step. After synchronization, the input array is copied to the 'old input' storage. Then, to relieve Arena from additional work, the SWI generates a simulation entity for every element of the input array that has changed. At the next simulation time step these entities will trigger elements of the Arena logic, and thus the evaluation of the new values. Finally the simulation run is resumed.

3.4 The Virtual I/O System (VIOS)

The VIOS was developed to provide a module which is capable of signal pre-processing. This adds a tool to the SoftCom family which relieves the simulation of doing low-level tasks like the logical or mathematical evaluation of signals.

Signal processing is defined within calculation units called macros (Figures 8 and 9). Such a macro consists of one or more objects, defined by their inputs, their mathematical operation, and their output. An input to this kind of object may be any external signal, the output of another macro, or the output of another operation object. The operation defined for such an object may either be one of Boolean nature ('AND', 'OR', 'XOR', 'NOT') or of 'mathematical' origin ('addition' 'subtraction', 'division', and 'multiplication'). A macro is restricted to contain either objects of the Boolean operator type, or objects of the 'mathematical' type. The interaction of the Boolean or 'mathematical' objects leads to the overall behavior of a macro.

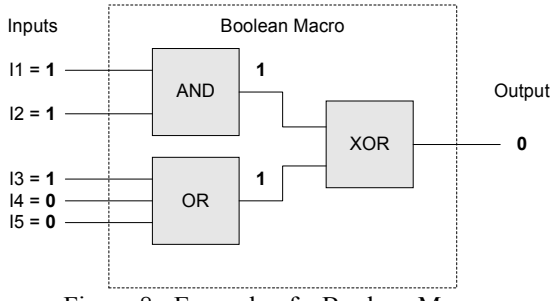

Figure 8: Example of a Boolean Macro

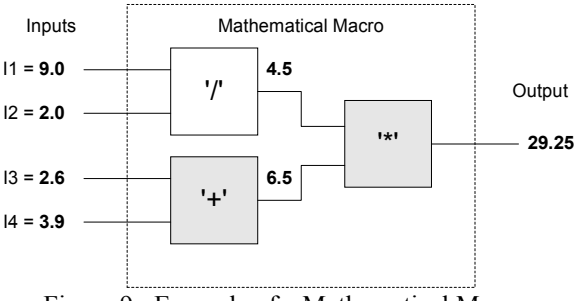

Figure 9: Example of a Mathematical Macro

Figure 8 gives an an example of a Boolean macro, which calculates the logical equation: Output = (I1 AND I2) XOR (I3 OR I4 OR I5). The example shows the result and the in-between states of the calculation.

In Figure 8 an example of a mathematical macro is given. This macro calculates the equation: Output = (I1/I2)*(I3+I4). The example also shows the result and intermediate values of the calculation.

3.5 The SoftCom Manager (SCM)

Tasks such as 'system update', 'system configuration' and 'runtime control' are usually quite simple as long as the system is restricted to a single computer. But they rapidly

grow in complexity with the number of computers involved. Thus configuration and maintenance of bigger systems can become a time-consuming job.

The strategy to simplify this job is to update and configure the system on a single computer, which then forwards this information to all other computers of the system. Based on this strategy, the SoftCom Manager was developed as a centralizing tool for configuration and runtime control.

3.5.1 System Configuration and Update

The configuration of the distributed system consists of four stages (Figure 10):

- In the first stage, the SCM connects to the remote computers, which are part of an actual SoftCom session. Using these connections the SCM then copies the required program files (including missing system files) to the target devices.
- In the second stage, the SCM starts all local and remote programs needed for the project. Each program sets up a direct link to the SCM (such a link is called 'manager channel'). After all links are established, the SCM time synchronizes the remote computers to the local time.
- In the third stage configuration data is exchanged between SCM and the other programs. This process also includes the registration of the communication objects (such as the IODD pins or some SWI pins) to ensure the uniqueness of their names. These names are used to identify the location of an object. The location information is needed when another object wants to set up a link to this object.
- In the fourth and last stage all communication links are analyzed and established. This is followed by a system wide 'start' signal.

Figure 10 is a small example for this configuration process. Elements of this sample-project are one SWI and one IODD, both running on a different computer (B) than the SCM (A).

The whole configuration with all active programs and their communication objects is visualized within a treelike structure inside the SCM. This makes reconfiguration simple and gives a structural overview of the system.

3.5.2 Runtime Control

The SCM gives the user the opportunity to display error messages and warnings reported by the various parts of a SoftCom project. It also enables the user to send commands like 'System Stop' or 'System Start' and reconfiguration messages throughout the system. The messages needed to fulfill these tasks are transferred via the connections originally set up for configuration.

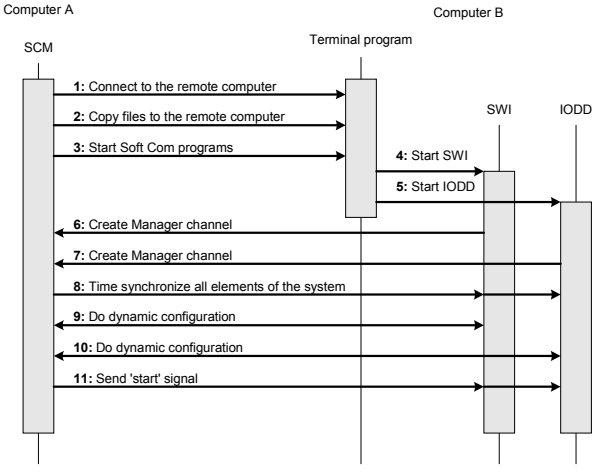

Figure 10: Timing diagram showing how the SCM Sets Up a Small SoftCom Project

4 CONCLUSIONS AND FUTURE WORK

Soft-Commissioning provides a modularized hardware-in-the-loop architecture, which runs on 'general purpose' operating systems. The TCP/IP based communication protocol used provides platform and location independence of the various parts. The following system members have been developed up to now:

- The IODD, which is used to establish a link between the I/O hardware and the SoftCom system
- The SWI, which is used to enable data exchange between the simulator and the system
- The VIOS, which can be switched into the signal path to provide fast signal pre-processing
- The SCM, which centralizes system configuration and runtime control

Future research might include the development of a 'High Level Architecture' (HLA) – module to gain access to the HLA's capabilities in the area of distributed simulation.

It is also intended to add communication channels based on 'high speed data transmission' protocols like ATM or ADSL. This would make greater distances between the various elements of the system possible without increasing the reaction time.

ACKNOWLEDGMENTS

The authors want to express their thanks to Ao. Univ.-Prof. Dr. Eugen Brenner, Technical University of Graz, who supported the development of the SoftCom Architecture with valuable suggestions.

REFERENCES

Albert, J., J. Tomaszunas. 1998. Rapid Prototyping von speicherprogrammierbaren Steuerungn an virtuellen Maschinen. *Industrie Management* 14 (1998): 34-38.

Baril, A. Baril. 1999. Using Windows NT in Real-time Systems. In *Proceedings of the 5th IEEE Real-Time Technology and Applications Symposium*, Vancouver, Canada, 2-4 June 1999.

Boggs, D. R., J. C. Mogul, C. A. Kent. 1988. WRL Research report 88/4 Measured Capacity of an Ethernet: Myths and Reality. In *Proceedings of the SIGCOMM'88 Symposium on Communications Architectures and Protocols*, Stanford, USA, 16-18 August 1988.

Boot, R., J. Richert, H. Schütte. 1999. Automatisierter Test von Steuergeräten in einer Hardware-in-the-Loop-Simulationsumgebung. In *Proceedings of Mess- und Versuchstechnik im Fahrzeugbau*, Mainz, Germany, 29-30 April 1999, 69-98.

Cota-Robles, E., J. P. Held. 1999. A Comparison of Windows Driver Model Latency Performance on Windows NT and Windows 98. In *Proceedings of OSDI'99 - Third USENIX Symposium on Operating Systems Design and Implementation*, New Orleans, USA, February 1999, 159-172.

DMSO, Defense Modeling and Simulation office. High Level Architecture Homepage. <http://hla.dmso.mil>

Dougall, D. 1999. Kürzere Inbetriebnahmezeiten bei reduziertem Risiko. *Industrie Report International* 8.9 (1999): 4-6.

Eppinger, A., C.A. Kricke, B. Ebinger. 1995. Interaktive Hardware-in-the-Loop-Simulation – Anwendungsgebiete im Steuergeräte-Entwicklungsprozeß, Interactive Hardware-in-the-Loop-Simulation – Applications in the ECU-Function Development Process. *VDI-Berichte* no. 1189 (1995): 91-107.

Feldmann, K., C. Schnur. 1998. A 3D-Kinematics-Based Simulation Tool For Testing Of Logic Control Software Of Production Systems. In *Proceedings of the 10th European Simulation Symposium and Exhibition*, Nottingham, UK, 26-28 October 1998.

Hanselmann, H. 1998. Beschleunigte Mechatronik – Entwicklung durch Rapid Control Prototyping und Hardware-in-the-Loop Simulation. *at-Automatisierungstechnik* 46 (1998): 3.

Hanselmann, H. 1996. Hardware-in-the-Loop Simulation Testing and its Integration into a CACSD Toolset. In *Proceedings of the IEEE International Symposium on Computer-Aided Control System* Design, Dearborn, USA, 15-18 September 1996, 152-156.

Heursch, S., M. Mächtel. 2000. Linux in Eile. Magazine Article, January 2000.

IEEE Std 802-1990. 1990. *IEEE Standards for Local and Metropolitan Area Networks*: Overview and Architecture. 31 December 1990.

Jones, M.B., J. Regehr. 1999. The Problems You're Having May Not Be The Problems You Think You're Having. In *Proceedings of HotOS-VII - 7th Workshop on Hot Topics in Operating Systems*, Rio Rico, USA, March 1999.

Kiffmeier, U., R. Otterbach, H. Schütte. 1997. Real-Time Simulation of a 3-D Vehicle Dynamics Model on the DEC Alpha Processor. In *Proceedings of the 15th IMACS World congress on Scientific Computation, Modeling and Applied Mathematics*, Berlin, Germany, 24-29 August 1997, vol. 6 Application in Modeling and Simulation, 463-468.

Klein, U., S. Lange, S. Straßburger, K. C. Ritter, R. Jesse, T. Schulze. 1999. Integration von Echtzeit-Online-Informationen in verteilte Simulationen auf Basis der High Level Architecture. In *Proceedings of Simulation und Visualisierung*, Magdeburg, Germany, 4-5 March 1999.

Klumpf, M., C. A. Kricke, B. Ebinger. 1999. Automatisierter Test von Steuergeräten in der Automobilindustrie, Automated Test of Automotive ECUs. *VDI-Berichte* no. 1470 (1999): 69-98.

Kweon, S. K., K. G. Shin, Q. Zheng. 1999. Statistical Real-Time Communication over Ethernet for Manufacturing Automation Systems. In *Proceedings of the 5th IEEE Real-Time Technology and Applications* Symposium, Vancouver, Canada, 2-4 June 1999.

Luchetta, A., G. Manduchi. 1999. General Purpose Architecture for Real-Time Feedback Control in Nuclear Fusion Experiments. In *Proceedings of the 5th IEEE Real-time Technology and Applications Symposium*, Vancouver, Canada, 2-4 June 1999.

Mattern, D. 1998. "Soft" Real-Time Applications Under Windows NT. In *Proceedings of the ASME Dynamic Systems and Control Division*, 15-20 November 1998.

Meier, H., K. Kreusch. 1998. CNC-Test an virtuellen Werkzeugmaschinen. *Zeitschrift für wirtschaftlichen Fabrikbetrieb ZWF,* 93(1998): 415-417.

Microsoft. 1995. Real-Time Systems And Microsoft Windows NT. *White paper, Microsoft Developer Network*. 29 June 1995.

Obenland, K. M., T. Frazier, J. S. Kim, J. Kowalik. 1999. Comparing the Real-time Performance of Windows NT to an NT Real-time Extension. In *Proceedings of the 5th IEEE Real-time Technology and Applications Symposium*, Vancouver, Canada, 2-4 June 1999.

Ptak, A., K. Foundy. 1998. Real-Time Spacecraft and Hardware-in-the-Loop Testing. In *Proceedings of the 4th IEEE Real-Time Technology and Applications Symposium*, Denver, USA, 3-5 June 1998.

Ramamritham, K., C. Shen. 1998. Using Windows NT For Real-Time Applications: Experimental Observations And Recommendations. In *Proceedings of the 4th IEEE Real-Time Technology and Applications Symposium*. Denver, Denver, USA, 3-5 June 1998.

Rieger, K. W. Schiehlen. 1995. Echtzeitsimulation eines Fahrzeugmodells mit aktiver Federung – Hardware-in-the-Loop Experimente, Realtime Simulation of a Vehicle Model with Active suspension – Hardware-in-the-Loop Experiments. *VDI Berichte* no. 1189 (1995):17-34.

Sailer, U. Essers. 1995. Modulare Interface-Elektronik zur Kommunikation zwischen einem Echtzeitsimulator und real vorhandenen Bremselektroniken (Modular Interface Electronics for the Communication between a Real-Time Simulator and Electronic Braking Control Units). *VDI-Berichte* no. 1189 (1995): 55-69.

Schulze, T., U. Klein, S. Straßburger, H. P. Menzler. 1998. Traffic Simulation based on the High level Architecture. In *Proceedings of the 1998 Winter Simulation Conference*, Washington, USA, 13-16 December 1998, 1095-1103.

Spörl, T., B. Ebinger, B. Heppner, W. Sienel. 1998. Einsatz von Hardware-in-the-Loop Testsystemen in der Getriebesteuerungsentwicklung. *VDI-Berichte* no. 1393 (1998): 417-432.

Spurgeon, C. 1993. Network Reading List: TCP/IP, UNIX and Ethernet. Document Version 4.0, June 1993.

Straßburger, S. 1999. On The HLA-Based Coupling Of Simulation Tools. In Proceedings of the ESM'99 - 13th European Simulation Multiconference, Warsaw, Poland, 1-4 June 1999, 45-51.

Vorderwinkler, M., T. Eder, R. Steringer, M. Schleicher. 1999. An Architecture for Soft-Commissioning, Verifying control software by linking discrete event simulators to real world control systems. In *Proceedings of the ESM'99 - 13th European Simulation Multiconference*, Warsaw, Poland, 1-4 June 1999, 191-198.

Whorter, S., B. Baker, G. Malan. 1997. Simulation System For Control Software Validation. In *Proceedings of the 1997 SCS Simulation Multiconference,* Atlanta, USA, 6-10 April 1997.

AUTHOR BIOGRAPHIES

HARALD SCHLUDERMANN is with Profactor Produktionsforschungs GmbH, a non- profit manufacturing research institute located in Steyr, Austria. As a student of Telematics at the Technical University of Graz he is currently working on the development of the SoftCom architecture in connection with his diploma thesis. His email address is <Harald.Schludermann@profactor.at>.

THOMAS KIRCHMAIR is researcher at Profactor and responsible for the company's software development and implementation activities. His research interests include object orientated software engineering, industrial image processing and development managment. He received a Dipl.-Ing. degree in Communication and Radio-Frequency Engineering from Vienna University of Technology. His email address is <Thomas.Kirchmair@profactor.at>.

MARKUS VORDERWINKLER is responsible for the simulation activities at Profactor. His research interests include simulation based system analysis, design and optimization, advanced robotics and industrial control systems. He received a doctorate in Electrical Engineering from Vienna University of Technology. His email address is <Markus.Vorderwinkler@profactor.at>.