Proceedings of the 1999 Winter Simulation Conference
P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

INTERFACE DRIVEN DOMAIN-INDEPENDENT MODELING ARCHITECTURE FOR "SOFT-COMMISSIONING" AND "REALITY IN THE LOOP"

Franz Auinger
Markus Vorderwinkler
Georg Buchtela

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

ABSTRACT

As industrial manufacturing and automation systems grow in complexity, there is a need for control software engineering support. *Soft-Commissioning* and *Reality in the Loop (RIL)* are two novel approaches which allow coupling simulation models to real world entities and allow the analyst to pre-commission and test the behavior of a system, before it is completely built in reality. To be flexible and fast in building up a simulation model fulfilling the requirements of Soft-Commissioning and RIL there is a need for a component-based modeling architecture. In this paper we define the characteristic requirements, and derive an architecture out of them, which is discussed from different aspects. Finally we briefly present a simple example.

1 INTRODUCTION

Designing, implementing and maintaining industrial control systems requires a deep understanding of the controlled processes, their structure and their behavior.

1.1 Motivation

During the *planning and implementation phases*

- ➔ Control engineers have to rely on design and implementation information provided by engineers of other disciplines. This data are often not delivered in the required form and seldom on time.
- ➔ It is very difficult to test and pre-commission a control system (hardware and software) before implementing and coupling it with the plant to be controlled.
- ➔

Therefore control programs are often implemented and finished right on the spot during the plant startup phase, which is not only expensive but also risky and error-prone.

During the *maintenance phase* of a plant's life-cycle it is a challenge to modify a control system with minimal down times. For example, testing modified control strategies forces down-times for implementation, tests and commissioning of a plant. Because these down-times are unproductive they are again an expensive side effect of maintaining technical systems.

Another motivation comes from the fields of Intelligent- and Holonic Manufacturing Systems (IMS, HMS) (Van Brussel 1994). In these areas autonomous, intelligent agents negotiate how they intend to process a specific order and reconfigure themselves according to their capabilities and the currently required functionality (overview in Shen and Norrie 1999). In such systems, it is difficult to predict the entire system behavior after a reconfiguration. Bodner and Reveliotis (1997) deals with a similar problem in the domain of Flexible Manufacturing Systems and tries to evaluate different system configurations and control policies with an object oriented simulation based framework ("Virtual Factories").

Especially to evaluate IMS/HMS based control policies it is necessary to model both the plant's hardware and its control software (strategical and shop floor layers) in great detail. In order to avoid extra work (re-modeling of the strategical layers), it seems useful to use the final control software of the strategical layers for testing and evaluating. Therefore it is desirable to link the existing parts of the control framework (strategical layers) to a detailed simulation model of the plant hardware.

1.2 State of the Art

As a general solution we propose to use a combination of simulation and "real world sections" for a full lifecycle-support for control system development and maintenance (Figure 1).

"Pure" simulation is based on information models and architectures derived from "information systems". Therefore many authors suppose the use of hierarchical and/or object oriented methods for modeling (example in Praehofer 1996, overview in Marayanan et al. 1998). Because of different

underlying architectures and requirements of "Automation Systems" in comparison to "Information Systems" (as pointed out in Kiesz 1995) it seems necessary to modify and adapt these methods and techniques.

Smith et al. (1994) and Praehofer et al. (1994) already proposed coupling simulation with shop floor equipment. But these approaches aim to control and monitor the machinery by the simulation directly and both approaches are tailored to the domain of Flexible Manufacturing Systems (FMS).

1.3 Objectives of this Work

In this paper we introduce a generic, multidimensional modeling architecture, suitable for combining simulation modules with real world objects.

1.4 Sections

In the following section we give a brief overview of the ideas and structures behind the concepts of *"Soft-Commissioning"* and *"Reality in the Loop" (RIL)*. A short discussion of the concepts will establish the major requirements on the desired architecture.

Based on these requirements we *identify independent aspects* for structuring, separating, modeling and building a system.

In section *4"* Architecture*"* we introduce and describe the general modeling architecture using UML notation.

In the section *5* " Application to problem domain*"* we derive a set of demands on a simulation system to be able to depict the architecture to simulation models and we discuss the characteristics of our solution and identify particularly qualified industrial domains.

Then we apply our architecture to the specific problem domain of flexible manufacturing systems and provide in section 6 "A Simple Example" of a pallet transfer system and its control.

Finally we briefly show how the presented work is related to a broader research effort in the field of industrial control engineering for manufacturing systems design and implementation.

The presented work is a part of the *Soft-Commissioning* (Vorderwinkler et al. 1999) and *Reality in the Loop (RIL)* (Graebe et al. 1997) projects which are R&D efforts from PROFACTOR® Manufacturing Research and are described in the following section. In addition it is a contribution to the HMS (Holonic Manufacturing Systems) project. The *HMS* project is an international research project funded by the European Union (Proj. No. IM-26508).

2 CONCEPTS OF "SOFT-COMMISSIONING" AND "REALITY IN THE LOOP"

Testing and debugging of PLC based control software is still a time consuming and expensive task, which in many cases requires full access to the final system hardware (Figure 1: Arrow 1) and therefore postpones software tests towards the end of a typical automation project.

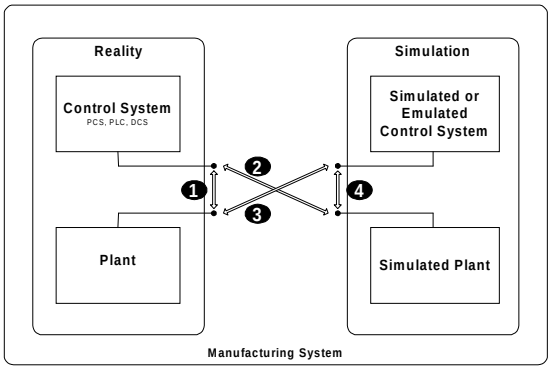

Figure 1: Possible Combinations between Reality and Simulation; the arrows represent possible connections. Arrow 1: traditional way to test control systems; Arrow 2: Soft-Commissioning; Arrow 3: Reality in the Loop; Arrow 4: Off-line Simulation.

To gain in flexibility and to support early testing of control software, (Vorderwinkler et al. 1999) introduced the idea of coupling the real control system with a simulation model of the plant (Figure 1: Arrow 2). He suggests to reuse simulation models of prior planning phases (e.g. material flow simulation) as shown in (Figure 1: Arrow 4) for control development and diagnostics. He also presents a layered software architecture for coupling a simulation program (in his case a discrete event simulator) to a control device such as PLC, PCS or DCS based on a virtual IO System Layer (VIOS).

On the other hand, Graebe et al. (1997) presented a decision support tool based on including "Reality in the Loop", in which real machines and workplaces can be synchronized and incorporated into a simulation of the remaining process (Figure 1: Arrow 3).

Both RIL and Soft-Commissioning generate a set of demands on the simulation models which are listed in the following section.

2.1 Requirements

R1 First of all the system to be analyzed has to be subdivided into a (simulation) model of the plant (e.g. machinery) and a (simulation) model of the control system (e.g. PLC, PLC software etc.) in order to replace one component by reality.

R2 To gain inter-project re-usability within a specific (engineering) domain, the simulation modules have to depict the natural composition hierarchy and the level of granularity of the specific application domain.

R3 To allow a hierarchical composition of compound models out of atomic models as well as to have a clear separation between the visualization (animation) and

the physical model of an object, and to be neutral to domain specific graphical representations, the visualization of the simulation needs to be separated from the physical model. The visualization should be attachable to a specific behavioral module (and therefore to its specific level of abstraction).

R4 Each object to be simulated must be described in a sufficiently complete set of modeling dimensions (e.g.: structure, behavior, state) which captures the properties of the real world representation in a form, which makes the simulated object behave like a real world object with regard to its interfaces.

R5 To achieve an open solution, the architecture should be independent of the chosen simulation formalism (discrete, continuous, multiformalism).

R6 To make real components and simulation modules fully pluggable against each other the simulation models have to provide interfaces similar to the interfaces of the real world plant. Ideally an input or output in the simulation model may "directly" be attached to the output of a sensor or the input of an actuator. As Vorderwinkler (1999) focuses on the interfacing between hardware and simulation, we don't deal with this topic here but focus on the logical interface architecture.

R7 To be open for extensions of domain specific module libraries, the modules need to be refineable. Nevertheless all of their interfaces must retain compatibility to former versions of the modules. This also applies to state variables, which contain information about the global behavior of the system.

R8 To provide reusability of simulation models over multiple project stages (requirements analysis, design, implementation, test, commissioning, maintenance etc.) the models have to be accessible and coupleable at different levels of abstraction and over different simulation formalisms.

R9 To be enable synchronization of the real world system states with the simulation system states, there has to be a general interface for initializing start up conditions.

R10 Interfaces of simulation components to real world components only cover the vertical interaction (see Figure 1) between control (-model) and plant (-model). To communicate their actual physical state to other components with which they are coupled through their (simulated) physical representation and physical laws, the models need to provide generic interfaces.

Although this list is far away from being complete one can deduce that the different items belong to different problem dimensions.

3 IDENTIFIED ASPECTS AND THEIR CLASSIFICATIONS

By applying the *"Separation of Concern"* principle to our problem (see also Czarnecki 1996), we get different views on our system. We now describe the properties of our solution by identifying different concerns of the previously listed requirements:

3.1 Decompositional Aspect Regarding Control and Visualization

Here we describe how a problem of the automation domain can be structured in a generic way, so that the resulting structure is independent of the specific application domain and is compliant to R1, R2 and R3 of the previous section.

Our architecture was derived from a very generic control engineer's point of view. The "manufacturing system" in Figure 2 is structured as an ordinary control loop. We subdivide the whole manufacturing system in a control system, the system which is controlled – called "plant" – and interfaces (sensors and actuators) between them. Humans interact in different ways with the system (directly or via an *HMI – Human Machine Interface*). The HMI can also be seen as a kind of interface, because it consists of (virtual and/or real) "switches" – the sensors – and actors in the form of single indicators (e.g. LEDs etc.) or in the form of more universal information displays(e.g. computer monitors). In other words it transforms physical interaction into control signals and vice versa, which will be of importance in section 4. In a similar way, the human operator can be interpreted as a control system and therefore as part of a superimposed control loop (which is the highest level of abstraction). On the other hand, the plant or the control system may contain autonomous subsystems which have their own control system, interfaces, plant and so on (higher levels of detail) (see Figure 2). Out of that we identify three different basic

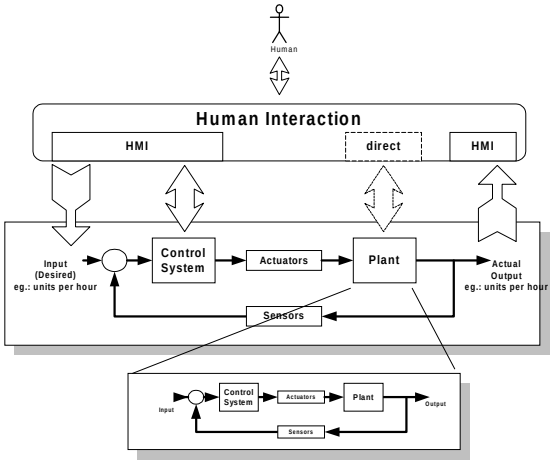

Figure 2: A manufacturing system can be seen as a control loop with human interaction to different sections of the loop. For Soft-Commissioning and RIL purposes this decomposition seems intuitive and can be continued hierarchically and in different levels of abstraction.

modeling component types namely *Control System Component*, *Interface Component*, and *Plant System Component*, which are introduced in Figure 3. Every Soft-Commissioning/RIL atomic- and composite component has to be modeled such that it can be attached to exactly one of the classification types of Figure 3.

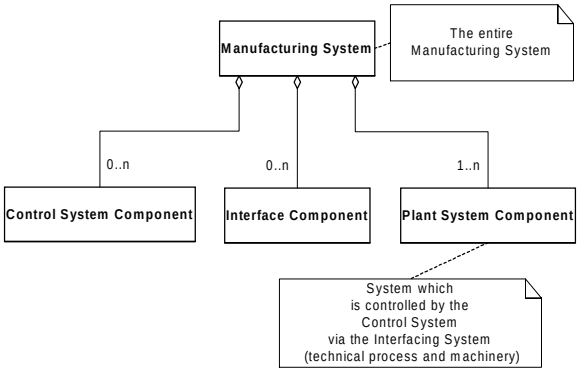

Figure 3: Modeling component types for a manufacturing system (Remarks: A human may be seen as *Control Component* and the HMI may be seen as *Interface Component*).

3.2 Description and Representation Aspects

Here we define a complete set of modeling views, which is capable of depicting all essential static, dynamic, and compositional properties of a real world object.

As suggested by multiple sources in the literature (Wang et al. 1998), a component (as an object) may be described by a set of normalized views such as structure view, behavior view and states. In Figure 4 these views are shown in a semantic net and are specified further. Here we will not focus on how to describe the different views during a component's life-cycle and in a simulation system because this depends on the modeling technique as well as the simulation system and the implementation language chosen. A description of a real world object must be capable of describing all of the suggested representations (structure, behavior, state) in order to be complete and executable.

Furthermore (according to R3), we strictly divide the visualization of the simulation from the physical simulation model by applying the observer pattern (Gamma et al. 1995) to our problem (see Figure 4).

Figure 4 also points out, that a model description will also contain additional informal representations such as help texts, documentation and so on, which would be attached to a model component.

3.3 Compositional and Interfacing Aspects

Requirements R3 and R6-R10 express demands on compositioning and interfacing aspects. Here we describe how the different module types (real world objects, simulated objects, Human Machine Interface, Simulation Visualization etc.) interact and how they can be combined (compositional aspects) (see also Figure 2: Decompositional Aspects).

Basic work for interfacing simulation components (regarding to R8) using different formalisms (discrete, continuous and multiformalism) is done in (Zeigler and Praehofer 1997) and will not be discussed here.

To have a clear distinction between different interfacing concerns we distinguish three bi-directional interface types.

- ➔ *Control Interface:* Its purpose is to provide an interface similar to the real world interface between a control system and a plant (compare R6), where it rather covers the tasks of (electrical) signal transportation (e.g. inputs and outputs of a PLC, electrical connections of a position switch etc.). (If simulation components should be coupled to electrical sensor connectors, the electrical signals have to be converted to a form which can be interpreted by the simulation module. This is done by the Soft-Commissioning Environment described in Vorderwinkler (1999). This interface cannot access internal states directly (comparable to encapsulation in object-orientation).

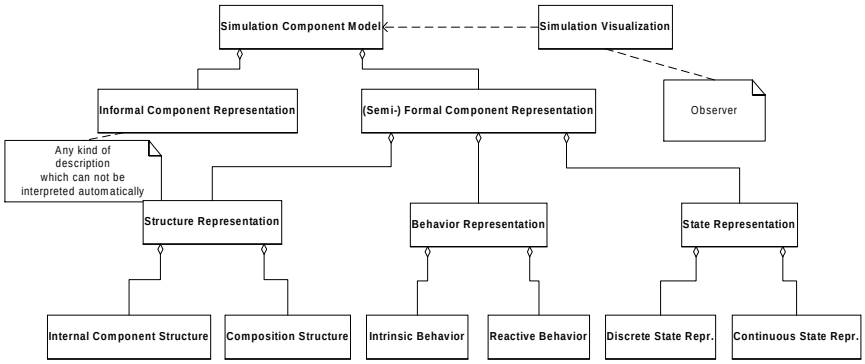

Figure 4: Descriptive dimensions of a simulation component.

- ➔ *Physical Interface:* Similar interface which has no direct access to internal states. Its purpose is to depict the physical coupling and the physical interaction between simulation components as claimed in R10. For example, if a pressure sensor is installed on a high pressure tube, the model of the sensor is connected to the model of the tube via the physical interface. A part of the interface coupling defines statical values such as position of the sensor on the tube whereas another section describes the dynamical coupling (coupling of differential equations etc.).
- ➔ *SoftCom-Interface:* The SoftCom-Interface is the only interface with direct access to the state representation. As demanded in R9 it is used to set state variables to a defined status (starting conditions etc). On the other hand (R3), it is accessed (reading access) by the simulation's visualization and statistical modules and is a prerequisite for the application of the observer pattern.

Figure 5 describes the relationships between the interfaces and a generic simulation component for our needs. Besides that, Figure 5 explains that a component consists of a state representation and a behavior representation. It also indicates that in our context components may be composed of multiple other (atomic or also composite) components. This property is sometimes referred as "self-similarity" in literature (VanBrussel et al. 1998). Furthermore a component may be connected to other components (via its interfaces) at the same level.

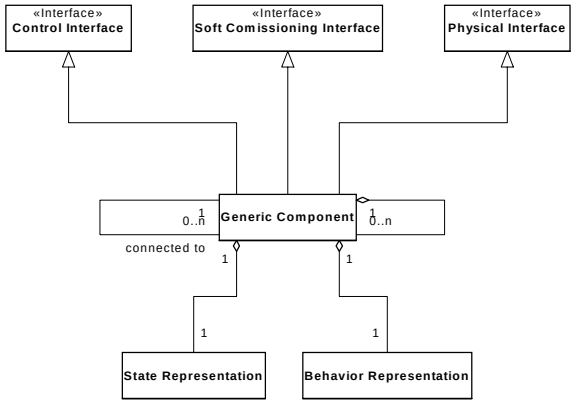

Figure 5: This UML Class diagram describes how a simulation component can be composed out of atomic components or out of composite modules. Furthermore it shows that every component inherits a set of defined interfaces and is described by a state and a behavior representation.

4 ARCHITECTURE

Combining the results of the last sections, we find the architecture shown in Figure 6. A typical component type as classified in Figure 6 will not use all of the interfaces shown in Figure 6. For example a sensor for pressure may have one physical input for „pressure" and one control output.

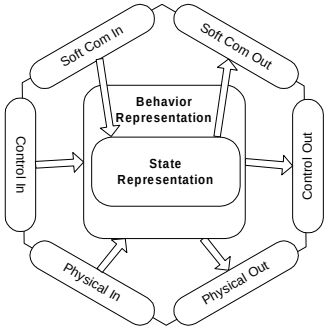

Figure 6: shows the generic architecture of a simulation component in its atomic form suitable for Soft-Commissioning and RIL. Where the control and the physical interfaces can only indirectly influence the system states, the Soft-Commissioning Interface allows to get and change the components state directly.

The compositional aspects which were structurally described in Figure 5 are outlined in Figure 7. One can observe that composite components are not only an encapsulating cover of other components and that they may have their own state and behavior description in parallel to the encapsulated components.

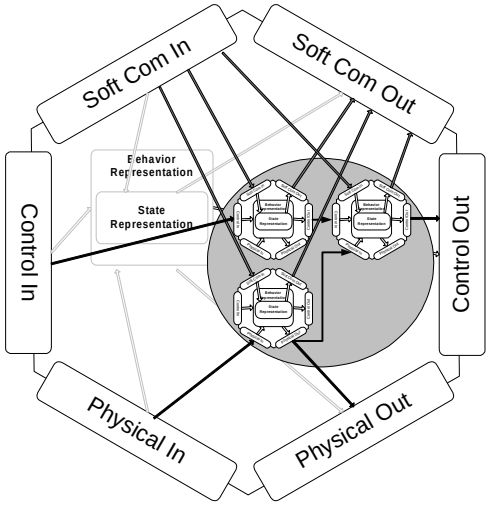

Figure 7: A component can be composed of other (atomic or again composite) components at different levels.

5 APPLICATION TO PROBLEM DOMAIN

5.1 Demands on a Simulation System

The developed architecture generates a set of demands on a simulation system:

- ➔ Visualization must be totally separable from the physical model
- ➔ The simulation system must provide real-time capability and access to the event-scheduler/time basis (for continuous simulation)
- ➔ The simulation system has to provide an open, programmable interface to other applications

5.2 Possible Application Domains

In principle the architecture is expected to be open enough to be independent of an application domain (Figure 8). However, one limiting factor might be complexity of the real world behavior. If processes get too complex to be modeled accurately, one should carefully ask what outcomes he expects from building the simulation model and then choose the economically sensible level of detail for the simulation model that will fulfill the expectations of the study. On the other hand, we state that every problem domain which (re)uses standard components (hardware, software, specification, implementation etc.) is especially qualified for our architecture.

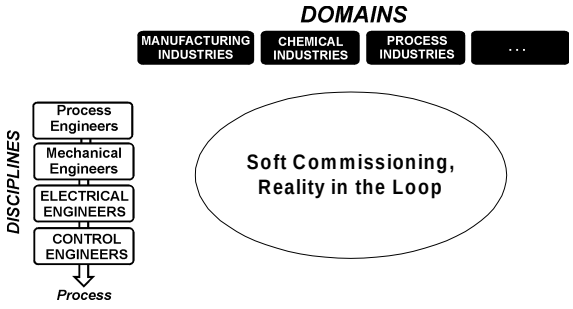

Figure 8: The architecture should be applicable to multiple domains and should integrate the engineering disciplines participated in a plant design.

Two typical application fields are

- ➔ *Discrete Processes* as Holonic Manufacturing Systems (to solve the problems as described in the introduction of this paper) or Flexible Manufacturing Systems (FMS)
- ➔ *Continuous Processes* (wherever Simulation makes sense and has been applied in the past)

6 A SIMPLE EXAMPLE

6.1 Simulation of a "Pallet Transfer System"

For the first evaluation of the presented architecture we selected a system of manageable size which could be analyzed and fully understood in detail, but which is still of moderate complexity.

To meet these requirements we have chosen a pallet transfer system which conveys workpieces from one assembly station to the next. The basic functions of this PLC-controlled system are quite simple. However, an overlapping material flow requires a relatively complex control strategy. With respect to keep downtimes short, it makes sense to fully validate the software before loading it down to the PLC. Figure 9 shows the layout of the transfer system.

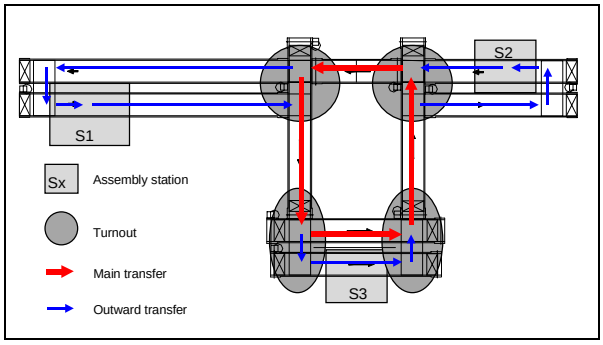

Figure 9: PLC-controlled pallet transfer system, used as testbed for the Soft-Commissioning Architecture.

The actuators which are controlled by a commercial PLC are mainly stoppers and lifters along the transfer lines which control the flow of the pallets through the system. Inductive sensors before each actuator indicate arriving pallets and act as input signals to the controller. To distinguish different product types all pallets are equipped with inductive identifiers which may be read by the PLC on request. Further I/O-signals establish communication links between the transfer system and the assembly stations which are used to synchronize the transfer and assembly processes.

A picture of the Soft-Commissioning environment developed at Profactor is given in Figure 10. Element (1) shows the enhanced simulator executing the process model and the real-time animation of the transfer system. Element (2) shows the SoftCom-interface used to link the simulator to an external controller. Finally, element (3) shows the PLC used to control the transfer system and which is connected to the interface via its parallel I/Os.

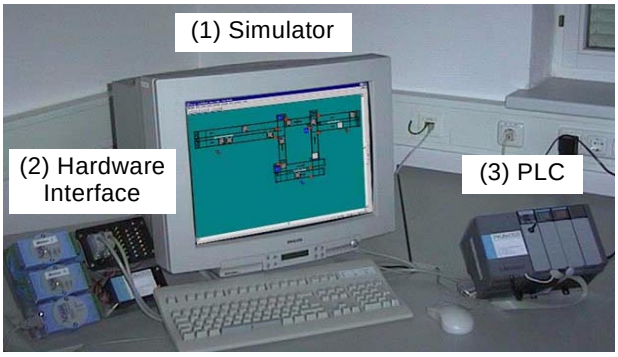

Figure 10: Picture of a minimal Soft-Commissioning Environment

6.2 Modeling the Transfer System

First of all, a classical simulation model of the entire transfer system was developed based on the discrete event simulation package ARENA® (Kelton et al. 1998). This first model containing the whole control logic was primarily intended to design and optimize the manufacturing process. For example, the minimum of pallets to achieve an optimal product schedule had to be determined before developing the final control strategy.

Based on the optimized process, both the control software for the external PLC as well as enhancements to the simulation model were developed in the further steps. Principal modifications to the model were made by separating the process and the control logic. In fact, the model used for Soft-Commissioning does not contain control logic any more and acts more or less as a detailed image of the physical process providing realistic sensor and actuator signals for the external PLC. To provide a visual feedback of the controlled process for presenting and debugging a schematic animation of the transfer system including the virtual sensors and actuators was developed.

Stepping back to the ARENA's simulation language SIMAN, it was possible to separate the process and control logic. To synchronize simulation time with real-time and to access the external hardware, ARENA's capability of calling C-routines was exploited. (A special add-on for ARENA called ARENA RT is available which supports real-time synchronization and message exchange over TCP/IP. However, for this pilot implementation ARENA RT was not used, in order to show a general solution applicable to any simulation package. ARENA RT also supports switching between internal simulation logic and external control logic simplifying the process of separating classical simulation logic and simulation logic based on external control.) Linking into the event scheduler of the simulator by one of the provided functions it was possible to synchronize the simulation clock as well as to update the internal simulation data with the external I/O signals.

6.3 Coupling Simulator and External Controller

After connecting the PLC to the simulator both the simulation model and the controller had to be initialized properly in order to reflect the status of the transfer system after power-on. In addition, the transfer system had to be filled with empty pallets by a special initialization logic.

On the basis of the traced system states, malfunctions could be allocated to a certain passage of the control software (e.g. a specific subroutine for a stopper etc.). If real-time is not necessary (entirely event driven control programs), the process may also be stepped through in single step mode with the benefit of watching all signal transitions.

Compared to tests with real systems, the simulated model has the advantage that one may replay almost any interesting or "suspicious" sequence until the reason for the observed behavior becomes clear. Using simulation we were free to set up almost any test scenario, including those, that would be risky for a real system.

In our first experiments, we quickly found some malfunctions of the stoppers caused by the usage of inverse logic which the programmer had overseen due to time pressure. In further experiments, the efficiency of the material flow was analyzed under real control. The first statistics indicated some problems at the turnouts. A pile-up was caused by priority control used for the conveyor intersections. It proved, that the control rules themselves were be programmed correctly, but some assumptions regarding the turnouts made during the concept phase were wrong. Such failures in the control program would not have been easily detected if using traditional evaluation methods, and in most cases, would appear only after final implementation. Especially for such logical failures, the combination of process simulation and real controller is an optimal method to detect them by a combined logical and logistical check.

7 CONCLUSIONS

In this paper we have developed a generic modeling architecture for coupling "real world" modules to simulation modules as an underlying modeling framework for "Soft-Commissioning" and "Reality in the Loop". Although the presented approach produces a set of demands on a simulation system to implement it, it is generally independent of the simulation formalism and thus can be applied to continuous, discrete and multiformalism simulation systems.

A first test case applying our modeling architecture to an pallet transfer system proved the suitability of our concept. However the modularization according to the developed SoftCom structure was still a time consuming task using a commercial simulator.

8 FUTURE WORK

To spread the presented ideas and methods to real industrial applications, the discussed architecture has to be extended in order to support the whole life cycle of a technical system including requirements engineering and "classical" simulation analysis.

To make the methods more practical for applications a framework has to be developed which provides work flow support for the whole design process.

REFERENCES

Bodner, D. A., and S. A. Reveliotis. 1997. Virtual Factories: An Object-Oriented Simulation-Based Framework for Real-Time FMS Control. In *Proc. of IEEE International Conf. on Emerging Technologies and Factory Automation*, Los Angeles. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Czarnecki, K. 1996. Separation of Concerns – objektorientierte Frameworks und das generative Paradigma. In *Objektspektrum* (6).

Gamma, E., R. Helm, R. Johnson, J. Vlissides. 1997. *Design Patterns – Elements of Reusable Object-Oriented Software*. Addison Wesley.

Graebe S., M. Schleicher, R. Steringer, and G. Zeichen. 1997. Reality in the Loop; In *Proc. of MSD'97 – Manufacturing System Design.* Magdeburg.

Kelton D. W., R. P. Sadowski, and D. Sadowski. 1998. *Simulation with Arena*. McGraw-Hill, ISBN 0-07-561259-3.

Kiesz, J. U. 1995. *Objektorientierte Modellierung von Automatisierungssystemen – Software Engineering für Embedded Systems.* Springer Verlag, 1995.

Marayanan, S., D. A. Bodner, U. Sreekanth, T. Govindaraj, L. F. Mc Ginnis, and C. M. Mitchell. 1998. Research in Object-Oriented Manufacturing Simulations: An Assessment of the State of the Art. In *IIE Transactions*, 30(9), Sept., 795-810, Kluwer Academic Publishers.

Praehofer H., W. Jacak, G. Jahn, and G. Haider. 1994. Supervising Manufacturing System Operation by DEVS-based Intelligent Control; In *Proc of AIS'94*, Gainesville, USA.

Praehofer H. 1996. Object Oriented Modeling and Configuration of Simulation Programs; In *Proc of European Meeting on Cybernetics and Systems Research*, 259-264.

Shen W., and D. H. Norrie. 1999. Agent-Based Systems for intelligent Manufacturing: A State-of-the-Art Survey. In *Int'l Journal of Knowledge and Information Systems*, 1(2), 129-156. http: //www.acs.ucalgary.ca/~wshen/papers/ survey-abm.htm. [accessed 6/1999]

Smith J. S., R. A. Wysk, D. Sturrock, S. Ramaswamy, G. Smith G., and S. B. Joshi. 1994. Discrete Event Simulation for Shop Floor Control. In *Proc. of WSC'94 – 1994 Winter Simulation Conference*, 962-969.

Van Brussel H., J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters. 1998. Reference Architecture for Holonic Manufacturing Systems: PROSA; *Computers in Industry* (37), 255-274. Elsevier.

Van Brussel H. 1994. Holonic Manufacturing Systems, The Vision Matching the Problem. In *Proc. of 1st European Conf. on Holonic Manufacturing Systems*, IFW-Hannover, Hannover, Dec 1 1994.

Vorderwinkler M., T. Eder, R. Steringer, and M. Schleicher. 1999. An Architecture for Soft-Comissioning – Verifying Control Software by Linking Discrete Event Simulators to Real World Control Systems. In *Proc of ESM'99 – 13th European Simulation Multiconference*, Warsaw, June 1-4, 1999, 191-198.

Wang Binjun, Wie Ge, and Kegang Hao. 1997. Three-Dimensional Object-Oriented Model. In *Proc. of TOOLS'97*, Beijing, Sept. 1997.

Zeigler B.P., H. Prähofer. Interfacing Continuous and Discrete Models for Simulation and Control. In *Int. Conf. on Environmental Systems*, Denvers, Mass. USA, July 1998

AUTHOR BIOGRAPHIES

FRANZ AUINGER is a member of the research staff at Profactor Produktionsforschungs GmbH, a non-profit manufacturing research institute located in Steyr, Austria. His research interests include control systems, software engineering, holonic manufacturing systems and simulation. He received a Dipl.-Ing. in Electrical Engineering from Vienna University of Technology.

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 degree in Electrical Engineering from Vienna University of Technology and is a member of IEEE.

GEORG BUCHTELA is head of the Holistic Engineering Department at Profactor. He received a doctorate degree of engineering from Vienna University of Technology. Besides his interests in continuous simulation he is engaged in organization and optimization of industrial processes.