HIERARCHICAL INTERFACE-BASED SUPERVISORY CONTROL OF A BOTTLING PLANT

Jan H. Richter, Florian Wenck

Department of Process Automation, Hamburg University of Technology, Schwarzenbergstrasse 95, 21079 Hamburg, Germany,
wenck@tu-harburg.de

Abstract: In this paper, the application of the Hierarchical Interface-based Supervisory Control method to controlling a large scale manufacturing example is presented adopting a layered organizational hierarchy of a factory. The generic and easily maintainable implementation using programmable logic controllers, a personal computer and TCP/IP over Ethernet for communication is explained in detail. To assess the controlled plant performance, business parameters are estimated using a throughput diagram and utilized to derive directions for further improvement.

Keywords: Discrete-event systems, supervisory control, implementation, manufacturing, large-scale systems, programmable logic controllers, evaluation, remote control

1. INTRODUCTION

Present manufacturing requires complex production processes and high flexibility regarding product mixes. Modern consumers demand high quality products and short delivery. In the light of industrial applications in manufacturing, mathematically verified supervisory controls for discrete event systems (DES) become more important for two major reasons. First, increasing quality standards and reliable performance in the production process are important issues. Second, the conventional industry approach to supervisory control design, based on experience and trial-and-error without mathematical foundation, requires much effort. Correctness of a supervisory control in this context means on one hand to ensure desired behavior of the controlled system, on the other hand to meet some control specific properties, e.g. nonblocking.

The Supervisory Control Theory (SCT), introduced by Ramadge and Wonham (1989), provides algorithms to synthesize correct supervisory controls from a given plant model in the context of given specifications. The acceptance of SCT is mostly limited to academical and experimental applications (Balemi *et al.*, 1993; Brandin, 1996) and small-scale industrial examples (de Queiroz and Cury, 2002). The restriction to small-scale systems is mainly based on the computational complexity problem related to the calculation of a global plant model. Structured supervisory control approaches are necessary to avoid state space explosion.

This work deals with the application of the Hierarchical Interface-based Supervisory Control (HISC) approach, initiated by Leduc *et al.* (2001*a*), to a hardware simulation of a bottling plant. The plant complexity requires the use of structured approaches. The HISC approach works by decomposition of a complex system into a high level subsystem, communicating with several low level subsystems. Subsequently the implementation on programmable logic controllers (PLC) at the low level and on a master control station at the high level as well as the implementation of an order manager and business-oriented evaluation results are shown. The TCP/IP protocol is used for communication between low and high level via industrial Ethernet, thus the implementation enables remote control over the Internet. The bottling plant

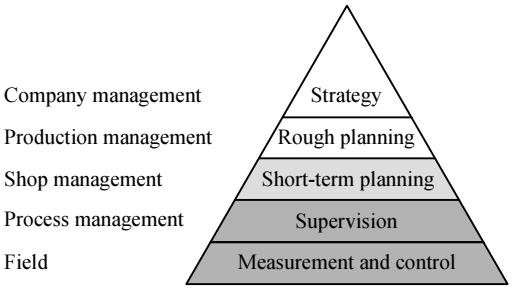


Figure 1. Organizational hierarchy of a company

can be considered as a large-scale system, consisting of four work cells connected by a distributed transport system. In the hierarchical multi-layer model depicted in Figure 1, this work mainly addresses the field and process management layers. The also implemented order manager and business evaluation environment belong to the shop management layer above.

The rest of this paper is organized as follows. Section 2 provides the basis for this work. The bottling plant and the supervisory control design are presented in Section 3. Section 4 details the implementation of the designed supervisory control, subsequently followed by some evaluation results in Section 5. Finally, the paper concludes with Section 6.

2. PRELIMINARIES

2.1 SCT Basics and Structured Approaches

In this work, DES are represented by deterministic finite automata (dfa). The theory of languages is used to express their underlying behavior.

Consider σ as an event and Σ as a finite alphabet of events. Let the Kleene-closure Σ^* of Σ denote the set of all finite sequences of events s including the empty sequence ε. A formal language over Σ is any subset $L \subseteq \Sigma^*$. A plant is modelled as a generator $G = (X, \Sigma, \delta, x_0, X_\mathrm{m})$ with state set X, Σ as above, partial transition function $\delta : X \times \Sigma \rightarrow X$, initial state x_0 and $X_\mathrm{m} \subseteq X$ the set of marker states. G generates the regular language $L(G)$ expressing the uncontrolled behavior of G. Σ is partitioned into the disjoint partition of controllable and uncontrollable events $\Sigma = \Sigma_\mathrm{c} \cup \Sigma_\mathrm{uc}$ and into the disjoint partition of observable and unobservable events $\Sigma = \Sigma_\mathrm{o} \cup \Sigma_\mathrm{uo}$. Generally, the two partitions are independent from each other. However, in most technical applications all controllable events are also observable. The desired controlled behavior of a system is reflected by a specification language $K \subseteq L(G)$. For a detailed introduction to SCT, refer to (Wonham, 2002).

A proper supervisory control to achieve the desired behavior can be synthesized automatically using SCT. In a closed loop, the event sequence generated by G is controlled in the sense that undesired future events are disabled by the supervisory control. This is subject to restrictions due to uncontrollable events, which cannot be disabled.

The standard synthesis procedure in SCT results in a monolithic supervisory control, derived from a global plant model (Wonham, 2002). This result is inappropriate for two reasons. First, it is impossible to set up a global plant model for systems of industrial size neither ad hoc nor by composition. Second, global sensing and acting of a monolithic supervisory control is not desired or impossible in many systems. Structured approaches can be used to weaken the aforementioned difficulties. The modular approach was introduced by Ramadge and Wonham (1988) to reduce the complexity of resulting supervisory controls and refined to avoid the explicit computation of the global plant model (de Queiroz and Cury, 2000*a*; de Queiroz and Cury, 2000*b*). However, all observable events must be available to all supervisory controls. The decentralized approach (Rudie and Wonham, 1992) takes local availability of sensors and distributed control hardware into consideration. A decentralized supervisory control acts locally, based on local sensor information. A global plant model is still necessary for synthesis. Finally, the HISC approach as the structured approach applied in this work is summarized in Subsection 2.2.

2.2 Hierarchical Interface-based Supervisory Control

The HISC approach and its characteristics are now presented briefly. A detailed description can be found in (Leduc *et al.*, 2001*a*; Leduc *et al.*, 2001*b*). An application example with similar complexity as the bottling plant is described by Leduc and Lawford (2004).

Contrasting all previously mentioned approaches, the HISC approach does not yet provide a synthesis method for supervisory controls. It rather defines test and verification algorithms for controllability and nonblocking of intuitively designed specifications, based on given plant models. An extension to a synthesis approach is being considered (Leduc and Lawford, 2004).

The basic idea of this approach is the combination of bi-level hierarchy and low level modularity for plant modelling and supervisory control design. Communication between the two levels happens through interface automata, which couple the low level components with the high level. This architecture is shown in Figure 2. Complete systems consist of n low level components ($n > 1$), n interfaces and one high level component, called n^{th} degree parallel interface systems (PIS). The trivial case ($n = 1$), the so-called serial interface system (SIS), is the basic building block for the nonblocking and controllability analysis. An n^{th} degree PIS can be decomposed into n SIS.

The event alphabet of the global system Σ is partitioned into four disjoint sub-alphabets, namely Σ_H

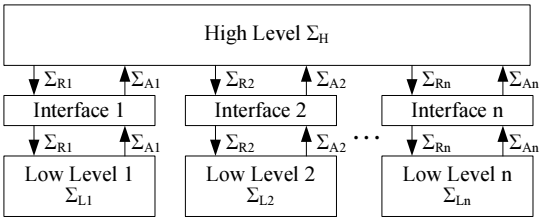


Figure 2. HISC architecture

high level, Σ_{L} low level, Σ_{R} request and Σ_{A} answer events. The latter three are sums of the respective disjoint low level event sets. $\Sigma := \dot{\cup}_{j\in\{1,\ldots,n\}}[\Sigma_{\mathrm{L_j}} \dot{\cup} \Sigma_{\mathrm{R_j}} \dot{\cup} \Sigma_{\mathrm{A_j}}] \dot{\cup} \Sigma_{\mathrm{H}}$. Events in Σ_{H}, Σ_{L} appear only in the high level and low level, respectively. Events related to the interfaces Σ_{R}, Σ_{A} appear in both levels and symbolize the communication within the hierarchy (cf. Figure 2). Every low level component i consists of a local plant model $\mathcal{G}_{\mathrm{Li}}$, a local supervisor $\mathcal{S}_{\mathrm{Li}}$, derived from local specifications, and an interface G_{Ii}, to specify the communication between high level and the low level component. The high level consists of a global plant model $\mathcal{G}_{\mathrm{H}}$, a global supervisor $\mathcal{S}_{\mathrm{H}}$, derived from global specifications, and all interfaces of the low level components $G_{\mathrm{I1}},\ldots, G_{\mathrm{In}}$. The high level sends a request to a low level component, which then performs the triggered task and sends back an answer event after task completion. Figure 3 shows this structure conceptually for a 2nd degree PIS.

Since the focus of this work is on control design and implementation rather than on verification, the details of the HISC approach are not provided here. A verbose description can be found in (Leduc *et al.*, 2001*a*) and (Leduc *et al.*, 2001*b*).

2.3 Business-oriented Performance Evaluation

From a business-oriented perspective, a production system is evaluated by business parameters, for instance lead time and utilization. These parameters can be visualized by using the throughput diagram,

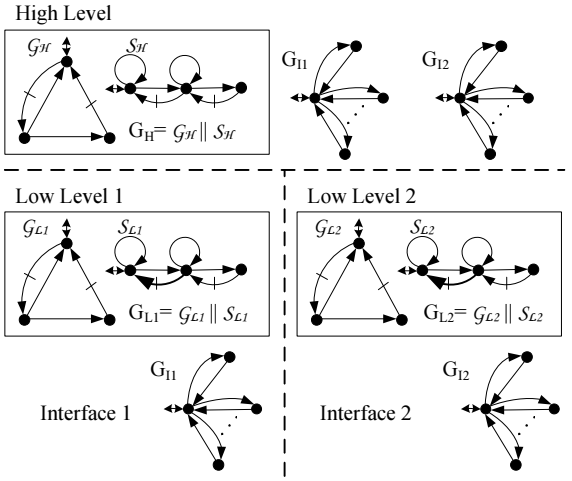


Figure 3. HISC structure for a 2nd degree PIS

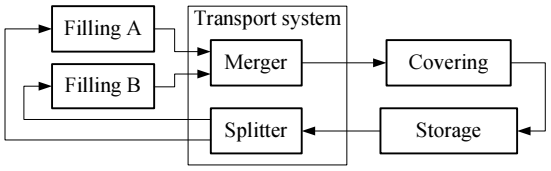


Figure 4. Plant structure and product flow

which shows the input, inventory and output trend of a production system. Relevant business parameters are derivable from this diagram. The diagram is used for analysis, monitoring and diagnosis of manufacturing flow. Business-oriented control rules to improve the manufacturing flow, e.g. release control and lot splitting, can be deduced from throughput diagram data (Wiendahl, 1995). A long term recording of business parameters is further evaluated statistically.

3. PLANT AND CONTROL DESIGN

3.1 Plant and Control Hardware

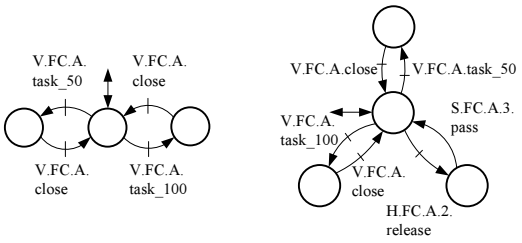
The bottling plant and the control hardware are well-suited for treatment using the HISC approach due to their given structure. The plant consists of four work cells, namely filling cells A and B (FCA, FCB), covering cell (CC) and storage cell (SC), mutually connected by a transport system (TS), divided into a merger and splitter part as shown in Figure 4.

Six different kinds of bottles can be filled and covered, differing in level and mixing ratio. Empty bottles start at the storage cell (from a raw material buffer) and enter their respective filling cell. After filling, the bottles move on to the covering cell to be covered by a press and to be subsequently stored in their final product buffer in the storage cell. Mix-product bottles bypass the covering and storage cells after the first filling cycle, re-enter the filling cells via the feedback loop and are then treated as before. Only one bottle can be served simultaneously at each work cell, but all work cells are equipped with limited capacity input buffers to store incoming bottles, thus inventory$>$1 is possible.

To every work cell and the transport system, an 80186-20MHz microprocessor based PLC by Beck© is attached. The interfaces of the controllers consist of digital I/O ports as well as two serial connectors and one IEEE802.3 10MBit/s Ethernet connector to provide Web, FTP and Telnet services over TCP/IP or UDP. The installed preemptive multi-tasking and real-time RTOS operating system for IPC@Chip is used. PLC programs are created using the C programming language.

3.2 Plant Models and Specifications

The following assumptions have been made for modelling. All controllable events are also observable. De-



(a) Valve FCA (b) Interface FCA

Figure 5. Automata diagrams

rived from the physical plant structure, the models are built in a component-based manner, thus most parts of the plant can be considered as a product system (components work fully in parallel). Principally, controllable events can be triggered at any time, thus at any system state. In case of no effect to the related actuator, this leads to selfloops in the automata model. If the related actuator is already in the desired position, the resulting selfloop is omitted in the automaton diagram.

Since the HISC approach provides no synthesis method, the actual design was done using insight and experience. To this end, some specifications were designed to represent formally a verbal description of a certain complete task, such as filling a bottle (functionality). As a result, they are possibly far from minimal. Others forbid undesirable machine states (safety).

Low level plant models and specifications. Each included work cell and the transport system is reflected by one logical low level component. The low level components are represented by generators $G_{\mathrm{L}}^{\mathrm{i}}$, each consisting of a plant component $\mathcal{G}_{\mathrm{L}}^{\mathrm{i}}$ and a supervisor component $\mathcal{S}_{\mathrm{L}}^{\mathrm{i}}$ derived from the related low level specifications as defined in Subsection 2.2, $i \in \{FCA, FCB, CC, SC, TS\}$. Only a small fraction of the model can be presented here due to its size.

Consider the model of the valve in FCA, a part of $\mathcal{G}_{\mathrm{L}}^{\mathrm{FCA}}$, as depicted in Figure 5(a). The valve can fill the bottle completely (event $V.FC.A.task\_100$) or halfway (event $V.FC.A.task\_50$). To fill a bottle is a self-contained task, which can be completed by FCA alone without interaction with other cells, thus the low level component is exclusively responsible for execution and supervision in this case.

A filling cycle is requested by the high level through the FCA interface. After the filling is complete, FCA sends an answer to the high level. The communication between low and high level is modelled as an interface automaton, depicted in Figure 5(b). In this example, the interface automaton is defined over the alphabet $\Sigma_{\mathrm{R}} \cup \Sigma_{\mathrm{A}} = \{V.FC.A.task\_50, V.FC.A.task\_100, H.FC.A.2.release\} \cup \{V.FC.A.close, S.FC.A.3.pass\}$.

The structure of the interface indicates another possibility, which is not part of the valve model. $H.FC.A.2.release$ requests the filling cell to transfer the bottle without any filling, acknowledged to the high level by $S.FC.A.3.pass$. The filling task is desired only if a bottle is placed under the valve before, thus a low level specification must restrict the execution of the task to respect this condition. Low level specifications concern the corresponding low level component only. Basically, low level specifications prohibit undesired states or coordinate fixed, internal tasks. From the component-oriented point of view, no communication with the outside world is necessary to fulfil an internal low level task.

High level plant models and specifications. The high level is represented by a generator G_{H}. It consists of the high level plant model given by the synchronous product $\mathcal{G}_{\mathrm{H}} = \mathcal{G}_{\mathrm{H}}^{\mathrm{FCA}} || \mathcal{G}_{\mathrm{H}}^{\mathrm{FCB}} || \mathcal{G}_{\mathrm{H}}^{\mathrm{CC}} || \mathcal{G}_{\mathrm{H}}^{\mathrm{SC}} || \mathcal{G}_{\mathrm{H}}^{\mathrm{TS}}$ and the high level supervisor $\mathcal{S}_{\mathrm{H}}$ derived from high level specifications. High level plant models are more abstract and not restricted to one component. Contrariwise, they are used to express the coupling between several low level components. High level specifications are used to coordinate the interaction between the low level components. As an example, consider the covering cell input buffer. Restricting the number of bottles in its input buffer is only possible by coordinating and supervising the output of the two feeding cells (FCA,FCB) and the output of the covering cell itself (c.f. Figure 4). It is impossible for any single low level component to fulfil this specification alone. Instead, coordination between several low level components is necessary. Hence, as an example, all buffer management specifications are located at the high level.

The complexity issue. The bottling plant can be classified as a large-scale system. 22 low level plant model automata, 12 low level specification automata and 5 interface automata reflect the complete low level of the bottling plant. 10 high level plant model automata and 13 high level specification automata are defined to create $\mathcal{G}_{\mathrm{H}}$ and $\mathcal{S}_{\mathrm{H}}$, respectively. Under the worst case assumption, no states are deleted by the cartesian product under composition. Figure 6 shows a complexity estimate for the transport system and the global bottling plant model based on this assumption. From the figures it is clear that the plant is too complex for monolithic control design.

Name	No. of Aut.	No. of States
serial subsystem TS		
$G_{\mathrm{H}}^{\mathrm{TS}} = \mathcal{G}_{\mathrm{H}}^{\mathrm{TS}} \| \mathcal{S}_{\mathrm{H}}^{\mathrm{TS}}$	20	$\approx 5.5 \times 10^{15}$
$G_{\mathrm{L}}^{\mathrm{TS}} = \mathcal{G}_{\mathrm{L}}^{\mathrm{TS}} \| \mathcal{S}_{\mathrm{L}}^{\mathrm{TS}}$	5	$\approx 2.2 \times 10^{4}$
$G_{\mathrm{I}}^{\mathrm{TS}}$	1	5
flat serial subsystem TS		
$G^{\mathrm{TS}} = G_{\mathrm{H}}^{\mathrm{TS}} \| G_{\mathrm{L}}^{\mathrm{TS}} \| G_{\mathrm{I}}^{\mathrm{TS}}$	26	$\approx 6.1 \times 10^{20}$
flat bottling plant		
$G = G_{\mathrm{H}} \| G_{\mathrm{L}} \| G_{\mathrm{I}}$	62	$\approx 9 \times 10^{47}$

Figure 6. Complexity estimate for the bottling plant

4. IMPLEMENTATION

4.1 Architecture

The proposed implementation architecture, depicted in Figure 7, is based on the given hardware structure of the plant and supports the HISC approach. Logical interfaces and components are adopted as physical interfaces and components, thus all interface events are transmitted by physical Ethernet connections. Each logical low level component is equipped with a PLC and the high level is implemented on a control station personal computer, equipped with respectively distinct IP addresses. Only vertical communication is allowed. The hardware interface inside the PLC links the physical and logical layers, hence separates between timed and un-timed levels of abstraction. Timing is therefore present at the HW-interface software level, e. g. for the bottle filling events mentioned in section 3.2. The low level supervisor module acts as a filter to pass requests from the high level to the hardware interface, if admissible. Furthermore this module is responsible for compliance with local specifications and has to report completed low level tasks to the high level. The interface module implements the bidirectional communication between low and high level. The high level supervisor module is responsible for compliance with high level specifications and filters commands from the order manager module, which accepts orders defined by a user and coordinates order tracing. It also issues commands to the low level.

4.2 Programming

The implementation of the HISC supervisory control requires software on both levels due to the distributed nature of the hardware implementation. All developed programs are independent from the number and structure of both interfaces and specifications. In this sense, the software is generic and supports fast supervisor update and maintenance.

Low level implementation. A program sequence of two concurrent tasks is running on every PLC. Figure 8 details the two tasks. A connection server task receives the incoming requests from the high level and records every request into a variable of the main task. The interfaces and specifications are updated under consideration of the received request events within the main task and answer events are sent to the high level, if necessary. The next step is the selection of possible subsequent events, to be executed within the following program step. This sequence is repeated until all requests are settled. The low level implementation uses the concept of polling for sensor reading and is done in ANSI-C.

High level implementation. The implementation of the high level supervisor comprises two tasks. First, a connection server task is needed again to receive answer events from the low level. The update procedure is similar to the low level within the main task. In addition, the main task endues an interface to receive, coordinate and sort commands from the order manager. Those commands are deleted by the main task after execution. Order manager and high level supervisor interact by means of a shared data structure, a list. Access to this list is mutually exclusive for consistency. The order manager writes events to the list, and the high level supervisor reads events from the list and deletes them after execution. The order management module includes an interface to the user, while the high level supervisor runs in the background without user interaction. An order is defined as a sequence of special events interpreted as command requests, acceptable by the high level supervisor. Execution of the listed events in the correct order result in a completed product. A FIFO queue is implemented for every work cell of the bottling plant to store orders. Released orders are first stored in the queue, related to the first work cell in the process (the storage cell), to wait for their execution. The order moves to the queue related to the next station after execution of all requests related to the present work cell. The high level programs and their graphical user interfaces are implemented in an object oriented manner using the SmallTalk programming language.

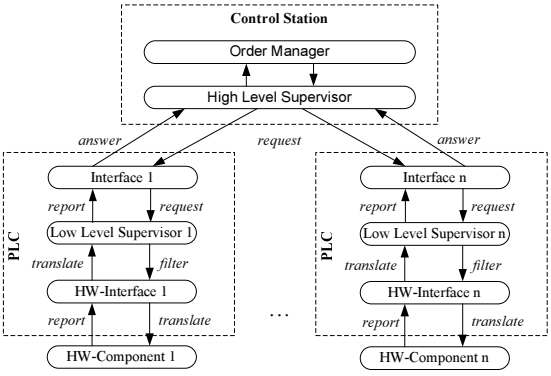


Figure 7. HISC implementation architecture

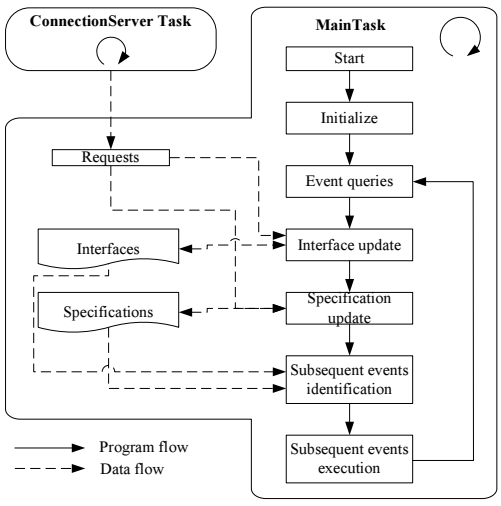


Figure 8. Flow chart of the low level implementation

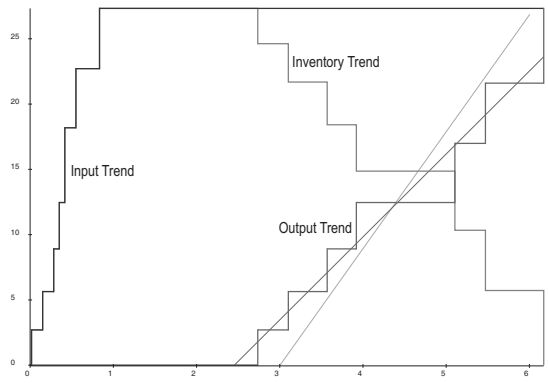


Figure 9. Entire bottling plant throughput diagram

5. PERFORMANCE EVALUATION

The order manager records time stamps and order work content whenever an order moves from one work cell queue to another. This information is used to construct throughput diagrams. Figure 9 depicts such a diagram for a specific order batch and the entire plant. Benchmark testing leads to the following results and statements, where a prior industrial control implementation represents the benchmark. The mean inventories of the work cells are lower than when using the industrial control. This result is a direct consequence of the use of the HISC approach itself. A state transition only happens if the related event occurs, e.g. when the next sensor is activated. Timed state transitions are not admissible in the HISC approach, but exist in the industrial benchmark control. This leads to ineffective idle situations and forces lead times extension. An unfavorable definition of the logical components adds to the problem. Presently, the transport system is defined as one single logical component. Due to the high coupling of TS with all other work cells and the fact that any low level component may perform at most one interfaced task at any given time, idle situations are expanded. The logical decomposition of TS into its two physical components merger and splitter as shown in Figure 4 decreases the coupling degree and thus the idle time. The decomposition is possible because of the concurrent hardware structure of merger and splitter.

6. CONCLUSION AND FUTURE WORK

A supervisory control was designed for a complex bottling plant along the lines of the HISC approach. It was implemented in a distributed hardware environment reflecting the logical work cell decomposition of the HISC approach. A generic controller was implemented on the PLCs and the control station PC, supporting quick and easy control maintenance. Bottle timing data was recorded and used to generate work cell throughput diagrams for business parameter estimation and subsequent performance evaluation. Future work should include a decomposition of the transport system to reduce the plant idle time. In addition, the polling concept for sensor reading was observed to operate on its timing limits due to the speed of bottles on their conveyor belt. A transition to the concept of interrupt-based sensor reading may help avoiding related problems, but may introduce problems caused by nondeterminism emerging in the program flow.

REFERENCES

Balemi, S., G.J. Hoffmann, P. Gyugyi, H. Wong-Toi and G.F. Franklin (1993). Supervisory Control of a Rapid Thermal Multiprocessor. *IEEE Trans. Automatic and Control* **38**(7), 1040–1059.

Brandin, B.A. (1996). The Real-time Supervisory Control of an Experimental Manufacturing Cell. *IEEE Trans. Robotics and Automation* **12**(1), 1–14.

de Queiroz, M. H. and J. E. R. Cury (2000*a*). Modular Control of Composed Systems. In: *Proceedings of the American Control Conference*. Chicago, USA.

de Queiroz, M. H. and J. E. R. Cury (2000*b*). Modular Supervisory Control of Large Scale Discrete Event Systems. In: *Discrete Event Systems: Analysis and Control* (R. Boel and G. Stremersch, Eds.). pp. 103–110. Kluwer Academic Publishers.

de Queiroz, M.H. and J.E.R. Cury (2002). Synthesis and Implementation of Local Modular Supervisory Control for a Manufacturing Cell. In: *Proc. of 6th Workshop on Discrete Event Systems*. Zaragoza, Spain.

Leduc, R.J. and M. Lawford (2004). Hierarchical Interface-based Supervisory Control of a Flexible Manufacturing System. *Submitted to IEEE Trans. Contr. Sys. Techn.*

Leduc, R.L., B.A. Brandin, M. Lawford and W.M. Wonham (2001*a*). Hierarchical Interface-based Supervisory Control: Serial Case. In: *Proc. of 40th Conf. Decision Contr.*. Orlando, USA.

Leduc, R.L., W.M. Wonham and M. Lawford (2001*b*). Hierarchical Interface-based Supervisory Control: Parallel Case. In: *Proc. of 39th Allerton Conf. Comm. Contr. Comp.*

Ramadge, P. J. and W. M. Wonham (1988). Modular Supervisory Control of Discrete-Event Systems. *Mathematics of Control, Signals and Systems* **1**(1), 13–30.

Ramadge, P.J. and W.M. Wonham (1989). The Control of Discrete Event Systems. *Proc. IEEE, Special Issue on Discrete Event Dynamic Systems* **77**(1), 81–98.

Rudie, K. and W.M. Wonham (1992). Think Globally, Act Locally: Decentralized Supervisory Control. *IEEE Trans. Automat. Contr.* **37**(11), 1692–1708.

Wiendahl, H.-P. (1995). *Load-Oriented Manufacturing Control*. Springer-Verlag. Berlin, Germany.

Wonham, W. M. (2002). *Notes on Control of Discrete-Event Systems*. Dept. of Elec. and Comp. Eng., University of Toronto, Toronto, Canada.