Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Decentralized Supervisory Control of a Distributed Filling Shop Using Standardized Models

Florian Wenck†

Abstract— **This paper deals with the application of the decentralized supervisory control approach to a complex, event-driven filling shop. Due to complexity reduction and for simplicity, the modeling is done in terms of subsystem composition and by using standardized models. The latter one increases the reusability of subsystem models and reduces the gap between current modeling methodologies and the object-oriented paradigm. Standardized models for plant description representing either repetitive hardware components or their linkage are strictly separated, thus their instances can be used as basic building blocks to create the global plant model. In analogy to [1], the achieved decentralized supervisory control is implemented using programmable logic controllers and a standard PC as well as TCP/IP for communication and Internet-based remote control. The desired controlled plant behavior is verified by Gantt charts recorded during operation.**

I. INTRODUCTION

Discrete-event system (DES) representations are a common way to model a complex manufacturing system. Normally, its behavior does not satisfy some kind of requirement, thus a controller needs to be attached to the system restricting its behavior in a specific range, given in terms of specifications. A system is complex in this context, if composition of subsystem models is necessary to derive a global system model and if specifications or locations of the system hardware in far-off sectors require some distributed control structure.

The Supervisory Control Theory (SCT) offers a powerful framework to deal with such problems. The initial SCT publication by Ramadge and Wonham in 1987 assumes full observation of the entire event set of the plant. Under such assumption, the property of controllability acts exclusively as a necessary and sufficient condition for the existence of a monolithic supervisory control that achieves a desired behavior of a DES [2]. To extend the SCT to partial observation problems, the natural projection is introduced as an observation mask to reflect unobservability of some events in the plant. Supervisory control under partial observation requires the additional property of observability to verify monolithic supervisory control existence [3]. The decentralized supervisory control approach by natural projection is used, if several supervisors have to control a common plant jointly. Therefore, the property of co-observability is presented for the case where the desired plant behavior is given as a prefix-closed language [4], and for the nonprefix-closed case [5]. The latter one postulates necessary and sufficient conditions for the existence of a decentralized solution. In [6] the normality property is defined, equivalent to observability under the assumption that all controllable events are as well observable. This is mostly the case in real applications. The restrictiveness of normality is weaken in [7]. Several extensions of the approaches can be found e.g. in [8] and [9].

Regarding modeling, design and implementation, the use of standardized models reduces the effort for modeling as well as simplifies object-oriented treatment. In the standard SCT, no formal method is given for decomposition of the global system into included subsystems. However, repetitive hardware components at different locations and their linkage can be found in most real applications. The objective is to find a proper decomposition of the global system down to a layer were such components can be found. The components should be modelled by standardized models independent from both their absolute location and their linkage, to enhance reusability. Standardized models can be implemented as classes in an object-oriented way. A model for a specific hardware component is then an instance of the related class. This paper shows both that the considered complex filling shop can be fully described by instances of only a few standardized models and a decentralized supervisory control achieved from those instances is designed, implemented and successfully tested.

The SCT and the decentralized supervisory control approach by natural projection are summarized in Sec. II, followed by a detailed description of the distributed filling shop in Sec. III. Subsequently, the modeling, the controller design and its implementation are presented in Sec. IV, V and VI, respectively. Finally, the paper concludes with Sec. VII.

II. NOTATION AND PRELIMINARIES

A. SCT Basics

In this paper the RW-notation is used to describe DES [10]. The structure of DES is represented by deterministic finite automata (dfa), interpreted in terms of generators. The theory of languages is used to express the behavior of DES generated by the corresponding dfa.

Consider a dfa $G = (X, \Sigma, \delta, x_0, X_\mathrm{m})$ were X denotes the finite set of states, Σ the set of all events σ in G, $\delta : X \times \Sigma \rightarrow X$ the partial transition function, x_0 the initial state and $X_\mathrm{m} \subseteq X$ the subset of marker states. Marking is a modeling issue to emphasize the completion of some operations. The Kleene-closure Σ^* of Σ denotes the set of all finite strings s, derived from concatenation of arbitrary events $\sigma \in \Sigma$, including the empty string ε. A formal language over Σ is any subset

† Florian Wenck is with the Department of Process Automation, Hamburg University of Technology, D-21071 Hamburg, Germany wenck@tu.harburg.de

0-7803-9568-9/05/$20.00 ©2005 IEEE

$L \subseteq \Sigma^*$. The prefix closure of L, indicated by $\overline{L}$, consists of all prefixes of strings in L. In general $L \subseteq \overline{L}$. L is prefix-closed, if $L = \overline{L}$. The uncontrolled behavior of G is represented by the language $L(G) = \overline{L(G)}$ containing all strings of events that G can generate. Its marked behavior denoted by $L_m(G)$ includes only those strings whose generation lead to marker states. Σ is subdivided into the disjoint partitions of *controllable* and *uncontrollable* $\Sigma = \Sigma_c \dot\cup \Sigma_{uc}$ and *observable* and *unobservable* events $\Sigma = \Sigma_o \dot\cup \Sigma_{uo}$. In most technical applications $\Sigma_c \subseteq \Sigma_o$, in general the partitions are independent from each other. A specification language $K \subseteq L(G)$ reflects the desired controlled behavior of G. For further details about SCT, refer to [10]. A proper supervisory control to achieve K can be synthesized automatically using SCT. In a closed loop, s generated by G so far is supervised and undesired subsequent events are disabled by the control actions of the supervisory control. This is subject to restrictions due to events $\sigma \in \Sigma_{uc}$, which cannot be disabled.

The standard synthesis procedure in SCT results in a monolithic supervisory control, derived from a global plant model [10]. This result is inappropriate for two reasons. First, it is impossible to set up a centralized supervisory control for industrial systems of a distributed structure. 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, e.g modular approach [11] or decentralized approach using natural projection [5]. Both approaches postulate the distribution of the control task onto several supervisors acting jointly on a single, global plant. The latter approach additionally takes local sensing and acting into consideration and is used within this work. This approach is summarized in Subsec. II-B.

B. Decentralized Supervisory Control by Natural Projection

In decentralized supervisory control by natural projection the global plant G is controlled by several supervisors, whereas each supervisor can observe and control only a subset of Σ. It represents a distributed control system where supervisors at different sites see the effect of different sets of sensors and control different sets of actuators. This is theoretically represented by a distinct natural projection for each supervisor. In addition to the basic SCT problems *with or without tolerance*, decentralized supervisory control problems can be classified into *local* and *global* control problems [5]. For the latter one, specifications are given by overall tasks for the entire system, which cannot be decomposed into several meaningful subtasks, in contrast to local problems. A local problem is solved, if all supervisors can fulfill their local specifications isolated by controlling their controllable events. The control task for the filling shop can be defined as a local problem. The approach is resumed in a nutshell for two supervisors.

Let $\Sigma_0 \subseteq \Sigma$ denote a subset of observable events. The *natural projection P* deletes unobservable events from strings $s \in \Sigma^*$ and obtains the order of the remaining events. $P : \Sigma^* \to \Sigma_o^*$ is defined by:

$$P(\varepsilon) = \varepsilon \qquad P(\sigma) = \varepsilon, \sigma \in \Sigma \setminus \Sigma_o$$
$$P(\sigma) = \sigma, \sigma \in \Sigma_o \quad P(s\sigma) = P(s)P(\sigma), s \in \Sigma^*, \sigma \in \Sigma$$

The corresponding *inverse natural projection* $P^{-1} : \Sigma_o^* \to 2^{\Sigma^*}$ of a given string $t \in \Sigma_o^*$, produces the set of all strings $T = P^{-1}(t)$ in Σ^* with the property $P(T) = t$. Note that P^{-1} is not an inverse function in the sense that $P^{-1}(P(s)) = s$ for $s \in \Sigma^*$. P and P^{-1} are extended to languages as usual. A decentralized supervisory control for a *local control problem with tolerance* is defined formally as follows.

Given DES G with event set Σ. Let S_1, S_2 be supervisors that can observe $\Sigma_{1,o}$, $\Sigma_{2,o}$ and can control $\Sigma_{1,c}$, $\Sigma_{2,c}$, respectively. $\Sigma_{1,o}$, $\Sigma_{1,c} \subseteq \Sigma_1$ and $\Sigma_{2,o}$, $\Sigma_{2,c} \subseteq \Sigma_2$ and Σ_1, $\Sigma_2 \subseteq \Sigma$. $L_{r,1}$, $L_{r,2}$ are the minimal required languages and $L_{a,1}$, $L_{a,2}$ the maximal admissible languages for S_1, S_2. Note that $L_{r,1}$, $L_{a,1} \subseteq \Sigma_1^*$ and $L_{r,2}$, $L_{a,2} \subseteq \Sigma_2^*$. P_1, P_2 are the natural projections $P_i : \Sigma^* \to \Sigma_i^*$, $i \in \{1,2\}$. Find S_1, S_2 that satisfy

$$L_{r,1} \subseteq L(S_1/G) \subseteq L_{a,1} \tag{1}$$
$$L_{r,2} \subseteq L(S_2/G) \subseteq L_{a,2} \tag{2}$$

This leads to the solution of the following problem

$$L(G) \cap \bigcap_{i=1}^{2} P_i^{-1}(L_{r,i}) \subseteq L(\tilde{S}_1 \wedge \tilde{S}_2/G) \subseteq L(G) \cap \bigcap_{i=1}^{2} P_i^{-1}(L_{a,i}) \tag{3}$$
$$L(\tilde{S}_1 \wedge \tilde{S}_2/G) = L(\tilde{S}_1/G) \cap L(\tilde{S}_2/G) \tag{4}$$

with $\tilde{S}_i$ the *global extension* of S_i. For controllable events, $\tilde{S}_i$ performs the same control action as S_i on $\Sigma_{i,c}$ and enables all events in $\Sigma \setminus \Sigma_{i,c}$. Additionally for observable events, $\tilde{S}_i$ acts similar as S_i for events in $\Sigma_{i,o}$ and remains at the same state for events in $\Sigma \setminus \Sigma_{i,o}$ (selfloop transitions). To calculate a global extension, the inverse natural projection must be applied on the language of the respective supervisor. In order to solve the problem stated in (3), each supervisor has to solve its own range problem by checking the following conditions that verify the existence of the supervisors.

$$L_{r,i}^{\downarrow \mathscr{O}} \subseteq L_{a,i}^{\uparrow \mathscr{C}} \quad , \quad i \in \{1,2\} \tag{5}$$

with $L_{r,i}^{\downarrow \mathscr{O}}$ the infimal prefix-closed observable superlanguage of $L_{r,i}$ and $L_{a,i}^{\uparrow \mathscr{C}}$ the supremal controllable sublanguage of $L_{a,i}$. However, the solution in (5) does not guarantee that S_1, S_2 can work together correctly. In order to prevent conflicts, the following nonconflict condition must be verified by selecting the specifications K_1 and K_2 each from their own range $L_{r,i}^{\downarrow \mathscr{O}} \subseteq L_{a,i}^{\uparrow \mathscr{C}}$, $i \in \{1,2\}$.

$$\overline{P_1^{-1}(K_1)} \cap \overline{P_2^{-1}(K_2)} = \overline{P_1^{-1}(K_1) \cap P_2^{-1}(K_2)} \tag{6}$$

III. THE FILLING SHOP

The filling shop considered here as the global plant is a subarea of the bottling plant presented in [1]. Both the plant and the control hardware are well suited for treatment using the decentralized approach due to their given distributed structure. A plant schematic is illustrated in Fig. 1.

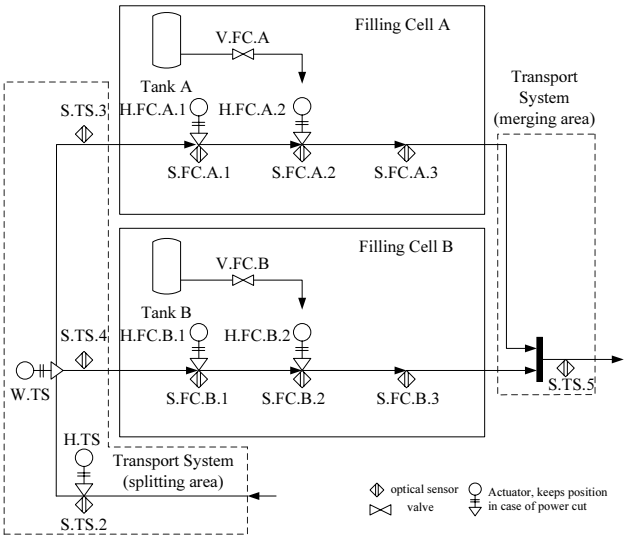


Fig. 1. Plant schematic of the filling shop

A. Plant and Product Description

The filling shop consists of two distributed filling cells (FC), namely filling cell A (FCA) and filling cell B (FCB), connected by a conveyor-based transport system (TS), divided into merger and splitter area as shown in Fig. 1. Included sensors and actuators are labeled in the following manner. *HardwareComponentType.Location[.Number]*, where the first element is one of $\{$Sensor (S), Router (W), Stopper (H), Valve (V)$\}$. Directions of included arcs represent the possible product and raw material flow. It can be noticed by inspection of Fig. 1 that FCA and FCB are identical in construction.

Four different kinds of bottles can be filled, differing in level and sort of liquid. Empty bottles enter the splitting area of the TS from an infinite input source, and are directed by the router W.TS to their designated FC A or B. Each FC fills an arriving bottle either completely (products $A100$, $B100$) or half-full ($A50$, $B50$). After filling process, the bottles are routed to an infinite output sink via the merging area of the TS.

B. Control Hardware and Control Objectives

Each FC and the TS is equipped with a 20MHz microprocessor based PLC by Beck©, considering signals from sensors and to actuators related to FCA, FCB and TS respectively. Digital I/O ports as well as two serial and one IEEE802.3 10MBit/s Ethernet gateway for Web, FTP and Telnet sevices over either TCP/IP or UDP protocol exist for each PLC. The installed RTOS operating system for IPC@Chip is used, offering preemptive multi-tasking and real-time functionality. Furthermore a standard PC is attached, considering signals from all sensors and to all actuators in the filling shop by making use of the implemented bidirectional signal forwarding ability of the PLC over the Ethernet gateway.

Generally, the entire control law is designed as customary under three aspects. The plant should fulfil predefined desired tasks and should neither reach undesired states in the sense of safety nor in the sense of blocking. To be specific, the filling shop is controlled to produce a fixed perseverative sequence $\gamma = B100A100B50A50$ of filled bottles under the above mentioned constraints. In terms of regular expressions, the output of the filling shop at any time is described by $(\alpha)+(\gamma)^*(\alpha)$ with $\alpha = (\varepsilon + B100 + B100A100 + B100A100B50)$. Contrariwise to the work done in [1], the order management is limited to this specific sequence and thus included in the entire control law.

IV. STANDARDIZED MODELING

This section details the modeling procedure for the filling shop. Standardized modeling is described both conceptually and problem related. The tag name notation for event labeling used in this work is defined as follows. *HardwareComponentType.Location[.Number].Key*, whereas *Key* describes a particular action or property.

A. Concept of Standardized Models

It is comprehensible, that the complexity of the filling shop renders an ad-hoc design of the global plant model G impossible. Composition of subsystems is required, whereas no course of action is given for subsystem decomposition.

Most systems in manufacturing consist of sets of units or components, whose elements are repeatedly located at different sites of the plant. One purpose of standardized modeling is to improve model reusability by choosing a level of decomposition such that the obtained subsystems reflect this repetitive hardware components. Subsequently, those hardware components are modelled independently from their location.

Along the lines of automata representation, those *standardized component models* are labeled by *generic events* with *Location[.Number]* empty instead of specific instances of events. E.g. generic event $V.task\_100$ representing the start of an arbitrary completely filling cycle in the plant is instantiated by $V.FC.A.task\_100$ or $V.FC.B.task\_100$.

From an object-oriented modeling perspective, this situation is reflected in terms of a class diagram in Fig. 2 wherein the notation of the *Unified Modeling Language* [13] is used. Instances of the class `Automaton` consist of arbitrary numbers (`[0..*]`) of states, transitions and generic events.

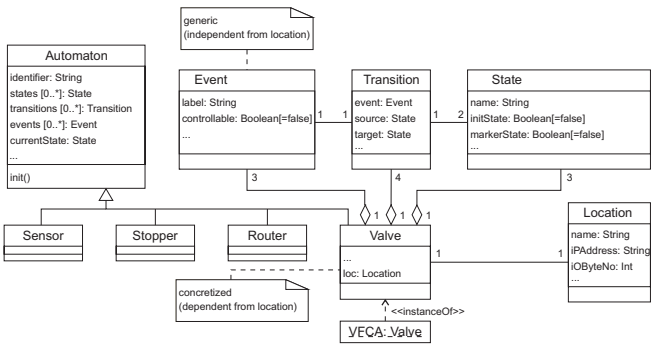


Fig. 2. UML class diagram, representing a part of the structural model

The generic event *V.task_*100 belongs to the range of values for the attribute `label` of class `Event`. The concretization takes place in the subclass `Valve` by assigning a specific location, thus an instance of `Valve`, e.g. `VFCA:Valve`, contains the specific event *V.FC.A.task_*100. For supervisor implementation, instances of `Location` specify all needed hardware information related to considered events, for instance IP address and I/O port number of the control hardware.

Any system $G = (X, \Sigma, \delta, x_0, X_m)$ represented by a composition of instances of standardized component models $G_i = (X_i, \Sigma_i, \delta_i, x_{0i}, X_{mi})$ so far is a purely product system. Hence an aggregation of subsystems working fully in parallel is represented by composition of asynchronous automata [12]. $G = \|_{i \in I} G_i$, $I = \{1, ..., n\}$ with $\Sigma_j \cap \Sigma_k = \emptyset$, $\forall j \neq k$, $j,k \in I$. The resulting event set Σ of G is the union of all disjoint event sets of G_i, $\Sigma = \dot{\cup}_{i \in I} \Sigma_i$.

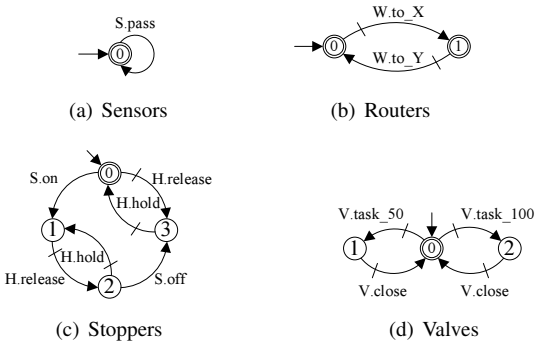
The linkage of the hardware components by the TS is represented by instances of *standardized linkage models* instead of coordinating them by specifications [14] or by using special compositional operators [15]. Instances of both kinds of standardized models are used as basic building blocks to determine the global plant model. To look upon standardized models as classes and their specific realizations as instances of such classes, as shown exemplary in Fig. 2, supports object-oriented modeling, implementation and simulation.

B. Standardized Models of the Filling Shop

The following assumptions are considered for modeling: Conveyors in the plant always run. Inventories of liquid and empty bottles are infinite. Sensor activation only happens due to the presence of a bottle. Selfloop transitions related to events $\sigma \in \Sigma_c$ are omitted.

Standardized component models. The following hardware components are suitable for standardization, verifiable by inspection of Fig. 1: Isolated optical sensors, routers, stoppers with optical sensors and valves. Conveyors are excluded from this consideration, due to the above mentioned assumption. Each instance of each standardized component provides the same functionality.

Isolated optical sensors are responsible for indicating the position of a bottle in the process. They provide only the uncontrollable generic event *S.pass* to acknowledge the passage of a bottle. Their standardized model is depicted in Fig. 3(a). Routers are used to control the bottle flow. From a fixed incoming direction, bottles can be routed to one of two outgoing directions (X,Y). A router is described by two states, representing its current outgoing direction. The switching is reflected by the controllable generic events *W.to_X* and *W.to_Y*, Fig. 3(b). Stoppers with optical sensors stop the flow of bottles in their hold position and release only one bottle at the same time in their release position. In addition, an optical sensor recognizes the presence of a bottle at the stopper. The uncontrolled work cycle is illustrated in Fig. 3(c). Stoppers change their respective position by the controllable generic events *H.hold* and *H.release* with or without an available bottle (*S.on*, *S.off*). Finally, the standardized component model for valves is shown in Fig. 3(d). Valves provide two tasks, namely filling a bottle completely or half-full. These tasks are triggered by the controllable generic events *V.task_*100, *V.task_*50, respectively.



(a) Sensors (b) Routers (c) Stoppers (d) Valves

Fig. 3. Standardized component automata diagrams

Along the lines of object-oriented modeling, the standardized automata diagram for valves is represented by a class `VALVE` with attribute/value pair `events[3]:=(V.task_50,V.task_100,V.close)` for containing events. To obtain the specific model for e.g. the valve in FCA, `Valve` and `Location` are instantiated as shown in Fig. 4. The bracketed quantifier after the attributes can be identified in Fig. 2 as well, by inspecting the valued aggregation relations. It is easily recognizable that simply another location must be assigned for instantiation to obtain the specific model for the valve in FCB.

Standardized linkage models. The linkage of the components in the plant, is mostly defined over sensor activation sequences, derived from the flow of bottles on the fixed TS. The key idea is, that a considered sensor (*S.con*) can only be activated by a bottle if at least one of its predecessor sensors (*S.pre*) has been activated by the same bottle before.

Two different kinds of standardized linkage models are defined. One to represent basic patterns in a TS such as splitting and merging of product flow or simple serial arrangement of sensors (Fig. 5(a) - 5(c)), another to describe the number of bottles present between a sensor and one of its successor sensors (not necessary the direct successor), Fig. 5(d). Combinations of these standardized linkage models cover all possible situations in the plant.

Because of their underlying nature, linkage models can contain events related to different locations. This requires

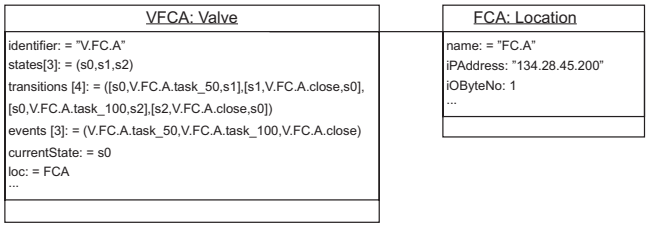
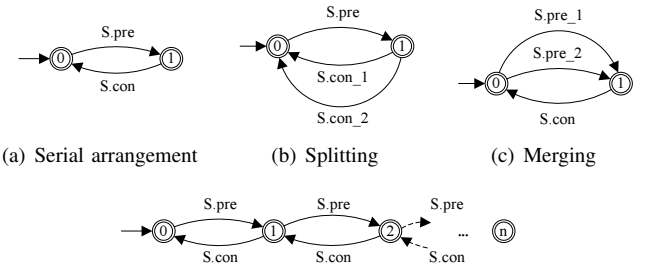


Fig. 4. Instances, representing the model of the valve in FCA



(a) Serial arrangement (b) Splitting (c) Merging

(d) Bottle count

Fig. 5. Standardized linkage automata diagrams

a substitution of the `loc` attribute in subclasses related to linkage models by the two attributes shown in Fig. 6. The bracketed quantifiers reflect the difference between the linkage patterns.

V. DECENTRALIZED CONTROL DESIGN

This section details the design of the decentralized control applied to the plant. Preliminarily, an overview of all needed instances of standardized models to describe the uncontrolled behavior of the plant completely is presented as well as some selected specifications.

A. Instances Creation and Computations

For all included hardware components, concrete models are created by instantiation of the standardized component models described in Sec. IV and successively combined by synchronous product to achieve the product system. Instances of linkage models are treated similar.

All computations are performed by TCT software. Refer to [10] for TCT `NOTATION`, used in this work. The subsequent notation is used for instances. [*Identifier; {EventSet}; (NoOfStates,NoOfTransitions)*]. In order to render the identical construction of FCA and FCB, index i is defined in advance and valid henceforth for compactness.

$$i \in \{A, B\} \tag{7}$$

Instances creation of component models. Five isolated optical sensors are located in the plant. [$STSk$; $\{S.TS.k.pass\}$; (1,1)] $\forall k \in \{3,4,5\}$ and [$SFCi3$; $\{S.FC.i.3.pass\}$; (1,1)] $\forall i$. The router is instantiated by [WTS; $\{W.TS.to\_A$, $W.TS.to\_B\}$; (2,2)] and the five included stopper/sensor combinations by [$HTS/STS2$; $\{H.TS.release$, $H.TS.hold$, $S.TS.2.on$, $S.TS.2.off\}$, (4,6)] and [$HFCik/SFCik$; $\{H.FC.i.k.release$, $H.FC.i.k.hold$, $S.FC.i.k.on$, $S.FC.i.k.off\}$; (4,6)] $\forall i$, $\forall k \in \{1,2\}$. The valves are concretized by [$VFCi$; $\{V.FC.i.task\_50$, $V.FC.i.task\_100$, $V.FC.i.close\}$, (3,4)] $\forall i$.

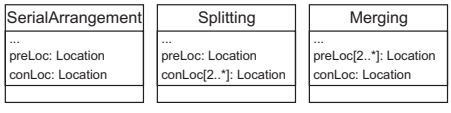


Fig. 6. Subclasses of `Automaton` related to linkage models

Three subsystem models FCA, FCB, TS as an intermediate step and the product system model $PLANT$ of the global plant are derived by synchronous product of instances.

$$\begin{aligned}
FCAH &= \texttt{SYNC}[HFCA1/SFCA1, HFCA2/SFCA2] \\
FCBH &= \texttt{SYNC}[HFCB1/SFCB1, HFCB2/SFCB2] \\
FCiVS &= \texttt{SYNC}[VFCi, SFCi3]\ \forall i \\
FCA &= \texttt{SYNC}[FCAH, FCAVS]\ (48,256) \\
FCB &= \texttt{SYNC}[FCBH, FCBVS]\ (48,256) \\
TSHW &= \texttt{SYNC}[HTS/STS2, WTS] \\
TS &= \texttt{SYNC}[TSHW, STS3, STS4, STS5]\ (8,44) \\
PLANT &= \texttt{SYNC}[FCA, FCB, TS]\ (18432,297984)
\end{aligned}$$

Instances creation of linkage models. In the plant, the product flow is split between TS and FCA, FCB and merged again after filling. [$SPLIT$; $\{S.TS.2.off$, $S.TS.3.pass$, $S.TS.4.pass\}$; (2,3)] and [$MERGE$; $\{S.FC.A.3.pass$, $S.FC.B.3.pass$, $S.TS.5.pass\}$; (2,3)].

Constructionally, the number of bottles in each FC is limited to three. Bottles are counted using $S.TS.3$, $S.TS.4$ respectively as increasing and $S.TS.5$ as common decreasing sensors. [$FCACOUNT$; $\{S.TS.3.pass$, $S.TS.5.pass\}$; (4,6)] and [$FCBCOUNT$; $\{S.TS.4.pass$, $S.TS.5.pass\}$; (4,6)]. Finally, one linkage model embodies that the position of the router mutually excludes the activation of $S.TS.3$ and $S.TS.4$. [$EXCL$; $\{W.TS.to\_B$, $W.TS.to\_A$, $S.TS.3.pass$, $S.TS.4.pass\}$; (2,4)].

Regarding FCA and FCB, two serial arrangements of sensors are related to each FC, [$SFCi1$; $\{S.FC.i.1.off$, $S.FC.i.2.on\}$; (2,2)], [$SFCi2$; $\{S.FC.i.2.off$, $S.FC.i.3.pass\}$; (2,2)] $\forall i$ as well as one internal bottle counting between input buffer and nozzle position [$FCiC$; $\{S.TS.k.pass$, $S.FC.i.1.on$, $S.FC.i.1.off\}$; (5,6)], $i = A(k = 3)$, $i = B(k = 4)$ due to constructional limitations.

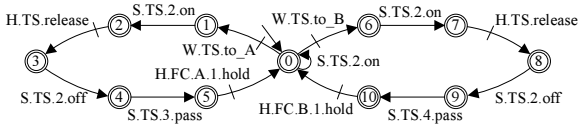
The synchronous product of all instances of linkage models for the TS ($LINKTS$) and each FC ($LINKFCA, LINKFCB$) are computed.

$$\begin{aligned}
COUNT &= \texttt{SYNC}[FCACOUNT, FCBCOUNT] \\
SPMEEX &= \texttt{SYNC}[SPLIT, MERGE, EXCL] \\
LINKTS &= \texttt{SYNC}[COUNT, SPMEEX]\ (256,736) \\
FCiSA &= \texttt{SYNC}[SFCi1, SFCi2]\ \forall i \\
LINKFCA &= \texttt{SYNC}[FCASA, FCAC]\ (20,50) \\
LINKFCB &= \texttt{SYNC}[FCASB, FCBC]\ (20,50)
\end{aligned}$$

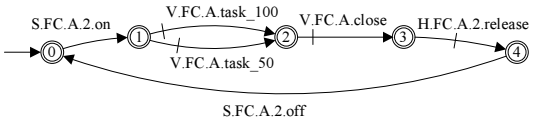
B. Local Specifications

Specification design is done as established by formalization of informal word models. Only two of them can be presented exemplary due to quantity. All specifications are local specifications.

To achieve the desired bottle sequence γ, a TS specification for proper feeding of FCA and FCB is designed, Fig. 7(a). The specification ensures that if a decision has been taken to feed a bottle from TS to e.g. FCA ($W.TS.to\_A.$) and



(a) Feeding specification (TS)



(b) Filling specification (FCA)

Fig. 7. Automata diagrams for prefix-closed specification languages

a bottle is available (*S.TS.2.on*), a decision for the next bottle can only be taken after the present bottle is both released by TS (*H.TS.release*) and arrived at FCA (*S.TS.3.pass*) and if the first stopper in FCA is prepared to stop the present bottle (*H.FC.A.1.hold*). The alternating run of the two possible feeding decisions is guaranteed by another specification.

A specification related to each FC is responsible for correct filling cycles. Fig. 7(b) shows this specification for FCA. The valve should only open (*V.FC.A.task_100* or *V.FC.A.task_50*), if a bottle is present under the nozzle (*S.FC.A.2.on*) and the bottle should not be released (*H.FC.A.2.release*) before the valve is closed again (*V.FC.A.close*).

Six local specifications for TS, synchronous product *TSSP* (560,1322), and seven for each FC, synchronous products *FCASP* and *FCBSP* each with (144,308), are defined to reflect the desired behavior of the entire filling shop. They ensure also buffer management and crash avoidance of bottles.

C. Decentralized Supervisory Control for the Filling Shop

The decentralized supervisory control architecture applied to the filling shop G is depicted in Fig. 8. G is controlled by three supervisors derived from local specifications, one for each FC and one for the TS. AND logic is used for fusing decisions of local supervisors to obtain a global decision, so a controllable event is enabled when all involved local supervisors permit it to occur. Subsec. V-B illustrated that all (local) specification languages K_{n} are generated by synchronous products of single automata. Therefore the solution for local decentralized control problems as defined in Subsec. II-B is presented for zero tolerance adopting the architecture in Fig. 8.

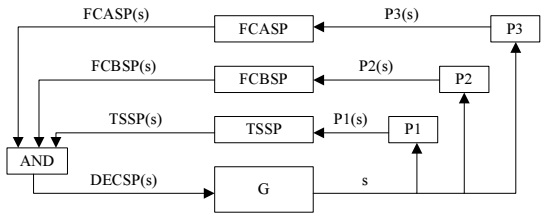


Fig. 8. Decentralized control architecture of the filling shop

$$L_{\mathrm{r,n}} = K_{\mathrm{n}} = L_{\mathrm{a,n}} \quad \forall n, n \in \{FCASP, FCBSP, TSSP\} \qquad (8)$$

Local plant models. In consideration of the distinct natural projections, each supervisor observes a different local model, namely *FCALOC*, *FCBLOC* and *TSLOC*, of the global plant model *PLANT* with local event sets Σ_{FCALOC}, Σ_{FCBLOC} and Σ_{TSLOC}, respectively. The FC do not share events with each other, $\Sigma_{\mathrm{FCALOC}} \cap \Sigma_{\mathrm{FCBLOC}} = \emptyset$, but both have common events with the TS, $\Sigma_{\mathrm{FCiLOC}} \cap \Sigma_{\mathrm{TSLOC}} \neq \emptyset \ \forall i$.

The classification of an event as a local event mainly derives from the availability of a physical connection or not. The local event sets are not detailed in this paper.

The observers local models are computed using TCT by computation of observers local product systems first, followed by synchronization with linkage models.

$$\begin{aligned}
FCAP &= \mathtt{PROJECT}[PLANT, \mathtt{IMAGE}[\Sigma_{\mathrm{FCALOC}}]] \\
FCALOC &= \mathtt{SYNC}[FCAP, LINKFCA]\ (128, 592) \\
FCBP &= \mathtt{PROJECT}[PLANT, \mathtt{IMAGE}[\Sigma_{\mathrm{FCBLOC}}]] \\
FCBLOC &= \mathtt{SYNC}[FCBP, LINKFCB]\ (128, 592) \\
TSP &= \mathtt{PROJECT}[PLANT, \mathtt{IMAGE}[\Sigma_{\mathrm{TSLOC}}]] \\
TSLOC &= \mathtt{SYNC}[TSP, LINK]\ (1024, 3840)
\end{aligned}$$

Controllability and nonconflicting check. Controllability of each supervisor w.r.t. its local model is checked by inspecting the control data. To be specific, the control data file provided by TCT is scanned to ensure that only controllable events need to be disabled.

Finally it has to be investigated, if all supervisors can work together properly by checking nonconflicting. Therefore the global extension of each supervisor is computed ex ante by adding selfloops at each state for all events that are not contained in the respective local event set of local specifications (inverse natural projection).

$$\begin{aligned}
FCiCD &= \mathtt{CONDAT}[FCiLOC, FCiSP] \ \checkmark \ \forall i \\
TSCD &= \mathtt{CONDAT}[TSLOC, TSSP] \ \checkmark \\
FCASPX &= \mathtt{SELFLOOP}[FCASP, [\Sigma \setminus \Sigma_{\mathrm{FCALOC}}]] \\
FCBSPX &= \mathtt{SELFLOOP}[FCBSP, [\Sigma \setminus \Sigma_{\mathrm{FCBLOC}}]] \\
TSSPX &= \mathtt{SELFLOOP}[TSSP, [\Sigma \setminus \Sigma_{\mathrm{TSLOC}}]]
\end{aligned}$$

$$\begin{aligned}
\mathtt{NONCONFLICT}[FCiSPX, TSSPX] &\quad \checkmark \ \forall i \qquad (9) \\
\mathtt{NONCONFLICT}[FCASPX, FCBSPX] &\quad \checkmark
\end{aligned}$$

The results show that *FCASP*, *FCBSP* and *TSSP* are proper local supervisors for decentralized supervisory control.

VI. IMPLEMENTATION AND TESTING

Events $\sigma \in \Sigma_{\mathrm{FCiLOC}}$ are related exclusively to sensors and actuators of their respective area FCi. The corresponding supervisors *FCASP* and *FCBSP* are directly implemented on their corresponding PLC using the C programming language. Σ_{TSLOC} contains events related to both FC (e.g. *H.FC.A.1.hold*, c.f. Fig. 7(a)) and the TS. To avoid installing additional cable joints, TS specifications containing

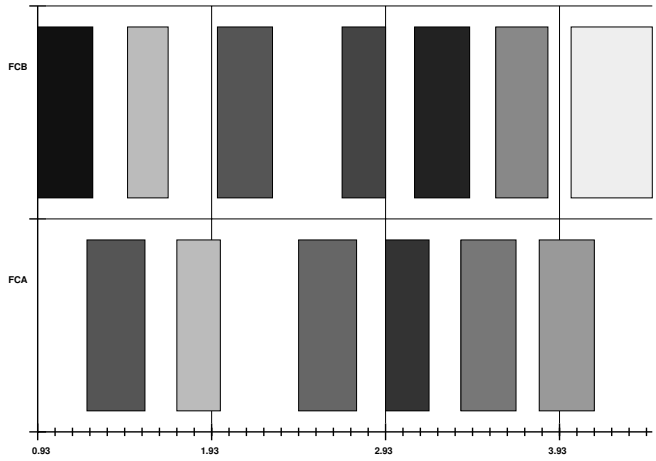


Fig. 9. Order allocation on FCB (top) and FCA (bottom)

events $\sigma \in (\Sigma_{\text{TSLOC}} \cap \Sigma_{\text{FSALOC}})$ or $\sigma \in (\Sigma_{\text{TSLOC}} \cap \Sigma_{\text{FSBLOC}})$ are implemented on the PC using SmallTalk programming language. The remaining part of *TSSP* is implemented as above. The implementation is based in the concept of polling for sensor reading. This is acceptable, due to the slow speed of the bottles on their conveyor belt and desirable to avoid problems caused by nondeterminism in the program cycle.

In contrast to object-oriented simulation, plant models are needed exclusively for supervisory control design in this work. Furthermore, the concept of standardized models is used for supervisor implementation. To be specific, specifications such as the filling specification shown in Fig. 7(b) are implemented independent from their location and subsequently concretized for FSA and FSB, respectively. Due to the fact that specifications can contain events related to several locations, such specifications are treated similar to linkage models. Specifying an attribute by stating its class as in Fig. 2 can be omitted in case of using SmallTalk programming language (un-typed language).

The decentralized supervisory control is tested in operation. Fig. 9 depicts a Gantt chart recorded while operation. A Gantt chart is a graphical representation of the duration of tasks against the progression of time and well established in planning and scheduling. From the figure, the alternating width and delayed start dates of the bars, signifying alternating production of either completely (wide) or half-full (small) filled bottles, is recognizable and verifies the production of the desired product sequence γ. FCA and FCB work to capacity in their respective steady state due to the effects of buffer management and crash avoidance specifications on the product flow. This is expressed by the increasing density and width of bars over time in Fig. 9.

VII. CONCLUSIONS AND FUTURE WORKS

The decentralized supervisory control approach has been applied to a complex filling shop in the context of a simplifying modeling methodology. Standardized modeling was introduced in details and its high usability was verified by application to a large-scale DES. Based on the models, the decentralized supervisory control was designed and implemented on a distributed control hardware and successfully tested in operation.

Future work should cover the implementation of a flexible order management and investigations regarding decentralized supervisory control architectures with an extended logic for fusing decisions of local supervisors. Due to the potential of TCP/IP, the inclusion of direct communication channels between the supervisors should be considered as an objective.

REFERENCES

[1] J.H. Richter and F.Wenck, "Hierarchical interface-based supervisory control of a bottling plant", *Proceedings of the 16th IFAC World Congress*, Praque, Czech Republic, 2005.

[2] P.J. Ramadge and W.M. Wonham,"Supervisory control of a class of discrete event processes", *SIAM J. Control and Optimization*, vol. 25, no. 1, 1987, pp. 206-230.

[3] F. Lin and W.M. Wonham,"On observability of discrete-event systems", *Inform. Sci.*, vol. 44, no. 2, 1988, pp. 173-198.

[4] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaija,"Supervisory control of discrete event processes with partial observation", *IEEE Trans. Automat. Contr.*, vol. 33, no. 3, 1988, pp. 249-260.

[5] K. Rudie and W.M. Wonham, "Think globally, act locally: Decentralized supervisory control", *IEEE Trans. Automat. Contr.*, vol. 37, no. 11, 1992, pp. 1692-1708.

[6] F. Lin and W.M. Wonham, "Decentralized supervisory control of discrete-event systems", *Inform. Sci.*, vol. 44, no. 3, 1988, pp. 199-224.

[7] S. Takai and T. Ushio, "A modified normality condition for decentralized supervisory control of discrete event systems", *Automatica*, vol. 38, no. 1, 2002, pp. 185-189.

[8] T. Yoo and S. Lafortune, "A general architecture for decentralized supervisory control of discrete-event systems", *in Discrete Event Systems: Analysis and Control*, Kluver Academic Publishers, 2000, pp. 111-118.

[9] G. Barrett and S. Lafortune, "Decentralized supervisory control with communicating controllers", *IEEE Trans. Automat. Contr.*, vol. 45, no. 9, 2000, pp. 1620-1638.

[10] W.M. Wonham, "Supervisory control of discrete-event systems", *LN, Dep. of Elec. and Comp. Eng.*, Univ. of Toronto, Canada 2004.

[11] P.J. Ramadge and W.M. Wonham,"Modular supervisory control of discrete-event systems", *Maths. of Control, Signals and Systems*, vol. 1, no. 1, 1988, pp. 13-30.

[12] P.J. Ramadge and W.M. Wonham,"The control of discrete event systems", *Proceedings IEEE, Special Issue on Discrete Event Dynamic Systems*, vol. 77, no. 1, 1989, pp. 81-98.

[13] Object Management Group, Inc., *Unified Modeling Language Specification*, Version 1.5, 2003.

[14] M.H. de Queiroz and J.E.R. Cury,"Modular supervisory control of large scale discrete event systems", *in Discrete Event Systems: Analysis and Control*, Kluver Academic Publishers, 2000, pp. 103-110.

[15] V. Chandra, Z. Huang, W. Qui and R. Kumar,"Prioritized composition with exclusion and generation for the interaction and control of discrete event systems", *Mathematical and Computer Modeling of Dynamical Systems*, vol. 9, no. 3, 2003, pp. 255-280.