# Online Fault Detection in the Modular Supervisory Control of an Experimental Manufacturing Cell

G. Kovács*, L. Piétrac†, B. Kiss*, E. Niel†
*Department of Control Engineering and Information Technology,
Budapest University of Technology and Economics, Budapest, Hungary
†Laboratoire Ampère, INSA de Lyon, Villeurbanne, France

***Abstract*— This paper presents an application of watchdog based fault detection methods to the supervisory control of an experimental manufacturing cell. Fault detection is implemented by carrying out slight modifications on the previously designed, modular supervisory control architecture, without using additional sensor devices. Different strategies for avoiding fault propagation are also presented.**

## I. Introduction

The need for fail-safe and fault-tolerant systems has arisen significantly in the latest decade. In some applicational fields, e.g. in automotive industry or particularly in manufacturing systems, the dependability of systems has become a crucial point of controller design [1]. In case of large-scale, sophisticated systems, such as manufacturing lines, the theory of supervisory control was introduced to assure safe operation complying with the formal specifications [2].

In the field of discrete event systems, several methods have been proposed for fault detection, failure identification and diagnosis, see, for example, [3] or [4]. However, despite the availability of general and theoretically proven solutions, these methods are often hard to use in everyday practice. On the other hand, there exist well-known, practice-oriented solutions for the problem of fault detection, which are familiar to system engineers. Their greatest disadvantage is their lack of formalism and their need for intuitive human intervention. However they have proven to be useful through many years, their application cannot guarantee formal proof of safe behavior for fault-critical systems.

Authors have proposed a practice-oriented, low cost, online fault detection method based on the well-known architecture of watchdog structures [5], [6]. The presented method is placed in the framework of Supervisory Control Theory, and steps of the methodology can be easily automated by suitable algorithms.

This paper presents the application of the proposed fault-detection methods on an experimental manufacturing cell. The controller of the plant is designed using the principle of modular control [7], and the presented operations do not refer only to the actual system, but are able to illustrate the general methodology of fault detection using watchdog structures.

The remaining part of the paper is organized as follows. Section II gives a short overview on Supervisory Control Theory, and on the proposed watchdog-based fault-detection methods. Section III presents the experimental manufacturing cell and its controller, while Section IV gives the procedure of integrating watchdog-based fault-detection methods. Section V concludes the paper.

## II. Preliminaries

### A. Supervisory Control Theory

For the sake of self-contained presentation some notions of Supervisory Control Theory (SCT) are summarized. For further details, the reader is referred to [8].

In the framework of SCT, systems are modelled by Finite State Machines (FSMs). The system $G$ is described by the 5-tuple $G = \{Q^G, \Sigma^G, \rho^G, q_0^G, Q_m^G\}$, where $Q^G$ is the set of states, $\Sigma^G$ is the set of events as alphabet, $\rho^G = Q^G \times \Sigma^G \to Q^G$ is the partial transition function, $q_0^G$ is the initial state and $Q_m^G$ is the set of marking states. The event set $\Sigma$ can be devided to the disjoint sets of controllable and uncontrollable events, so that $\Sigma^G = \Sigma_C^G \cup \Sigma_U^G$, where $\Sigma_C^G \cap \Sigma_U^G = \emptyset$. The plant $G$ can be considered as a generator, which outputs the symbols of $\Sigma$.

The goal of supervisory control is to synthesize a supervisor, which is capable to restrict the operation of the plant $G$ to meet the specification desribed by the automaton $E$. If the system happens not to be controllable regarding to the specifications, the supremal controllable sublanguage can be found [9], [10]. The supervisor itself is a function, describing which controllable events should be enabled and disabled in the particular states of the plant. Based on the supervised system $S/G$, the controller model $C = \{Q^C, \Sigma^C, \rho^C, q_0^C, Q_m^C\}$ can be extracted by the selecting one of possible trajectories in order to assure deterministic behavior. The controller model can be extended by a control map, $\Theta = Q^C \times \Sigma_C^C \to \{0, 1\}$, describing that the controllable events should be disabled or not in a particular state of the controller model.

### B. Watchdog-based fault detection

Although watchdog structures are used for fault detection since the early ages of digital computing, their use have not been formalised in the SCT framework for a long time. In this paper, the most important features of wathcdog-based fault detection will be given, for further details the reader is referred to [5], [6].

*1) Definitions:* Watchdogs are used to observe the completion of a task. A *task*, denoted by $T$, is a part of the trajectory of the controller model, which can be clearly distinguished from other activities: $T_i = \{Q_i^T, \Sigma_i^T, \rho_i^T, q_{0,i}^T, Q_{M,i}^T\}$, where $Q_i^T \subseteq Q^C$ and $\Sigma_i^T \subseteq \Sigma^C$. If there exists one and only one transition leaving the initial state of the task $T_i$, $q_{0,i}^T$, the task is said to be possible to put under the guard of a watchdog. In the sequel, only such tasks are considered. The controllable event, corresponding to the transition leaving $q_{0,i}^T$, and therefore indicating the start of the task, will be referred to as the *command event* of the task $T_i$ and will be denoted by $\sigma_i^{CMD} \in \Sigma_C^C$. The events associated to the transitions leading to the final states, $q_{m,i} \in Q_{M,i}^T$ of the task indicate its succesful completion, so will be referred to as *confirmation events*, and will be denoted by $\sigma_{i,j}^{CONF} \in \Sigma^C$. The controller comprises a set of alarm handling states, denoted by $Q_{AH}$, which initialize alarm handling procedures. If an alarm event, generated by the watchdog, occurs during the execution of a task, an alarm handling procedure, depending on the task has to be started. To do so, the controller model should pass to one of its alarm handling states, $q_{AH,i} \in Q_{AH}$, defined by the function $\xi : T \rightarrow Q_{AH}$. The definition of the alarm handling procedure is left open to the system designer.

*2) Watchdog structures:* Watchdogs are counter-timer structures, equipped with a memory register, a comparator, and an alarm logic. They are used to observe whether a given task is completed succesfully in a predefined time period, and their functionnality can be pictured as follows. In the idle state of the watchdog, the value corresponding to the desired time period is loaded into the memory register, and then the counter is enabled, so the watchdog passes to its running state. The actual value of the counter and the memory register are compared each clock cycle, and if the former reaches the latter, an alarm signal is emitted and the watchdog is driven to its alarm state. The alarm logic maintains the alarm signal until its reset. If the watchdog is stopped before the counter reaches its final value, the counter is deactivated, and the watchdog returns to its idle state. So, the Idle $(q_0)$, Running $(q_1)$ and Alarm $(q_2)$ states are required for the discrete-event model of the watchdog. The START, STOP and RESET events are generated by the controller and are therefore controllable, while the ALARM event is generated by the watchdog itself, so is considered to be uncontrollable.

In distributed control environments, it is vital to notify other controllers on the failure of a subsytem to avoid fault propagation. Assume that the subsystem $G_1$ is under the supervision of $C_1$, and is equipped with a watchdog. The controller $C_2$ is associated to the subsytem $G_2$, and for some operations, $G_2$ needs resources represented by $G_1$. The communication of faults between the two controllers can be assured by using a simple query-response philosophy, allowing $G_2$ to query the state of the watchdog associated to $G_1$. Communication is taken place using the controllable QUERY event, and the uncontrollable R_IDLE and R_ALARM response events, which indicate whether a fault has been detected by the watchdog. Note

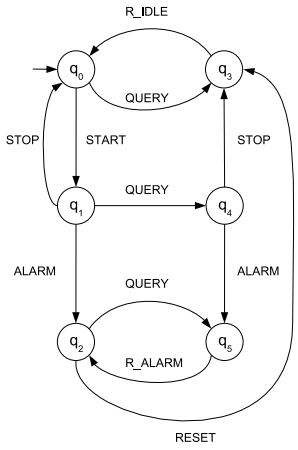


Fig. 1. Discrete-event model of the watchdog

that while the watchdog is running, there is no relevant information on the failures of $G_1$. The QUERY event leads the watchdog to a so-called query state, where the appropriate response event is immediately generated. The extended model is shown in Fig. 1.

*3) Principles of watchdog-based fault detection techniques:* As mentioned above, watchdogs are capable to indicate if a task has not been completed in a given time period, so they can be used to detect the consequences of even complex failures without additional sensor devices. To implement watchdog-based fault detection methods, only a few simple modifications should be carried out on the controller models of previously designed supervisory control structures. Here only the principles of the methods will be presented, detailed formalism is given in [6].

We show first how to put a task $T_i$ under the guard of a watchdog depicted in Fig. 1. To detect the possible failures occuring during the execution of a task, the watchdog should be started before the beginning of the given task. At first a new state, ${q_{i,0}^T}'$ should be defined and all transitions leading to the initial state of the transition, $q_{i,0}^T$, should be redefined so that they lead to ${q_{i,0}^T}'$. Only one transition, associated to the START event, launching the watchdog, should be defined from ${q_{i,0}^T}'$ to $q_{i,0}^T$. Similarily, the watchdog should be stopped after the task is completed, i.e. after the generation of the confirmation event. To do so, new states $q'_{m,i} \in {Q_M^T}'$ should be defined associated to the final states of the task, namely $q_{m,i} \in Q_M^T$. Transitions leaving the states of $Q_M^T$ should be redefined so that they lead to the appropriate states of ${Q_M^T}'$, and a transition associated to the STOP event should lead from the states of ${Q_M^T}'$ to the original entry states of the transitions leaving the corresponding final states. Transitions originating from states not included in the state set of the task and leading to any of its final states, should be redefined so that they lead to the corresponding state of ${Q_M^T}'$. To launch the alarm handling procedure, transitions leaving any state of the task but its first and last states should be defined with the appropriate alarm handling state as their entry state.

For modular and distributed environments, two strategies are presented, using the previously introduced query-response feature of extended watchdog structures. The Wait-for-OK strategy is suitable for simple systems, while the Multimodal strategy deals with multiple operational modes, and therefore applicable for more complex components.

Suppose that tasks of $G_1$ are not overlapping, they use the same watchdog, so querying the watchdog provides information on the whole functionnality of $G_1$. If $G_2$ cannot operate without a resource represented by $G_1$, or the failure of $G_1$ causes faulty behavior of $G_2$, it is vital to check the status of the watchdog associated to $G_1$ before starting the given operation. In this case, the Wait-for-OK strategy provides a solution for starting the given operation only if no failure is detected in $G_1$. Let us assume that the operation of $G_2$ needing the resource represented by $G_1$ is started from the state $q_j \in Q_2$. The status of the watchdog associated to $G_1$ should be checked before entering $q_j$, and $q_j$ can be entered only if no failure is detected. Therefore two new states, $q'_j$ and $q''_j$ should be added and transitions leading to $q_j$ should be redefined to enter $q'_j$. Two new transitions, one leading from $q'_j$ to $q''_j$ (respectively $q''_j$ to $q_j$), associated to the QUERY (respectively R_IDLE) event of the watchdog of $G_1$ should be defined. It ensures that the given task is started only if the R_IDLE response event is generated, i.e. no fault has occured in $G_1$. Note that the watchdog generates an R_IDLE response upon its reset, so the given operation of $G_2$ can be started immediately upon the handle of the failure in $G_1$.

Multimodal strategy can be applied if the given subsystem has more operational modes, generally a nominal and a degraded mode. While $G_2$ needs resource represented by $G_1$ in the nominal mode, in case of lack of this resource, it can switch to its degraded mode, where it can continue its operation without using the given resource. Here the approach of Kamach [11],[12] will be used to deal with mode changes, assuming that each operational mode has its own controller model. Controller models pass to the so-called inactive states ($q^1_{IA}$ and $q^2_{IA}$ for the nominal and degraded modes, respectively) upon the deactivation of the modes. Upon the reactivation of the nominal (respectively degraded) mode, the controller model passes from its inactive state $q^1_{IA}$ (respectively $q^2_{IA}$) to its return state, $q^1_{RET} \in Q^C_1$ (respectively $q^1_{RET} \in Q^C_1$).

Assume that in nominal mode, the operation of $G_2$ needing the resource of $G_1$ is started from the state $q_k$. Then, before entering $q_k$, the status of the watchdog associated to $G_1$ should be queried, and $q_k$ should be entered only if no failure is detected. Otherwise, the controller should deactivate its nominal mode by passing to its inactive state. For, two new states, $q_k{}'$ and $q_k{}''$ should be defined, and transitions entering to $q_k$ should be redefined so that they enter $q_k{}'$. From $q_k{}'$ to $q_k{}''$ a transition associated to the QUERY event should be defined. Then the controller model should start the given task, i.e. pass to $q_k$ if the R_IDLE event is generated, or pass to its inactive state $q^1_{IA}$ if the R_ALARM event is generated. From the inactive state, a transition associated to the R_IDLE event and leading to the return state should be defined to allow the reactivation of the nominal mode. The strategy to follow is the same in case of the degraded mode. At each duty cycle, status of the watchdog associated to $G_1$ should be queried, and if found to be in its idle state, the degraded mode should be deactivated. Modifications are similar to those in case of the nominal mode.

It can be proved that the extension presented here does not influence the behaviour of the supervised system if no failure occurs [6].

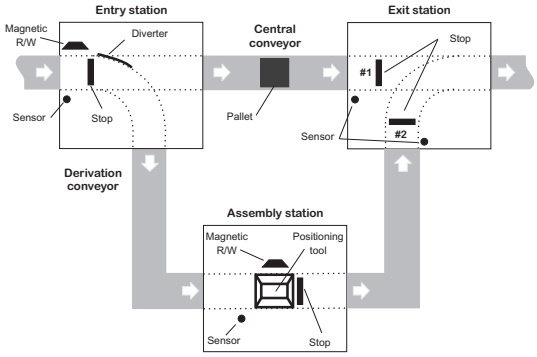


Fig. 2. Sketch of the assembly cell

## III. SUPERVISORY CONTROL OF THE EXPERIMENTAL MANUFACTURING CELL

### A. Presentation of the system

The experimental manufacturing line is located at the site of AIP-RAO, in Villeurbanne, France. The line is built up from six assembly cells, connected by a central conveyor. Workpieces are carried by pallets equipped with rewritable magnetic labels, on which the order of the assembly cells to be passed are stored. In this paper, only one of the cells will be dealt with.

As shown in Fig. 2, the assembly station is served by a derivational conveyor, connected to the central conveyor by an entry and an exit station. Pallets travelling on the central conveyor are stopped at the entry station, and their magnetic label is read. According to the information read out, they are dispatched towards the derivational conveyor or continue their way on the central conveyor, according to the configuration of the pneumatic diverter system. At the assembly station, the pallets are blocked by a pneumatic stop, and their magnetic label is read out again. The assembly process is modelled by a positioning operation, during which a pneumatic positioning tool is used to keep the pallet in a fixed position. After their release, the label of the pallets is updated, and they continue their way and re-enter the central conveyor through the exit station. The exit station acts as a traffic policeman, which stops the arriving pallets, and in order to avoid collisions or stuck of pallets, allows only one at a time to enter the next session of the central conveyor, with a priority of the derivational conveyor over the central one.

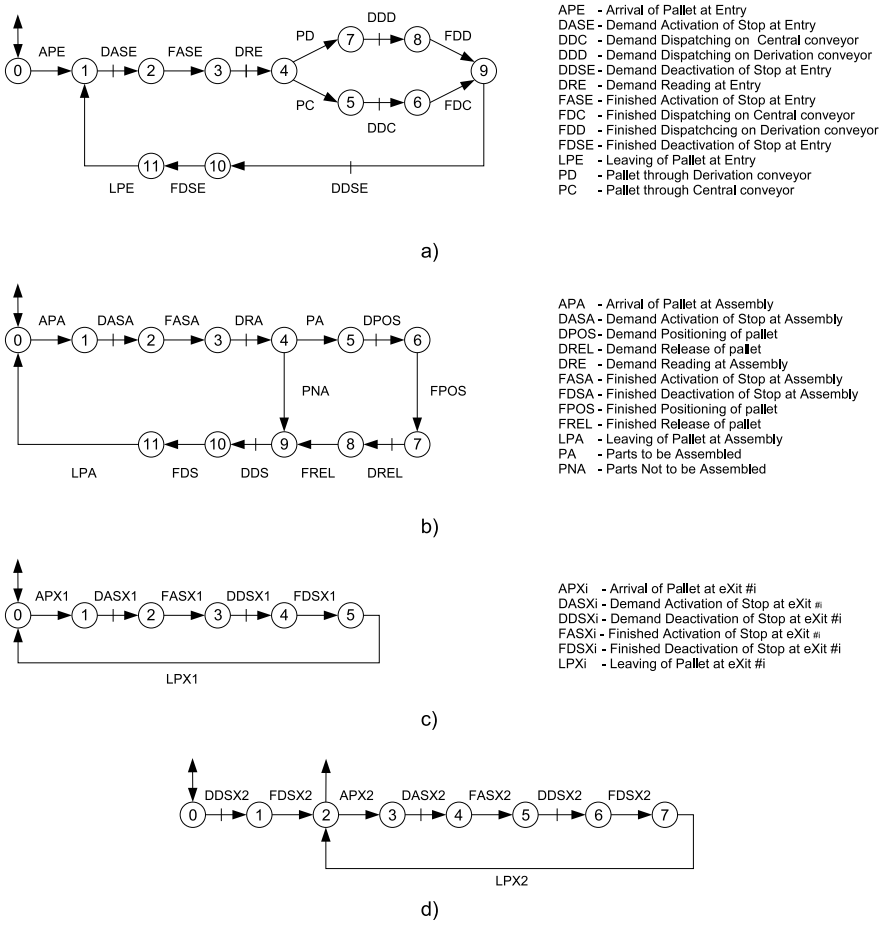


Fig. 3. Controller models. a) Entry station b) Assembly station c) Exit station #1 d) Exit station #2

### B. Supervisory control architecture

The aim for desinging a supervisory control structure for the presented manufacturing cell is to synchronize the operation of the components in order to achieve desired operation. Following the approaches presented in [7] and [13], we have chosen to implement a modular supervisory controller. However the controller is implemented on a single PLC, design principles and software realization are also use the modular approach.

Since the aim of this paper is to present how watchdog-based fault detection techniques can be integrated to existing supervisory control architecture, the synthesis of the supervisors is not presented here. Readers interested in the details of supervisor synthesis are referred to the original paper [7].

Resulting controller models are given by Fig. 3. Here the control map is not defined, since only one controllable transition leaves each state of the controller model, so it is straightforward that only that transition should be enabled.

## IV. Fault detection and failure handling

### A. Fault situations

The configuration of pneumatic components ensures that in case of the cut of pressure the stops are in their lowered position and diverters route the pallets towards the central conveyor, so their failures do not cause critical problems. However, in case of the exit station, where cooperation of individual pneumatic components is needed, their faults can cause the stuck of pallets and therefore complete block of the central conveyor. At the assembly station, the fault of the operation, modeled by a simple positioning, can also cause critical malfunctions. Therefore, fault detection methods should be implemented to monitor the operation of the assembly station and the exit station. However, the operation of the entry station should be also adjusted according to failure situations.

To implement fault-detection methods, at first independent watchdogs should be associated to the subsystems where the occurence of failures is assumed, namely to the assembly station and the exit station. To distinguish their events, the postfixes '_A' and '_X' are added for the watchdogs associated to the assembly station and the exit station, respectively.

### B. Fault detection and failure handling at the assembly station

The failure of the assembly station is indicated by the time elapsed between the positioning and the release of the pallet by the positioning tool, so the task to be put under the guard of the watchdog is given by the trajectory passing through the states $q_5, q_6, q_7, q_8, q_9$. Therefore, the demand for positioning will be used as

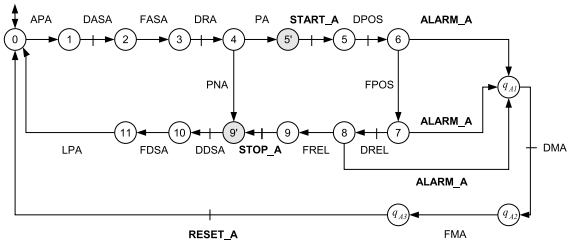


Fig. 4. Extended controller model of the assembly station

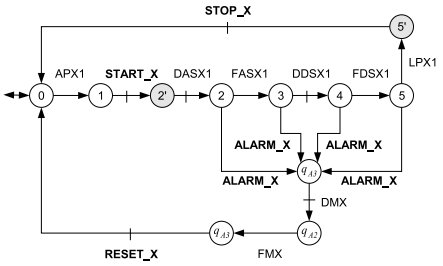


Fig. 5. Extended controller model of the exit station

command event, so $\sigma_1^{CMD} = \text{DPOS}$ and the succesful release of the pallet will be used as confirmation event, so $\sigma_1^{CONF} = \text{FREL}$. According to the principles presented in Section II-B.3, the new states $q_5'$ and $q_9'$ are added. In order to start the watchdog before executing the task, the transition associated to DPOS is redefined to leave $q_5'$, and a new transition leading from $q_5'$ to $q_5$, associated with START_A is added. Similarily, to stop the watchdog after the succesful completion of the task, the transition leaving $q_9$ is redefined so that it leaves $q_9'$, and a new trasition associated to STOP_A is defined from $q_9$ to $q_9'$. The entry state of the transition leading from $q_4$ to $q_9$ should be modified to $q_9'$.

Failure handling procedure is modeled by the intervention of a human operator. At the alarm handling state $q_{A1}$, the DMA event is generated, signaling the demand for maintenance. The succesful intervention is indicated by the uncontrollable FMA event. As we can assume that the pallet is removed from the assembly station, which is manually re-initialized by the operator, the controller model passes to its initial state after the handling of the failure and resetting the watchdog.

The extension of the controller model according to the principles presented in Section II-B.3 is illustrated by Fig. 4, where newly added states and transitions are indicated by their grey background and bold labels, respectively.

### C. Fault detection and failure handling at the exit station

The failure of the exit station is indicated by the length of time the pallets arriving at the derivational conveyor stay blocked. Since they have priority over the ones arriving on the central conveyor, a failure can be assumed if they are not leaving the exit station in a relatively short period. The task to be put under the guard of the watchdog is defined by the trajectory leading through states $q_1, q_2, q_3, q_4, q_5$. Therefore, demanding the activation of the pneumatic stop of the derivational conveyor should be used as command event, so $q_2^{CMD} = \text{DASX1}$, and the signal of the presence sensor indicating that the pallet has left should be the confirmation event, so $\sigma_2^{CONF} = \text{LPX1}$.

The failure handling procedure is similar to the one used at the assembly cell. Here the controllable DMX event is used to demand the intervention, and its completion is indicated by the uncontrollable FMX event. The extension of the controller model is illustrated by Fig. 5.

### D. Failure handling at the entry station

The entry station plays an important role in the fail-safe operation of the assembly cell. However it is assumed not to break down, it is responsible for the avoidance of damage caused by collisions and blockage of the system. In case of the failure of the assembly cell, the objectives are the avoidance of stuck of pallets at the assembly station, and ensuring the continuous operation of the line. To meet these objectives, pallets should be redirected towards a manually operated backup cell, in which any assembly operation can be carried out by human operators. In case of the failure of the exit station, the central conveyor is assumed to be blocked, so the only possible solution is the prohibition of entering pallets to the blocked area to avoid stuck of workpieces. Although it means suspending the operation of the whole line, pallets have to be stopped at the entry station and not allowed to continue their way until handling the failure of the exit station.

For handling the failures of the assembly cell, at first the operation of the entry station in degraded mode should be defined by the followings. Pallets arriving at the entry station are stopped, and their label is read out. If they are found to be ordered to pass by the assembly cell, their label is rewriten by replacing the actual cell by the manually operated one. Then, the stop is deactivated, and pallets continue their way on the central conveyor. Assume that the inactive and return states of the operational modes have been already defined.

In nominal mode, the status of the assembly cell should be queried before dispatching a pallet towards the derivational conveyor at the state $q_7$. Therefore, two new states, $q_7'$ and $q_7''$ should be added, and the transition associated to PD, leading from $q_4$ to $q_7$ should be modified so that it leads to $q_7'$. The query of the watchdog is represented by the transition leaving $q_7'$ and leading to $q_7''$, associated with the QUERY_A event. The nominal mode should be deactivated or the pallet should be dispatched towards the derivational conveyor depending on the response event, so transitions should be defined leaving $q_7''$ and leading to $q_{IA}^1$ and $q_7$, associated with the events R_IDLE_A and R_ALARM_A, respectively. To enable the reactivation of the nominal mode if the failure of the assembly cell is handled, a transition leading from $q_{IA}^1$ to the return state, namely $q_7$, associated to the R_IDLE_A response event should be defined. The extension of the controller model of the degraded mode is similar. The status of the assembly cell should be queried before rewriting the label, and the degraded mode should be deactivated upon the R_IDLE_A response. Reactivation of the degraded mode is

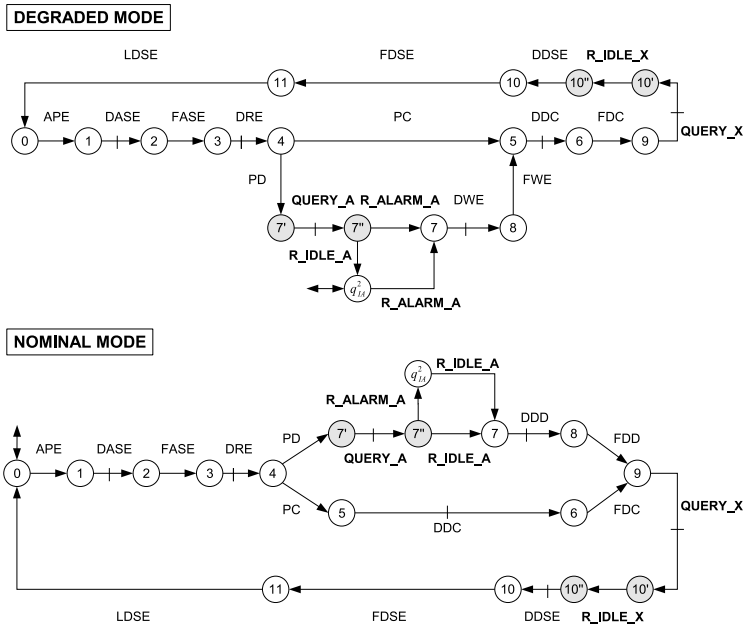


Fig. 6. Extended controller model of the entry station

forced by the R_ALARM_A event.

For handling the failures of the exit station, only the Wait-for-OK strategy can be used, since the pallets have to be blocked at the entry station until the failure is handled and therefore there is no degraded mode to swith to. The status of the watchdog associated to the exit station should be queried before allowing a pallet to continue its way, i.e. before deactivating the stop of the entry station. Therefore, to the controller model of the nominal mode two new states, namely $q_9'$ and $q_9''$ should be added, and the transition associated to the event DDSE, i.e. the deactivation of the stop, should be redefined to leave not $q_9$ but $q_9''$. Two new transitions, one leading from $q_9$ to $q_9'$ associated to the QUERY_X event, and one leading from $q_9'$ to $q_9''$ associated with the R_IDLE_X response event should be defined to let the deactivation of the stop demanded only if the watchdog associated to the exit station is in its idle state, e.g. no failure has occured. Controller model of the degraded mode should be extended similarily.

The complete extension of the controller models, including both strategies, is illustrated by Fig. 6.

## V. Conclusion

In this paper the application of simple, online fault-detection strategies to the modular supervisory control of an experimental manufacturing cell has been presented. It has been demonstrated that failures of subsystems controlled by different modular components can be detected without using additional devices (e.g. sensors) and that there exists a simple yet powerful method for the communication of failures in order to avoid fault propagation.

However, in this paper only one cell of the experimental manufacturing line has been presented. Future works include the handling of fault propagation between cells and the comparision of the presented solution with other methodologies.

## References


[1] G. Isermann, "Model-based fault-detection and diagnosis – status and applications," *Annual Reviews in Control*, vol. 29, pp. 71–85, 2005.

[2] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Boston: Kluwer Academic Publishers, 1999.

[3] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Failure diagnosis using discrete-event models," *IEEE Trans. Control Systems Technology*, vol. 48, pp. 105–120, 1996.

[4] S. Zad, R. Kwong, and W. Wonham, "Fault diagnosis in discrete-event systems: Framework and model reduction," *IEEE Trans. Automatic Control*, vol. 48, pp. 1199–1211, 2003.

[5] G. Kovács, B. Kiss, and E. Niel, "Watchdog - a practical approach of fault detection," *Proc. 12th IFAC International Symposium on Information Control Problems in Manufacturing*, vol. 1, pp. 327–333, 2006.

[6] G. Kovács, L. Piétrac, B. Kiss, and E. Niel, "On the formalization of integrating watchdog structures into supervisory control architectures," *European Control Conference*, 2007, accepted for publication.

[7] M. Nourelfath and E. Niel, "Modular supervisory control of an experimental automated manufacturing system," *Control Engineering Practice*, vol. 12, pp. 205–216, 2004.

[8] W. Wonham, *Notes on Control of Discrete Event Systems*. University of Toronto, 2002.

[9] R. Kumar, V. Garg, and S. Marcus, "On controllability and normality of discrete event systems," *Systems & Control Letters*, vol. 17, pp. 157–168, 1991.

[10] R. Brandt, V. Garg, R. Kumar, F. Lin, S. Marcus, and W. Wonham, "Formulas for calculating supremal controllable and normal sublanguages," *System & Control Letters*, vol. 15, pp. 157–168, 1990.

[11] O. Kamach, S. Chafik, L. Piétrac, and E. Niel, "Representation of a reactive system with different models," *Proc. IEEE International Conference on Systems*, vol. 4, pp. 263–267, 2002.

[12] O. Kamach, L. Piétrac, and E. Niel, "Multi-model approach to discrete event systems: Application to operating mode management," *Mathematics and Computers in Simulation*, vol. 70, pp. 396–407, 2006.

[13] L. Piétrac, S. Chafik, and E. Niel, "Théorie du contrôle par supervision: Un exemple d'une application décentralisée sur un système de production manufacturière," *APII-JESA*, vol. 38, pp. 315–346, 2004.