Available online at www.sciencedirect.com

Mathematics and Computers in Simulation 70 (2005) 394–407

MATHEMATICS AND COMPUTERS IN SIMULATION

www.elsevier.com/locate/matcom

Multi-Model approach to discrete events systems: Application to operating mode management

Oulaid Kamach*, Laurent Piétrac, Éric niel

Laboratoire d'Automatique Industrielle, INSA de Lyon, Bat. Antoine de St-Exupery, 27 Av. Jean Capelle, 69621 Villeurbanne cedex, France

Available online 19 January 2006

Abstract

In this paper, we propose a framework for designing suitable switching control decisions for discrete event systems (DES) whose structures change as they develop in different operating modes. Control decisions consist of either an event in a sequence to occur *enabling an event* or preventing the event from taking place *disabling an event*.

Our contribution enables to adopt different modeling approaches and ensures switching between all designed process models when there is commutation between the operating modes. Thus, in the context of supervisory control theory (SCT), we propose that each model automaton represents process functionning in a specific operating mode.

Specifications imposed on any operating mode could be conflicting. An attractive alternative is switching control, in which a different controller is applied to each operating mode [P. Charbonnaud, F. Rotella, S. Médar. Process Operating mode Monitoring Process: Switching Online the Right Controller, IEEE transactions on systems, Man and Cybernetics, Part C 31(1) pp 77-86. 2002; M. Zefran, J. Burdick, Design of switching controllers for systems with changing dynamics, in: Proceedings of the 37th conference on Decision and control, 1998, pp. 2113–2118]. Control of process functionning means that both process and specification models must be associated with one specific operating mode.

Based on supervisory control theory, our work focuses on operating mode management in particular when the process is subject to failure. The adopted approach (multi-model) assumes that only one attempted operating mode is activated at any one time, while the others are considered desactivated. The problem of commutation and tracking between all designed models (process and specification) is formalised by the proposed framework. In this context, several questions are raised. Is the process engaged in a state which is compatible with the atteined mode ? Are the specifications consistant with each starting state ?. Are the specification conflicting ? Can all defined states be reachable ?

To answer correctly these questions, a mode switching mechanism must be formalised.

Keywords: Multi-model approach; Operating mode management; Supervisory control theory; Switching controllers; Systems with changing dynamics

1. Introduction

A discrete event system (DES) is a special type of dynamic al systems. The "state" of these systems changes at discrete instants in time and the term "event" represents the occurrence of discontinuous change. Different DES models are

* Corresponding author.
E-mail address: oulaid.kamach@insa-lyon.fr (O. Kamach)

0378-4754/$32.00 © 2005 Published by Elsevier B.V. on behalf of IMACS.
doi:10.1016/j.matcom.2005.11.008

currently used for specification, verification and synthesis. The DES formalism allows the analysis and the assessment of different qualitative and quantitative properties of the existing physical systems. Therefore, if the technological development extends the functionalities of embedded controllers and their safe reliability, it can steadily increase the complexity of both modeling and synthesis processes. In fact, DES controls are increasingly to technologies whose main objectives are to obtain optimum performance characteristics requiring formal validation. The supervisory control theory (SCT) of Ramadge and Wonham [9,10] can be very helpful in relation to these performance characteristics, first by offering conventional synthesis of controlled dynamic invariant systems through feedback and, second, by verification of properties such as controllability and non blocking. However, in this theory, the complete plant (process) often results in a combination of components. The size of the resulting model increases exponentially with the number of components and controller synthesis becomes a laborious process. Component number will increase if we must also consider different process structures associated to different operating modes. Keeping in mind the advantages of SCT, we extend this theory in order to eliminate the following drawbacks.

(1) All components composing the global process are not required in each operating mode,
(2) Defined specifications for each model can be conflicting when commuting from one mode to another and can lead to the system blocking.

Obviously, when commutation is needed, it comprises changing both the structure and specifications, i.e., the several models are needed but not at the same time. Adapting the divide and conquer strategy makes this management easier and a multi-model approach seems natural.

The multi-model approach involves representing complex systems by a set of simple models, each of which describes the system in a given operating mode. Changing mode and process structure raises problems such as mode switching and model tracking. By studying mode switching, we define the commutation condition, model connection, process model tracking and the way the corresponding specifications are activated:

- commutation condition identifies the process states in which the operating mode is required;
- model connection results in global strategy for the controlled process;
- model tracking is defined for the process because of commutation event localisation;
- activating specifications means that the requirement must be consistant with the process starting state.

The paper is organized as follows: Section 2 introduces briefly SCT which is the basis of our approach. Sections 3 and 4 are devoted to the formalization of the problem of commutation modes and introduce the process tracking and the specification accommodation respectively. Section 5 comprises an illustrative example and Section 6 conclludes the paper.

2. Framework

This section introduces the main emboding SCT and the problem of considering operating modes. The original SCT framework is based on distinguishing process and specification models. The process is seen as an uncontrolled DES and is modeled by an automaton $G = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is a set of states, Σ is the alphabet, δ is the transition function (partial function), a partial function $\delta : Q \times \Sigma \longrightarrow Q$, $q_0 \in Q$ is the initial state, and $Q_m \subseteq Q$ is the subset of marker states. For any event $\Sigma \in \Sigma$ and state $q \in Q$, if $\delta(q, \Sigma)$ is defined, (i.e., there is some state in the process that we can reach from q via σ), we write $\delta(q, \sigma)!$. The definition of δ can be extended to a partial function for $Q \times \Sigma^*$, such that $(\forall\sigma \in \Sigma)(\forall s \in \Sigma^*)$, $\delta(q, s\sigma) = \delta(\delta(q, s), \sigma)$ and $\forall q \in Q$, $\delta(q, \epsilon) = q$. The set Σ^* contains all possible finite strings (i.e., sequence) over Σ plus the null string ϵ. The language generated by G, denoted by $L(G)$, is also called the closed behavior of G:

$$L(G) := \{s \in \Sigma^* \mid \delta(q_0, s)!\}.$$

This language describes all possible event sequences that the DES can undergo. Thus $L(G) \subset \Sigma^*$. The marked language is $L_m(G) := \{s \in \Sigma^* \mid \delta(q_0, s) \in Q_m\}$.

G is said to be a recognizer for $L_m(G)$. The marked states are used to model “deadlock” or “livelock” blocking.

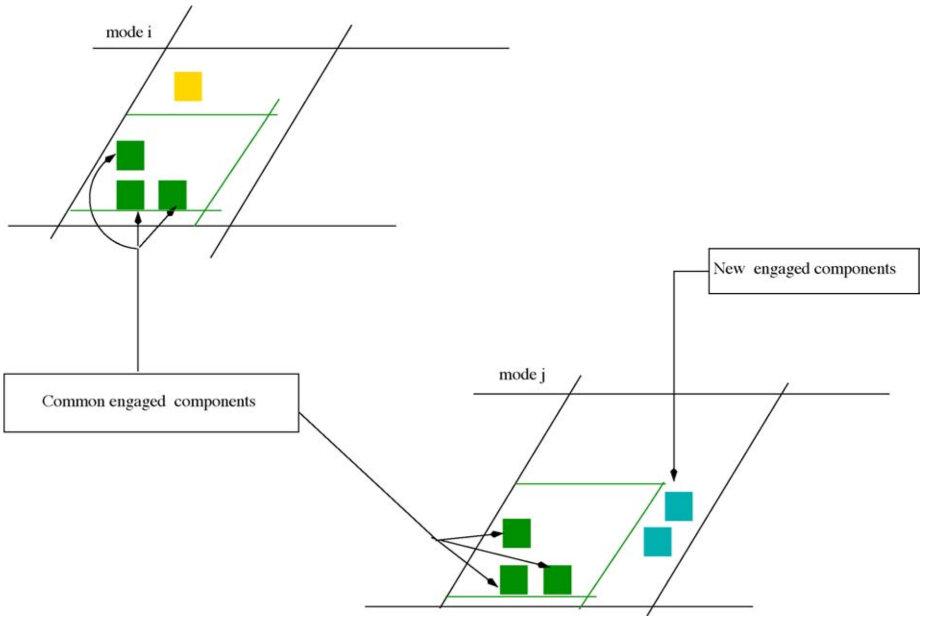

Fig. 1. Changing structure process.

A specification model S is also an automaton ($S = (X, \Sigma, \xi, x_0, X_m)$) and the controlled DES S/G is obtained by composition of G and S, i.e., $S/G = (X \times Q, \Sigma, \xi \times \delta, (x_0, q_0), X_m \times Q_m)$ where $\xi \times \delta : X \times Q \times \Sigma \longrightarrow X \times Q : (x, q, \sigma) \mapsto (\xi(x, \sigma), \delta(q, \sigma))$ provided $\xi(x, \sigma)!$ and $\delta(q, \sigma)!$. For more details on SCT, the reader is referred to [1,12,11].

To establish such supervision on G, we partition the set of events Σ into the disjoint sets Σ_c of controllable events and Σ_{uc} of uncontrollable events. Controllable events are those events whose occurrence can be prevented (i.e., may be disabled). Uncontrollable events are those events which cannot be prevented and are deemed premanently enabled.

In most cases, the system can be subdivided into several subsystems. Similarly, process and specification models are the combinations of several simples models. Because of state graph manipulation, a current SCT application problem is the explosion in state numbers as component numbers increase. This explosion is often dealt with by performing horizontal (modular or decentralized) or vertical (hierarchical) decomposition of the underlying control problem [6–8,14,3,13].

Let us take as a motivating example, a production system, which must manufacture various products and react rapidly to failures. Different system use corresponds to different operating modes. Adjustment and maintenance modes are examples of other operating modes that are also necessary for the production system. However, a system does not require all components in each operating mode (as shown in Fig. 1). Furthermore, specifications differ for every operating mode because the objectives of each one are different. Previous approaches are difficult to put into practice on a multi-operating mode system because they consider only a single model of the system and because of multiple specifications may be in conflict.

We assume that the process model can change its structure when commuting from one operating mode to another by engaging new components. For instance, Fig. 1 shows that there are common components engaged in two operating modes and some components do not contribute to production in mode i, but they intervene when a commutation from mode i to j is performed.

3. Commutation process models

This section focuses on guaranteed functioning under failure which, whilst causing degraded production, does allow continuity of service.

Reactive systems must be flexible to perform under controlled fault. This flexibility involves taking into account different operating modes. We are interested in modeling these operating modes by applying a multi-model approach, which involves designing model process control for each operating mode.

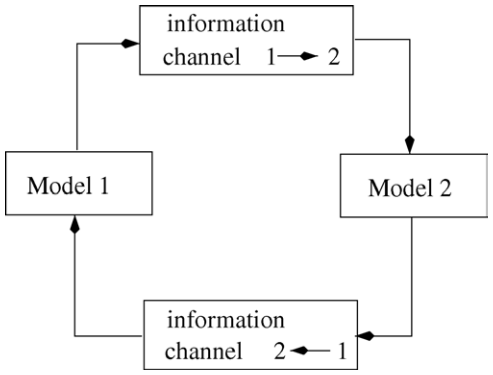

Fig. 2. Information channel in charge of the commutation process.

We look first at the process model problem.

Define $\Lambda = \{1, 2, \dots, m\}$ as a set containing indices of all required operating modes. $Card(\Lambda)$ represents the number of process models to be designed. In the case of two operating modes, $\Lambda = \{1, 2\}$ and $Card(\Lambda) = 2$.

An operating mode fixes the set of the components required to perform the task. These components make up the process in a given mode and, in an SCT context, the process is modelled by a model automaton. Commutation between modes tacke place when a particular event called commutation events occurs. If we consider just one structure such that there is always a subset of common components between two modes, the behavior of this subset must be tracked to define correctly the states from which an operation must be taken. A tracking mechanism is therefore introduced, ensured by information channels inserted between processe models. The role of these channels is to record all event sequences generated by the activated process until an commutation event occurs (see Fig. 2).

Let $i \in \Lambda$. We define G_i as an uncontrolled DES process taken to be an automaton of mode i. Formally:

$$G_i = (Q_i, \Sigma_i, \delta_i, q_{0,i}, Q_{m,i})$$

We suppose that $\Sigma_i \cap \Sigma_j \neq \emptyset$, i.e., we assume that common components can be found between two modes i and j.

Definition 3.1. let $\Sigma' = \cup_{ij}\{\alpha_{ij} \,|(i \neq j)\}$, the set of commutation events. The commutation event α_{ij} change the operating mode of the system from mode i to mode j,

- we assume that $\forall i \in \Lambda$, $\Sigma' \cap \Sigma_i = \emptyset$,
- in the case of two operating modes, the set Σ' contains two events α_{12} and α_{21}.

For simplicity, we consider the case of two operating modes. Initially, we assume that the process is engaged in mode 1. Thus, the system model is G_1 and all other models $(G_i, i \neq 1)$ are desactivated. When commutation event α_{12} occurs, the process model becomes G_2. However, in this case, we must correctly determine the starting state of G_2 after commutation from G_1. To do this, we first extend G_1 and G_2 by adding respectively an inactive state $q_{\mathrm{in},1}$ to the state set of the model G_1 and an inactive state $q_{\mathrm{in},2}$ to the state set of the model G_2 based on term significative state suggested by Dangoumau [4]. Occurrence of commutation event α_{12} will lead model G_1 to inactive state $q_{\mathrm{in},2}$ and the process model G_2 will be activated from $q_{\mathrm{in},2}$. The activated process model at a given time is thus the only model for which the current state is different to the inactive state.

So for model G_i, the extended model is defined as follows:

$$G_{i,\mathrm{ext}} = (Q_{i,\mathrm{ext}}, \Sigma_{i,\mathrm{ext}}, \delta_{i,\mathrm{ext}}, q_{0,i,\mathrm{ext}}, Q_{\mathrm{m},i,\mathrm{ext}}) \quad \text{with :}$$

(1) $Q_{i,\mathrm{ext}} = Q_i \cup \{q_{\mathrm{in},i}\}$: extended set states with an inactive state,
(2) $\Sigma_{i,\mathrm{ext}} = \Sigma_i \cup \Sigma'$: extended alphabet with the set of commutation events,
(3) $q_{0,i,\mathrm{ext}} = \begin{cases} q_{0,i} & \text{if } i = 1 : \text{we assume initially that model process } G_1 \text{ is in its initial state} \\ q_{\mathrm{in},i} & \text{if } i \neq 1 : \text{meaning that other models } G_i (i \neq 1) \text{ are assumed desactivated} \end{cases}$
(4) $Q_{\mathrm{m}}, i, \mathrm{ext} = Q_{\mathrm{m},i}$: marked state which equal to $Q_{\mathrm{m},i}$ because $q_{\mathrm{in},i}$ will never be marked state,

(5) the extended transition function is defined as follows:
- $\forall q \in Q_i$ and $\forall \sigma \in \Sigma_i$, if $\delta_i(q, \sigma)!$, then $\delta_{i,\text{ext}}(q, \sigma) := \delta_i(q, \sigma)$: extended transition function is the same as the transition function if we consider only non extended alphabet Σ_i,
- $\forall q \in Q_i$ from which commutation event $\alpha_{ij} \in \Sigma'$ ($i \neq j$, and $i, j \in \{1, 2\}$) can occurs, then $\delta_{i,\text{ext}}(q, \alpha_{ij}) := q_{\text{in},i}$: extended transition function allows model G_i to be desactivated if the commutation event occurs.

With regard to the process, the main aim of operating mode management is to define the starting state of model G_2 ($\delta_{2,\text{ext}}(q_{\text{in},2}, \alpha_{12})$) and, in turn, the return state of process model G_1 ($\delta_{1,\text{ext}}(q_{\text{in},1}, \alpha_{21})$).

We note that $G_{2,\text{ext}}$ is initially in inactive state $q_{\text{in},2}$, but, at the occurrence of the commutation event α_{12}, $G_{2,\text{ext}}$ must leave $q_{\text{in},2}$ to reach state $q \in Q_2$. Channel information introduced in Fig. 2 is materialized by the projection function π_{ij} ($i \neq j$) defined as follows [5]:

Definition 3.2.

$$\pi_{ij} : \Sigma_i^* \longrightarrow \Sigma_j^* \quad \text{such that} \quad \forall \sigma \in \Sigma_i \quad \text{and} \quad \forall s \in \Sigma_i^* :$$

$$\pi_{ij}(\varepsilon) = \varepsilon$$

$$\pi_{ij}(s\sigma) = \begin{cases} \pi_{ij}(s)\sigma & \text{if } \sigma \in \Sigma_i \cap \Sigma_j \\ \pi_{ij}(s) & \text{if } \sigma \in \Sigma_i \setminus \Sigma_j \end{cases}$$

The projection function not constrained the two alphabets Σ_i and Σ_j. In the particular case where $\Sigma_j \subseteq \Sigma_i$, this function corresponds to the natural projection classically used in SCT [7,13]. The actions of π_{ij} on a string s is just to "erase" all occurrences of σ in s such that $\sigma \notin \Sigma_i \cap \Sigma_j$. The projection function π_{ij} allows to track only the behavior of the common components between the two operating modes i and j.

To track the process behavior in mode 1, we use projection π_{12}, which is the mapping from Σ_1^* to Σ_2^*. Projection π_{12} identifies in G_2 the output states of intersection elements of G_1, when α_{12} occurs.

The following Proposition gives formally the state form which the model G_2 will be activated (the starting state), i.e., the determination of the transition $\delta_{2,\text{ext}}(q_{\text{in},2}, \alpha_{12})$.

Proposition 3.3. *Denote by follow(s) the set of events which follow the sequence of events s.* $\forall s \in L(G_1)$, *such that* $\alpha_{12} \in follow(s)$, *the starting state of the model* G_2 *is given by*: $\delta_{2,\text{ext}}(q_{\text{in},2}, \alpha_{12}) = \delta_2(q_{0,2}, \pi_{12}(s))$

Proof. Let $s \in L(G_1)$, such that $\alpha_{12} \in follow(s)$. So $s = \sigma_1, \sigma_2, \sigma_3, \sigma_4 \ldots \sigma_n$ is the sequence events generated ower alphabet Σ_1. However, Σ_1, as shown in Fig. 3, can be decompsed in two disjoint sets: $\Sigma_1 = (\Sigma_1 \setminus \Sigma_2) \cup (\Sigma_1 \cap \Sigma_2)$.[1] According to this partition two cases are possible:

Case 1 ($\pi_{12}(s) = \varepsilon$). This implies that $\forall \sigma \in s$, $\sigma \notin \Sigma_1 \cap \Sigma_2$. In this case no common components has evolued. On the other hand, after the occurrence of the sequence of events s, all common components of models G_1 and G_2 remain in their initial state. Thus, the occurrence of commutation event α_{12} lead $G_{2,\text{ext}}$ from the inactive state $q_{\text{in},2}$ to the initial state of G_2. This state is the Cartesian product of all the initial states of each component constituting the model G_2. So

$$\delta_{2,\text{ext}}(q_{\text{in},2}, \alpha_{12}) = \delta_2(q_{0,2}, \pi_{12}(s)) = \delta_2(q_{0,2}, \varepsilon) = q_{0,2}$$

Case 2 ($\pi_{12}(s) \neq \varepsilon$). This implies that there exists k events σ_i such that $i \in \{1, \ldots, n\}$ and $\sigma_i \in \Sigma_1 \cap \Sigma_2$. Then $\pi_{12}(s) = s_i$ with s_i the ordered sequence of events σ_i. Thus we are in the situation where at least one of the common components i changed state. Then the reachable state in the model G_2 is not necessary its initial state. Hence the reached state in the model G_2 will not have to be its initial state. It will be determined by:

$$\delta_{2,\text{ext}}(q_{in,2}, \alpha_{21}) = \delta_2(q_{0,2}, \pi_{12}(s)) = \delta_2(q_{0,2}, s_i). \qquad \square$$

Let us now assume the G_1 is desactivated, i.e., is in the inactive state $q_{in,1}$ and G_2 is activated, i.e., G_2 is in state $q \in Q_2$ ($q \neq q_{in,2}$). If α_{21} occurs, G_2 will be desactivated, but G_1 will leave $q_{in,1}$ to reach state $q \in Q_1$. As before, we must now define the return state $\delta_{1,\text{ext}}(q_{in,1}, \alpha_{21})$.

[1] $\Sigma_1 \setminus \Sigma_2 = \{\sigma \in \Sigma_1 | \sigma \notin \Sigma_2\}$.

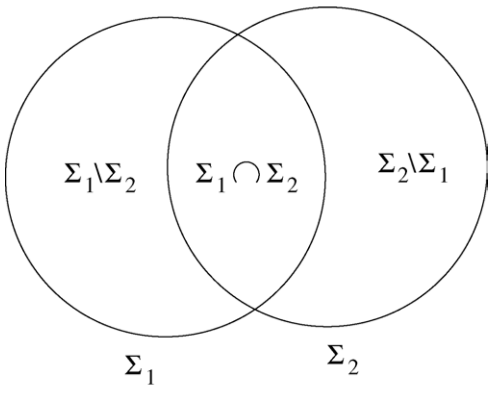
$\Sigma_1 \backslash \Sigma_2$
$\Sigma_1 \cap \Sigma_2$
$\Sigma_2 \backslash \Sigma_1$
Σ_1
Σ_2

Fig. 3. Partition of Σ_1.

Reciprocally, we introduce $\pi_{21} : (\Sigma_2)^* \longrightarrow (\Sigma_1)^*$ which has the similar definition as π_{12}.

The following Proposition provides the formal framework for the determination of this recovering state (state from which the model G_1 will be activated).

Proposition 3.4. *$\forall s \in L(G_1)$ such that $\alpha_{12} \in follow(s)$, and $\forall s' \in L(G_2, \delta_2(q_{0,2}, \pi_{12}(s)))$[2] such that $\alpha_{21} \in folllow(s')$, the recovering state of the model G_1 is given by*:

$$\delta_{1,\mathrm{ext}}(q_{in,1}, \alpha_{21}) = \delta_1(q_{0,1}, \pi_{12}(s)\pi_{21}(s'))$$

Proof. According to whether $\pi_{12}(s)$ and $\pi_{21}(s')$ are equal to ε or not, and according to the evolution or not of common components, four cases are to be studied.

Case 1 ($\pi_{12}(s) = \varepsilon$ and $\pi_{21}(s') = \varepsilon$). None the common components between G_1 and G_2 did not evolve, which imply that $\forall \sigma \in s, \sigma \in \Sigma_1$ and $\sigma \notin \Sigma_2$ then $\delta_2(q_{0,2}, \pi_{12}(s)) = \delta_2(q_{0,2}, \varepsilon) = q_{0,2}$.

The model G_2 admits its own intial state $q_{0,2}$ as the starting state after the occurrence of the commutation event α_{12}. Similarly for the recovering state, since $\delta_1(q_{0,1}, \pi_{12}(s)\pi_{21}(s')) = \delta_1(q_{0,1}, \varepsilon) = q_{0,1}$.

Thus, the model G_1 admits as recovering state:

$$\delta_{1,\mathrm{ext}}(q_{\mathrm{in},1},\ \alpha_{21}) = \delta_1(q_{0,1}, \pi_{12}(s)\pi_{21}(s')) = \delta_1(q_{0,1},\ \varepsilon) = q_{0,1}$$

Case 2 ($\pi_{12}(s) = \varepsilon$ and $\pi_{21}(s') \neq \varepsilon$). In the mode 1 no common component evolved, but in the mode 2, at least a common component between G_1 and G_2 evolved.

$\pi_{12}(s) = \varepsilon$ implies that $\delta_2(q_{0,2}, \pi_{12}(s)) = \delta_2(q_{0,2}, \varepsilon) = q_{0,2}$.

However, if $\pi_{21}(s') \neq \varepsilon$, there exists $\sigma' \in s'$ such that $\sigma' \in \Sigma_1 \cap \Sigma_2$. So the recovering state in the model G_1 can be different of the initial state $q_{0,1}$. This state (recovering state) is determined by the set of common events in the two alphabets Σ_1 and Σ_2 in s'. Thus, we can write:

$$\delta_{1,\mathrm{ext}}(q_{in,1}, \alpha_{21}) = \delta_1(q_{0,1}, \pi_{12}(s)\pi_{21}(s')) = \delta_1(q_{0,1}, \pi_{21}(s'))$$

Case 3 ($\pi_{12}(s) \neq \varepsilon$ and $\pi_{21}(s') = \varepsilon$). In the mode 1 at least one common components is evolved. But no common components between G_1 and G_2 did not evolve in the mode 2.

$\pi_{12}(s) \neq \varepsilon$ implies that there exists $\sigma \in s$ such that $\sigma \in \Sigma_1 \cap \Sigma_2$. So at least one common component changed state before the occurrence of the commutation event α_{12}. Thus, the starting state of the model G_2 is not necesseraly $q_{0,2}$ but a state given by $\delta_2(q_{0,2}, \pi_{12}(s))$. By taking into account that $\pi_{21}(s') = \varepsilon$, this implies that the recovering state in the model G_1 is given following the occurrence of an commutation event α_{21} as follows:

$$\delta_{1,\mathrm{ext}}(q_{\mathrm{in},1}, \alpha_{21}) = \delta_1(q_{0,1}, \pi_{12}(s)\pi_{21}(s')) = \delta_1(q_{0,1},\ \pi_{12}(s))$$

[2] $L(G_2, \delta_2(q_{0,2}, \pi_{12}(s))) = \{u \in (\Sigma_2)^* | \delta_2(\delta_2(q_{0,2}, \pi_{12}(s)), u)!\}$

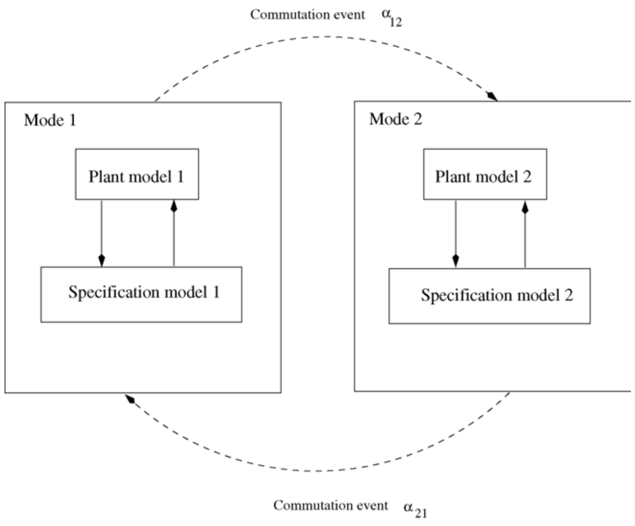

Fig. 4. Multi-model structure.

Case 4 ($\pi_{12}(s) \neq \varepsilon$ and $\pi_{21}(s') \neq \varepsilon$). At least one common component has evolved in the two modes 1 and 2. By adopting the previous reasoning, it follows that:

$$\delta_{1,\text{ext}}(q_{\text{in},1}, \alpha_{21}) = \delta_1(q_{0,1}, \pi_{12}(s)\pi_{21}(s')) \qquad \Box$$

Remark 1. Since each process model has a unique inactive state, we have a nondeterministic problem. Indeed, from an inactive state $q_{\text{in},j}$, several states can be reached for the same commutation event. To overcome this problem, we define a set of events allowing occurrences of commutation event $\alpha_{ij} : \alpha_{ij} = \alpha_{ijk}$ if $\delta_{j,\text{ext}}(q_{0,j}, \pi_{ij}(s)) = q_{k,j}$ to be distinguished in model G_j.

4. Commutation specification models

In Section 3, we studied the switching mechanism between different processes models. We extended these models to determine their compatible states of connection. In this Section, we etablish the switching mechanism between specification models by researching the states from which these models must be activated, while ensuring adequacy between current process dynamics and control decisions. Each process model is associated with a specification model (Fig. 4), so a change of operating mode may allow different dynamics for the selected (activated) process model. In this case, the associated specification model must be activated from a state that allows suitable control decisions to be made with respect to the new process model dynamics.

As seen above for a process model (Section 3), we assume that one specification model is activated for a given operating mode.

Commutation event α_{ij} leads specification model S_i to inactive state $x_{\text{in},i}$ but specification model S_j must leave inactive state $x_{\text{in},j}$ to reach compatible state $x \in X_j$ with the new dynamics for process G_j.

If we assume that, after commutation from mode i to mode j, the starting state of process model G_j is q, so the new dynamics for process model G_j is $L(G_j, q) := \{s \in \Sigma_j^* | \delta_j(q, s)!\}$. State $x \in X_j$ is called compatible for the new process model G_j dynamics ($L(G_j, q)$) if the intersection specification language from state $x \in X_j (L(S_j, x) := \{s \in \Sigma_j^* | \xi_j(x, s)!\})$ and the generated language $L(G_j, q)$ equals the corresponding desired language ($L(S_j, x) \cap L(G_j, q) = K_{j,q}$, where $K_{j,q}$ is the desired language from state q).

Formally, for specification model $S_i = (X_i, \Sigma_i, \xi_i, x_{0,i}, X_{\text{m},i})$, we define the extended specification model $S_{i,\text{ext}} = (X_{i,\text{ext}}, \Sigma_{i,\text{ext}}, \xi_{i,\text{ext}}, x_{0,i,\text{ext}}, X_{\text{m},i,\text{ext}})$ as follows:

(1) $X_{i,\text{ext}} = X_i \cup \{x_{\text{in},i}\}$.

(2) $\Sigma_{i,\text{ext}} = \Sigma_i \cup \Sigma'$.

(3) $x_{0,i,\text{ext}} = \begin{cases} x_{0,i} & \text{if } i = 1 \\ x_{\text{in},i} & \text{if } i \neq 1 \end{cases}$

(4) the extended transition function $\xi_{i,\text{ext}}$ is defined as follows:
- $\forall x \in X_i$ and $\forall \sigma \in \Sigma_i$, if $\xi_i(x, \sigma)$ exists, then $\xi_{i,\text{ext}}(x, \sigma) := \xi(x, \sigma)$,
- $\forall x \in X_i$ from which α_{ij} ($i \neq j$ and $i, j \in \{1, 2\}$) can occur, then $\xi_{i,\text{ext}}(x, \alpha_{ij}) := x_{\text{in},i}$.

The above extended specification model is incomplete.

Transition function $\xi_{i,\text{ext}}(x_{\text{in},i}, \alpha_{ij})$ (for $i, j \in \{1, 2\}$ and $i \neq j$) reamins to be defined.

The main problem in this section is the determination of an adequate starting state $\xi_{i,\text{ext}}(x_{\text{in},i}, \alpha_{ji})$ that allows correct control decisions to be made with the new process dynamics to achieve the desirable behavior associated with the new operating mode.

There are common components to both modes, so commutation from mode i to mode j is performed somewhere along the paths to state $(q, x) \in Q_i \times X_i$. The starting state $x \in X_j$ depends on event sequences $s \in L(S_i/G_i)$ checking $\mathit{follow}(s) = \alpha_{ij}$.

We therefore need to identify paths $s \in L(S_i/G_i)$, along which commutation event α_{ij} should occur.

First, we identify all pairs of states $(q, x) \in Q_i \times X_i$, where for $s \in L(S_i/G_i)$ and $\alpha_{ij} \in \mathit{follow}(s)$, such that $\delta_i \times \xi_i((q_{0,i}, x_{0,i}), s) = (q, x)$.

Let Q'_i such set state, and let K_{i,Q'_i} be its describing language such that $K_{i,Q'_i} := \{s \in L(S_i/G_i) \mid \delta_i \times \xi_i((q_{0,i}, x_{0,i}), s) \in Q'_i\}$. It is easy to show that the language K_{i,Q'_i} is partitioned into a set of languages $K_{i,q}$, where $K_{i,q}$ contains all event sequences which lead to a state $q \in Q'_i$, i.e., $K_{i,Q'} = \cup_{q \in Q'_i} K_{i,q}$, where $K_{i,q} := \{s \in L(S_i/G_i) \mid \delta_i \times \xi_i((q_{0,i}, x_{0,i}), s) = q\}$.

If we assume that $\forall s \in K_{i,q}$, there is one desired language from the starting state $\delta_{j,\text{ext}}(q_{0,j}, \pi_{ij}(s))$.

When commutation event α_{ij} occurs, we assume that the starting state of model G_j is q'_j and this state is given by Proposition 3.3. There is therefore a unique state $q \in Q'_i$ and an event sequence s in $K_{i,q}$ such that $\delta_i \times \xi_i((q_{0,i} \times \xi_{0,i}), s) = q$ and $\delta_j(q_{0,j}, \pi_{ij}(s)) = q'_j$. The desired language from starting state q'_j is built up according to event sequence s. Let $K_{j,q}$ be such a language.

The following Proposition expresses the starting state of specification model S_j compatible with state q'_j.

Proposition 4.1. *The starting state of S_j is given by the solution to the following problem: find a unique x_k such that: $L(G_j, q'_j) \cap L(S_j, x_k) = K_{j,q'_j}$ where K_{j,q'_j} is the desired language from q'_j.*

Proof. suppose there are two states x_k and $x_{k'}$, such that $x_k \neq x_{k'}$ and

$$L(G_j, q'_j) \cap L(S_j, x_k) = K_{j,q'_j}. \tag{1}$$

$$L(G_j, q'_j) \cap L(S_j, x_{k'}) = K_{j,q'_j}. \tag{2}$$

Eqs. (1) and (2) mean that $\forall s \in K_{j,q'_j}$:

$$\delta_j \times \xi_j((q'_j, x_k), s) = \delta_j \times \xi_j((q'_j, x_{k'}), s).$$

So $(\delta_j(q'_j, s), \xi_j(x_k, s)) = (\delta_j(q'_j, s), \xi_j(x_{k'}, s))$. We then get $\forall s \in \Sigma_j^*$.

$$\xi_j(x_k, s) = \xi_j(x_{k'}, s).$$

For $s = \epsilon$, we have $x_k = x_{k'}$, contradicting the fact that $x_k \neq x_{k'}$. □

We now assume that specification model S_j is activated, i.e., the assiciated model is in state $x \in X_j$ and S_i is desactivated, i.e., is in an inactive state $x_{\text{in},i}$. If the commutation event α_{ji} occurs S_j will be desactivated but S_i will leave $x_{\text{in},i}$ to enter a state $x \in X_i$. We must, as before, define the return state of specification model S_i and a similar method can be used.

We assume that the return state of model automaton G_i is q_i so the return state of specification model S_i is given by the solution to the following problem: find an unique x_l such that $L(G_i, q_i) \cap L(S_i, x_l) = K_{i,q_i}$, where K_{i,q_i} is the desired language from q_i.

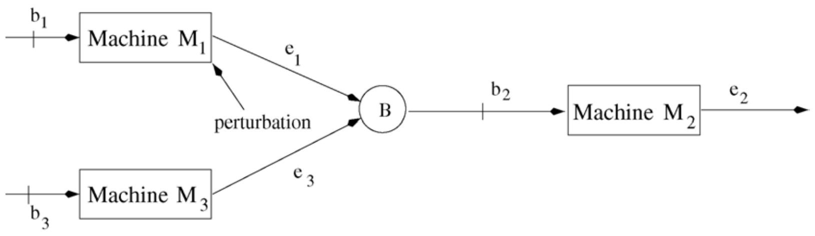

Fig. 5. Diagram of production unit example.

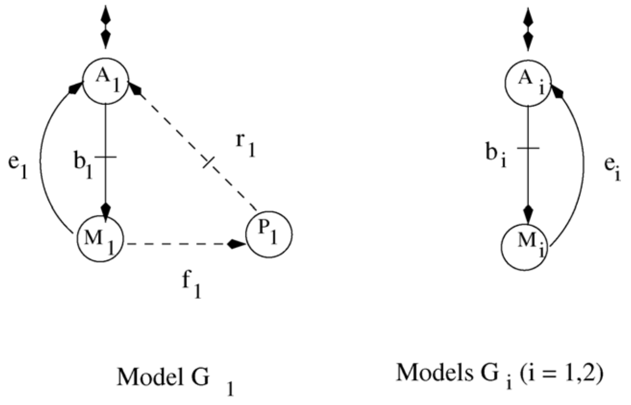

Fig. 6. Automata models of machines M_i (for $i \in \{1, 2, 3\}$).

5. Illustrative example

The proposed approach is illustrated by means of a production example. This system features three machines, as shown in Fig. 5.

Initially, buffer B is empty and machine M_3 is performing another task outside the unit, but it intervenes when M_1 breaks down. With event b_1 (respectively b_3), M_1 (repectively M_3) picks up a workpiece from an infinite bin and places it in buffer B after completing its work (event e_1, repectively e_3). M_2 operates similarly, but takes its workpiece from B (event b_2) and places it in an infinite output bin when it has finished its task (event e_2). It is assumed that only M_1 can break down (event f_1) and be repaired (event r_1) (as shown in Fig. 6). Two operating modes are designed for the overall system: a nominal mode (G_{n}), in which M_1 and M_2 produce and a degraded mode (G_{d}), in which M_3 replaces M_1. These two modes are built up from models of M_1, M_2, and M_3, but they exclude f_1 and r_1 events, which are considered as commutation events between modes $\Sigma' = \{f_1, r_1\}$.

Initially, the system runs in the nominal mode described by the model G_{n}. When f_1 occurs, the system switches to the degraded mode described by the model G_{d}. Occurrence of r_1 allows G_{d} to switch to G_{n} (Fig. 7). This means that only one operating mode is activated at one time. M_2 is considered as the common component and will be associated with this component thereby tracking the process and the specification.

In the example, the set of events $\Sigma_{\mathrm{global}} = \{b_1, b_2, b_3, e_1, e_1, e_3, f_1, r_1\}$ can be partitionned into 3 sets: $\Sigma_{\mathrm{n}} = \{b_1, b_2, e_1, e_2\}$, the nominal mode set, $\Sigma' = \{f_1, r_1\}$ commutation event set and $\Sigma_{\mathrm{d}} = \{b_3, e_3, b_2, e_2\}$ degraded mode set.

The commutation event (failure event f_1) can occur from state $q_{1,\mathrm{n}}$ and $q_{2,\mathrm{n}}$. We can show that $\forall s \in L(G_{1,\mathrm{n}})$, such that $f_1 \in follow(s)$ and $\delta_{\mathrm{n}}(q_{0,\mathrm{n}}, s) = q_{1,\mathrm{n}}$ (respectively $\delta_{\mathrm{n}}(q_{0,\mathrm{n}}, s) = q_{1,\mathrm{n}}$) and find that $\pi_{\mathrm{nd}}(s) = (b_2e_2)^{\mathrm{n}}$ (where $\mathrm{n} \in \mathbb{N}^*$[3] and $(b_2e_2)^0 = \epsilon$) (respectively $\pi_{\mathrm{nd}}(s) = (b_2e_2)^{\mathrm{n}}b_2$). In this case therefore, the adequate starting state (Proposition 3.3) of degraded model is $\delta_{\mathrm{d,ext}}(q_{\mathrm{in,d}}, f_1) = \delta_{\mathrm{d}}(q_{0,\mathrm{d}}, \pi_{\mathrm{nd}}(s)) = \delta_{\mathrm{d}}(q_{0,\mathrm{d}}, (b_2e_2)^{\mathrm{n}}) = q_{0,\mathrm{d}}$ (respectively $\delta_{\mathrm{d,ext}}(q_{\mathrm{in,d}}, f_1) = \delta_{\mathrm{d}}(q_{0,\mathrm{d}}, \pi_{\mathrm{nd}}(s)) = \delta_{\mathrm{d}}(q_{0,\mathrm{d}}, (b_2e_2)^{\mathrm{n}}b_2) = q_{2,\mathrm{d}}$).

[3] $\mathbb{N}^*$ is the set of positive integers: 1,2,3,. . . .

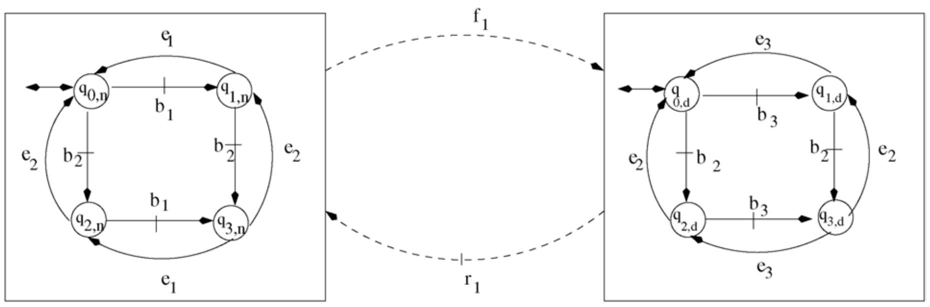

Fig. 7. Nominal and degraded process model.

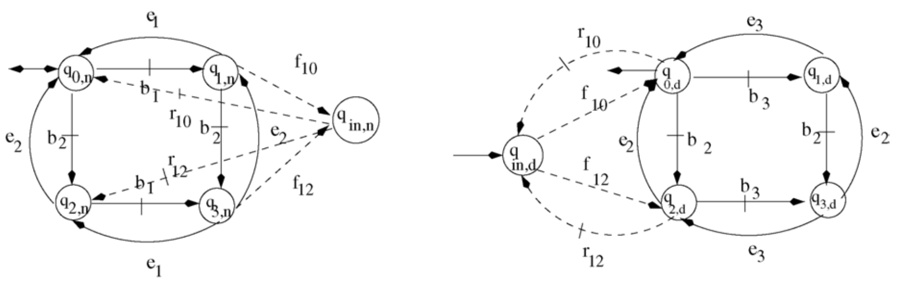

Fig. 8. Extended nominal and degraded porcess model.

Similarly, and by applying Proposition 3.4, the return states of the nominal model, when commutation event r_1 occurs, are $q_{0,\mathrm{n}}$ and $q_{2,\mathrm{n}}$. The extended nominal and degraded process models are shown in Fig. 8.

The nominal process model specification is such that: the buffer must not overflow to 1 nor underflow to 0 (see Fig. 9).

Similary, the degraded process model specification is: the buffer must not overflow to 1 nor underflow to 0 (see Fig. 10).

The initial state of the nominal mode specification is $x_{0,\mathrm{n}}$. However, since the degraded process model has two starting states $q_{0,\mathrm{d}}$ and $q_{2,\mathrm{d}}$, i.e., two different dynamics ($L(G_{\mathrm{d}}, q_{0,\mathrm{d}})$ and $L(G_{\mathrm{d}}, q_{2,\mathrm{d}})$), the degraded mode specification model can be also activated from a state possibly different to the initial state $x_{0,\mathrm{d}}$.

When commutation event f_1 occurs, the nominal specification model will be desactivated but the degraded specification will be activated and conversely when event r_1 occurs.

The extended nominal and the degraded specification models are represented in Figs. 11 and 12, respectively.

The controlled process in the nominal mode is shown in Fig. 13.

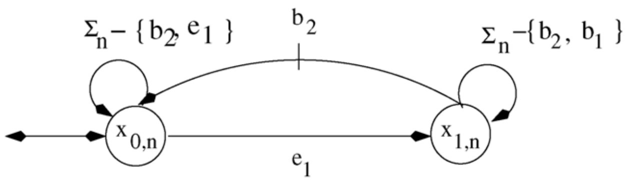

Fig. 9. Nominal mode specification model.

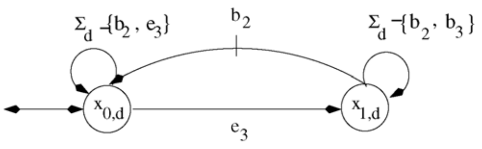

Fig. 10. Degraded mode specification model.

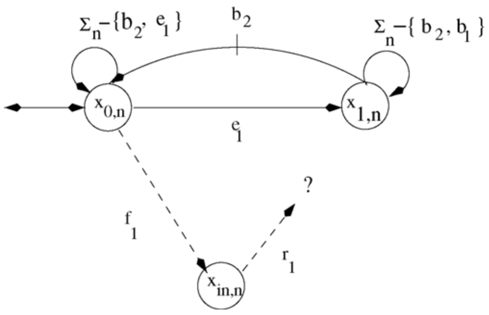

Fig. 11. Extended nominal mode specification.

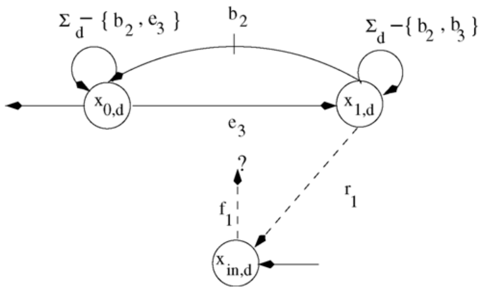

Fig. 12. Extended specification of the degraded mode.

We are especially intersted in states $q_{1,\mathrm{n}}$ and $q_{4,\mathrm{n}}$ of the controlled process $S_{\mathrm{n}}/G_{\mathrm{n}}$, in which commutation event f_1 can occur. We then want to find languages $K_{\mathrm{n},q_{1,\mathrm{n}}}$ and $K_{\mathrm{n},q_{4,\mathrm{n}}}$ in relation to states $q_{1,\mathrm{n}}$ and $q_{4,\mathrm{n}}$ respectively, i.e., $K_{\mathrm{n},q_{1,\mathrm{n}}} = \{s \in L(S_{\mathrm{n}}/G_{\mathrm{n}}) | \delta_{\mathrm{n}} \times \xi_{\mathrm{n}}((q_{0,\mathrm{n}}, x_{0,\mathrm{n}}), s) = q_{1,\mathrm{n}}\}$ and $K_{\mathrm{n},q_{4,\mathrm{n}}} = \{s \in L(S_{\mathrm{n}}/G_{\mathrm{n}}) | \delta_{\mathrm{n}} \times \xi_{\mathrm{n}}((q_{0,\mathrm{n}}, x_{0,\mathrm{n}}), s) = q_{4,\mathrm{n}}\}$. We assume that the commutation event occurs in a state $q_{1,\mathrm{n}}$.

So $\forall s \in K_{\mathrm{n},q_{1,\mathrm{n}}}$, the state buffer is empty ($x_{0,\mathrm{n}}$), so the desired degraded model langage in at state $q_{0,\mathrm{d}}$ is $K_{\mathrm{d},q_{0,\mathrm{d}}} = \{b_3,\ b_3e_3,\ b_3e_3b_2,\ \ldots\}$. The starting state of specification model S_{d} permitting the desired language $K_{\mathrm{d},q_{0,\mathrm{d}}}$ to be reached is given by the solution of the following equation:

$$L(G_{\mathrm{d},q_{0,\mathrm{d}}}) \cap L(S_{s,x_l}) = K_{\mathrm{d},q_{0,\mathrm{d}}}.$$

If we assume that commutation event f_1 can occur from state $q_{4,\mathrm{n}}$ as before $\forall s \in K_{\mathrm{n},q_{4,\mathrm{n}}}$ and the output state of the buffer is empty. Thus, the desired degraded model language at state $q_{2,\mathrm{d}}$ becomes $K_{\mathrm{d},q_{2,\mathrm{d}}} = \{b_3, e_2,\ b_3e_3, e_2, b_3e_3b_2, \ldots\}$.

As a result, the starting state of the specification model S_{d} in this case is given by the solution of the following equation:

$$L(G_{\mathrm{d},q_{2,\mathrm{d}}}) \cap L(S_{s,x_l}) = K_{\mathrm{d},q_{0,\mathrm{d}}}.$$

State x_l checking the last equation is $x_{0,\mathrm{d}}$.

The extended degraded model is shown in Fig. 14.

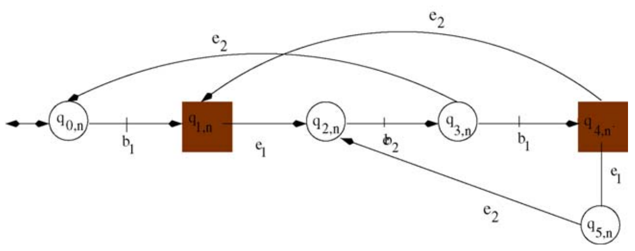

Fig. 13. The controlled process in the nominal operating mode.

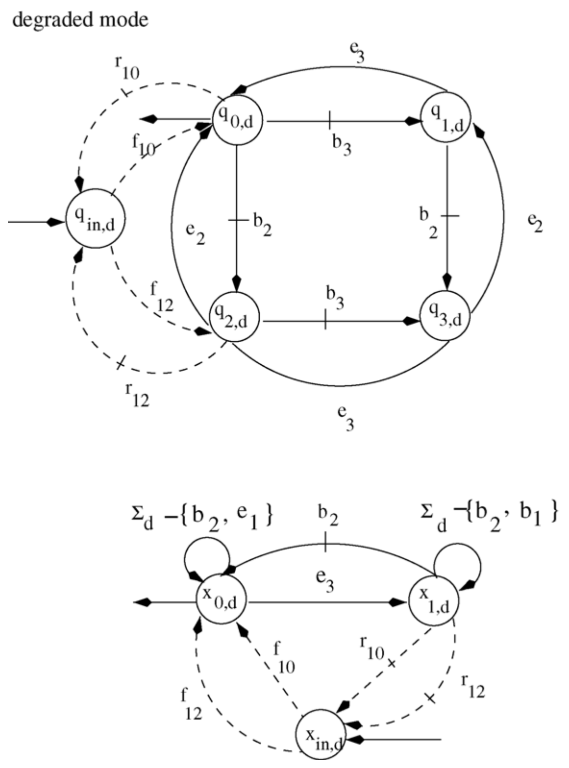

Fig. 14. Extended degraded process and the corresponding specification model.

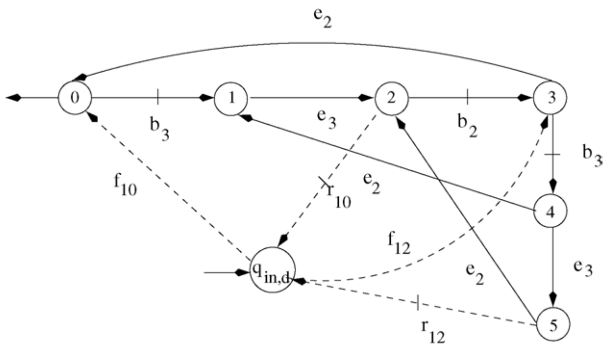

Fig. 15. Extended degraded controller.

Using “TCT”,[4] the extended degraded controller is shown in Fig. 15.

We can find the return state of the nominal specification model S_n in the same way. Finally, the extended nominal process and specification model are shown in Fig. 16.

Using “TCT”, the extended nominal controller is shown in Fig. 17.

6. Conclusion

The proposed approach allows commutation between different models of a global system reacting to exceptional situations such as failure event occurrence. The major contribution of this paper considers reactive systems with different

[4] “TCT” is a tool allowing SED simulation and is a program for the synthesis of supervisory controls for untimed discrete-event systems.

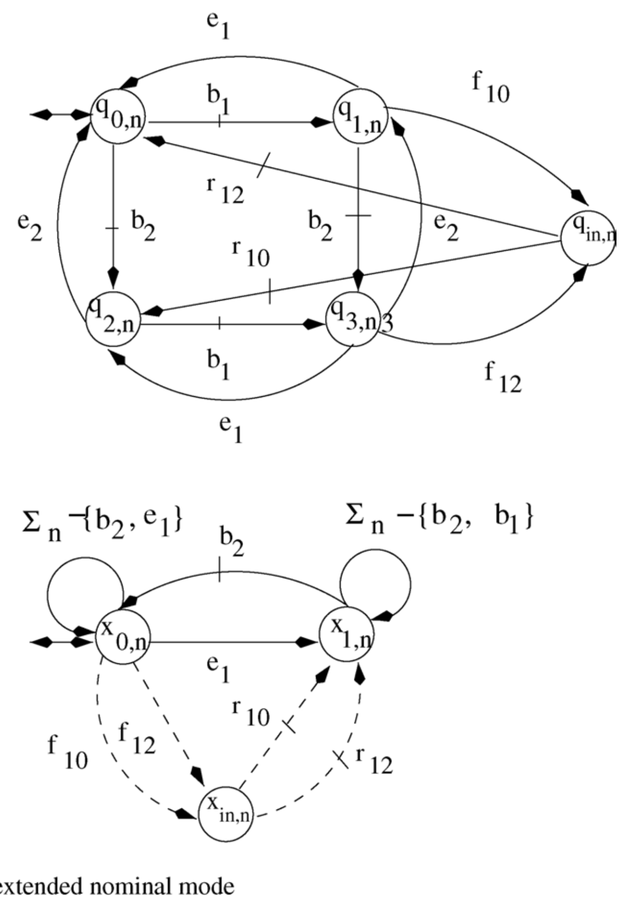

Fig. 16. Extended nominal process and corresponding specification model.

objectives. Each objective (i.e., operating mode) is represented by a set comprising a model process and a specification. Assuming that different models evolve independently, the main problem is then to deactivate model G_i and commute to a model G_j as soon specification one. G_j will be considered as the process model until an exceptional event occurs. A formal framework based on tracking events is proposed to ensure commutation. This framework introduces a new definition of the projection function.

Propositions 3.3, 3.4, and 4.1, represent the principal results of this paper. They define formally the starting and return states of a new activated process model in a new structure system after commutation.

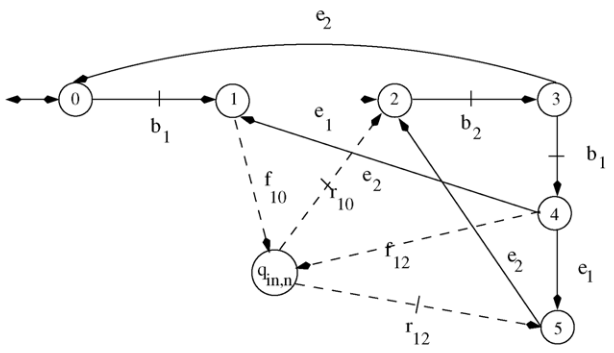

Fig. 17. Extended nominal controller.

References

[1] G.C. Cassandras, S. Lafortune, Introduction to Discrete Event Systems, first ed., Kluwer Academic Publishers, Boston, 1999, pp. 822.

[2] P. Charbonnaud, F. Rotella, S. Médar, Process Operating mode monitoring process: Switching online the right controller, IEEE Transact. Syst. Man Cybernetics, Part C 31 (1) (2002) 77–86.

[3] S. Chafik. Proposition d’une structure de contrôle par supervision hiérarchique et distribuée: Application à la coordination, Ph.D. thesis, Laboratoire d’Automatique Industrielle, INSA de Lyon, 2000.

[4] N. Dangoumau. Contribution à la gestion des modes des systèmes automatisés de production. Thèse de Doctorat: Université des Sciences et Technologiques de Lille, France, 2000, p 181.

[5] O. Kamach, S. Chafik, L. Pietrac and E. Niel representation of a reactive systems with different models, IEEE SMC, Hammamet, Tunisia, reference TA2L4 in CDROM, 6–9 octobre 2002.

[6] F. Lin, W. Wonham, Decentralised supervisory control of discrete event systems, Inf. Sci. 25 (5) (1987) 1202–1218.

[7] F. Lin, W. Wonham, On observability of discrete event systems, Inf. sci. 44 (2) (1988) 173–198.

[8] F. Lin, W. Wonham, Decentralised control and coordination of discrete-event systems with partial observation, IEEE Transact. Automatic Control 35 (12) (1990) 1330–1337.

[9] P. Ramadge, W. Wonham, Supervisory control of class of discrete event processes, SIAM J. Control Optimisation 25 (1) (1987) 206–230.

[10] P. Ramadge, W. Wonham, Control of discrete event systems, IEEE Transact. Automatic Control 77 (1) (1989) 81–98.

[11] K. Rudie, S. Lafortune, F. Lin, Minimal communication in a distrubted discrete-event control system., in: Proceedings of the American Control Conference, 1999, pp. 1965–1970.

[12] W. M. Wonham. Notes on control of discrete-event systems, notes de cours, depertment of Electrical and Coputer Engineering, University of Toronto, http://www.control.toronto.edu/people/profs/wonham/, 2002.

[13] K.C. Wong, J.G. Thistle, R.P. Malhame, H.H. Hoang, Supervisory Control of distributed Systems: conflict resolution, Discrete Event Dynamics Syst. 10 (1988) 131–186.

[14] T. Yoo, S. Lafortune, New Results on decentralised supervisory control of discrete event systems, IEEE Conference on Decision and Control 2000, Sydney Australia, December 2000, pp. 1–6.

[15] M. Zefran, J. Burdick, Design of switching controllers for systems with changing dynamics, in: Proceedings of the 37th Conference on Decision and Control, 1998, pp. 2113–2118.