GENERALIZATION OF DES MULTI-MODELING

Oulaid KAMACH, Samir CHAFIK, Laurent PIETRAC and Eric NIEL

Laboratoire d'Automatique Industrielle
Institut National des Sciences Appliquées
Bat St Exupéry 25 av Jean Cappelle-69621 Villeurbanne CEDEX-France
Tel : (33) 4 72 43 62 14
Fax : (33) 4 72 43 85 35
Email: {name}@lai.insa-lyon.fr

Abstract: Abstract: DES multi-modeling appears to be well adapted to management of production system operating modes. Associating a specific model of the process to be controlled and its specifications is in fact natural. However, conceptual problems involving the control aspect may arise, when an admissible distinctive behavior set is specified without considering the ensuing complexity. The aim of this paper is to specify and validate formally operating mode management under generalized conditions. Basically, the paper extends the model commutation problem (process-limited) from one-to-one to one-to-all. Its main results concern the generalized tracking mechanism for a different process behavior combination.

Keywords: multi-models, reactive systems, operating modes, discrete event systems..

1. INTRODUCTION

Today, industrial system efficiency results in not only high productivity rates but also high reactivity performance. This means that, whilst a process is well controlled for a given requirement, an unwanted event must cause it to operate differently (products must continue depending on the system reactivity). When more that one unwanted event is considered, process multiple behavior can be accepted. We assume that the process remains unchanged in an operating mode, but process potentiality (structure and performance characteristics) changes drastically when an exceptional event occurs and this means that the original process has changed. Well structured reactivity will depend firstly on the organization of data emitted from the enterprise level to the execution plant and secondly on control adaptability. Operating mode management offers both an industrial and a scientific challenge in relation to this last point. The main problems encountered in this area are correct specification definition, exhaustive validation and modified process behavior management.
Generally, specification description needs to include nominal and exceptional admissible behaviors. If nominal behaviors are not unique and their definition is laborious (full power operation, downgraded operation, etc.), taking exceptional behaviors into account increases complexity. Reasons for an improperly defined specification set are probably lack of well adapted methodology (even when sectarian methods exist or depend on standards adaptation ISA 88[1]) or insufficient knowledge of legal commutation procedures.
Validation ensures correctness of all predefined requirements at design stage and will establish whether required operating modes are possible, well connected and sufficiently accessible. Thus, validation considers not only internal mode behavior, but also mode commutations, which must establish a set of conditions governing commutation, starting state and recovery state from one mode to another.
Partial contributions to solving this problem have been provided using empirical approaches (GEMMA[2]) but these are limited for small systems. Other contributions offer a more appropriate modeling aspect (Statechart, (HAREL, 1996))

[1] International standard for flexibility in production: www.s88.info
[2] Guide d'Etudes des Modes de Marche et d'Arrêt

involving conciseness, but they remain insufficient for proving properties essential to validation. Automata theory provides a more formal context and its extension will form the basis of our proposal described in the following sections.
Formal commutation has been solved and submitted in terms of model tracking from one mode to another. This paper attempts to generalize the process model commutation problem, when a distinctive operating mode combination is considered. Process model tracking mechanisms are studied especially with respect to starting state recognition, on the first hand, and recovery state recognition on the other hand. Information channels are used to ensure commutation enabling as described in (Lin and Wonham, 1988, Wong et al, 2000). To maintain presentation clarity, two distinct theorems will be introduced; the first associated with starting state search channel, the second with recovery state.
It should be recalled that only distinctive process models are considered.
The paper is structured as follows:
Section 2 presents preliminaries required for understanding the power of supervisory control theory. Section 3 defines the multi-model concept, adapted here solely to distinctive process behavior, and illustrates it using an example. Section 4 presents the information channel for solving the starting state search process and this is formally expressed by theorem 1. Recovery state will be presented in section 5 and is associated with theorem 2.
Finally, this paper provides conclusions and details research prospects.

2. PRELIMINARIES

This section introduces supervisory control theory (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 Discrete Event System (DES) and is designed by an automaton G. This automaton is an event generator so that $G = (Q, \Sigma, \delta, q_0, Q_m)$ with Q the set of states, Σ the set of event labels, and $\delta : Q \times \Sigma \to Q$ the partial transition function which is defined at each $q \in Q$ for a subset of events $\sigma \in \Sigma$. q_0 is the initial state while $Q_m \subseteq Q$ is the set of marker states which represent the end of tasks or final states of process. Σ^* contains all possible finite strings over Σ plus the empty string ε. The definition for δ can be extended to a partial function $\delta : Q \times \Sigma^* \to Q$ such that $\delta(q,\varepsilon) = q$ $(\forall\, q \in Q)$ and $\delta(q,s\sigma) = \delta(\delta(q,s),\sigma)$ with $\sigma \in \Sigma$ and $s \in \Sigma^*$.
The language generated by G is $(L(G) = \{s \in \Sigma^* \mid \delta(q_0,s)!\})$[3] and its marked language is $L_m(G) = \{s \in \Sigma^* \mid \delta(q_0,s) \in Q_m\}$. $L_m(G)$ can be calculated by Arden's lemma (Wonham, 2002), and this will be used in section 4.
Arden's lemma:

Let A and B, two regular[4] languages.

1- A^*B is always a solution of the equation $X = AX+B$

2- If $\varepsilon \notin A$, then A^*B is the unique solution of the equation $X = AX+B$.

The specification model E is also an automaton, and the controlled DES S/G is obtained by composition of G and E. S/G represents the evolution of the process G restricted by a supervisor S. For further explanation of theory principles, the reader is referred to (Wonham, 2002) or (Cassandras and Lafortune, 1999) (Rudie et al, 1999).
In most cases, the system can be broken down into numerous subsystems. Similarly, process and specification models are the combination of several simple models. Therefore, a current SCT application problem is the explosion in the number of states as the number of components increases. This explosion is often handled by performing horizontal (modular or decentralized) or vertical (hierarchical) break-down of the underlying control problem (Lin and Wonham, 1988, Yoo and Lafortune, 2002, Wong et al, 2000, Chafik and Niel, 2001).
In the other words, production systems must manufacture various productions and react rapidly to failures, if they are to be competitive. Different system use corresponds to different operating modes. Adjustment and maintenance modes are examples of other operating modes that are absolutely necessary for system use. However, a system does not require all components in each operating mode. 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 one process and because specifications must be in mutual conflict. Based on an example of two operating modes, (Kamach et al, 2002) present a 2-model approach, in which each process model uses different components of the global system and each operating mode corresponds to one model. In next section, we will extend this proposal to the general case of any number of operating modes. In this paper, we restrict ourselves to process models only.

3. DES MULTI-MODELING DESIGN

This section focuses on modeling operating modes by applying a multi-model concept, which involves designing a model process for each operating mode. The problem of commutation between all designed models is formalized by a proposed framework. In this case, commutation is investigated as a channel transmitting information defining the starting state (return state respectively) for each model operating in one specific mode. Commutation will be ensured by

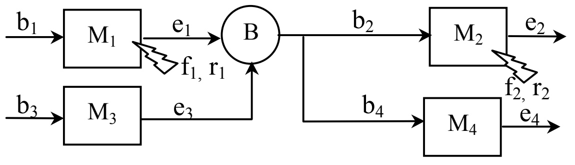
[3] we write $\delta(q,s)!$ as an abbreviation of $\delta(q,s)$ is defined.

[4] A regular expression over Σ is a formal expression obtained by a finite number of applications of operations +, ., *

an information channel formally defined using the notion of projection.

3.1 Example of multi-model process

The aim is to generalize the formalism introduced by (Kamach et al, 2002) to n models (with $n > 2$). To introduce the proposed approach, we consider a simple manufacturing system, in which four different overall system models are considered: nominal mode is represented by model G_n and there are three downgraded modes G_{d1}, G_{d2} and G_{d3} (figure 3). This system features four machines as shown in figure 1. Initially, buffer B is empty and machines M_3 and M_4 are performing other tasks outside the unit, but which intervene when M_1 (respectively M_2) breaks down (event f_1, respectively f_2 represented by f_i in figure 2). With event b_1 (respectively b_3), M_1 (respectively M_3) takes a workpiece from an infinite bin and enters $q_{1,1}$ or $q_{3,1}$ state of G_n (respectively $q_{2,1}$ or $q_{4,1}$ states). It then deposits it in buffer B after completing its work. M_2 (respectively M_4) operates similarly, but takes its workpiece from B and enters $q_{1,2}$ or $q_{2,2}$ state (respectively $q_{3,2}$ or $q_{4,2}$). It then deposits it in an infinite output bin, when it has finished its task.



bi : beginning of a task on M_i i ={1,.., 4}
e_i : end of task on M_i : i ={1,.., 4}
f_j : failure of M_j : j ={1, 2}
r_j : repair of M_j : j ={1, 2}

Figure 1.a : schematic description of the production unit example

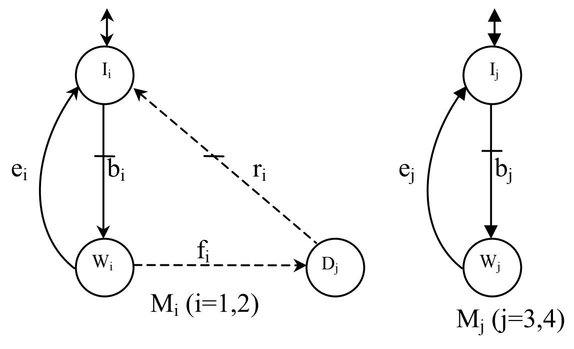
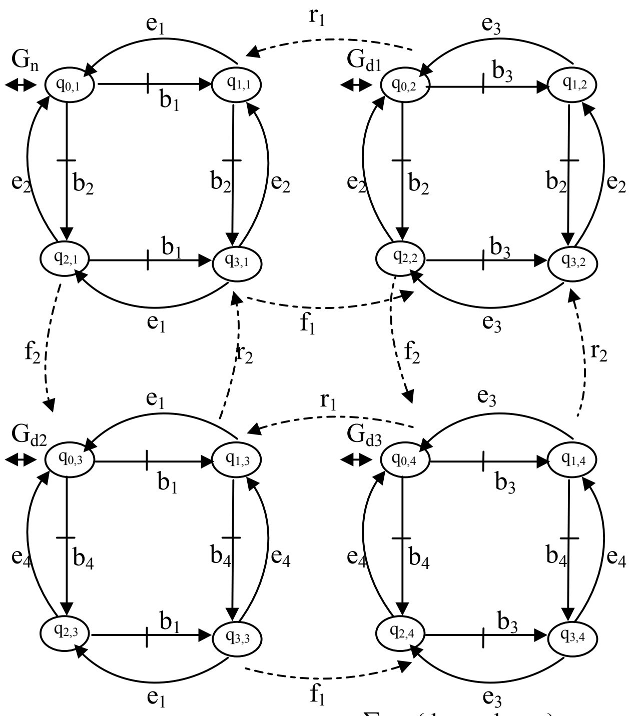


Figure 2 : automata models of machines M_i, M_j

We assume that only M_1 and M_2 can break down and that M_1 (respectively M_2) can not be repaired if M_3 (respectively M_4) is working.
Possible operating modes are represented in figure 3.



$\Sigma = \Sigma_n \cup \Sigma_{d1} \cup \Sigma_{d2} \cup \Sigma_{d3} \cup \Sigma'$
with $\Sigma' = \{f_1, r_1, f_2, r_2\}$

$\Sigma_n = \{ b_1, e_1, b_2, e_2\}$
$\Sigma_{d1} = \{ b_2, e_2, b_3, e_3\}$
$\Sigma_{d2} = \{ b_1, e_1, b_4, e_4\}$
$\Sigma_{d3} = \{ b_3, e_3, b_4, e_4\}$

Figure.3 : four possible production unit models

3.2 Formal description of multi-model commutation management

The aim is to determine formally each operating mode and the commutation conditions. To do this, we define Λ as a set containing indices of all models composing the global system with $\text{card}(\Lambda) = n < \infty$. card($\Lambda$) represents the number of models to be designed. In our case, $\Lambda = \{n, d1, d2, d3\}$, so $\text{card}(\Lambda) = 4$.
Let $\lambda_i \in \Lambda$ with $i \in \{1,..,4\}$. We define $G_{\lambda i}$ as an uncontrollable DES, taken to be an automaton of model λ_i. Formally $G_{\lambda i} = (Q_{\lambda i}, \Sigma_{\lambda i}, \delta_{\lambda i}, q_{0,\lambda i}, Q_{m,\lambda i})$.
We assume that $\Sigma_{\lambda i} \cap \Sigma_{\lambda j} \neq \varnothing$ and initially the system is described by $G_{\lambda 1}$.
Let us define $\Sigma'_{\lambda i,\lambda j} = \cup\{\alpha_{\lambda i,\lambda j}\}$ as the set representing the commutation event from $G_{\lambda i}$ (respectively $G_{\lambda j}$) to $G_{\lambda j}$ (respectively $G_{\lambda i}$). When commutation event $\alpha_{\lambda i,\lambda j}$, occurs, the process model becomes $G_{\lambda j}$. In this case, we must determine the arrival state of $G_{\lambda j}$ after commutation and must direct $G_{\lambda i}$ to an inactive state to disable its action. Intuitively, newly enabled $G_{\lambda j}$ must leave its inactive state and be directed to a state which compatible with the overall system evolution. To do this, we introduce theorem 1 which ensures commutation from $G_{\lambda i}$ to $G_{\lambda j}$ by using the trace or memory of all strings that can occur from $G_{\lambda 1}$ to $G_{\lambda i}$. This memory mechanism is important to ensure overall system tracking. Let us suppose the system is represented by G_n (figure 3). Commutation event f_1 (failure event) is possible from $q_{1,1}$ or $q_{3,1}$. If f_1 is

generated from $q_{1,1}$, then the memorized string occurring in G_n is $(b_1e_1)^*b_3$ and, intuitively, G_{d1} must be directed to $q_{0,2}$. But if f_l is occurred from $q_{3,1}$ then the memorized string is $(b_1e_1)^*b_1(b_2e_2)^*b_2$ or $(b_2e_2)^*b_2(b_1e_1)^*b_1$. In this case, G_{d1} must be directed to $q_{2,2}$. Theorem 1 formalizes these intuitive results. But before introducing this theorem, two steps are required. Firstly, we introduce an inactive state to any $G_{\lambda i}$ to disable its action. Secondly, we introduce channel information (represented by the projection map) to ensure process tracking. The projection (noted $\pi_{\lambda_i \to \lambda_j}$ in the remainder of the paper) tells us whether any component belonging simultaneously to $G_{\lambda i}$ and $G_{\lambda j}$ (machine M_2 for G_n and G_{d1}) is working or not, to decide whether to direct $G_{\lambda j}$ to a state in which M_i is working, thereby ensuring system tracking.

a) extension of $G_{\lambda i}$ and $G_{\lambda j}$

Let us extend $G_{\lambda i}$ and $G_{\lambda j}$ by adding an inactive state $q_{in,\lambda i}$ to the state set of the model $G_{\lambda i}$ and an inactive state $q_{in,\lambda j}$ to the $G_{\lambda j}$ state set respectively. Occurrence of commutation event $\alpha_{\lambda i,\lambda j}$ will direct model $G_{\lambda i}$ to its inactive sate $q_{in,\lambda i}$ and activate $G_{\lambda j}$ from $q_{in,\lambda j}$. So, for model $G_{\lambda i}$, the extended model will be defined as follows:

$G_{\lambda i,ext} = (Q_{\lambda i,ext}, \Sigma_{\lambda i,ext}, \delta_{\lambda i,ext}, q_{0,\lambda i,ext}, Q_{m,\lambda i,ext})$ wit h

$Q_{\lambda i,ext} = Q_{\lambda i} \cup \{q_{in,\lambda i}\}$

$\Sigma_{\lambda i,ext} = \Sigma_{\lambda i} \cup \Sigma'_{\lambda i,\lambda j}$

$q_{0,\lambda i,ext} = q_{0,\lambda 1} \quad \text{if} \quad \lambda i = \lambda 1$

$q_{0,\lambda i,ext} = q_{in,\lambda_i} \quad \text{if} \quad \lambda i \neq \lambda 1$

$Q_{m,\lambda i,ext} = Q_{m,\lambda i}$

$\delta_{\lambda i,ext}$ is defined as follows (Kamach et al., 2002, 2003):

1) ∀∀ θ ∈ Θλι, ανδ ∀ σ Σλι, ιφ δλι(θ, σ)!, τηεν

$\delta_{\lambda i,ext}(q, \sigma) := \delta_{\lambda i}(q, \sigma)$;

2) ∀∀ θ ∈ Θ λι from which $\alpha_{\lambda i,\lambda j}$ can occur (with $i \neq j$ then $\delta_{\lambda i,ext}(q, \alpha_{\lambda i,\lambda j}) := q_{in,\lambda_i}$.

Similarly, $G_{\lambda j,ext}$ will be defined by the same way that $G_{\lambda i,ext}$.

b) formalization of system tracking

The aim is to define $\delta_{\lambda j,ext}(q_{in,\lambda i}, \alpha_{\lambda j,\lambda i})$. Initially, $G_{\lambda j,ext}$ is in inactive state $q_{in,\lambda j}$. When commutation event $\alpha_{\lambda i,\lambda j}$ occurs, $G_{\lambda j,ext}$ will leave $q_{in,\lambda\varphi}$ to reach a state $q \in Q_{\lambda j}$. As shown in figure 4, when event f_l occurs, $G_{d1,ext}$, which is in $q_{in,d1}$, can be directed to $q_{0,2}$, $q_{1,2}$, $q_{2,2}$ or $q_{3,2}$. So $G_{d1,ext}$ becomes a nondeterministic automaton. To prevent this nondeterministic situation, we introduce projection map $\pi_{\lambda i \to \lambda j}$ (Kamach et al., 2002) as:

$\pi_{\lambda i \to \lambda j} : \Sigma_{\lambda i} \to \Sigma_{\lambda j}$ so that

$$\pi_{\lambda i \to \lambda j}(\sigma) = \begin{cases} \sigma & \text{if } \sigma \in (\Sigma_{\lambda i} \cap \Sigma_{\lambda j}) \\ \varepsilon & \text{otherwise} \end{cases}$$

We extend $\pi_{\lambda i \to \lambda j}$ to be defined over a language, so that:

$(\pi_{\lambda i \to \lambda j})_{ext} : \Sigma^*_{\lambda i} \to \Sigma^*_{\lambda j}$ such that:

$(\pi_{\lambda i \to \lambda j})_{ext}(\varepsilon) = \varepsilon$ and

$$(\pi_{\lambda i \to \lambda j})_{ext}(s\sigma) = \begin{cases} (\pi_{\lambda i \to \lambda j})_{ext}(s)\sigma & \text{if } \sigma \in (\Sigma_{\lambda i} \cap \Sigma_{\lambda j}) \\ (\pi_{\lambda i \to \lambda j})_{ext}(s) & \text{otherwise} \end{cases}$$

That is, $(\pi_{\lambda i \to \lambda j})_{ext}$ is a projection whose effect on a string $s \in \Sigma^*_{\lambda i}$ is to eliminate all events σ of s that do not belong to $(\Sigma_{\lambda i} \cap \Sigma_{\lambda j})$. Projection $(\pi_{\lambda i \to \lambda j})_{ext}$ allows, from $G_{\lambda j}$, identification of the output states of intersection elements in $G_{\lambda i}$ when $\alpha_{\lambda i,\lambda j}$ occurs. Thus, from $(\pi_{\lambda i \to \lambda j})_{ext}(s)$, we can determine whether components belonging to $G_{\lambda i}$ and $G_{\lambda j}$ are working or not to direct $G_{\lambda j}$ to a state compatible with a component situation.
Note that in the remainder of this paper we will express $(\pi_{\lambda i \to \lambda j})_{ext}$ as $\pi_{\lambda i \to \lambda j}$.
The above demonstration attempts to prove that generalization of commutation resolution assumes a recurrent form. It shows that the information channel, materialized by the projection function, retains only the common components maintained from mode λ_i to λ_j.

4. DETERMINING $G_{\lambda i,ext}$ STARTING STATES

Let us suppose that the set of commutation events produced from $G_{\lambda 1}$ to $G_{\lambda j}$ is
$\alpha_{\lambda 1,\lambda l}$, $\alpha_{\lambda l,\lambda k}$,… $\alpha_{\lambda j,\lambda i}$, $\alpha_{\lambda i,\lambda n}$ where $1<l<k<\ldots<j<i<n$, *i.e.* model $G_{\lambda n,ext}$ is now activated. The starting state of this model is determined by transition function $(\delta_{\lambda n,ext}(q_{in,\lambda n}, \alpha_{\lambda i,\lambda n}))$ given by theorem 1.

Theorem 1
$\forall$ n $\geq$ l, the extended transition function $(\delta_{\lambda n,ext}(q_{in,\lambda n}, \alpha_{\lambda i,\lambda n}))$ *of model $G_{\lambda n,ext}$ is given by*

$$\delta_{\lambda n,ext}(q_{in,\lambda n}, \alpha_{\lambda i,\lambda n}) =$$
$$\delta_{\lambda n}\left[q_{0,\lambda n}, \pi_{\lambda i \to \lambda n}(\pi_{\lambda j \to \lambda i}(\ldots(\pi_{\lambda l \to \lambda k}(\pi_{\lambda 1 \to \lambda l}(s_1)s_l)..)s_i)\right]$$

□

Theorem 1 permits determination of the next state to be reached in $G_{\lambda n,ext}$ newly enabled by scanning all strings generated from $G_{\lambda 1}$ to $G_{\lambda i}$. To illustrate this theorem, we consider the models shown in figure 4.
The aim is to determine possible starting states of model $G_{d1,ext}$ after generation of failure event f_l in initial model $G_{n,ext}$. $\alpha_{\lambda 1,\lambda 2} = \alpha_{v,\delta 1} = f_l$ can occur from state $q_{1,1}$ or state $q_{3,1}$ of $G_{n,ext}$ because f_l failure is possible only when M_l is working (event b_l).

Case1) First, let us suppose that f_1 has occurred from $q_{1,1}$. According to theorem 1, we have

$$\delta_{d1,ext}(q_{in,d1}, \alpha_{n,d1}) = \delta_{d1,ext}(q_{in,d1}, f_1) =$$
$$\delta_{d1}(q_{0,d1}, \pi_{n\to d1}(s_1)).$$

This means we need to determine string s_1. To do this, we introduce language

$$K_{q_{1,1}} = \left\{\omega \in \Sigma_n^* \mid \delta_n(q_{0,1},\omega) = q_{1,1}\right\}$$

$K_{q_{1,1}}$ is the set of event sequences belonging to G_n, so that f_1 is the next event to occur following generation of string ω. $K_{q_{1,1}}$ can be determined from Arden's lemma, as follows:

Let us consider model G_n in figure 3. The aim is to determine ω such that $\delta_n(q_{0,1},\omega) = q_{1,1}$. To do this, we mark state $q_{1,1}$ of G_n and obtain the following equation system:

$$\begin{cases} q_{0,1} = b_1q_{1,1} + b_2q_{2,1} & (1) \\ q_{1,1} = b_2q_{3,1} + e_1q_{0,1} + \varepsilon & (2) \\ q_{2,1} = b_1q_{3,1} + e_2q_{0,1} & (3) \\ q_{3,1} = e_2q_{1,1} + e_1q_{2,1} & (4) \end{cases}$$

The aim is to determine $q_{0,1}$.

Then (4) $\Rightarrow q_{3,1} = e_1(b_1q_{3,1} + e_2q_{0,1}) + e_2(b_2q_{3,1} + e_1q_{0,1} + \varepsilon)$

$= (e_1b_1 + e_2b_2)q_{3,1} + (e_1e_2 + e_2e_1)q_{0,1} + e_2$ (4')

(4') $\Rightarrow q_{3,1} = A\, q_{3,1} + B$ where

$A = e_1b_1 + e_2b_2$

$B = (e_1e_2 + e_2e_1)q_{0,1} + e_2$.

Since $\varepsilon \notin A$, then (4') allows unique solution A^*B, i.e.

$q_{3,1} = (e_1b_1 + e_2b_2)^*[(e_1e_2 + e_2e_1)q_{0,1} + e_2]$

$\Rightarrow q_{3,1} = (e_1b_1 + e_2b_2)^*(e_1e_2 + e_2e_1)q_{0,1} + (e_1b_1 + e_2b_2)^*e_2$ (4'')

Substituting $q_{3,1}$ in (1) and (2) we obtain:

$$\Rightarrow \begin{cases} q_{1,1} = b_2(e_1b_1 + e_2b_2)^*(e_1e_2 + e_2e_1)q_{0,1} + b_2(e_1b_1 + e_2b_2)^*e_2 + e_1\, q_{0,1} + \varepsilon \\ q_{1,2} = b_1(e_1b_1 + e_2b_2)^*(e_1e_2 + e_2e_1)q_{0,1} + b_1(e_1b_1 + e_2b_2)^*e_2 + e_2\, q_{0,1} \end{cases}$$

$$\Rightarrow \begin{cases} q_{1,1} = [b_2(e_1b_1 + e_2b_2)^*(e_1e_2 + e_2e_1) + e_1]q_{0,1} + b_2(e_1b_1 + e_2b_2)^*e_2 + \varepsilon \quad (1') \\ q_{1,2} = [b_1(e_1b_1 + e_2b_2)^*(e_1e_2 + e_2e_1) + e_2]q_{0,1} + b_1(e_1b_1 + e_2b_2)^*e_2 \quad (2') \end{cases}$$

Replacing (1') and (2') in equation (1), we obtain:

$q_{0,1} = [b_1b_2(e_1b_1 + e_2b_2)^*(e_1e_2 + e_2e_1) + b_1e_1]q_{0,1} + b_1b_2(e_1b_1 + e_2b_2)^*e_2 + b_1 + [b_2b_1(e_1b_1 + e_2b_2)^*(e_1e_2 + e_2e_1) + b_2e_2]q_{0,1} + b_2b_1(e_1b_1 + e_2b_2)^*e_2$

$\Rightarrow q_{0,1} = [(b_1b_2 + b_2b_1)(e_1b_1 + e_2b_2)^*(e_1e_2 + e_2e_1) + b_1e_1 + b_2e_2]q_{0,1} + (b_1b_2 + b_2b_1)(e_1b_1 + e_2b_2)^*e_2 + b_1$. (1'')

(1'') $\Leftrightarrow q_{0,1} = Aq_{0,1} + B$.

According to Arden's lemma, $\varepsilon \notin [(b_1b_2 + b_2b_1)(e_1b_1 + e_2b_2)^*(e_1e_2 + e_2e_1) + b_1e_1 + b_2e_2]$, then (1'') allows unique solution $A^*B = K_{q_{1,1}}$ with

$K_{q_{1,1}} = [(b_1b_2 + b_2b_1)(e_1b_1 + e_2b_2)^*(e_1e_2 + e_2e_1) + b_1e_1 + b_2e_2]^*[(b_1b_2 + b_2b_1)(e_1b_1 + e_2b_2)^*e_2 + b_1]$.

We can now determine the starting state of $G_{d1,ext}$ by applying theorem 1. In fact,

$\pi_{n\to d1}(K_{q_{1,1}}) = [b_2(e_2b_2)^*e_2 + b_2e_2]^*[b_2(e_2b_2)^*e_2] = (b_2e_2)^*$

because $b_2(e_2b_2)^*e_2 = (b_2e_2)^*$, $(b_2e_2)^* + b_2e_2 = (b_2e_2)^*$, then $[b_2(e_2b_2)^*e_2 + b_2e_2]^* = ((b_2e_2)^*)^* = (b_2e_2)^*$, because in regular algebra (Wonham, 2001) $(L^*)^* = L^*$.

i.e. $[b_2(e_2b_2)^*e_2] = (b_2e_2)^*$. Since $L^*L^* = L^*$, then $[b_2(e_2b_2)^*e_2 + b_2e_2]^*[b_2(e_2b_2)^*e_2] = (b_2e_2)^*$

So $\delta_{d1,ext}(q_{in,d1}, f_1) = \delta_{d1}(q_{0,d1}, \pi_{n\to d1}(K_{q_{1,1}})) =$

$\delta_{d1}(q_{0,d1}, (b_2e_2)^*) = q_{0,2}$ (figure 4).

Theorem 1 therefore confirms the intuitive results referred to in section 3.2.

Case2) Now if f_1 occurs from $q_{3,1}$ of $G_{n,ext}$, . Arden's lemma gives us $K_{q_{3,1}}$ so that

$$K_{q_{3,1}} = \left\{\omega \in \Sigma_n^* \mid \delta_n(q_{0,1},\omega) = q_{3,1}\right\}$$

Marking the state $q_{3,1}$ of $G_{n,ext}$,, the above equation system becomes:

$$\begin{cases} q_{0,1} = b_1q_{1,1} + b_2q_{2,1} \\ q_{1,1} = b_2q_{3,1} + e_1q_{0,1} \\ q_{2,1} = b_1q_{3,1} + e_2q_{0,1} \\ q_{3,1} = e_2q_{1,1} + e_1q_{2,1} + \varepsilon \end{cases}$$

As previously, we obtain:

$K_{q_{3,1}} = [(b_1b_2 + b_2b_1)(e_1b_1 + e_2b_2)^*(e_1e_2 + e_2e_1) + b_1e_1 + b_2e_2]^*[(b_1b_2 + b_2b_1)(e_1b_1 + e_2b_2)^*]$.

So $\pi_{n\to d1}(K_{q_{3,1}}) = (b_2e_2)^*b_2$

Therefore

$\delta_{d1}(q_{0,d1}, \pi_{n\to d1}(K_{q_{31,1}})) = \delta_{d1}(q_{0,d1}, (b_2e_2)^*b_2)$

$= q_{2,2}$ of $G_{d1,ext}$ (figure 4).

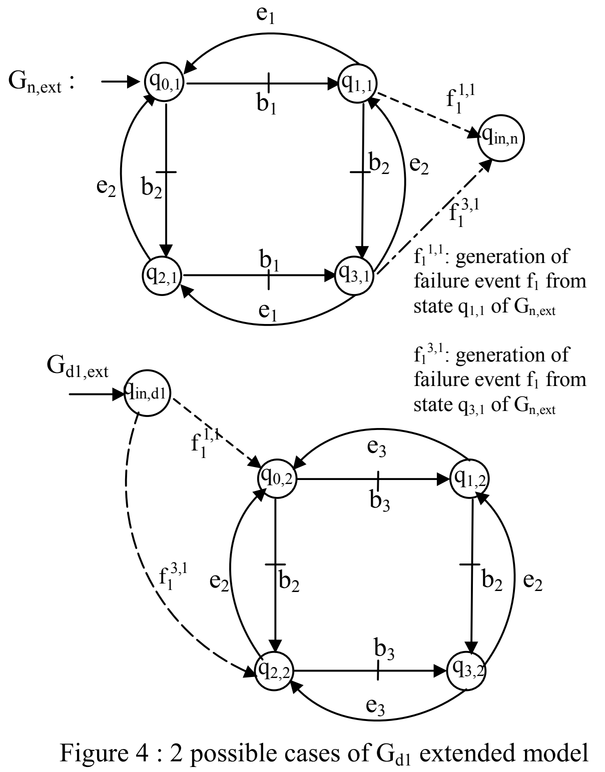


Figure 4 : 2 possible cases of G_{d1} extended model after generating f_1.

With a view to generalizing commutation, suppose the system is represented by $G_{d1,ext}$ and that failure event f_2 has occurred (figure 3). Machine 4 will then replace machine 2. Production will be ensured by machines 3 and 4 and the system will be represented by model $G_{d3,ext}$. In this case, we must determine possible starting states of model $G_{d3,ext}$, knowing that the system is now represented by model $G_{d1,ext}$. From $G_{d1,ext}$, two cases are possible:

Case 3) starting state of $G_{d1,ext}$ is $q_{0,2}$ (figure 4)
In $G_{d1,ext}$, failure event f_2 can occur from states $q_{2,2}$ or $q_{3,2}$. In this case, we compute the language $K_{q_{2,2}}$ ($K_{q_{3,2}}$ respectively). So that

$$K_{q_{2,2}} = \left\{\omega\in \Sigma_{d1}^* \mid \delta_{d1}(q_{0,2},\omega) = q_{2,2}\right\} \text{ and}$$

$$K_{q_{3,2}} = \left\{\omega\in \Sigma_{d1}^* / \delta_{d1}(q_{0,2},\omega) = q_{3,2}\right\}$$

1) Suppose f_2 is generated from $q_{2,2}$ of $G_{d1,ext}$ ($f_2^{2,2}$ in figure 5.a). In this case, we mark state $q_{2,2}$ and, as above, we obtain

$\pi_{d1\to d3}(K_{q_{2,2}}) = (b_3e_3)^*$.

Thus, if the starting state of $G_{d1,ext}$ is $q_{0,2}$ and f_2 has occurred from $q_{2,2}$, then from theorem 1 we obtain:

$\delta_{d3}(q_{0,4}, \pi_{d1\to d3}(\pi_{n\to d1}(K_{q_{1,1}})K_{q_{2,2}}) = \delta_{d3}(q_{0,4}, (b_3e_3)^*)$

$=q_{0,4}$ (figure 5.b). The starting state of $G_{d3,ext}$ will then be $q_{0,4}$.

2) f_2 is now generated from $q_{3,2}$ of $G_{d1,ext}$ ($f_2^{3,2}$ of figure 5.a).
We mark state $q_{3,2}$, then

$\pi_{d1\to d3}(K_{q_{3,2}}) = b_3(e_3b_3)^* = (b_3e_3)^*b_3$

Therefore $\delta_{d3}(q_{0,4}, \pi_{d1\to d3}(\pi_{n\to d1}(K_{q_{1,1}})K_{q_{3,2}})$

$=\delta_{d3}(q_{0,4}, (b_3e_3)^* b_3) = q_{1,4}$. i.e. if the starting state of $G_{d1,ext}$ is $q_{0,2}$ and if f_2 has occurred from $q_{3,2}$, , then based on theorem 1, $G_{d3,ext}$ will be directed to state $q_{1,4}$ (figure 5.b**)**.

Case 4) Suppose now that the starting state of $G_{d1,ext}$ is $q_{2,2}$ **(**figure 4**)**, thus

$$K_{q_{2,2}} = \left\{\omega\in \Sigma_{d1}^* \mid \delta_{d1}(q_{2,2},\omega) = q_{2,2}\right\} \text{ and}$$

$$K_{q_{3,2}} = \left\{\omega\in \Sigma_{d1}^* \mid \delta_{d1}(q_{2,2},\omega) = q_{3,2}\right\}$$

1) Suppose first that f_2 is generated from $q_{2,2}$ of $G_{d1,ext}$ ($f_2^{2,2}$ of figure 5.c).
We mark state $q_{2,2}$, and so $\pi_{d1\to d3}(K_{q_{2,2}}) = (b_3e_3)^*$.

Therefore, based on theorem 1,

$\delta_{d3,ext}(q_{in,d3}, f_2) = \delta_{d3}(q_{0,4}, (b_3e_3)^*) = q_{0,4}$.

Thus, from $G_{\lambda 1,ext}$ and after $f_l^{1,3}$ and $f_2^{2,2}$, have occurred, $G_{d3,ext}$ will be directed to state $q_{0,4}$ (figure 5.d).

2) If f_2 is now generated from $q_{3,2}$ of $G_{d1,ext}$ ($f_2^{3,2}$ in figure 5.c)
As above, we mark state $q_{3,2}$.

Then $\delta_{d3,ext}(q_{in,d3}, f_2) = \delta_{d3}(q_{0,4}, b_3(b_3e_3)^*) = q_{1,4}$

Thus, after $f_l^{1,3}$ and $f_2^{3,2}$ have occurred, $G_{d3,ext}$ will be directed to state $q_{1,4}$ (figure 5.d).
At this stage we can verify theorem 1. In fact

$\delta_{\lambda_{4,ext}}(q_{in,d3}, f_2^{3,2}) = \delta_{d3}(q_{0,4}, \pi_{d1\to d3}(\pi_{n\to d1}(K_{q_{3,1}})K_{q_{3,2}})))$

$= q_{1,4}$.

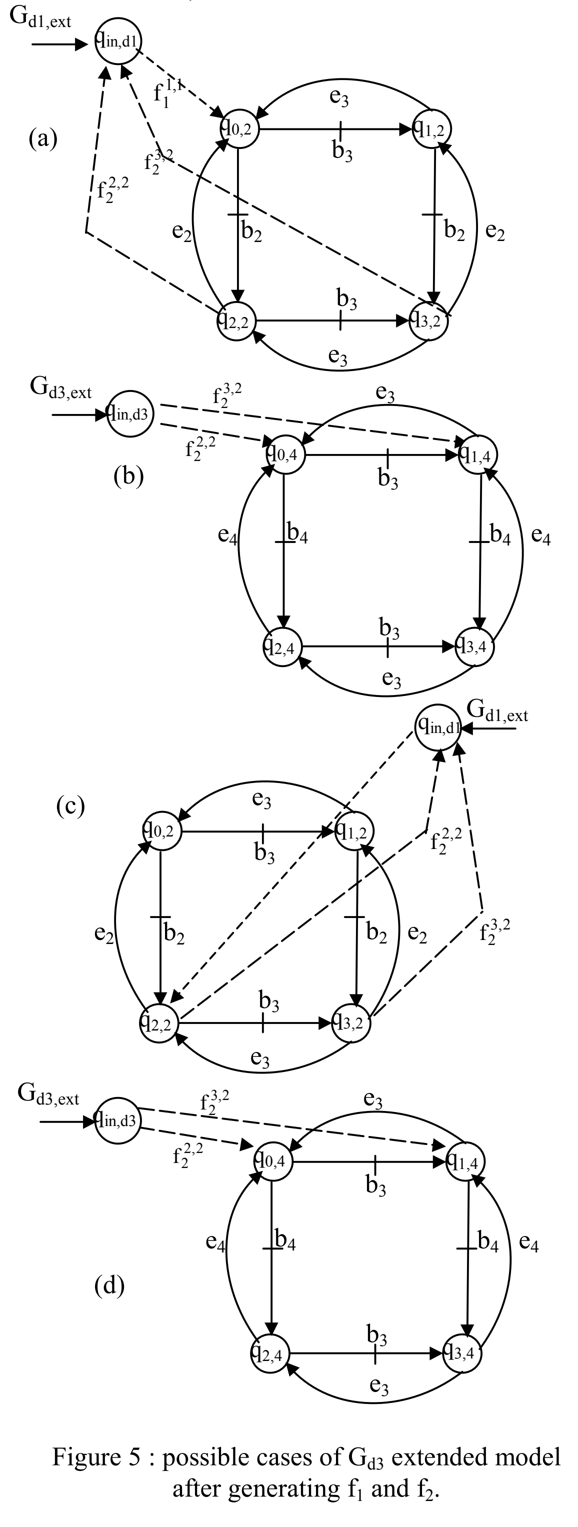


Figure 5 : possible cases of G_{d3} extended model after generating f_l and f_2.

5. DETERMINING $G_{\lambda i,ext}$ RECOVERY STATES

We assume that after generating commutation events $\alpha_{\lambda 1,\lambda l}$, $\alpha_{\lambda l,\lambda k}$,…, $\alpha_{\lambda j,\lambda i}$, $\alpha_{\lambda i,\lambda n}$, event $\alpha_{\lambda n,\lambda i}$ (in our case, a repair event) can occur. In this case, $G_{\lambda n,ext}$ will be directed to its inactive state $q_{in,n}$ and $G_{\lambda i,ext}$ will be simultaneously activated, leaving its inactive state $q_{in,i}$ to enter recovery state $q \in G_{\lambda i}$. To do this, the entire system evolution history must be known i.e. all strings generated from $G_{\lambda 1}$ to $G_{\lambda i}$ and all strings that will occur from $G_{\lambda n}$ to $G_{\lambda l}$, which will be reached a second time, must be memorized. Theorem 2 is introduced to ensure this mechanism.

Theorem 2:
$\forall n \geq l$ the extended transition function $\delta_{\lambda\, i,ext}(q_{in,\lambda i}, \alpha_{\lambda n,\lambda i})$ of model $G_{\lambda i,ext}$ is given by:

$$\delta_{\lambda i,ext}(q_{in,\lambda i}, \alpha_{\lambda n,\lambda i}) = \delta_{\lambda i,ext}(q_{0,\lambda_i}, \pi_{\lambda n\to\lambda i}(\pi_{\lambda i\to\lambda n}(\pi_{\lambda j\to\lambda i}(\ldots(\pi_{\lambda l\to\lambda k}(\pi_{\lambda l\to\lambda 1}(s_1)s_l)\ldots)s_i)s_n)))$$

Let consider the unit production example and suppose that after generation of f_1 and f_2 failure events, unit production is represented by model $G_{d3,ext}$. At this level, repair event r_1 (r_2 respectively) can occur. If this is the case, the system will then be directed to $G_{d2,ext}$ ($G_{d1,ext}$ respectively). On the other hand, when describing our unit production, we have assumed that M_1 can not be repaired if M_3 is working. Consequently in $G_{d3,ext}$, repair event r_1 can occur only from states $q_{0,4}$ or $q_{2,4}$.
However, in cases 3 and 4 described in the last section, we demonstrated that after generating f_1 and f_2, $G_{d3,ext}$ is activated and directed to $q_{0,4}$ or $q_{1,4}$ states (figure 5). Two cases can therefore be distinguished.

Case 5) starting state of $G_{d3,ext}$ is $q_{0,4}$ ($f_2^{2,2}$ in figure 6.a).
r_1 can then occur from $q_{0,4}$ or $q_{2,4}$

1) Suppose that r_1 has occurred from $q_{0,4}$,
$K_{q_{0,4}} = \left\{\omega \in \Sigma_{d3}^* \mid \delta_{d3}(q_{0,4},\omega) = q_{0,4}\right\}$, we mark $q_{0,4}$ and then
$K_{q_{0,4}} = [(b_3b_4 + b_4b_3)(e_4b_4 + e_3b_3)^*(e_3e_4 + e_4e_3) + b_3e_3 + b_4e_4]^*$
$\Rightarrow \pi_{d3\to d2}(K_{q_{0,4}}) = (b_4e_4)^*$
So $\delta_{d2,ext}(q_{in,d2}, r_1) =$
So $\delta_{d2,ext}(q_{in,d2}, r_1) =$
$\delta_{d2}(q_{0,3}, \pi_{d3\to d2}(\pi_{d1\to d3}(\pi_{n\to d1}(K_{q_{1,1}})K_{q_{2,2}})K_{q_{0,4}}) = q_{0,3}$
(figure 6.b).

2) if r_1 has occurred from $q_{2,4}$, then
$K_{q_{2,4}} = \left\{\omega \in \Sigma_{d3}^* \mid \delta_{d3}(q_{0,4},\omega) = q_{2,4}\right\}$, by marking $q_{2,4}$ we obtain $\pi_{d3\to d2}(K_{q_{2,4}}) = (b_4e_4)^*\, b_4$
So $\delta_{d2,ext}(q_{in,d2}, r_1) =$
$\delta_{d2}(q_{0,3}, \pi_{d3\to d2}(\pi_{d1\to d3}(\pi_{n\to d1}(K_{q_{1,1}})K_{q_{2,2}})K_{q_{2,4}}) = q_{2,3}$
(figure 6.b)

Case 6) if the starting state of $G_{d3,ext}$ is $q_{1,4}$ ($f_2^{3,2}$ of figure 6.c), applying theorem 2 provides us with the same results as those of case 5.

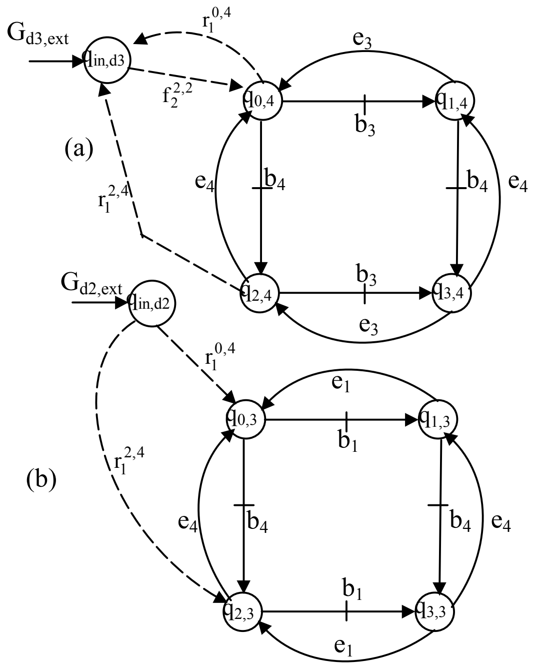


Figure 6 : possible cases of G_{d2} extended models after generating f_1, f_2 and r_1

6. CONCLUSION

We conclude that the proposed method ensures commutation between different models of an overall system reacting to exceptional situations, such as failure event occurrence A major contribution of this paper is that it considers reactive systems with different objectives. Each objective (or operating mode) is represented by a process model. If we assume that different models develop independently, the main problem is then to de-activate model $G_{\lambda i}$ and to commute to model $G_{\lambda j}$, which will be considered as the current process model until an exceptional event occurs. A formal framework based on tracking events is proposed in to ensure commutation. This framework extends for the generalization case the projection definition. Theorems 1 and 2 represent the main contribution of this paper; they allow us to determine with a recurrent form the arrival state of a model after any commutations. Results presented in (Kamach et al, 2003) allow us to obtain controller synthesis for each operating mode.

REFERENCES

Cassandras, B.A. and Lafortune, S (1999). Introduction to Discrete Event Systems. *1st edition: Kluwer Academic Publishers, Boston,* 822 p.

Chafik, S. and E. Niel (2001) Du Contrôle par Supervision Hiérarchique Distribuée. *MSR: modélisation des systèmes réactifs*, pp. 55-70. Toulouse, France,

Harel, D. and A. Naamadl (1996) Statemate semantics of statecharts. *ACM transitions of software engineering and methodology*, vol. 5, no.

4. pp. 293-333.

Kamach, O., S. Chafik., L. Pietrac and E. Niel (2002). representation of a reactive systems with different models. *IEEE SMC*. reference TA2L4 in CDROM, 6-9. Hammamate, Tunisia

Kamach, O., L. Pietrac and E. Niel (2003). multi-model approach for discrete event systems: application to operating modes management. I*EEE CESA*. Reference S2-R-00-0315 in CD ROM. école centrale de Lille, France.

Lin, F. and Wonham, W. M (1988). Decentralised supervisory control of discrete event systems. *Information sciences*, vol. 25, no.5, pp. 1202-1218.

Ramadge, P and Wonham, W. M (1987). Supervisory control of class of discrete event processes. *SIAM Journal of Control and optimisation*, Vol. 25, no.1, pp. 206-230.

Rudie, K. S, Lafortune, and F.Lin (1999). Minimal communication in a distributed Descrete-Event Control System. *In proceedings of the american control conferenc ,* pp. 1965-1970.,

Wong, K.C., J.G. Thiste., R.P. Malhame. and H.H. Hoang (2000). supervisory control of distributed systems: conflict resolution. Discrete event dynamic systems, vol. 10, pp. 131-186

Wonham, W.M. (2002). notes on control of Discrete event systems, Dep of electrical and computer engineering, University of Toronto, Canada. URL: http://odin.control.toronto.edu/DES/

Yoo, T. and Lafortune, S (2002). New Results: a new architecture with a dynamic decision fusion rules. *6th international workshop on discrete event systems,* pp. 11-17. Saragossa, Spain,