Formal Approach to Multimodal Control Design: Application to Mode Switching

Grégory Faraut, Laurent Piétrac, Eric Niel

Formal Approach to Multimodal Control Design: Application to Mode Switching

Gregory Faraut, *Member, IEEE*, Laurent Piétrac, *Member, IEEE*, and Eric Niel

***Abstract*—A framework based on Supervisory Control Theory (SCT) is proposed to assist the design of multi-modal control for discrete-event systems (DESs). Our purpose handled modes which are conceptualized by using multi-model approach. Each mode represents a running part of the system, depending on the requirements to enforce and resources to activate. The resulted framework aims to design each mode independently first, and resolves conflicting connections between them secondly. The proposal carries out a formal way to build the final ready-to-use control laws. A flexible manufacturing system illustrates this approach.**

***Index Terms*—Automata, discrete-event systems (DESs), mode switching, multimodel control design, multimodal system, supervisory control theory (SCT).**

I. Introduction

INITIATED by Ramadge and Wonham [1], Supervisory Control Theory (SCT) has significatively improved results in the discrete-event systems (DESs) domain. Properties such as safety, liveness, controllability, observability and, more recently, diagnosability have been introduced to assess formally control architecture. Basically supported by finite-state machines, SCT applicability for industrial application schemes does not seems easy. In fact, the design of real control applications implies very large models, with two main problems. The first pertains to scalability because of state-space explosion: real system models may be too large to be computed. The second problem pertains to the interpretation of the models: larger models are difficult to understand even if computation is successful. To solve scalability, several approaches have been proposed in terms of control architecture: modular [2], [3], decentralized [4], [5], hierarchical [6], [7], and even hierarchical and distributed [8], [9].

Even if a decomposition is used to reduce complexity, these approaches always handle the whole process and the whole specification. However, for numerous systems, user requirements depend also upon specific instants. For example, sequential modes for hybrid systems (chemical batch processes need to be cleaned and prepared before production [10], [11]), switched systems whose control objectives and system structure (components and their interactions) changes in a discontinuous manner. Designing such systems implies particularities either for process or specification.

- Within hybrid systems, one different continuous model is used for each dynamic regime (a mode) [12], [13], and one discrete model manages switches among modes.
- Within reconfigurable systems, designed to time-to-market reduction [14]–[16], the management of a set of control laws is well adapted to fast changing schemes.
- Within fault-tolerant systems [17], [18], specifications or components structure temporarily fulfil some particular aims when faults occur.

From a control point-of-view, one can remain with these assumption, that the set of requirements need not be entirely fulfilled at the same moment. Nominal functioning, degraded functioning, underpower functioning, etc., define modes in which a set of requirements would have to be enforced by the controller, acting in each mode on a part of the process. This kind of control law, which defines a sequence of operating modes, is not exclusive to manufacturing systems. Embedded systems [19]–[21] for cars also apply particular control laws according to the active operating mode: startup, ABS on/off, cruise control on/off... For each mode, some components of the whole system are engaged, and requirements may be very different from one to another.

In DES, numerous works are focused on multimodal control law. However, most of them applied compositional formalisms on modeling configurations: for instance state charts [22], mode charts [23], hierarchical finite-state machines, and mode automata [24]. In these approaches, the model of the process is not specifically representative of the state of the process, and thus, is unable to automatically detect (using the synthesis or validation) the issues about mode switching. Formal approach like SCT [25] seems to be more convenient for mode management by distinguishing process and specifications. With this approach, the authors of [26] have studied the problem to automatically find the restart state, after correction of an error. However, in these works, only the nominal mode of productivity is studied: the method does not assume that errors and/or corrective actions are included into the control. Our approach, quite the contrary, has the purpose to design all running modes of the system, including the actions of recuperation after a failure. Based on the works of [27] and [28], the proposed framework is able to take several modes into account. First, we define the model of the (controlled or not) and the models of specifications in a considered mode. This is an usual way in the industry to study and to design independently the mode between them. The models then

Manuscript received December 03, 2008; revised May 01, 2009. First published September 01, 2009; current version published November 06, 2009. Paper no. TII-08-12-0193.R1

The authors are with the AMPERE Laboratory, National Institute of Applied Science, University of Lyon, 69621 Villeurbanne cedex, Lyon, France (e-mail: gregory.faraut@insa-lyon.fr; laurent.pietrac@insa-lyon.fr; eric.niel@insa-lyon.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2009.2028135

are extended to include the mode switching [29]. The significant contribution, as presented in this article, is the identification of incompatibility for the mode switching, and the inconsistency (several states are possible to be reached by one commutation, meaning a lost of information) of the specifications. The proposed framework thus helps the designer throughout the design of control law of modes.

In Section III, we present a framework based on SCT devoted to functioning mode management. The proposed framework aims to design each functioning mode independently, and then simplify design and interpretation, considering for the discussed mode only the engaged components and associated requirements of that particular mode. The approach is illustrated on a conventional example in Section IV and is compared with the centralized approach. However, to give a better understanding of the proposition, Section II recalls basic notions of SCT.

II. Supervisory Control Theory

Ramadge and Wonham's theory [25] underpins the study of DES control. This theory is based on separation between the model representing what the system **can do** (the uncontrolled process), the model of what the system **must or must not do** (the liveness and safety properties of the process), the model of what the system **does** (the controlled process), and the model of what the system **should do** (the desired language).

The automaton $G = (Q, \Sigma, \delta, q_0, Q_m)$ models the uncontrolled process [30], with Q the finite-state set, Σ the finite alphabet of symbols (event labels), $\delta : Q \times \Sigma \rightarrow Q$ the partial transition function, q_0 the initial state, and $Q_m \subseteq Q$ the set of marked states. States exist for periods of time (duration), whereas events occur instantaneously, asynchronously, and at virtually random (unpredictable) times. For a machine, examples of states are "idle," "operating," "broken down," and "under repair." Examples of events are "machine starts to work," "breaks down," "completes work," or "start to repair." Marked states are used to model ends of tasks, states to be reached or states in which the system can be stopped.

Let Σ^* be the set of all finite sequences or strings of events in Σ, including the empty string ϵ. The function δ is extended to $\delta : Q \times \Sigma^* \rightarrow Q$. Any subset of Σ^* is called a language over Σ. The languages associated with G are the closed behavior $L(G) = \{s \in \Sigma^* | \delta(q_0, s) \text{ is defined}\}$ and the marked behavior $L_m(G) = \{s \in \Sigma^* | \delta(q_0, s) \in Q_m\}$. $L(G)$ represents the set of all possible trajectories, i.e., all possible system behaviors, whereas $L_m(G)$ represents the subset of trajectories leading to a marked state.

Let us assume that automaton G models the uncontrolled behavior of the process. This behavior is not satisfactory and must be restricted to a subset of $L(G)$ [31]. Let the specification represented by an automaton $E = (X, \Sigma, \xi, x_0, X_m)$ with X the state set, Σ the same alphabet as in G, x_0 the initial state and X_m the set of marked states. This specification models the liveness and safety requirements of the process. The objective is to adjoin a supervisor, denoted by S, to interact with G. To do this, the alphabet Σ is partitioned into two disjoint subsets Σ_c and Σ_{uc}, which comprise controllable and uncontrollable events, respectively. The controllable events are the events that can be prevented from happening by supervisor S, the uncontrollable events cannot. Formally, the supervisor S is a function from the language generated by G to the power set (the set of all subsets) of Σ: $S : L(G) \rightarrow 2^{\Sigma}$. Our goal is to find a controlled process, modeled by an automaton $H = (Y, \Sigma, \tau, y_0, Y_m)$, such that:

- The marked language of H is included in that of G: $L_m(H) \subseteq L_m(G)$.
- This controlled process satisfies the specification: $L_m(H) \subseteq L_m(G \times E)$.
- This controlled process is controllable, i.e., a supervisor S such that $L(S/G) = L(H)$ exists.

If $L(G \times E)$ is controllable with respect to $L(G)$, then $H = G \times E$. Else, we can determine [32] the automaton that generate the largest controllable sublanguage of $L(G \times E)$, called "supremal controllable sublanguage." If this automaton exists, the controlled process is nonblocking and minimally restrictive.

III. Multimodal Design

A. Overview

In this paper, a multimodal standpoint with a representation by multimodel approach is adopted to design a system. This system is composed by different and numerous components (plants, actuators, sensors, etc.) activated to achieve tasks in accordance to functional and safe requirements. This system can also be critical so that it needs to have a high availability even if one of these components fails. This is only possible if a number of alternative components are available. These available components are capable of replacing the failed components and keeping the same quality of production.

The considered mechanism will be implemented for systems which can operate in one single mode (production, initialization, etc.) once. In automatic control within mode switching ability, our contribution comprises proposing a framework, in which:

- Each mode is studied independently and separately. In most cases, a mode is characterized by the components used, generating events involving a commutation, and by a set of requirements, modeled independently of the requirements of the others modes, having to be fulfilled when the system is working in this mode. In this independent study, the SCT theory is applied "conventionally" in each mode (limited to the centralized control structure in this paper).
- The intermodal framework includes the intermodal specifications, used to extend the behavior of modes and having all possible trajectories, and the switch specifications, used to limit the commutation phenomena, i.e., forbid undesired switch trajectories. Of course, this will depend on whether switch events can be observed and controlled or not.

The main problem is to determine the state of the models (process, controlled process and specification), when a mode of the system has to leave the initial mode to commute to another one, called the final mode. In fact, the commutation is possible if the initial mode is in a compatible state with the final mode. Compatible means the state of the shared components between initial and final mode are the same. It also means the requirements existing in both modes are still respected even if the system switches modes.

The following sections describe each of the steps necessary in order to build, in a formal way, the final control law for each mode respecting requirements. To do so, we successively explain the intramodal framework—giving the internal behavior of each mode—and the intermodal framework allowing to check the connection between modes and identifying which ones of them are allowed or forbidden to finally obtain the control law.

To give a better understanding, Table II, shown at the end of this paper, refers to the notation used.

B. Definitions

A system is composed of different components. The dynamic of each component is the same regardless of the system mode. These dynamics include possible failures and recoveries. Such events will be used to model switching modes.

Definition 1: A set of components is denoted by $\mathcal{C} = \{C_1, C_2, \ldots, C_i\}$, where $i \in \mathbb{N}$ and $i \geq 1$. A component C_i is modeled by an automaton G^{C_i} where $G^{C_i} = (Q^{C_i}, \Sigma^{C_i}, \delta^{C_i}, q_0^{C_i}, Q_m^{C_i})$, with:

- Q^{C_i} is the state set of the component C_i;
- Σ^{C_i} is the event set of the component C_i, including two partitions:
  — $\Sigma^{C_i} = \Sigma_c^{C_i} \cup \Sigma_{uc}^{C_i}$ with $\Sigma_c^{C_i} \cap \Sigma_{uc}^{C_i} = \emptyset$. Σ_c and Σ_{uc} are, respectively, the controllable and uncontrollable events of the component C_i.
  — $\Sigma^{C_i} = \Sigma_{\rightleftarrows}^{C_i} \cup \Sigma_{\circlearrowleft}^{C_i}$ with $\Sigma_{\rightleftarrows}^{C_i} \cap \Sigma_{\circlearrowleft}^{C_i} = \emptyset$ $\Sigma_{\rightleftarrows}^{C_i}$ is the set of switch events. $\Sigma_{\circlearrowleft}^{C_i}$ are the other events.
- δ^{C_i} is the transition function and includes $\delta_{\rightleftarrows}^{C_i}$ which represents the set of switch transitions;
- $q_0^{C_i}$ is the initial state of the component C_i;
- $Q_m^{C_i}$ is the marked states set of the component C_i.

♦

Definition 2: A set of modes is denoted by $\mathcal{M} = \{M_1, M_2, \ldots, M_n\}$, where $n \in \mathbb{N}$ and $n \geq 1$ (by convention, we assume the initial active mode is M_1). We define $\mathcal{C}^{M_j}$ as the set of components used in the mode M_j, where $\mathcal{C}^{M_j} = \mathcal{C}_{\circlearrowleft}^{M_j} \cup \mathcal{C}_{\leftarrow}^{M_j} \cup \mathcal{C}_{\rightarrow}^{M_j}$ such that:

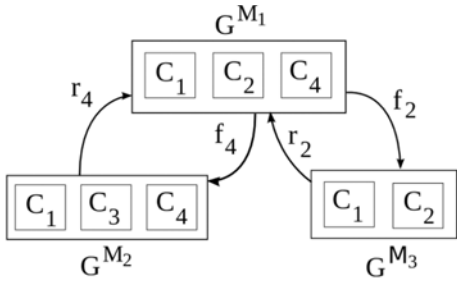
- $\mathcal{C}_{\circlearrowleft}^{M_j}$ is the set of components representing the intramodal behavior of **the process in** the mode M_j;
- $\mathcal{C}_{\leftarrow}^{M_j}$ is the set of components that lead the system to enter into the mode M_j;
- $\mathcal{C}_{\rightarrow}^{M_j}$ is the set of components that lead the system to exit mode M_j;
- $\mathcal{C}_{\rightleftarrows}^{M_j} = \mathcal{C}_{\leftarrow}^{M_j} \cup \mathcal{C}_{\rightarrow}^{M_j}$ is the set of switch components.

♦

No particular relation is assumed to exist between $\mathcal{C}_{\circlearrowleft}^{M_j}, \mathcal{C}_{\leftarrow}^{M_j}$ and $\mathcal{C}_{\rightarrow}^{M_j}$ except that they are all included in $\mathcal{C}$: in particular, a component can be included in:

- $\mathcal{C}_{\circlearrowleft}^{M_j}$, but not to be a switch component of $\mathcal{C}_{\rightleftarrows}^{M_j}$.
- $\mathcal{C}_{\circlearrowleft}^{M_j}$ and in $\mathcal{C}_{\leftarrow}^{M_j}$. It means this component is used in the mode and is necessary to enter into this mode.
- $\mathcal{C}_{\leftarrow}^{M_j}$ or $\mathcal{C}_{\rightarrow}^{M_j}$. It means this component is necessary to represent the switch behavior of the mode M_j (enter or exit).

Fig. 1 is an example of switching modes. We have three modes, M_1, M_2 and M_3. The components C_1, C_2 and C_4 are used in mode M_1. From this nominal mode, a switch is possible to the degraded mode M_2, by the switch event f_4 generated if the component C_4 breaks down, or to M_3, by the switch event f_2 if it is the component C_2 that breaks down. Thus,



$\mathcal{M} = \{M_1, M_2, M_3\}$
$\mathcal{C} = \{C_1, C_2, C_3, C_4\}$
$\mathcal{C}^{M_1} = \{C_1, C_2, C_4\} = \mathcal{C}_{\circlearrowleft}^{M_1}$
$\mathcal{C}_{\rightleftarrows}^{M_1} = \{C_2, C_4\}$
$\mathcal{C}^{M_2} = \{C_1, C_3, C_4\}$
$\mathcal{C}_{\circlearrowleft}^{M_2} = \{C_1, C_3\}$ and $\mathcal{C}_{\rightleftarrows}^{M_2} = \{C_4\}$
$\mathcal{C}^{M_3} = \{C_1, C_2\}$
$\mathcal{C}_{\circlearrowleft}^{M_3} = \{C_1\}$ and $\mathcal{C}_{\rightleftarrows}^{M_3} = \{C_2\}$

Fig. 1. Example of mode decomposition.

the mode M_3 is composed of component C_1 representing its internal behavior, i.e., $\mathcal{C}_{\circlearrowleft}^{M_3} = \{C_1\}$ and of the component $C_2(\mathcal{C}_{\leftarrow}^{M_3} = \mathcal{C}_{\rightarrow}^{M_3} = \{C_2\})$ because it is this component that generates the event leading to a switch from the mode M_1 to M_3. In the same way, the mode M_2 is composed of the components C_1 and $C_3(\mathcal{C}_{\circlearrowleft}^{M_3} = \{C_1, C_3\})$, but also with the components $C_4(\mathcal{C}_{\leftarrow}^{M_2} = \mathcal{C}_{\rightarrow}^{M_2} = \{C_4\})$ that generates the event r_4 responsible for the switch.

Definition 3: Let a mode automaton $G^{\mathcal{M}}$ representing the switch behavior of the system, described in requirements. Formally, this automaton is denoted by $G^{\mathcal{M}} = (Q^{\mathcal{M}}, \Sigma^{\mathcal{M}}, \delta^{\mathcal{M}}, q_0^{\mathcal{M}}, Q_m^{\mathcal{M}})$ such that:

- $Q^{\mathcal{M}} = \mathcal{M}$;
- $\Sigma^{\mathcal{M}} = \bigcup_{C_i \in \mathcal{C}} \Sigma_{\rightleftarrows}^{C_i}$;
- $\delta^{\mathcal{M}} : \mathcal{M} \times \Sigma^{\mathcal{M}} \to \mathcal{M}$ is the transition function of mode automaton;
- $q_0^{\mathcal{M}} = M_1$;
- $Q_m^{\mathcal{M}} \subseteq \mathcal{M}$.

♦

This automaton aims to easily add information on which mode the system is in. It also allows the addition of strategies to switch by modifying it.

C. Intramodal Design

The intramodal design, illustrated in Fig. 2, is very similar to the supervisory control theory used to synthesize the control law [31]. The objective of this first framework, a subpart of the general proposed framework, is to ensure the internal behavior enforcing the intramodal specification is correct and that each mode is reliable, well-built and optimal according to the requirements (illustrated by the green book). For each mode M_j, the process $G_{\text{in}}^{M_j}$, the model representing the internal behavior (in) of the mode M_j, results from parallel composition [31] of automata G^{C_i}, models of components used in this mode. It is defined on $\Sigma_{\text{in}}^{M_j} = \bigcup_{C_i \in \mathcal{C}_{\circlearrowleft}^{M_j}} \Sigma^{C_i}$. The specification $E_{\text{in}}^{M_j}$, defined on the alphabet $\Sigma_{\text{in}}^{M_j}$, results from the product composition of the model of each specification E_{in}^{l,M_j} to be complied

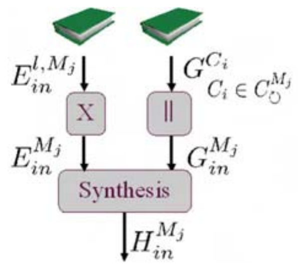


Fig. 2. Intramodal design framework.

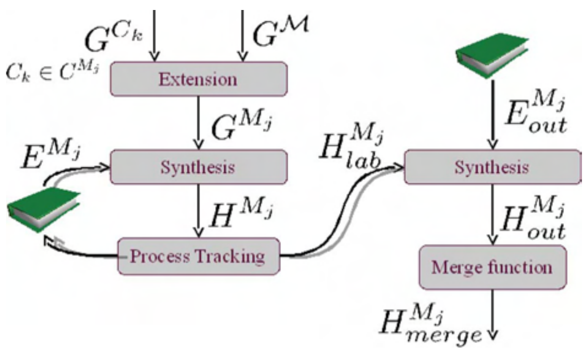


Fig. 3. Intermodal design framework.

with in this mode. After designing the required n modes, the designer obtains n uncontrolled processes $G_{\text{in}}^{M_j}$, n specifications $E_{\text{in}}^{M_j}$ and n controlled processes $S_{\text{in}}^{M_j}/G_{\text{in}}^{M_j}$ (denoted $H_{\text{in}}^{M_j}$).

Each model of $H_{\text{in}}^{M_j}$ represents the control law of the mode M_j. Nevertheless, these models are not interconnected, and this is the main focus of the next section.

D. Intermodal Design

The intramodal framework focuses only on components used to represent the internal behavior and the requirements having to be fulfilled for each mode. This framework ensures the existence of one control law and that the requirements are respected. The running of the system remains flawless as long as it operates within one of these modes. However, as previously said in the introduction, the commutation phenomena are a very particular problem in mode management. The second framework aims to handle these phenomena correctly. This section focuses on the intermodal behavior of modes, and takes the behavior that could lead to switching between modes into account. The proposed framework is shown in Fig. 3.

This new framework includes successive steps in order to identify all trajectories connecting modes and, in some cases, to forbid them. The first step extends the mode's behavior. The second one synthesizes these extended mode behaviors of mode by the extended intramodal specifications regarding these new dynamics resulted from the intermodal specifications. The next process tracking step identifies trajectories allow a switch from one mode to another. Concerning the steps process tracking (step three) and the merge function (step five), it is well-known that merging states in an automaton can cause nondeterminism [31]. The switch events are renamed in process tracking (Section III-D3) to avoid this. Thus, the knowledge about the switch events that produced them has been preserved. In other words, the merge function has been anticipated in renaming all identified switch events during the procedure in step three. Using both of these functions allows the reduction of complexity without generating a nondeterministic automata. A second synthesis considering switch specifications (step four) is realized to forbid the undesired trajectories. The final models resulting from the fifth step are the control law of mode.

1) Extension of Process: The controlled process in the intramodal framework is built by composition of the components used in each mode and the requirements that have to be respected when the system is running in one of these modes. Nevertheless, assuming that the internal behavior is totally represented, it is not necessarily the case for the external behavior, i.e., all possible commutations between modes could be not totally known. Some components are indeed not taken into consideration in the intramodal framework for a particular mode, but could be important from a switch standpoint. The extension then takes all the components that are necessary to represent the internal and external behavior of modes into account. Extended models G^{M_j} result in parallel composition of both included components G^{C_i} in $\mathcal{C}^{M_j}$ (and not only $\mathcal{C}_{\circlearrowleft}^{M_j}$), and the mode automaton $G^{\mathcal{M}}$.

Definition 4: Let G^{M_j} be such that

$$G^{M_j} = (Q^{M_j}, \Sigma^{M_j}, \delta^{M_j}, q_0^{M_j}, Q_m^{M_j})$$

where $G^{M_j} = G^{\mathcal{M}} \|_{C_k \in \mathcal{C}^{M_j}} G^{C_k}$

These models ensure that the whole behavior is represented and allow to detect all trajectories relying modes.

2) Synthesis With Extended Specification: The synthesis of the intramodal framework only deals with built specifications regarding taken components in the intramodal behavior of modes. In this step, we have to take the whole specification of the considered mode into account in order to fulfill the intermodal behavior of each mode. Thus, there are two types of specifications.

- Extended intramodal specifications. These specifications have to be extended according to the newly added components in the extension step. Indeed, some components do not represent the intramodal behavior of modes and may have an influence on intramodal specification. For this reason, the intramodal specification have to be extended. The specification E^{M_j} represents extended intramodal specification.
- Intermodal specifications represent specifications which are not necessary in the intramodal behavior, but can modify the trajectory to switch. These specifications have to be taken into account also.

The synthesis procedure is applied in the same way as the synthesis in intramodal framework. For each mode, we obtain the model H^{M_j} which represents both intramodal and intermodal behaviors and respects the requirements of the mode M_j.

3) Process Tracking: In fact, models H^{M_j} are too rich in states: they also contain some states that do not correspond to the internal behavior or to a commutation between modes. It is a consequence of the parallel composition of the component models G^{C_i} and the mode automaton $G^{\mathcal{M}}$. These states will be removed in the last step of the intermodal framework (merge function), that may also cause an indeterminism (several transitions with the same event from a single state). To avoid that, we will add an information on each event occurrence (a label).

It is possible only if a single state exists in the final controlled process H^{M_k} after a commutation from a single state in the initial controlled process H^{M_j}.

Formally, the procedure uses the next definitions to find the equivalent states between modes. However, the first definition is focused on subsets of Σ necessary for others following definitions.

Definition 5: $\Sigma_{\leftrightarrows}^{M_i} \subset \Sigma^{M_i}$ is the set of commutation events of the mode M_i: $\Sigma_{\leftrightarrows}^{M_i} = \bigcup_{n \in \mathcal{C}_{\leftrightarrows}^{M_i}} \Sigma_{\leftrightarrows}^{\mathcal{C}_n}$. $\Sigma_{\leftrightarrows}^{M_i} = \Sigma_{\leftarrow}^{M_i} \cup \Sigma_{\rightarrow}^{M_i}$, where $\Sigma_{\leftarrow}^{M_i}$ (resp. $\Sigma_{\rightarrow}^{M_i}$) is the event set that lead the system to enter into (resp. exit) the mode M_i. ♦

We define the states set $Y^{M_j}_{M_j \xrightarrow{\alpha} M_k}$ of H^{m_j}, where a switch event α exists and leads from the initial mode M_j to the final mode M_k.

Definition 6:

$$\begin{aligned} Y^{M_j}_{M_j \xrightarrow{\alpha} M_k} = \{ & y \in Y^{M_j} | M_j, \\ & M_k \in Q^{\mathcal{M}}, \alpha \in \Sigma^{\mathcal{M}} \cap \Sigma_{\rightarrow}^{M_j} \cap \Sigma_{\leftarrow}^{M_k}, \\ & \delta^{\mathcal{M}}(M_j, \alpha) = M_k, \tau^{M_j}(y, \alpha) \text{ is defined}\} \end{aligned}$$

♦

Let the language $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_j})$ be the sublanguage of $L(H^{M_j})$ leading from the initial state y_0 of H^{M_j} to a state y, where a switch event α can occur.

Definition 7:

$$L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_j}) = \left\{ s \in \Sigma^{M_j*} | y \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k}, \tau(y_0, s) = y \right\}$$

♦

The identification results in the extended projection function P_{M_j,M_k} defined as follows:

Definition 8: Let $P_{M_j,M_k} : \Sigma^{M_j*} \to \Sigma^{M_k*}$ such as $\forall \sigma \in \Sigma^{M_j}$ and $\forall s \in \Sigma^{M_j*}$

$$P_{M_j,M_k}(\varepsilon) = \varepsilon$$

$$P_{M_j,M_k}(s\sigma) = \begin{cases} P_{M_j,M_k}(s)\sigma, & \text{if } \sigma \in \Sigma^{M_j} \cap \Sigma^{M_k} \\ P_{M_j,M_k}(s), & \text{if } \sigma \in \Sigma^{M_j} \backslash \Sigma^{M_k} \end{cases}.$$

In other words, this function takes a language defined on alphabet Σ^{M_j} (representing the alphabet of the initial mode), and *erases* the events that are not included on alphabet Σ^{M_k} (representing the alphabet of final mode). More details are given in [27] and [33]. In the next definition, this projection is used on $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_j})$ to find the projected language on the alphabet of the final mode M_k.

Definition 9: Let an initial mode be called M_j and a final mode be called M_k. $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_j})$ is the language of H^{M_j} that leads to a state y, where a switch event α occurs. $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k})$ is the projected language on the alphabet of the final mode M_k, such that: $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) = P_{M_j,M_k}[L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_j})]$. ♦

Based on this definition, we can define two properties. First of all, if all languages $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k})$ of $y \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k}$ are in $L(H^{M_k})$, this means at least one connection state exists in $L(H^{M_k})$. In this case, we say H^{M_k} is compatible with H^{M_j} (this does not mean that H^{M_j} is compatible with H^{M_k}). It is not the case, specifications have to be modified.

Definition 10: H^{M_k} is compatible with H^{M_j} iff (if and only if): $\forall y \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k} (L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) \subseteq L(H^{M_k}))$. ♦

If H^{M_k} is compatible with H^{M_j}, it is possible that from a single state of H^{M_k}, several states of H^{M_j} are reachable. In this case, it is not possible to define a single state of connection. It is also possible that from several states of H^{M_j}, a single state of H^{M_k} is reached. Thus, an information contained in the model H^{M_j} has disappeared in the model H^{M_k}, which could lead to a problem when returning to the initial mode. If these two cases do not occur, there is no problem and all the switch events can be labeled with a subindex. We say H^{M_k} is consistent with H^{M_j} (this does not mean that H^{M_j} is consistent with H^{M_k}).

Definition 11: Let H^{M_k} be compatible with H^{M_j}. H^{M_k} is consistent with H^{M_j} iff:

- $\forall y_l \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k} \exists! y_k \in Y^{M_k}(L^{y_l}_{M_j \xrightarrow{\alpha} M_k}(H^{M_j}) \subseteq L^{y_k}(H^{M_k}))$ with $L^{y_k}(H^{M_k}) = \{s \in \Sigma^{M_k*} | y_k \in Y^{M_k}, \tau(y_0, s) = y_k\}$;
- $\forall y_1, y_2 \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k}(y_1 \neq y_2 \Leftrightarrow L^{y_1}_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) \cap L^{y_2}_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) = \emptyset)$.

♦

In the following procedure, we apply the above definitions to track trajectories representing commutations.

Procedure 1: For each $\delta^{\mathcal{M}}(M_j, \alpha) = M_k$ of the mode automaton $G^{\mathcal{M}}$:

1) For each $y \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k}$ [definition 6]:
   a) We calculate $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_j})$ [definition 7];
   b) We calculate $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k})$ [definition 9]. Two cases are possible.
      i) $(L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) \nsubseteq L(H^{M_k}))$. So, H^{M_k} is not compatible with H^{M_j} and we can stop the procedure for this transition [definition 10];
      ii) $(L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) \subseteq L(H^{M_k}))$: maybe H^{M_k} is compatible with H^{M_j}. Two cases are possible:
         A) $\exists y_1, y_2 \in Y^{M_k}[L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_j} \subseteq L^{y_1}(H^{M_k})$ and $L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_j}) \subseteq L^{y_2}(H^{M_k})]$ or $\exists y' \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k}(y \neq y' \Leftrightarrow L^y_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) \cap L^{y'}_{M_j \xrightarrow{\alpha} M_k}(H^{M_k}) \neq \emptyset)$: H^{M_k} is not consistent with H^{M_j} and we can stop the procedure for this transition [definition 11].
         B) else, maybe H^{M_k} is consistent with H^{M_j}. $\delta^{\mathcal{M}}(M_j, \alpha) = M_k$ is then considered as valid. The new name is operated on the transitions functions of H^{M_j} and H^{M_k} such as $\tau^{M_j}(y, \alpha)$ and $\tau^{M_k}(y', \alpha)$ exist and are changed by $\tau^{M_j}(y, \alpha_l)$ and $\tau^{M_k}(y', \alpha_l)$ with l a subindex. The alphabet of the newly built models is given by: $\Sigma^{M_j}_{\text{lab}} = \Sigma^{M_j} \cup \{\alpha_l\}$.

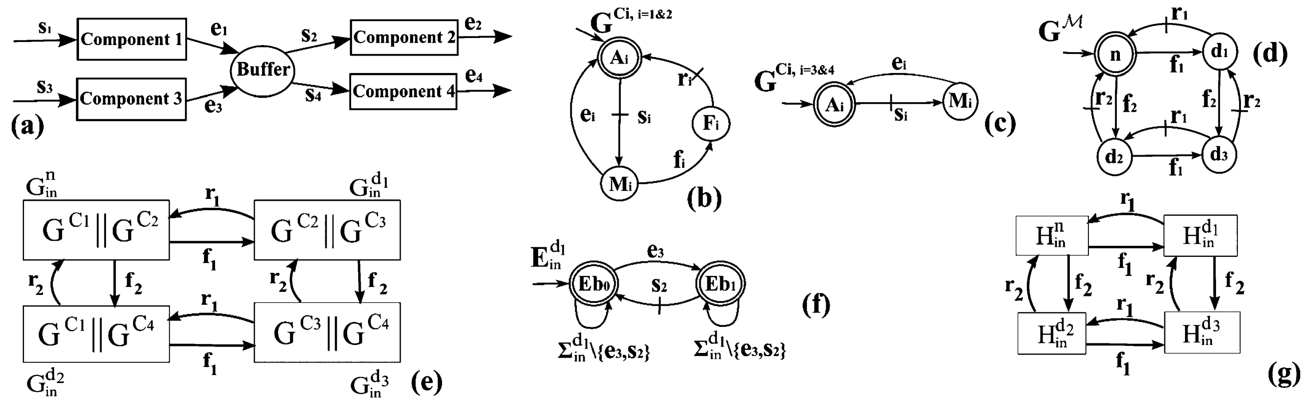


Fig. 4. Manufacturing system example: (a) the studied system; (b) models G^{C_i} for components C_1 and C_2; (c) model G^{C_i} for components C_3 and C_4; (d) the model of the mode automaton $G^{\mathcal{M}}$; (e) a modal decomposition standpoint of the system, with the components used in each mode; (f) the model of specification $E_{\text{in}}^{d_1}$ representing the behavior of the buffer in the nominal mode; and (g) a modal decomposition standpoint of the controlled processes for each mode.

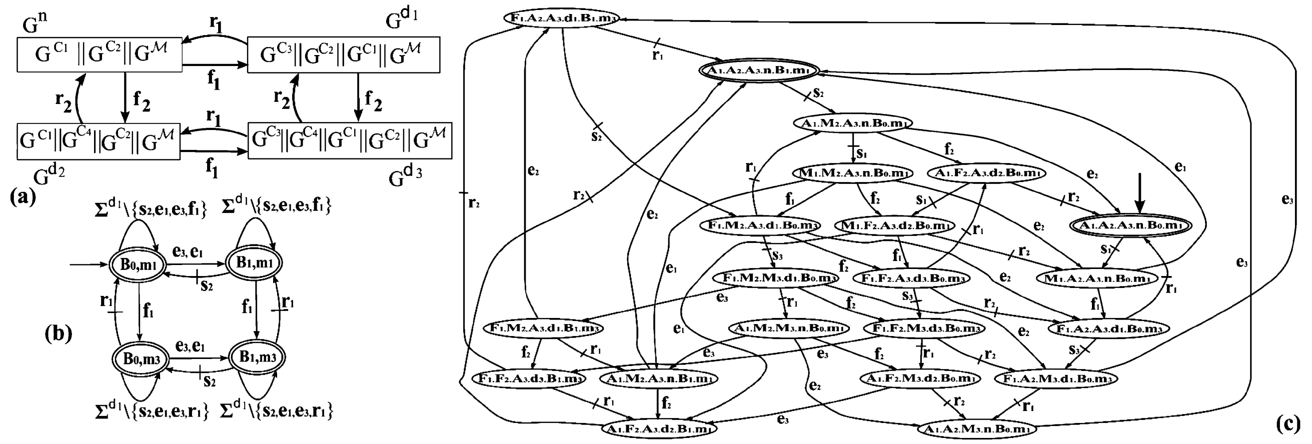


Fig. 5. Manufacturing system example: (a) a modal decomposition standpoint with the controlled processes extended; (b) the model E^{d_1} representing the extended specification for the degraded mode d_1; and (c) the model H^{d_1} representing the extended controlled process for the degraded mode d_1.

2) After the last $y \in Y^{M_j}_{M_j \xrightarrow{\alpha} M_k}$, if the procedure was not stopped before this step, H^{M_k} is consistent with H^{M_j}. We can go to the next transition $\delta^{\mathcal{M}}(M_j, \alpha') = M_k$. ♦

When the switch events detected by an incompatibility or by an inconsistency has been listed, there are two methods to solve it. Either we go in the loop of the framework intermodal illustrated in Fig. 3 to modify the intermodal specifications, Section III-D2, or we create a new switch specification to forbid the undesired switch events which have been detected. This step of switch specification is the next step in the intermodal framework.

4) Synthesis With Switch Specifications: The switch specifications step is about trajectories detected as undesired during the process tracking, and not modified by the intermodal specifications. Building these models of specifications is really easy, because we use the new switch events label as we did in the last Section III-D3. In other words, the language of switch specifications is defined on the alphabet $\Sigma_{\text{lab}}^{M_j}$ without the switch events we desire to forbid. Applying these specifications, represented by the models $E_{\text{out}}^{M_j}$, gives the new models $H_{\text{out}}^{M_j}$ At the end of this step, the models are under control and no model of the modes has more than one switch event with the same label, the undesired switch events have been forbidden and there is only one other switch event in another mode that has the same label for each switch event. The model of modes can now be reduced by using a merge function.

5) Merge Function: The merge function reduces the complexity of the model by keeping only the intramodal behavior of each mode and including the useful intermodal behavior. To obtain the smallest size of each mode, the mode automaton $G^{\mathcal{M}}$ used during the extension step is used one more time. Each state of mode then has a name including the state where $G^{\mathcal{M}}$ is. In other words, we know for each state the mode in which the system is in. As we are now only interested in the intramodal behavior of each mode, we just have to *remove* the behavior which does not represent the intramodal behavior and add an *idle* state representing the mode when it will be deactivated to meet the specification that only one mode can be active in the same time. To do this, we execute the next procedure.

Procedure 2: Let $H_{\text{out}}^{M_j}$ and $H_{\text{merge}}^{M_j}$ be automata, where $H_{\text{merge}}^{M_k}$ is the model reduced by the merge function of the model $H_{\text{out}}^{M_j}$. The procedure to merge states is described next.

1) We determine in $H_{\text{out}}^{M_j}$, a merge set $Y_{\text{mer}}^{M_j} \subset Y_{\text{out}}^{M_j}$. The states included in $Y_{\text{mer}}^{M_j}$ are the insignificant states to the

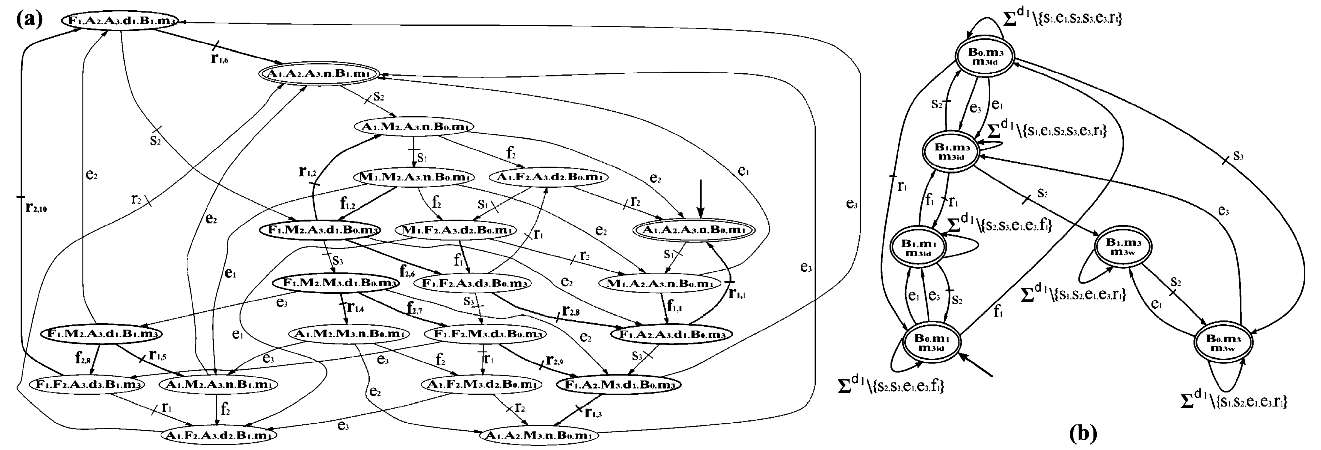


Fig. 6. Manufacturing system example: (a) the model $H_{\text{lab}}^{d_1}$ representing the extended controlled process of the degraded mode d_1 and including the label of switch events and (b) the new model E^{d_1} (extended specification of degraded mode d_1) including the missing requirement to solve the incompleteness.

mode. Insignificant state to the mode M_j are all states which do not have M_j as part of their name (this part was given by the mode automaton $G^{\mathcal{M}}$). These states are certainly added during the extension of the different models.

2) All states in $Y_{\text{mer}}^{M_j}$ are replaced by one new state called $y_{id}^{M_j}$.
3) We remove all self-loops at $y_{id}^{M_j}$.
4) If the initial state is included in $Y_{\text{mer}}^{M_j}$, then $y_{id}^{M_j}$ is the new initial state.
5) If a marked state is included in $Y_{\text{mer}}^{M_j}$, then $y_{id}^{M_j}$ is a marked state. ♦

This procedure results in models which only represent the internal behavior of each mode and an idle state needed for restricting it to one active mode at a time. Formally, the automaton $H_{\text{merge}}^{M_j}$ is defined as follows:

Definition 12: Let $Y_{\text{mer}}^{M_j}$ be the set of all states which do not have M_j as part of their name.

$H_{\text{merge}}^{M_j} = (Y_{\text{merge}}^{M_j}, \Sigma_{\text{merge}}^{M_j}, \tau_{\text{merge}}^{M_j}, y_{\text{merge},0}^{M_j}, Y_{\text{merge},m}^{M_j})$ such that:

- $Y_{\text{merge}}^{M_j} = (Y_{\text{out}}^{M_j} \backslash Y_{\text{mer}}^{M_j}) \cup \{y_{id}^{M_j}\}$.
- $\Sigma_{\text{merge}}^{M_j} = \Sigma_{\text{out}}^{M_j} \backslash \bigcup_{C_i \in (\mathcal{C}_{\rightleftarrows}^{M_j} \backslash \mathcal{C}_{\circlearrowleft}^{M_j})} \Sigma_{\circlearrowleft}^{C_i}$.
- $\tau_{\text{merge}}^{M_j} = \tau_{\text{out}}^{M_j} \backslash \{(y_a, s, y_b) | s \in \Sigma_{\text{out}}^{M_j}, y_a, y_b \in Y_{\text{mer}}^{M_j}, \tau_{\text{out}}^{M_j}(y_a, s) = y_b\}$.
- $$y_{\text{merge},0}^{M_j} = \begin{cases} y_{\text{out},0}^{M_j}, & \text{if } y_{\text{out},0}^{M_j} \notin Y_{\text{mer}}^{M_j} \\ y_{id}^{M_j}, & \text{if } y_{\text{out},0}^{M_j} \in Y_{\text{mer}}^{M_j} \end{cases}.$$
- $$Y_{\text{merge},m}^{M_j} = \begin{cases} Y_{\text{out},m}^{M_j}, & \text{if } Y_{\text{out},m}^{M_j} \cap Y_{\text{mer}}^{M_j} = \emptyset \\ Y_{\text{out},m}^{M_j} \backslash Y_{\text{mer}}^{M_j} \cup \left\{ y_{id}^{M_j} \right\}, & \text{if not.} \end{cases}$$

♦

The aim of the steps process tracking and merge function are to avoid the nondeterminism involving the reduction complexity of models of mode. Between the last two steps, we can use the synthesis (step four) again to forbid the undesired trajectories. The final models resulted in the fifth step are the control law of mode.

IV. Example

A. Requirements

Consider the manufacturing system illustrated in Fig. 4(a), the system comprises four components and one buffer. The components are used to process a part and the buffer is used as a storage between the components with a maximal capacity of 1. The components C_i are modeled by the automaton denoted G^{C_i} and are shown Figs. 4(b) and (c). The events s_i and e_i represent a new task and the end of the task, respectively. While all these events are observable, events s_i and r_i are controllable and e_i and f_i are not. The system has four modes, such as $\mathcal{M} = \{n, d_1, d_2, d_3\}$. Fig. 4(d) shows the model of the mode automaton $G^{\mathcal{M}}$, which represents the switch behavior between modes. The first one, which is the initial mode, is the nominal mode n. The other modes are the degraded modes d_1, d_2 and d_3 which, respectively, depend on whether the component C_1, C_2 or C_1 and C_2 will fail. In the case of malfunction, the component C_1 is replaced by the component C_3 and the component C_2 by the component C_4. This malfunction is modeled with the event f_i while the repairing is modeled with the event r_i. So we have: $\mathcal{C}_{\circlearrowleft}^n = \mathcal{C}_{\leftarrow}^n = \mathcal{C}_{\rightarrow}^n = \{C_1, C_2\}, \mathcal{C}_{\circlearrowleft}^{d_1} = \{C_2, C_3\}, \mathcal{C}_{\leftarrow}^{d_1} = \mathcal{C}_{\rightarrow}^{d_1} = \{C_1, C_2\}, \mathcal{C}_{\circlearrowleft}^{d_2} = \{C_1, C_4\}, \mathcal{C}_{\leftarrow}^{d_2} = \mathcal{C}_{\rightarrow}^{d_2} = \{C_1, C_2\}, \mathcal{C}_{\circlearrowleft}^{d_3} = \{C_3, C_4\}, \mathcal{C}_{\leftarrow}^{d_3} = \mathcal{C}_{\rightarrow}^{d_3} = \{C_1, C_2\}$.

B. Intramodal Design

The modal decomposition of the system, where modes use components necessary to run, is modeled in Fig. 4(e). These models must be controlled according to the requirements defined by a specification automaton, like the model of the nominal specification $E_{\text{in}}^{d_1}$ representing the requirement of the buffer, shown in Fig. 4(f). It is unlikely that $\mathrm{L}(G_{\text{in}}^{M_j} \times E_{\text{in}}^{M_j})$ is not controllable with respect to $\mathrm{L}(G_{\text{in}}^{M_j})$, thus we use the supremal controllable sublanguage of $\mathrm{L}(H_{\text{in}}^{M_j})$ to control our four modes, as illustrated in Fig. 4(g).

C. Intermodal Design

The first step is an extension of each mode with the components not already included in the internal behavior but which are

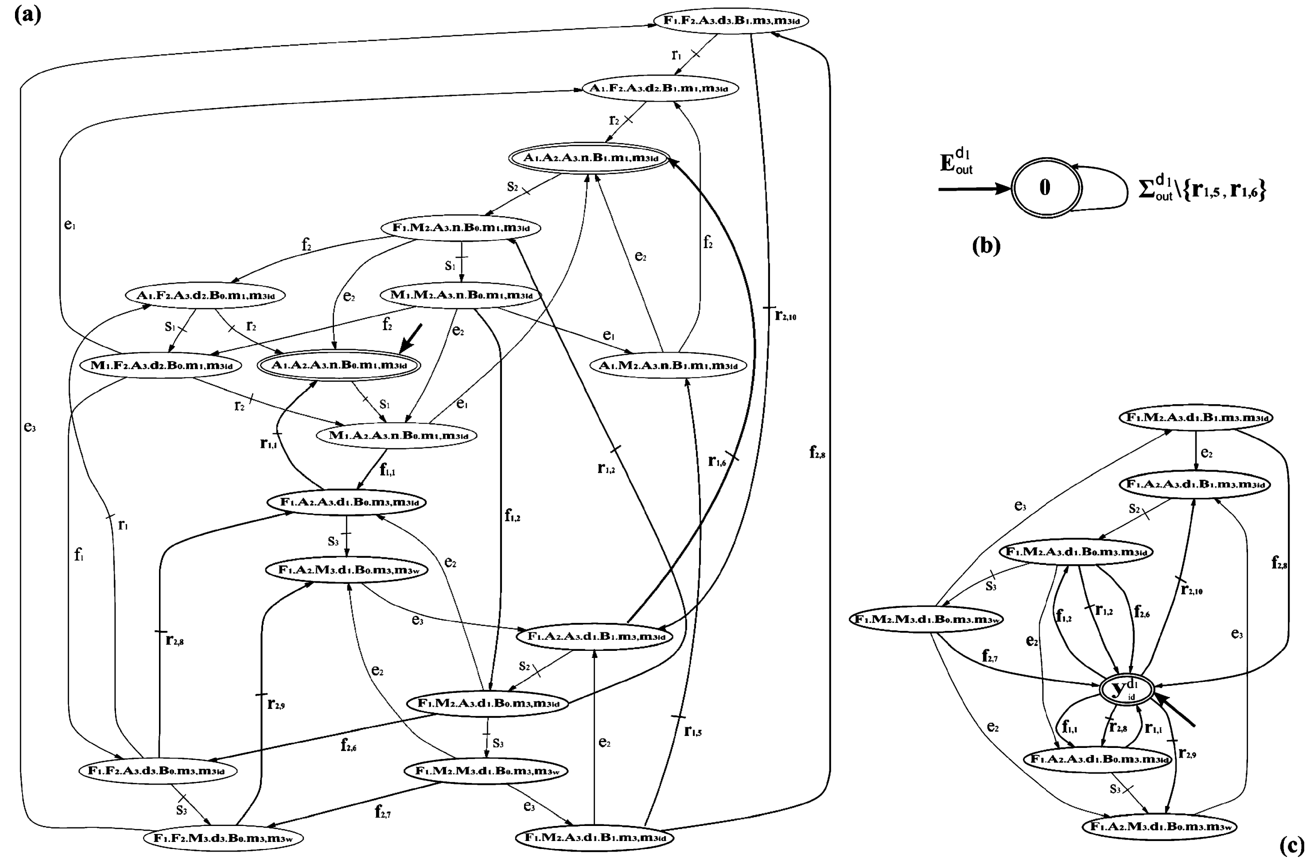


Fig. 7. Manufacturing system example: (a) the new model $H_{\mathrm{lab}}^{d_1}$ (extended controlled process of degraded mode d_1) including the new model E^{d_1} (modified intermodal specifications); (b) the model $E_{\mathrm{out}}^{d_1}$ (switch Specification) to forbid undesired commutations $r_{1,5}, r_{1,6}$; and (c) the merged model $H_{\mathrm{merge}}^{d_1}$, of degraded mode d_1, representing the final control law for this mode.

necessary to represent the switch behavior. This extension is illustrated Fig. 5(a) by a modal decomposition standpoint. The model E^{d_1} representing the extended specification for the degraded mode d_1 is represented in Fig. 5(b). It includes the extended intramodal specification that is the buffer but expressed on the alphabet Σ^{d_1} (and not $\Sigma_{\mathrm{in}}^{d_1}$ like in the intramodal design). This specification also includes an intermodal requirement—the activation or the deactivation of the machines 1 and 3 following if the system is in the mode nominal n or degraded d_1. This second specification was not present in the intramodal design because it has no effect on the internal behavior. The extended controlled process H^{d_1} of the degraded mode d_1 is illustrated in Fig. 5(c). The next step is the process tracking to characterize the switch events and identify those to imply a deadlock.

Fig. 6(a) represents the model $H_{\mathrm{lab}}^{d_1}$ for the degraded mode d_1 and is the result of the process tracking step. At the end of this step, some problems are identified throughout the switch events sets $r_{1,1}, r_{1,3}, r_{1,6}$ and $r_{1,2}, r_{1,4}, r_{1,6}$. In each set, the projection of their language results in the same switch event in the final mode. This is an incompleteness between requirements, i.e., it misses information in the requirements. This information is about when the system can repair the machine 1 with regards to machine 3. This missing requirement has to be added by modifying the model of the intermodal specification. The modified intramodal specification for the degraded mode d_1 is shown Fig. 6(b).

The intermodal specification once modified and the process tracking step reused, only events $r_{1,5}, r_{1,6}$ remain problematic, as illustrated in Fig. 7(a), by the new model H^{d_1}, which includes the added requirements. Nevertheless, these requirements do not solve the incompleteness for these events. A closer look showed that the incompleteness due to these events is caused by the fact that the final mode does not have the behavior of machine 3 and, thus, does not know the buffer can be reduced by this machine 3. This may sound strange since the state in final mode exists, but the connection allowing the commutation does not. The easiest way for the designer to correct this incompleteness is to forbid these events in the model of the switch specification, as displayed Fig. 7(b), representing the model $E_{\mathrm{out}}^{d_1}$. Thanks to rename command in the process tracking step, it is really easy to design the model of the switch specifications. The last figure, as displayed in Fig. 7(c), represents the model $H_{\mathrm{merge}}^{d_1}$ after the application of the merge function for the degraded mode d_1. This merge function merges all states that do not have d_1 as part of their name. This merge results a new *idle* state, which is the new initial state for degraded modes. The models $H_{\mathrm{merge}}^{M_j}$ are the final control law for each mode and ensure a reliable commutation between modes.

D. Comparison

In this section, the classical centralized approach is compared with our proposed multimodel approach. The Table I, gives the

TABLE I
COMPARISON BETWEEN THE APPROACHES

		uncontrolled process	specification	controlled process
centralized approach		$G: 36, 12, 168$	$E: 18, 12, 141$	$H: 24, 12, 56$
multi-model approach	nominal mode	$G^n_{in}: 9, 8, 24$ $G^n = G^n_{in}$	$E^n_{in}: 2, 8, 14$ $E^n = E^n_{in}$	$H^n_{in}: 12, 8, 25$ $H^n = H^n_{in}$ $H^n_{merge}: 7, 15, 19$
	degraded mode d_1	$G^{d_1}_{in}: 6, 6, 14$ $G^{d_1}: 18, 10, 66$	$E^{d_1}_{in}: 2, 6, 10$ $E^{d_1}: 6, 10, 43$	$H^{d_1}_{in}: 9, 6, 16$ $H^{d_1}: 18, 10, 41$ $H^{d_1}_{merge}: 7, 14, 18$
	degraded mode d_2	$G^{d_2}_{in}: 6, 6, 14$ $G^{d_2}: 18, 10, 66$	$E^{d_2}_{in}: 2, 6, 10$ $E^{d_2}: 4, 10, 32$	$H^{d_2}_{in}: 8, 6, 13$ $H^{d_2}: 16, 10, 34$ $H^{d_2}_{merge}: 7, 14, 18$
	degraded mode d_3	$G^{d_3}_{in}: 4, 4, 8$ $G^{d_3} = G$	$E^{d_3}_{in}: 2, 4, 6$ $E^{d_3} = E$	$H^{d_3}_{in}: 6, 4, 8$ $H^{d_3} = H$ $H^{d_3}_{merge}: 7, 14, 18$

TABLE II
NOTATION USED IN PAPER

Notation	Meaning
$\mathcal{C}$	Set of components C_i, where $i \in \mathbb{N}$ and $i \geq 1$
$\mathcal{M}$	Set of modes M_j. Initially the active mode is the mode M_1
$\mathcal{C}^{M_j}$	Set of components used in the mode M_j, with the set of components representing the internal behavior ($\mathcal{C}^{M_j}_{\circlearrowleft}$), and the sets of components necessary to enter into ($\mathcal{C}^{M_j}_{\leftarrow}$) or to exit from ($\mathcal{C}^{M_j}_{\rightarrow}$) this mode
G^{C_i}	Model of the component C_i
$G^{M_j}_{in}$	Model of the intramodal uncontrolled process in the mode M_j
E^{l,M_j}_{in}	Model of an intramodal specification l in the mode M_j
$E^{M_j}_{in}$	Model of the intramodal specification in the mode M_j
$H^{M_j}_{in}$	Model of the intramodal controlled process in the mode M_j
$G^{\mathcal{M}}$	Mode automaton representing the switch behavior of the system as described in requirements
G^{M_j}	Extended model of the uncontrolled process in the mode M_j
E^{l,M_j}	Extended model of a specification l in the mode M_j
E^{M_j}	Extended specification in the mode M_j
H^{M_j}	Extended model of controlled process in the mode M_j
$H^{M_j}_{lab}$	Labeled model of process under control in the mode M_j (after the process tracking step)
$E^{M_j}_{out}$	Model of the switch specification in the mode M_j
$H^{M_j}_{out}$	Model of controlled process in the mode M_j, after the integration of switch specification
$H^{M_j}_{merge}$	Merged controlled process in the mode M_j (final model)

size of the different models. Each model has three numbers, meaning, respectively, the number of states (in the set of states), the number of events (in the set of events), and the number of transition in the automaton.

Through the intramodal study, Table I shows the different models are much smaller than the models built in centralized approach. The number of transitions is by example divided by a value comprised between 2 and 7 depending on the mode considered. This reduction gives an easier way for the designer to perform a good interpretation of these models, which is one of the aims of this work.

During the intermodal study, the last models of the controlled process, i.e., the models $H^{M_j}_{\text{merge}}$, are approximately three times smaller that their equivalent models H, built in the centralized approach, when we compare the number of states and the number of transitions. The number of events is a little bit bigger due to the process tracking step when some switch events are labeled. A comparison on the built models through the intermodal framework (the models G^{M_j}, E^{M_j} et H^{M_j}) show they are a similar size, but with a reduced complexity because the number of transitions is halved. A negative point is the study of the degraded mode d_3. Indeed, in this mode, all components are considered to build the (uncontrolled) process. This is the worst case that could happen for the framework, because the model of the process in the mode d_3 is the same as the process in the centralized approach: $G^{d_3} = G$. This also is the case for the models of specifications ($E^{d_3} = E$), and the model of controlled process: $H^{d_3} = H$. This is because in our example the single specification E is also the single specification in mode d_3. This is also the worst case for specifications.

Are these models necessary? The models of the intramodal framework allow the designer, in charge of the synthesis, to discuss with the users of the studied system. These models have to be the most clear that it is possible to reach, and this is clearly the case for the framework. The models G^{M_j}, E^{M_j} and H^{M_j} built in the intermodal study are only used by the designer that do not work, at this moment, with models simpler than in the centralized approach. On the contrary, his work is more complicated because he has to use more models. In the same way, the final models ($H^{M_j}_{\text{merge}}$) are intended to the communication between the designer and the users, and then have to be as simplest as possible, that effectively the case with this framework. With these results, we think our objectives about the simplification of the models, to be easier to analyze, are for a big part reached.

V. Conclusion

The main contribution of this paper is to present an advanced framework using a multimodal standpoint and allowing to design a system by multimodel approach. The multimodal standpoint is an usual way in industry to design a system. The proposed framework uses a multimodel approach allowing to decompose the system in numerous control laws (one by mode). The first step of the framework is an intramodal study where each mode is studied independently. The second step, and the major contribution of this work, focuses on formal way to design complete modes including its different switch dynamics, to identify some incompatibilities during the mode switching and forbid these incompatibilities by using SCT. Being an offline study, this work gives a formal method to design a control law from scratch while meeting all requirements the system needs. Current research involves defining strategies when incompatible states have been recognized using uncontrollable switch events and when the supremal controllable does not give satisfaction on the set of requirements.

References

[1] W. M. Wonham and P. J. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM J. Control and Optim.*, vol. 25, no. 3, pp. 637–659, 1987.

[2] M. Nourelfath and E. Niel, "Modular supervisory control of an experimental manufacturing system," *Control Engineering Practice*, 2003.

[3] J. Komenda, J. van Schuppen, B. Gaudin, and H. Marchand, "Supervisory control of modular systems with global specification languages," *Automatica* vol. 44, no. 4, pp. 1127–1134, Apr. 2008 [Online]. Available: http://www.sciencedirect.com/science/article/B6V21-4RD9FJ3-4/2/6ada02eeaf3c8569 c7ea747b08c09f3a

[4] S. Jiang and R. Kumar, "Decentralized control of discrete event systems with specializations to local control and concurrent systems," *IEEE Trans. Syst., Man, Cybern., Part B*, vol. 30, no. 5, pp. 653–660, Oct. 2000.

[5] G. Barrett and S. Lafortune, "Decentralized supervisory control with communicating controllers," *IEEE Trans. Autom. Control*, vol. 45, no. 9, pp. 1620–1638, 2000.

[6] Z. Chao and Y. Xi, "Necessary conditions for control consistency in hierarchical control of discrete-event systems," *IEEE Trans. Autom. Control*, vol. 48, pp. 465–468, Mar. 2003.

[7] R. J. Leduc, B. A. Brandin, M. Lawford, and W. M. Wonham, "Hierarchical interface-based supervisory control—Part i: Serial case," *IEEE Trans. Autom. Control*, vol. 50, no. 9, pp. 1322–1335, Sept. 2005.

[8] S. Chafik and E. Niel, "Hierarchical-decentralized solution of supervisory control," in *Proc. 3rd Int. Symp. Math. Modelling, MATHMOD*, Vienna, Austria, 2000, vol. 2, pp. 787–790.

[9] K. Anderson, J. Richardson, B. Lennartson, and M. Fabian, "Synthesis of hierarchical and distributed control functions for multi-product manufacturing cells," in *Proc. IEEE Int. Conf. Autom. Sci. Eng., CASE'06*, Oct. 2006, pp. 325–330.

[10] S. Balemi, G. J. Hoffman, G. P. , H. Wong-Toi, and F. G. F. , "Supervisory control of a rapid thermal multiprocessor," *IEEE Trans. Autom. Control*, vol. 38, no. 7, pp. 1040–1059, 1993.

[11] Ansi/Isa-88.01. Batch Control Part 1: Models and Terminology 1995, S. Isa, The Instrumentation and A. Society.

[12] M. Zefran and J. Burdick, "Design of switching controllers for systems with changing dynamics," in *Proc. 37th Conf. Decision and Control (CDC)*, 1998, pp. 2113–2118.

[13] X. D. Koutsoukos, K. X. He, M. D. Lemmon, and A. P. J. , "Timed petri nets in hybrid systems: Stability and supervisory control," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 8, pp. 137–173, 1998.

[14] P. Berruet, J. Lallican, A. Rossi, and J. Philippe, "A component based approach for the design of fms control and supervision," in *Proc. IEEE Int. Conf. Syst., Man/ Cybern.*, Oct. 10–12, 2005, vol. 4, pp. 3005–3011.

[15] F. Frizon De Lamotte, J.-L. Berruet, Pascal, and Philippe, "Using model engineering for the criticality analysis of reconfigurable manufacturing systems architectures," *Int. J. Manuf. Technol. Manage.*, vol. 11, no. 3/4, pp. 315–337, 2007.

[16] E. E. Almeida, J. E. Luntz, and D. M. Tilbury, "Event-condition-action systems for reconfigurable logic control," *IEEE Trans. Autom. Sci. Eng.*, vol. 4, no. 2, pp. 167–181, Apr. 2007.

[17] P. E. Miyagi and L. A. M. Riascos, "Modeling and analysis of faulttolerant systems for machining operations based on petri nets," *Control Engineering Practice*, vol. 14, pp. 397–408, 2006.

[18] A. A. Castelnuovo, L. L. Ferrarini, and L. L. Piroddi, "An incremental petri net-based approach to the modeling of production sequences in manufacturing systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 4, no. 3, pp. 424–434, 2007.

[19] A. Balluchi, L. Benvenuti, M. D. Di Benedetto, C. Pinello, and A. L. Sangiovanni-Vincentelli, "Automotive engine control and hybrid systems: Challenges and opportunities," *Proc. IEEE*, vol. 88, no. 7, pp. 888–912, 2000.

[20] W.-E. Ting and J.-S. Lin, "Nonlinear control design of anti-lock braking systems combined with active suspensions," in *Proc. 5th Asian Control Conf.*, 2004, pp. 611–616.

[21] B. Guvenc and E. Kural, "Adaptive cruise control simulator: A low-cost, multiple-driver-in-the-loop simulator," *IEEE Control Syst. Mag.*, vol. 26, no. 3, pp. 42–55, Jun. 2006.

[22] D. Harel, "Statecharts: A visual formalism for complex systems," *Sci. Comput. Program.* vol. 8, no. 3, pp. 231–274, June 1987 [Online]. Available: citeseer.ist.psu.edu/harel87statecharts.html

[23] F. Jahanian and A. Mok, "Modechart: A specification language for real-time systems," *IEEE Trans. Softw. Eng.*, vol. 20, no. 12, pp. 933–947, 1994.

[24] F. Maraninchi and Y. Rémond, "Mode-automata: A new domain-specific construct for the development of safe critical systems," *Sci. Comput. Program.*, vol. 1, no. 46, pp. 219–254, Mar. 2003.

[25] P. Ramadge and W. Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, Jan. 1989.

[26] K. Andersson, B. Lennartson, and M. Fabian, "Restarting flexible manufacturing systems; Synthesis of restart states," in *Proc. 8th Int. Workshop Discrete Event Syst.*, Jul. 2006, pp. 201–206.

[27] O. Kamach, L. Piétrac, and E. Niel, "Multi-model approach to discrete events systems: Application to operating mode management," *Math. Comput. Simulation*, vol. 70, no. 5–6, pp. 394–407, 2005.

[28] O. Kamach, L. Pietrac, and E. Niel, "Supervisory uniqueness for operating mode systems," in *Proc. 16th IFAC World Congr.*, Prague, Jul. 4–8, 2005. [Online]. Available: http://www.ifac-papersonline.net/Detailed/28724.html

[29] G. Faraut, L. Piétrac, and E. Niel, "Identification of incompatible states in mode switching," in *Proc. IEEE Int. Conf. Emerging Technol. Factory Autom. ETFA*, Sep. 2008, pp. 121–128.

[30] W. M. Wonham, "Supervisory Control of Discrete-Event Systems. Ece 1636f/1637s 2006-07" Graduate, Dept. Elect. Comput. Eng., Univ. Toronto, Toronto, ON, Canada, 2006 [Online]. Available: www.control.toronto.edu/people/profs/wonham/, course notes.

[31] C. G. Cassandras and S. Lafortune, , C. G. Cassandras and S. Lafortune, Eds.*, Introduction to Discrete Event Systems*, 2nd ed. New York: Springer, 2007.

[32] R. Kumar, V. K. Garg, and S. I. Marcus, "On controllability and normality of discrete event dynamical systems," *Syst. Control Lett.* vol. 17, no. 3, pp. 157–168, 1991 [Online]. Available: http://home.eng.iastate.edu/rkumar/

[33] O. Kamach, L. Pietrac, and E. Niel, "Forbidden and preforbidden states in the multi-model approach," in *Proc. IMACS Multiconf. Computat. Eng. Syst. Appl.*, Oct. 2006, vol. 2, pp. 1550–1557.

Gregory Faraut (M'08) received the B.Sc. and M.Sc. degrees in electronic, electrotechnic, and automatic from the University of Nice-Sophia Antipolis, Nice, France, in 2004 and 2006 respectively. He is currently working towards the Ph.D. degree in the AMPERE team (UMR CNRS) at the National Institute of Applied Science, Lyon, France.

His research interests include discrete-event systems (DESs), supervisory control theory (SCT), operating mode management, and embedded systems.

Laurent Piétrac (M'04) received the B.Sc. degree in mechanical engineering and the M.Sc. degree in flexible manufacturing systems from the University of Paris VI, Paris, France, in 1990 and 1993, respectively, and the Ph.D. degree in automatic control from the Ecole Normale Suprieure de Cachan, Cachan, France, in 1999.

Since 1999, he has been an Associate Professor with the National Institute of Applied Science, Lyon, France, in the Department "First Cycle" and then in the Department of Industrial Engineering. His research interests include discrete-event systems (DESs), supervisory control theory (SCT), operating mode management and manufacturing execution systems (MESs). Applications are in relation with control applications in manufacturing processes and embedded systems.

Eric Niel received the B.Sc. and M.Sc. degrees in electronic, electrotechnic and automatic from the University of Science, Orléans, France, and the Ph.D. degree in automatic control from the University of Science and Technology, Lille, France, and the HDR (Habilitation à Diriger des Recherches) from the University of Sciences, INSA, Lyon, France, in 1978, 1979, 1983, and 1994, respectively.

Since 1985, he has been an Assistant Professor with the Department of Mechanical Engineering, National Institute of Applied Science, Lyon, France, and then Professor in the Department of Industrial Engineering. He published a book on *Risk Management* (Edition Hermes Lavoisier), and is recently in charge of a collection in Technique de l'Ingénieur. His research interests in the AMPERE team (UMR CNRS) lie in discrete-event systems and system dependability including supervisory control theory, stabilizing controller, hierarchical control, operating mode management, stochastic systems, and malfunctioning assessment. Applications are in relation with process and control design in manufacturing, energy production and embedded systems.