DESIGN OF SWITCHING SUPERVISORS FOR
REACTIVE CLASS DISCRETE EVENT
SYSTEMS

Oulaid Kamach, Laurent Piétrac and Eric Niel

INSA de Lyon, Laboratoire d’Automatique Industrielle
(LAI)
Bat. Antoine de St-Exupery
25, Av. Jean Capelle
69621 Villeurbanne cedex. France
oulaid.kamach@insa-lyon.fr

Abstract: Based on the decentralized supervisory control, Lin and Wonham (1988), Lafortune et al. (2001), the present paper proposes a contribution to the supervisory control of systems (with different production objectives) that react to exceptional events (failure events for instance). Existing work on decentralized control of Discrete Event Systems (DES) focuses on problems where each decentralized control supervisor controls and observes some events in a system and must together achieve some prescribed goals Rudie and Wonham (1992), Yoo and Lafortune (2002). In this model the supervisors act simultaneously on the plant (process) and are sometimes conflicting. Our approach is based on the concept of decentralized control and we propose a procedure which allows in one hand to avoid the conflict problem and on the other hand to manage the commutation (switching) between two operating modes: nominal and degraded modes labelled respectively $N$ and $D$. The developed control strategy for tackle the conflict problem and for manage the switching between two operating modes will be also implemented by the automata.

Copyright® 2005 IFAC

Keywords: Supervisory control theory, operating mode management, switching controllers, systems with changing dynamics, decentralized supervisory control, conflict problem.

1. INTRODUCTION

The Ramadge and Wonham (RW) control theory allows investigation of a number of theoretical control problems, such as feedback-based synthesis of controlled dynamic invariant and concepts such as controllability and observability. However, the process to which this theory is applied is frequently the product of simple components, its state size increases exponentially with the number of components and supervisor synthesis therefore becomes laborious. A common way of dealing with the state explosion is by applying horizontal (modular and decentralized) or vertical (hierarchical) decom-
position of the underlying control problem. The term reactive system, as used in this context, is different from that used in computer systems. In this case, a reactive system is essentially a production system representing different production objectives or operating modes, which are conflicting. However, our contribution can be apply to computer systems, traffic systems, communication protocols, etc. The purpose of a reactive system is to maintain full system operation, when an exceptional event occurs (e.g. a failure event or a repair event). Failure occurrence in this type of system causes some resources to be suspended, but the system itself must remain in service during failure, but with different objectives than in nominal operating mode. Nominal (or normal) mode is then replaced by another mode, which is designed to be blocking with respect to the nominal service mode. However, the two modes do not operate concurrently on the system. It should be noted that a production system can have multiple operating modes Kamach (2004), but in this paper we consider only two operating modes: nominal and degraded mode. In this context, a supervisory control approach based on the decentralized control concept is proposed to manage all system conflicting modes. The controlling action of the degraded mode supervisor is suspended, when there is no failure. When a failure event occurs, the degraded mode supervisor is activated, while the control action of the nominal mode supervisor is suspended until a repair event occurs. The recently activated supervisor must know the current system state to be able to follow its behavior, because it switches from one mode to another. We propose a procedure, which ensures the nonconflicting supervisors in the decentralized architecture and the switching between two operating modes.

The two legal decentralized modes can have opposing specifications. In fact, this paper we suppose that $\Sigma_N \cap \Sigma_D \neq \emptyset$.

![Fig. 1. Reactive system restricted to nominal and degraded modes](image)

Then nominal and degraded specifications are not completely independent each other. Now define $G = (Q, \Sigma, \delta, q_0, Q_m)$ as the generator of the system, assumed to be initially in nominal mode. The nominal process model is defined by: $P_N(L(G)) := L(G_N)$ where $G_N = (Q_N, \Sigma_N, \delta_N, q_0, Q_m, \Sigma_N)$ and $\Sigma_N$ is the set of nominal events of $G_N$. Similarly, we define the degraded process by: $P_D(L(G)) := L(G_D)$ where $G_D = (Q_D, \Sigma_D, \delta_D, q_0, Q_m, \Sigma_D)$ $P_N$ and $P_D$ represent nominal and degraded natural projections respectively and are defined as follows:

$$P_N : \Sigma^* \rightarrow \Sigma_N^*$$
$$P_N(\epsilon) = \epsilon$$
$$P_N(s\sigma) = \begin{cases} P_N(s)\sigma & \text{if } \sigma \in \Sigma_N \\ P_N(s) & \text{otherwise} \end{cases}$$

$$P_D : \Sigma^* \rightarrow \Sigma_D^*$$
$$P_D(\epsilon) = \epsilon$$
$$P_D(s\sigma) = \begin{cases} P_D(s)\sigma & \text{if } \sigma \in \Sigma_D \\ P_D(s) & \text{otherwise} \end{cases}$$

In the conjunctive architecture the supervisors act simultaneously on the process. However, this strategy leads sometimes to the conflict problem. In our approach we will adopt the conjunctive architecture Yoo and LaFortune (2002), Cassandras and LaFortune (1999), and we will manage the switching between nominal and degraded modes. We assume that at any times one mode is activated and we will proposed a procedure which allows to avoiding the conflict problem by activating at one times a suitable supervisor.

We start by recalling some key definitions and concepts that are needed before we can present our new result.

Let use define the following classes of languages:

2. DECENTRALIZED NOMINAL AND DEGRADED SUPERVISORS

In this section, we describe formally and model mathematically the reactive system supervisory control problem.

Two entities are distinguished in Figure 1

1. Nominal decentralized supervisory control
2. Degraded decentralized supervisory control.
\[ L_{\sigma} := \{ s \in L(G) \mid s \sigma_f \in L(G) \}, \]

\[ L_{\sigma_r} := \{ s \in L(G) \mid s \sigma_r \in L(G) \} \]

\[ L_{\sigma} \text{ denotes the set of all traces (sequences) } s \text{ that end by a failure event } \sigma_f, \]

\[ L_{\sigma_r} \text{ denotes the set of all traces } s \text{ that end by a repair event } \sigma_r. \]

\[ L_{\sigma_f} := \{ s \in L(G) \mid s \notin L_{\sigma_f} \} \]

\[ L_{\sigma_r} := \{ s \in L(G) \mid s \notin L_{\sigma_r} \} \]

\[ L_{\sigma_f} \text{ denotes the set of all traces } s \text{ that do not belong to } L_{\sigma_f}, \]

\[ L_{\sigma_r} \text{ denotes the set of all traces } s \text{ that do not belong to } L_{\sigma_r}. \]

Let us define a nominal decentralized supervisor (nominal supervisor for the sake of brevity) who acts on the process in the nominal operating mode, denoted by \( S_N \), as

\[ S_N : P_N(L(G)) \rightarrow \Gamma := \{ \gamma \in 2^\Sigma : \Sigma_{uc} \subseteq \gamma \} \text{ such that } \forall s \in L(G) : \]

\[ S_N(P_N(s)) = \begin{cases} \Sigma - \Sigma_{uc} \cup \{ \sigma \in \Sigma_{uc} \mid P_N^{-1}(P_N(s))\sigma \in K \bsn {0} \} & \text{if } s \notin L_{\sigma_f} \cup L_{\sigma_r} \\ \Sigma & \text{if } s \in L_{\sigma_f} \cup L_{\sigma_r} \end{cases} \]

\( \Gamma \) represents a particular subset of events to be enabled by the supervisor.

In the nominal operating mode, namely no failure event occurs or after occurrence of repair event, the nominal supervisor disable some controllable events in order to achieve a given desired language \( (K) \). But if a failure event occurs the nominal supervisor decision rule enables \( \Sigma_{uc} \) by default.

Now let us define a degraded decentralized supervisor (degraded supervisor for the sake of brevity) who act on the process in the degraded operating mode, denoted by \( S_D \), as

\[ S_D : P_D(L(G)) \rightarrow \gamma \text{ such that } \forall s \in L(G) : \]

\[ S_D(P_D(s)) = \begin{cases} \Sigma - \Sigma_{uc} \cup \{ \sigma \in \Sigma_{uc} \mid P_D^{-1}(P_D(s))\sigma \in K \bsn {0} \} & \text{if } s \notin L_{\sigma_f} \cup L_{\sigma_r} \\ \Sigma & \text{if } s \in L_{\sigma_f} \cup L_{\sigma_r} \end{cases} \]

At the occurrence of the failure event, namely in the degraded operating mode, the degraded supervisor disable some controllable events in order to achieved a given desired language \( (K) \). But if a repair event occurs the degraded supervisor decision rule enables by default all generated process events.

Two supervisors \( S_N \) and \( S_D \) cooperate in order to achieve a given desired controlled behavior \( K \).

**Proposition 1**

Under the foregoing definition of \( S_N \) and \( S_D \) the languages \( L(S_N/G) \) and \( L(S_D/G) \) are non-conflicting, namely

\[ L(S_N/G) \cap L(S_D/G) = \overline{L(S_N/G) \cap L(S_D/G)}. \]

**Proof**

The inclusion

\[ L(S_N/G) \cap L(S_D/G) \subseteq L(S_N/G) \cap L(S_D/G) \]

is automatic. In fact:

We have

\[ L(S_N/G) \cap L(S_D/G) \subseteq L(S_N/G) \]

Rightarrow

\[ L(S_N/G) \cap L(S_D/G) \subseteq L(S_N/G) \cap L(S_D/G) \]

We have

\[ L(S_N/G) \cap L(S_D/G) \subseteq L(S_D/G) \]

So

\[ L(S_N/G) \cap L(S_D/G) \subseteq L(S_N/G) \cap L(S_D/G) \]

Now we prove the inverse inclusion, namely

\[ L(S_N/G) \cap L(S_D/G) \subseteq L(S_N/G) \cap L(S_D/G) \]

Let \( s \in L(S_N/G) \cap L(S_D/G) \rightarrow s \in L(S_N/G) \) and \( s \in L(S_D/G) \).

We distinguished two cases:

**Case 1:** \( s \in L_{\sigma_f} \cup L_{\sigma_r} \)

In this case \( \forall \sigma \in \Sigma, \) such that \( s \sigma \in L(G) \) and \( \sigma \in S_N(P_N(s)), \) namely \( \sigma \) is enabled after the sequence \( s \) by \( S_N. \)

In this case the supervisor \( S_D \) enables all generated events of \( \Sigma, \) in particular \( \sigma. \) So \( \sigma \in S_D(P_D(s)). \) Then \( s \sigma \in L(S_N/G) \cap L(S_D/G) \) consequently \( s \in L(S_N/G) \cap L(S_D/G) \)

**Case 2:** \( s \in L_{\sigma_f} \cup L_{\sigma_r} \)

In this case \( \forall \sigma \in \Sigma, \) such that \( s \sigma \in L(G) \) and \( \sigma \in S_D(P_D(s)), \) namely \( \sigma \) is enabled after the sequence \( s \) by \( S_D. \)

In this case the supervisor \( S_N \) enabled all generated events of \( \Sigma, \) in particular \( \sigma. \) So \( \sigma \in S_N(P_N(s)). \) Then \( s \sigma \in L(S_N/G) \cap L(S_D/G) \) consequently \( s \in L(S_N/G) \cap L(S_D/G). \)

**\( \Box \)**

3. IMPLEMENTATION OF SUPERVISORY CONTROLS BY AUTOMATA

In the previous section, we studied the switching control decisions between different supervisors and we design the nonconflicting supervisors for systems whose dynamics change as they evolve on two operating modes. While theoretically convenient, the abstract definition of a supervisory control as a map \( S_i : P_i(L(G)) \rightarrow \gamma \) does not in itself provide a concrete representation for practical implementation. So this section will be devoted
to the practical implementation by constructing two corresponding automata, where each one implements the one supervisor according a given operating mode. In fact, the nominal mode supervisor $S_N$ will be implemented by $E_N$. $E_N$ is defined by the 5-tuple: $E_N := (X_N, \Sigma_N, \xi_N, x_{0,N}, X_{m,N})$. Similarly, $E_D$ implements the degraded mode supervisor $S_D$, with $E_D := (X_D, \Sigma_D, \xi_D, x_{0,D}, X_{m,D})$. We describe our proposition via the example in Figure 2. Automaton $E_N$ implements the nominal supervisor $S_N$ and $E_D$ implements $S_D$.

Recall that, the proposed work in Section 2 for avoid the conflict problem between the supervisor $S_N$ and $S_D$ is as follows: in the nominal mode only the nominal supervisor $S_N$ acts on the process, on the other hand the degraded supervisor $S_D$ enables by default all generated process events. When a failure event $\sigma_f$ occurred, the situation is reversed and the degraded supervisor $S_D$ acts on the process by disabling some controllable events, whereas the default action control for the nominal supervisor $S_N$ is to enabled all the generated process events. In order to implement this strategy in the context of automaton, we must extend each automaton by adding to each one a particular state called inactive state $x_{i,in}$. The inactive state $x_{i,in}$ represents the default action for the supervisor $S_i$ which is enable all of the generated process events, i.e. in this state all the generated process events are supposed enabled by default.

More precisely, in the nominal mode $E_n$ must be in a state $x \in X_N$ different of the inactive state $x_{N,in}$ in order to act on the process. On the other hand the automaton $E_D$ must be in inactive state $x_{D,in}$ preventing thus to act simultaneously on the process by enabling all the generated process events. With the occurrence of the failure event $\sigma_f$, $E_D$ must be in a state $x \in X_D$ different of the inactive state $x_{D,in}$ in order to act on the process, whereas $E_N$ must be in inactive state $x_{N,in}$ in order to enable all the generated process events and not act simultaneously on the process. The more difficult problem of the commutation between modes is the determination of the adequate reachable state from the inactive state $x_{i,in}$ ($i \in \{N,D\}$). This state denoted starting state must be compatible with the current state of the process to ensuring that appropriate control is exercised.

Formally for $i \in \{N,D\}$ the extended automata $E_{i,ext}$ is given as follows (see Figure 3):

$E_{i,ext} := (X_{i,ext}, \Sigma_{i,ext}, \xi_{i,ext}, x_{0,i,ext}, X_{m,i,ext})$ where:

- $X_{i,ext} = X_i \cup \{x_{i,in}\}$
- $\Sigma_{i,ext} = \Sigma_i \cup \{\sigma_f, \sigma_r\}$
- $x_{i,0,ext} = \left\{\begin{array}{ll}
  x_{i,0} & \text{if } i = N \\
  x_{i,in} & \text{if } i = D
\end{array}\right.$
- $\xi_{i,ext}$ is defined as follows:
  1. $\forall x \in X_i$ and $\forall \sigma \in \Sigma_i$ if $\xi_i(x, \sigma)$ exists, then: $\xi_{i,ext}(x, \sigma) := \xi_i(x, \sigma)$
  2. $\forall x \in X_N$ (resp. $x \in X_D$) from which the failure event can occurs (resp. the repair event) $\xi_{N,ext}(x, \sigma_f) := x_{N,in}$ (resp. $\xi_{D,ext}(x, \sigma_r) := x_{D,in}$)
  3. $\xi_{N,ext}(x_{N,in}, \sigma_r)$ and $\xi_{D,ext}(x_{D,in}, \sigma_f)$ will be defined after.

We assume that the failure event can occurs in state $x_0$ of $E_N$ and the repair event can occurs in state $x_2$ of $E_D$, the extended automaton is given by the Figure 3.

The above extended automaton is incomplete. Transition function $\xi_{i,ext}(x_{D,in}, \sigma_f)$ and $\xi_{N,ext}(x_{N,in}, \sigma_r)$ remains to be defined.

To solve this problem i.e. the determination of the reachable state from the inactive state which achieve the desired language, we adopt the following notations and definitions:

Let $K$ be the desired language, such that $K \subseteq L(G)$ and $K \neq \emptyset$.

$K_1 = \{s_1 \in K \mid s_1\sigma_f \in K\}$ inclus the set of all traces belonging to $K$ and end by a failure event $\sigma_f$. 

---

**Fig. 2. nominal and degraded specification**

**Fig. 3. extended nominal and degraded specification**
\( K_2 = \{ s_2 \mid s_3 \sigma \in K \} \) denotes the set of all traces that following the set of all traces of \( K_1 \) and leads to desired language \( K \).
\( K_3 = \{ s_3 \mid s_3 \sigma \in K \} \) is the subset of the traces belonging to \( K \) and end by a repair event \( \sigma \).
\( K_4 = \{ s_4 \mid s_3 \sigma \in K \} \) denotes the set of all traces that following the set of all traces of \( K_3 \) and achieve the desired language \( K \).

**Definition 1**

A state \( x_j \in X_N \) is said to be compatible with a state \( x_j \in X_D \) if and only if:

1. \( \forall s \in K_1 \) such that \( \xi_N(x_i, s) \) exists
   \[ \Rightarrow \forall s' \in K_2 \text{, } \xi_D(x_j, s')! \]
2. \( \forall s \in K_3 \) such that \( \xi_D(x_j, s) \) exists
   \[ \Rightarrow \forall s' \in K_4 \text{, } \xi_D(x_j, s')! \]

Intuitively, compatibility means that, if one leaves \( E_N \) (resp. \( E_D \)) from a state \( x_i \) (resp. \( x_j \)), we must find among the states-space \( X_D \) a state \( x_j \) (resp. \( x_i \)) of the automaton \( E_D \) (resp. \( E_N \)) which allows to achieve the desired language.

After having to define the compatibility concept between two states \( x_i \in X_N \) and \( x_j \in X_D \), we now will build a compatibility matrix. This matrix takes into account only the states from which the failure or repair event occurs. Let \( M \) a such matrix:

\[
M = \begin{pmatrix}
m_{11} & m_{12} & \cdots & m_{1n} \\
m_{21} & m_{22} & \cdots & m_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
m_{m1} & m_{n2} & \cdots & m_{nn}
\end{pmatrix}
\]

with

\[ m_{ij} = \begin{cases} 
1 & \text{if } x_i \text{ and } x_j \text{ are compatible} \\
0 & \text{otherwise}
\end{cases} \]

Let us recall that when a failure or repair event occurred from a state \( x_i \in X_i \), we must determine correctly a compatible state \( x_j \) which will be reachable from the inactive state \( x_i, in \), such that whether one acts on the process from this state \( (x_j) \), the desired language will be achieved.

However, each automaton had only direct observations (and was not given information from the other automata), then they would not be able to determine correctly the reachable state from \( x_i, in \). The idea then is to allow the automaton to communicate with each other so that helps the other to find among the set of states of automaton \( E_i \), the reachable state from \( x_i, in \) which allow to achieve the desired language.

So if \( E_D \) (resp. \( E_N \)) receives information on the state of \( E_N \) (resp. \( E_D \)) from which the failure event occurred (resp. repair event), it easily to determine by basing on the preceding definition of compatibility, the adequate reachable state from the inactive state \( x_{D,in} \) (resp. \( x_{N,in} \)).

In order to ensuring the commutation between modes, we use two information channels (or mapping) labelled \( \text{commun}_{N,D} \) (information sent up by \( E_N \) to \( E_D \)) and \( \text{commun}_{D,N} \) (information sent up by \( E_D \) to \( E_N \)). The channels information is interpreted as carrying information of the state automaton from which the failure or repair event occurs. But after determine formally the information channels, we re-extended each automaton by adding in each state of one, where the failure or repair event can occur, its compatible state. So the extended automaton is given as follows: \( E_{i,ext} \) := \( (X_{i,ext}, \Sigma_{i,ext}, \xi_{i,ext}, x_{0,i,ext}, X_{m,i,ext}) \) where \( \Sigma_{i,ext}, \xi_{i,ext}, x_{0,i,ext} \) and \( X_{m,i,ext} \) have the same above definition, the only difference here is the extended state space \( X_{i,ext} \).

In fact each state \( x_i \in X_N \) (resp. \( x_j \in X_D \)) will be associated with a compatible state \( x_j \in X_j \) (resp. \( x_i \in X_N \)), such that \( m_{ij} = 1 \). Then the extended states-space can be written as follows: \( X_{i,ext} = \{(x_i, x_j) \mid m_{ij} = 1 \} \cup \{x_{i, in}\} \), consequently the global state \( x \in X_{i,ext} \) has the form \((x_i, x_j)\) or \(x_{i, in}\), where \( x_j \) is a compatible state of \( x_i \). In the Figure 4, we assume that the compatible state of \( x_0 \) is \( x_2 \) and the compatible state of \( x_1 \) is \( x_2 \) (by basing on the definition of compatibility).

![Compatibility states](image)

**Fig. 4.** compatibility states

Formally the information channels is given as follows:

**Definition 2**

\[ \text{commun}_{N,D} : X_{N,ext} \times \sigma_f \longrightarrow \text{Labelled}(\sigma_f) \]

such that \( \forall (x_i, x_j) \in X_{N,ext} \)

\[ ((x_i, x_j), \sigma_f) \mapsto \sigma_{f_j} \]

**Definition 3**
\[ \text{commu}_{D,N} : X_{D,\text{ext}} \times \sigma_r \rightarrow \text{Labelled}(\sigma_r) \]

such that \( \forall (x_i, x_j) \in X_{D,\text{ext}} \)

\( ((x_i, x_j), \sigma_r) \mapsto \sigma_{r_j} \)

Based on the information of state from which the failure or repair event occurred and on the compatibility definition, the information channel provides a corresponding failure or repair event which allow to determine correctly the reachable state form the inactive state \( x_{i,in} \) from which we make the correct control decisions. Finally the complete automata is given as follows (see Figure 5):

- \( X_{i,\text{ext}} = \{(x_i, x_j) \mid x_i \in X_i \text{ with } x_j \in X_D \text{ and } m_{ij} = 1\} \cup \{x_{i,in}\} \)
- \( \Sigma_{i,\text{ext}} = \Sigma_i \cup \{\sigma_f, \sigma_r\} \)
- \( x_{i,0,\text{ext}} = \{x_{i,0} \text{ if } i = N \}
  \quad \{x_{i,in} \text{ if } i = D \} \)
- \( \xi_{i,\text{ext}} \) is defined by:
  1. \( \forall x \in X_i \text{ and } \forall \sigma \in \Sigma_i \text{ if } \xi_i(x, \sigma) \text{ exists, then: } \)
    \[ \xi_{i,\text{ext}}(x, \sigma) := \xi_i(x, \sigma) \]
  2. \( \forall x \in X_N \text{ (resp. } x \in X_D \text{) from which the failure event can occurs (resp. the repair event) } \xi_{N,\text{ext}}(x, \sigma_f) := x_{N,\text{in}} \text{ (resp. } \xi_{D,\text{ext}}(x, \sigma_r) := x_{D,\text{in}} \text{) } \)
  3. \( \xi_{N,\text{ext}}(x_{N,\text{in}}, \sigma_r) = x_i \text{ if } \)
    \( \text{commu}_{D,N}(x_j, x_i, \sigma_r) = \sigma_r \text{ and } \)
    \( \xi_{D,\text{ext}}(x_{D,\text{in}}, \sigma_f) = x_j \text{ if } \)
    \( \text{commu}_{D,N}(x_i, x_j, \sigma_f) = \sigma_f. \)

![Diagram](image)

Fig. 5. complete specifications models

4. CONCLUSION

An approach based on decentralized supervisory control has been proposed in the present paper. We have presented a framework for designing two nonconflicting supervisors for systems whose dynamics change as they evolve on two operating modes: nominal and degraded mode. Such systems form a subclass of reactive systems. A natural control strategy for avoid the conflict problem between the set of local supervisors in the decentralized architecture is: initially i.e in the nominal mode, one appropriate supervisor who acts on the process but the other (the supervisor which assure the degraded mode) enables by default all generated process events. When a failure event \( \sigma_f \) occurred i.e. in the degraded mode, the situation is reversed the degraded supervisor \( S_D \) acts on the process by disabling some controllable events, whereas the default action control for the nominal supervisor \( S_N \) is to enable all the generated process events. We also implemented this control strategy by the automaton and we have proposed a procedure which assure the commutation modes. The proposed procedure allows, when a commutation mode is happen, to determine the starting states of the automaton that achieve the desired language.

REFERENCES


