RECONFIGURATION OF DISCRETE EVENT SYSTEM CONTROLLERS WITH DYNAMIC SENSING SET

Jing Liu and Houshang Darabi

University of Illinois at Chicago, Department of Mechanical and Industrial Engineering

Abstract: This paper presents a finite automaton model to describe the control reconfiguration of discrete event systems (DES) with respect to the dynamic changes of observation means, especially when the changes are not acceptable for the conventional observability theory. The model includes other classes of events besides the regular DES events, such as repair and failure events of the observation means. Given a regular DES controller, through a systematic procedure it is extended to include the effect of those events on the control. The potential application of this work is the optimization of the reconfiguration strategies. *Copyright © 2005 IFAC*

Keywords: supervisory control, observability, sensor failures, formal methods.

1. INTRODUCTION

This paper models the control reconfiguration of DES with respect to the change of observation means. Specifically, the effect of the reconfiguration strategies on DES when the change of observation means is not acceptable for the conventional observability theory is addressed.

The investigation on the observability theory of DES was triggered by two parallel and original works of (Lin and Wonham, 1988) and (Cieslak, et al., 1988). The DES observability is critical for the DES controller as the controller takes the control actions according to what it observes from the DES.

It is assumed that for each event, there is a sensor reporting its occurrence to the controller. If a sensor is not working properly or its communication channel with the controller fails, the controller cannot obtain the information regarding the occurrence of the corresponding event. In this case, from the controller's point of view, this event is unobservable. Once the sensor and/or communication channel are repaired, that event becomes observable again.

In practice, the sensors or communication channels can be broken down or repaired at any time during the control. In a sense, in certain state the controller may "see" a different set of events than it does in the initial state. In other words, the observability of events changes over time. In conventional observability theory of the DES (Lin and Wonham, 1988), it is assumed that the set of observable events remains unchanged. Furthermore, there are some events which are critical for keeping the right control - the "must-be-observed" events. However, if these events are not seen by the controller, the controlled DES will go out of control. But there are some occasions where the controller can survive even if some of the "must-be-observed" events are not observed according to (Darabi, et al., 2003). Here they propose a control switching policy to discover an appropriate control policy and switch to the new policy on the fly. The search for the new control policy and the switching actions are performed by another control agent called the mega-controller. The proposed switching theory in this work is based on the set of observable projections originally introduced in (Haji-Valizadeh and Loparo 1996). Therefore if a new control policy is found when the set of observable events changes, then this policy can survive for infinite time given that the observable events set does not change again after the switch. Such policies, called infinite time policies, have some limitations according to (Liu and Darabi 2004). In fact the infinite time policies cannot provide the controller maximum survival time subject to the dynamic observable event set.

In (Liu and Darabi 2004) the authors have developed the class of finite time policies. This class is a superset of the infinite time policies introduced by (Darabi, et al., 2003). This class, in addition to the infinite time policies, consists of all control policies that are feasible for a finite duration of time. A finite time observation policy is good for the current state, but may not be good for other states. Using finite time policies provides a more general solution for the

controller reconfiguration upon the change of event observability. In addition, (Liu and Darabi 2004) offers a new reconfiguration strategy, so called control feedback adjustment, to resolve control conflicts.

One of the problems generated by the work (Liu and Darabi 2004) is the implementation of the control reconfiguration strategies through a mega-controller. The mega-controller interacts with the DES controller to adapt the control to the dynamic changes of observation means. In this paper, the authors develop a finite automaton model to describe combined behavior of the mega-controller and the DES controller. By this model, the reader can have a full picture of how the mega-controller evolves upon the change of observation means. This finite automaton includes other classes of events besides the regular DES events. For example repair and failure events of sensors can change the state of this finite automaton. Our development starts from the regular DES controlled finite automaton and through a systematic procedure it extends the regular automaton to include the behavior of additional events. One of the main applications of the extended finite automaton (not discussed in this paper) is in optimizing the reconfiguration decisions made by the mega-controller.

The paper is organized as follows. Section 2 gives the preliminaries. Section 3 provides the algorithm to generate the extended finite automaton. Section 4 presents an illustrative example for building the extended model, and section 5 concludes the paper and discusses the future research.

2. PRELIMINARIES

According (Ramadge and Wonham 1987), a DES is modeled by finite automaton $G = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is the set of states, Σ is the finite set of events (which can be partitioned into two disjoint subsets, controllable events set Σ_c, and uncontrollable events set Σ_{uc}), $\delta : Q \times \Sigma \to Q$ is the transition functions, q_0 is the initial state, and $Q_m \subseteq Q$ is the set of marked states. G is said to be blocking if $\overline{L_m} \subset L$, and nonblocking if $\overline{L_m} = L$, where $L = \{s \in \Sigma^* : \delta(q_0, s) \text{ is defined}\}$ is the language generated by G (Σ^* is the set of finite length strings of Σ). G models the uncontrolled plant. The supervisor **S** (or controller) interacts with G (or plant) in a closed loop manner (Fig. 1), and it assures that the plant does not violate a given set of specifications. The specifications are the conditions that the designers wish to impose on the plant. Mathematically, $\mathbf{S} = (S, \Phi)$, where

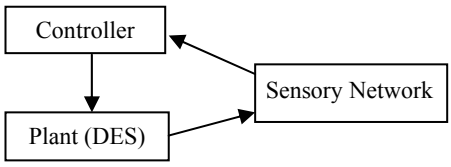

Fig. 1. DES control loop

$S = (X, \Sigma, f, x_0, X_m)$ is a deterministic automaton with state set X, initial state x_0, a marked subset, $X_m \subseteq X$ and transition function $f : X \times \Sigma \to X$; $\Phi : X \times \Sigma \to \{1, 0, dc\}$ is the feedback function. $\Phi(x, \sigma) = 1$ (0) indicates that the control action at state x is to enable (disable) event σ. "dc" is an abbreviation for "don't care", which implies that the enabling or disabling of σ at x doesn't affect the behavior of G. $\Gamma : X \to 2^{\Sigma}$ is the active event function, and is defined as $\Gamma(x) = \{\sigma : f(x, \sigma) \text{ is defined}\}$ for all $x \in X$. In other words, $\Gamma(x)$ includes all the events that are enabled by **S** at state x. The controlled plant language, so called coupled language, is shown by $K = L(\mathbf{S} / G)$.

3. EXTENDED CONTROLLER FINITE AUTOMATON MODEL

It is assumed that a DES and its controller, as stated above, are given. It is also assume that there is a sensor associated with each event reporting its occurrence to the controller. Therefore, the sensor status change results in change in event observability. To develop the algorithm that generates the finite automata model with extended sensory events and reconfiguration decisions, some definitions are first provided.

The *sensor status* of an event σ, ss_σ, can be failed, represented by 0 and working, represented by 1. The change of sensor status is triggered by two sets of events, sensor repair event set, R and sensor breakdown event set B. $b_\sigma \in B$, $r_\sigma \in R$ are the sensor breakdown and repair event for σ respectively.

An *extended state* y is represented by "$\bar{X}\_SS$" where $\bar{X} \subset X$ and $SS = (ss_{\sigma_1}, ss_{\sigma_2}, \cdots, ss_{\sigma_i}, \cdots, ss_{\sigma_n})$ is the sensor status vector (ss_{σ_i} is the sensor status of event σ_i in $\Sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_i, \cdots, \sigma_n\}$). As defined before the element of *SS* are 0's or 1's.

Extended state $y := \bar{X}\_SS$ is a *conflict state* if there exist $x_1, x_2 \in \bar{X}$ and $\sigma \in \Sigma$ such that $\Phi(x_1, e) = 1$, $\Phi(x_2, e) = 0$.

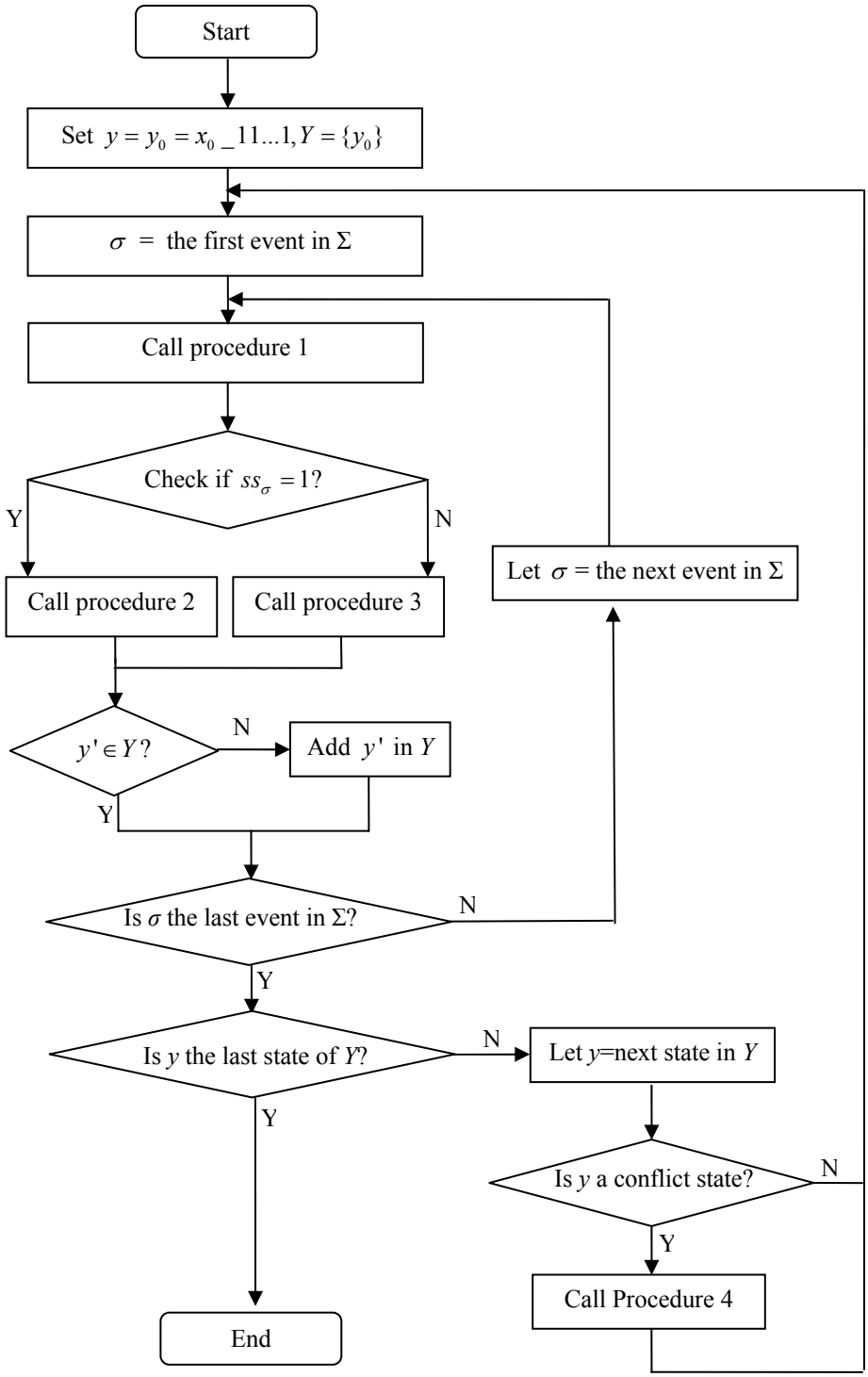

Fig. 2. Flow chart for building reconfigured controller Θ

Accordingly, define the *conflict event set* of the conflict state by $CE(y) = \{\sigma \mid \exists x_1, x_2 \in \overline{X}, \sigma \in \Sigma, \Phi(x_1, \sigma) = 1, \Phi(x_2, \sigma) = 0\}$, the *enabling state set* for the conflict event $\sigma \in CE(y)$ by $ES(y) = \{x \mid x \in \overline{X},\ \sigma \in CE(y), \Phi(x, \sigma) = 1\}$, and the *disabling state set* for the conflict event $\sigma \in CE(y)$ by $DS(y) = \{x \mid x \in \overline{X}, \sigma \in CE(y), \Phi(x, \sigma) = 0\}$.

In order to have a full representation of the controller reconfiguration evolution, consider all the possible sensor status changes at any state. Given a controller $\mathbf{S} = (S, \Phi)$, and S is five-tuple automaton $S = (X, \Sigma, f, x_0, X_m)$, the extended controller automaton with a six-tuple automaton $\Theta = (Y, E, g, y_0, Y_m, inf)$ is modeled. The elements in Θ are illustrated as follows:

(1) Y – the set of extended states
The members of Y are the states of the extended automaton and are generated using the flowchart in Figure 2.

(2) E – event space
$E = \Sigma \cup B \cup R$

(3) y_0 – initial state
$y_0 := x_0 \_ SS$

(4) Y_m – marked subset
$Y_m = \{y \mid y := x \_ SS, x \in X_m\}$.

(5) *inf* – infeasible state
The infeasible state is any state that is reached from state y_0 through string s such that $s \in L$ but $s \notin K$.

(6) g – transition function
$g : Y \times E \to Y \cup \{inf\}$

Accordingly, define active event function as $AE : Y \to 2^E$ and $AE(y) = \{e : g(y,e)$ is defined$\}$.

Figure 2 shows the algorithm for constructing Θ . The inputs to these algorithms include the regular controller **S** and the initial extended state y_0 . All the procedures used in Figure 2 are based on the occurrence of events in Σ and the sensory events.

Procedure 1: called for every event $\sigma \in \Sigma$, let transition function $g(y,\sigma) = y'$ and $y' := \bar{X}' \_ SS'$ is computed as follows,

a) $\bar{X}_1 = \{f(x,\sigma) \mid x \in \bar{X}, f(x,\sigma)$ is defined.$\}$;

b) $\bar{X}_2 = \{f(x,t) \mid x \in \bar{X}_1,\ t = \{\sigma \mid \sigma \in \Sigma$ and $ss_\sigma = 0\}^*$, $f(x,t)$ is defined.$\}$;

c) $\bar{X}' = \bar{X}_1 \cup \bar{X}_2, SS' = SS$, $y' := (\bar{X}_1 \cup \bar{X}_2) \_ SS$.

d) Add σ to $AE(y)$.

Procedure 2: called when $ss_\sigma = 1$.

(1) Add b_σ to $AE(y)$.

(2) $g(y,b_\sigma) = y'$, the algorithm for computing y' is as follows,

a) $\bar{X}' = \{f(x,t) \mid x \in \bar{X}$,
$t = \{\sigma' \mid \sigma' \in \Sigma$, $ss_{\sigma'} = 0$ or $\sigma' = \sigma\}^*\}$

b) $y' := \bar{X}' \_ SS'$ where the element

$$ss_{\sigma'} = \begin{cases} 0, & \text{if } \sigma' = \sigma; \\ ss_\sigma, & \text{otherwise.} \end{cases}$$

Procedure 3: called when $ss_\sigma = 0$.

(1) Add r_σ to $AE(y)$.

(2) $g(y,b_\sigma) = y'$, $y' := \bar{X} \_ SS'$ where the element

$$ss_{\sigma'} = \begin{cases} 1, & \text{if } \sigma' = \sigma; \\ ss_\sigma, & \text{otherwise.} \end{cases}$$

Procedure 4 (Feedback Adjustment): called when in a conflict state. At the conflict state $y := \bar{X} \_ SS$, there are two ways to adjust the feedback to resolve the conflict:

(1) Enable the conflict event $\sigma \in CE(y)$ at all the states in $DS(y)$. $g(y,\sigma) = y'$, and $g(y,\sigma) = inf$ where $y' := \bar{X}' \_ SS'$ and $\bar{X}' = \{f(x,\sigma) \mid x \in ES(y)\}$. This makes the reconfigured controller automaton Θ nondeterministic. At state $x \in DS(y)$, enabling σ causes the controller to enter an infeasible state.

(2) Disable the conflict event $\sigma \in CE(y)$ at all the states in $DS(y)$. Remove even σ from $AE(y)$. If $|\, AE(y) \,| = 0$, then state y is a deadlock.

Discussion The four procedures deal with the event occurrences and conflict resolving. Events include the regular events (Σ), sensor breakdown events (B) and sensor repair events (R). At an extended state $y := \bar{X} \_ SS$, consider all the events that possibly can happen. First, every event in Σ is examined to find the ones that can take place. Then, it is investigated if the sensor breakdown or repair events are possible to happen or not according to the current sensor status. For doing so, three rules are used. (1) Event $\sigma \in \Sigma$ can happen if and only if there is a state $x \in \bar{X}$ where σ is active. (2) Sensor breakdown event b_σ can happen if and only if $ss_\sigma = 1$, i.e. sensor for event σ can fail only when it is at work. (It is assumed that all sensors are working at the initial state). (3) Sensor repair event r_σ can happen if and only if $ss_\sigma = 0$, i.e. sensor for event σ can be repaired only when it is failed. During this process, control conflict could come up as a result that some states requiring different feedback policy on certain events are aggregated together due to the unavailability of sensors. The procedures are explained one by one in the followings.

Procedure 1 handles the occurrence of events in Σ (the event set of given controller). For the state set part $\bar{X}$, the states where that event is enabled are aggregated up, and the resultant state set are further combined with the states that cannot be differentiated due to the sensor failure. The sensor status vector remains the same.

Procedure 2 copes with the occurrence of sensor breakdown events. Once the failure takes place, the sensor cannot report the occurrence of corresponding event to the controller. The states that are reachable from the given state by the unobservable strings are aggregated together. The element in sensor status vector for that event changes from 1 to 0 accordingly.

Procedure 3 takes care of the case that sensor is failed. At this case, that sensor repair event is possible to happen. When it happens, it only affects the sensor status vector.

Procedure 4 deals with the control conflict resolving. There are two ways to tackle the conflict, either enabling or disabling the events that cause the conflict. Which way is overweight the other one is different from case to case.

From the generation of Θ it is evident that the feedback adjustment at conflict states could generate states not reachable or not feasible. Taking different feedback policy adjustments affects the evolution path of the finite automaton Θ, and thus affects the chance of leading to infeasible or blocking states. As mentioned in the introduction, the selection between different actions outlined by Θ requires an optimization model that uses Θ as its input. Another input to this model can be a reconfiguration criterion such as maximizing the probability of reaching to a marked state. However, the objective of this paper is not to address such an optimization framework.

4. An Illustrative Example

The controller automaton is shown in Figure 3, and the feedback of each state is listed in table 1. Event set $\Sigma = \{e_1, e_2, e_3\}$, and all are controllable. $c_m = -5$, $c_I = 4$ and $c_b = 2$.

For this example, configure the controller under the policy that event e_1, e_2 are to be enabled and e_3 is to be disabled if there is a conflict.

Due to the space limitation, the figure of the finite automaton of the reconfigured controller cannot be shown completely. Here only a part of the figure in Figure4 is shown.

The evolution of the extended controller Θ is illustrated as follows,

(1) At the initial state 1_111,

- event e_1 is active, call procedure 1.
  Because $f(1, e_3) = 3$, $g(1\_111, e_1) = 3\_111$.
- $ss_{e_2} = 1$, call procedure 2. Add b_{e_2} to $AE(1\_111)$.
  Because $f(1, e_2) = 2$, $g(1\_111, b_{e_1}) = 12\_101$.

(2) At state 12_101,
State 12_101 is conflict, and $CE(12\_101) = \{e_1\}$.
If e_1 is enabled at $DS(12\_101) = \{2\}$,
then $g(12\_101, e_1) = 3\_101$,
and $g(12\_101, e_1) = inf$.

(3) At state 3_111,
e_3 is active, follow procedure 1, $f(3, e_3) = 4$,
$g(3\_111, e_3) = 4\_111$ which is the marked state.

(4) At state 34_110,

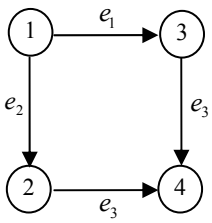

Fig. 3. Controller automaton of the illustrative example.

Table 1 Feedback table of the illustrative

	e_1	e_2	e_3
1	1	1	dc
2	0	dc	1
3	dc	0	1
4	dc	dc	0

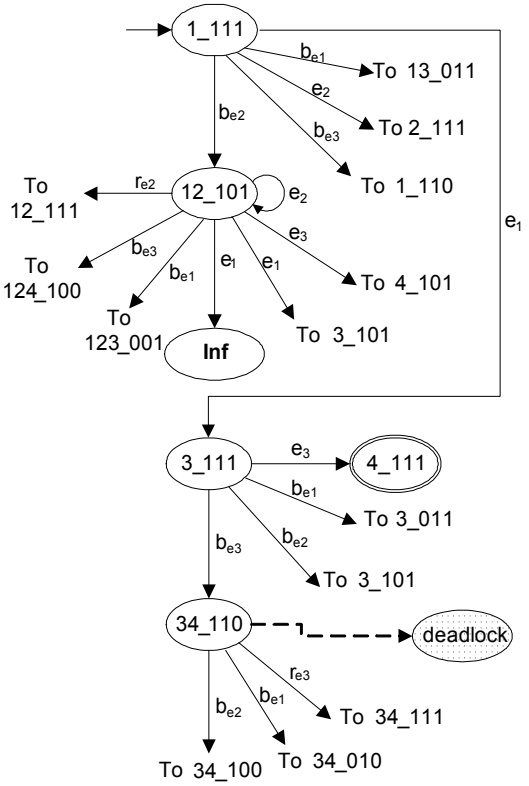

Fig. 4. Partial representation of the extended automaton Θ

State 34_110 is conflict, and $CE(34\_110) = \{e_3\}$.
If e_3 is disabled at $ES(34\_110) = \{3\}$,
then $g(34\_110, e_3) = \emptyset$, so there is a chance that state 34_110 becomes a deadlock itself.

The transitions between states are listed in Table 2.

5. CONCLUSION AND FUTURE RESEARCH

This paper provides a formal model that embeds the controller reconfiguration options upon the change in

Table 2. Transitions between states for illustrative example.

State Name	State No.	Events								
		e_1	e_2	e_3	b_{e1}	b_{e2}	b_{e3}	r_{e1}	r_{e2}	r_{e3}
1_111	1	3	2		7	8	9			
2_111	2			4	10	11	12			
3_111	3			4	13	14	15			
4_111	4									
inf	5									
block	6									
13_011	7	7	5,10	16		20	21	19		
12_101	8	5,14	8	17	20		23		22	
1_110	9	15	12		21	23				
2_011	10			16		24	25	2		
2_101	11			17	24		26		2	
24_110	12			6	25	26				27
3_011	13			16		28	29	3		
3_101	14			17	28		30		3	
34_110	15			6	29	30				31
4_011	16					32	33	4		
4_101	17				32		34		4	
4_110	18				33	34				4
13_111	19	3	2,5	4	7	35	36			
123_001	20	5,20	5,20	32			38	35	37	
134_010	21	21	5,25	6		38		36		39
12_111	22	5,3	2	4	37	8	40			
124_100	23	5,30	23	6	38				40	41
2_001	24			32			42	11	10	
24_010	25			6		42		12		43
24_100	26			6	42				12	44
24_111	27			6	43	44	12			
3_001	28			32			45	14	13	
34_010	29			6		45		15		46
34_100	30			6	45				15	47
34_111	31			6	46	47	15			
4_001	32						48	17	16	
4_010	33					48		18		16
4_100	34				48				18	17
123_101	35	5,14	5,35	17	20		50		49	
134_110	36	15	5,12	6	21	50				51
123_011	37	5,37	5,10	16		20	52	49		
1234_000	38	5,38	5,38	6				50	52	53
134_011	39	39	5,10	6		53	21	51		
124_110	40	5,15	12	6	52	23				54
124_101	41	5,14	41	6	53		23		54	
24_000	42			6				26	25	55
24_011	43			6		55	25	27		
24_101	44			6	55		26		27	
34_000	45			6				30	29	56
34_011	46			6		56	29	31		
34_101	47			6	56		30		31	
4_000	48							34	33	32
123_111	49	5,3	2,5	4	37	35	57			
1234_100	50	5,30	5,50	6	38				57	58
134_111	51	3	2,5	6	39	58	36			
1234_010	52	5,52	5,25	6		38		57		59
1234_001	53	5,53	5,53	6			38	58	59	
124_111	54	3,5	2	6	59	41	40			
24_001	55			6			42	44	43	
34_001	56			6			45	47	46	
1234_110	57	5,15	5,12	6	52	50				60
1234_101	58	5,14	5,58	6	53		50		60	
1234_011	59	5,59	5,10	6		53	52	60		
1234_111	60	5,3	2,5	6	59	58	57			

the availability of observation means. In this model, all the possible changes are considered. During the creation of the model, four control reconfiguration procedures are adopted which cover all the possible situations of sensor status changes. This model is useful for the studying the reaction of controller to the changes in the sensor status information. It can be also used for the optimization of reconfiguration strategies. The optimization problem cannot be solved directly by this model, but it could be converted to a Markov decision process that provides the optimal reconfiguration strategies. The discussion of the optimization framework will be the subject of our future research.

REFERENCES

Cieslak, R., C. Desclaux, A. S. Fawaz and P. Varaiya (1988). Supervisory control of discrete-event processes with partial observations. *IEEE Transactions on Automatic Control*, **vol. 33**, **no. 3**, pp. 249-260.

Darabi, H., M. A. Jafari and A. L. Buczak (2003). A control switching theory for supervisory control of discrete event systems. *IEEE Transactions on Robotics and Automation*, **vol. 19, no.1**, pp. 131-137.

Haji-Valizadeh, A. K. and A. Loparo (1996). Minimizing the cardinality of an events set for supervisors of discrete-event dynamical systems. *IEEE Transactions on Automatic Control*, **vol. 41, no. 11**, pp. 1579-1593.

Lin, F. and W. A. Wonham (1988). On observability of Discrete event systems. *Information Sciences*, **vol. 44, no. 3,** pp. 173-198.

Liu, J. and H. Darabi (2004). Control Reconfiguration of Discrete Event Systems Controllers with Partial Observation. *IEEE Transactions on Systems, Man and Cybernetics*, Part B, **vol. 34, no. 6**, pp. 2262-2272.

Ramadge, P. J. and W. M. Wonham (1987). Supervisory control of a class of discrete event Processes. *SIAM Journal of Control and Optimization*, **vol. 25, no.1**, pp. 206-230.