World Academy of Science, Engineering and Technology
International Journal of Electrical and Computer Engineering
Vol:5, No:1, 2011

Fail-safe Modeling of Discrete Event Systems using Petri Nets

P. Nazemzadeh, A. Dideban, M. Zareiee

***Abstract*—**In this paper the effect of faults in the elements and parts of discrete event systems is investigated. In the occurrence of faults, some states of the system must be changed and some of them must be forbidden. For this goal, different states of these elements are examined and a model for fail-safe behavior of each state is introduced. Replacing new models of the target elements in the preliminary model by a systematic method, leads to a fail-safe discrete event system.

***Keywords*—**Discrete event systems, Fail-safe, Petri nets, Supervisory control.

I. INTRODUCTION

DISCRETE event dynamic systems have found many applications in synthesis and modeling of industrial processes, networks and transportation. Discrete event systems are systems with discrete states that evolve in response to events. These systems are usually modeled by finite automata. Several methods exist for designing controllers based on automata system models, however these methods often involve exhaustive searches or simulations of system behavior, making them impractical for systems with large numbers of states and events. Petri Net is a very appropriate and useful tool for the study of discrete event systems because of its modeling power and mathematical properties [1],[2]. Modeling discrete event systems with Petri nets may help address some of these difficulties. Petri nets have a simple mathematical representation employing linear matrix algebra making them particularly useful for analysis and design [2],[3].

Fault is anything that changes the behavior of a system such that the system does no longer satisfy its purpose. Fault may occur due to different reasons such as internal event in the system, Change in the environmental conditions, Wrong control action given by the operator and error in the design of the system [4],[5].

Today fault-tolerant is an important issue that is considered in many industrial processes. In the present of a fault tolerant controller, the goal is to diagnose faults when they occur and implement a method to keep the system accomplishing its specifications even in a degraded performance, and if the specifications cannot be accomplished any more, keep the system away from failure and danger states. A key issue is that local faults are prevented from developing into failures that can stop production or cause safety hazards [6].

After fault is detected, usually the controller is redesigned or reconfigured to keep the performance of the system. Authors of [7] introduced a framework for fault tolerant supervisory control of discrete event systems and defined two different specifications for non faulty and overall plant. Reference [8] suggested a method to switching the controller from a supervisory control system to a new one. So on, when a fault occurs, we can switch the controller to a new controller that has designed for that faulty state. A method for model reconfiguration was introduced by Iordache and Antsaklis[9]. Miagi and Riascos used knowledge based model and neural networks to define the necessary treatment for each fault [5].

One of the most important issues of fault-tolerant control design is to keep the safety of the system. Keeping the system in a safe mode after the occurrence of a fault, requires necessary knowledge about devices and elements of the system. In this paper, situations of faulty statuses of target elements are presented by Petri net models. These places indicate the state of target elements when fault occurs. If the models of target elements are replaced by these new models, no faulty element and part of the system can be activated anymore. So, the fail-safe model is achieved.

This paper is organized as follows. Section II introduces the preliminaries and basics of Petri Nets. Section III presents different states of the system when a fault is occurred and obtains their models to achieve a fail-safe model. This method is clarified by an example in section IV. The conclusion of this paper is given in section V.

II. PRELIMINARIES

A. Petri Nets

In this section we define some basics of Petri Nets and its properties that are useful in this paper. We suppose that the reader is familiar to the basic of Petri Nets. For more details refer to [10], [11].

A Petri net is a 5-tuple $\Re = \{P, T, W^-, W^+, Mo\}$ where P is the set of places, T is the set of transitions, W^-: $(P \times T) \rightarrow N$ is the input function, W^+: $(T \times P) \rightarrow N$ is the output function and M_0 is the initial marking. The incidence matrix W is calculated by $W = W^+ - W^-$.

A transition is called enabled, if all of its input places are marked.

P. Nazemzadeh is with the Department of electrical and computer eng. Semnan university, Semnan, Iran (e-mail: p_nazemzade@semnan.ac.ir).

A. Dideban is with the Department of electrical and computer eng. Semnan university, Semnan, Iran (corresponding author to provide e-mail: adideban@semnan.ac.ir).

M. Zareiee is with the Department of electrical and computer eng. Semnan university, Semnan, Iran (e-mail: meisamzareiee@gmail.com)

Definition 1. [10]The function $M : P \rightarrow N$ is called marking. M is an n-member vector that introduces the state of the system at each time by representing the existence of token in each place (n is the number of places). *M* is the vector of the set $\{0,1\}^N$ □

Definition 2. [12] The function Support(*X*) of a vector $X \in \{0,1\}^N$ is:

Support(*X*) = the set of marked places in the marking *X* □

Definition3. [16]Constraints are linear inequalities on the PN markings and are written in the form of:

$L.M \leq b$

Where M is the marking vector, $L \in Z^{n_c \times n}, b \in Z^{n_c}$, n is the number of places of the Petri net model, and n_c is the number of constraints. □

There are some methods to apply constraints in a Petri Net. One of the most useful methods is using use p-invariant properties [3],[13]. In this method we need to add one place to the system for applying each constraint.

Theorem1.[3] the Petri net controller W_c for applying a set of n_c constraints is as follows (if $b - L.M \geq 0$)

$W_c = -L.W$

$M_{p_0} = b - L.M_0$

B. Fail-safe

In this section safety and fail-safe systems and the goal of using these systems are defined. For more details refer to [4].

Definition4. A safety system is a part of control that protects a technological system from permanent damage. A safety system can not go to a danger state. □

*Definition5***.** A fail-safe system is a system which can keep the safety of overall system and its components after the occurrence of faults. □

In the occurrence of faults, some states of the system may cause a failure in the faulty device or the overall plant. In this situation, these states must be forbidden to achieve a fail-safe system. Thus, if the system is in these states, we need to change the state of the system immediately and if not, the system must be prevented from going to one of these states. A fail-safe controller must avoid danger states to keep the safety of the system.

III. FAIL-SAFE MODEL

When a fault occurs, the action of some elements must be stopped and prevented to keep the safety of the system. These elements are called "target elements". In this paper we investigate different statuses of target elements and introduce a Petri net model that can reach the fail-safe properties.

In a Petri net model, places describe the various statuses of each device and element of the system. So, for each desired fault, we must obtain the places which describe the forbidden status of target elements and forbid the activation of these places when the fault has occurred. We call these places as "the target places" and show them by P_T.

When a fault occurs, two different situations are possible for each target place. One situation is for the cases in which the target place is marked and the other is for the cases in which the target place is not marked.

For each target place, we need to determine a model for these two situations, called "faulty element model". After that, the fail-safe model can be achieved by replacing each target place with its faulty element model.

A. Faulty element model of marked target places

In this situation, the controller must take the token of the target place away to stop its action. This token must add to another place. For this purpose, two cases will be possible in respect to the redundancy existence.

Case 1. A redundant element exists for the target element.

In this case, the system must be able to activate the redundant element. Thus the token of the target place needs to be removed from it and added to the place describing the same status of the redundant element. The Petri Net model of this case is shown in fig. 1.

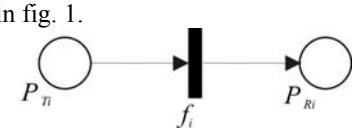

Fig. 1. The faulty element model for redundancy activation of the situation *A*

In fig. 1 P_{Ti} is the target place and P_{Ri} is its corresponding redundant place. f_i is a transition which fires when the fault number *i* occurs.

Case 2. There is no redundancy for the target element

In this case, the token of the target place must be removed after the occurrence of fault and added again when the fault is recovered. For this goal we need to accumulate the token of the target place in an additional place. So there must be a new place showing the occurrence of fault while the target element was activated. This place must get the token of target place as soon as fault is diagnosed and give it back to the target place, when the fault is recovered. Fig. 2 Shows the Petri net model of this case.

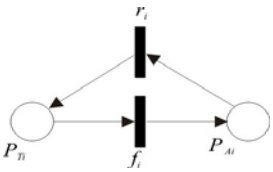

Fig. 2. The faulty element model of situation *A* when no redundancy exists

In this case, the new place keeps the information of the target place. So, we call it the acummulator place. r_i is the transition of the fault repairing event. It fires after the fault repairs. The output arc of r_i could be changed in different systems and is depend on the design and requirements of the system. For example in some systems after the repair of fault, we need to restart the system from its initial state or go to some previous statuses of the target element. Thus the designer can change the output places of the transition labeled by event r_i.

Note that there may be more than one target place for each fault. In this situation a same model must be determined for each target place.

B. Faulty element model of unmarked target places

In this situation, if the fault occurs, the activation of the target place must be avoided. So, we need to synthesize a supervisor which can prevent the target place to being activated, when the system is faulty. For this subject, we first define a model which shows the faulty status of the system corresponding to the target place. This model has a place which must be marked as soon as the fault occurrence is diagnosed and unmarked when that fault is recovered. So, this model is as fig. 3. In this model the place P_{Fi} describes that the fault has occurred while the target element had not been activated.

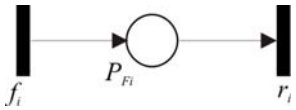

Fig. 3. faulty status description for unmarked target place

Now we must design a supervisor which can forbid the activation of target place when the place P_{Fi} is marked.

Consider part of the main model consists of the target place and its input and output transitions. This sub-model is shown in fig. 4. ${}^{\circ}P_{Ti}$ is the set of input transitions of the target place and $P_{Ti}{}^{\circ}$ is the set of its output transitions.

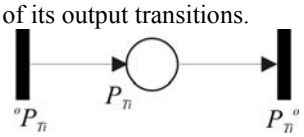

Fig. 4. the main model of the target place

If the fault occurs while the target place is marked, the place P_{Fi} does not need to be marked, because the system is in situation *A*. But if the fault occurs while P_{Ti} is not marked, P_{Fi} needs to be marked and must not allow the target place to be activated. This leads to a new constraint in the system as follows:

$m_{Fi}+m_{Ti} \le 1$

Using the method introduced in [3],[13] we have:

$$W = \begin{bmatrix} & {}^{\circ}P_{Ti} & P_{Ti}{}^{\circ} & f_i & r_i \\ P_{Ti} & 1 & -1 & 0 & 0 \\ P_{Fi} & 0 & 0 & 1 & -1 \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

$$W_c = -L.W = \begin{bmatrix} -1 & 1 & -1 & 1 \end{bmatrix}$$

Fig. 5 shows the controlled Petri net model for this situation.

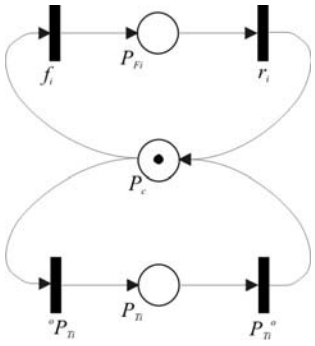

Fig. 5 The faulty element model of situation *B*

C. Faulty element model of the redundant elements

A redundant element may also be a target element. While the redundant elements are activated because of the disability of the major elements, a major element could not be a redundance of its redundant element. So, the redundant elements behave as like as the elements of *case 2*. Note that we can have a redundancy for a redundant element, if there are more than one redundant elements for the major element.

After the faulty element models of all target places are determined, the fail-safe model can be achieved by changing the model of each target place with its faulty element model. In this changing, all of the places and transitions of the main model are remained and some new places and transitions as introduced in section III are added to the system.

Definition6. A fail-safe Petri net is a 5-tuple $\Re_{fs}=\{P_{fs}, T_{fs}, W_{fs}^{-}, W_{fs}^{+}, Mo_{fs}\}$. □

$P_{fs} = P \cup P_F \cup P_A \cup P_c$, where P is the set of places of the initial model, $P_T \subseteq P$, P_F is the set of places that each of them shows the faulty status of a target element while the target place was not marked. P_A is the set of places that each of them shows the faulty status of a target element while the target place was marked and P_c is the set of control places synthesized for preventing the activation of target places.

$T_{fs} = T \cup f \cup r$, f is the set of fault events and r is the set of repair events.

$W_{fs}= W_{fs}^{+}-W_{fs}^{-}$ is the fail-safe incidence function and determined by the combination of the incidence matrix of the initial model and the faulty element models.

Mo_{fs} is the initial state and is determined as follows :

$$\mathrm{M}_{0_{\mathrm{fs}}} = \begin{cases} M_0 & \forall p_i \in P \\ 0 & \forall p_i \in P_F \\ 0 & \forall p_i \in P_A \\ 1 & \forall p_i \in P_c \end{cases}$$

IV. A PRACTICAL EXAMPLE

Consider a reactor system with one tank, a valve for chemical materials, a motor which pumps the input liquid, a mixer motor and an output valve. The tank has two level detectors for high and low levels. When the low level switch is acted the input pump M_{in} turns on to transfer the liquid to the tank until the high level is not acted. After the action of

high level switch, the valve of chemical materials V_c must be turned on for 10 seconds. Then the motor turns on for 2 minutes to merge the liquid with chemical materials. After that the output valve turns on and transfers produced materials of the tank to the output valve until the acting of the low level switch. Fig. 6 shows the Petri net model of this system.

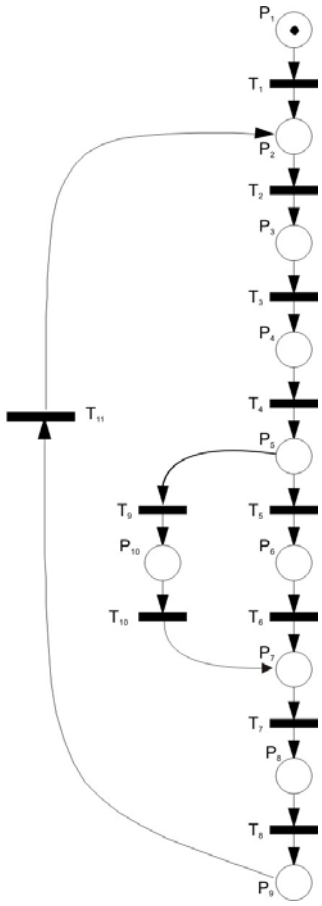

Fig. 6. Main model of the example

The places description and events action of the model shown in fig.6 are introduced in table 1.

Now we want to consider two different faults in the system. (a) A Leakage in the tank and (b) failure in the mixer motor. First the fail-safe model for each of the faults must be determined. Synchronizing fail-safe models and the main model cause to the fault-tolerant controller.

a. Leakage (or break) in the tank :

If a leakage is detected, any new inputs to the tank must be prevented while the tank is not repaired. As it can be seen in the example, the inputs of tank are from the input pump and the valve V_C. Thus, the target elements are the "on" statuses of pump and valve V_C and the target places are places P_2 and P_4, and because there is no redundancy for these target elements, we need to imply *case2* for marked target places. The faulty element model of leakage in the tank is shown in fig. 7.

TABLE I
PLACES DISCRIPTION AND TRANSITIONS EVENTS OF THE EXAMPLE

Place	Description	Transition	event
P_1	System off	T_1	Start key
P_2	input pump on	T_2	The tank is in high level
P_3	Wait for chemical materials	T_3	Start command of the chemical valve
P_4	chemical valve on	T_4	10 seconds
P_5	Wait for mixing	T_5	Start command of the mixer motor
P_6	The mixer motor on	T_6	120 seconds
P_7	Wait for output valve	T_7	Start command of the output valve
P_8	The output valve on	T_8	Tank is in low level
P_9	System is ready for restarting its operation	T_9	Start command of the mixer motor +1 seconds
P_{10}	The redundant mixer motor on	T_{10}	120 seconds
		T_{11}	Restart command of the process

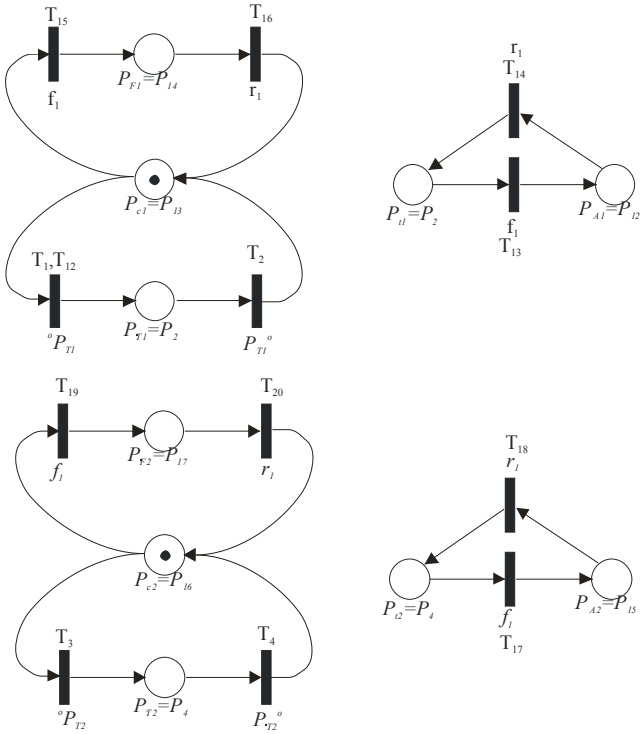

Fig. 7 the faulty element model for leakage in the tank

b. Failure in the mixer motor

In this situation, the activation of the mixer motor is forbidden. Thus, the target element is the "ON status" of motor and the target place is place P_6. A redundant motor exists and so, the *case1* of the marked target places needs to be applied. Figure 8 shows the faulty element model for the failure in the mixer motor.

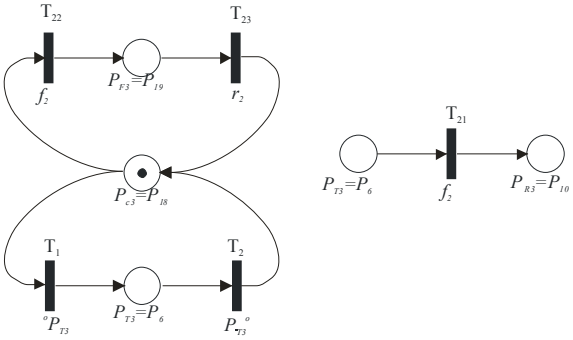

Fig. 8 the faulty element model for a fault in the mixer motor

Now we need to determine the faulty element model of the redundant motor. As introduced in section III, for marked redundant place, *case2* must be applied. This model is shown in fig. 9.

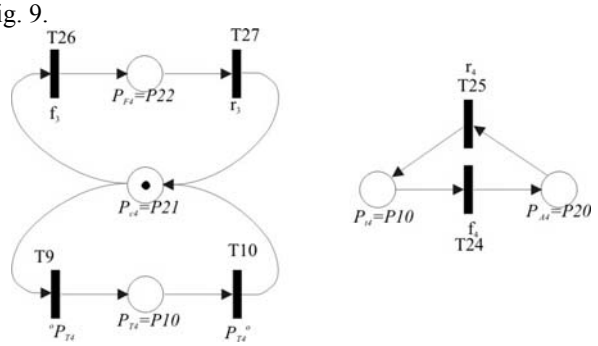

Fig. 9 the faulty element model for the redundant motor

As introduced in section III, we can achieve the fail-safe controller by synchronizing the fail-safe models and the main model. The final controller for this example is shown in fig. 10.

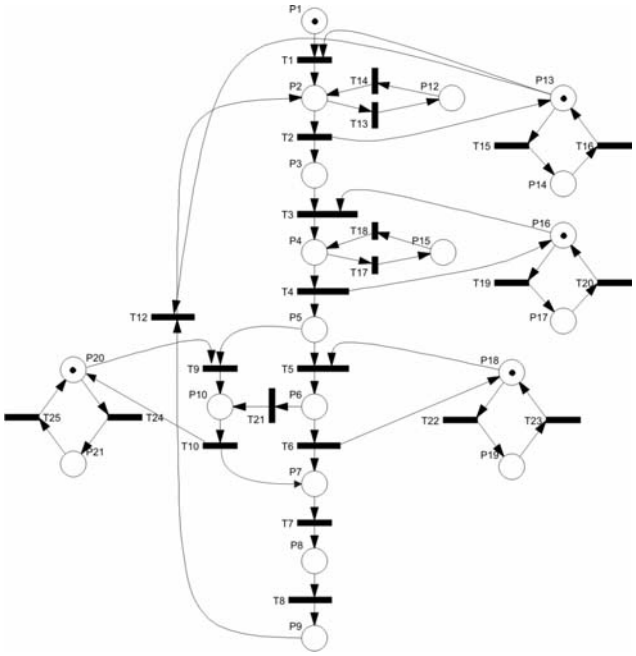

Fig. 10 the fail-safe Petri net model of the example

Using the model shown in fig. 9 keeps the safety of the system in the occurrence of a local fault in the tank or the mixer main motor. For example, consider that the system is in the state $support(M)=P_5P_{13}P_{16}P_{18}$. in this state all of the devices are off and the system is waiting for the start command of the mixer. Consider that the motor become faulty before the mixer start command. In this situation the transition T_{22} fires and the system is going to the state $support(M)=P_5P_{13}P_{16}P_{19}$. So the transition T_5 could not fire and the transition T_9 fires one second after the start command of the mixer and activates the redundant motor. So, after the start command of the mixer the system will go to the state $support(M)=P_{10}P_{13}P_{16}P_{19}$ instead of the state $support(M)=P_6P_{13}P_{16}P_{19}$. The mixer will work by the redundant motor while the main motor is faulty and the safety of the main motor and system is remained.

V. Conclusion

A model based system reconfiguration for synthesizing a fail-safe controller for discrete event systems with controllable events was introduced. In this method we added a new model to the Petri Net model of the system called fault model. After the synchronized composition of this model with the main model, the future activations of the faulty element of the system are forbidden. So, that part of the system is prevented from more failures or damages. The system is then tries to enforce the duties and specifications of the forbidden part by the other parts and elements.

This method is useful for discrete event systems with controllable events and transitions. If the system has uncontrollable transitions, but all of the input transitions of the target places are controllable, there are no changes in the design of the fault-tolerant system. But if there is at least one uncontrollable transition in the input transitions of the target places, this method will not be a complete method. Our future work is to expand this method to the systems with uncontrollable events.

References

[1] B.H. Krogh, and L.E. Holloway, "*Synthesis of Feedback Control Logic for Discrete Manufacturing Systems",* Automatica, Vol. 27, No. 7, pp. 641-651, 1991.

[2] A. Giua, and F. Dicesare, "*supervisory design using Petri Nets*" *IEEE proceeding of the 30th conference on decision and control*, England 1991.

[3] C. Yamalidou, J., Moody, M. Lemmon, and P. Antsaklis, "*Feedback Control of Petri Nets Based on Place Invariants*", Automatica, Vol. 32, No. 1, pp. 15-28, 1996.

[4] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, "*Diagnosis and Fault-Tolerant Control*". Springer-Verlag Berlin, 2006.

[5] P.E. Miyagi, and L.A.M. Riascos, "*Modeling and analysis of fault-tolerant systems for machining operations based on Petri Nets*", Control Engineering Practice, vol.14, pp. 397-408, 2006.

[6] M. Blanke, C.W. Frei, F. Kraus, R.J. Patton, and M. Staroswiecki. What is fault-tolerant control? In *Preprints of 4th IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes, SAFEPROCESS' 2000*, pages 40–51, Budapest, Hungary, 2000.

[7] Q. Wen, R. Kumar, J. Huang, and H. Liu, "*A framework for fault-tolerant supervisory control of discrete event systems*," IEEE

Transactions on Automatic Control, Vol. 53, No. 8, pp. 1839-1849, 2008.

[8] H. Darabi, M.A. Jafari, and A.L. Buczak, "*A control switching theory for supervisory control of discrete event systems*", IEEE Transactions on Robotics and Automation, Vol. 19, pp. 131–137, 2003.

[9] M.V. Iordache, and P.J. Antsaklis, "*Resilience to failure and reconfigurations in the supervision based on place invariants*", in Proc. American Control Conference, pp. 4477–4482, 2004.

[10] David, R., and Alla, H., "*Discrete, Continuous, and Hybrid Petri Nets*", Springer, 2005.

[11] Hrúz, B. and Zhou, M.C., "*Modeling and Control of Discrete-event Dynamic Systems with Petri Nets and other tool*", springer, 2007.

[12] Dideban, A., and Alla, H., "*Reduction of Constraints for Controller Synthesis based on Safe Petri Nets*", Automatica, Vol.44, No. 7, pp. 1697-1706, 2008.

[13] Moody J.O., and Antsaklis P.J."*Petri Net Supervisors for DES with Uncontrollable and Unobservable Transitions*", IEEE Transactions on Automatic Control, Vol.45, No.3, pp. 462-476, 2000.