

# Chapter 2

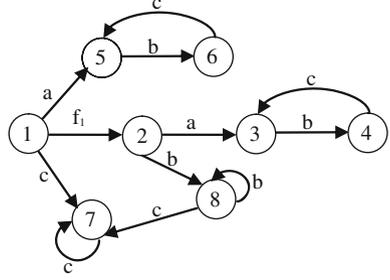
## Centralized Diagnosis of Discrete Event Systems

### 2.1 Introduction

As we have seen in Chap. 1, centralized diagnosis requires one centralized model of the targeted system associated with one centralized diagnoser (see Fig. 1.7a). The latter collects observations about the system behaviors in one central point. Then, it treats these observations in order to make decision about the occurrence of a fault and its responsible elements (e.g., actuator, sensor, etc.). Examples of a centralized diagnosis structure can be found in [22, 23, 27, 39] and the references therein.

This chapter will focus on two significant event-based approaches of centralized diagnosis of DES: diagnoser and supervision pattern approaches. In these approaches, the system is represented (modeled) by an automaton in order to achieve or to solve the problem of diagnosis. The events, which change the system state, are divided into two disjoint sets: observable and unobservable events. The faults are considered as unobservable events. Thus, the automaton is nondeterministic with unobservable transitions. All the information relevant to the diagnosis problem of a system is captured in the framework of events generated by this system. Therefore, diagnosis problem is solved by observing the set of observable event sequences or strings (words). In other words, the occurrence, if any, of failure events, is inferred using the set of generated words containing only observable events.

Model  $G$ , generating formal language  $L$ , can represent either the normal and faulty behaviors of the system (diagnoser approach) or only specified faulty behaviors (supervision patterns). In the first case, the nondeterministic automaton  $G$ , representing the system under consideration, is converted into deterministic one by considering only the observable events (observable transitions). The resulting deterministic automaton is called observer  $Obs(G)$ . Each node of  $Obs(G)$  contains the states that the system can be in in response to the occurrence of an observable event sequence. Then, the diagnosis problem is solved by building a diagnoser  $Diag(G)$ . The latter is an observer but information about the occurrence or not of each fault is added to each of its nodes. This information is represented by a label indicating whether the system is fault-free ( $N$ ) or fault  $f$ , belonging to fault partition  $\Sigma_{F_i}$  occurred. In the second case, a faulty behavior in response to the occurrence of fault  $f$  is modeled as a set of partial observable trajectories (traces or event sequences) that one wants

**Fig. 2.1** System model  $G$ 

to recognize their occurrence. The diagnosis of the occurrence of  $f$  is achieved by matching between the real behavior of the system and the compiled faulty behavior.

In both cases, the following assumptions hold:

- (A1) Language  $L$  generated by model  $G$  is live. This means that each state of  $G$  has a transition.
- (A2) There does not exist in  $G$  any cycle of unobservable events. A2 ensures that  $G$  does not contain sequences of unobservable events whose length can be infinite.
- (A3) Faults cause a distinct change in the system status but do not necessarily bring the system to a halt.

## 2.2 Diagnosability Notion

The diagnosability notion is based on the fact that a system is diagnosable if and only if any pair of faulty/non-faulty behaviors can be distinguished by their projections to observable behaviors. Before defining the necessary and sufficient conditions for a system to be diagnosable, let us consider the following definitions.

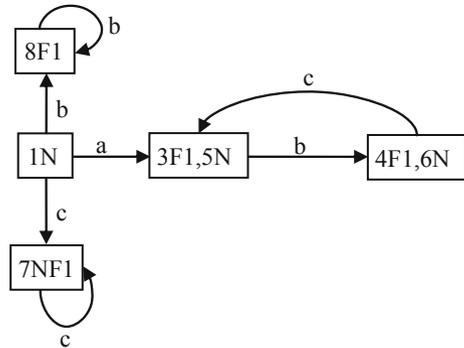
**Definition 2.1** Language  $L$  generated by system model  $G$  is the set of all the event sequences or trajectories  $u$  that the system can execute.

**Example 2.1** Let us take the example of model  $G$  represented in Fig. 2.1. The language generated by  $G$  is:  $L = \{f_1 a(bc)^*, a(bc)^*, f_1 bb^*, cc^*, f_1 cc^*\}$ .

**Definition 2.2** Let  $\Sigma$  be the set of all events that can be generated by a system  $S$ . Let  $\Sigma^*$  denote the set of all event sequences that can be formed using the events in  $\Sigma$ . Let  $\Sigma_o$  denotes the set of observable events generated by system  $S$ . Then,  $\Sigma_o^*$  is the set of all event sequences that can be formed using observable events in  $\Sigma_o$ . The projection function  $P : \Sigma^* \rightarrow \Sigma_o^*$  allows erasing all unobservable events in an event sequence.

**Example 2.2** Let us take the example of Fig. 2.1 where:  $\Sigma = \{a, b, c, f_1\}$  and  $\Sigma_o = \{a, b, c\}$ . The projection  $P(u)$  of event sequence  $u = f_1 a(bc)^* \in \Sigma^*$  is  $a(bc)^* \in \Sigma_o^*$  since  $f_1$  is unobservable event.

**Fig. 2.2** Diagnoser for the example of Fig. 2.1



**Definition 2.3** A diagnoser state  $q_D$  contains a set of pairs  $(x, l)$  where  $x$  is a state of system model  $G$  and  $l$  is a fault partition label or normal label  $N$ . Diagnoser state  $q_D$  is said to be  $F_i$ -certain state if for all the pairs  $(x, l)$ ,  $l$  is equal to  $F_i$ .

Diagnoser state  $q_D$  is said to be  $F_i$ -uncertain if it contains two pairs  $(x_1, l_1)$  and  $(x_2, l_2)$  where  $l_1 = F_i$  and  $l_2 \neq F_i$ .

Diagnoser state  $q_D$  is ambiguous if there is at least two pairs  $(x_1, l_1)$  and  $(x_2, l_2)$  where  $x_1 = x_2$  and  $l_1 = F_i$  and  $l_2 \neq F_i$ .

**Example 2.3** Let us take the example of Fig. 2.1. Figure 2.2 shows the corresponding diagnoser  $D$ . Diagnoser state  $q_D = (8F1)$  is  $F_1$ -certain state since it contains only the fault label  $F_1$ . Diagnoser state  $q_D = (3F1,5N)$  is  $F_1$ -uncertain state since it contains two states with fault label  $F_1$  and the normal label  $N$ . Diagnoser state  $q_D = (7F1,7N)$  is  $F_1$ -ambiguous state since it contains the state model 7 which has the normal and  $F_1$  labels.

**Definition 2.4**  $F_i$ -indeterminate cycle in diagnoser  $D$  is a cycle composed exclusively of  $F_i$ -uncertain states. It indicates the presence in  $L$  of two event sequences  $u_1$  and  $u_2$  such that they both have the same observable projection,  $P(u_1) = P(u_2)$ , and  $u_1$  contains a failure event while  $u_2$  does not.

**Example 2.4** Cycle  $(bc)^*$  in Fig. 2.2 is  $F_1$ -indeterminate cycle since it corresponds to two cycles in the model  $G$  of Fig 2.1; the first cycle is formed by states with normal label and the second cycle by states with  $F_1$  fault label.

**Definition 2.5** Two conditions are required for a system to be  $F_i$  diagnosable [39]:

1. after the occurrence of fault  $f$  belonging to the fault partition of label  $F_i$ , the diagnoser must visit or reach an  $F_i$ -certain diagnoser state within a finite number of observable events,
2. the diagnoser must not contain any indeterminate cycle.

In the next, the diagnosability notion will be studied using several examples.

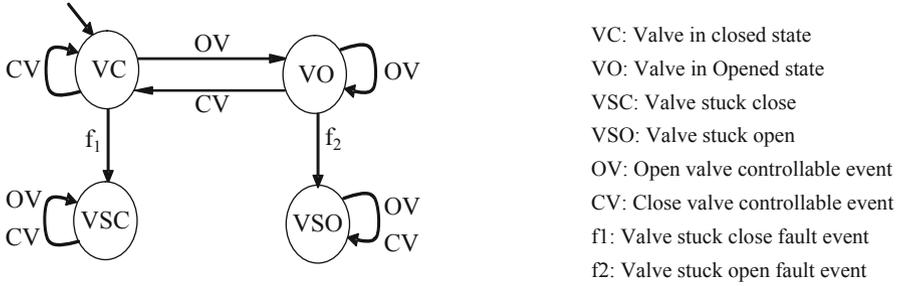


Fig. 2.3 Normal and faulty behaviors for the valve model

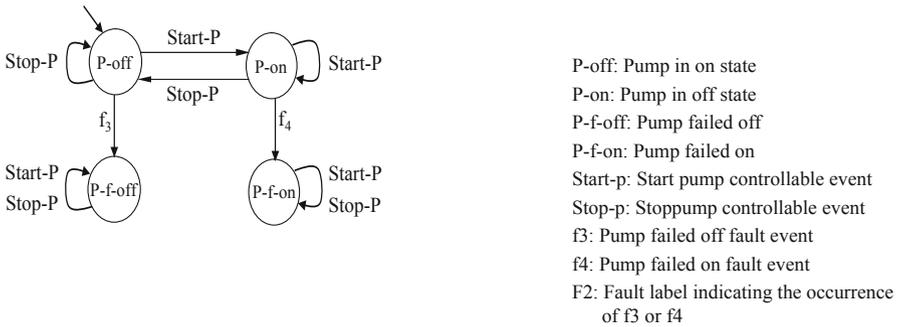


Fig. 2.4 Normal and faulty behaviors for the pump model

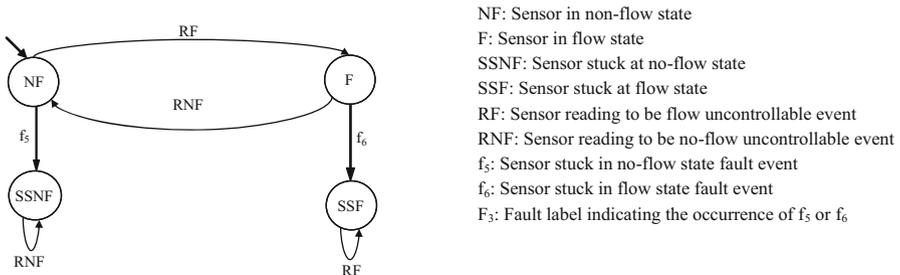


Fig. 2.5 Normal and faulty behaviors for the flow sensor model

### 2.3 Diagnoser Approach

In Diagnoser approach [39], the diagnosis problem is the task of assigning to each observed string (word) of events a diagnosis state with one of the following status: “normal”, “faulty” or “uncertain”. The uncertainty can be reduced by continuing to make observations.

The diagnoser approach works as follows. The system to be diagnosed is supposed to be composed of  $n$  individual components. Typically, these components consist of equipment (actuators and sensors) and controllers. However, they can also represent

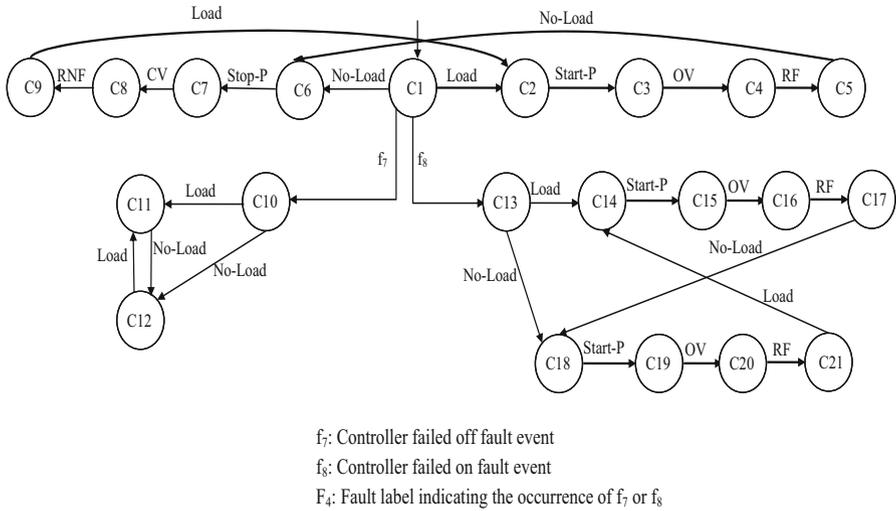
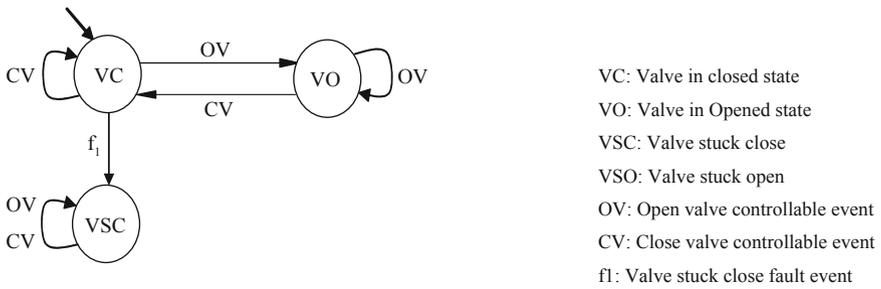
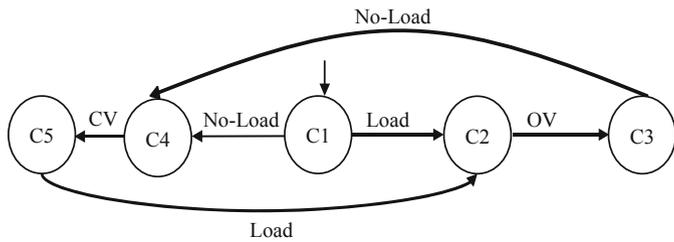


Fig. 2.6 Normal and faulty behaviors for the controller model



a Model of valve  $V$



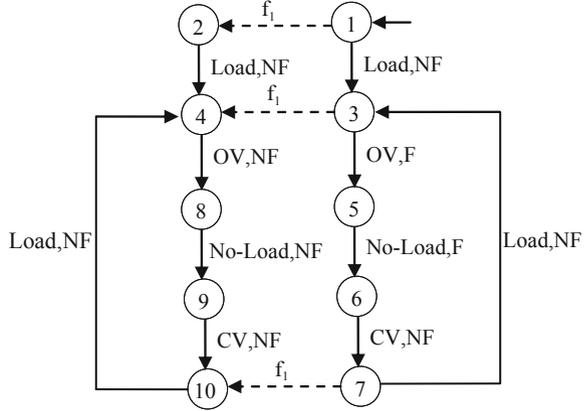
b Controller model

Fig. 2.7 Components models for Example 2.6

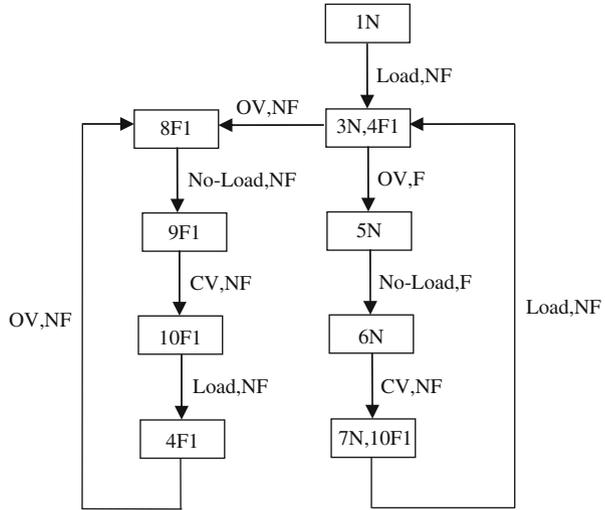
the process itself (e.g., a tank). The construction of the diagnoser is based on the following steps [39]:

- Step 1:** Build the Finite State Machine (FSM) models for the system components. The model representing the normal behavior of each component is firstly built.

**Fig. 2.8** Composite model of the example of Fig. 2.7



**Fig. 2.9** Diagnoser for the example of Fig. 2.8



Then, the failed behaviors corresponding to the occurrence of a predefined set of faults are integrated in the component normal FSM model.

2. **Step 2:** Obtain the global model of the system by applying the standard synchronous composition operator between the components individual models.
3. **Step 3:** Generate the global sensor map that lists the discrete sensor readings for each state of the global model built in step 2. Convert the sensor readings into observable event framework as follows. Reading of the sensor output is considered as an observable event after immediately the execution of a control command. Each transition of the global model (built in step 2) will be renamed by adding the corresponding sensor reading event to the original observable event. In the case that the execution of a command leads to a change in the sensor reading, this sensor change is considered as an observable event. Thus, a transition associated with the sensor change reading is added to the model.

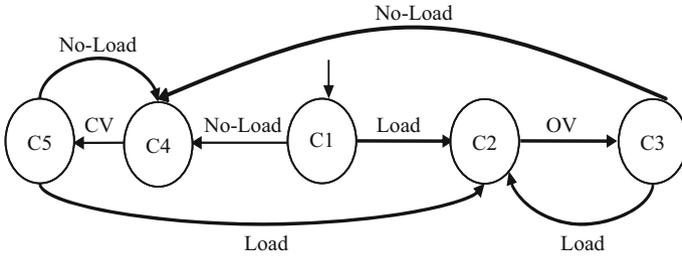


Fig. 2.10 New controller model for the example of Fig. 2.7

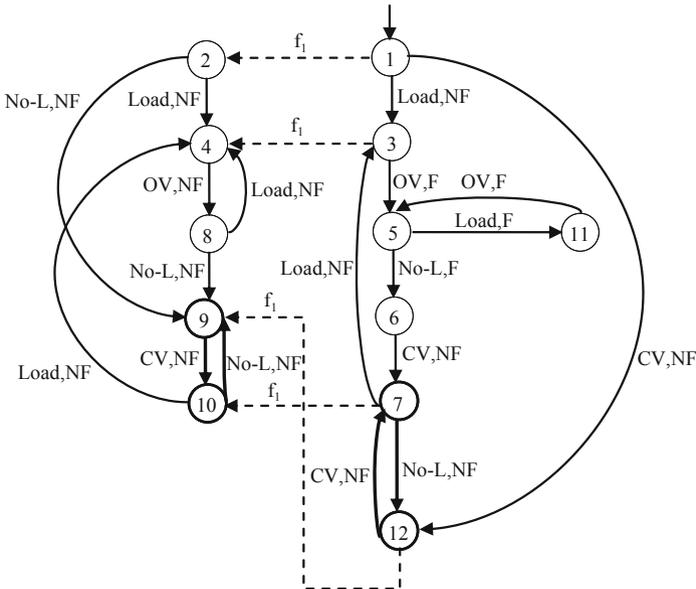


Fig. 2.11 Composite model for Example 2.7

4. **Step 4:** Construct the diagnoser based on the use of the global model. The diagnoser contains only observable events. Each of its states includes one or more of labels indicating a fault-free ( $N$ ) or the occurrence of faults belonging to predefined fault partitions:  $\Pi_{F_i}, \forall i \in \{1, \dots, d\}$ .

In the next, this approach is explained using a simple example of a valve, a pump, a discrete flow sensor and a controller.

**Example 2.5** Let us take the following example extracted from [39]. This example consists of a pump, a valve and a controller. Let us suppose that the system is equipped with one sensor to indicate the presence of a flow at the output of the valve. This sensor has one of the two outputs:  $F$  to indicate the presence of a flow and  $NF$  to indicate no flow. Let us consider the behavior of the valve  $V$ . The valve can be in one of two different states: closed ‘ $VC$ ’ and opened ‘ $VO$ ’. Let us assume that the valve can fail due to a stuck-at-on fault or to a stuck-at-off fault. ‘ $VSO$ ’ represents the state

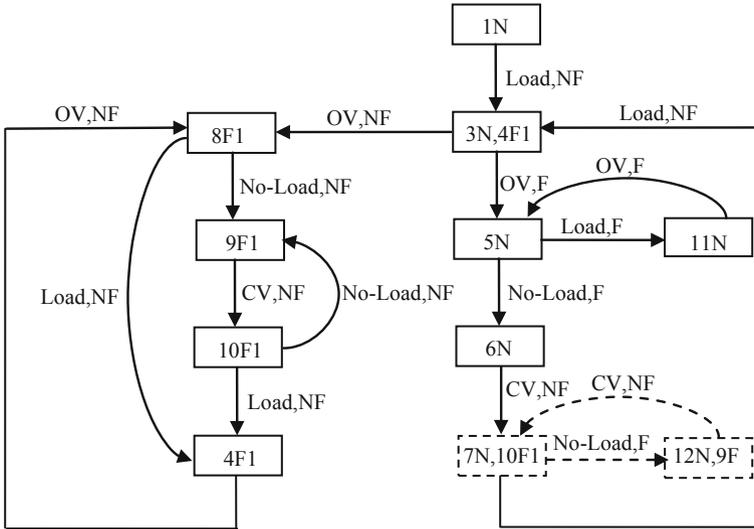
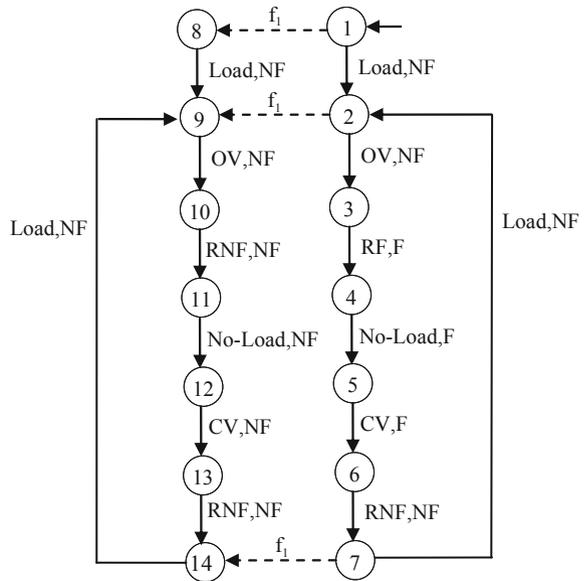


Fig. 2.12 Diagnoser for the example of Fig. 2.11

Fig. 2.13 Composite model for Example 2.8



of valve stuck-open and ‘VSC’ represents the state of valve stuck-closed. In Fig 2.3 the FSM modeling the normal and faulty behaviors of valve  $V$  is depicted. The system events are: ‘OV’ that represents the command of opening the valve, ‘CV’ that represents the command of closing the valve,  $f_1$  and  $f_2$  that represent respectively the stuck-close and stuck-open events. In a normal behavior, the valve is in ‘VC’ and can change its state when one of commands ‘OV’ and ‘CV’ is sent by the controller. The valve may fail. Either a stuck-close fault ( $f_1$ ) or a stuck-open fault ( $f_2$ ) can occur.

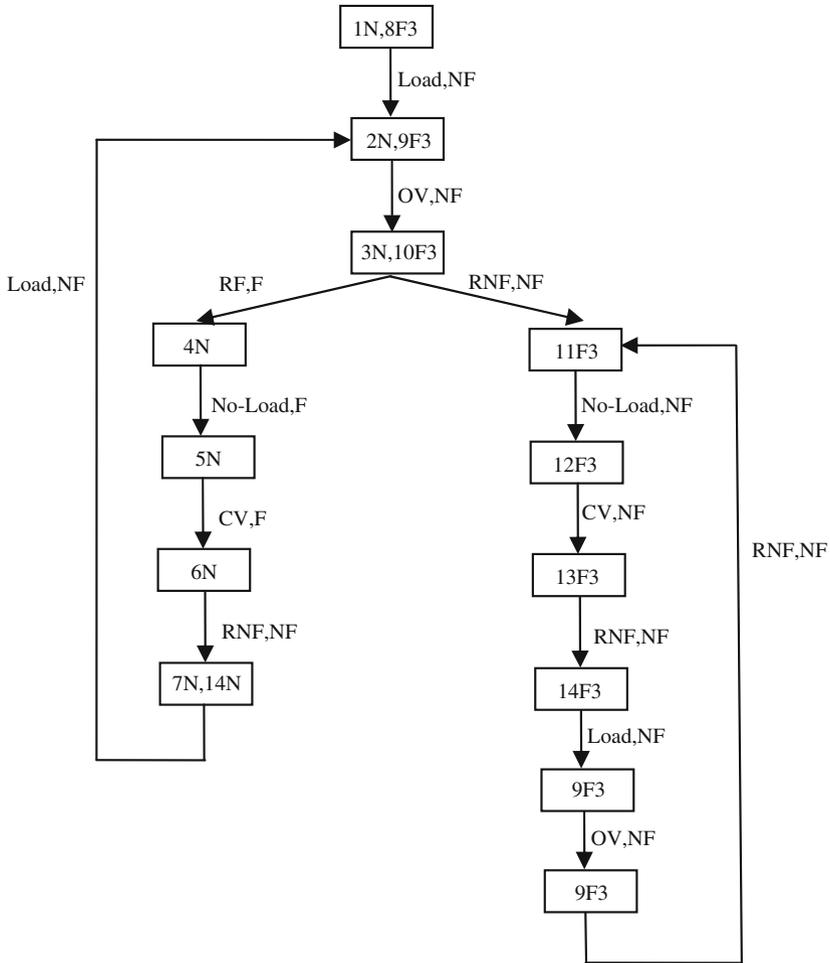


Fig. 2.14 Diagnoser for the example of Fig. 2.13

After the occurrence of one of these faults, both commands (open and/or close the valve) do not change the state of the valve. Events ‘OV’ and ‘CV’ are controllable and thus observable events, while  $f_1$  and  $f_2$  are faulty and unobservable events and we have to infer their occurrence on the basis of observable event sequences. Figure 2.4 and 2.5 show, respectively, the FSM modeling the normal and faulty behaviors of the pump and the flow sensor. Figure 2.6 shows the command issued by the controller. In normal operation mode, when there is a load in the system, the controller responds by starting the pump and opening the valve. When there is no load anymore in the system, the controller stops the pump and closes the valve. When the controller fails off (i.e., when event  $f_7$  occurs), it does not sense the presence of load on the system and therefore it does not send any of the above commands. While, when the controller

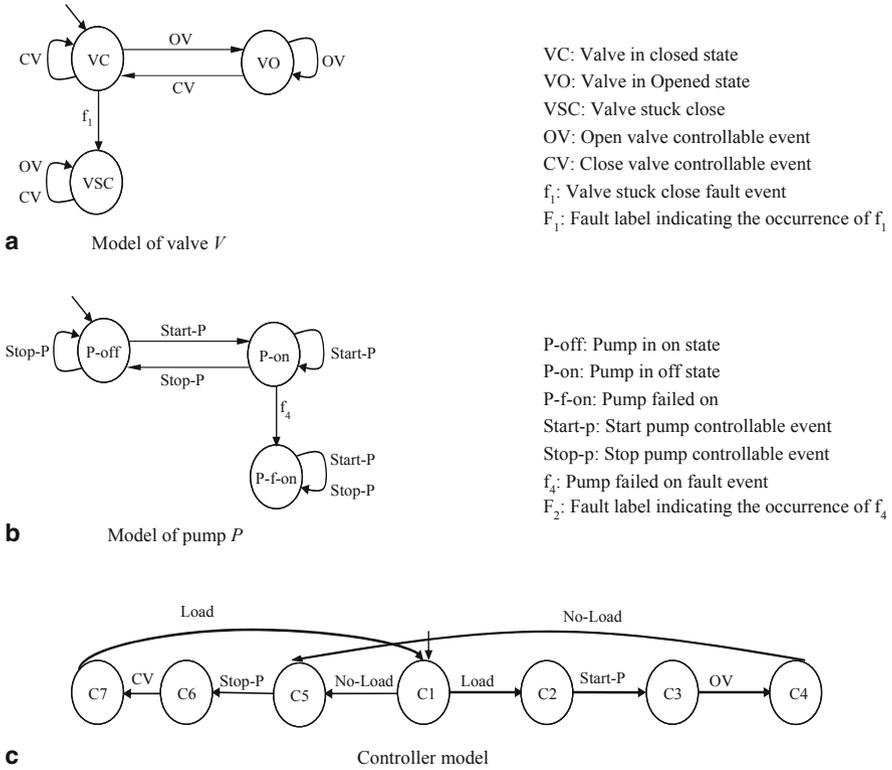


Fig. 2.15 Components and controller models for Example 2.9

fails on (when  $f_8$  occurs), it assumes a presence of load and consequently it issues the command sequence  $\langle Start-P \rangle \langle OV \rangle$  regardless of whether a load is actually present or not. It is supposed that the controller does not fail during operation. If it does fail, the fault occurs at the start of operation.

**Example 2.6** In order to facilitate the example of pump-valve (Example 2.5), let us suppose that only the valve can fail in the stuck close failure mode (VSC). Therefore, failure state 'VSO' in the valve model of Fig. 2.3 can be removed. Let us suppose that the pump is always in its on state. The pump model can thus be removed. Since, we are not interested in diagnosing sensor faults; the model of the sensor can be also removed as well as its events from the controller model. Therefore, we have the following models for the valve and the controller shown in Fig. 2.7.

The composite model (obtained by the synchronous composition of the system component models of Fig. 2.7) is depicted in Fig. 2.8. The sensor reading  $\{NF, F\}$ , considered as an observable event, will be added to this composite model by associating it to each command event. To achieve that, the flow sensor map must be constructed. The sensor map helps to associate to each observable event (typically



<http://www.springer.com/978-1-4614-0030-1>

Discrete Event Systems

Diagnosis and Diagnosability

Sayed-Mouchaweh, M.

2014, VII, 69 p. 58 illus., 4 illus. in color., Softcover

ISBN: 978-1-4614-0030-1