Chapter 6

Transformation of Deterministic Models into State space Models for Safety analysis of safety Critical Systems: A Case Study of Nuclear Power Plant

The state space models has been successfully applied in engineering, statistics, computer science and economics to solve a broad range of dynamical systems problems; like safety analysis, reliability analysis, performability analysis, etc. However, embedding the complete and accurate system requirements in such models is quite challenging. Analyzing model with incomplete or inaccurate requirements gives inaccurate results. UML is a proven and easy approach to capture all the system requirements. This Chapter proposes a methodology to transform the UML

model into the state space model. The resultant model will embed all the system requirements and hence can be used to analyze the critical attributes of the systems. The methodology is validated on 13 sets of operational profile of different safety critical systems (SCSs) of a Nuclear Power Plant and shown on Reactor Core Isolation Cooling System (RCICS).

6.1 Introduction

Dynamical analysis of the system is used to describe the behavior of complex systems and hence is useful for verification and validation. Validation ensures the conformance of functional and non-functional requirements. Non-functional requirements are often expensive but add quality. Non-functional requirements are constraints on the system design. They may arise from user requirements, technical disciplines or the external environment. They are often "ilities", can be divided into product or support constraints and include reliability, safety, security, etc. Therefore, SCSs need to be validated with respect to the non-functional requirements.

State space models are widely used to analyze non-functional requirements. However, these models are not easily understood by all stakeholders and creation of these models requires technical expertise. UML can capture all the requirements and be easily understood by all the stakeholders. But it cannot support dynamical analysis for which state-space models are required. In in this paper, we attempt to transform the UML model into a state space model, Petri Nets for the system

dynamical analysis. Petri net [42] is a graphical and mathematical tool and widely used for dynamical analysis of the systems. We validate the approach on an NPP system, known as RCICS.

The remainder of this Chapter is as follows: In section 6.2, we give the literature survey of the methodologies for dynamical analysis of the systems along with their shortcomings. A complete case study of RCICS along with its model is given in the section 6.3. Section 6.4 describes our approach to transform UML state-chart diagrams into Petri Nets for dynamical analysis. Also, in this section, we demonstrate and validate our approach on the case study. Section 6.5 concludes this chapter.

6.2 Related Work

Etienne Andre et al. [143] proposed an automated translation of UML state machine into the colored PNs using model transformation techniques. One of the limitations of this methodology is that translation is not entirely straight forward. Also, this approach inefficient for large UML state machine diagram due to some exponential loops in it. Further, this does not address the "ilities" of a system quantification.

Christine Choppy et al. [144] proposed a framework for the formal verification of USCDs. This framework used for model checking by translation of a state chart diagram into a hierarchical colored Petri net. By the use of example, this approach

illustrates that whenever required properties are not checked, there is a chance to revise the model into a more satisfactory one. However, it lacks a strong validation on realistic case study to check its efficiency.

Monica Singh et al. [145] proposed an approach for dynamic aspect verification and validation of a safety-critical system. This approach is validated by use of theorem prover Z notation tool, which significantly increased the correctness of system's aspect verification. However, this does not address the "ilities" of a system quantification and hence the implementation domain of this approach is limited.

Shuang Liu et al. [146] proposed a formal operational semantics to cover all specification of UML state machines which used to show the dynamic behavior of a software system. This approach uses automated verification techniques viz. model checking. Further, this approach addresses the synchronous and asynchronous communications between state machines. However, it lacks to define the constraints formally and action language.

Robert G. Pettit IV and Hassan Gomaa [147] proposed a modeling and analyzing approach which uses behavioral design patterns for the concurrent object-oriented software designs. This approach maps UML object to colored PN in the form of reusable templates. Further, it addresses the deficiency of behavioral modeling, viz. real time capturing of critical information, reactiveness, etc. This paper lacking of state-space analysis to analyze the effect of system wide state changes.

Lalit Singh and Hitesh Rajput [148] proposed an approach to perform safety analysis, in the design phase, of safety critical computer based systems (CBSs) and validated on real case study. This approach uses state space model to construct a mathematical model. However, the constructed state-space model does not use the features of system requirements directly to build the model that's why there might be possibilities to miss some important requirements.

Lalit el. al[149] proposed an approach to verify the design of instrumentation and control systems. However, SCSs do have more stringent safety requirements. Further only reliability aspect has been considered in this paper.

Lalit el. Al [150] proposed an approach for parameter estimation in Markov model of software reliability for early prediction, on which the dependability analysis depends. However, the emphasis has been given in reliability quantification and not on safety analysis.

Lalit et. al [151] proposed an approach for estimating transition probability matrix among the states of Markov chain during the early stages of system development life cycle to take the design decisions. But in case of SCSs, without addressing safety aspect during this phase, it will not be beneficial as the system may again undergo to the design phase from the final testing stage.

6.3 A Case Study

In this section, we give a complete case study of RCICS as safety critical control system along with its failure modes. All the possible failures occur due to failure of intended function of hardware, software, and or both.

6.3.1 RCICS Overview

The reactor core isolation cooling system (RCICS) is required by NPP technical specifications for performing a safe action and is shown in Figure 6.1. The purpose of the RCICS is to ensure that the makeup water to the reactor vessel for adequate core cooling when the main steam lines are isolated, and the regular supply of water to the reactor vessel is lost. There are different events, in which RCICS takes part and prevent the overheating of reactor fuel. These events are: 1) A complete plant shutdown occurs under conditions of a loss of the feedwater system be-fore the reactor is depressurized to a point where the shutdown cooling system can be placed into operation, and 2) The reactor pressure vessel (RPV) is isolated in conjunction with a loss of coolant flow from the feedwater system. The RCICS consists of pumps, condensate storage tank (CST), suppression pool (SP) , valves, batteries, sensors, and a control system requisite to deliver water to the reactor vessel at operating conditions. The RCICS is the only permanent installed system that not only works on normal power unavailability but works on the failure of backup power source (the emergency diesel generators).

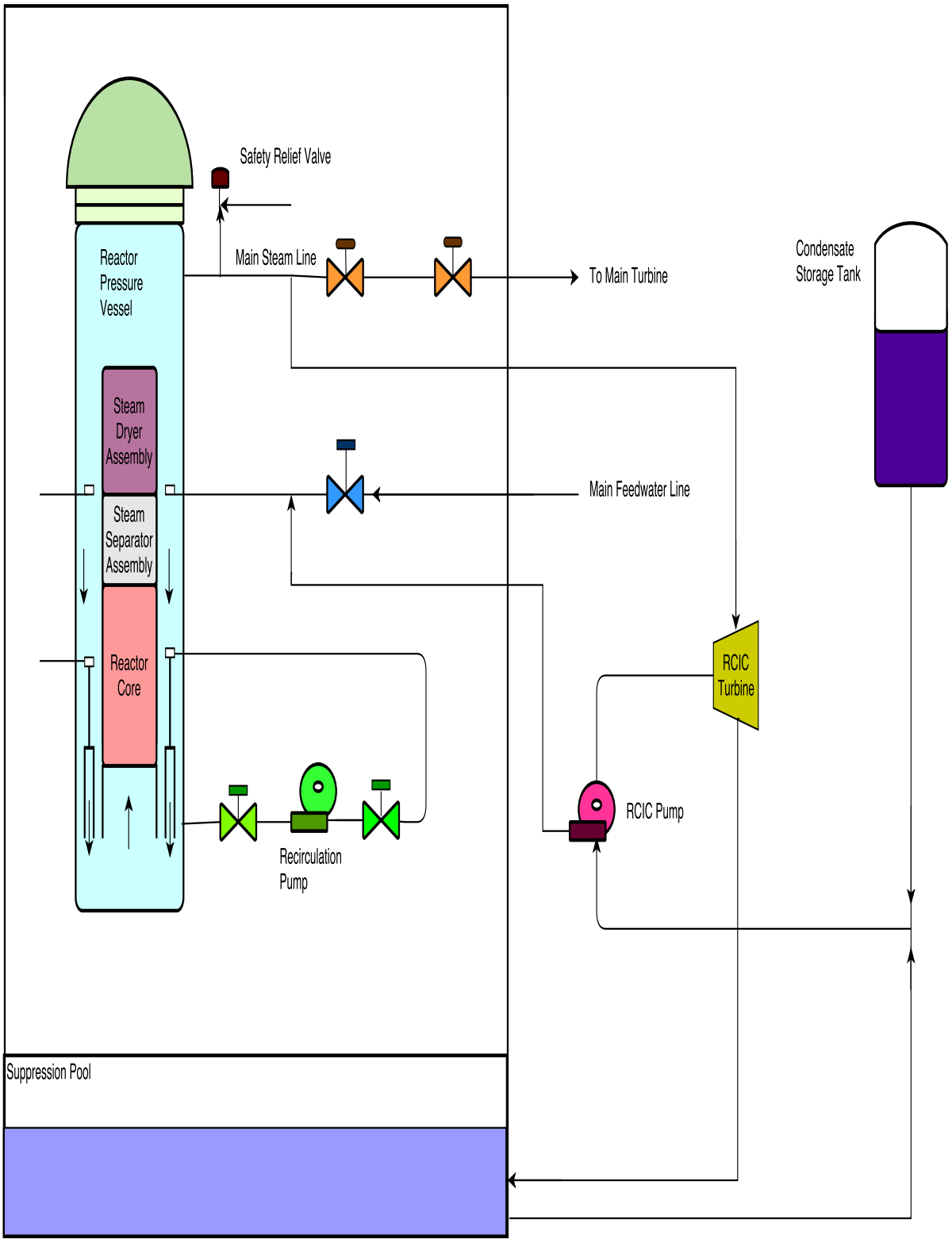

FIGURE 6.1: The RCIC system outlay

6.3.2 RCICS Operation and Operating Modes

In normal condition, initiation of the RCICS is automatically accomplished on demand. But, when an automatic controller is not working, or the required voltage of batteries is low so that automatic controller, not function properly, the operator can manually operate to maintain water flow rate. It initiates automatically when it gets a signal of low reactor water level. Upon system initiation steam supply to the RCICT, then RCICT exhausts portion of decay heat steam from RPV to SP, and simultaneously it drives the RCICP to transfer water from either CST or the SP the CST to the RPV. In parallel, re-circulation loop starts working to cool down the core of RPV. Re-circulation loop contains RVs and RP to operate the circuit. RCICS also used SRV is used to maintain desired pressure of RPV. When the water level reaches its desired limit, to stop steam-driven-turbine RCICT SRV is used so that RCICP stops.

From the operational point of view, the RCICS operates in two different modes depending on the mission time of the system was operated for the particular event to provide coolant flow to the RPV. These modes are 1) short-term mode (STM), and 2) long-term mode (LTM).

In STM, the RCICS to start automatically on low RPV water level signal or they requires operators to manually start the system to mitigate the RPV water level transient. In this mode, feed water is available or is restricted within a few minutes

($\approx \leq 15$ minutes) to maintain a regular water level. In LTM, RCICS works for more than 15 minutes. In this mode, RCICS may have to be work for several hours.

6.4 The Proposed Method For Safety Analysis By Use Of Conversion The UML Model into Petri Net Model with A Case Study Illustration

In this section, we propose a framework to perform safety analysis of the system. This framework concentrates on the critical steps for deriving a PN from UML diagram to quantify all the possible hazards of a critical control and safety system. The framework is informative to know 1) to what extent the system is safe and 2) what type of risks are associated with the system. We use this framework for the many safety critical systems. The framework contains six phases as shown in Figure 6.2. Each phase is described as follows.

6.4.1 Phase 1: Requirements Analysis

In this phase, all the functional requirements of RCICS are captured, analyzed and modeled using UML modeling techniques. The objective of RCICS is to provide cooling to the reactor in case of: 1) Loss of to the "ultimate heat sink" (the river,

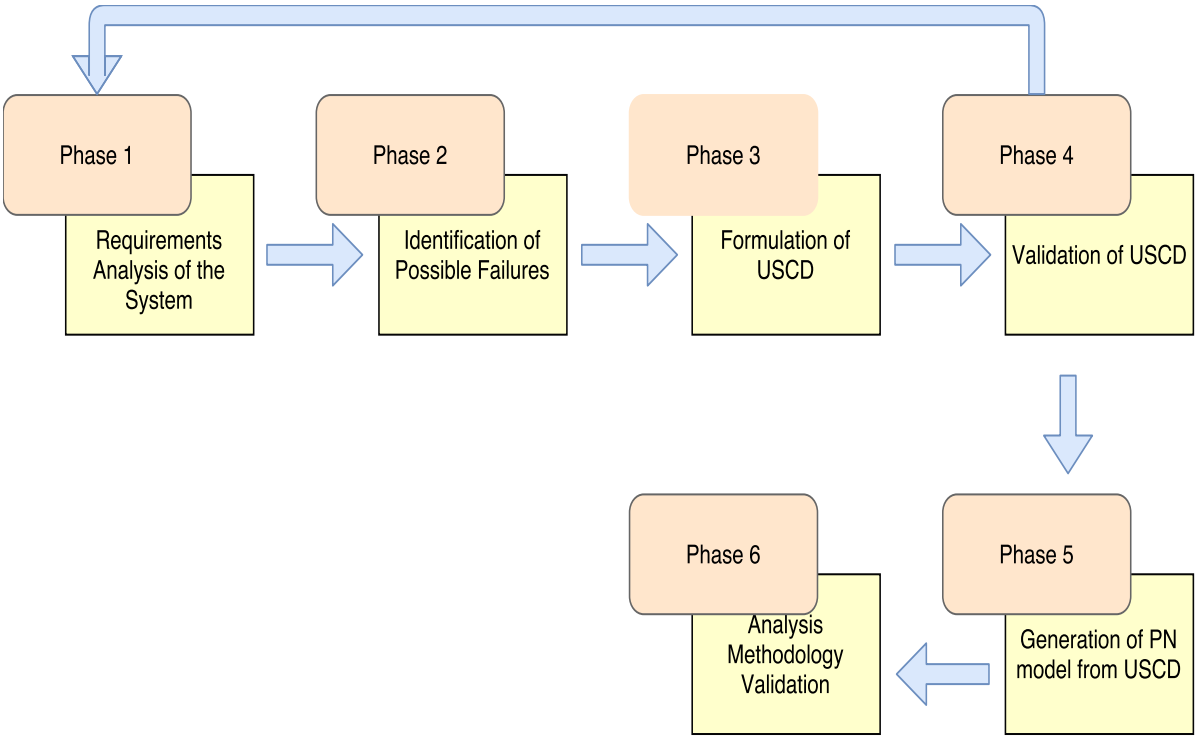

FIGURE 6.2: UML to PN mapping framework

sea or lake used for cooling), and 2) Loss of all electrical power and operates to maintain water flow rate. The behavior of main components under specific conditions are shown in Table 6.1. RCICS is independent of the following systems: (i) AC electrical system, (ii) Plant service air system (provide air under pressure for control purpose), and (iii) External cooling systems such as) the ultimate heat sink (river, sea or lake).

6.4.2 Phase 2: Identification of Possible Failures

In this phase, we identify all the possible modes of failures of the system. The system is regarded as failed if it fails to provide cooling to the reactor or water level

TABLE 6.1: Behavior of the main components of RCICS under specific conditions

Component(s)	Behavior	Conditions
RCICT, RCIC Pump (RCICP), and Recirculation Pump (RP)	Shutdown	▪ Turbine runs over the set speed. ▪ Exhaust pressure from turbine is too high. ▪ Suction pressure for the pump goes too low. ▪ It receives an automatic shutdown signal.
Recirculation Valves (RV) and Safety Relief Valve (SRV)	Fails to open	▪ Mechanical failures. ▪ DC power supply failure.

is out of the specified range. These failures are identified by the use of qualitative approaches, like mind maps, checklists, history of failures, etc. From an analysis of these failures, we identify hazards, keeping in mind that these hazards represent a catastrophic risk. In the worst case, every failure would be treated as a hazard, i.e., hazards $\subseteq$ failures.

RCICS system is composed of many components that include a control system, to manage all the components control failure of the system; sensors, to acquire raw process parameters; batteries, to operate the valve control; CST, to provide makeup water on demand and also used as normal suction source; ST, as an alternate source of water for the RCICP and also used to condense the turbine exhaust steam; RCICT, designed to operate with reactor-decay-heat-generated steam supply

TABLE 6.2: Possible operational failures and their triggering events

Event No.	**Description**
e_1	Sensor fails to sense the process parameters
e_2	Control system fails
e_3	Water below the minimum marked level in RPV
e_4	Water reached the maximum marked level in RPV
e_5	DC-Power fails
e_6	RCICT fails to start
e_7	RCICT fails to shut down
e_8	RCICP fails to on
e_9	RP fails to on
e_{10}	RV fails to open
e_{11}	SRV fails to open
e_{12}	SRV fails to close
e_{13}	System reset

as a source of energy to drive the turbine-driven pump RCICP; RCICP, to inject water (injection flow:182 m^3/h) into the reactor from CST or SP; RP-RVs, used to re-circulation of water in RPV to cool down the core; SRV, to maintain RPV pressure within desirable limits and a Risk alarm alert to the operator, when functional deviation of any component noted.

When the initiation signal is received by RCICS, required actions occur automatically to maintain the water flow rate in RPV. In this process, there are 13 possible failures that we consider as triggering events and are shown in Table 6.2.

6.4.3 Phase 3: Formulation of USCD

Based on the system requirements, we constructed the UML State Chart Diagram (USCD) of the system in this phase. We use USCD, since it is capable to model all

the possible conditions of a system along with its associated transitions. USCD of RCICS is shown in Figure 6.3.

6.4.4 Phase 4: Validation of USCD

In this phase, we assured that the constructed USCD of the system based on system requirements meets the needs of the customer and other identified stakeholders. If any functional requirement(s) are found unaddressed, we return back to phase 1 to incorporate them.

6.4.5 Phase 5: Generation of PN Model from USCD

In this phase, PN is constructed from corresponding USCD. This conversion depends on the complex type of state-chart diagram, i.e., the simple or orthogonal state-chart diagram. The simple state-chart diagram does not contain any fork and join operations, whereas orthogonal state-chart diagram contains join or fork to model the parallelism and concurrency. The algorithms to map state-chart diagram into PN model are as follows:

1) Transformation of simple state-chart diagram into PN:

In this case, PN place is generated by replacement of each state. A transition is either output transition or input transition in the state-chart diagram. For every input transition, if it has input event then add input event to the transition input

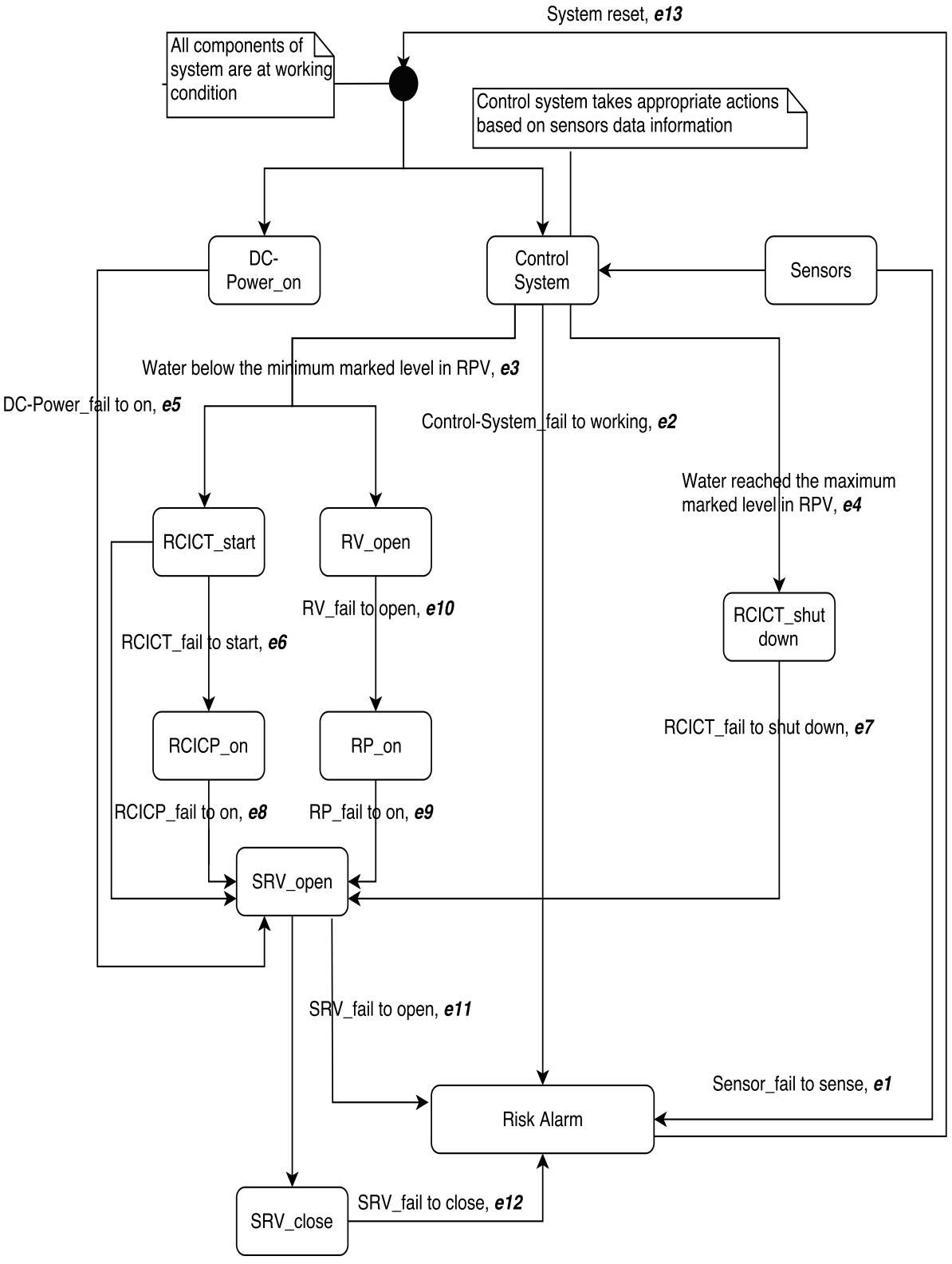

FIGURE 6.3: UML state-chart diagram of RCICS

of PN, otherwise create input event. Likewise, for each output transition, if it has output event then add output event to the transition output of PN. After generation of these places and transitions corresponding to state and transitions, arcs of PN will be used to link such as USCD output to PN transition. The detailed algorithm is shown using flowchart, in Figure 6.4.

2) Transformation of orthogonal region USCD into the PN model:

This transformation has two parts. In the first part, PN places are obtained, whereas in the second part PN transitions and arcs are obtained from the USCD. These two algorithms are shown using flowcharts, given in Figure 6.5 and Figure 6.6 respectively. In flow chart diagram Figure 6.5, orthogonal states are used to construct parallel lanes and flow chart diagram 6.7 is used to map fork or join operations into corresponding transitions.

Mapping of USCD into PN of the RCICS using these algorithms is shown in Figure 6.6. Each event is a place in PN with a self-loop that always contains one token.

6.4.6 Phase 6: Analysis Methodology Validation

In this phase, we analyze the PNs structural and behavioral properties, such as deadlock, liveness, and boundness. Analysis of PN model gives valuable information, like deadlock, liveness, boundness, etc. It also aids in verification of critical properties, such as mutual exclusion, etc. In addition, it carries out a quantitative assessment of critical quality attributes, viz. reliability, security, safety, etc. These

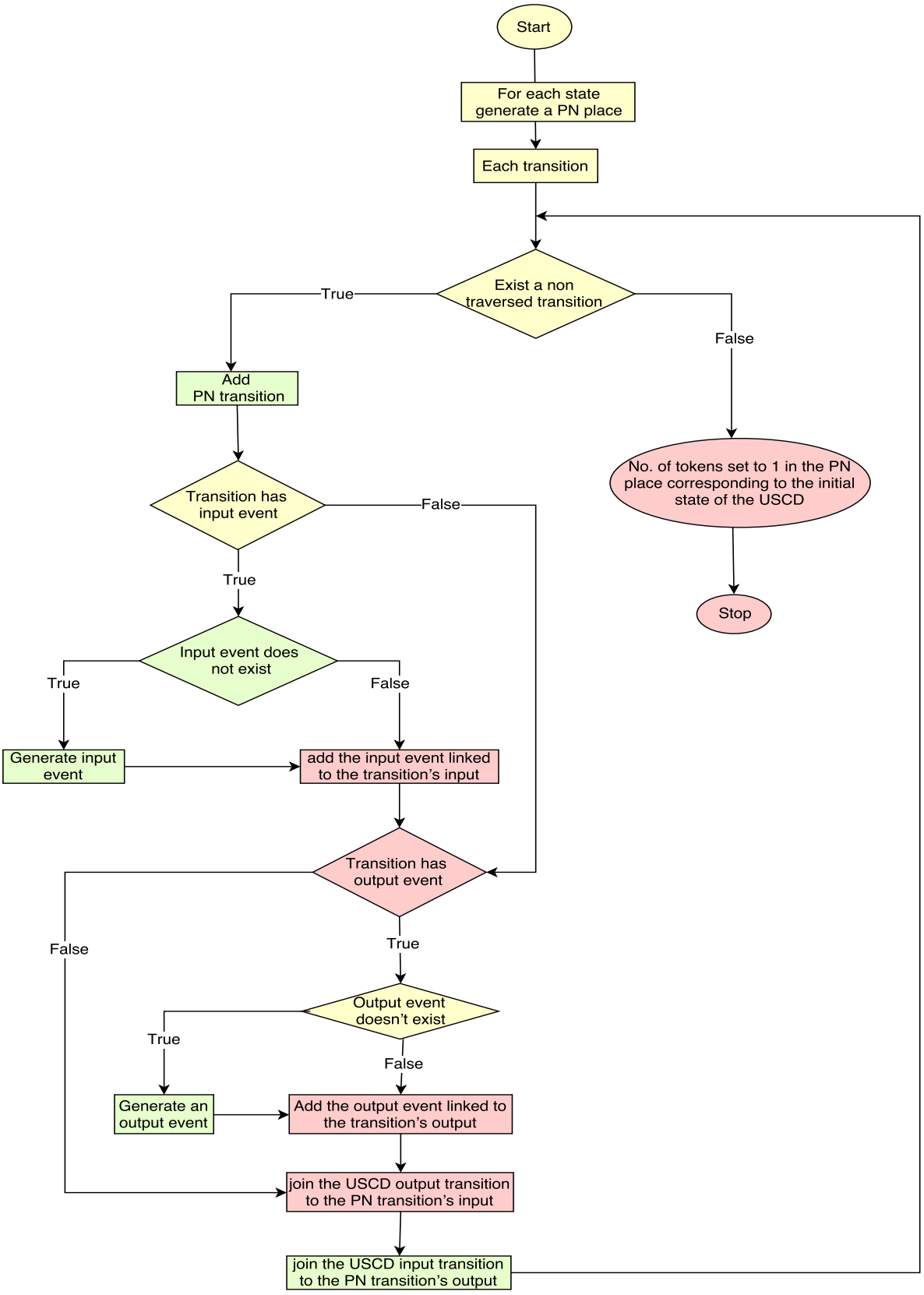

FIGURE 6.4: Flow chart for Mapping of Simple USCD into PN

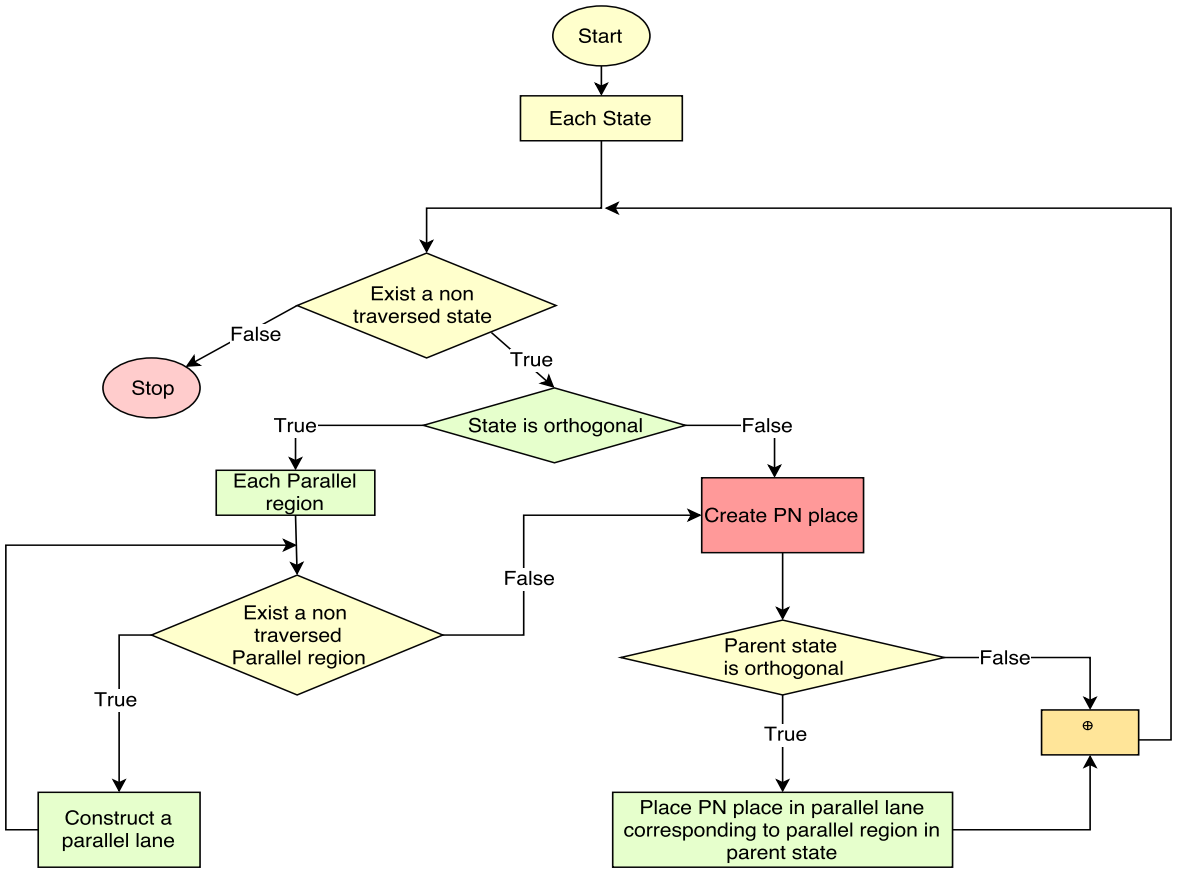

FIGURE 6.5: Flow chart for orthogonal region USCD into PN places

quality attributes are extremely vital for SCSs. The numerous analysis techniques using PN model is described in [42]. To validate our approach, we performed a quantitative safety assessment of RCICS and compared it with an assessment based on three years of operational-profile data. Our analysis took into account the failure of DC-Power, RCICT, RCICP, RP, RV, and SRV. All hazards and its required highest failure rate h^{-1} based on SIL (IEC 61508) are shown in Table 6.3.

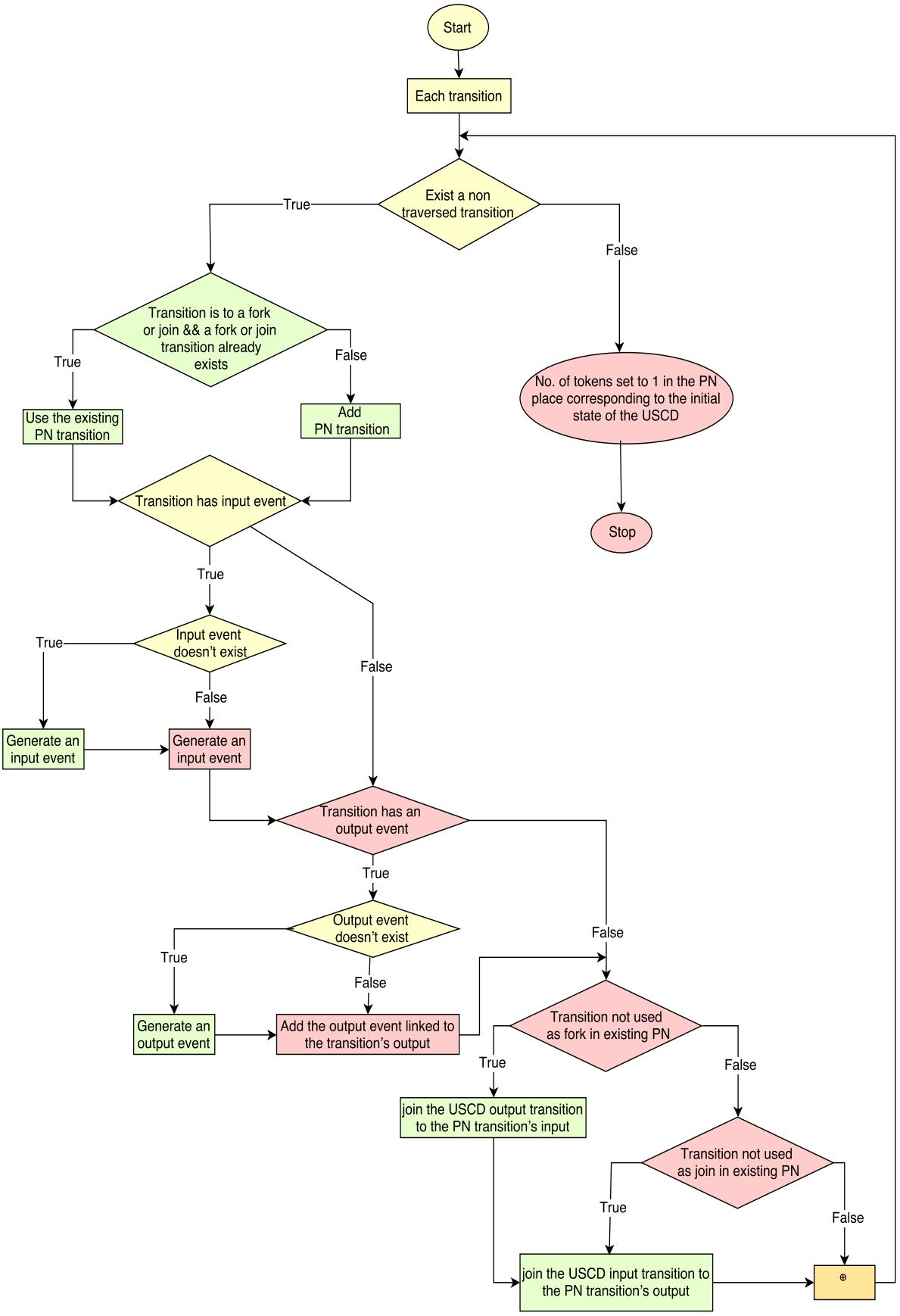

FIGURE 6.6: Flow chart for orthogonal region USCD into PN arcs and transitions

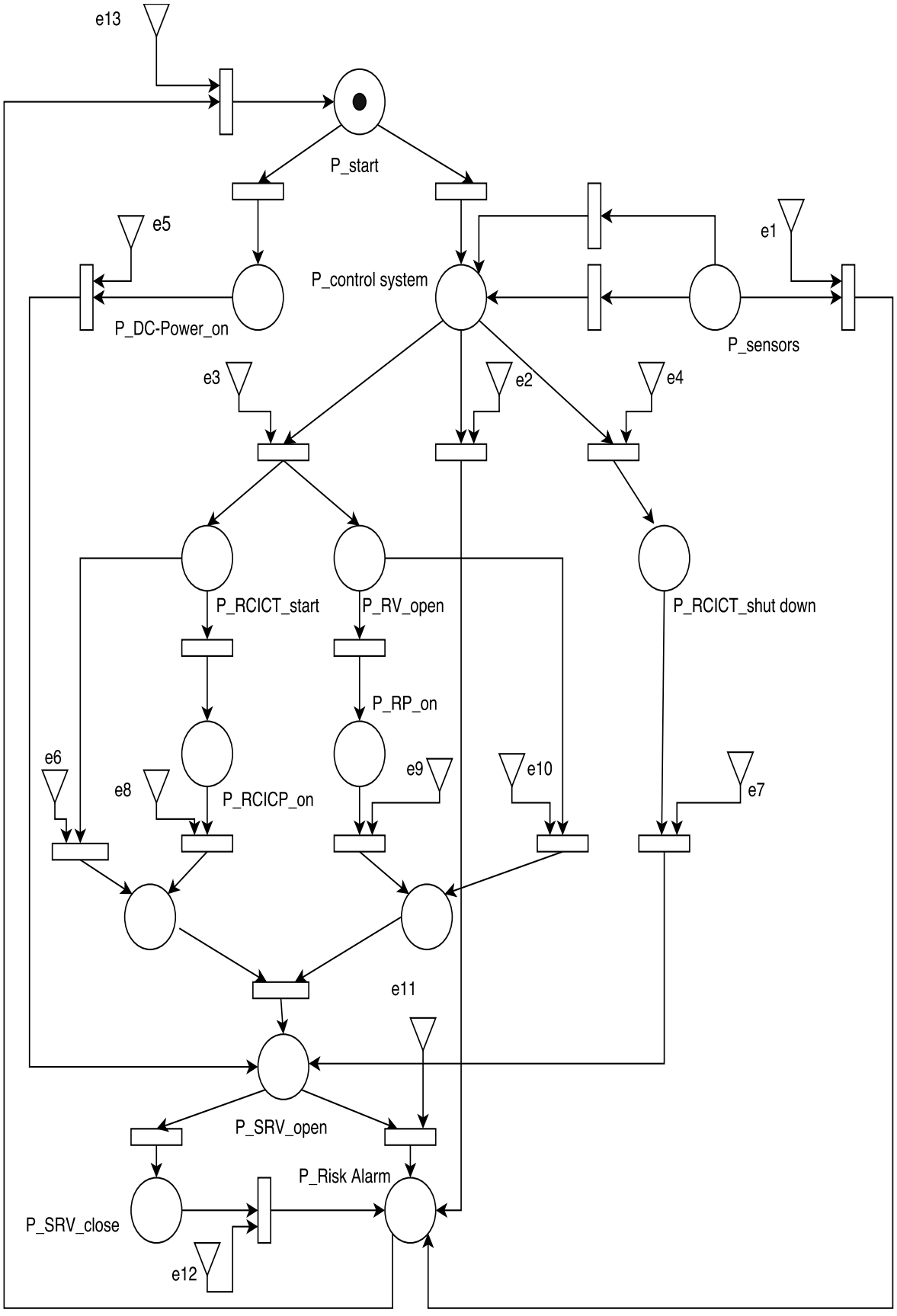

FIGURE 6.7: The RCICS USCD converted into PN

TABLE 6.3: Possible failures and SIL related to each hazard mapped to RCIC controller system

Failures Mapped to RCICS			
Component	**Transition Associated**	**Explanation**	**Hazard**
DC-Power	DC-Power *on*→ DC-Power *off*	DC-Power *off* when it should *on*	H_1
RCICT	RCICT *start* → RCICT *shutdown*	RCICT *shutdown* when it should *start*	H_2
	RCICT *shutdown* → RCICT *start*	RCICT *start* when it should *shutdown*	H_3
RCICP	RCICP *off* → RCICP *on*	RCICP *off* when it should *on*	H_4
RP	RP *off* → RP *on*	RP *off* when it should *on*	H_5
RV	RV *close* → RV *open*	RV *close* when it should *open*	H_6
SRV	SRV *close* → SRV *open*	SRV *close* when it should *open*	H_7
	SRV *open* → SRV *close*	SRV *open* when it should *close*	H_8

SIL (IEC 61508) related to each Hazard				
Hazard	**Severity**	**SIL**	**Required Failure Rate h^{-1}**	**Markov State Transition Parameter**
H_1	*H*	*4*	$< 10^{-9}$	$\lambda_{0,1}$
H_2	*H*	*4*	$< 10^{-9}$	$\lambda_{0,2}$
H_3	*L*	*1*	$< 10^{-6}$	$\lambda_{0,3}$
H_4	*H*	*4*	$< 10^{-9}$	$\lambda_{0,4}$
H_5	*L*	*1*	$< 10^{-6}$	$\lambda_{0,5}$
H_6	*L*	*1*	$< 10^{-6}$	$\lambda_{0,6}$
H_7	*H*	*4*	$< 10^{-9}$	$\lambda_{0,7}$
H_8	*L*	*1*	$< 10^{-6}$	$\lambda_{0,8}$

$$P = \begin{array}{c|cccccccccccccc} M & P_0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 & P_{10} & P_{11} & P_{12} & P_{13} & P_e \\ \hline M_0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ M_1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ M_2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ M_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ M_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ M_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ M_6 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ M_7 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ M_8 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ M_9 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ M_{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ M_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ M_{12} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ M_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ M_{14} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ M_{15} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ M_{16} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array} \tag{6.1}$$

Table 6.3 shows the eight hazardous states, H_1 to H_8. We use Markov chain reachability tree model to figure out the probability of system failure from PN

model. We generate marking matrix (called transition probability matrix of the entire system and in this matrix, all event places are denoted by P_e) represented by Equation 6.1 and corresponding reachability tree is given in Figure 6.8.

From this reachability graph, we generate Markov model having one working state and 8 failure states $(M_1, M_6, M_7, M_9, M_{12}, M_{13}, M_{14}, M_{15}) \cong$ $(F_1, F_2, F_3. F_4, F_5, F_6, F_7, F_8)$, as shown in Figure 6.9.

This Markov model can be solved as follows.

$$P_o\,(success) = e^{-\left(\sum_{i=0}^{6} \lambda_{0,i}\right)t} \tag{6.2}$$

$$P_i\,(F = F_i) = \sum \lambda_{0,i} \times P_o\,(success) \tag{6.3}$$

where,

$P_o(success)$: Probability of system success;

$P_i\,(F = F_i)$: Probability that the system is in hazardous state;

$\lambda_{0,i}$: Failure rate;

The probability of all Hazardous states of each hazard based on Equations 6.2, 6.3 for $t =$ 1 hour of the system exposure is as follows:

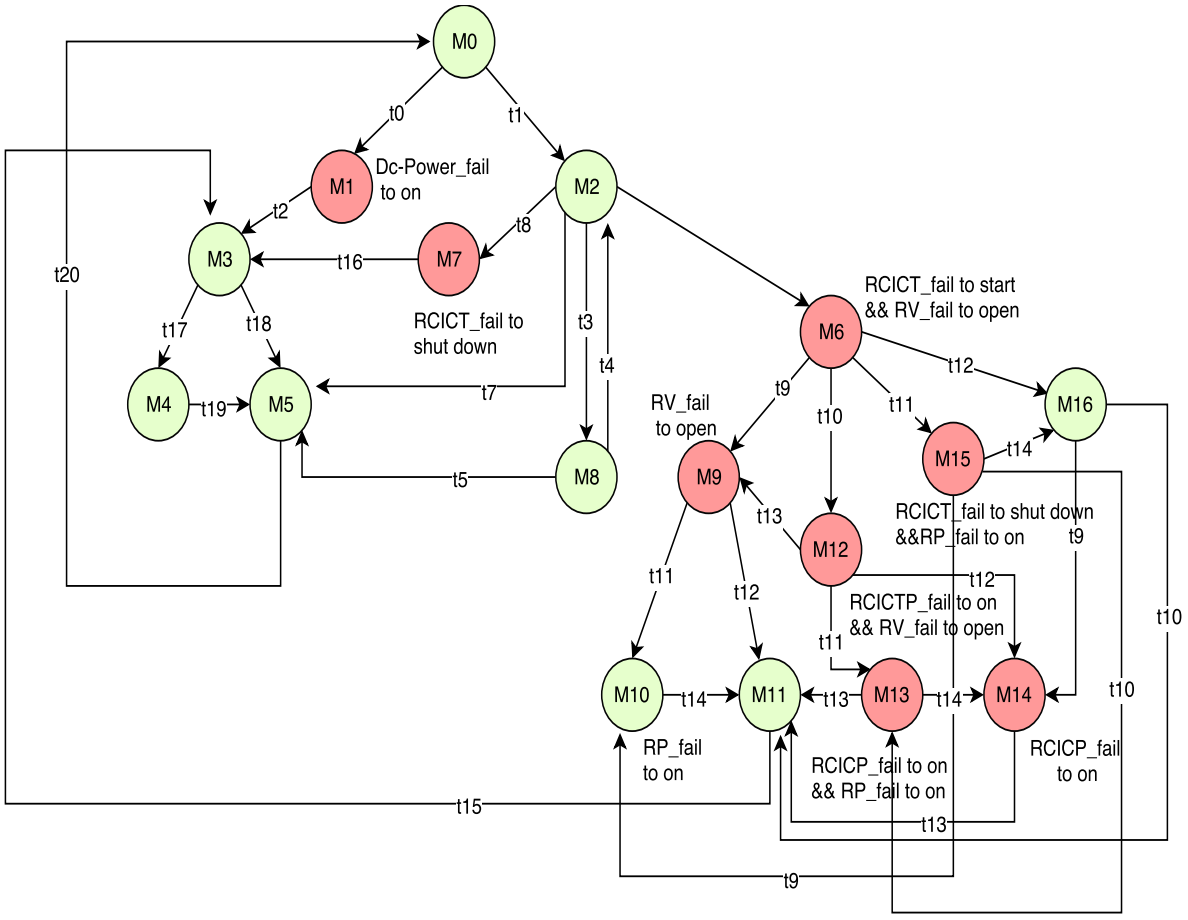

FIGURE 6.8: The Reachability tree for RCICS

$$
\begin{aligned}
P_o(Success) &= 9.99995996 \times 10^{-1} \\
P_1\,(F = F_1) &= 0.999995996 \times 10^{-9} \\
P_2\,(F = F_2) &= 0.999995996 \times 10^{-9} \\
P_3\,(F = F_3) &= 0.999995996 \times 10^{-6} \\
P_4\,(F = F_4) &= 0.999995996 \times 10^{-9} \\
P_5\,(F = F_5) &= 0.999995996 \times 10^{-6} \\
P_6\,(F = F_6) &= 0.999995996 \times 10^{-6} \\
P_7\,(F = F_7) &= 0.999995996 \times 10^{-9} \\
P_8\,(F = F_8) &= 0.999995996 \times 10^{-6}
\end{aligned}
\tag{6.4}
$$

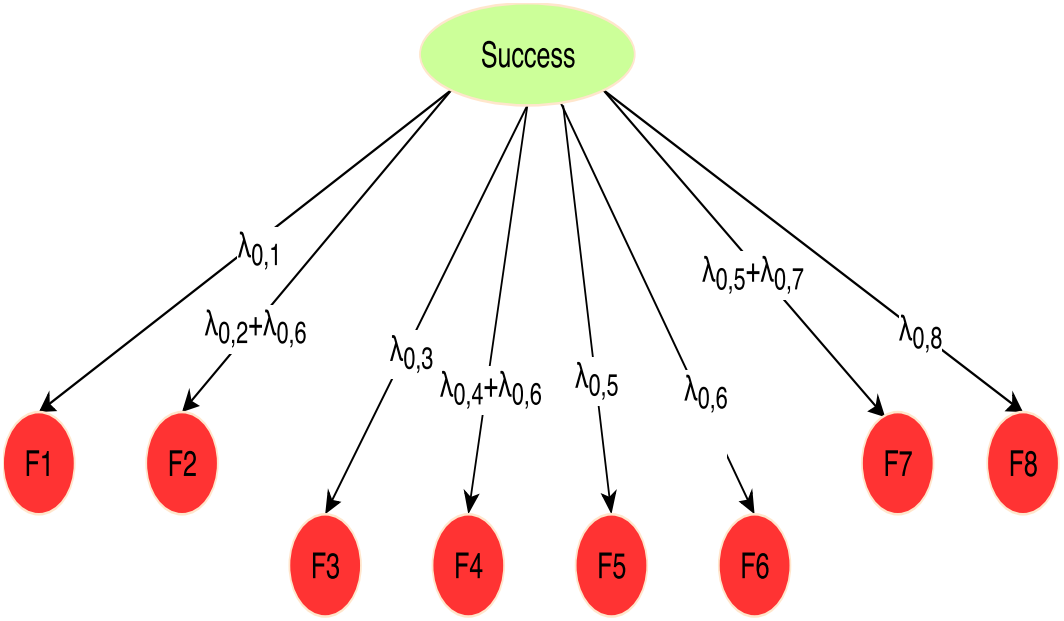

FIGURE 6.9: The Markov model of Hazardous states for RCICS

From the operational profile data of 3 years, the computed failure rate [143] of this system is given by:

$$\lambda = 1.81 \times 10^{-6} \tag{6.5}$$

From Equations 6.4 and 6.5, it can be seen that the hazard rate, based on the operational profile date and computed by transforming UML start chart model into PN model is of the same order. Hence it proves the validity of our methodology.

6.5 Conclusions

State-space models have a potential to verify the system with respect to its structural and behavioral properties. However, there is no standard mechanism to create state

space models of the systems, directly from the functional requirements. UML can capture all the requirements and be easily understood by all the stakeholders and hence it is easy to do UML modeling. In the present paper, we propose a framework for safety analysis in terms of quantitative probabilistic hazard assessment by using conversion UML state-chart diagrams into Petri Nets for system analysis of the critical control and safety system. The failure rates used in this paper are based on SIL (IEC 61508). Our proposed framework addressed the existing limitations, described in Section 6.2. The approach has been validated on 13 sets of operational profile of different safety critical systems of NPP and in this chapter; it is demonstrated on RCICS. The result of the proposed approach shows its effectiveness.

In the next and last chapter of the thesis, we present the overall discussion, concluding remarks and future scope about the entire research work.