Chapter 2

Preliminaries

Safety Critical System (SCS) calls for an understanding of various concepts, terminologies, definitions, and modeling analysis techniques. There are good deals of discussion available in the literature that are being built upon preliminary understanding in the areas of dependability engineering regarding safety related issues. In this chapter we provide the preliminaries underline definitions, concepts, etcetera that are needed to critical system and constitutes the fundamental understanding for developing the further reasoning throughout this thesis. In next section, we give a brief overview of dependability as safety is one of the attributes. After that, we talk about Computer Based System (CBS) - A system driven by the embedded software would always consist of both hardware and software components, and such a system is popularly known and may also be termed as CBS. There are various types of CBS deployed in the safety critical and safety related applications that need to be designed as per safety guides of the related authority for deployment. The classification of

CBS is also described herein. Next, basic dependability mathematics and various important dependability attribute metrics are discussed. Finally, the studies on state space models needed to understand different aspects of system safety and related concepts are summarized.

2.1 Dependability

The migration from analog system to digital systems for instrumentation and control (I&C) has increased the complexity of the instrumentation day by day. The I&C systems being developed are computer-based consisting of embedded digital hardware and software components. These systems are performing many diverse and highly complex functions that are integral to the safety-critical requirements of a SCS, and the failure of an I&C system could lead to risk significant events. To prevent such situations from arising, there is a great need for these systems to be dependable; i.e., these systems must provide a specified quality of service. It is the system designer's responsibility for demonstrating that a given system is dependable (Carter, 1987). Hence, there is a definite need for dependability analysis to be performed during the design phase.

A systematic elucidation of the concepts of dependability comprises of three parts: 1) the threats to, 2) the attributes of, and 3) the means by that dependability is attained, as shown below in Figure 2.1.

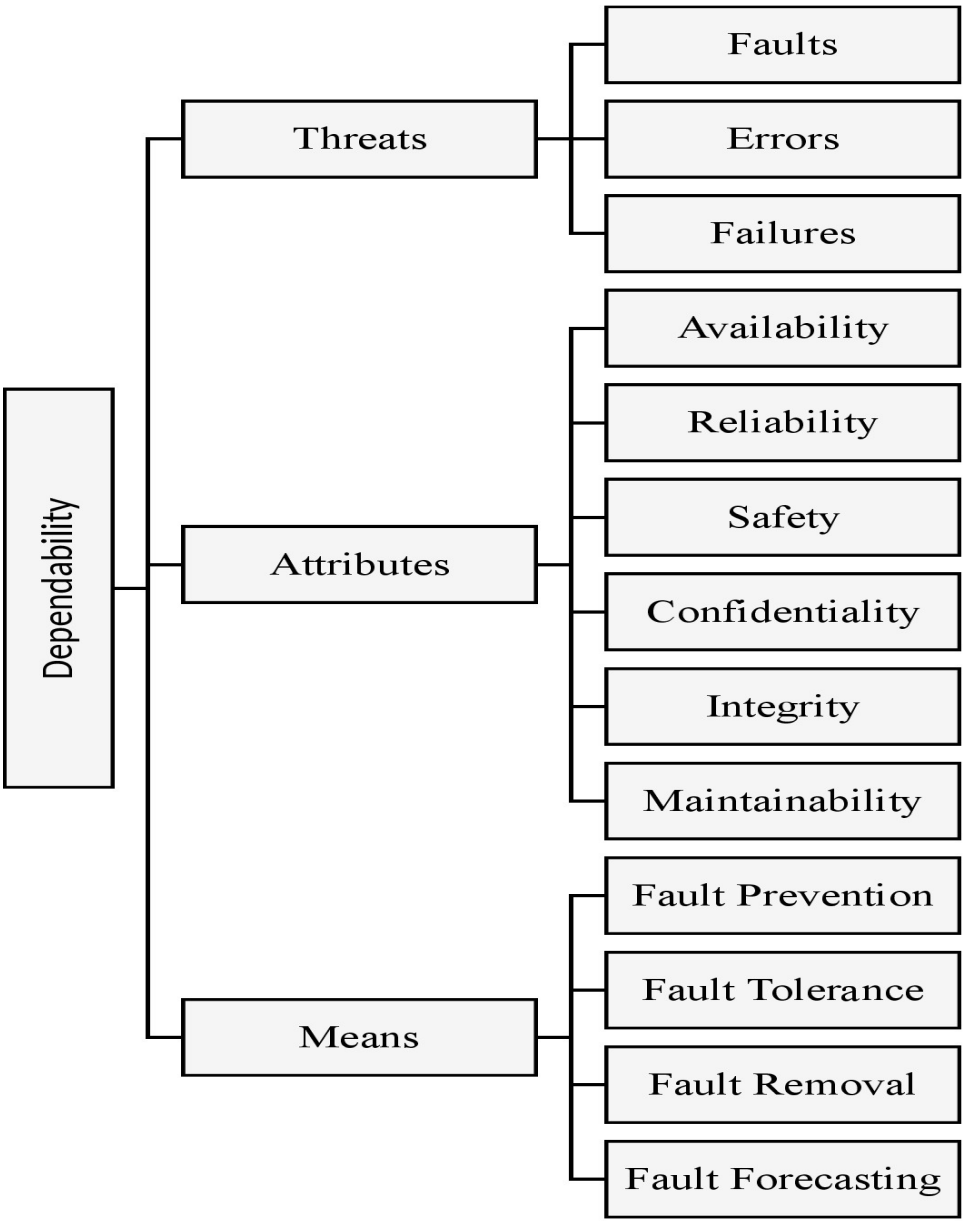

FIGURE 2.1: The Dependability Tree

Computing systems are characterized by five basic properties: 1) functionality, 2) usability, 3) performance, 4) cost, and 5) dependability. The ability of a computing system to deliver service that can justifiably be relied on is known as dependability. The service provided by a system is its behavior since it is perceived by their user(s); a user is another system (may be physical or human) that interacts with the prior at

the service interface. The function of any system is what the system is designed to do, and is defined by the functional specification. Correct service is provided when the service outfits the system function. A system failure is an event which happens when the provided service differs from correct service. A failure is thus a transition from improper service to inappropriate service, i.e., to not fulfilling the system function. The delivery of inappropriate service is a system outage. A transition from inappropriate service to appropriate service is service restoration. Based on the definition of failure, another definition of dependability which is complement of the first definition is offering a criterion for adjudicating whether or not the delivered service can be trusted: is a system ability to avoid frequent failures or more severe, and outage periods that are longer, than is satisfactory to the user(s). In the opposite case, the dependability of a system is no longer: it suffers from a dependability failure, which is a meta-failure.

2.1.1 The Threats: Faults, Errors, and Failures

There is a cause and effect relationship among these terms, and this relationship is depicted in the three-universe model, as shown in Figure 2.2. In this model, a fault occurs in the physical universe. The fault itself is a physical defect, inadequacy or flaw that take place within some hardware or software component (Johnson, 1989).

An example of a fault could be a broken ground connection in a printed circuit board or an infinite loop in a computer program. The manifestation of a fault is

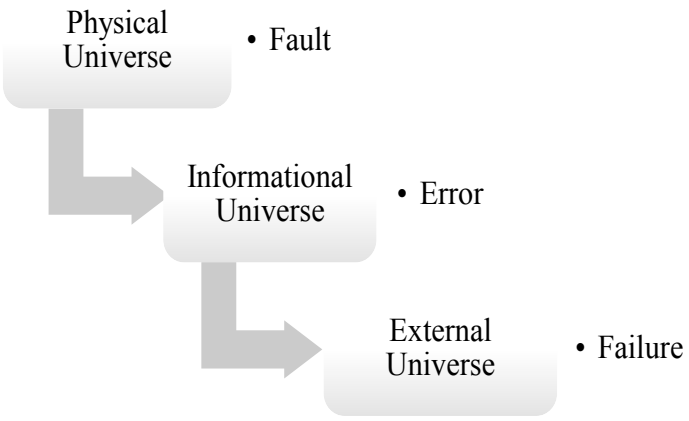

FIGURE 2.2: Three-universe model (Jhonson, 1989; Laprie, 1985)

an error, which occurs in the informational universe. An error is any deviancy from correctness or accuracy. It can be represented by parity errors or by typographical mistakes. Finally, the third universe is the external universe that contains failures. A failure is merely the nonperformance of some expected action. Hence, a failure is the consequence of both a fault and error. It is the cause and effect relationship among the components of the three-universe model that implies two important parameters: fault latency and error latency. Fault latency is merely the duration of time between the incident of a fault and its resulting error. Similarly, error latency is the duration of time between the occurrence of an error and its resulting failure.

In explaining the three-universe model, the causes of faults are described as their differences with errors and failures. To completely understand the characteristics of faults, which is required for accurate dependability modeling, additional fault attributes must be examined.

2.1.2 The Attributes of Dependability

Dependability is an integral concept that comprises the following primary attributes:

- **Availability:** ability to readiness for correct service,
- **Reliability:** ability to steadiness of correct service,
- **Safety:** ability to avoid or suppress catastrophic magnitudes on the user(s) and the environment,
- **Confidentiality:** absenteeism of unapproved exposé of information,
- **Integrity:** absenteeism of improper system state changes,
- **Maintainability:** ability to undertake repairs and alterations.

2.1.3 The Means to Attain Dependability

A dependable computing system development requires the integrated usage of a set of four methods:

- **Fault prevention:** how to inhibit the incident or primer of faults,
- **Fault tolerance:** how to provide correct service in the incidence of faults,
- **Fault removal:** how to decrease the number or severity of faults,

- **Fault forecasting:** how to assess the present number, the future occurrence, and the possible consequences of faults.

Dependability analysis can be done on hardware and software or hybrid system. Among its attributes, availability and reliability can be quantified. Availability is useful for the repairable systems, while reliability is useful; for non-repairable systems. Traditional reliability concepts (for hardware) can be used for software reliability analysis and prediction.

2.2 Classification of Computer Based System

CBS can be classified into following [23]:

1. **Bespoke Systems**- These systems are designed for a specific application. The software is developed from scratch and runs on raw hardware, which is configured using required hardware modules. The likely off-the-shelf software component these systems might employ is the real-time kernel or executive whose implementation details may not be available due to commercial reasons.

2. **Embedded Systems**- These are function-modules with embedded software, which provide limited flexibility to select functional parameters. These modules may have simple communication, analog, and digital Input/output (I/O) interfaces (e.g., single loop controllers, smart sensors).

3. **Programmable Controller based Systems**- The programmable controllers are general purpose, programmable process control and information systems which are available off-the-shelf. The hardware and software of these systems are designed to be configurable for various types of applications.

4. **Systems based on General Purpose Computers**- These can be either stand-alone systems or networked configurations for performing control, operator information functions, etc.

2.3 Basic Dependability Mathematics

This section will discuss the basics of mathematical theory that is pertinent to the study of safety analysis. The vital concepts of probability theory are described firstly. Next, the elements of component hazard rate or function are discussed. Further, various types of distributions used in safety studies are briefly explained.

2.3.1 Concepts of Probability Theory

Probability can be seen as in what degree of confidence an event will take place. This event is uncertain and could be exemplified by a failure in a specific hardware part. However, the event has to be associated with an outcome space, which is the possible outcome of the event, e.g. an event might have the possible outcome set $\psi = \{1; 2; 3; 4; 5\}$. This outcome set is then associated with a probability distribution

P, which maps events to real values as

$$P(\psi) = 1 \tag{2.1}$$

$$P(\psi) \geq 0, \forall \psi \in \nabla \tag{2.2}$$

Extensions to Equations 2.1 and 2.2 imply several interesting conditions where

$$P(\emptyset) = 0 \tag{2.3}$$

$$P(\psi_1 \cup \psi_2) = P(\psi_1) \ + P(\psi_2) \ - \ P(\psi_1, \psi_2) \tag{2.4}$$

will be of great interest.

Conditional probability is another important aspect. It becomes apparent when there are two or more events that are dependent upon each other, e.g., one event, δ denotes hazard and another event, ω denotes failure, hence a failure can be caused by a hazard.

The conditional probability are formally denoted as

$$P(\omega|\delta) = P(\omega, \delta)/P(\delta) \tag{2.5}$$

Note that if $P(\delta)$ and $P(\omega)$ were independent events, $P(\omega|\delta) = P(\delta)P(\omega)$ and hence, $P(\omega \mid \delta) = P(\omega)$. However, keeping the dependency, further investigation of Equation 2.5 gives

$$P(\omega, \delta) = P(\delta)P(\omega|\delta) \tag{2.6}$$

which is called the *Chain Rule* and is written more generally as

$$P(\delta_1 \ldots \delta_k) = P(\delta_1)P(\psi_2|\delta_1) \ldots P(\delta_k|\delta_1 \ldots, \delta_{k-1}) \tag{2.7}$$

where, $\delta_1 \ldots \delta_k$ are events.

Additionally, an important implication of the chain rule is *Bayes' Rule*, which allows derivation of conditional probabilities, based on "inverse" conditional probabilities, as

$$P(\omega|\delta) = (P(\delta|\omega)P(\omega))/P(\delta) \tag{2.8}$$

So far we have considered basic equations in probability theory using any types of events in a specified set. By introducing random variables, as an extension to the event notion, probabilities associated with attributes of the outcome of an event are possible. Here attributes of a hazard (*the Random Variable*) might be a single component failure, two components failure, etcetera. The probability distributions

over such an attribute are denoted as $P(Hazard = Single\ Component)$. Furthermore, the random variable can have different properties, e.g., having discrete sets of possible values or having continuous infinite sets of possible values. There exists a wide range of distributions, some of them have been discussed in the coming section.

2.3.2 Time to Failure, Reliability, Hazard rate and their relations

Time to failure: Let a repairable system is observed until n failure times t_1,t_2, $\ldots$, t_n occur, where $0 < t_1 < t_2 < \cdots < t_n$. Let $T > 0$ be the random variable representing the time to next failure. Then the time to failure denotes the probability that the time to failure T is in some interval $(t, t + \Delta t)$ as

$$P(t \leq T \leq t + \Delta t)$$

Whereas, the PDF $f(t)$ and CDF $F(t)$ are given as,

$$P(t \leq T \leq t + \Delta t) = F(t + \Delta t) - F(t) \cong f(t)\Delta t \tag{2.9}$$

$$F(t) = P(0 \leq T \leq t) = \int_0^t f(x)dx \tag{2.10}$$

Reliability function: The reliability function is the probability of success at time t i.e. the probability that the time to failure exceeds t. Mathematically, it is given by

$$R(t) = P(T > t) = \int_t^{\infty} f(x)\, dx \qquad (t > 0) \tag{2.11}$$

Therefore, CDF $F(t)$ of T is given by

$$F(t) = \int_0^t f(x)\, dx = P(0 \leq T \leq t) = 1 - R(t) \tag{2.12}$$

The reliability function is also known as survival function of T. $R(t)$ decreases from 1 to 0, $\forall\ t \in [0, \infty)$. Hence $f(t)$, $F(t)\ and\ R(t)$ are equivalent representatives of the random variable$T$.

Hazard Rate: The hazard rate is defined as the limit of the failure rate as the interval Δt approaches zero.

$$h(t) = \lim_{\Delta t \to 0} \frac{F(t + \Delta t) - F(t)}{(\Delta t)\, R(t)} = \frac{f(t)}{R(t)} \tag{2.13}$$

The hazard rate is an instantaneous rate of failure at time t, given that the system survives up to t. It is known as intensity function also.

Converting,

$$h(t) = \frac{f(t)}{R(t)} = \frac{dF(t)}{dt}\frac{1}{R(t)}$$

$$\frac{dF(t)}{dt} = -\frac{R(t)}{dt}$$

$$\Longrightarrow \frac{dR(t)}{dt} = -h\,(t)\,dt$$

Integrating both sides w.r.t. t,

$$\ln R(t) \;= -\int_0^t h(x)dx + c$$

$$\because R\,(0) = 1,\; c = 0$$

$$\therefore R\,(t) = \exp[-\int_0^t h(x)dx] \tag{2.14}$$

Differentiating,

$$f\,(t) = h(t)\exp[-\int_0^t h(x)dx] \tag{2.15}$$

2.3.3 Probability Distributions Used in Safety Studies

We describe some of the popular statistical or probability distributions that are commonly used for reliability, quality, and safety analysis of CBS [1], [17].

2.3.3.1 Exponential Distribution

The exponential distribution is most extensively used distribution in safety estimation. It is the lone distribution which has constant hazard rate and is used to model the "useful life" of numerous engineering systems [24]. The PDF of the exponential distribution is

$$f(t) = \lambda e^{-\lambda t}, \forall \in [0, \infty) \tag{2.16}$$

where, t is time and λ is the distribution parameter.

The exponential CDF can be derived from its PDF as follows:

$$\therefore F(t) = \int_0^t f(t)\, dt = 1 - e^{-\lambda t} \tag{2.17}$$

The reliability function is the complement of the CDF:

$$R(t) = 1 - F(t) = e^{-\lambda t} \tag{2.18}$$

The hazard function is the ratio of the PDF and its reliability function; for the exponential distribution it is

$$H(t) = \frac{f(t)}{R(t)} = \frac{\lambda e^{-\lambda t}}{e^{-\lambda t}(t)} = \lambda \tag{2.19}$$

Since the exponential hazard function is constant (=λ), hence, it is the cause for the memoryless property of the exponential distribution. The memory less property means the probability of failure in a specific time interval is the same regardless of the starting point of that time interval.

2.3.3.2 Weibull Distribution

In 1933, the Weibull distribution was presented by P. Rosin and E. Rammler [25]. It has a wide-ranging of applications in risk/hazard calculations due to its flexibility in modeling different distribution shapes. It can be used to model time to failure of mechanical, electrical, electronics, and software components. In addition to being the most useful distribution function in safety analysis, it is also useful in classifying failure types, troubleshooting, scheduling preventive maintenance, and inspection activities. The Weibull PDF is

$$f(t) = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1}\left(e^{-\left(\frac{t}{\alpha}\right)^{\beta}}\right), \qquad t > 0 \tag{2.20}$$

where, α and β are scale parameters and distribution shape, respectively. It is important because of the three reasons: 1) It is most commonly used for the distribution of lifetimes; 2) It is related to the power law process and is used for repairable systems; 3) if debugging the system, i.e., bringing it back to a new system, then the assumption that the times between failures $T_1, T_2, \ldots, T_n$ are independent identical distribution Weibull random variables may be reasonable.

The Weibull CDF can be derived from its PDF as follows:

$$\therefore F(t) = \int_0^t f(t)\,dt = 1 - e^{-\left(\frac{t}{\alpha}\right)^{\beta}}, \qquad t > 0 \tag{2.21}$$

The reliability function is the complement of the CDF:

$$R(t) = 1 - F(t) \; = e^{-\left(\frac{t}{\alpha}\right)^{\beta}} \tag{2.22}$$

The hazard function is the ratio of the PDF and its reliability function; for the exponential distribution it is

$$H(t) = \frac{f(t)}{R(t)} = \frac{\beta}{\alpha}\left(\frac{t}{\alpha}\right)^{\beta-1} \tag{2.23}$$

2.3.3.3 Rayleigh Distribution

This is another continuous random variable distribution and is named after John Rayleigh (1842–1919) [26]. The distribution is often used in the theory of sound and in safety studies. The PDF of Rayleigh Distribution is

$$f(t) = kte^{-\frac{kt^2}{2}}, \qquad t > 0, k > 0 \tag{2.24}$$

Where, α is the distribution parameter.

The Rayleigh CDF can be derived from its PDF as follows:

$$\therefore F(t) = \int_0^t f(t)\,dt = -e^{-\frac{kt^2}{2}}, \qquad t > 0 \tag{2.25}$$

The reliability function is the complement of the CDF:

$$R(t) = 1 - F(t) = 1 + e^{-\frac{kt^2}{2}} \tag{2.26}$$

The hazard function is the ratio of the PDF and its reliability function; for the Rayleigh distribution it is

$$H(t) = \frac{f(t)}{R(t)} = \frac{kte^{-\frac{kt^2}{2}}}{1 + e^{-\frac{kt^2}{2}}} \tag{2.27}$$

It can be seen as a special case of Weibull distribution for $\beta = 2$ and $= (\frac{k}{2})^{\frac{1}{2}}$.

2.4 Dependability Attribute Prediction Metrics

Related metrics help in gaining confidence on system dependability attribute which we are going to discuss in this section. Dependability attribute's metrics are derived from the requirements of technology used, customers' requirements and consideration of required applications. The important dependability attribute

metrics include failure rate, meantime to failure (MTTF), and mean time to repair (MTTR).

2.4.1 Failure Rate

The failure rate of a component is defined as no. of failures per unit time [1]. Generally, failures occur due to undiscovered defects during the process of design, defects in components, or wrong assembly. Mishandling, shock, unusual environment stress often causes the useful life failures. We cannot eliminate an effect of such failures completely even the best design and guard techniques. The failure rate of hardware is assumed that the failure rate (t) is constant Watson [27] argues about this assumption. He suggests that instead of a bathtub curve, a curve representing a decreasing failure rate (roller coaster curve) during useful life may be more appropriate. Due to wear and tear after the useful life cause increasing failure rates on the product over time.

Software reliability, however, does not show the same characteristics similar as hardware. In a case of software, the failure rate is not constant during operational time due to continuous defect identification and removal process. Further, software does not wear and tear out and, therefore, software does not have an increasing failure rate as hardware does after operational phase.

Mathematically, relationship between reliability function R(t), failure function F(t) and failure rate or hazard rate λ(t) expressed as

$$\lambda\,(\mathrm{t}) \;= \lim_{\tau \to 0} \frac{1}{\tau}\{\frac{\mathrm{R}\,(\mathrm{t}) - \mathrm{R}\,(\mathrm{t}+\tau)}{\mathrm{R}\,(\mathrm{t})}\;\} = \;-\;\frac{1}{\mathrm{R}}\frac{\mathrm{dR}}{\mathrm{dt}} \;\;=\;\; -\frac{\mathrm{d}}{\mathrm{dt}}\,[\ln \mathrm{R}\,(\mathrm{t})\;] \tag{2.28}$$

where,

t: time before which no failure occurs; τ : is the small time interval & $\tau > 0$.

The Reliability function R(t) is calculated from Equation 2.28 as follows:

$$\mathrm{R}\,(\mathrm{t}) = \exp[\;-\;\int_0^{\mathrm{t}} (\mathrm{x})\,\mathrm{dx}\;\;] \tag{2.29}$$

The integral of the failure rate in the exponent is known as cumulative failure rate or cumulative hazard function, H(t):

$$\mathrm{H}\,(\mathrm{t}) \;\;\;=\;\;\; \int_0^{\mathrm{t}} (\mathrm{x})\,\mathrm{dx} \tag{2.30}$$

For the sake of simplicity many reliability models, assumes that the failure rate is constant (independent of time, λ(t) =λ). Hence, Equation 2.29 can be rewritten as:

$$\mathrm{R(t)} \;=\; \exp\,[-\lambda\;\mathrm{t}] \tag{2.31}$$

This metric is highly used in Reliability Engineering to take replacement, repair actions for the critical components of safety critical systems.

2.4.2 Mean Time to Failure

In reliability analysis, MTTF is a mean lifetime of an item. It is average time during which item will be expected to last in operation [28]. MTTF has a modeling assumption that failed item cannot be repaired (infinite repair time). The MTTF can be used to find out whether the redesigned system is better than the earlier item in demonstration test plans. This metric indicates life distribution of an item but fail to provide information regarding the behavior failure distribution of the item.

The mathematical relationship between MTTF and reliability function R(t) is as follows:

$$\mathrm{MTTF} = \int_0^\infty \mathrm{R}\,(\mathrm{t})\,\mathrm{dt} \tag{2.32}$$

If we consider failure rate as constant then $\lambda(\mathrm{t}) = \lambda$ and Equation 2.32 became:

$$\mathrm{MTTF} = \frac{1}{\lambda} \tag{2.33}$$

Mean time to failure (MTTF) denotes the expected time to failure for a non-repairable system, hence this metric helps to take precautionary decisions to avoid the malfunctioning of SCSs.

2.4.3 Mean Time to Repair

Mean time to repair (MTTR) is the usual time required to settle a fizzled item and return it to generation status [29]. The MTTR can be ascertained by separating the total time required for support by the aggregate number of repairs inside of a particular time allotment. For basic mission hardware, mean time to repair can dramatically affect the association's primary concern. Taking too long to repair gear can mean item scrap, missed requests and soured business connections. Mathematically, MTTR is represented as:

$$\text{MTTR} = \frac{\text{Total Maintenance Time}}{\text{No. of Repairs}} \tag{2.34}$$

The amount of time required to repair a system and bring it back online is the "time to repair" which is a critical metric to promise the concerned authorities on the time that will take bring the system back to the normal operating conditions.

2.4.4 Availability and Average Availability

The availability is the likelihood that a system will certainly be operating at an offered time. The average availability signifies the mean part of the moment the system is running over an offered time frame. For a repairable system, if it is fixed to an "as excellent as brand-new" condition whenever it fails, the average availability is represented as:

$$A_{avg}= \quad \frac{MTTF}{MTTF + MTTR} \tag{2.35}$$

Therefore, availability enhancement requires raising MTTF and also lowering MTTR. The major constraint related to the metric of average availability depends on the fact that it cannot show frequency of failings or maintenances needed. For this reason, it is just used to analyze the repairable systems where the key issue is availability as opposed to reliability.

This metric is very important for the SCSs which have to come on demand like shutdown systems of Nuclear Power Plant, containment isolation to avoid radiation leak to the public domain, etc.

2.4.5 Some other important metrics

- *Probability of failure on demand (POFOD):* It can be specified as probability of the system failure when service request comes. This metric is useful for odd and relatively infrequent services when demanded. Such metric is suitable and relevant for the safety-critical system where services are rarely demanded, but it could be a serious threat if the system fails to do so. Therefore, $POFOD=$ 0.002 implies that there is the chance of 2/1000 times for a failure when service requested.

- *Rate of occurrence of failure (ROCOF):* It is the probability of system failures which are probably to be monitor relative to the no. of system execution or to a certain time period (e.g., 24 hours). Therefore, in the above example, the $ROCOF = 2/1000$. The reciprocal of this metric is nothing but mean time to failure (MTTF).

- *Probability of failure for a given output (POFGO):* It is the probability of the specific output of the system, due to software failure.

2.5 State Space Model and Related Concepts

The necessity of early prediction of system safety is discussed in Chapter 1, Introduction. Some of the advantages of system safety early prediction is mentioned. Before discussing the existing approaches for system safety early prediction, we introduce some important state space models along with concepts and notations which are frequently used in this thesis.

2.5.1 State Machines

There are several means of modeling the behavior of systems, and the utilization of state machines is one of the oldest and most widely known. State machines permit us to ruminate about the "state" of a system at a specific point with time and depict the behavior of the system predicted on that state. The usage of this modeling

approach is not restricted to the development of systems. Actually, the thought of state-based behavior can be followed back to the initial things to consider of physical matter. For instance, H_2O can be found in three different states simply visible in nature shown in Figure 2.3: vapor (steam, fog, clouds), liquid (water), and solid (ice).

In each one of these states, the behavior of H_2O differs. Also, the method of forcing transitions among the three states is well-defined. Several other natural and manufactured systems can also be modeled by defining: 1) the probable states the system can hold, 2) the behavior in each of these states, and 3) the way the system transitions among the states, including which states are "linked" and which are not.

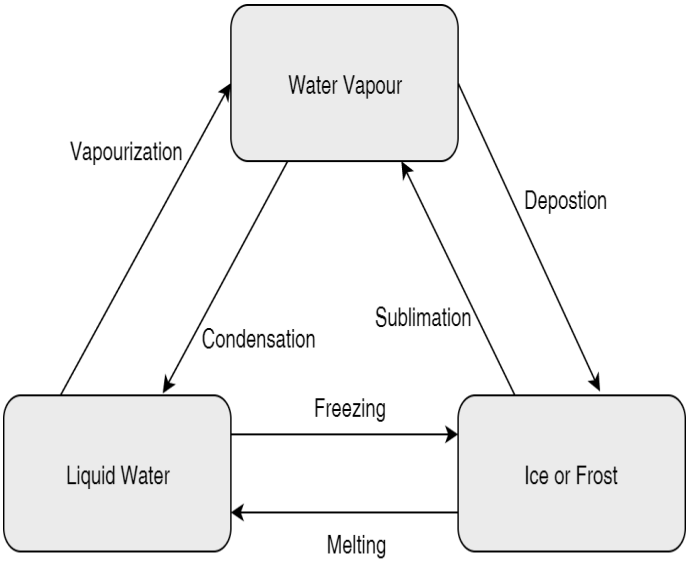

FIGURE 2.3: State machine representation of different physical behavior of H_2O

We can use an identical approach to model and design CBS by determining what states the system can maintain, what inputs or events leads to state transitions,

and the way the system will respond in each state. In such a model, we view the execution of the system as a series of transitions that change the system over its several states.

2.5.1.1 Conceptual Definition of a Finite State Machine

We can recognize various key characteristics of a system which can be modeled with a finite state machine:

- The set of states used to describe system must be finite.
- The set of inputs and events that can prompt transitions among states of a system must be finite.
- The system's behavior at a certain point in time is contingent upon the present state and the input or event which happened at that time.
- For every state the system may be in, behavior is defined for each probable input or event.
- The system has a specific initial state.

2.5.1.2 Mathematical Definition of a Finite State Machine

As a computer science and engineering student, we frequently need to define systems in a formal, mathematical way. It permits us to cause about systems both in general

as well as in particular instances. Formal notation also assistance to eradicate ambiguity when we communicate with others regarding system designs. An FSM is defined by a 5-tuple as

$$M = (\Sigma,\ Q,\ q_0,\ F,\ \delta) \tag{2.36}$$

where,

Σ: a set of symbols representing input to M;

Q: a set of states of M;

$q_0 \in Q$: the start state of M;

$F \subseteq Q$: the set of final states of M;

$\delta : Q \times \Sigma \rightarrow Q$: the transition function;

2.5.2 Unified modeling Language (UML)

The Unified modeling Language (UML) is one of the fascinating and useful tools in the domain of system engineering. The UML is a visual modeling language that allows system developer to develop blueprints that capture their versions in a standard, simple-to-realize way, and offers a mechanism to share and communicate these visions to other folks properly.

Communicating the vision is utmost significance. Before the advent of the UML, system development was usually a hit-or-miss proposition. The analyst may have

misunderstood the consumer. The analyst may have made a document the consumer could not comprehend. To include to the mess, analyst usually produced wordy, voluminous requirements documents that had been hard for the other individuals to work with project staff. Paradoxically, the sheer excess weight of these documents usually permitted critical requirements (and dependencies between requirements) to slip via the cracks. As a result, the outcomes of the analysis may not have been clear to the system developers, who subsequently may have developed a system that was hard to use and did not resolve the consumer's original problem. It is why, several of the prolonged-standing systems in use nowadays are clunky, cumbersome, and difficult to use.

It is essential to organize the design approach in a way that analyst, customers, and other concerned system development can comprehend and agree on. The UML offers such organization. Since UML is the standard for object modeling and contains diagrams for modeling a variety of components including structural, behavioral, and interaction [30]. Specifically related to the work here, State-chart Diagrams are significantly more common than most of other forms of behavior modeling. It is due both to the ease of representation due to a rich array of descriptive features as well as inclusion in the UML standard and subsequently a robust set of tool support.

State-charts were developed initially by David Harel as "a broad extension of the conventional formalism of state machines and state diagrams". This variant of state machines, called Harel State-charts. It uses the concept of Venn diagrams and directed graphs to express hierarchy and connectedness in a way that sought to solve

the problem of state explosion present in typical state machines [31]. This formalism has been co-opted into UML State-chart Diagrams (USCD), which now has its formalisms. Since USCD is similar to Harel State-charts, hence, it is considered as a variant of Harel State-charts, and is administered by the UML standards body [30], [32].

2.5.2.1 Types of State-charts

There are two related types of State-charts which were original proposed by David Harel and defined by UML. While the more commonly used are UML State-chart Diagrams, it is still useful to compare the two as a basis for the choice to use UML State-chart Diagrams. Both Harel State-charts and UML State-chart Diagrams are executed very similarly, although they are represented slightly differently. Transitions move execution between states. These transitions can be based on an event, guard, or always actionable (empty). When an event occurs, if the current state has a transition away from it that uses that event, the transition is taken. If a guard, which is a Boolean expression, is evaluated to true at any point then that transition is taken. It is also possible to have transitions away from groups of states, which are denoted by a parent state away transition. Similarly, transitions can go to a set of states, denoted by a transition to a parent state where the default sub-state is then transitioned to.

Harel State-charts

Harel State-charts were initially described informally, using illustrative diagrams and describing the behavior of transitions. However, more clarification and a formalism was necessary and was published shortly after the paper introducing Harel State-charts [31]. Updates and new ways of formalizing Harel State-charts are often defined when a new simulation or verification engine or methodology is created, not always staying entirely in line with the original [33]. Harel State-charts are made up to states defined as "blobs", where there is no strict hierarchical requirement imposed to allow for the blobs to be used as a Venn diagram that shows blobs containing portions of other blobs [30], [34], [35]. An example that shows this lack of strict blob subsets are in Figure 2.4. We will see that state D is in C which would not be possible in UML State-chart Diagrams. A Harel State-chart is a 7-tuple represented as

$$C \;=\; (s,\; S,\; t,\; p,\; e,\; d,\; E) \tag{2.37}$$

where,

S: a finite set of states;

s: the start state and $s \in S$;

t: a transition function which maps the set of states, S onto a set was any element is in S;

p: a parent function that maps the set of states, S onto a set was any element is in S;

e: an event function that maps the set of transitions, $S \times S$ to a set of events;

d: a default transition function that maps the set of states to their default substate if it exists, or itself;

E: a finite set of events;

The Harel State-chart in Figure 2.4 can also be described mathematically as a Harel State-chart,C where:

$C = (s,\ S,\ t,\ p,\ e,\ d,\ E)$

$s = A$

$S = (\ A,\ B,\ C,\ D,\ E,\ F,\ G,\ H,\ I)$

$t = (\ A,\ B,\ C,\ D,\ E,\ F,\ G,\ H,\ I) \rightarrow (\emptyset, \emptyset, \emptyset, \emptyset, (F)\ ,\ (E, G)\ , (H)\ ,\ (G,\ I)\ , \emptyset)$

$p = (\ A,\ B,\ C,\ D,\ E,\ F,\ G,\ H,\ I) \rightarrow (\emptyset, (A)\ , (A)\ , (A, C)\ , (B)\ , (D)\ , (D)\ , (C)\ , (C))$

$d = (\ A,\ B,\ C,\ D,\ E,\ F,\ G,\ H,\ I) \rightarrow (\ B,\ E,\ I,\ D,\ E,\ F,\ G,\ H,\ I)$

$E = (\ e_1,\ e_2,\ e_3,\ e_4,\ e_5,\ e_6)$

And the event function e is:

$$\begin{bmatrix} & A & B & C & D & E & F & G & H & I \\ A & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ B & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ C & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ D & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ E & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & e_1 & \emptyset & \emptyset & \emptyset \\ F & \emptyset & \emptyset & \emptyset & \emptyset & e_2 & \emptyset & e_3 & \emptyset & \emptyset \\ G & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & e_4 & \emptyset \\ H & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & e_5 & \emptyset & e_6 \\ I & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \end{bmatrix}$$

UML State-chart Diagrams

UML State-chart Diagrams are a variation of Harel State-charts. One of the largest differences between Harel State-charts and UML State-chart Diagrams is that sub-states must be entirely contained within their parent [36]. A UML State-chart Diagram example is shown in Figure 2.5, which is visibly different from a Harel State-chart as all states are fully enclosed in a single parent state.

The UML State-chart in Figure 2.5 can also be described mathematically as a Harel State-chart,C where:

$C = (s,\ S,\ t,\ p,\ e,\ d,\ E)$

$s = A$

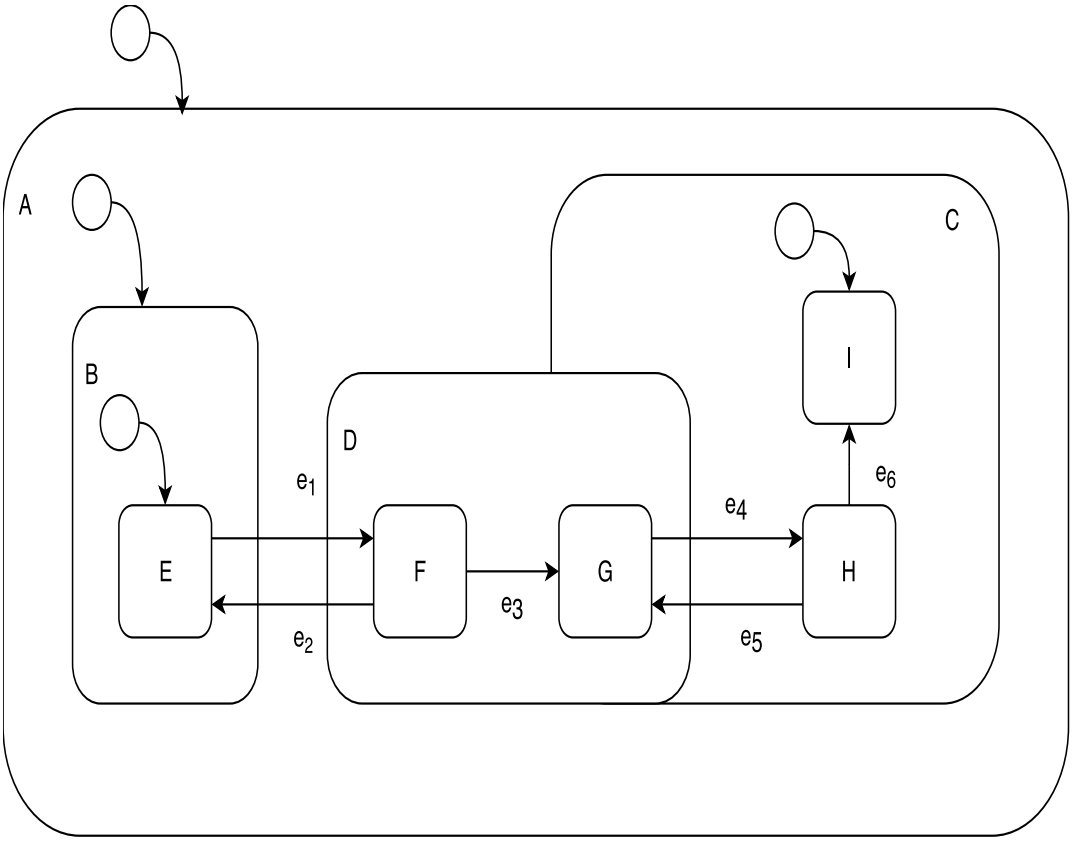

FIGURE 2.4: Harel State chart example

$S = (A, B, C, D, E, F, G, H, I)$

$t = (A, B, C, D, E, F, G, H, I) \rightarrow (\emptyset, \emptyset, \emptyset, \emptyset, (F), \ (E, G), (H), (I), \emptyset)$

$p = (A, B, C, D, E, F, G, H, I) \rightarrow (\emptyset, (A), (A), (A), (B), (D), (D), (C), (C))$

$d = (A, B, C, D, E, F, G, H, I) \rightarrow (B, E, \ I, D, E, F, G, H, \ I)$

$E = (e_1, e_2, e_3, e_4, \ e_5)$

And the event function e is:

$$
\begin{bmatrix}
 & A & B & C & D & E & F & G & H & I \\
A & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\
B & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\
C & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\
D & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\
E & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & e_1 & \emptyset & \emptyset & \emptyset \\
F & \emptyset & \emptyset & \emptyset & \emptyset & e_2 & \emptyset & e_3 & \emptyset & \emptyset \\
G & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & e_4 & \emptyset \\
H & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & e_5 \\
I & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset & \emptyset
\end{bmatrix}
$$

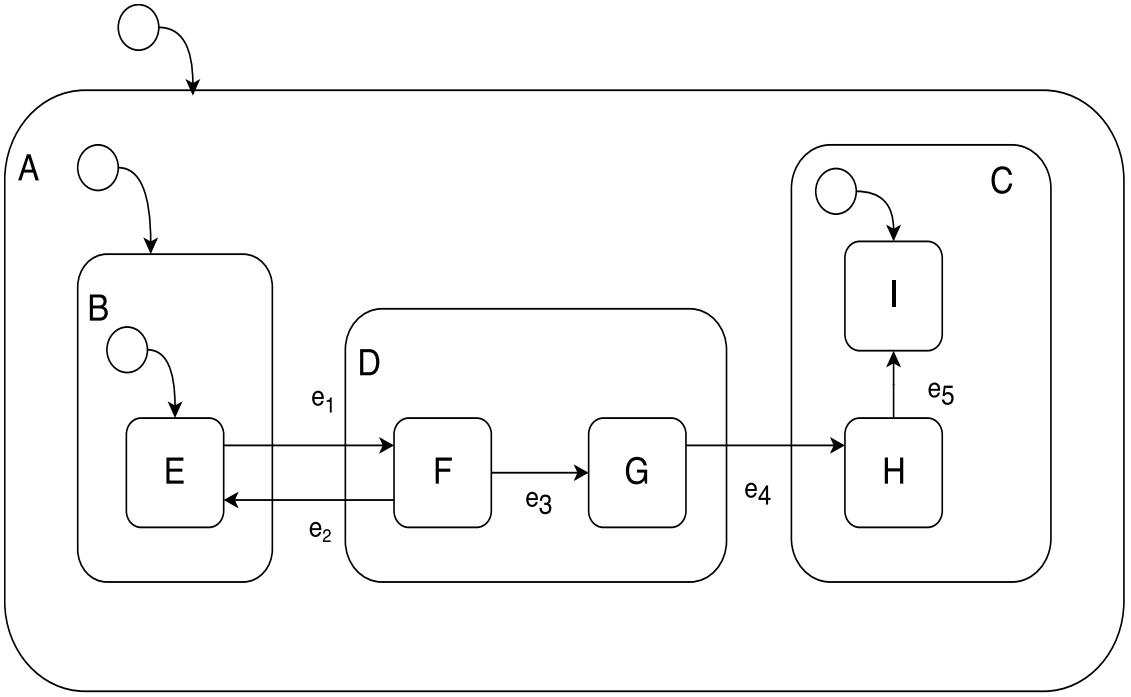

FIGURE 2.5: UML State chart example

2.5.3 Markov Model

The basis for the understanding of Markov models is the notion of Markov Chains. Markov chains are series of random variables in which the future variables are computed by the current variable only and are independent of how the computation of the current state from its antecedents has taken place. In result, the "memory less" property is characterized by them.

A stochastic process $\{X(t)|\ t \in T\}$ is called a Markov process if for any $t_0 < t_1 < \ldots . t_n < t_{n+1}$, the conditional distribution of X(t_{n+1}) for given values of X(t_0), X(t_1),..., X(t_n) depends only on X(t_n) and not on the previous values. The values that $X(t)$ can assume are in general called "states," all of which together form a "state space" Ω.

Based on the Markov chain, Markov models are seen as a graphical representation of these chains. These models can be seen as working in conjunction with State Machines which are diagrams over transitions between different states of a system.

Extending the state machine, Markov models can model entire control systems by incorporating faults and failures. It is done by assigning probabilities to transitions, namely the Markov chain probabilities. These transition probabilities are given in a transition matrix, such as

$$P = \begin{bmatrix} P_{1,1} & P_{1,2} & \dots & P_{1,m} \\ P_{2,1} & P_{2,2} & \dots & P_{2,m} \\ \dots & \dots & \ddots & \vdots \\ P_{m,1} & P_{m,2} & \dots & P_{m,m} \end{bmatrix} \tag{2.38}$$

where,

$P_{i,j}$ is the probability of transitioning between state i and j.

Furthermore, Markov processes are categorized reliant upon whether T and Ω are discrete (countable) or continuous (uncountable). Thus, reliant upon state spaces and associated time, there are four types of Markov processes. We discussed here, two types of Markov models which are commonly used. The first one is discrete time Markov chains (models), DTMC and another one is continuous time Markov chains (models), CTMC.

2.5.3.1 Discrete Time Markov Chains (DTMC)

In DTMC, the state changes at discrete points. Let $T \in \{0, 1, 2, \dots\}$ is the parameter space and $P = [p_{ij}]$ is a transition probability matrix. It is a stochastic matrix since the sum of all elements in a row of P is 1.

The state probability vector at time step n denoted by $\pi(n)$ can be iteratively computed using:

$$\pi\left(n\right) = \pi\left(n-1\right) P \tag{2.39}$$

In terms of the initial probabilities $\pi\left(0\right)$ [37],

$$\pi\left(n\right) = \pi\left(0\right) P\left(n\right) \tag{2.40}$$

where, P(n) is called n-step transition probability matrix of DTMC. Let $p_{ij}(n)$ is $(i,j)^{th}$ entry of the matrix P(n), represents the probability of reaching state j at time step n, starting from state i. Markov Chain can be of two types: (i)Irreducible: if every state can be reached from every other state. (ii) Absorbing: if there is at least one state i, from which there is no outgoing transition.

In case of irreducible DTMC, the metric of interest is the probability of being in state i at time step n and in steady state [38]. The state probability vector at time step n can be computed using Equation 2.40. To compute the state probability vector in the steady state, we take limits on both sides of Equation 2.39. This gives us the following system of equations for computing the probability vector of the system in the steady state:

$$\pi = \pi.P \tag{2.41}$$

$$\pi.e = 1 \ldots\ldots .2.32 \quad where\ e = (1, 1, .., 1)^T$$

In case of an absorbing DTMC, there are three metrics of interest for state i:

1. the probability of being in state i at time step n.
2. the probability of being in state i in the steady state.
3. expected number of visits to each one of the non-absorbing states i.

Let P is the transition probability matrix of an absorbing DTMC with A absorbing states and a total of S states. Let the transient or non-absorbing states are labeled 1,....S-A, and the absorbing states are labeled S-A + 1, . . . , S. The transition probability matrix P of an absorbing DTMC can be partitioned as:

$$P = \begin{bmatrix} Q & C \\ O & I \end{bmatrix}$$

where, Q is an $(S - A) \times (S - A)$, I is an identity matrix, O is an $A \times (S - A)$ matrix of zeros, and C is an $(S - A) \times A$ matrix. Let F(n) denote the probabilities of being absorbed in state j starting from the transient state i in n steps. Then F is given by:

$$F = \sum_{l=0}^{n} Q^l C \tag{2.42}$$

For steady state probability, $l \to \infty$

$$F = \sum_{l=0}^{\infty} Q^l C = (I - Q)^{-1} C \tag{2.43}$$

$(I - Q)^{-1}$ is called fundamental matrix M and is given by:

$$(I - Q)^{-1} = I + Q + Q^2 + \ldots. = \sum_{l=0}^{\infty} Q^l \tag{2.44}$$

It is used to compute the expected number of times the process visits state j before absorption, given that it started in state i. Let X_{ij} is the corresponding random variable. Then

$$E\left[X_{ij}\right] = A_{ij} \tag{2.45}$$

Let V_j is the expected number of times the process visits state j before absorption. Then, assuming that the DTMC starts in state i, we have:

$$V_j = A_{ij} \tag{2.46}$$

2.5.3.2 Continuous Time Markov Chains (CTMC)

Continuous time Markov chains are usable in situations where transitions between states do not occur at specific time steps as in the discrete case [38]. Here the transition probabilities, in a time interval of dt, between states i and j are given as

$$P_{i,j} = \lambda_{i,j} dt \tag{2.47}$$

, where, $\lambda_{i.j} \geq 0$ is the constant conditional failure intensity or failure rate, defined as "the probability that the component fails per unit time." Its reciprocal is the mean time to failure, hence if no transition is possible, the transition rate becomes zero. Based on this, a transition matrix can be defined as [40]

$$P = \begin{bmatrix} P_{1,1} & P_{1,2} & \dots & P_{1,m} \\ P_{2,1} & P_{2,2} & \dots & P_{2,m} \\ \dots & \dots & \ddots & \vdots \\ P_{m,1} & P_{m,2} & \dots & P_{m,m} \end{bmatrix}$$

$$= \begin{bmatrix} 1 - \sum_{\substack{k=1 \\ k \neq 1}}^{m} \lambda_{1,k} dt & \lambda_{1,2} dt & \dots & \lambda_{1,m} dt \\ \lambda_{2,1} dt & 1 - \sum_{\substack{k=1 \\ k \neq 2}}^{m} \lambda_{2,k} dt & \dots & \lambda_{2,m} dt \\ \dots & \dots & \ddots & \vdots \\ \lambda_{m,1} dt & \lambda_{m,2} dt & \dots & 1 - \sum_{\substack{k=1 \\ k \neq m}}^{m} \lambda_{m,k} dt \end{bmatrix} \tag{2.48}$$

With the transition matrix derived and ready to use, determining probabilities of being in a specific state after a given time, is done by defining $Q_j(t + dt)$, for state j at time $t + dt$.

Consider a two state Markov model as shown in Figure 2.6 with initial condition$Q_1(0) = 1$ & $Q_2(0) = 0$, it follows that

$$Q_2\left(t\ +\ dt\right) = \lambda_{1,2}dtQ_1\left(t\right) + (1 - \lambda_{2,1}dt)Q_2(t) \tag{2.49}$$

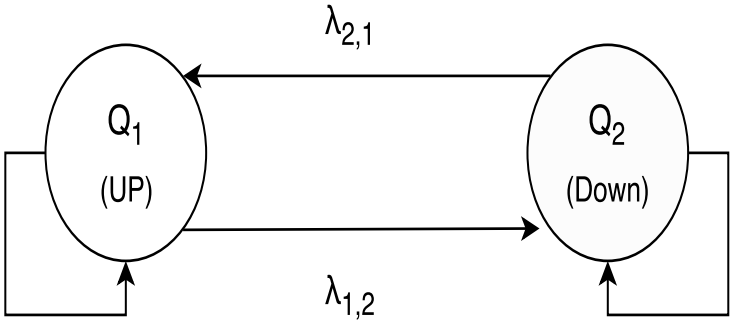

FIGURE 2.6: Simplex system example

At any time, model has

$$Q_1\left(t\right) + Q_2\left(t\right) = 1 \tag{2.50}$$

From Equation 2.49 and 2.50, we have

$$Q_2\left(t\ +\ dt\right) = \lambda_{1,2}dt(1 - Q_2\left(t\right)) + (1 - \lambda_{2,1}dt)Q_2(t) \tag{2.51}$$

$$\Rightarrow \left(\frac{Q_2\left(t\ +\ dt\right) - Q_2(t)}{dt}\right) = \lambda_{1,2} - \left(\lambda_{1,2} + \lambda_{2,1}\right)\ Q_2\left(t\right) \tag{2.52}$$

$$\Rightarrow Q_2\left(t\right) = \left(\frac{\lambda_{1,2}}{\lambda_{1,2} + \lambda_{2,1}}\right)\left(1 - e^{-(\lambda_{1,2}+\lambda_{2,1})dt}\right) \tag{2.53}$$

Generalize the above two state model into a model with n states, the probability of being in state i at time t yield

$$Q_i(t) = \left(\frac{\lambda_{incoming\ edges}}{\lambda incoming\ edges + \lambda_{outgoing\ edges}}\right) \times \left(1 - \quad e^{-\left(\lambda_{incoming\ edges}\ +\ \lambda_{outgoing\ edges}\right)dt}\right) \tag{2.54}$$

2.5.4 Petri Net

Several analysis methods primarily based on state spaces, e.g., based on some system model or system log, state spaces are created that can be utilized to verify behavioral properties like deadlocks, boundness, liveness, etc. State spaces can be inspected automatically without having any human interpretation. Even so, for an intensive knowledge of the system's behavior the analyst demands to examine and interpret the corresponding state space. State spaces are well-known for the verification and representation of critical systems [41].

By analyzing state spaces, far more insights can be acquired into the systems. Petri Nets [42] offer a properly established visual formalism in which the meaning of different components, this kind of as places, is clear. Further, a Petri Net provides ambiguity free visualization.

A PN consist of a set of places (P), transitions (T), and directed arcs (A). PN's places are represented by circles and transitions are represented by bars in the graphical notation. Arcs are connected from transitions to places and from places to transitions. Places may have tokens, which are represented by a number of black dots.

2.5.4.1 Standard Petri Net

A PN is a bipartite directed graph. Places (P) are notion by circles or ovals, whereas, transitions (T) and arc represented by squares or rectangles, and by lines (I or O) respectively. Mathematically, a PN can be represented by $C = (\ P,\ T,\ I,\ O)$ in which P and T are disjoint sets of nodes, and I and are sets of edges, where, $I\ \subseteq P \times T$, and $O \subseteq\ T \times P$ [30]. The places in a Petri Net, or P in the definition above, are represented as circles while the transitions, or T in the definition above, are represented as rectangles. The I and O represent the connections between the places and transitions. A visual example of a Petri Net is shown in Figure 2.7.

This PN in Figure 2.7 can also be described mathematically as a PN,C, where:

$C\ =\ (\ P,\ T,\ I,\ O)$

$P\ = \{\ P_1,\ P_2,\ P_3,\ P_4,\ P_5,\ P_6\}$

$T = \{\ t_1,\ t_2,\ t_3,\ t_4,\ t_5\}$

$I\,(t_1) = \{P_1, P_2\}$

$I\,(t_2) = \{P_2, P_3\}$

$I\,(t_3) = \{P_4,\ P_5\}$

$I\,(t_4) = \{P_6\}$

$I\,(\ t_5) = \{P_6\}$

$O\,(\ t_1) = \{P_4\}$

$O\,(\ t_2) = \{P_5\}$

$O\,(\ t_3) = \{P_6\}$

$O\,(\ t_4) = \{P_1\}$

$O\,(\ t_5) = \{P_3\}$

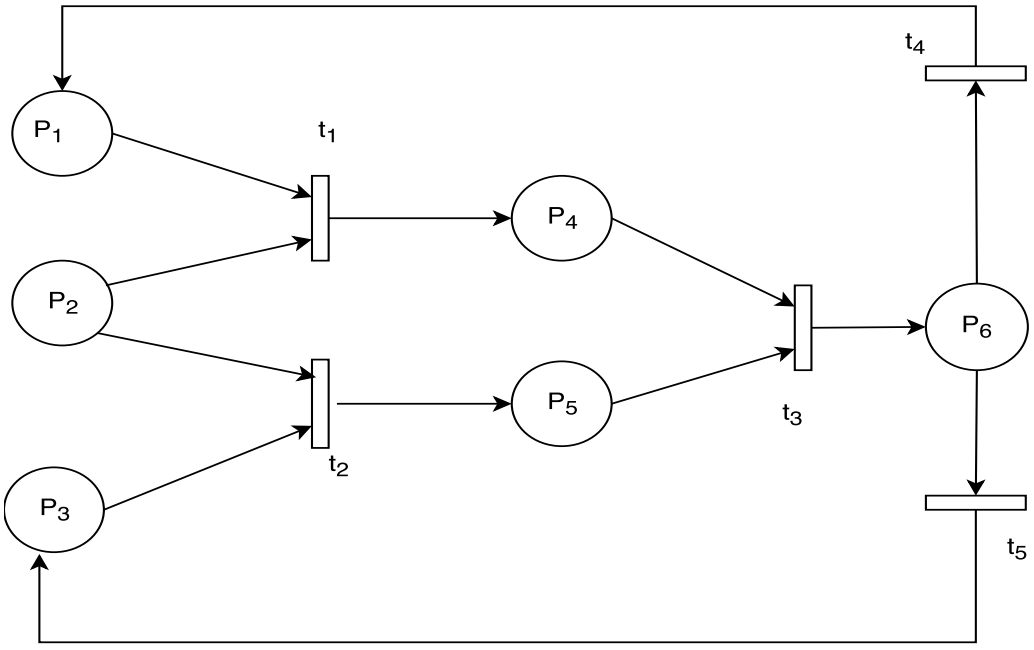

FIGURE 2.7: Petri Net visual example

2.5.4.2 Marked Petri Net

However, even though there is executable logic defined in the way the places and transitions interconnect, there is a need for data that the executable logic acts on. This gives the extension of marked Petri nets. A marked PN is a 5-tuple (P, T, I, O, M) in which (P, T, I, O) is a standard PN and M is a finite set of mappings of places into natural numbers [30]. Places store a number of tokens, initially defined by markings in M. Transitions are enabled if and only if all input places have at least one marking per connection of the enabled transitions one is fired, either randomly or by selection, at which point each input place decreased by one token per connection and each output place increased by one token per connection. A visual example of a Marked PN is shown in Figure 2.8.

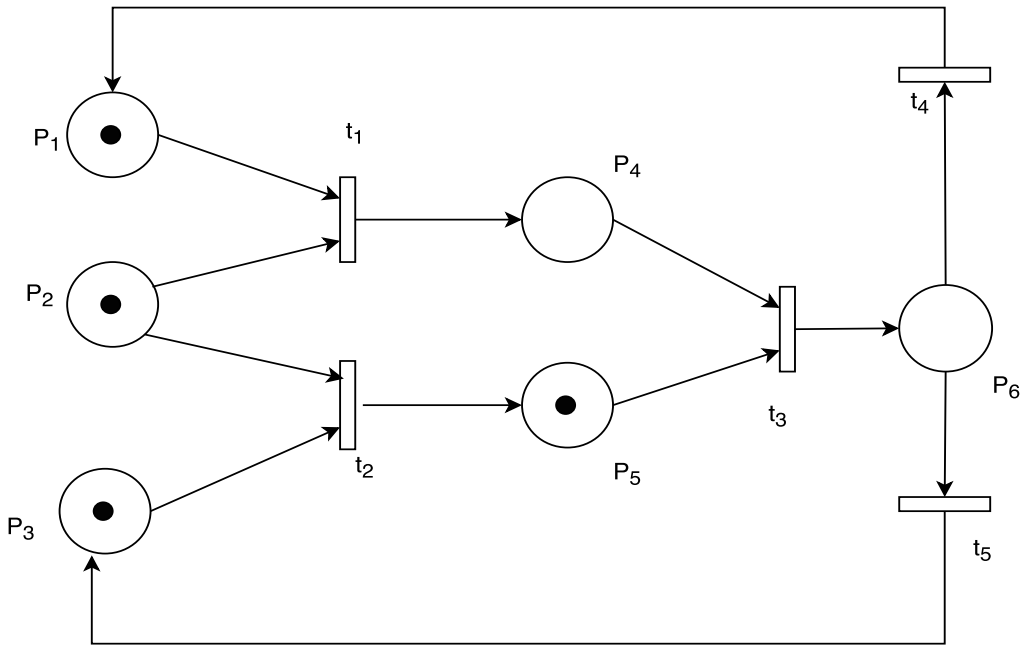

FIGURE 2.8: Marked Petri Net visual example

In the example of Figure 2.8, the initial marking $M_0 = (1, 1, 1, 0, 1, 0)$. Initially, t_1 and t_2 are enabled since all its inputs have tokens, whereas, due to unavailability of required tokens t_3, t_4, and t_5 are not enabled.

If t_1 fires then token of P_1 and P_2 decreased by one and token of P_4 increased by one. Now the new marking would be $M_1 = (0, 0, 1, 1, 1, 0)$. Similarly, when t_3 enabled and fired the new marking is $M_2 = (0, 0, 1, 0, 0, 1)$. Next, t_4 and t_5 enabled and fired, the corresponding marking would be $M_3 = (1, 0, 1, 0, 0, 0)$, $M_4 = (0, 0, 2, 0, 0, 0)$ respectively.

If t_2 fires the new marking would be $M_5 = (1, 0, 0, 0, 2, 0)$. Finally, there is no more transition enabled for firing. Hence, execution of marked PN is completed.

2.5.4.3 Event Driven Marked Petri Net

It should be noted that the PNs that have been presented thus far are all closed systems. They do not naturally interact with other systems or the environment around them. Fortunately, an extension has been created to address interaction from outside of the closed system: Event-Driven Petri nets, which are defined by 7-tuple $(E_{in}, E_{out}, P, T, I, O, M)$ in which (P, T, I, O, M) is a marked Petri net, E_{in} and $E_{out}$are set of input events and set of output events respectively.

Events, both in and out, are treated as places with self-loop token which reference activities outside of the Petri net. A graphical example is shown in Figure 2.9, where e1 and e2 are input and output events, respectively.

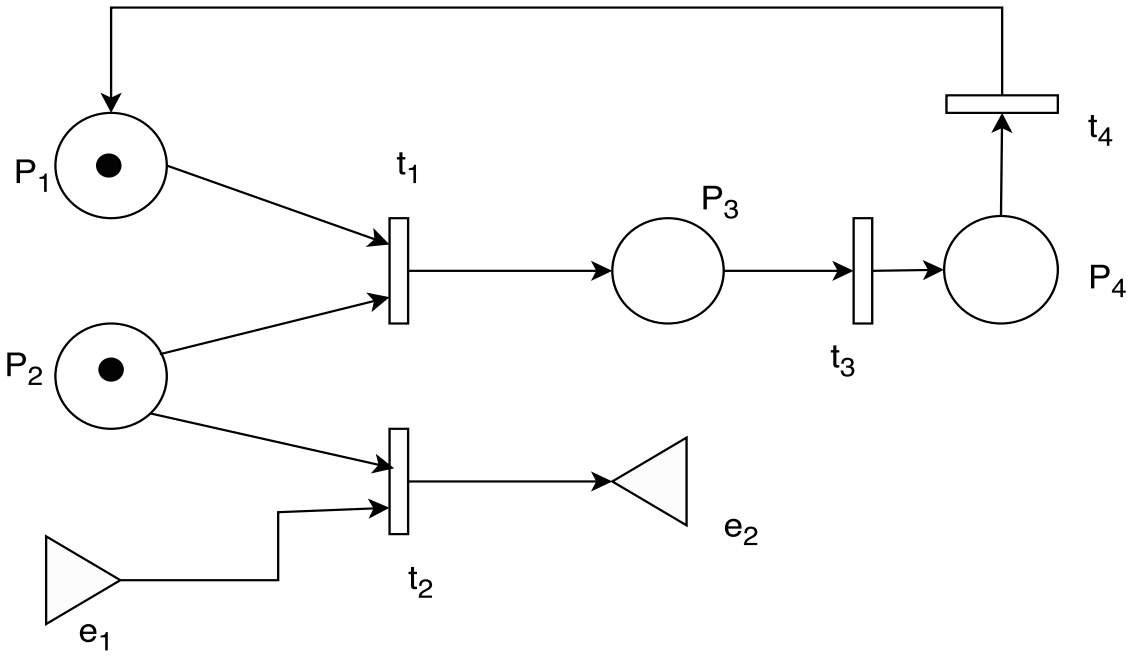

FIGURE 2.9: Event Driven Marked Petri Net visual example

Researchers are continuously paying the effort to obtain the properties of the model to fit their applicability in modeling real systems. Authors have extended and continue to extend the basic model PN to obtain more useful modeling tool that can be fitted to model different types of real problems like performance evaluation, and scheduling problems of dynamic systems.

2.6 Conclusion

This chapter gives foundation related to safety needed to comprehend the further thinking all through this thesis is presented. It also discusses dependability as safety attribute. Further, in order to design and evaluate the models, various state space model has been adopted in the course of action. These state space models: state

transition machine, Harel state chart, UML state chart diagram, Markov model and PN are presented and discussed.

After, having explained the preliminaries, concepts, definitions, acronyms, terminology now a literature review for the identify state-of-the-art for reliability and safety is possible be in Chapter 3, are reporting such a systematic review work conducted by us is appears in Chapter 3. The full contribution of thesis includes the review work presented in Chapter 3.