

Chapter 6

Dependability Analysis Techniques

Prediction is very difficult, especially if it's about the future.
— Anonymous

Abstract An overview of the techniques traditionally used in dependability analysis that are compliant with current industrial standards (i.e., the International Electrotechnical Commission standards) is provided. In particular, the focus is on those techniques that are chosen as target formalism of model transformations (described in Chap. 7), that is Fault Tree analysis and Petri Net analysis.

6.1 Introduction

Many analysis techniques have been applied by the engineers to assess the system dependability during the last four decades. Such techniques are used for the prediction, verification, and improvement of dependability properties, mainly reliability, availability, maintainability, and safety. They help in answering the questions that are most pressing to an engineer, such as: Is the flight control system able to tolerate N simultaneous equipment failures? When a shutdown occurs, how long does it take to recover the system? Is the system able to provide the service to the user during a given time period? And so on.

A list of the primary techniques, recommended by the international standard (IEC-60300-3-1 2003) for the dependability assessment, is shown in Table 6.1: they are mainly used early in the software life cycle, that is during the requirement and design activities. It is worth to mention that the list is a coarse classification provided by the standard of the primary dependability techniques: for example, the Petri Net analysis technique encompasses all the Petri Net semantics together with the correspondent solution methods.

There is no a general rule for the selection of the best technique to be applied for assessing the dependability of a specific system. Nevertheless, the IEC standard

provides several guidelines to support the engineer in the choice of the most appropriate ones. Herein, we summarize the main criteria that should be considered for the technique selection.

Table 6.1 Primary dependability analysis techniques IEC-60300-3-1 (2003)

Technique	Other standards
Event Trees Analysis (ETA)	IEC-62502 (2010)
Failure Mode and Effect Analysis (FMEA)	IEC-60812 (1985), MIL-STD-1629a (1980), ANSI/IEEE-STD-352 (1987), SAE-ARP-4761 (1996); BS-5760-5 (1991)
Failure Mode, Effect, and Criticality Analysis (FMECA)	IEC-60812 (1985); MIL-STD-1629a (1980); BS-5760-5 (1991)
Fault Trees Analysis (FTA)	IEC-61025 (2006), ANSI/IEEE-STD-352 (1987), SAE-ARP-4761 (1996)
Functional Failure Analysis (FFA)	SAE-ARP-4761 (1996)
HAZard and OPERability studies (HAZOP)	IEC-61882 (2001)
Markov analysis	IEC-61165 (2006), ANSI/IEEE-STD-352 (1987)
Petri Net analysis (PN)	ISO/IEC-15909-1 (2004)
Preliminary Hazard Analysis (PHA)	MIL-STD-882c (1993), MIL-STD-882d (2000)
Reliability Block Diagrams analysis (RBD)	IEC-61078 (2006), ANSI/IEEE-STD-352 (1987)
Truth table	ANSI/IEEE-STD-352 (1987)

6.1.1 Applicability in the Life Cycle

The techniques in Table 6.1 are typically applied during the requirement and design stages of the software development process, and their applicability depends on the available system specification.

The *early* stages techniques (e.g., PHA, FFA, HAZOP) are used in the requirement analysis or at the very beginning of the design process and they focus on the analysis of the abstract concepts of the system. They enable to identify potential system failure modes then providing a feedback to the engineer for the definition of the system architecture.

The *late* techniques (e.g., Markov analysis, PN, Truth tables) are used when the detailed design specification becomes available. Such techniques allow to evaluate system reliability/availability and to perform trade-off analysis on different design alternatives. Finally, there are techniques (e.g., ETA, FMEA, FMECA, FTA, RBD) that can be used *across* different stages: initial models are constructed during the requirement analysis and, then, refined to a more detailed level as data become available in order to make decisions and trade-offs.

6.1.2 Aim of the Analysis

The aim of the analysis can be either *qualitative* or *quantitative*, in turn, we can use the same criteria to classify the supporting techniques. In particular, there are techniques that only support qualitative analysis, such as PHA, HAZOP, and FFA, while most of the remaining ones in the list can be used for both qualitative and quantitative analyses. The main objectives of the qualitative analysis are the identification of the component failure modes, their consequences at system level, and the cause–effect paths, as well as the determination of possible repair/recovery strategies. The quantitative analysis aims at defining numerical reference data to be used as input parameters of reliability/availability models, estimating availability/reliability metrics, under probabilistic or stochastic assumptions, performing component criticality and sensitivity analysis.

6.1.3 Bottom-Up/Top-Down

Bottom-up methods mainly deal with the effects of single faults and the starting point of the analysis is the identification of the faults at component level. Each component fault is considered separately and its effects at the next higher system level are studied. The analysis is iterated to discover the effects at all functional levels up to the system level (e.g., the user-service level). On the other hand top-down methods enable to analyze the effects of multiple faults. The analysis starts by considering a failure mode at the highest level of interest and then proceeds backward by identifying the causes of the failure.

In practice, to ensure the completeness of the analysis, combinations of bottom-up and top-down techniques are often applied. For example, cause-consequence analysis is carried out by using both ETA and FTA, where the root of the event tree is the top event of a fault tree; the FTA is applied to analyze the causes of a failure, while ETA is applied to analyze the consequence of initiating events.

6.1.4 Cause–Effect Relationships Exploration

This criterion, proposed by Mauri (2000) for safety analysis techniques, classifies the techniques depending on how they explore the relationship between causes and effects. There are different ways to proceed: *deductive* techniques start from known effects to seek unknown causes (e.g., FTA, FFA), *inductive* techniques start from known causes to forecast unknown effects (e.g., ETA, FMEA, FMECA), while *exploratory* techniques relate unknown causes to unknown effects (e.g., HAZOP, PHA).

6.1.5 Dependencies Modeling

An important criterion to be taken into account when selecting a dependability analysis technique is the capability of modeling time or sequence-dependent behavior as well as state-dependent events. For example, after failure, the system operates in degraded mode (time-dependency); the system fails only if event A is preceded by B and not vice versa (sequence-dependency); different failure or repair characteristics depending on the system state (state-dependency). A similar criterion is provided by Nicol et al. (2004) considering quantitative analysis techniques, where the latter are classified into combinatorial and state-based. Combinatorial techniques, such as FTA and RBD, do not enumerate all possible system states to obtain a solution and dependability measure are computed by adopting simpler approaches. However, they do not easily capture the aforementioned types of dependencies and imperfect coverage. On the other hand state-based techniques, such as Markov analysis and PN, are more comprehensive and they enable explicit modeling of complex relationships.

Table 6.2 Characteristics of the dependability analysis techniques IEC-60300-3-1 (2003)

Technique	Life-cycle	Aim	Bottom-up/ Top-down	Cause-effect relationship exploration	Dependencies modeling
ETA	across	ql./ qn.	bottom-up	inductive	yes
FMEA	across	ql.	bottom-up	inductive	no
FMECA	across	qn.	bottom-up	inductive	no
FTA	across	ql./ qn.	top-down	deductive	no
FFA	early	ql.	bottom-up	deductive	no
HAZOP	early	ql.	bottom-up	exploratory	no
Markov analysis	late	qn.	top-down	NA	yes
PN	late	ql./ qn.	top-down	NA	yes
PHA	early	ql.	bottom-up	exploratory	no
RBD	across	ql./ qn.	top-down	NA	no
Truth table	late	ql./ qn.	NA	inductive	no

NA: the criterion is not applicable with respect to this technique

Table 6.2 summarizes the characteristics of the primary dependability analysis techniques (Table 6.1) according to the aforementioned criteria: the applicability in the life cycle (second column); the aim (third column), i.e., qualitative (ql.) vs/ quantitative (qn.); bottom-up vs/ top-down (fourth column); the type of exploration of the cause–effect relationship (fifth column); and, finally, whether they support the modeling of dependencies (sixth column).

In the rest of this chapter, we focus on those analysis techniques that are the target of the transformation approaches considered in Chap. 7, namely FTA (Sect. 6.2) and PN analysis (Sect. 6.3). Other techniques and software tools will be briefly presented in Sects. 6.4 and 6.5, respectively.

6.2 Fault Tree Analysis

A Fault Tree (FT) is an acyclic graph with internal nodes that are logic gates (e.g., AND, OR, K-of-M), external nodes that represent component/sub-system faults (basic events, undeveloped events) and, possibly, transfer gates. The latter are used to indicate that the rest of the tree is developed in another part or page of the diagram; they are suitable for the construction of models following a hierarchical approach, then enhancing the readability as well as the reusability (e.g., the same sub-tree can contribute to the occurrence of different failure modes, then it can be reused in different fault-tree models).

FTs are used to represent and analyze the component faults whose combined occurrence causes a system failure to occur. It is a top-down technique, then the FT construction begins by considering a system failure mode—the *top event*—and terminates when either the basic events which provoke such a failure are all identified or the desired level of detail is reached. When several failure modes need to be investigated, then different FTs should be generated, one for each failure mode.

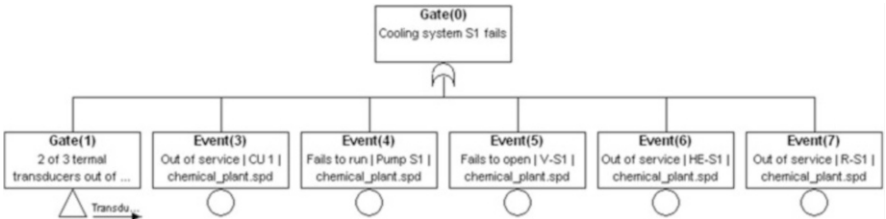


Fig. 6.1 Failure of the cooling system

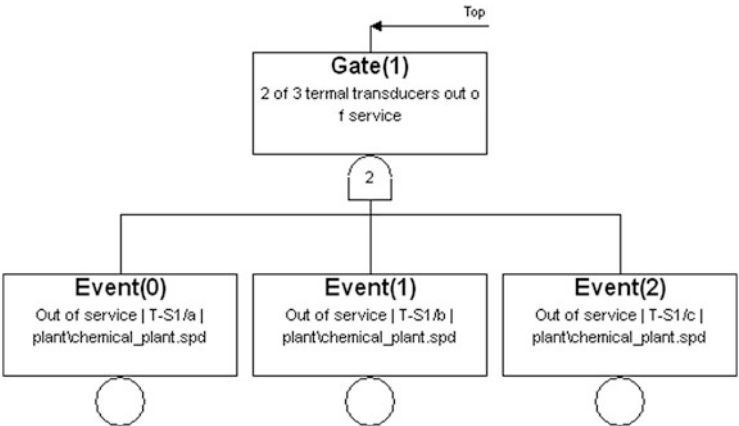


Fig. 6.2 Failure of 2-out-3 thermal transducers

Figure 6.1 shows an example of FT modeling the cooling system failure in a chemical power plant (Contini et al. 1999). The top event (*Gate(0)*) is caused by one of the events at the lower level, which are then connected with an OR gate. The *Event(i)* ($i = 3, \dots, 7$) are basic events, while *Gate(1)* is a transfer gate that refers to the Fault Tree of Fig. 6.2. The latter represents the failure of a redundant component. Indeed, there are three thermal transducers and the failure occurs when at least two of them are out of service: then the basic events *Event(i)* ($i = 1, 2, 3$) are connected with a 2-of-3 logic gate.

Originally, in FT only independent basic events could be modeled. Later, to enhance the modeling capabilities, FTs have been extended to include various types of gates. In particular, *Dynamic Fault Trees* (DFT), defined by Dugan et al. (1992), include special purpose gates capturing sequence dependencies which frequently arise when modeling fault tolerant computer systems. Such special gates are: *functional dependency* gate, for modeling situations where one component’s correct operation is dependent upon the correct operation of some other component; *spare* gate, for modeling cold, warm, and hot pooled spares; and *priority-AND* gate, for modeling ordered AND-ing of events. Figure 6.3 summarizes the basic event

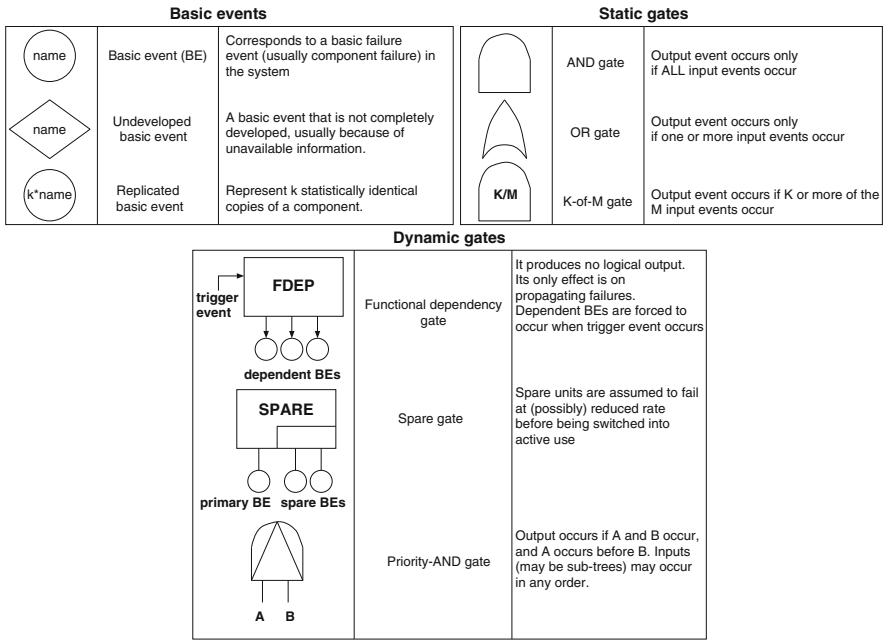


Fig. 6.3 Basic event construct, static and dynamic gates

constructs and the static gates common to both FT and DFT, as well as the dynamic gates of DFT.

Once an FT model is built, both qualitative and quantitative analyses can be carried out. The qualitative analysis is aimed at identifying all the combinations of basic events that cause the top event to occur. Combinations are called *cut sets* and they are ranked according to the number of basic events, since the smaller the number of events that cause the top event, the less resilient to failure is the modeled system. A *minimal cut set (MCS)* is a cut set that does not contain any subset of basic events that is still a cut set. The list of all the *MCSs* provides a valuable information to the analyst since it enables the identification of the potential weak points of the system, an essential step to initiate the corrective actions. Table 6.4 lists the MCSs (second column), ordered by rank, of the FT example.

The main objectives of the quantitative analysis are the evaluation of the probability of occurrence of the top event, considering the system mission time, and the identification of the MCSs which contribute most significantly to the system failure. The probability may correspond to the unreliability or unavailability metrics, computed at a given time instant, associated with the system (see Chap. 2); observe that, for non-repairable systems, the two metrics are equivalent.

The FTA assumes that the basic events, i.e., the component faults and, in case, the component repairs are statistically independent events. Many FT tools accept as input parameters constant fault/repair rates, thus implicitly assuming that the failure and repair times of each component are exponentially distributed and restricting the analysis to this case. Table 6.3 shows the input parameters of the FT example, where all the components are repairable. There exist several FT solution algorithms,

Table 6.3 Input parameters of the FT example

Events	Fault rate [1/hours]	Repair rate [1/hours]
Event(0), Event(1), Event(2)	$2.4 \cdot 10^{-5}$	$2.5 \cdot 10^{-1}$
Event(3)	$8.9 \cdot 10^{-5}$	$1.25 \cdot 10^{-1}$
Event(4)	$3.7 \cdot 10^{-4}$	$8.3 \cdot 10^{-2}$
Event(5)	$8.1 \cdot 10^{-5}$	$8.3 \cdot 10^{-2}$
Event(6)	$2.4 \cdot 10^{-5}$	$4.1 \cdot 10^{-2}$
Event(7)	$9.1 \cdot 10^{-5}$	$4.1 \cdot 10^{-2}$

mainly based on sums of disjoint products (Rai et al. 1995) and binary decision diagrams (Rauzy 1993; Zang and Trivedi 1999). The former rely on the computation of the MCSs and they are less efficient than the latter. The binary decision diagram techniques are normally used for solving very large fault trees. Concerning DFTs, when a dynamic gate is part of a model then the latter is solved via a Markov chain, rather than by using traditional methods.

Table 6.4 shows the results of the FT analysis of the example, computed for a given mission time, using the StarsStudio-Astra tool (Contini et al. 1999); in particular, the unavailability of each MCS (third column) and the cumulated unavailability (fourth column). The MCSs are ordered according to their unavailability, from the highest to the lowest. The cumulated unavailability at the i^{th} row of the table corresponds to the probability of failure of at least one of the first i MCSs,

i.e., $Pr\{MCS_1 \text{ OR } MCS_2 \text{ OR } \dots \text{ OR } MCS_i\}$. The cumulated unavailability at the last row corresponds to the unavailability of the top event.

Table 6.4 The MCSs of the example and analysis results (mission time 8.760 hours)

N	MCS	Rank	Unavailability	Cumulated Unavail.
1	Event(7)	1	$3.986 \cdot 10^{-1}$	$3.986 \cdot 10^{-1}$
2	Event(3)	1	$1.949 \cdot 10^{-1}$	$5.158 \cdot 10^{-1}$
3	Event(4)	1	$1.850 \cdot 10^{-1}$	$6.054 \cdot 10^{-1}$
4	Event(6)	1	$1.051 \cdot 10^{-1}$	$6.469 \cdot 10^{-1}$
5	Event(5)	1	$4.050 \cdot 10^{-2}$	$6.612 \cdot 10^{-1}$
6	Event(2), Event(0)	2	$1.473 \cdot 10^{-2}$	$6.662 \cdot 10^{-1}$
7	Event(1), Event(2)	2	$1.473 \cdot 10^{-2}$	$6.711 \cdot 10^{-1}$
8	Event(0), Event(1)	2	$1.472 \cdot 10^{-2}$	$6.759 \cdot 10^{-1}$

6.3 Petri Net Analysis

Petri Net analysis relies upon the use of Petri Nets (PN) as modeling formalism, defined by C.A. Petri in 1962 as part of his thesis dissertation. A PN is a bipartite directed graph, where P and T are the disjoint sets of nodes, namely *places* and *transitions*. The former, signified by circles, are used to model conditions; the latter, graphically depicted by bars, represent events/activities that may occur in the system. The directed arcs $A = (P \times T) \cup (T \times P)$, shown by arrows, describe which places are pre- or post-condition for which transitions, they are called, respectively, input and output places. Places may contain tokens (drawn as black dots) and a token assignment over the set of places is called *marking*.

The PN behavior is governed by the transition enabling and firing rules. A transition is *enabled* in a given marking, when all of its input places contain at least one token. An enabled transition may *fire* and, upon firing, it removes one token from each of its input places and add one token in each of its output places, causing a change of marking (i.e., a change of state).

Given an initial marking of a PN, it is possible to compute the set of all the markings reachable from the initial one by transition firings (i.e., reachability set), provided that every place of the net is bounded. The reachability graph associated with a PN is a directed graph whose nodes are the markings in the reachability set and each arc connecting a marking M to a marking M' represents the firing of a transition enabled in M and leading to M' .

Since the Petri's original proposal, many classes of PN have been proposed in the literature in order to enhance their modeling and analysis capabilities. In particular, Stochastic Petri Nets (SPN) are of interest when dependability analysis is a concern; they are defined as extensions of un-timed PN, where each transition is characterized by a random variable exponentially distributed, modeling its firing time.

SPN is a modeling formalism at higher level of abstraction with respect to Continuous Time Markov Chains (CTMCs). An SPN is usually solved by deriving

the underlying CTMC, which is isomorphic to its reachability graph, and then by using the wide variety of available techniques for CTMCs. Nevertheless, such techniques suffer the well-known state space explosion problem, then for large complex models alternative techniques are more suited, such as discrete-event simulation (Kelling 1996), approximation techniques and bound techniques based on the solution of linear programming problems (Campos and Silva 1992).

Several extensions have been proposed to the basic SPN formalism, they include: arcs with multiplicity, immediate transitions that fire with zero time delay (depicted by thin black bars), transition priorities, inhibitor arcs from places to transition that provide the zero-test capability (depicted by a small circle arrow-end). The most popular variant of SPN are Generalized Stochastic Petri Nets (GSPN—Ajmone-Marsan et al. 1994). Other extensions allow the modeler more flexible firing rules, such as Stochastic Activity Networks (SAN—Sanders and Meyer 2001). Similarly, Stochastic Reward Nets (SRN—Muppala et al. 1993) and Stochastic Well-Formed Nets (SWN—Chiola et al. 1993) include guards and enabling functions.

On the other hand, there have been several proposals on relaxing the “exponential distribution transition firing time” assumption, then making the stochastic process associated with the SPN not still a CTMC. For example, Extended Stochastic Petri Nets (ESPN—Dugan et al. 1985) are SPN where transitions can be characterized by general firing time distributions. Their underlying process is a semi-Markov process when certain restrictions are met. Deterministic and Stochastic Petri Nets (DSPN—Lindemann 1998) have immediate, exponential, and deterministic transitions. The stochastic process associated with a DSPN is a Markov regenerative process. Markov Regenerative Stochastic Petri Nets (MRSPN—Choi et al. 1994) and Concurrent Generalized Stochastic Petri Nets (CGSPN—Vita et al. 1995) are generalizations of DSPNs and still have a Markov regenerative process associated. Numerical techniques, based on the solution of the underlying stochastic process, have also been proposed for the aforementioned SPN variants.

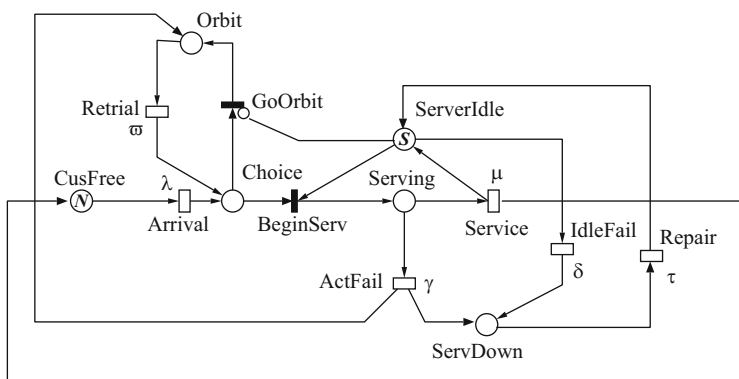


Fig. 6.4 GSPN model of multiserver retrial system with dependent breakdowns

Figure 6.4 shows the GSPN model of a multiserver retrial system (from Gharbi and Dutheillet 2011) with N customers (initial marking of place *CusFree*) and a service station characterized by S identical and parallel servers (initial marking of place *ServerIdle*). Each customer is either free (place *CusFree*), under service (*Serving*) or in orbit (*Orbit*). On the other hand, each server can be either operational—idle (*ServIdle*) or busy (*Serving*)—or non-operational (*ServDown*). Customer requests are assigned to operational idle server randomly. If there is an idle server at the moment of the request arrival, then the latter is processed immediately. Otherwise, if all the servers are not available (i.e., busy or down), the customer joins the orbit. These two alternative conditions are modeled by the two conflicting immediate transitions *BeginServ* and *GoOrbit*, respectively. All the timed transitions are characterized by exponentially distributed firing times (the firing rate parameters are shown near the transitions). Observe that it is possible to model a state-dependent breakdown discipline: indeed, the server can fail either in idle state or in busy state with different failure rates (i.e., $\delta \neq \gamma$).

The SPN models can be used both to verify qualitative properties and to compute metrics. In the dependability context, qualitative analysis may consist in verifying the unreachability of unsafe states, e.g., by using model checking techniques on the SPN reachability graph, and validating the model consistency via structural analysis techniques (which are based on the property of the SPN as a graph).

From the quantitative analysis point of view, SPNs are used to compute dependability metrics, under transient or steady-state assumptions. In transient analysis, the model behavior is observed during a finite time interval, while steady-state analysis assumes the observation of the model behavior during a *sufficiently large* period, such that the SPN state variables, and then dependability metrics, are time-independent. Typically, reliability/unreliability and instantaneous availability metrics are computed under transient analysis assumptions while mean value metrics, such as MTTF, steady-state availability and failure rate can be estimated under the steady-state assumption. The input parameters of an SPN model boil down to transition firing rates/weights and, possibly, place initial markings: Table 6.5 shows the input parameters of the GSPN model in Fig. 6.4.

Table 6.5 Input parameters of the GSPN example

Transition	Rate Param.	Value [1/s.]	Place	Marking Param.	Value
Arrival	λ	4	CusFree	N	50
Service	μ	10	ServerIdle	S	5
Retrial	ϖ	1			
ActFail	γ	$5 \cdot 10^{-2}$			
IdleFail	δ	$1 \cdot 10^{-1}$			
Repair	τ	$5 \cdot 10^{-1}$			

Dependability metrics of an SPN model can be defined as reward functions, considering the associated underlying CTMC (Goseva-Popstojanova and Trivedi 2000). For example, let

$$r_M = \begin{cases} 1 & \text{if } M \in O \\ 0 & \text{if } M \in F \end{cases}$$

a state reward that partitions the set of reachable markings $RS(M^0)$ of the SPN into two subsets of markings: O that represents the set of operational system states and F that represents the system failure states.

The most simplest availability measures are based on such a reward:

$$\begin{aligned} A(t) &= \sum_{M \in RS(M_0)} r_M \sigma_M(t) = \sum_{M \in O} \sigma_M(t) \\ A_\infty &= \sum_{M \in RS(M_0)} r_M \sigma_M = \sum_{M \in O} \sigma_M \\ \bar{A}(t) &= \frac{1}{t} \sum_{M \in RS(M_0)} r_M \int_0^t \sigma_M(\tau) d\tau = \frac{1}{t} \sum_{M \in O} \int_0^t \sigma_M(\tau) d\tau \end{aligned}$$

where $A(t)$, A_∞ , $\bar{A}(t)$ are, respectively the instantaneous, steady-state and interval availability metrics, and $\sigma_M(t) = Pr\{X(t) = M\}$, $\sigma_M = \lim_{t \rightarrow \infty} \sigma_M(t)$ are the transient and steady-state probabilities of the marking process associated with the SPN being in marking M at time instant $t \geq 0$. Similar reward functions can be defined for reliability metrics.

Considering the GSPN model in Fig. 6.4, several dependability metrics can be computed depending on the definition of the set of operational states (i.e., the subset O). We can compute, e.g., the steady-state availability of at least $0 < N_A \leq S$ servers, by defining the subset O as follows:

$$O = \{M \in RS(M_0) \mid M(\text{ServerIdle}) + M(\text{Serving}) \geq N_A\}.$$

Table 6.6 shows the results computed using the GreatSPN (2002) tool.

Table 6.6 Steady-state availability of at least $0 < N_A \leq S$ servers

N_A	A_∞
1	0.999988
2	0.999464
3	0.990342
4	0.911475
5	0.573473

6.3.1 Non-stochastic Petri Nets

Besides SPN, other classes of non-stochastic Petri Nets, enhanced with timing specification capabilities, have been proposed in the literature. The most well-known are Time Petri Nets (TPN—Berthomieu and Diaz 1991), where each transition is

characterized by an earliest and a latest firing time $[a, b]$. The latter are relative to the instant at which the transition was last enabled. Hence, if the transition has been last enabled at time τ , then it may not fire before $\tau + a$ and it must fire before or at $\tau + b$, unless it is disabled before by the firing of a conflicting transition. Most of the analysis techniques are enumerative, i.e., they are based on the construction of the state space associated with the TPN model. TPN have been applied mainly for the verification of timeliness properties. However, there are also some proposals for the safety assessment (Leveson and Stolzy 1987) and the timing failure risk assessment (Bernardi et al. 2011a).

6.4 Other Techniques

In the following, the other standard techniques considered in Tables 6.1 and 6.2 are briefly presented: for an in-depth analysis, we suggest the reader to refer to the standards indicated in Table 6.1 and to the bibliography at the end of the book.

6.4.1 Event Tree Analysis

Event Tree analysis (ETA) is an inductive technique used to evaluate the consequences of an initiating event and the probability of each of the possible sequences that can occur.

The Event Tree (ET) is a logical structure suitable to model the consequences of the initiating event (e.g., a node breakdown), identifying the states (success or unsuccess) of all the mitigation systems; the result is a set of different possible scenarios, each associated with an occurrence probability.

The Fault Tree analysis is generally used to calculate the probabilities of event occurrences. Indeed, each event (branch) in the ET can be interpreted as the top event of an FT: the value thus computed represents the conditional probability of the occurrence of the event, given that the events which precede on that sequence have occurred. Multiplication of the conditional probabilities for each branch in a sequence gives the probability of that sequence.

In the case of structural dependencies it is possible to combine ET and FT techniques in a profitable way, linking one FT to each ET branch. This combined technique is called *ET with boundary conditions* (Stapelberg 2008) and consists in decomposing the system so as to identify the supporting part or functions upon which some components and systems are simultaneously dependent. The supporting parts thereby identified appear explicitly as system event tree headings, preceding the dependent protection systems and components. Since the dependent parts are extracted and explicitly treated as boundary condition in the ET, all the conditional probabilities are made independent and the probability of the accident sequences can be computed by simple multiplications.

6.4.2 FMEA and FMECA

Failure Mode and Effect Analysis (FMEA) is an inductive analysis technique used to study the effects of component failure modes on a system. FMEA starts from knowledge of component failure modes and considers the effects of each failure on sub-systems and the system. It implies the study of all the components in a system and is often applied to higher-level assemblies and systems. FMEA helps to check whether the components, with their known failure modes, fulfill system level safety requirements. The results of the FMEA may be to accept the proposed components or, perhaps, to issue recommendations for maintenance checks, or to ask for components to be replaced.

It is common to use FMEA to determine the presence or absence of single points of failures in a system design. FMEA is basically a qualitative technique; Failure Mode, Effect and Criticality Analysis (FMECA) extends FMEA by introducing a criticality analysis to verify whether failure modes with severe effects have sufficiently low occurrence probability. Both the techniques produce tabular outputs.

6.4.3 Functional Failure Analysis

FFA techniques is applied to the early system design in order to identify hazards from a functional perspective. Indeed, the objective of FFA is the identification of the system functions that contribute to hazards, and thus assigning them a criticality level. It is a deductive technique that considers three types of failure condition: (1) function not provided when requested; (2) function provided when not required; and (3) malfunction. The output is a set of tables which give for each function, for each failure condition, and for each operational state, a description of effects, mitigation procedures, and often the type of analysis that has to be performed to have the system accepted by regulatory authorities.

6.4.4 HAZard and OPerability Studies

HAZard and OPerability study (HAZOP) is a structured and systematic examination of a planned or existing process/operation in order to identify and evaluate hazards to personnel, equipment, or operation. It was initially developed to analyze chemical process systems in the early 1970s but has later been extended to other types of systems, including software systems in the early 1990s.

HAZOP is a qualitative technique based on guidewords and it is often carried out by a multidisciplinary team during the early stages of the system life cycle (i.e., requirement analysis, high-level design) to anticipate hazards. A guideword (e.g., none, more of, less of, part of, more than, other) describes a hypothetical

deviation from the normally expected characteristic of a process flow. Driven by these guidewords, failure causes and their effects are listed. Each effect is then considered and actions are proposed either to mitigate it or to decrease the likelihood of the failure cause. The information gathered with HAZOP is represented in tabular form.

6.4.5 Markov Analysis

Markov analysis is a stochastic technique that enables to specify the dependence of failure or repair characteristics of individual components on the state of the system. It is a technique suitable for the dependability evaluation of complex system structures and complex repair and maintenance strategies. It should be observed that Markov analysis often represents the basis of some previously introduced formalisms, such as SPN and DFT.

The simplest Markov model is a Markov chain, that is a Markov process with a discrete state space. A Markov chain can be defined for a discrete set of times (i.e., discrete-time Markov chain—DTMC) or for time taking nonnegative real values (i.e., continuous time Markov chain—CTMC). For dependability applications, the normal reference model is the CTMC. A lot of literature exists on the topic, the interested reader may refer, for example, to Papoulis (1965), Cox and Miller (1965), Kulkarni (1995), Trivedi (2001).

Let $Z(t)$ be a stochastic process defined over the discrete state space Ω . $Z(t)$ is a CTMC if, given any ordered sequence of time instants ($0 < t_1 < t_2 < \dots < t_m$), the probability of being in state $\mathbf{x}^{(m)}$ at time t_m depends only on the state occupied by the system at the previous instant of time t_{m-1} and not on the complete sequence of state occupancies. This property, which is usually referred to as the *Markov property*, can be rephrased by saying that the future evolution of the process only depends on the present state and not on the past. Formally, the Markov property may be written as:

$$\begin{aligned} &P\{Z(t_m) = \mathbf{x}^{(m)} \mid Z(t_{m-1}) = \mathbf{x}^{(m-1)}, \dots, Z(t_1) = \mathbf{x}^{(1)}\} \\ &= P\{Z(t_m) = \mathbf{x}^{(m)} \mid Z(t_{m-1}) = \mathbf{x}^{(m-1)}\}. \end{aligned}$$

6.4.6 Preliminary Hazard Analysis

PHA, a qualitative technique, is used early in the system life cycle (i.e., requirement analysis and early design) and is aimed at identifying the safety critical areas, providing an initial assessment of hazards and defining requisite hazard controls and subsequent actions. It is not a formal technique, typically consists in brainstorming with the use of checklists to help in the identification of hazards. The technique produces tabular output.

6.4.7 Reliability Block Diagram Analysis

A Reliability Block Diagram (RBD) is a graphical representation of the system's components and connectors. An RBD consists of blocks and lines: the blocks represent system components (generally, hardware components), the lines describe the connections between components. If any path through the system is successful, then the system succeeds, otherwise it fails.

It is assumed that connectors have reliability one (i.e., do not fail) and that both the system and its components are in one of two states: either up or down. Hence, in an RBD, each component can be seen as a switch that is closed when the component is up and open when the component is down. The system will be up only when a path exists between the input and output nodes.

An RBD can always be constructed as connected groups of three basic patterns: (a) components in series; (b) components in active redundancy; or (c) components in standby redundancy. Components in series is the simplest form where if one or more of the components are down then the system is down.

When the system contains redundant structures, then not all of the components belonging to a redundant structure are required to be up for successful operation of the system. An " m/n redundant group" is a group of n components where only (any) m of them has to be up for the group to be considered up. There are two forms of redundancies: active and standby redundancy.

In the case of active redundancy all the n components in the group are active when the system is operating, but only m components are necessary to be up for the group to be up. In a standby redundancy, only m components are required to be in an active state and the remainders ($n - m$) are in passive state. When an active component fails then one of the passive components is switched on in its place. The failure time distribution associated with a component depends on whether the latter is in an active or passive state (the failure rate of a component in an active state is generally much larger than its failure rate when it is in a passive state).

RBD is both a qualitative and quantitative (stochastic) technique: in the latter case, it can be used to evaluate the system reliability or availability, given the reliability, availability of the individual components. The failures/repairs of the components are assumed statistically independent.

6.4.8 Truth Table

The truth table method consists of listing all possible state combinations (operating state, failed state) for the various components that make up a system and studying their effects. The first steps in the application of the method are similar to those of a FMECA. The failure modes of the components as well as their failure states should be listed once the system has been broken down into a manageable size. The system state is represented by a state vector that is a combination of

component states, each component being represented by either its operating state or its failed state. The truth table is completed by analyzing the effect of all the component state vectors: a system failure state is labeled as 0 (zero) and a system operational state is labeled as 1 (one). The probability of the system failed state is computed by summing the occurrence probability of each state vector resulting in the system failed state. The truth table method is difficult to apply to complex systems since the number of state increases exponentially.

6.5 Tools Overview

Software tool support is essential to perform dependability analysis and many tools exist. In the following, we present a not exhaustive list (Table 6.7), focusing on those supporting Fault Tree (FT) or Stochastic Petri Net (SPN) formalisms and their variants. Most of them enable the application of several dependability analysis techniques, besides FT and SPN.

Table 6.7 Dependability tools

Tool	ETA	FMECA	FTA	HAZOP	Markov	SPN	RBD
SHARPE			✓		✓	✓	✓
SURF-2					✓	✓	
GreatSPN						✓	
TimeNET						✓	
Möbius			✓		✓	✓	
DEEM						✓	
Galileo			✓				
StarsStudio-Astra	✓	✓	✓	✓			
FaultTree+	✓		✓		✓		

- *SHARPE* (Sahner and Trivedi 1987; Trivedi 2002)—Symbolic Hierarchical Automated Reliability and Performance Evaluator— is a toolkit for specifying and analyzing performance, reliability, availability, and performability models. It supports combinatorial modeling, with FTs, Reliability Block Diagrams and product-form queueing networks, and state-based modeling, with Markov and semi-Markov reward models and generalized SPNs. From the analysis point of view, steady-state, transient, and interval-based measures can be computed. *SHARPE* is a multi-formalism tool that enables hierarchical model composition by combining results from different kinds of models.
- *SURF-2* (Béounes et al. 1993) enables modeling with Markov chains and generalized SPNs. The tool provides support to the RAMS comparative analysis of different system architecture designs. Reward structures can be added to the

model to get QoS measures in terms of dependability, performance, and cost. SURF-2 supports both transient and steady-state analyses.

- *GreatSPN* (Baarir et al. 2009)—GRaphical Editor Analyzer for Timed and Stochastic Petri Nets—enables modeling with generalized SPNs, stochastic Well-Formed nets (SWNs), Deterministic and SPNs (DSPN), and SPNs with general (not exponentially) distributed timed transitions. It also provides support to net composition by using the package *algebra* (Bernardi et al. 2001). GreatSPN includes both transient and steady-state solvers: numerical techniques, based on the solution of the underlying Markov chain, are available for transient and steady-state analyses, while simulators can be used only for steady-state analysis.
- *TimeNET* (Zimmermann 2012) has been designed especially for models with non-Markovian SPNs. It supports Deterministic and SPNs (DSPN), Extended DSPNs (EDSPNs), colored SPNs, discrete-time SPNs, and hierarchically structured models. Transient and steady-state analyses are available for DSPNs, steady-state analysis only for EDSPNs with no concurrently enabled deterministic transitions, steady-state approximation for DSPNs with concurrently enabled deterministic transitions, transient and steady-state simulation for non-Markovian SPNs, and discrete-time analysis for DSPNs with concurrently enabled deterministic transitions. TimeNET also provides functionalities for the steady-state analysis and simulation of colored SPNs.
- *Möbius* (Clark et al. 2001)—Model-Based Environment for Validation of System Reliability, Availability, Security, and Performance—provides an infrastructure to support multiple interacting modeling formalisms and solvers. Several modeling formalisms are supported: Stochastic Activity Networks, FTs, Buckets and Balls (a generalization of Markov chains), and PEPA (a process algebra). Möbius allows combining (atomic) models to form composed models through replication and join operators and customizing measures of system properties (e.g., reliability, availability, performance, and security). Measures can be computed at specific time instants, over periods of time or when the system reaches steady state. Parametrized models can be constructed and sensitivity analysis is supported by the tool. Möbius solvers include distributed discrete-event simulators and Markov chain numerical solvers.
- *DEEM* (Bondavalli et al. 2004)—Dependability Modeling and Evaluation of Multiple Phased Systems—is a tool for dependability model-based evaluation of multiple phased systems. It enables the MPS modeling through deterministic and SPNs. DEEM solvers implement efficient analytical solution techniques based on the separability of the Markov regenerative process underlying the deterministic and SPN: both steady-state and transient analyses are supported. Dependability measures are defined through the general mechanism of marking-dependent reward functions.
- *Galileo* (Sullivan et al. 1999) is a software tool for dynamic FT modeling and analysis. Dynamic FT models can be created in either a textual or a graphical representation. Galileo includes a dynamic FT solver based on an efficient algorithm that first modularizes an FT into statistical independent sub-trees,

which are then solved using either a dynamic sub-tree solver or a static sub-tree solver (Dugan et al. 1997).

- *StarsStudio-Astra* (Contini et al. 1999) is a software package that supports several dependability modeling and analysis techniques: FTA, ETA, FMECA, and HAZOP. The tool allows to apply ET and FT techniques in a combined manner (i.e., Event tree with boundary condition technique) by linking one FT model to each ET branch. The FMECA and HAZOP analysis can be carried out by filling FMECA/HAZOP tables; besides the tables, the tool can generate histograms, which contain the number of FMECA failure modes or HAZOP items versus severity, frequency and risk classes, and risk tables, which summarize the situation of all the FMECA failure modes or HAZOP items with respect to the risk.
- *FaultTree +* (Isograph 2012) is a software package that allows dependability modeling and analysis with FTs, ETs, and Markov models. The tool computes reliability and availability metrics, minimal cut sets in FT models. It enables to define different types of distributions for failure events and includes failure data libraries (NPRD11 2011; IAEA-478 1988) for a wide variety of component types including mechanical, electromechanical, and electronic assemblies.

Model-Driven Dependability Assessment of Software
Systems

Bernardi, S.; Merseguer, J.; Petriu, D.C.

2013, XVI, 187 p. 55 illus., Hardcover

ISBN: 978-3-642-39511-6