This chapter introduces the concept of system dependability (reliability, availability, safety and maintenance) and the associated definitions (fault, error, failure). One important attribute of dependability is safety. Safety management is in general related to people and everyday life. It is a difficult and high cost activity.

1.1. Introduction

The aim of this book is to describe the general principles behind the designing of a dependable software-based package. This first chapter will concentrate on the basic concepts of system dependability as well as some basic definitions.

1.2. Dependability

1.2.1. Introduction

First of all, let us define dependability.

**Definition 1.1 (Dependability).—** Dependability can be defined as the quality of the service provided by a system so that the users of that particular service may place a justified trust in the system providing it.

In this book, definition 1.1 will be used. However, it is to be noted that there are other more technical approaches to dependability. For reference, for the IEC/CEI
1069 ([IEC 91]), dependability measures whether the system in question performs exclusively and correctly the task(s) assigned to it.

For more information on dependability and its implementation, we refer readers to [VIL 88] and [LIS 95].

Figure 1.1 shows that a system is a structured set (of computer systems, processes and usage context) which has to form an organized entity. Further on, we shall look at software implementations found in the computer/automatic part of the system.

Dependability is characterized by a number of attributes: reliability, availability, maintainability and safety, as seen in RAMS\(^1\).

New attributes are starting to play a more important role, such as security, and we now refer to RAMSS\(^2\).

---

1 Reliability, Availability, Maintainability and Safety.
2 Reliability, Availability, Maintainability, Safety and Security.
1.2.2. Obstacles to dependability

As indicated in [LAP 92], dependability in a complex system may be impacted through three different types of event (see Figure 1.2): failures, faults and errors.

The elements of the system are subject to failures, which can lead the system to situations of potential accidents.

DEFINITION 1.2 (FAILURE).– Failure (sometimes referred to as breakdown) is a disruption in a functioning entity’s ability to perform a required function. As the performance of a required function necessarily excludes certain behaviors, and as some functions may be specified in terms of behaviors to be avoided, the occurrence of a behavior to be avoided is a failure.

From definition 1.2 follows the necessity to define the notions of normal (safe) behavior and of abnormal (unsafe) behavior with a clear distinction between the two.

Figure 1.3 shows a representation of the possible states of a system (correct vs. incorrect) as well as all the possible transitions among those states. The states of the system can be classified into three families:

– correct states: there are no dangerous situations;
– safe incorrect states: a failure has been detected but the system is in a safe state;
– incorrect states: the situation is dangerous and out of control; potential accidents.
When a system reaches a state of safe emergency shutdown, there may be a complete or partial disruption of the service. This status may allow a return to the correct state after repair.

The failures can be random or systematic. Random failures are unpredictable and are the result of a number of degradations involving the material aspects of the system. Generally, random failures can be quantified due to their nature (wear-out, ageing, etc.).

Systematic failures are deterministically linked to a cause. The cause of failure can only be eliminated by a resumption of the implementation process (design, fabrication, documentation) or a resumption of the procedures. Given its nature, systematic failures are not quantifiable.

The failure is an observable external manifestation of an error (the standard CEI/IEC 61508 [IEC 08] refers to it as an anomaly).

**Definition 1.3 (Error).** Error is an internal consequence of an anomaly in the implementation of the product (a variable or a state of the flawed program).

In spite of all the precautions taken in the design of a component, this may be subject to flaws in its conception, verification, usage and maintenance in operational conditions.

**Definition 1.4 (Anomaly).** An anomaly is a non-conformity introduced in the product (e.g. an error in a code).
From the notion of an anomaly (see definition 1.4), it is possible to introduce the notion of a fault. The fault is the cause of the error (e.g. short-circuiting, electromagnetic perturbation, or fault in the design). The fault (see definition 1.5), which is the most widely acknowledged term, is the introduction of a flaw in the component.

**DEFINITION 1.5 (FAULT).--** A fault is an anomaly-generating process that can be due to human or non-human error.

---

To summarize, let us recall that trust in the dependability of a system may be compromised by the occurrence of obstacles to dependability, i.e. faults, errors and failures.

Figure 1.4 shows the fundamental chain that links these obstacles together. The occurrence of a failure may entail a fault, which in turn may bring one or more error(s). This (these) new error(s) may consequently produce a new failure.

The relationship between the obstacles (faults, errors, failures) must be seen throughout the system as an entity, as shown by the case study in Figure 1.5.
The vocabulary surrounding dependability has been precisely defined. We shall henceforth only present the concepts useful to our argument; [LAP 92] contains all the necessary definitions to fully grasp this notion.

1.2.3. Obstacles to dependability: case study

Figure 1.6 illustrates an example of how failures can occur. As previously mentioned, a failure is detected through the behavior of the system as it diverges from what has been specified. This failure occurs at the limits of the system because of a number of errors, which are internal to the system, and has consequences on the working out of the results.

In our case study, the source of errors is a fault in the embedded executable. These faults may be introduced either by the programmer (a bug) or the tools (in generating the executable, downloading tools, etc.) or they can occur because of failures of the material (memory failure, short-circuiting of a component, external perturbation (e.g. EMC\(^3\)), etc.).

---

3 EMC stands for Electromagnetic Compatibility, EMC is the branch of electrical engineering that studies unintentional generation, propagation and reception of electromagnetic energy.
in the manufacturing of the material, etc.), when installing, using and/or maintaining the software. The diagram in Figure 1.7 may then be declined for those various situations. Figure 1.7 shows an example of the impact of a human error.

**Error**

![Diagram of human error impact]

**Figure 1.7. Impact of human error**

At this point, it is interesting to note that there are two families of failures, i.e. systematic and random failures. Random failures result from the production process, age, wear-out, degradations, external phenomena, etc. Systematic failures can be reproduced as they originate in the design. Let us note that random failures may also result from a fault in the design such as underestimating the effect of temperature on the processor. As we shall see later, there are various techniques (diversity, redundancy, etc.) that can be used to detect and/or bring random failures under control. Systematic failures may pose a challenge as the issue of quality is involved and verification and validation are required.

### 1.2.4. Safety demonstration

The previous section clarified some of basic concepts (fault, error and failure). The systematic search for failures and their effects on the system is performed through activities such as preliminary hazard analysis (PHA), failure mode and effects analysis (FMEA), fault tree analysis (FTA), etc.

These types of analysis are now standard for dependability management and demonstration (see e.g. [VIL 88]) and imposed by the standards. All these analyses are used to construct a safety demonstration which is then formalized through a safety record: the safety case. The generic standard CEI/IEC 61508 [IEC 08], which
is applicable to electronics and programmable electronics systems, covers this point and proposes a general approach.

1.2.5. Summary

When designing a software package, one should bear in mind three possible types of failure:

– random failures of the material components;
– systematic failures in the design, either material or software-based;
– specification “errors” in the system; these may have serious consequences on the operation and maintenance of the system.

1.3. Conclusion

In this chapter we have presented the basic notions related to dependability through definitions and examples. In the following chapters of this book, we shall show how to decline and take into account these notions and principles in order to make the system dependable in spite of the presence of a fault.

It is to be noted that with systems that impact on safety, standards are to be applied that encompass aspects of bringing failure safety and management under control.

1.4. Bibliography


[IEC 91] IEC, 1991

