AS

vital to
omobiles, to
engineers are

r any
knowledge of

ition, through

ne the

is of computer-

ng the nuclear,

engineers
consider

published a
as many years

75

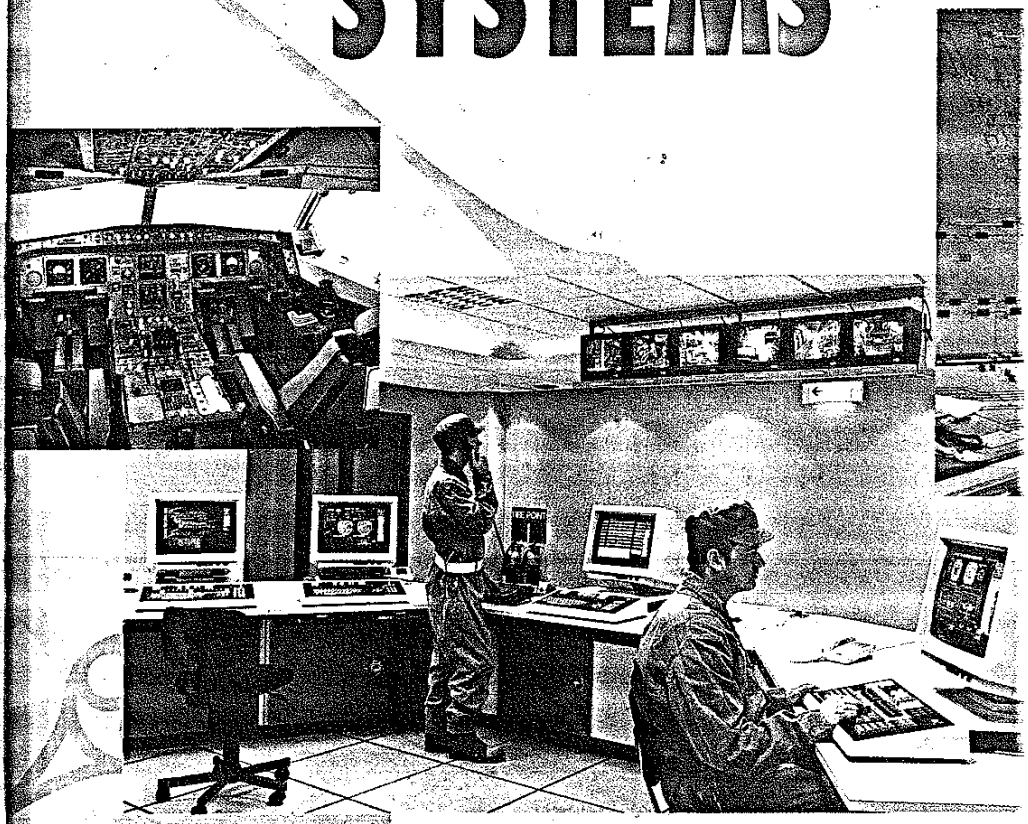
SAFETY-CRITICAL COMPUTER SYSTEMS

Storey

ADDISON WESLEY

42787

SAFETY-CRITICAL COMPUTER SYSTEMS



Neil Storey

ADDISON-WESLEY

Safety-Critical Computer Systems

1 Introduction

CHAPTER CONTENTS

1.1 Computers in critical applications

Since the invention of the microprocessor in the early 1970s the cost of computer-based technology has fallen steadily. This has led to a dramatic increase in the use of computers, to a point where they now outnumber the humans on our planet. The majority of microprocessors are not used in desktop computers or other applications conforming to our traditional image of a computer. Instead, they are in the form of 'embedded systems', where the presence of the processor is largely invisible to the outside world. The range of such applications covers a vast spectrum, from sophisticated aircraft flight control systems to washing machines, and from nuclear shutdown systems to antilock brakes. Automotive and domestic products represent the highest-volume areas, with production being measured in millions. At the other extreme are dedicated industrial control systems that may be unique.

A major distinction between embedded systems and the applications normally associated with general-purpose machines relates to the likely consequences of incorrect operation. Although it may be extremely annoying if a word processor crashes, it is unlikely that the operator will suffer any direct harm as a result. However, the failure of many embedded systems can result in direct, and possibly very serious, harm to one or more people. In some instances the importance of correct operation is clear: failure of the control system of a nuclear reactor could endanger the lives of thousands, if not millions of people, and could have implications for the global environment. On a smaller scale, failure of a critical avionics system could potentially cause an air crash killing hundreds of people. Similarly, failure of an automotive engine management or braking system could risk the lives of several people. In these

cases the likely consequences of failure are obvious and the industries concerned have, over many years, developed methods of dealing with the risks involved. Many of these techniques form the basis of the later chapters of this book. Unfortunately, in many cases the consequences of failure of an embedded system are less clear, leading to an assumption that its operation has no implications for the safety of its users or the public. Many domestic appliances come into this category. In practice, any system that controls even modest amounts of power has the capacity to do harm. If, for example, the controller of a washing machine or a pop-up toaster fails in such a way that the heater is left on, this could result in a fire that could kill. Although such an event might seem unlikely, experience shows that, because of the large numbers involved, domestic appliances are a significant cause of personal injury.

From the above discussion it is clear that the operation of many embedded computer systems has a direct influence on the safety of their users and the public. Such systems are often referred to as **safety-related systems**, or sometimes as **safety-critical systems**.

1.2 Safety

Before looking in detail at the use of computers in safety-related systems it is important to discuss the more general topic of **safety**. Definitions of safety vary considerably. A sufficient definition for our needs is:

> *Safety is a property of a system that it will not endanger human life or the environment.*

From this definition we can more completely define the term safety-related system as follows:

> *A **safety-related system** is one by which the safety of equipment or plant is assured.*

This definition covers a wide variety of equipment, from a microswitch that ensures that a guard is closed before a machine may be operated, to a nuclear shutdown system. It also encompasses systems whose primary role is to ensure safety, and equipment that must provide safety while carrying out some other function. An example of the latter might be an aircraft autopilot.

The term **safety-critical system** is normally used as a synonym for a safety-related system, although in some cases it may suggest a system of high criticality.

Having defined terms relating to safety and safety systems, it is important to note that no system can be absolutely safe. Our goal in designing a system is to make it adequately safe for its given role. Unfortunately, determining an adequate level of safety inevitably involves personal judgement and opinion. This is complicated by the fact that safety is an emotive subject and people's perceptions of it are both varied and illogical. It is often observed, for example,

that many people are afraid of flying, despite the fact that statistically they are safer in the air than when driving to the airport.

Although decisions on the appropriate level of safety for a given application require judgement, there are several techniques of analysis that can be used to assist in these decisions. The implications of failure vary greatly between applications, and this leads to the concept of **levels of integrity** that reflect the importance of correct operation. Once a project has been assigned a **safety integrity level**, this will determine the methods of design and implementation used for the system. We shall return to look at the meaning of 'integrity' in Chapter 2, and will discuss the various integrity levels in Chapter 4.

As many embedded computer systems are safety related it is prudent to begin *all* computer-based projects by investigating their safety implications. In the UK, Ministry of Defence standards require such an analysis to be carried out on all military software (MoD, 1991), and there is good reason to follow this practice for computer systems as a whole, for civil as well as military applications. If a system is found to have safety implications it must be allocated an integrity level reflecting its level of criticality. This will then determine the methods used for the design, construction and testing of the unit. The techniques used to investigate the safety implications of a system come under the headings of hazard analysis and risk analysis. The former looks at the identification of situations that could endanger human life or the environment, and the latter considers the risks associated with these events. These topics are discussed in Chapters 3 and 4 respectively.

The scope of safety

Considerations of safety have implications for the complete system, and for all stages of its life, from inception to decommissioning.

Safety can only be assured by considering all aspects of a system, including both hardware and software. We shall see in later chapters that software often plays a pivotal role in ensuring system safety. However, software alone cannot provide such assurance, as its correct operation is dependent on the system hardware. Similarly, in many cases hardware alone cannot satisfy requirements of safety, and an integrated approach to safety is essential. Consideration must be given to all aspects of the system, including sensors, actuators, cabling, connectors, communications links and power supplies. Human operators or users should also be considered as part of the overall system and their possible actions examined.

Safety cannot be produced simply by good design. The behaviour of any system may be upset by mistakes made during its production, installation or use. Failure to fit a component correctly or the use of an incorrect part may invalidate protective features within the design, and so compromise its safety. Similarly, installing or using a system inappropriately may result in a dangerous situation that is beyond the scope of the original design. Safety is very closely linked with **quality management**, and this very important topic will be discussed

in Chapter 13. Safety issues continue throughout the working life of a system, and may include considerations of its eventual decommissioning. This has particular relevance in the nuclear industry.

Because safety considerations affect all the stages of a system's life, they also affect everyone who is connected with it, either directly or indirectly. This relationship is illustrated in Figure 1.1. Those affecting the safety of the system include the customer, the designers and those responsible for production and installation. Maintenance staff, and the operator or user of the system, may both affect and be affected by the safety of the system. Also to be considered are the general public, who might suffer the effects of any failure of the system. In extreme cases, as we have seen in the case of nuclear accidents, this group can potentially include the population of the entire world.

The safety aspects of a computer system may be divided into a number of categories. The first of these represents what might be termed the **primary safety** of the system itself. This includes dangers from electrocution or electric shock, and from burns or fire caused directly by the computer's hardware. The second class represents the **functional safety** of the system. This covers aspects concerned with equipment that is directly controlled by the computer, and is related to the correct functioning of the computer hardware and its software.

A third category represents what might be termed **indirect safety**, as in the case of **safety-related information systems**, and relates to the indirect consequences of a computer failure or the production of incorrect information. These considerations are of relevance in a wide range of systems, such as medical imaging and patient records systems. A well-known example illustrating such matters was the implementation of the London Ambulance Service

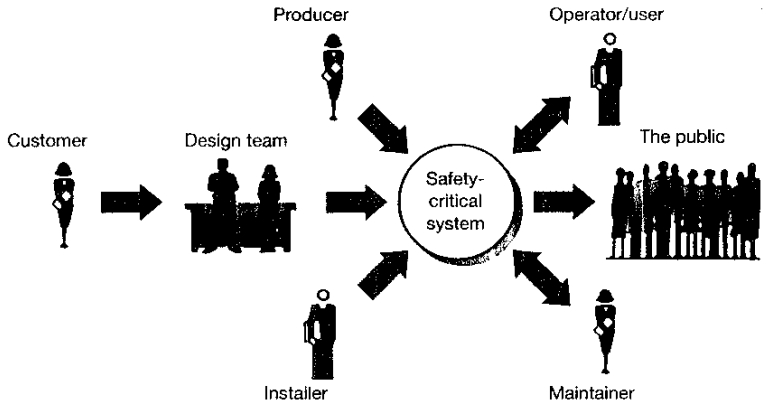


Figure 1.1 A safety-critical system.

dispatcher. When first installed, this system, which routes 999 phone calls to the appropriate emergency services, caused very long delays in dealing with emergencies. The safety implications of such delays are very clear. Other examples from this category relate to automated tools that are used in the design of safety-related equipment. Faults within such tools could lead to incorrect designs, that could in turn endanger life. From this example it is clear that the use of computer packages such as databases, finite element analysis utilities and even word processors could be considered to have safety implications. The hardware of the computers used to run these packages must also be considered, as a system fault could also lead to incorrect data.

Whereas safety is all-encompassing this text cannot be, and is restricted to considerations of functional safety. We shall therefore consider only those aspects of safety related to the direct effects of computer-controlled equipment. This restriction is in line with the definitions of safety and safety-related system given earlier in this section.

Forms of safety-related system

In most cases the safety-related systems that fall within the scope of this book may be categorized as being either control systems or protection systems.

Control systems are used to determine the operation of some form of equipment or plant. In some cases these control functions have no safety implications. This might be because the system being controlled is not capable of dangerous actions, or because safety is managed by some other system or subsystem. Alternatively, the control system may perform a dual role of controlling the equipment and providing safety functions.

Protection systems use sensors to detect fault conditions and produce outputs to mitigate their effects. In many cases the action of the protection system is to shut down the equipment concerned. Such systems are often called **shutdown systems**.

In the case of both control systems and protection systems the overall system configuration is similar, as shown in Figure 1.2. The equipment or system with which the application is concerned is often termed the **equipment under control** or **EUC**. This will have inputs from, and outputs to, the environment through which it performs its designed function. The EUC could be a complete process or production facility, such as a power station or chemical plant, or it could be a relatively small piece of equipment, such as a domestic appliance or an automotive component. In some industries the EUC is normally referred to as the **plant**. The control or protection system interacts with the EUC through **sensors** and **actuators** (or **effectors**) that are used to monitor and control certain parameters. The operation of the control or protection system is determined by the functions or algorithms implemented within it. Control systems are not always safety related but protection systems, by their very nature, usually are.

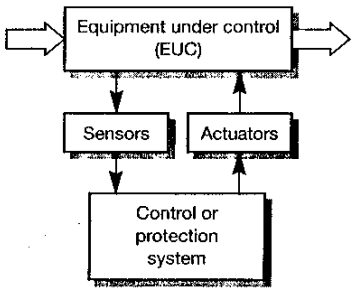


Figure 1.2 A typical control or protection system configuration.

Computers in safety-related systems

Within the field of industrial control it is common to use the term **programmable electronic system (PES)** to represent all forms of computer-based equipment. This term includes not only conventional computers and microcontrollers, but also other software-controlled units such as **programmable logic controllers (PLCs)**.

Clearly, not all control or protection systems are computer based. Indeed, we shall see as we discuss the various topics within this book that there are many cases where non-programmable methods have considerable advantages. If, for example, the temperature of a liquid can be kept to safe levels using a thermostat, this is preferable to using a complex and hence expensive microcomputer system to achieve the same end. However, there are many cases where a programmable electronic system is advantageous, or even essential, to achieving the control and safety features required.

When a computer-based system is adopted, the control or protection system will have two major components: hardware and software. Both are essential, and both directly affect the safety of the system. This arrangement is shown in Figure 1.3.

Advantages of computer-based systems

The most obvious advantage of computer-based systems is their processing power, which allows them to perform complex control functions that might be impossible by other means. They can take inputs from many sources, drive many outputs and perform complex calculations. Their operation is also characterized by high speed, low power consumption and a small physical size.

Modern digital devices are extremely reliable. Most systems based on such devices also possess a high degree of integration, resulting in a small number of components with relatively few interconnections. Such systems

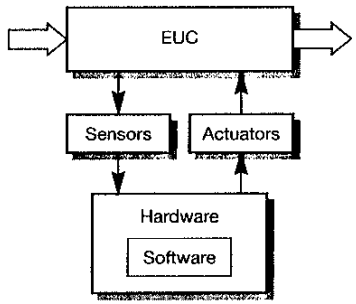


Figure 1.3 A computer-based control or protection system.

therefore tend to be highly reliable, both because of the nature of their components and because of the small number of interconnections, which are often a major cause of failure.

Another important property of computer-based systems is their flexibility: the system's characteristics may be drastically changed by modifications to the software without the need for hardware alteration. This often provides an efficient method of system upgrading at a relatively low cost.

In most cases the cost of implementing a programmable electronic control system is less than that of alternative methods. This is particularly true as the complexity of the required system increases. The additional benefit of software upgradability further increases the cost advantage. In low-volume projects development costs predominate, with software being responsible for the majority.

The availability of considerable processing power allows more sophisticated safety strategies to be implemented. These might include the use of self-diagnostic routines, condition monitoring, range checking and the implementation of interlocks.

Disadvantages of computer-based systems

The primary disadvantages of programmable electronic systems in safety-related applications stem directly from their inherent **complexity**.

By their very nature all computer-based systems are complex. Even minimal microprocessor arrangements have at their heart a component with hundreds of thousands of components and an extremely complex behaviour. Software too is associated with great complexity: even relatively simple programs may represent many thousands of possible execution paths.

Complexity is a major obstacle to safety. Complex systems are more difficult to design and are therefore more likely to contain design errors. They are more difficult to test and are thus more likely to contain undetected faults.

They are also more difficult to understand, and may therefore be more susceptible to human errors in installation or use.

One of the desirable properties of any safety-related system is **predictability**, and with simple components this is generally achievable. A simple switch, for example, can be designed so that it will normally 'fail safe', which, depending on the application, may mean to fail 'open' or to fail 'closed'. In any event it is likely that it will fail in one of these two modes, which are termed its possible **failure modes**. A simple arrangement of a small number of switches could conceivably fail in a number of ways, the number being determined by the combination of all the possible failure modes of the individual parts. By careful planning it may be possible to arrange a series of tests that would detect any of these patterns of failure, so providing an exhaustive test of the unit. It might also be possible to provide an alternative unit, so that if any of these failures were detected a backup could be provided.

In complex digital devices such as microprocessors the number of possible failure modes is so large that it may be considered to be infinite. In such cases it becomes impossible to devise exhaustive tests for such a device, and therefore the detection of failure will always be unreliable.

The problems associated with complexity are equally applicable to software. All but the simplest of programs are too complex to test exhaustively in any true sense. As in many safety-related systems much of the complexity is implemented within software, testing represents a serious problem.

1.3 Developing safety-related systems

Having identified not only advantages, but also disadvantages to the use of computers within safety-related systems, it is clear that a programmable solution will not always be ideal for a given application. However, in many cases the advantages outweigh the problems and a computer-based approach is adopted. In certain circumstances a computer-based system is the only viable method of producing the required functions.

The process of developing a safety-related computer system may be both involved and time consuming. Like all development projects it has various phases, and again as with all projects, these may be represented diagrammatically using a **lifecycle model**.

Various lifecycle models are used, each accentuating some aspect of the project. Thus some are more suitable for corporate planning, others for resource management or costing. Each technique has its advocates and its critics, and it is not within the scope of this text to discuss their merits. Figure 1.4 shows a typical development lifecycle model which has its origins in that given in the STARTS guide (1989). For obvious reasons this is often referred to as the 'V' lifecycle model. The model identifies the major elements of the development process and indicates the sequential nature of much of the

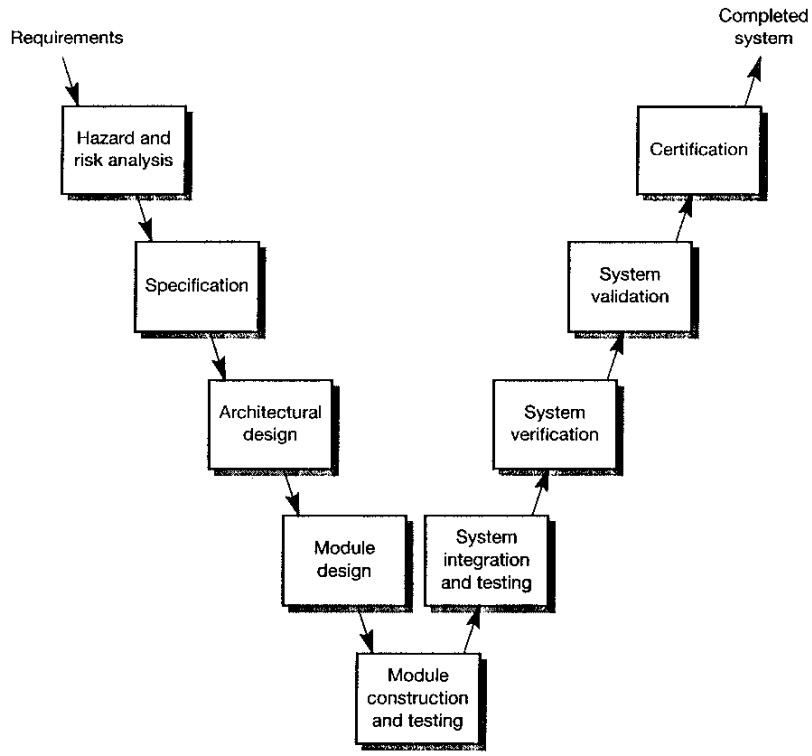


Figure 1.4 A typical development lifecycle model.

work. One of the attractions of this model is that its form emphasizes a 'top-down' approach to design (the left-hand arm of the diagram) and a 'bottom-up' approach to testing, as shown on the right-hand arm. Such models are only an approximation to the development process: in practice the various stages are not always performed in such a strictly sequential manner. Design often involves a great deal of iteration, with a series of operations being performed repeatedly until a satisfactory result is obtained. A degree of parallelism is also possible. Also missing from this simple model is any indication of information flow between the various stages. In practice, data from the specification stages of the work forms an input to the testing and verification stages of the project. We shall return to look at the use of lifecycle models in more detail in Chapter 5.

The starting point of any development project is determined by the **system requirements**. Generally, the term 'requirements' is taken to represent an

almost abstract definition of what the system should do. Before the system can be implemented these abstract requirements must be formalized into a **functional requirements document**. This process is often referred to as the 'requirements capture' phase of the project. Using these definitions it is clear that the requirements documents are an attempt to describe what the system should do, and it is conceivable that they contain mistakes or are incomplete. The implications of limitations in the requirements documents may be considerable, as much of the remainder of the development programme is aimed at implementing the system described within these documents. For this reason, great care is taken in their preparation.

Once the functional requirements of the system have been established, hazard and risk analyses are performed to identify potential dangers in the system and to allocate an overall level of integrity. One of the outputs from these analyses is a statement of the **safety requirements** of the system. This defines what the system must and must not do, in order to ensure safety. The safety requirements then sit alongside the functional requirements in determining what the system must achieve.

From the functional and safety requirements of the system a specification is produced, which will include measures for safety assurance in line with the integrity level assigned. The specification attempts to define a system that will completely fulfil these requirements, but in reality it is possible for mistakes to be made at this stage. Requirements are often written in natural languages, such as English, which are subject to ambiguity. A misunderstanding of some aspect of the requirements may lead to a specification that is incomplete or incorrect. Errors of specification are a major problem in the production of highly critical systems, as much of the testing performed is aimed at establishing that the system meets this specification.

Once a specification has been produced, this is used as the basis for the top-level design that defines the system architecture. One of the major aspects of this process is to partition the system into hardware and software. This hardware–software trade-off is a vital part of the design phase and must take into account many diverse considerations, such as the volume of production and the operating speed. Safety requirements also play a large part in this process. The top-level design will inevitably split the project into a number of more manageable modules to simplify the design and testing processes. Specifications will then be produced for each module, these specifications being used later for module testing. Top-level design is followed by the detailed design of both the hardware and the software of each of the modules. When this design stage is complete the modules will be constructed and tested individually. This testing forms part of the process of verification which is used to establish that each module satisfies its specification. Verification continues throughout the development lifecycle and forms an important aspect of each phase. Verification techniques will be discussed in some detail in later chapters.

Once the various modules have been completed and verified, the process of system integration may begin. Methods of system integration vary considerably. In some companies integration is performed progressively, with more and more modules being combined and tested, producing an overall system of increasing complexity. In others a 'big-bang' approach is adopted where the complete system is assembled and tested as a whole. Both techniques have their positive and negative aspects, and these will be discussed in more detail in Chapter 5.

Once the system is complete and appears to be functioning correctly, the verification and validation of the entire system may begin. Note that these two terms are subject to very varied definition and many engineers seem unsure as to the distinction between them. Possibly because of this, the two functions are often grouped together as '**verification and validation**' or simply '**V & V**'. Within this text we shall adopt the most common distinction between the two terms, which is that

> ***Verification*** *is the process of determining that a system, or module, meets its specification*

and

> ***Validation*** *is the process of determining that a system is appropriate for its purpose.*

It can be seen from these definitions that verification seeks to show that the system corresponds to its specification. This is often performed at the module level initially, to show that each module performs its specified function. Validation, by contrast, sets out to determine whether the system as a whole accurately meets the requirements of the user. It therefore includes considerations of the correctness of the specification itself. We shall look at verification and validation in more detail in Chapter 12.

In many highly critical projects the final stage is to convince some external regulating body that the system is safe, and thereby to achieve **certification**. The process of certification varies considerably between one industry and another, although the general principles are common. In most cases standards and guidelines set out the design and development techniques that should be used in order to achieve certification. The nature of these guidelines, and the process of certification, are discussed in Chapter 14.

The lifecycle model of Figure 1.4 may be applied to the development of any system. In projects that are not safety related the hazard analysis phase may be performed only cursorily, although in order to establish that the project has no safety implications some form of analysis must be performed. For any system that is safety related a more detailed hazard and risk analysis phase is required in order to determine an appropriate integrity level for the project. This will then help to determine both the system design and the development methods adopted for the remainder of the project. For systems with few or no safety functions, formal certification is not normally required and the project will usually end with customer acceptance testing. Highly critical systems will generally go through a formal certification stage, determined by the standards used within that industry.

Faults, errors and system failures

In order to progress to looking at the production of safety-critical systems in more detail we need to define a few more terms. These are faults, errors and system failures, and are defined as follows:

> *A **fault** is a defect within the system.*
>
> *An **error** is a deviation from the required operation of the system or subsystem.*
>
> *A **system failure** occurs when the system fails to perform its required function.*

System **faults** may take a number of forms. For example, the failure of a hardware component represents a fault, as does a mistake within a piece of software. A mistake in the design of the system also represents a fault. The presence of a fault *may* lead to an **error**, which is the mechanism by which the fault becomes apparent. If, for example, a memory device within a computer were to fail the failure could go undetected for some time if that particular device were not accessed. The system would then possess a fault, but this would not have manifested itself by producing an error. Only when the device was used would its faulty operation cause an error, which could then affect the functioning of the system. Similarly, a software fault will only cause problems when the affected part of the program is executed. A software 'bug' may lie dormant within a rarely used subroutine for many years until a particular sequence of events satisfies the conditions required for its execution. During that period the system would have contained the software fault, but this would have had no effect on the system. However, when the code is finally executed, the fault *may* result in an error that could affect the system's operation. Such an error would take the form of a deviation from the intended operation of the system, for example the production of an incorrect value during a calculation, or the execution of an inappropriate routine.

If the presence of an error were to cause the system as a whole to deviate from its required operation, this would be termed a **system failure**. However, if the system were appropriately designed, it might be possible for it to continue to perform its required functions, even in the presence of one or more errors. One of the goals of a safety-critical system is that errors should not result in system failure. This could be achieved by interconnecting several modules in such a way that an error within one did not result in failure of the entire system.

It is useful to divide faults into two distinct classes. The first of these, termed **random faults**, are associated with hardware component failures. All physical components are subject to failure, and thus all systems are subject to random faults. When working within their correct operating environment, individual components fail randomly. However, statistical data gathered on large numbers of similar devices may enable predictions to be made concerning the probability of a component failing within a given period of time. This in turn allows the overall performance of the system to be predicted. Estimation of reliability on the basis of statistical data on individual components is discussed in Chapter 7.

The second class of faults are termed **systematic faults**. This covers many forms, including design faults and mistakes within the specification of the system. All software faults come within this category as they are faults within the design of the software. Systematic faults are not random and are thus not usually susceptible to statistical analysis. It is therefore more difficult to predict their effect on the reliability of the system.

Faults may also be categorized in other ways, for example by their duration. Some faults are permanent whereas others are transient in nature. All design faults and most random component failures are permanent, although their effects may not be visible at all times. However, some hardware faults are transient, such as the effects of a noise spike, or an alpha particle hitting a memory device. Although the fault is transient in these cases, the errors they produce may remain in the system unless some action is taken to remove them. Further discussion of the nature of faults is given in Chapter 6.

A fault-free system would operate perfectly, as it would be free from all design and component faults. There would be no errors and therefore no system failures. Unfortunately, the complete elimination of faults is impossible, first because all components are subject to random failure due to wear, ageing or other effects, and secondly because it is not possible to achieve perfection in design. Therefore faults are inevitable.

From the above discussion it is clear that much of the process of producing safety-critical systems is concerned with 'fault management'. In non-critical systems good design and the use of good-quality components can often produce acceptable performance. In highly critical systems additional measures must be taken to overcome the effects of faults. Broadly, these measures may be divided into four groups of techniques:

- fault avoidance
- fault removal
- fault detection
- fault tolerance.

Fault avoidance techniques aim to prevent faults from entering the system during the design stage. Fault avoidance is the primary aim of the entire design process, and various aspects of this topic are covered throughout the text. Of specific interest is the use of formal methods, as discussed in Chapter 11.

Fault removal methods attempt to find faults within a system before it enters service. These include both hardware and software testing techniques, which are discussed in Chapter 12.

Fault detection techniques are used during service to detect faults within the operational system so that their effects may be minimized. Fault detection is discussed in Chapter 6.

Fault tolerance techniques are designed to allow the system to operate correctly in the presence of faults. These techniques are described in Chapter 6.

Unfortunately, none of these methods is completely effective, and in highly critical applications a combination of these techniques is used so that system failures are kept to an acceptable level.

1.4 Costs and benefits

Safety is expensive. The goal of any project is therefore to achieve an appropriate compromise in terms of cost and safety. Experience shows that the cost of employing a systematic approach to achieving adequate safety is invariably far less than that incurred in the rectification of problems following an accident (HSE, 1993).

The costs of increasing the safety of a system must be balanced against the additional expenditure that will be required. Inevitably this will involve making value judgements concerning the value of human life and suffering. It is tempting to say that human life is beyond price and that no expenditure is too great if it might save a single life. However, such arguments fly in the face of logic.

It is well known that we could make vehicles safer by reducing their speed. It is estimated that a reduction in road speed of one mile per hour would reduce accidents by 5%. If we halved the maximum speed allowed on our roads we would cut fatalities considerably. This being the case, would we be prepared to accepted increased journey times and greater congestion on our roads in order to save lives? Should we return to requiring a man with a red flag to walk in front of all cars? Ultimately we have to make a value judgement as to what level of safety is acceptable. Similar decisions have to be made on financial grounds. We could no doubt increase the safety of buses and trains by increasing spending on safety features. We could, for example, insist on seatbelts and safety airbags for all passengers. If this move doubled the cost of the vehicles so that travelling costs increased (perhaps encouraging people to use less safe methods of transport) would this be justified? Again, value judgements are required.

Ultimately we have to put a value on human life, in both financial and other terms. We are clearly prepared to put people's lives at risk in order to bring benefits to the population as a whole. If this were not the case, then no large-scale civil engineering projects would be possible. Planning in many industrial sectors is done on the basis of assigning an actual monetary value to each life lost. Given that this is true, where do we draw the line? How much risk is acceptable and what benefits are necessary to justify them?

One of the considerations involved in the assessment of acceptable levels of risk is linked to the questions 'Who gains from the system' and 'Who is put at risk by the system'? If the person at risk stands to gain some form of benefit, then this may be a more acceptable situation than if a system benefits one

individual but places others at risk. For example, it might be acceptable for a person to race a car on a racetrack and risk his life for the sake of the excitement involved, but it is less likely to be acceptable for him to race on the open road where others are put at risk.

There are no simple answers to questions concerning the acceptability of risk: ultimately it is matter of personal judgement. However, in order to assess the issues it is useful to be able to quantify the risks and benefits involved. Such an assessment makes use of hazard analysis and risk analysis, as discussed in Chapters 3 and 4.

Legal aspects

Having considered some of the social and moral implications of safety it is also necessary to look at another major consideration, that of legal liability.

The manufacturer of a product or system that causes harm to individuals or to the environment may be liable to either criminal or civil prosecution, or both (Davis, 1994; Wichmann, 1992). Criminal liabilities arise from legislation on both **safety at work** and **consumer protection**. Civil liabilities are centred on contract law and **sale of goods** legislation. In both cases the various laws vary considerably between countries, with major differences between, for example, Europe and North America. Successful prosecutions could result in fines or the imprisonment of the employees responsible.

Although it is not within the scope of this book to give any form of legal advice, it is perhaps worth making a few observations on the problems involved. First, we have seen that there is no such thing as absolute safety. Various techniques are available to help us to increase our confidence in the safety of a system, but no manufacturer can be absolutely sure of the safety of his products. Secondly, if a manufacturer is found to be responsible for human casualties, the financial implications may be enormous. This leaves the manufacturer in a very difficult position in knowing how to fulfil his moral, ethical, legal and commercial responsibilities.

Faced with these problems there are several steps that a manager may take to protect himself and his organization. Obvious measures include providing insurance against civil claims, although it is becoming extremely expensive to provide adequate cover against public liability. The use of proper labelling and warning signs may also help to divert liability away from the manufacturer towards the user of a system. However, the greatest responsibility of the manufacturer is that he should take adequate care in the design and manufacture of his products.

As it is unreasonable to expect any system to be perfect, a manufacturer may be able to defend the failure of one of his products by establishing that the system was as safe as could reasonably be expected, given the '**state of the art**'. In order to establish this, it will be necessary to demonstrate that appropriate design and development techniques were followed. Given the complexities of the processes involved in developing safety-critical systems it seems likely that a

good way to establish that one has followed such practice is to show that the development was performed in line with appropriate guidelines and standards. Where such guidance exists within an industry, non-conformance with these recommendations might make justification of the correctness of the development extremely difficult.

The designer's obligations

Designers of safety-critical systems share, with other members of the company, a responsibility for the systems they produce. These responsibilities include professional, social, moral, financial and legal considerations. In order to discharge these obligations designers must be aware of the implications of the work they do, and ensure that others within their organizations are similarly informed. It is also their responsibility to ensure that they are familiar with the various techniques and development tools that may assist them in their work. It is every designer's duty to make all systems as safe as is reasonably possible.

The material within this text is not sufficient to enable an engineer competently to design highly critical systems. It does, however, provide grounding in the techniques used and may at least enable engineers to establish whether their work requires additional attention to safety. It may also encourage some to go on to specialize in this interesting and highly rewarding field of engineering.

REFERENCES

Davis D. (1994). Safety critical systems – legal liabilities. *Computing and Control*, 5(1), 13–17

HSE (1993). *The Costs of Accidents at Work*. HS(G)96. London: Health and Safety Executive

MoD (1991). Interim Defence Standard 00-55 *The Procurement of Safety Critical Software in Defence Equipment*. Glasgow: Directorate of Standardization

STARTS Purchasers' Group (1989). *The STARTS Purchasers' Handbook: Software Tools for Application to Large Real Time Systems* 2nd edn. Manchester: National Computing Centre Publications

Wichmann B.A., ed. (1992). Legal liability for software in safety-related systems. In *Software in Safety-Related Systems*. (Wichmann B.A., ed.), pp. 27–33. Chichester: John Wiley

FURTHER READING

Bennett P., ed. (1993). *Safety Aspects of Computer Control*. Oxford: Butterworth-Heinemann

Hazards Forum (1995). *Safety-Related Systems: Guidance for Engineers*. London: The Hazards Forum

Redmill F. and Anderson T., eds (1993). *Safety Critical Systems: Current Issues, Techniques and Standards*. London: Chapman & Hall

PROBLEMS

1.1 Which industrial sectors are responsible for the highest-volume use of microcomputers? Give examples of some very high-volume products.

1.2 Which industrial sectors manufacture computer-based systems in very low volumes? Give examples of such systems.

1.3 Under what circumstances are the operations of a computer system likely to be safety related?

1.4 Give examples of safety-related computer systems within five distinct industrial sectors.

1.5 List four examples of computer-based domestic appliances where the controller is safety related.

1.6 Define the term 'safety'.

1.7 Explain the meanings of the terms 'safety-related system' and 'safety-critical system'.

1.8 How may a system be made absolutely safe?

1.9 Why is it appropriate to assign integrity levels to projects? Give examples of applications that might be assigned a high level of integrity, and others that might be assigned a lower level of integrity.

1.10 Explain the functions of hazard analysis and risk analysis.

1.11 What factors would be considered in the assignment of an integrity level for an application?

1.12 Distinguish between the terms 'primary safety', 'functional safety' and 'indirect safety'.

1.13 Explain the meanings of the terms 'control system' and 'protection system'. Which of these systems is safety related?

1.14 What is meant by the initials PES? What systems are encompassed by this expression?

1.15 List the advantages and disadvantages of computer-based systems.

1.16 Why is complexity a barrier to safety?

1.17 Why are computer systems less predictable than simpler systems?

1.18 Sketch a simple lifecycle model to describe the development of a safety-critical system.

1.19 Describe the meaning and importance of 'requirements capture'. What is the output of this process?

1.20 Suggest a possible cause of errors in the specification of a system. Why are such errors important?

1.21 How are the individual modules of a system integrated during the development process?

1.22 Define the terms 'verification' and 'validation'. How do the processes of verification and validation differ in terms of their scope and objectives?

1.23 What process generally forms the final phase of system development in highly critical systems?

1.24 Which aspects of the development lifecycle model of Figure 1.4 are common to all systems and which are unique to safety-critical systems?

1.25 Define the terms 'fault', 'error' and 'system failure'.

1.26 Explain the distinction between 'random' and 'systematic' faults and give examples of each.
1.27 Explain how faults may be classified by their 'duration', giving examples of each class.
1.28 Why are value judgements inevitable in the assessment of safety requirements?
1.29 Why are there legal implications to the design of safety-critical systems, and how may a company protect itself from prosecution?
1.30 Why are standards and guidelines of relevance to the legal liability of a company?

2 Safety Criteria

CHAPTER CONTENTS

2.1 Introduction

All engineering projects begin with a perceived need. This takes the form of a set of **requirements** for a system to perform a given task. Often these are termed the **customer requirements** for the project. In order to develop a system to meet these requirements they must first be documented by the preparation of a **requirements document**, which attempts to set out the needs of the customer in an unambiguous manner. Once completed this document forms a basis for the remaining stages of the design process.

The requirements for a given system will clearly include the functions to be performed by that system and will vary considerably from one application to another. For example, the requirements for an autopilot might include a need to measure accelerations and to compute relative positions, whereas those for a washing machine controller would be quite different. In addition to these **functional requirements** there are several more general characteristics which will be of importance in the final system. These **non-functional requirements** include such considerations as ease of maintenance, size and cost.

In safety-related systems, safety issues also form part of the requirements. Because of the importance of these safety considerations it is common to produce a separate **safety requirements document**, which sets out what is required of the system to ensure adequate safety. This document details what the system must do, and what it must not do, to achieve safety, and also lists other issues of importance. For example, in a laser control system it might define that a safety guard must be closed if the laser is active, and that the laser must not be activated while a door is open. The safety requirements document will also specify more general system characteristics that have implications for the safety of the system. These include considerations such as its reliability and

its failure modes. Other requirements might relate to the positioning of the unit, the physical protection required or necessary administrative procedures.

The various functional, non-functional and safety requirements of the system result in a list of system-specific properties, together with a set of more general system characteristics. This latter category might include considerations of:

- reliability
- availability
- failsafe operation
- system integrity
- data integrity
- system recovery
- maintainability
- dependability.

In the next two sections we look first at various system characteristics that have particular importance in safety-related systems, and then at the form of safety requirements. The final section considers the safety case that is used to document the safety aspects of a system for external certification or assessment.

2.2 System requirements

There are many characteristics that are advantageous in almost all engineering systems. These include low cost, high reliability and ease of maintenance. However, the relative importance of these attributes, and the necessity for other characteristics, varies greatly between applications. In this section we look at a number of factors that are of particular relevance to safety-related systems, and discuss their importance in a range of applications. The various topics may be seen as 'design aims' for a particular system.

Reliability

As with almost all the terms discussed in this section, the term reliability may be defined in a number of ways. Here we will adopt the following:

> ***Reliability** is the probability of a component, or system, functioning correctly over a given period of time under a given set of operating conditions.*

Here, 'functioning correctly' is taken to mean 'operating as defined within its specification', and it is assumed that the unit was functioning correctly at the

beginning of the period in question and that no maintenance is carried out during this period.

It can be seen from the above definition that the reliability of a component or system varies with time. A system might have a certain probability of working correctly over a period of one month, and a much lower probability of operating correctly for a year. When assessing reliability it is therefore very important to consider the performance of the system over an appropriate length of service. The period used will vary considerably, depending on the application. In situations where maintenance is expensive or impossible, a unit might be required to operate without maintenance for several decades. Examples might include communications satellites, robot spacecraft or cardiac pacemakers. In other situations the period of use might be much shorter. In military applications the period of interest might be limited to the duration of a particular 'mission' and might be as short as a few hours, or even a few minutes.

Reliability is of particular relevance in applications where continuous, uninterrupted operation is essential to the maintenance of safety. An example of such a system is a flight-critical aircraft system, where the safety of the aircraft is dependent on the system functioning correctly throughout its flight. However, reliability is an issue of great importance in all engineering systems, and we shall return to discuss this topic in more detail in Chapter 7.

Availability

This term may be defined as follows:

> *The **availability** of a system is the probability that the system will be functioning correctly at any given time.*

Like reliability, the availability of a system varies with time. However, unlike reliability, the availability of a system relates to a particular point in time, rather than to a given period.

Another way of looking at availability is that it represents the fraction of time for which a system is functioning correctly. For example, if within a period of 1000 hours a system is out of operation for a total of one hour owing to several periods of failure, its average availability during the period is 999/1000 or 0.999.

High availability is a very common design goal in many systems that are not directly safety related. Time-sharing computers, banking systems and telephone exchanges are all examples of systems where revenue is lost while the system is inoperative. In such cases availability is often more important than reliability, as a system failure may not itself be a problem provided that the system can be repaired quickly. Telephone exchanges are often specified to have a total 'downtime' of a few hours throughout their lifetime of several decades.

Availability may also be of great importance in safety-critical systems. This is often the case in situations where the system in question is not used

continuously. A typical example might be a nuclear reactor shutdown system. This is employed relatively infrequently, and the safety of the plant is affected by the probability that it will work correctly when needed. In this case availability is of greater importance than reliability.

For safety-critical systems the requirements for availability are high and the numerical values are consequently very close to unity. For this reason it is common to describe such systems in terms of their **unavailability**, this being obtained by subtracting the availability from one. Thus a system with an availability of 0.999 has an unavailability of 0.001, or 10^{-3}.

Clearly, availability is closely linked to issues of reliability but the two terms are not synonymous. We will look in more detail at factors determining reliability and availability in Chapter 7.

Failsafe operation

Some systems have the highly desirable characteristic of possessing a set of output states that can be identified as being 'safe'. In such circumstances the system can be designed to 'fail safe' by ensuring that it adopts these outputs in the event of failure and inability to recover. An example of a failsafe arrangement might be a railway signalling system. Here the failsafe state corresponds to all the signal lights being red (the 'stop' condition), with all the points locked in their previous positions. This should bring all the trains safely to a halt. In systems that possess failsafe states it is often appropriate to adopt this state if there is any doubt as to the correct operation of the system.

Unfortunately, many systems do not possess failsafe states. For example, in a fly-by-wire aircraft there is no combination of the various outputs from the avionics computers that will maintain safety – the only safe state for the aircraft is on the ground! Under these circumstances the system must continue operating to maintain safety.

System integrity

This term may be defined as follows:

> *The **integrity** of a system is its ability to detect faults in its own operation and to inform a human operator.*

Here the main emphasis is on fault detection rather than fault tolerance. System integrity is of particular importance in critical systems that possess failsafe states. In such circumstances the system may be designed so that it will enter its failsafe state if there is any uncertainty about the system's correctness. Examples of systems where high integrity is important include the railway signalling example cited above, and systems such as aircraft autopilots. In this latter example the pilot, and no doubt the passengers, would prefer that the aircraft be flown manually if there was any doubt about the correctness of the autopilot's operation.

Over the years the word **integrity** has become widely used in expressions such as **high-integrity system** and **safety integrity level**. When used in these ways integrity takes on a much broader meaning, encompassing considerations of reliability, availability and other factors associated with critical systems. In many ways integrity is here being used as a synonym for **dependability**, as defined later in this section. This diverse use of the term can sometimes lead to misunderstanding and is symptomatic of the problems of varied definitions within this field.

Data integrity

In many applications the data held within the system is of great importance for a variety of reasons.

> ***Data integrity** is the ability of a system to prevent damage to its own database and to detect, and possibly correct, errors that do occur.*

Although data integrity is of great importance in almost all applications, there are numerous applications where it is vital. Examples that are not safety related include banking and insurance company systems, where the financial value of the data is very high. Data integrity may also be of importance in safety-related systems where, for example, the current state of the system is deduced and stored rather than measured directly. A corruption of this data could destroy the system's image of its own state and thus affect its operation.

System recovery

Although every effort may be made during the design process to produce a system that will tolerate faults, it cannot be guaranteed that a system will not fail. This might be due, for example, to a transient fault such as a noise 'spike' on the power supply or the effects of a lightning strike nearby. In many applications, particularly those that do not have failsafe states, it is vital that the system can detect the failure and restart itself quickly. Depending on the nature of the application, the recovery process may need to determine the current status of the system, to take appropriate action to continue operation, and to maintain safety.

Maintainability

It is useful to define two terms at this point:

> ***Maintenance** is the action taken to retain a system in, or return a system to, its designed operating condition.*

***Maintainability** is the ability of a system to be maintained.*

Maintainability may be treated in a qualitative or a quantitative manner. When the latter approach is adopted it is often expressed in terms of the **mean time to repair (MTTR)** the system in the event of a failure. Its relevance to safety is clear, as the availability of a system depends not only on how often the system fails, but also on the time taken to restore it to operation. In Chapter 7 we shall discuss the relationship between these factors in more detail.

The maintenance of safety-critical systems is greatly affected by the nature of the application. In some cases maintenance may be performed while the system is in service. This may entail performing repairs or preventative maintenance on a 'live' system, or shutting it down temporarily. Alternatively, maintenance may only be possible between periods of service. This latter situation usually exists in aerospace or avionics applications, where maintenance is only possible at the end of a flight.

One aspect of maintenance that is often overlooked is the problem of **maintenance-induced failures** (HMSO, 1992). Unfortunately, experience shows that it is not reasonable to assume that all repairs will be completely successful, and it is quite possible for a repair to produce separate, apparently unrelated, faults.

Considerations of maintainability play an important part in the design of safety-critical systems (Whetton, 1994) and we shall return to this issue when we consider design in more detail in Chapter 5.

Dependability

This term covers considerations of reliability, availability, safety, maintainability and other issues of importance in critical systems.

***Dependability** is a property of a system that justifies placing one's reliance on it.*

Dependability is quantified in terms of the various factors, such as reliability and availability, that combine to produce a dependable system. Dependability is of great importance in all critical systems, although the importance of the various factors that contribute to it will vary between applications.

Conflict between system requirements

We have seen that the requirements of a safety-critical system take many forms. There are certain characteristics that are invariably beneficial in the design of a system. These include high reliability and high availability. However, the relative importance of these factors will vary between applications. Unfortunately, in some cases there are conflicts between the various requirements of a system.

Some conflicting requirements are self-evident and exist in all engineering disciplines. For example, it is common to have a conflict between a desire for high performance and a desire for low cost. Opposing pressures from other factors, such as considerations of size and functionality, are also likely.

Requirements associated with safety may likewise conflict with other system requirements. It is clear from the definitions given above and in Chapter 1 that reliability and safety are not synonymous. Indeed, there are instances where reliability and safety place conflicting requirements on a system. In any application that has a failsafe state the system can guarantee safety by remaining in that state, thereby satisfying its safety requirements. However, this system would not satisfy its functional requirements and would have zero reliability and zero availability. It could be argued that while such a system is active it is less safe than when it has failed, since its failsafe condition guarantees safety whereas its active state may not. Therefore, safety must be reduced in order to achieve a functional system. Ultimately a compromise is required between the functionality of a system, perhaps quantified in terms of its reliability or availability, and its safety performance. If we take as an example a nuclear shutdown system, the cost of shutting down the reactor is very high and consequently false trips of the protection system are to be avoided. Faced with an uncertain situation the designer must decide whether the system will shut down the reactor or continue in operation. Such a decision is clearly a trade-off between considerations of safety and those of performance.

In circumstances as described above it is tempting to deduce that safety is incompatible with reliability. This is incorrect. It is true that these two requirements place opposing pressures on the designer, but the final design must satisfy *both* requirements in order to be acceptable. In such cases the hazard analysis and risk analysis phases of the project must identify *acceptable* levels of safety and reliability, and the final design must achieve both objectives. As in all engineering projects, some compromises may be required in order to achieve a realizable specification.

2.3 Safety requirements

In addition to the system requirements discussed in the previous section a safety-critical system will have to satisfy certain specific safety requirements relating to its function and characteristics. In order to determine these system-specific requirements a series of tasks must be performed. The main stages of this process may be categorized as follows:

- identification of the hazards associated with the system;
- classification of these hazards;
- determination of methods for dealing with the hazards;
- assignment of appropriate reliability and availability requirements;

- determination of an appropriate safety integrity level;
- specification of development methods appropriate to this integrity level.

In order to determine the characteristics that a system must have in order to be safe, it is necessary to understand the ways in which the system could harm people or property. A computer-based system has the capability of doing harm through the various components it controls. For each component that has the potential to do harm it is necessary to identify all actions that could be dangerous. This in turn shows the system outputs that have safety implications. The capability to do harm to people, property or the environment, is termed a **hazard**, and in Chapter 3 we look at the process of hazard analysis which is used to identify potential dangers associated with the system.

Having identified the hazards associated with a system it is useful to classify them by their severity and their nature. The **severity** of a hazard is related to the consequences of any accident that might occur as a result of that hazard. The **nature** of a hazard has considerable impact on the manner in which it may be controlled. In some cases hazards arise as a consequence of factors that are under the direct control of the system. For example, a laser control system may be able, instantly, to turn off a source of radiation that can harm an individual. The situation may be more complicated if there is a delay between the control system selecting a safe condition and the equipment entering that state. Examples of such an arrangement would be a high-voltage source associated with some capacitance, or a moving part with considerable inertia. In each case the equipment could remain hazardous for some time after the control system turns off the source of the hazard. There are also cases where the relationship between the control function and the source of danger is less direct and the control system has only a limited influence on the source of potential harm. An example of such an arrangement might be a set of road traffic signals. Here the hazards are associated with cars and pedestrians, which are not under the direct control of the system.

The importance of a hazard is related to both its severity and its frequency of occurrence. These two factors are combined within the concept of **risk**, which is discussed in some detail in Chapter 4.

Having identified all the hazards within the system it is then necessary to identify methods of dealing with each of them. For each action that the system can perform it is necessary to decide the conditions for which that action is safe. Clearly, if a particular action is never safe the system must be redesigned to preclude that action. Once appropriate conditions for safety have been established, these become part of the safety requirements of the system. For hazards that are under the direct control of the system, defining conditions of safety is usually straightforward. For example, in the laser control system mentioned earlier the conditions for safe operation might be expressed as follows:

do not turn on the laser until it is 'safe'
turn off the laser if it is 'unsafe'

where the term 'safe' would be clearly defined. The definition of safe would certainly include consideration of the presence of people within the danger area. Mechanisms which ensure that potentially hazardous actions are only performed at times when they are safe are termed **interlocks**. In cases where delays are present, some form of **guard** would be necessary to keep people away from dangerous parts of the system until they had become safe. These guards might take the form of safety cages on machine tools or automatic barriers on railway crossings. The operation of the guard mechanism might be controlled by a condition of the form

keep people away from the equipment until it becomes 'safe'

where the presence of a safe condition would be detected by the system in some way. In applications where the system has only indirect control over the source of the hazard, guards can be used to detect dangerous conditions and to keep people away from hazardous areas while they persist. For example, a gas sensing system for a coal mine cannot control the levels of gas within the mine, but it can use warning signs and alarms to prevent people from entering the danger area if high levels of dangerous gases are present.

It can be seen that the design of safety-critical systems is based on the identification and control of the potentially dangerous actions (or inactions) of the system. One aspect of the control of hazards is based on the use of interlocks, which prevent dangerous operations from being performed unless it is safe to do so, and guards, which keep people away from the system while it is hazardous. Interlocks use **sensors** to detect the state of the system in order to detect dangerous conditions, and guards use **actuators** to prevent exposure to danger. The safety requirements documentation must identify the potential hazards of a system and, where appropriate, define an interlock mechanism that will ensure its safety.

In some cases the hazards associated with a system are not related to dangerous actions that may be controlled by the use of interlocks and guards. For example, if a computer is used at the heart of an aircraft flight control system, the failure of the computer itself represents a hazard that cannot be treated in this way. Safety requires that the computer function correctly, and considerations of reliability, availability and other issues form an essential part of the safety requirements of the system.

Where interlock and guard mechanisms are used, they may also have implications for the reliability and availability requirements of the computer system. Wherever possible, interlocks and guards should be provided by simple, non-programmable means that remove the processor from the task of ensuring safety (Leveson, 1991). Where this is not possible, these interlock mechanisms will themselves rely on the correct functioning of the computer and the dependability of the system must be appropriate to provide adequate safety. The requirements of the system in this respect will be determined by considerations of the associated risks. Risk analysis is discussed in some detail in Chapter 4.

Having assigned appropriate reliability and availability requirements to the system it is then useful to assign it to an appropriate **safety integrity level**.

This is used to distinguish between systems requiring different levels of dependability, or 'integrity', and determine the development methods to be adopted. Systems that are assigned a high integrity level will be subjected to much more rigorous design and testing methods than could be justified for less demanding applications. The assignment of integrity levels will be described in Chapter 4, and the choice of appropriate development techniques will be discussed in Chapter 5 and elsewhere within the text.

The role of standards

Within many industrial sectors a great deal of experience has been gained in the use of computer systems in safety-critical applications. Much of this knowledge is encapsulated within numerous standards and guidelines that are used to provide assistance to engineers working within these areas, and to help them to achieve uniformly high levels of quality and safety. Standards fulfil several important roles, including:

- helping staff to ensure that a product meets a certain level of quality;
- helping to establish that a product has been developed using methods of known effectiveness;
- promoting a uniformity of approach between different teams;
- providing guidance on design and development techniques;
- providing some legal basis in the case of a dispute.

In some industries all systems must conform to specific standards that stipulate both system requirements and the development methods to be used. In these industries certification of the final system depends on adherence to these standards and meeting the acceptance criteria that they define. In other areas the use of standards is more flexible, with regulating authorities negotiating with system designers on appropriate design and development techniques. Here a company may seek certification by claiming conformance to a particular standard, or may suggest alternative methods. The regulating authority may accept alternatives to those given within the standards, but is likely to be biased in favour of the established techniques. Generally companies will opt to follow an established standard, but may seek permission from the certifying body to deviate from it in certain areas where its requirements are seen to be inappropriate.

In addition to formal standards there are several **guidelines** and **codes of practice** that provide advice. These may be used to give guidance on how a system may be produced in order to satisfy a particular standard. Alternatively, they may provide more general information that is not related to projects within a specific industry. Some standards provide both requirements and guidelines within a single document. Perhaps for this reason, the term 'standard' is often used to refer to both formal standards and more informal guidelines.

Sector-specific standards and guidelines, such as those in the aerospace, nuclear or mining industries, identify and discuss the major hazards found within those industries. Levels of risk are associated with requirements for reliability and availability that will be considered acceptable by the regulatory authorities and by the industries themselves. As such, the standards provide a basis for the production of safety requirements for projects within those industries. More generic standards and guidelines have also been developed to provide more general guidance of relevance to all industrial sectors (HSE, 1987a, b; IEC, 1995). In Chapter 14 we shall return to the issue of standards when we consider the process of certification.

2.4 The safety case

In many industries highly critical systems will require certification by a regulating authority before they may be put into service. As part of the certification process, the operator of the system will be required to produce a **safety case** that sets out the safety justification for the system. This describes the design and assessment techniques used in the development of the system. The safety case is sometimes referred to as a **safety argument**, a **safety justification** or a **safety assessment report**. Even in situations where a safety case is not a formal requirement of a regulatory authority or customer, the provision of such a document is accepted as good engineering practice.

The argument within the safety case is normally based on engineering judgement rather than strict formal logic. This is generally supported by some form of probabilistic risk assessment, using techniques that will be discussed in later chapters.

It is not the purpose of the safety case to prove that the system in question is safe – such a proof is a theoretical impossibility since no system is absolutely safe. The safety case does, however, provide evidence that the risks associated with the system have been carefully considered and that steps have been taken to deal with them appropriately. The document must identify all matters significant to the safety of the system and show how these issues have been addressed. It must also give evidence of precautions taken to ensure safe operation. The safety case is concerned not only with design issues, but also with considerations of assessment and project management.

We shall look in more detail at the process of certification in Chapter 14.

REFERENCES

HMSO (1992). *Dangerous Maintenance*. London: Her Majesty's Stationery Office

HSE (1987a). *Programmable Electronics Systems: An Introductory Guide*. Health and Safety Executive. London: Her Majesty's Stationery Office

HSE (1987b). *Programmable Electronics Systems: General Technical Guidelines*. Health and Safety Executive. London: Her Majesty's Stationery Office

IEC (1995). Draft International Standard 1508. *Functional Safety: Safety-Related Systems*. Geneva: International Electrotechnical Commission

Leveson N.G. (1991). Software safety in embedded computer systems. *Comm. ACM*, 34(2), 34–46

Whetton C. (1994). Maintainability and its influence on system safety. In *Technology and Assessment of Safety-Critical Systems* (Redmill F. and Anderson T., eds), pp. 31–54. London: Springer-Verlag

FURTHER READING

Terry G.J. (1991). *Engineering System Safety*. London: Mechanical Engineering Publications

PROBLEMS

2.1 How are the safety aspects of the system requirements documented?

2.2 Define the term 'reliability' and explain why this factor varies with time. In what kind of application is high reliability of particular importance? Give two examples, not mentioned in the text, where reliability is of paramount importance.

2.3 Explain the meaning of the term 'availability'. What factors affect the relative importance of availability and reliability? Suggest two applications, not given in the text, where availability might be of more importance than reliability.

2.4 What is meant by the term 'unavailability'? Illustrate your answer with a numerical example.

2.5 Discuss the characteristics and advantages of 'failsafe' operation. Give two examples, not mentioned in the text, of systems that utilize failsafe states. Why do some systems not possess failsafe states?

2.6 Explain the meaning of 'system integrity'. In what classes of applications is integrity of great importance? Give two examples, not given in the text, of systems where high integrity is vital.

2.7 What is meant by 'data integrity'? Give an example to illustrate the importance of this characteristic.

2.8 Under what circumstances is system recovery a vital feature of a critical system? Give examples of applications where the ability to recover from a system crash is important.

2.9 Explain the meanings of the terms 'maintenance' and 'maintainability'. How are these factors related to safety? How is maintainability usually quantified?

2.10 What is meant by the term 'dependability'? How is this factor measured?

2.11 Give examples of conflicts that may exist between various aspects of a system's requirements. How can these problems be overcome?

2.12 Why are considerations of high reliability sometimes in conflict with requirements for safety? Under what circumstances could an unreliable system be safe?

2.13 What is meant by the term 'hazard'? Give some simple examples of hazards.

2.14 What factors combine to determine the risk associated with a hazard?

2.15 Give an example, not cited within the text, of a hazard that is under the direct control of a computer system.

2.16 Give an example, not given in the text, of a hazard that persists after the controller has selected a 'safe' state.

2.17 Give an example of a hazard, not cited in the text, where a controller has only indirect control over a hazard.

2.18 Explain, with the aid of examples, how interlocks and guards are used to control hazards.

2.19 Why is it preferable to implement interlocks and guards using simple, non-programmable components wherever possible?

2.20 What is meant by the term 'safety case'? What aspects are covered by a safety case?

7 System Reliability

CHAPTER CONTENTS

7.1 Introduction

Background

It can be argued that the field of reliability engineering was established largely as a result of problems experienced during the second world war. The vast resources allocated to military equipment resulted in previously unknown levels of production. However, much of the equipment failed very quickly, often before reaching service. Many of the problems could be attributed to the failure of electronic components that often had a functional lifetime of only a few hours. The American government was particularly concerned about the implications of these problems, and during the 1940s a number of research groups were established in the US in an attempt to improve the reliability of electronic components. This work was concerned with the electrical properties of the components, and also environmental factors such as vibration and shock. In 1950 the US Department of Defense set up an ad hoc group to look at the reliability of electronic equipment for the army, navy and air force, and this produced its final report in 1952. In August 1952 the US Department of Defense joined with representatives of the electronics industry to form the Advisory Group on Reliability of Electronic Equipment (AGREE). This began a five-year project that resulted in the production of a wealth of advice concerning the assessment of reliability during the manufacture of electronic systems. On the basis of this work, and much that followed, the US military issued a series of specifications for their suppliers, covering the reliability of electronic components. During the 1950s the material covered in these documents became established as good working practice within industry in both the military and civil sectors.

In the postwar years the increased reliance on complex electronic systems emphasized the need for high-reliability systems. The race into space, trans-oceanic telephone links, communications satellites, avionics and nuclear power, are all examples of emerging technologies that were heavily dependent on the use of sophisticated electronics that had to be reliable. In parallel with these needs were the equally important considerations of economics. Products that were unreliable would not sell. In order for electronics to fulfil its role within high-volume consumer products, components had to be dependable throughout the lifetime of the product.

Over the years the reliability of electronic components has increased enormously and in most consumer products greatly exceeds that of the other components. However, our increasing reliance on computer-based systems in critical applications pushes the requirement for reliability to the limit.

Applicability

In Chapter 6 we looked at several architectures that could be used to prevent a fault within a single component from producing a system failure. A reduction in system failures results in increased reliability, and in this chapter we look at methods of defining and quantifying reliability. High reliability is normally a necessary, but not sufficient, condition to guarantee safety.

We have seen in earlier chapters that systems may fail for a variety of reasons. In particular, faults may be random or systematic in nature. Random component failures can occur at any time, and it is not possible to predict when a particular device will fail. However, by observing a large number of similar devices it may be possible to perform a statistical analysis to allow an estimate to be made of the probability of failure within a certain time period. Failures caused as a result of systematic faults are not random in nature and therefore do not lend themselves to statistical analysis. Such failures may be predictable to some extent. For example, an overstressed device could fail whenever a particularly high load is applied, or a software fault might manifest itself at a set time. Once a systematic fault has been identified its likely effect on the reliability of the system may be studied, and in most highly critical applications any identified systematic faults would be removed. However, unidentified systematic faults represent a serious problem, as their effects are unpredictable and are not normally susceptible to statistical analysis. In computer-based systems the most common form of unidentified systematic fault is the software 'bug'.

Reliability engineers are divided on how to approach the problems associated with systematic faults such as those within software. Some say that since software faults are predictable, in that a fault will occur each time the code is executed, statistical analysis cannot be used. This leads to the view that we cannot apply a figure to the reliability of software. An alternative view is held by engineers who feel that it *is* appropriate to use statistical methods with software. They argue that because of the complexity of the software within a typical application, faults could take an almost limitless number of forms. The complex process involved in generating software means that faults could be randomly distributed throughout the code. The effects of these faults cannot be predicted and thus may be considered to be random in nature. Engineers who belong to this camp maintain that unknown software faults *are* sufficiently random to allow statistical analysis to be applied. It is important to note, however, that once a software bug has been identified it can no longer be considered to be random.

Faced with these two views of this issue it is difficult to give a definitive treatment to this topic. Unfortunately, those who decry the use of analytical techniques for software faults are unable to provide an alternative quantitative method of assessing their effects on reliability. Faced with this problem we shall turn our attention to the analysis and modelling of random failures. All engineers are happy with this method of treating random component failures. Some also believe that such methods may have a place in the analysis of software faults.

Reliability

Components that fail as a result of non-systematic faults will fail at a random time. For a given device it is not possible to predict the time of failure, but it *is* possible to quantify the rate at which members of a family of components will fail. Therefore, our definition of reliability is based on the probability of a device functioning correctly for a given period of time. In fact, reliability may be defined in a number of ways, some qualitative and others quantitative. Here we will take reliability to be the probability of a component or system functioning correctly over a given period of time under a given set of operating conditions. Clearly, by this definition reliability is a function of time and it is normally given the symbol $R(t)$.

If we consider a set of N identical components, all of which begin operating at the same time, then at some later time t, the number of components functioning correctly is $n(t)$, where

$$R(t) = \frac{n(t)}{N}$$

One may also define the term **unreliability**, which is the probability that a system will not function correctly over a given period of time. This term is given the symbol $Q(t)$ and is also called the probability of failure.

If the number of components to have failed during a time t is given the symbol $n_f(t)$, then

$$Q(t) = \frac{n_f(t)}{N}$$

and, from the definitions of reliability and unreliability, it is clear that

$$Q(t) = 1 - R(t)$$

Failure rate

Closely related to the reliability of a component is the rate at which such devices fail. The failure rate of a device or system is the number of failures within a given period of time. For example, if a device fails, on average, once in every 1000 hours of operation, it has a failure rate of 1/1000 failures per hour. The failure rate of a component normally varies with time and is given the symbol $z(t)$.

Failure rate may represent the frequency of the repeated failure of a single device, or the combined failure rate of a number of units. In the latter case the failure rate is the instantaneous rate at which components are failing, as a fraction of the number of devices still functioning. Thus

$$z(t) = \frac{1}{n(t)} \frac{\mathrm{d}n_{\mathrm{f}}(t)}{\mathrm{d}t}$$

Experience shows that the failure rate of electronic components normally exhibits distinctive characteristics, as shown in Figure 7.1. For obvious reasons, this characteristic is normally described as a 'bathtub' curve. Initially, components exhibit high 'infant mortality' owing to the presence of manufacturing faults that were not detected during the testing stage of their manufacture. As time passes the number of components containing these defects diminishes and the failure rate drops to a fairly constant level. At some later time the effects of ageing become apparent and the failure rate again rises as the devices 'wear out'. Manufacturers normally aim to use components only during the relatively constant part of this curve, and this is normally termed their 'useful life period'. In critical applications extended soak testing is used before systems are installed to catch any early failures. This is often in the form of 'accelerated life testing',

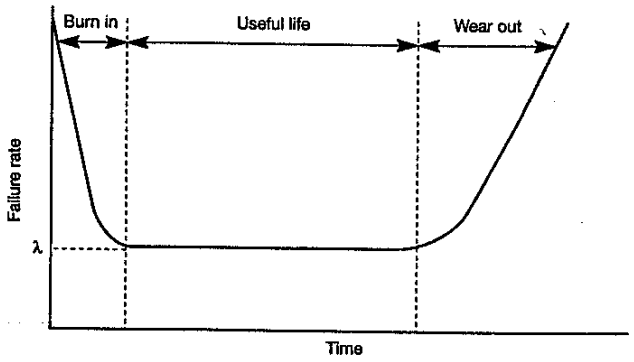


Figure 7.1 Typical variation of failure rate with time for electronic components.

using a more hostile environment than would be experienced by the unit in service. Such testing is often termed the 'burn-in' phase. This aims to produce faults in a few hours or days, that might take years to appear in normal operation. Critical systems would normally be replaced before their reliability falls to unacceptable levels owing to the effects of ageing. The relatively constant failure rate throughout the useful life of the unit is given the symbol λ.

It can be shown that during the useful-life stage the failure rate is related to the reliability of the device by the expression

$$R(t) = \mathrm{e}^{-\lambda t}$$

The exponential relationship between reliability and time is known as the **exponential failure law**. This indicates that for a constant failure rate reliability falls exponentially with time. Thus the probability of a system working correctly throughout a given period of time decreases exponentially with the length of this time period.

Time-variant failure rates

The exponential failure law described above assumes a constant failure rate. This relationship is widely used and is a good model for the random failure of hardware components. However, software failures are due to design faults and in some circumstances may be located and removed during the lifetime of the product. In this case the number of failures will tend to decrease with time and the constant failure rate model is no longer appropriate. This form of failure behaviour may be modelled by the Weibull distribution (Siewiorek and Swarz, 1982), which results in an expression for reliability of the form

$$R(t) = \mathrm{e}^{-(t/\eta)^{\beta}}$$

where β is the shape parameter and η is the characteristic life. It can be seen that for certain values of β the reliability increases with time, and approaches unity as t tends to infinity.

Although time-variant failure rates are of importance in the modelling of software and other systematic faults, the majority of reliability analysis is based on an assumption of a constant failure rate. For this reason the remainder of this chapter assumes this relationship.

Mean time to failure

Another way of describing the reliability of a system is to give its mean time to failure (MTTF), that is, the expected time that a system will operate before the first failure occurs. It can be shown that

$$\mathrm{MTTF} = \int_0^{\infty} R(t)\mathrm{d}t$$

and therefore, for the period where the failure rate is constant,

$$\text{MTTF} = \int_0^\infty e^{-\lambda t}\,dt = \frac{1}{\lambda} \tag{7.1}$$

This leads to the very simple result that the MTTF is the inverse of the constant failure rate of the system. Thus a system with a constant failure rate of 0.001 failures per hour will have a mean time to failure of 1000 hours. It is important to note that it is *not* reasonable to assume that such a system will operate correctly for 1000 hours. The reliability of a system at a time t is given by

$$R(t) = e^{-\lambda t}$$

and thus at a time $t = 1/\lambda$ (that is, at a time equal to its MTTF), the reliability is

$$R(t) = e^{-\lambda(1/\lambda)} = e^{-1} = 0.37 \tag{7.2}$$

Thus any given system has only a 37% chance of functioning correctly for an amount of time equal to the MTTF, or conversely a 63% chance of failing in this period. However, for a large number of units the MTTF represents the average time for which they would operate before their first failure.

Mean time to repair

The mean time to repair (MTTR) is quite simply the average time taken to repair a system that has failed, and to get it operational. The figure includes the time taken to detect the failure, locate the fault, effect a repair and reconfigure the system. Often the MTTR may be estimated during the design stage to allow system performance to be predicted, but may need to be determined experimentally when the system is operational.

Just as we describe the reliability of a system using its failure rate λ, we can quantify the repairability of a system using its repair rate μ, which is the average time taken to repair the system. It has units of repairs per hour. Just as the MTTF is $1/\lambda$, the MTTR is $1/\mu$.

Mean time between failures

If it can be assumed that once a failed system has been repaired its performance will be equivalent to the original system, then it is possible to predict the mean time between failures (MTBF), which is simply given by

$$\text{MTBF} = \text{MTTF} + \text{MTTR}$$

In most cases the time taken to repair the system will be small compared with the time for which the system operates, so in practice the MTBF will be numerically similar to the MTTF.

Availability

The availability of a system is the probability that the system will be functioning correctly at any given time. In other words, it is the fraction of the time for which it is operational. This can be expressed in terms of previously defined terms as

$$\text{Availability} = \frac{\text{Time system is operational}}{\text{Total time}} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

In critical systems the availability will normally be close to unity, and it is sometimes more convenient to describe a system in terms of its **unavailability**, where

$$\text{Unavailability} = 1 - \text{Availability}$$

It was noted in Chapter 2 that high availability is of paramount importance in some applications. Its relevance in terms of safety depends on whether safety can be guaranteed when the system is inoperative. In applications that have failsafe states it may be possible to maintain safety even when the computer-based system is not operating. In such cases availability may not be primary to safety, but will still be of importance to the overall performance of the system.

7.2 Reliability modelling

In the previous section we discussed several ways of describing the reliability of individual components. During the design stage of a project it is essential to be able to predict the final reliability of a complete system containing many parts. In this section we look at the use of reliability modelling to estimate the reliability of complex systems. The two most common methods are the 'combinational modelling' and the 'Markov state modelling' approaches.

Combinational models

Combinational reliability models allow the reliability of a system to be calculated from the reliability of its component parts. The components in question could be subsystems or individual electronic devices.

The model distinguishes between the situation where failure of any one of a number of components will cause system failure, and the case where several components must fail simultaneously to cause a malfunction. These two situations are modelled by the series and parallel models respectively. The symbols within reliability block diagrams are described within the international standard IEC 1078 (IEC, 1991).