Unit 1

11. Dependability and security

The dependability of systems is more important than their detailed functionality for the following reasons:

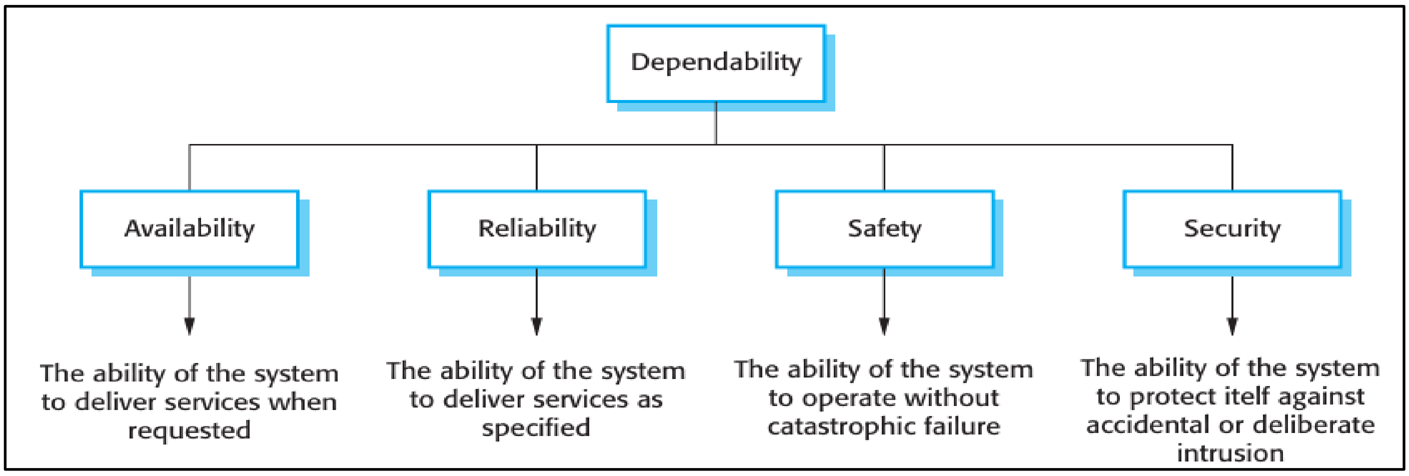
- *System failures affect a large number of people*. Many systems include functionality that is rarely used. If this functionality were left out of the system, only a small number of users would be affected. System failures, which affect the availability of a system, potentially affect all users of the system. Failure may mean that normal business is impossible.
- *Users often reject systems that are unreliable, unsafe, or insecure.* If users find that a system is unreliable or insecure, they will refuse to use it. Furthermore, they may also refuse to buy or use other products from the same company that produced the unreliable system, because they believe that these products are also likely to be unreliable or insecure.
- *System failure costs may be enormous.* For some applications, such as a reactor control system or an aircraft navigation system, the cost of system failure is orders of magnitude greater than the cost of the control system.
- *Undependable systems may cause information loss*. Data is very expensive to collect and maintain; it is usually worth much more than the computer system on which it is processed. The cost of recovering lost or corrupt data is usually very high.

When designing a dependable system, you therefore have to consider:

- *Hardware failure* System hardware may fail because of mistakes in its design, because components fail as a result of manufacturing errors, or because the components have reached the end of their natural life.
- *Software failure* System software may fail because of mistakes in its specification, design, or implementation.
- *Operational failure* Human users may fail to use or operate the system correctly. As hardware and software have become more reliable, failures in operation are now, perhaps, the largest single cause of system failures.

Dependability properties

The dependability of a computer system is a property of the system that reflects its trustworthiness. Trustworthiness here essentially means the degree of confidence a user has that the system will operate as they expect, and that the system will not 'fail' in normal use. It is not meaningful to express dependability numerically.


Dependability
Availability
Reliability
Safety
Security
The ability of the system to deliver services when requested
The ability of the system to deliver services as specified
The ability of the system to operate without catastrophic failure
The ability of the system to protect itelf against accidental or deliberate intrusion

There are four principal dimensions to dependability, as shown in Figure

- *Availability* Informally, the availability of a system is the probability that it will be up and running and able to deliver useful services to users at any given time.
- *Reliability* Informally, the reliability of a system is the probability, over a given period of time, that the system will correctly deliver services as expected by the user.
- *Safety* Informally, the safety of a system is a judgment of how likely it is that the system will cause damage to people or its environment.
- *Security* Informally, the security of a system is a judgment of how likely it is that the system can resist accidental or deliberate intrusions.

As well as these four main dependability properties, you may also think of other system properties as dependability properties:

- *Repairability* System failures are inevitable, but the disruption caused by failure can be minimized if the system can be repaired quickly. For that to happen, it must be possible to diagnose the problem, access the component that has failed, and make changes to fix that component. Repairability in software is enhanced when the organization using the system has access to the source code and has the skills to make changes to it. Open-source software makes this easier but the reuse of components can make it more difficult.
- *Maintainability* As systems are used, new requirements emerge and it is important to maintain the usefulness of a system by changing it to accommodate these new requirements. Maintainable software is software that can be adapted eco nomically to cope with new requirements, and where there is a low probability that making changes will introduce new errors into the system.
- *Survivability* A very important attribute for Internet-based systems is survivability. Survivability is the ability of a system to continue to deliver service whilst under attack and, potentially, whilst part of the system is disabled. Work on survivability focuses on identifying key system components and ensuring that they can deliver a minimal service. Three strategies are used to enhance survivability—resistance to attack, attack recognition, and recovery from the damage caused by an attack
- *Error tolerance* This property can be considered as part of usability and reflects the extent to which the system has been designed so that user input errors are avoided and tolerated. When user errors occur, the system should, as far as possible, detect these errors and either fix them automatically or request the user to reinput their data.

To develop dependable software, you therefore need to ensure that:

- You avoid the introduction of accidental errors into the system during software specification and development.
- You design verification and validation processes that are effective in discovering residual errors that affect the dependability of the system.
- You design protection mechanisms that guard against external attacks that can compromise the availability or security of the system.
- You configure the deployed system and its supporting software correctly for its operating environment.

In addition, you should usually assume that your software is not perfect and that software failures may occur. Your system should therefore include recovery mechanisms that make it possible to restore normal system service as quickly as possible.

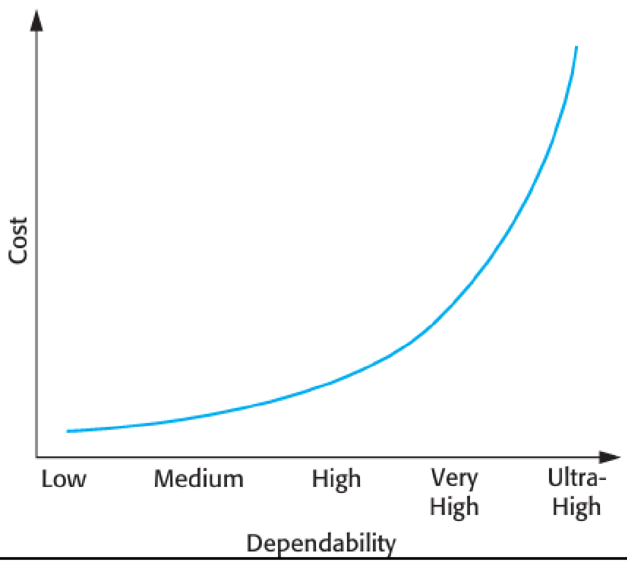


Figure 11.2 Cost/dependability curve

Above shows that the relationship between costs and incremental improvements in dependability. If your software is not very dependable, you can get significant improvements at relatively low costs by using better software engineering. However, if you are already using good practice, the costs of improvement are much greater and the benefits from that improvement are less. There is also the problem of testing your software to demonstrate that it is dependable. This relies on running many tests and looking at the number of failures that occur. As your software becomes more dependable, you see fewer and fewer failures.

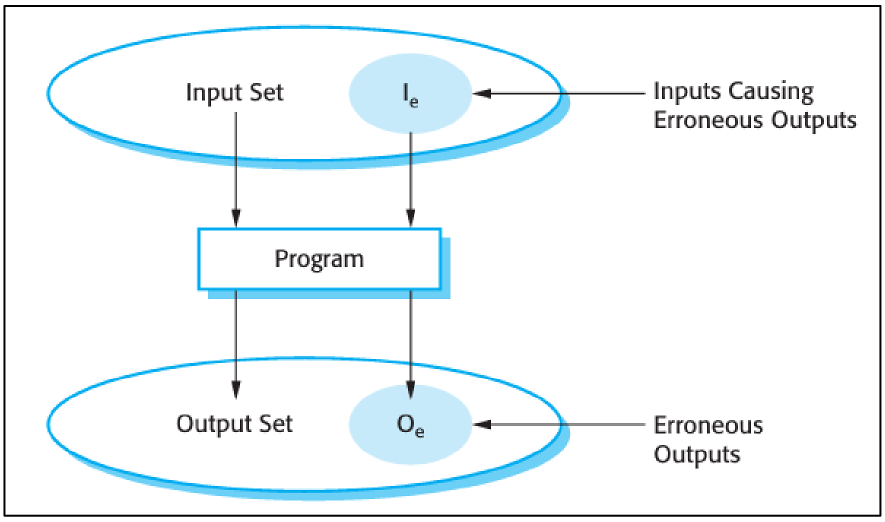
Availability and reliability

- Reliability The probability of failure-free operation over a specified time, in a given environment, for a specific purpose.
- Availability The probability that a system, at a point in time, will be operational and able to deliver the requested services.

Availability and reliability are obviously linked as system failures may crash the system. However, availability does not just depend on the number of system crashes, but also on the time needed to repair the faults that have caused the failure. Therefore, if system A fails once a year and system B fails once a month then A is clearly more reliable than B. However, assume that system A takes three days to restart after a failure, whereas system B takes 10 minutes to restart. The availability of system B over the year (120 minutes of down time) is much better than that of system A (4,320 minutes of down time).

Reliability Terminology

Term	Description
Human error or mistake	Human behavior that results in the introduction of faults into a system. For example, in the wilderness weather system, a programmer might decide that the way to compute the time for the next transmission is to add 1 hour to the current time. This works except when the transmission time is between 23.00 and midnight (midnight is 00.00 in the 24-hour clock).
System fault	A characteristic of a software system that can lead to a system error. The fault is the inclusion of the code to add 1 hour to the time of the last transmission, without a check if the time is greater than or equal to 23.00.
System error	An erroneous system state that can lead to system behavior that is unexpected by system users. The value of transmission time is set incorrectly (to 24.XX rather than 00.XX) when the faulty code is executed.
System failure	An event that occurs at some point in time when the system does not deliver a service as expected by its users. No weather data is transmitted because the time is invalid.



Most inputs do not lead to system failure. However, some inputs or input combi nations, shown in the shaded ellipse I_e in above figure, cause system failures or erroneous outputs to be generated. The program's reliability depends on the number of system inputs that are members of the set of inputs that lead to an erroneous output. If inputs in the set I_e are executed by frequently used parts of the system, then failures will be frequent. However, if the inputs in I_e are executed by code that is rarely used, then users will hardly ever see failures.

System faults do not always result in system errors and system errors do not necessarily result in system failures. The reasons for this are as follows:

- Not all code in a program is executed. The code that includes a fault (e.g., the failure to initialize a variable) may never be executed because of the way that the software is used
- Errors are transient. A state variable may have an incorrect value caused by the execution of faulty code. However, before this is accessed and causes a system failure, some other system input may be processed that resets the state to a valid value.
- The system may include fault detection and protection mechanisms. These ensure that the erroneous behaviour is discovered and corrected before the system services are affected.

Safety

Safety-critical systems are systems where it is essential that system operation is always safe; that is, the system should never damage people or the system's environment even if the system fails.

Safety-critical software falls into two classes:

- Primary safety-critical software This is software that is embedded as a con troller in a system. Malfunctioning of such software can cause a hardware malfunction, which results in human injury or environmental damage. The insulin pump software, is an example of a primary safety-critical system. System failure may lead to user injury.
- Secondary safety-critical software This is software that can indirectly result in an injury. An example of such software is a computer-aided engineering design system whose malfunctioning might result in a design fault in the object being designed. This fault may cause injury to people if the designed system malfunctions. Another example of a secondary safety-critical system is the mental health care management system, MHC-PMS. Failure of this system, whereby an unstable patient may not be treated properly, could lead to that patient injuring themselves or others.

Safety terminology

Term	Definition
Accident (or mishap)	An unplanned event or sequence of events which results in human death or injury, damage to property, or to the environment. An overdose of insulin is an example of an accident.
Hazard	A condition with the potential for causing or contributing to an accident. A failure of the sensor that measures blood glucose is an example of a hazard.
Damage	A measure of the loss resulting from a mishap. Damage can range from many people being killed as a result of an accident to minor injury or property damage. Damage resulting from an overdose of insulin could be serious injury or the death of the user of the insulin pump.
Hazard severity	An assessment of the worst possible damage that could result from a particular hazard. Hazard severity can range from catastrophic, where many people are killed, to minor, where only minor damage results. When an individual death is a possibility, a reasonable assessment of hazard severity is 'very high.'
Hazard probability	The probability of the events occurring which create a hazard. Probability values tend to be arbitrary but range from 'probable' (say 1/100 chance of a hazard occurring) to 'implausible' (no conceivable situations are likely in which the hazard could occur). The probability of a sensor failure in the insulin pump that results in an overdose is probably low.
Risk	This is a measure of the probability that the system will cause an accident. The risk is assessed by considering the hazard probability, the hazard severity, and the probability that the hazard will lead to an accident. The risk of an insulin overdose is probably medium to low.

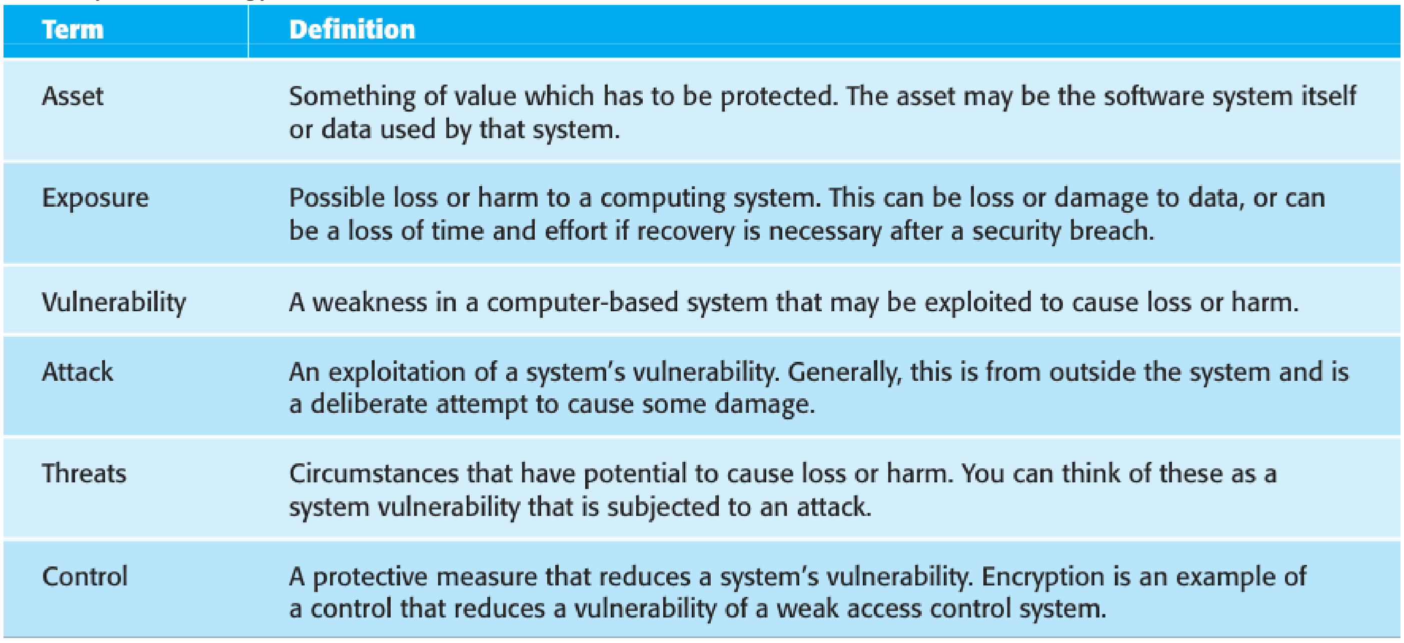
The key to assuring safety is to ensure either that accidents do not occur or that the consequences of an accident are minimal. This can be achieved in three complementary ways:

- *Hazard avoidance*, The system is designed so that hazards are avoided. For example, a cutting system that requires an operator to use two hands to press separate buttons simultaneously avoids the hazard of the operator's hands being in the blade pathway.
- *Hazard detection and removal,* The system is designed so that hazards are detected and removed before they result in an accident. For example, a chemical plant system may detect excessive pressure and open a relief valve to reduce these pressures before an explosion occurs.
- *Damage limitation*, The system may include protection features that minimize the damage that may result from an accident. For example, an aircraft engine normally includes automatic fire extinguishers. If a fire occurs, it can often be controlled before it poses a threat to the aircraft.

Security

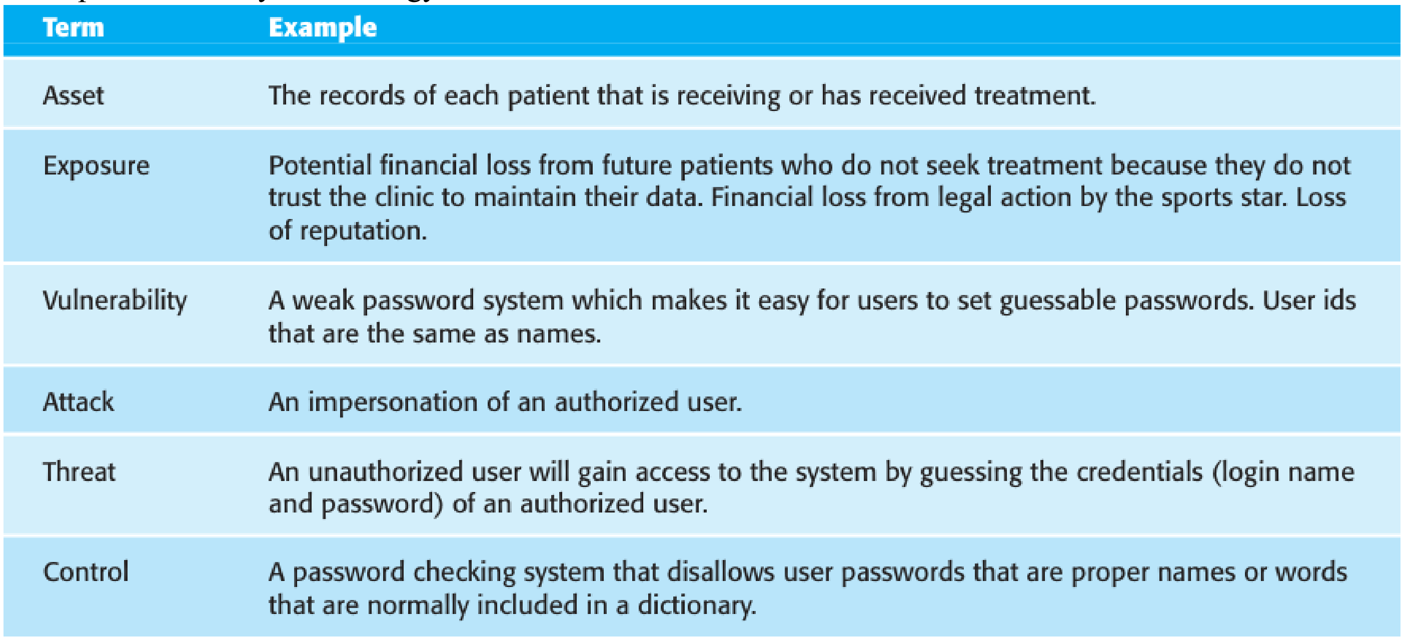
Security is a system attribute that reflects the ability of the system to protect itself from external attacks, which may be accidental or deliberate. These external attacks are possible because most general-purpose computers are now networked and are therefore accessible by outsiders. Examples of attacks might be the installation of viruses and Trojan horses, unauthorized use of system services or unauthorized modification of a system or its data.

Security terminology



Term	Definition
Asset	Something of value which has to be protected. The asset may be the software system itself or data used by that system.
Exposure	Possible loss or harm to a computing system. This can be loss or damage to data, or can be a loss of time and effort if recovery is necessary after a security breach.
Vulnerability	A weakness in a computer-based system that may be exploited to cause loss or harm.
Attack	An exploitation of a system's vulnerability. Generally, this is from outside the system and is a deliberate attempt to cause some damage.
Threats	Circumstances that have potential to cause loss or harm. You can think of these as a system vulnerability that is subjected to an attack.
Control	A protective measure that reduces a system's vulnerability. Encryption is an example of a control that reduces a vulnerability of a weak access control system.

Examples of security terminology



Term	Example
Asset	The records of each patient that is receiving or has received treatment.
Exposure	Potential financial loss from future patients who do not seek treatment because they do not trust the clinic to maintain their data. Financial loss from legal action by the sports star. Loss of reputation.
Vulnerability	A weak password system which makes it easy for users to set guessable passwords. User ids that are the same as names.
Attack	An impersonation of an authorized user.
Threat	An unauthorized user will gain access to the system by guessing the credentials (login name and password) of an authorized user.
Control	A password checking system that disallows user passwords that are proper names or words that are normally included in a dictionary.

In any networked system, there are three main types of security threats:

- Threats to the confidentiality of the system and its data These can disclose information to people or programs that are not authorized to have access to that information.
- Threats to the integrity of the system and its data These threats can damage or corrupt the software or its data.
- Threats to the availability of the system and its data These threats can restrict access to the software or its data for authorized users.

The controls that you might put in place to enhance system security are comparable to those for reliability and safety:

- *Vulnerability avoidance* Controls that are intended to ensure that attacks are unsuccessful. The strategy here is to design the system so that security problems are avoided. For example, sensitive military systems are not connected to public networks so that external access is impossible. You should also think of encryption as a control based on avoidance. Any unauthorized access to encrypted data means that it cannot be read by the attacker. In practice, it is very expensive and time consuming to crack strong encryption.

- *Attack detection and neutralization* Controls that are intended to detect and repel attacks. These controls involve including functionality in a system that monitors its operation and checks for unusual patterns of activity. If these are detected, then action may be taken, such as shutting down parts of the system, restricting access to certain users, etc.
- *Exposure limitation and recovery* Controls that support recovery from problems. These can range from automated backup strategies and information 'mirroring' to insurance policies that cover the costs associated with a successful attack on the system.