Software Dependability & Critical Systems

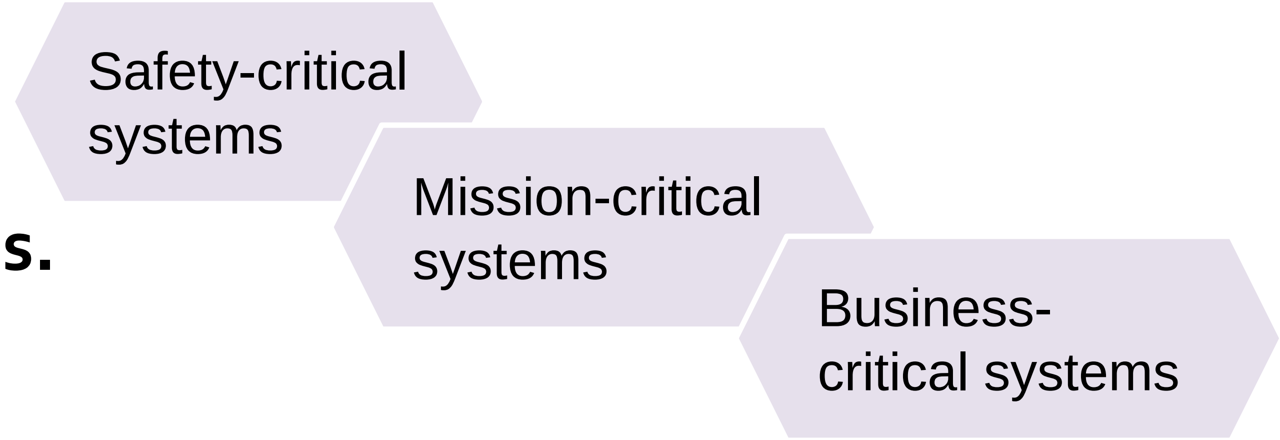
Table of Contents

8.1 Critical Systems

An event that occurs at some point in time when the system ***<u>does not</u>*** *deliver a service as expected by its users (e.g.__)*

What is a critical system?

- **A system whose** failures **can result in significant economic losses, physical damage or threats to human life**.
- There are 3 types of critical systems:
  a. **Safety-critical systems**
  b. **Mission-critical systems**
  c. **Business-critical systems.**



- <u>Dependability</u> is an essential attribute of critical systems.

8.1 Critical Systems

at some point in time the system ***does not*** *deliver a service as expected by its users*

There are 3 types of critical systems:

a. Safety-Critical Systems

- A system whose failure may cause:
  - injury,
  - loss of life or
  - major environmental damage.
- Examples:
  - **Airbag control system**
  - **Car reverse control system**
  - Flight auto-pilot control system
  - A control system for a chemical manufacturing plant
  - Nuclear reactor management system

b. Mission-Critical Systems

- A system whose failure may result in the failure of some goal-directed activity.
- Examples:
  - **GPS**
  - Navigational system for a spacecraft
  - Rocket-launcher system

c. Business-Critical Systems

- Failure of these systems can have serious consequences for the business
- Examples:
  - ERP
  - E-commerce website
  - Online banking system

8.1 Critical Systems

3 Causes of Failure:

- **Failure of <u>Hardware</u>** – due to *design and manufacturing errors or* components have reached their *end of life*.
- **Failure of <u>Software</u>** - due to *specification, design or implementation errors*.
- **Failure of <u>Operation</u>** – due to *Human errors*, perhaps the largest single cause of system failures in socio-technical systems.

<u>Costs</u> of Failure

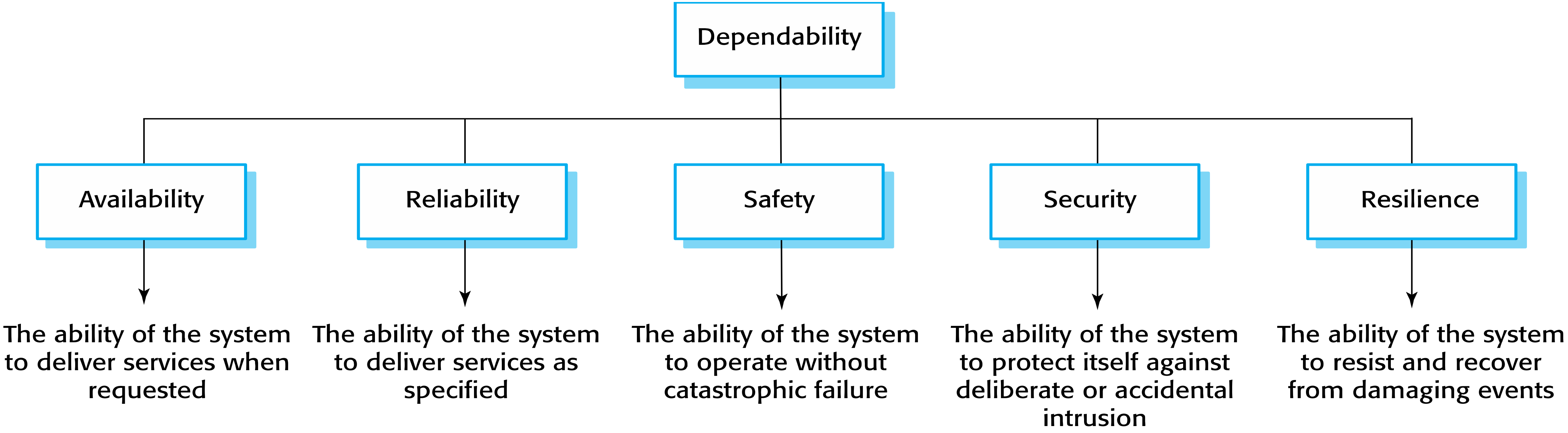
- Direct failure costs – system need to be replaced.
- Indirect failure costs – litigation (lawsuits) costs, deterioration of company's image and lost of business.

The costs of failure of a critical system are often very high

8.2 Dependability

- System dependability reflects the user's degree of trust in that system
  - i.e. the extent of the user's confidence that it will operate as users expect and that it will not fail in normal use.
- Covers the related systems attributes of reliability, availability and security.
  - These are all *inter-dependent.*

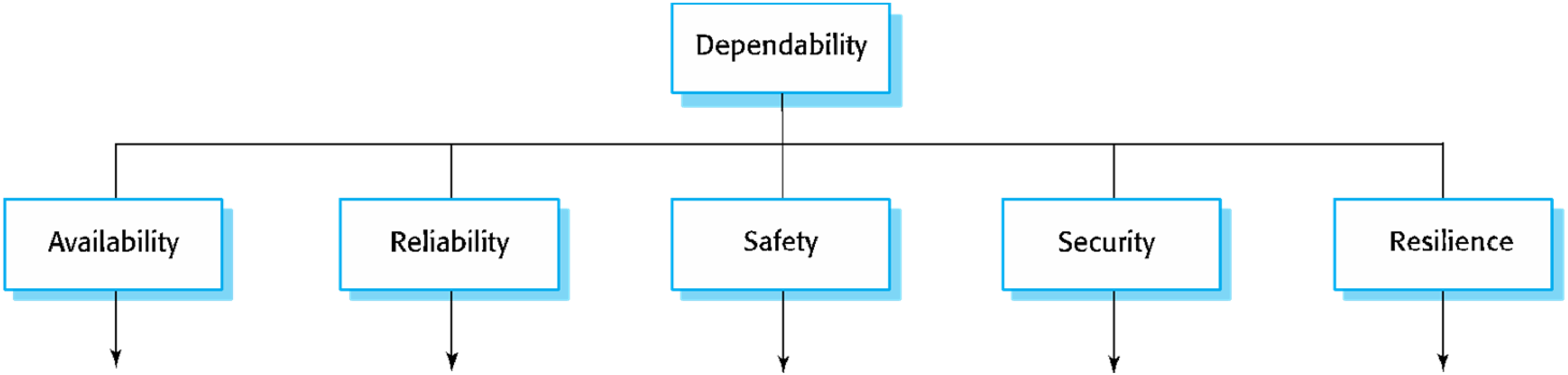
Dimensions of Dependability



Dimensions of Dependability

8.2 Dependability

In layman terms, System dependability refers to the availability, reliability, safety, security and resilience of a system



Availability: The ability of a system to be up and running and able to deliver useful services to users when needed

Reliability: The ability of a system to correctly deliver services as expected by users over a period of time.

Safety: The ability of a system to prevent or mitigate accidents, injuries, and other harmful events.

Security: The ability of a system to resist accidental or deliberate intrusions.

Resilience: The ability of a system to maintain continuity of its critical services in the presence of disruptive events such as equipment failure and cyber-attacks, etc

Principal Dependability Properties

8.2 Dependability

Availability, reliability, safety, security and resilience can also be describe as

Availability	The probability that the system will be up and running and able to deliver useful services to users.
Reliability	The probability that the system will correctly deliver services as expected by users.
Safety	A judgment of how likely it is that the system will cause damage to people or its environment.
Security	A judgment of how likely it is that the system can resist accidental or deliberate intrusions.
Resilience	A judgment of how well a system can maintain the continuity of its critical services in the presence of disruptive events such as equipment failure and cyber-attacks.

Other Dependability Properties

8.2 Dependability

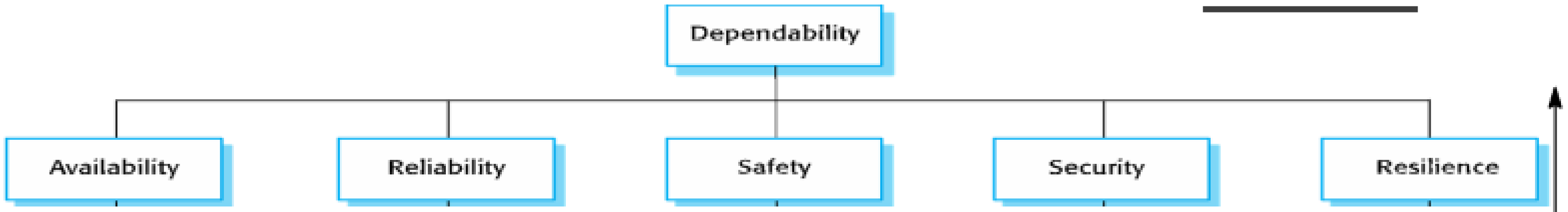
Repairability	Reflects the extent to which the system can be repaired in the event of a failure.
Maintainability	Reflects the extent to which the system can be adapted to new requirements.
Error tolerance	Reflects the extent to which user input errors can be avoided and tolerated.

Dependability Properties Inter-Dependencies

- **Availability & reliability**: Safe system operation depends on the system being available and operating reliably.
- **Reliability & security**: A system may be unreliable because its data has been corrupted by an external attack.
- **Availability & security** : Denial of service (DoS) attacks on a system are intended to make it unavailable.
- **Reliability, safety & security**: If a system is infected with a virus, you cannot be confident in its reliability or safety.

Dependability Costs

8.2 Dependability



- **Dependability costs increase exponentially when the** level of dependability required increased:
  - The use of more expensive development techniques and hardware to achieve higher levels of dependability.
  - The increased testing and system validation to convince the client and regulators that the required levels of dependability have been achieved.

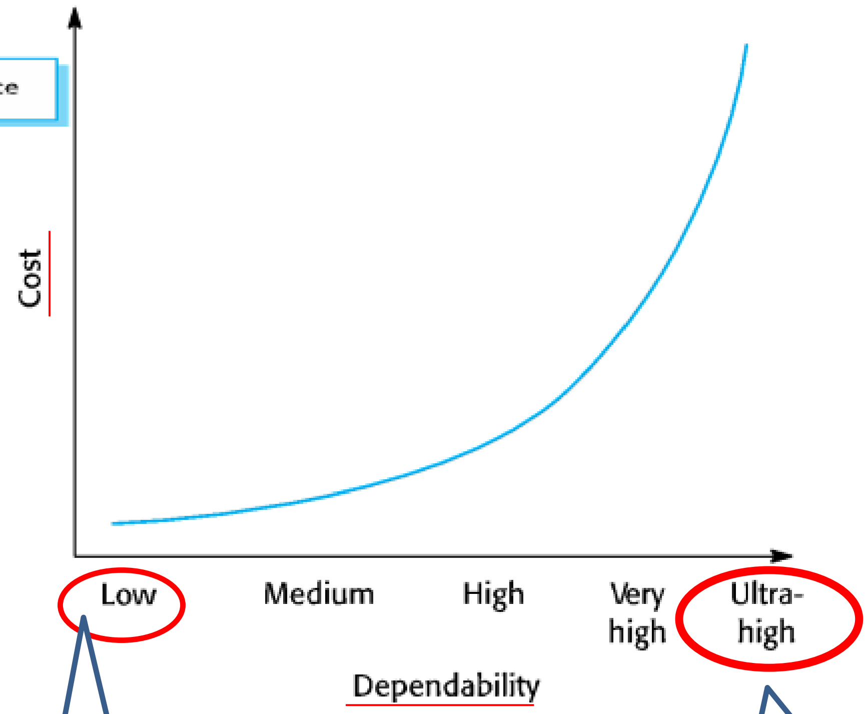


Fig: Cost/Dependability Curve

This system has high <u>availability</u> but is unreliable

This system is <u>reliable</u>, <u>safe</u> to use, <u>secured</u> and <u>available</u> 24/7/365

How to develop dependable software?

- Avoid the introduction of errors when developing the system.
- Design Verification & Validation (V&V) processes that are effective in discovering residual errors in the system.
- Design systems to be fault tolerant so that they can continue in operation when faults occur.
- Design protection mechanisms that guard against external attacks.

- Configure the system correctly for its operating environment.
  - Example of checklist: https://docs.oracle.com/cd/E13196_01/platform/docs81/deploy/checklist.html
- Include system capabilities to recognize external cyberattacks and to resist these attacks.
- Design recovery mechanisms to help restore system after a failure/cycber attack.

Regulated systems

8.3 Achieving Dependability

- Many critical systems (e.g., nuclear systems, air traffic control systems, medical devices) must be approved by an external regulator before the systems go into service.
- To achieve certification, companies developing safety-critical systems have to produce an extensive safety case to show that rules and regulations have been followed.
  - It can be as expensive develop the documentation for certification as it is to develop the system itself.

Redundancy and diversity

8.3 Achieving Dependability

2 approaches:

Redundancy

Keep more than one version of critical components so that a backup is available.

– E.g., if availability is critical (e.g. e-commerce systems), companies keep backup servers and switch to these automatically if failure occurs.

Diversity

Provide the same functionality in different ways in different components so that they will not fail in the same way.

– E.g., to provide resilience against external attacks, have different servers use different operating systems (e.g. Windows & Linux)

- Redundant and diverse components should be independent to avoid 'common-mode' failures
  - e.g., components implemented in different programming languages means that a compiler fault will not affect all of them.

Process Diversity and Redundancy

- Process activities such as validation, should not depend on a single approach, such as testing, to validate the system.
- Redundant and diverse process activities are important especially for verification and validation.
- Multiple, different process activities that complement each other and allow for **cross-checking** help to avoid process errors, which may lead to errors in the software.

Process Diversity and Redundancy

Challenges with redundancy & diversity

- Adding diversity and redundancy to a system increases the system **complexity**.
- This can increase the chances of error because of unanticipated interactions and dependencies between the redundant system components.
- Some engineers therefore advocate simplicity and extensive verification and validation. as a more effective route to software dependability.

Dependable Processes

8.3 Achieving Dependability

- To ensure a minimal number of software faults, have a well-defined, repeatable software process.
  - The process does not depend entirely on individual skills but can be enacted by different people.
  - The process activities includes significant effort devoted to V&V for fault detection.
- Regulators use information about the process to check if good software engineering practice has been used.

Repeatable process means
The process can be used in other projects by different team members, irrespective of who is involved in the development.

Dependable Processes Characteristics

8.3 Achieving Dependability

Explicitly defined

- There is a defined process model that is used to drive the software production process.
- Data must be collected during the process to prove that the development team has followed the process as defined in the process model.

Repeatable

- A process that does not rely on individual interpretation and judgment.
- The process can be repeated across projects and with different team members, irrespective of who is involved in the development.

Auditable	The process should be understandable by people apart from process participants, who can check that process standards are being followed and make suggestions for process improvement.
Diverse	The process should include redundant and diverse verification and validation activities.
Documentable	The process should have a defined process model that sets out the activities in the process and the documentation that is to be produced during these activities.
Robust	The process should be able to recover from failures of individual process activities.
Standardized	A comprehensive set of software development standards covering software production and documentation should be available.

Dependable Processes Activities

8.3 Achieving Dependability

Requirements reviews

To check that the requirements are complete and consistent.

Requirements management

To ensure that changes to the requirements are controlled and that the impact of proposed requirements changes is understood.

Formal specification

A mathematical model of the software is created and analyzed.

System modeling

Software design is explicitly documented as a set of graphical models, and the links between the requirements and these models are documented.

Design and program inspections

Different descriptions of the system are inspected and checked by different people.

Static analysis

Automated checks on the source codes of the program.

Test planning and management

A comprehensive set of system tests is designed. The testing process has to be carefully managed to demonstrate that these tests provide coverage of the system requirements and have been correctly applied in the testing process.

8.4 Availability and reliability

- **Reliability** The probability of failure-free system operation over a specified time in a given environment for a given purpose.

- **Availability** The probability that a system, at a point in time, will be operational and able to deliver the requested services.

- Availability & Reliability can be expressed **quantitatively**:
- **E.g.** if, the system availability is **0.999**, this means that, over some time period, the system is available for **99.9%** of that time
- **For reliability,** if on average, 2 inputs in every 1,000 cause failures, then the reliability, expressed as a rate of occurrence of failure = **0.002**

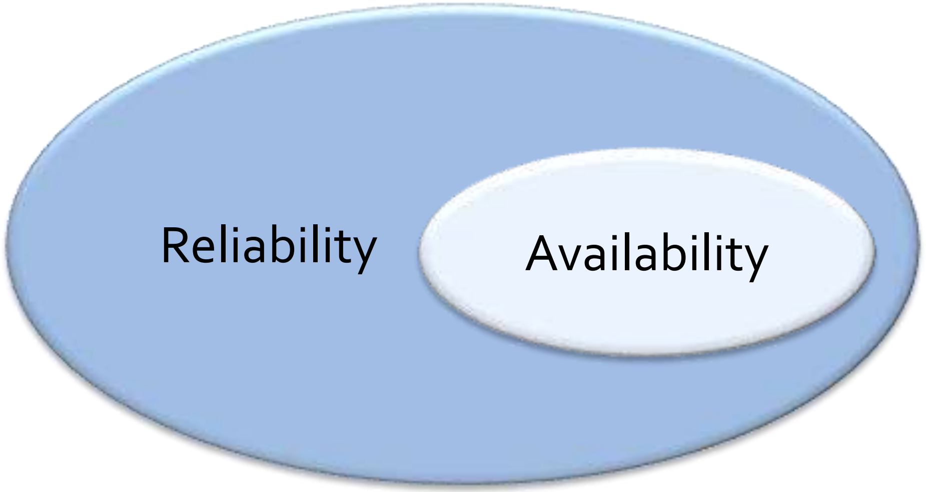
  *(Level of reliability = **99.8%**)*

Availability and Reliability

Availability and **Reliability** are closely related

- If a system is unavailable then it is not reliable
- **(e.g. a system that is often out of service is unreliable)**

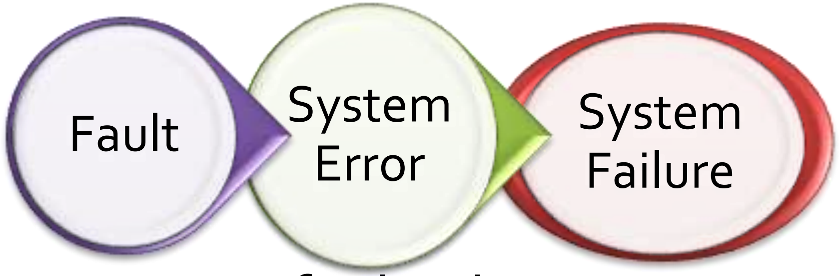
- **Availability** affect **Reliability**



Faults and Failures

8.4 Availability & Reliability

- Failures are a usually a result of system errors that are derived from faults in the system.



- However, faults do not necessarily result in system errors
  - The erroneous system state resulting from the fault may be transient and 'corrected' before an error arises.
  - The faulty code may never be executed.
- Errors do not necessarily lead to system failures
  - The error can be corrected by built-in error detection and recovery
  - The failure can be protected against by built-in protection facilities. These may, for example, protect system resources from system errors

System Fault is a characteristic of a software system that can lead to a system error

System Error can lead to system behavior that is unexpected by system users.

System failure when the system does not deliver a service as expected by users.

Faults and Failures

*at some point in time, **the system is <u>not able to</u> deliver a service as expected by its users***

System Fault $\xrightarrow{\text{May/may not produce}}$ **System Error** $\xrightarrow{\text{May/may not produce}}$ **System Failure**

- Reasons:
  - Not all code in a program is executed. **The** faulty code **may <u>never</u> be executed** because of the way the SW is used.
  - **Errors are transient (occur briefly).** A state <u>variable</u> may have an incorrect value caused by the execution of faulty code. But, before this is accessed and caused a system failure, some other system input may be processed that <u>resets</u> the state to a valid value.
  - **The system may have** fault detection and protection mechanisms to ensure that erroneous behaviour is discovered and corrected before the system services are affected.

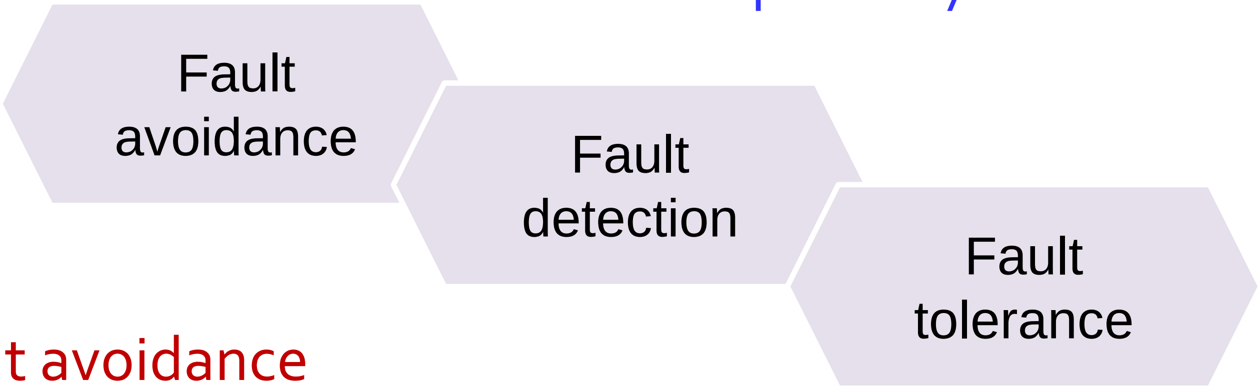
E.g Human Error, System Fault, System Error and System Failure

Term	Description
Human error	Human behavior that results in the introduction of faults into a system. *E.g.*, for hourly temperature readings, the programmer introduces a logic error: he computes the time for the next transmission just by adding 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. *E.g.*, 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. *E.g.*, 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. *E.g.*, *no* weather data is transmitted because the time is invalid.

Fault Management

8.4 Availability & Reliability

3 complementary approaches to improve system reliability:



- Fault avoidance
  - Development techniques are used to either minimize the possibilities of mistakes and/or trap mistakes before these result in system fault.
  - e.g., avoiding error-prone programming language constructs like pointers, use static analysis to detect program anomalies.
- Fault detection
  - Verification and validation techniques are used to discover and remove faults in a system before it is deployed.
  - e.g., implement systematic system testing and debugging.
- Fault tolerance
  - Use techniques that ensure that faults in a system do not result in system errors or failures.
  - The incorporation of **self-checking facilities** in a system and the use of **redundant system modules.**
  - e.g., scan disk

Reliability and specifications

8.4 Availability & Reliability

- Reliability can only be defined formally with respect to a system specification i.e. a failure is a deviation from a specification.
- However, many specifications are incomplete or incorrect – hence, a system that conforms to its specification may ‘fail’ from the perspective of system users.
- Furthermore, users don’t read specifications so don’t know how the system is supposed to behave.
- Therefore perceived reliability is more important in practice.

Availability Perception

- Availability is usually expressed as a percentage of the time that the system is available to deliver services e.g. 99.95%.

- However, this does not take into account 2 factors:
  - The number of users affected by the service outage.
    - Loss of service in the middle of the night is less important for many systems than loss of service during peak usage periods.
  - The length of the outage.
    - The longer the outage, the more the disruption.
    - Several short outages are less likely to be disruptive than one long outage.
    - Long repair times are a particular problem.

System Reliability Requirements

8.4 Availability & Reliability

- Functional Reliability Requirements (FRR) define system and software functions that avoid, detect or tolerate faults in the software and so ensure that these faults do not lead to system failure.
- Reliability is a measurable system attribute so non-functional reliability requirements may be specified quantitatively. These define the number of failures that are acceptable during normal use of the system or the time in which the system must be available.

Reliability Metrics

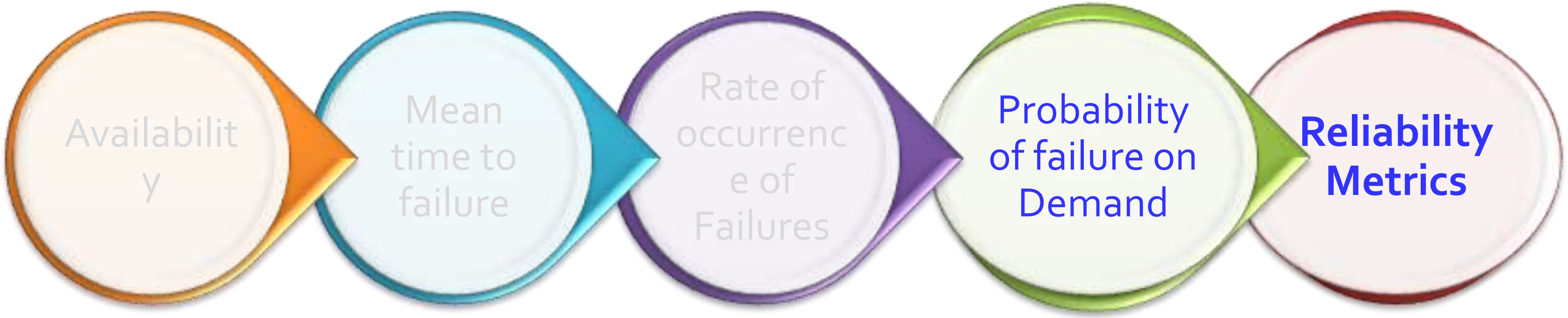
8.4 Availability & Reliability

- System reliability is measured by counting the number of operational failures and, where appropriate, relating these to the demands made on the system and the time that the system has been operational.
- A long-term measurement programme is required to assess the reliability of critical systems.

- Metrics
  - Probability of failure on demand (POFOD)
  - Rate of occurrence of failures
  - Mean time to failure
  - Availability

Probability of Failure on Demand (POFOD)

- This is the probability that the system will fail when a service request is made. Useful when demands for service are intermittent and relatively infrequent.
- Appropriate for protection systems where services are demanded occasionally and where there are serious consequence if the service is not delivered.
- Relevant for many safety-critical systems with exception management components
  - Emergency shutdown system in a chemical plant.



Rate of Fault Occurrence (ROCOF)

- Reflects the rate of occurrence of failure in the system.
- ROCOF of 0.002 means 2 failures are likely in each 1000 operational time units e.g. 2 failures per 1000 hours of operation.
- Relevant for systems where the system has to process a large number of similar requests in a short time
  - Credit card payment processing system, airline booking system

Mean Time to Failure (MTTF)

8.4 Availability & Reliability

- The predicted elapsed time between inherent failures of the system during normal system operation.
- Relevant for systems with long transactions i.e. where system processing takes a long time.
- MTTF should be longer than expected transaction length.

Availability

8.4 Availability & Reliability

- Measure of the fraction of the time that the system is available for use.
- Takes repair and restart time into account
- Availability of 0.998 means software is available for 998 out of 1000 time units.
- Relevant for non-stop, continuously running systems
  - telephone switching systems, Internet Streaming Service.

Availability specification

Availability	Explanation
0.9	The system is available for 90% of the time. This means that, in a 24-hour period (1,440 minutes), the system will be unavailable for 144 minutes.
0.99	In a 24-hour period, the system is unavailable for 14.4 minutes.
0.999	The system is unavailable for 84 seconds in a 24-hour period.
0.9999	The system is unavailable for 8.4 seconds in a 24-hour period. Roughly, one minute per week.

Non-functional Reliability Requirements

8.4 Availability & Reliability

- Non-functional reliability requirements are specifications of the required reliability and availability of a system using one of the reliability metrics (POFOD, ROCOF or AVAIL).

- Quantitative reliability and availability specification has been used for many years in safety-critical systems but is uncommon for business critical systems.
- However, as more and more companies demand 24/7 service from their systems, it makes sense for them to be precise about their reliability and availability expectations.

Functional Reliability Requirements

8.4 Availability & Reliability

Checking requirements that identify checks to ensure that incorrect data is detected before it leads to a failure.

Recovery requirements that are geared to help the system recover after a failure has occurred.

Redundancy requirements that specify redundant features of the system to be included.

Process requirements for reliability which specify the development process to be used.

Functional Reliability Requirements

8.4 Availability & Reliability

Examples of functional reliability requirements

RR1: A pre-defined range for all operator inputs shall be defined and the system shall check that all operator inputs fall within this pre-defined range. (Checking)

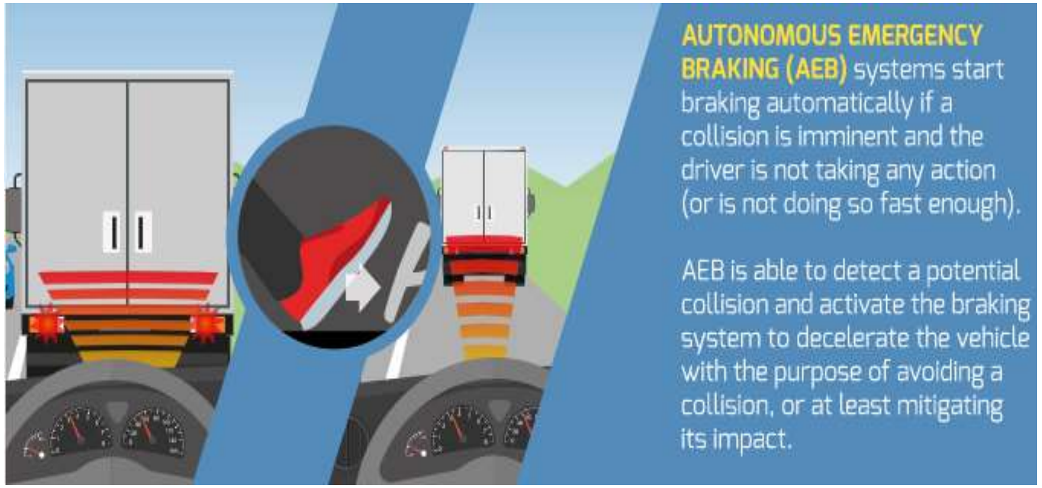
RR2: Copies of the patient database shall be maintained on two separate servers that are not housed in the same building. (Recovery, redundancy)

RR3: N-version programming shall be used to implement the braking control system. (Redundancy)

RR4: The system must be implemented in a safe subset of C and checked using static analysis. (Process)

8.5 Safety

- Safety reflects the system's ability to operate, normally or abnormally, without danger of causing human injury or death and without damage to the system's environment.
- Software is extensively used for checking and monitoring other safety-critical components in a system.
  - e.g., car braking system, aircraft engine components are monitored by software looking for early indications of component failure. This software is safety-critical because, if it fails, other components may fail and cause an accident.



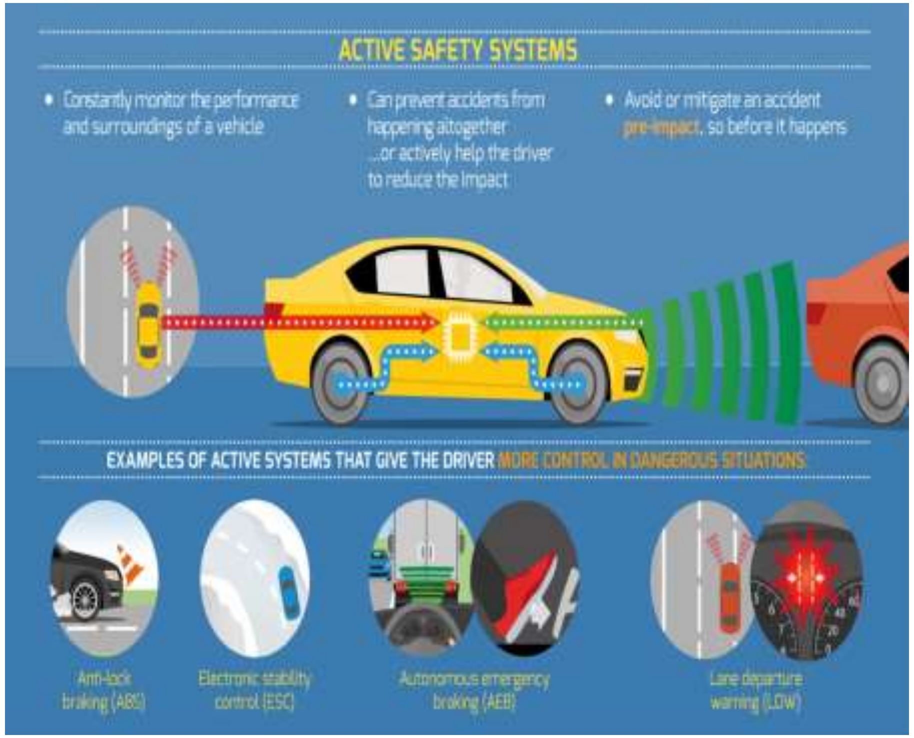
Safety vs. Reliability

8.5 Safety

Will do it accordingly when instructed

- Reliability is concerned with conformance to a given specification and delivery of service
- Safety is concerned with ensuring system cannot cause damage irrespective of whether or not it conforms to its specification.
  - System reliability is essential for safety but is not enough because a reliable system can be unsafe

A reliable system cannot guarantee safety



Reasons why a reliable system is not necessary safe:

System operators may be issuing the right command but at the wrong time

- **E.g**. a technician pressed a button that instructed the aircraft utility management software to raise the **undercarriage**. The software carried out the technician 's instruction perfectly.
- *What will happen ?*

(a system that perform according to specification can be unsafe)

- The system should have disallowed the command **unless the plane was in the air.**

Safety Critical Systems

8.5 Safety

- Systems where it is essential that system operation is always safe i.e. the system should never cause damage to people or the system's environment.
- Examples
  - Control and monitoring systems in aircraft
  - Process control systems in chemical manufacture
  - Automobile control systems such as braking and engine management systems

Safety Criticality

Primary safety-critical systems

- Embedded software systems whose failure can cause the associated hardware to fail and directly threaten people.
- Example: the insulin pump control system.

Secondary safety-critical systems

- Systems whose failure results in faults in other systems, which can then have safety consequences.

  Examples:
  - The Patient Information System is safety-critical as failure may lead to inappropriate treatment being prescribed.
  - Infrastructure control systems are also secondary safety-critical systems.

seatbelt reminder systems

Hazards

8.5 Safety

- Situations or events that can lead to an accident
  - Stuck valve in reactor control system
  - Incorrect computation by software in navigation system
  - Failure to detect possible allergy in medication prescribing system
- Hazards do not inevitably result in accidents – accident prevention actions can be taken.

Safety Achievement

8.5 Safety

3 ways to achieve safety for Safety critical system:

Hazard avoidance

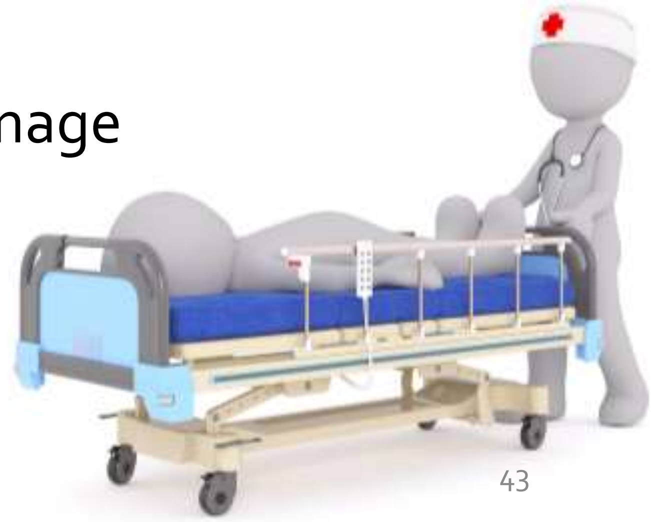
- The system is designed so that some classes of hazard simply car arise.

Hazard detection and removal

- The system is designed so that hazards are detected and remove before they result in an accident.

Damage limitation

- The system includes protection features that minimise the damage that may result from an accident.



3 Ways to Achieve Safety For Safety-Critical System:

- **Design system to avoid hazards.**
  - E.g. a cutting system requires the operator to **use 2 hands to press 2 separate buttons simultaneously**. This is to avoid the operator's hand being in the blade pathway.
- **Design system to detect hazards and removed it before they result in an accident.**
  - **E.g.** a chemical plant sys must be able to **detect excessive pressure** and open a relief valve to **reduce** these pressures b4 explosion occurs
- **Built-in protection features in a system** to **minimize** damage
  - E.g. includes an automatic fire extinguishers in an aircraft engine. If a fire occurs, it can be controlled b4 it poses a threat to the aircraft

Examples of Safety Requirements (SR)

Insulin-delivery system.

SR1: The system shall not deliver a single dose of insulin that is greater than a specified maximum dose for a system user.

SR2: The system shall not deliver a daily cumulative dose of insulin that is greater than a specified maximum daily dose for a system user.

SR3: The system shall include a hardware diagnostic facility that shall be executed at least four times per hour.

SR4: The system shall include an exception handler for all of the exceptions.

SR5: The audible alarm shall be sounded when any hardware or software anomaly is discovered and a diagnostic message shall be displayed.

SR6: In the event of an alarm, insulin delivery shall be suspended until the user has reset the system and cleared the alarm.

8.6 Security

- The security of a system is a system property that reflects the system's ability to protect itself from accidental or deliberate external attack.
- Security is essential as most systems are networked so that external access to the system through the Internet is possible.
- Security is an essential pre-requisite for availability, reliability and safety.

Security Dimensions

8.6 Security

Impact of breach of security

Confidentiality

- confidential information is exposed to people who are not authorised to read or use that information

Integrity

Information in a system may be damaged or corrupted making it unusable or unreliable.

Availability

System become unavailable

Security Levels

8.6 Security

Infrastructure security

concerned with maintaining the security of all systems and networks that provide an infrastructure and a set of shared services to the organization.

Application security

concerned with the security of individual application systems or related groups of systems.

Operational security

is concerned with the secure operation and use of the organization's systems.

System layers where security may be compromised

Application
Reusable components and libraries
Middleware
Database management
Generic, shared applications (browsers, e--mail, etc)
Operating System
Network / Computer hardware

Application vs. Infrastructure Security

- Application security is a software engineering problem where the system is designed to resist attacks.
- Infrastructure security is a systems management problem where the infrastructure is configured to resist attacks.
- The focus of this chapter is application security rather than infrastructure security.

System Security Management

8.6 Security

User and permission management

Adding and removing users from the system and setting up appropriate permissions for users

Software deployment and maintenance

Installing application software and middleware and configuring these systems so that vulnerabilities are avoided.

Attack monitoring, detection and recovery

Monitoring the system for unauthorized access, design strategies for resisting attacks and develop backup and recovery strategies.

Operational Security

8.6 Security

- Primarily a human and social issue
- Concerned with ensuring the people do not take actions that may compromise system security
  - E.g. Tell others passwords, leave computers logged on
- Users sometimes take insecure actions to make it easier for them to do their jobs
- There is therefore a trade-off between system security and system effectiveness.

Security terminology

Term	Definition
Asset	Something of value which has to be protected; may be the software system itself or data used by that system.
Attack	An exploitation of a system’s vulnerability. Generally, this is from outside the system and is a deliberate attempt to cause some damage.
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
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.
Threat	Circumstances that have potential to cause loss or harm. You can think of these as a system vulnerability that is subjected to an attack.
Vulnerability	A weakness in a computer-based system that may be exploited to cause loss or harm.

Examples of security terminology in a Patient Information System

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 a patient. 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.

Security vs. Safety

8.6 Security

Safety	Security
Safety problems are accidental – the software is not operating in a hostile environment.	In security, you must assume that attackers have knowledge of system weaknesses.
When safety failures occur, you can look for the root cause or weakness that led to the failure.	When failure results from a deliberate attack, the attacker may conceal the cause of the failure.
Shutting down a system can avoid a safety-related failure.	Causing a shut down may be the aim of an attack.
Safety-related events are not generated from an intelligent adversary.	An attacker can probe defenses over time to discover weaknesses.

Types of Security Requirement

8.6 Security

- Identification requirements
- Authentication requirements
- Authorization requirements
- Integrity requirements
- Intrusion detection requirements
- Non-repudiation requirements
- Privacy requirements
- Security auditing requirements
- System maintenance security requirements

Security Requirement

8.6 Security

Security requirements for Patient Information System

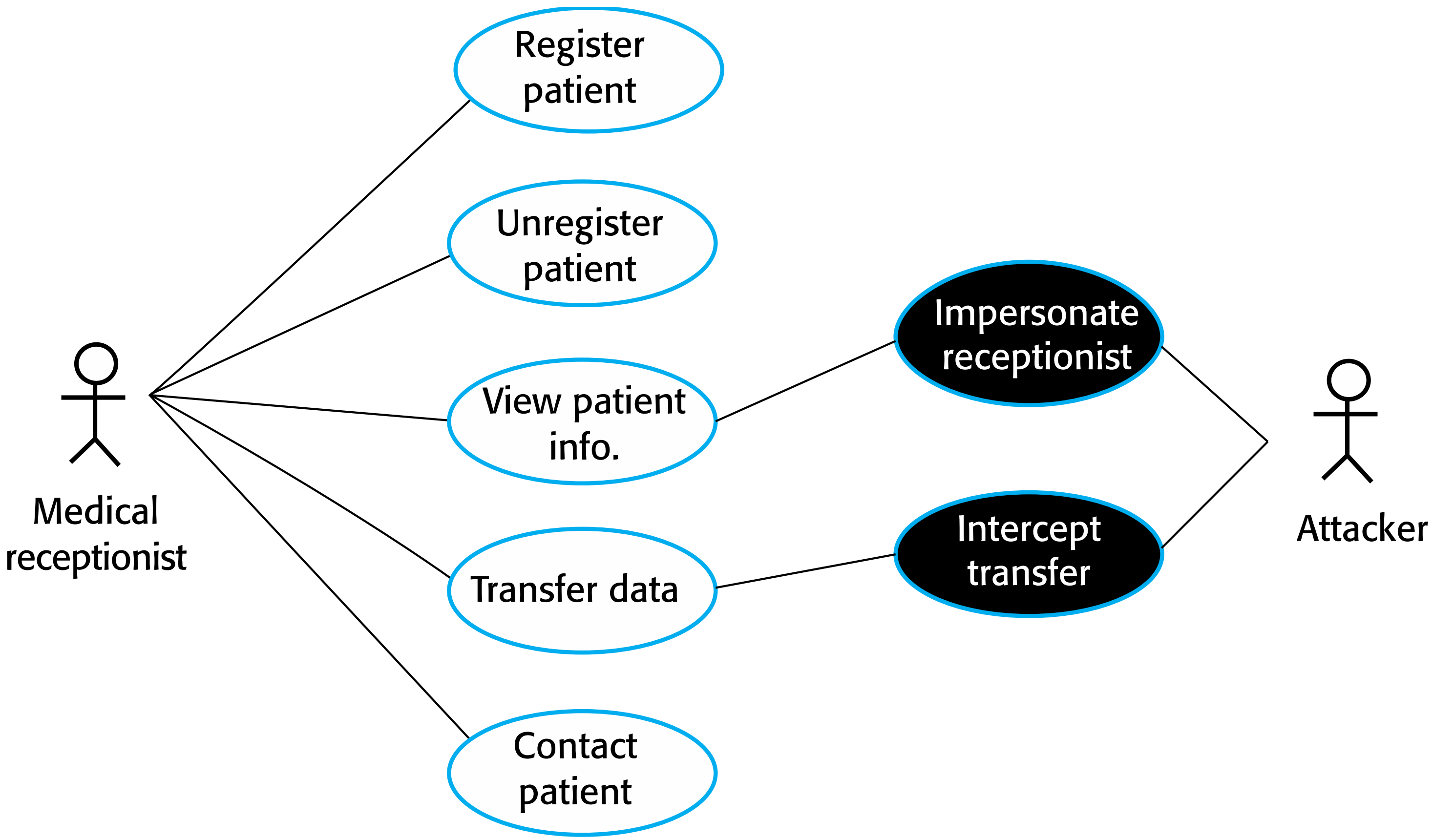
- Patient information shall be downloaded at the start of a clinic session to a secure area on the system client that is used by clinical staff.
- All patient information on the system client shall be encrypted.
- Patient information shall be uploaded to the database after a clinic session has finished and deleted from the client computer.
- A log on a separate computer from the database server must be maintained of all changes made to the system database.

Misuse Cases

8.6 Security

- Misuse cases are instances of threats to a system
- Interception threats
  - Attacker gains access to an asset
- Interruption threats
  - Attacker makes part of a system unavailable
- Modification threats
  - A system asset if tampered with
- Fabrication threats
  - False information is added to a system

Misuse cases for Patient Information System



Use case – Transfer data

Patient IS: Transfer data	
Actors	Medical receptionist, Patient records system (PRS)
Description	A receptionist may transfer data from the Patient IS to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient's diagnosis and treatment.
Data	Patient's personal information, treatment summary.
Stimulus	User command issued by medical receptionist.
Response	Confirmation that PRS has been updated.
Comments	The receptionist must have appropriate security permissions to access the patient information and the PRS.

Misuse case: Intercept transfer

Patient IS: Intercept transfer (Misuse case)	
Actors	Medical receptionist, Patient records system (PRS), Attacker
Description	A receptionist transfers data from his or her PC to the Patient IS on the server. An attacker intercepts the data transfer and takes a copy of that data.
Data (assets)	Patient’s personal information, treatment summary
Attacks	A network monitor is added to the system and packets from the receptionist to the server are intercepted. A spoof server is set up between the receptionist and the database server so that receptionist believes they are interacting with the real system.
Mitigations	All networking equipment must be maintained in a locked room. Engineers accessing the equipment must be accredited. All data transfers between the client and server must be encrypted. Certificate-based client-server communication must be used
Requirements	All communications between the client and the server must use the Secure Socket Layer (SSL). The https protocol uses certificate based authentication and encryption.

Secure Systems Design

8.6 Security

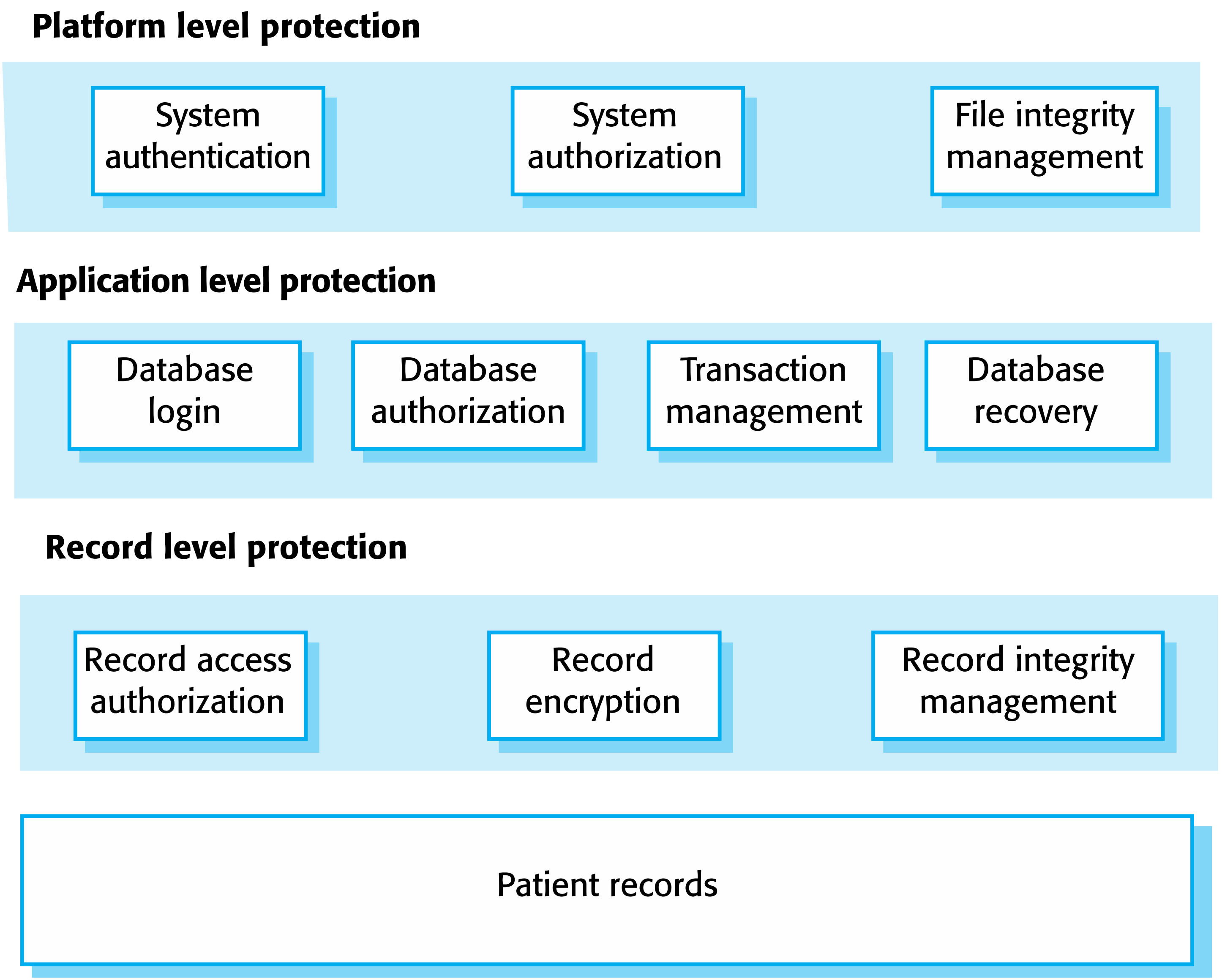
- Security should be designed into a system – it is very difficult to make an insecure system secure after it has been designed or implemented
- Adding security features to a system to enhance its security affects other attributes of the system
  - Performance
    - Additional security checks slow down a system so its response time or throughput may be affected
  - Usability
    - Security measures may require users to remember information or require additional interactions to complete a transaction. This makes the system less usable and can frustrate system users.

Security Requirements

8.6 Security

- A password checker shall be made available and shall be run daily. Weak passwords shall be reported to system administrators.
- Access to the system shall only be allowed by approved client computers.
- All client computers shall have a single, approved web browser installed by system administrators.

A layered protection architecture



How to Improve Security?

- **Avoid vulnerability** :
  - Design system so that vulnerabilities do not occur
  - **E.g.** if a system is **not connected to an external public network** then there is no possibility of an attack from members of the public

    → make a choice (convenience vs security)
- **Implement attack detection and elimination**:
  - Design system with the ability to detect vulnerability and remove them before they result in an exposure
  - **E.g. use virus checkers to find and remove viruses** before they infect a system
- **Limit exposure**:
  - Design system so that adverse consequences of a successful attack are minimised.
  - **E.g. Restore damage IS with backup** (comp. must hv a backup policy)

Summary

8.1 Critical Systems
8.2 Dependability
8.3 Achieving Dependability
8.4 Availability & Reliability
8.5 Safety
8.6 Security