CSE344
Software Engineering
(SWE)

Week – 5
Critical Systems
&
Their Specification

Critical Systems

- *Safety-critical systems*
  - Failure results in *loss of life, injury or damage to the environment*;
  - Chemical plant protection system;
- *Mission-critical systems*
  - Failure results in *failure of some goal-directed activity*;
  - Spacecraft navigation system;
- *Business-critical systems*
  - Failure results in *high economic losses*;
  - Customer accounting system in a bank;

What is the most important system property for regular SW systems?

- From the FAQ of the discussion in week 1:
- The SW process cost distribution is, roughly, as follows:
  - ***60% ↔ development costs***,
  - ***40% ↔ testing costs***.
- For *custom SW*, ***evolution*** *(or* ***maintenance****)* ***costs*** *often exceed development costs*.
-

Hence, the most essential SW system property for regular systems is SW maintainability !!!

System dependability

- For critical systems, usually the most important system property is the dependability of the system.
- The *dependability of a system* reflects the *extent of the user's confidence* that it will operate as users expect and that it will not 'fail' in normal use.
- *Usefulness (kullanışlılık)* and *trustworthiness (güvenilirlik)* are *not the same* thing. *A system does not have to be trusted to be useful.*

Importance of dependability

- *Undependable* (i.e., *unreliable*, *unsafe and/or insecure*) SW systems
  - may be *rejected by their users*.
  - may cause *information loss with a high recovery cost*.
- The costs of system failure may be very high.

Development methods for critical systems

- *The costs of critical system failure are so high that development methods may be used that are not cost-effective for other types of system.*
- Examples of development methods
  - ***Formal methods of software development***
  - Static analysis
  - External quality assurance

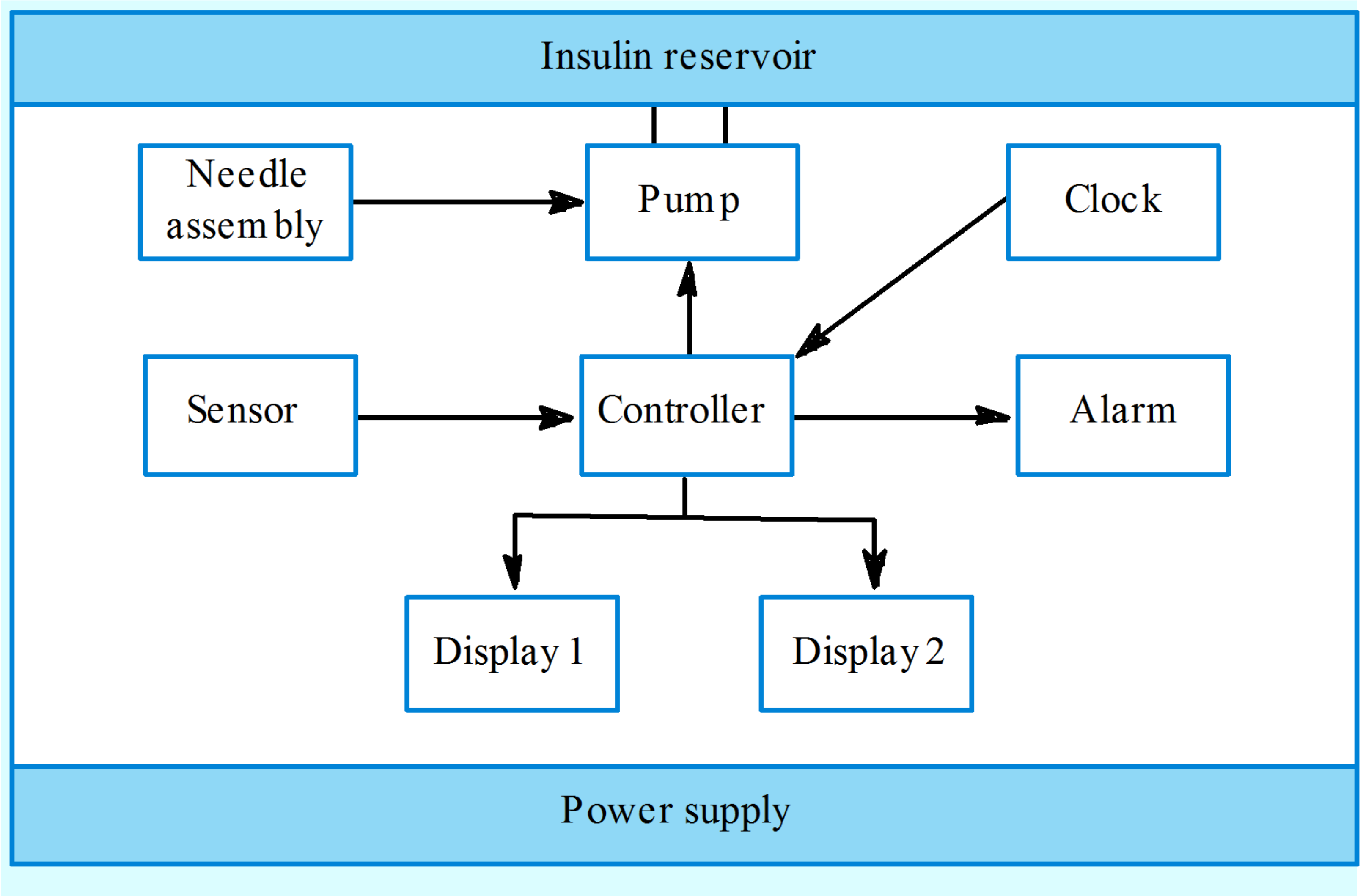
Failure types in socio-technical critical systems

- *Hardware failure*
  - Hardware fails because
    - of design and manufacturing errors or
    - components have reached the end of their natural life.
- *Software failure*
  - Software fails due to errors in its specification, design or implementation.
- *Operational failure*
  - Human operators make mistakes. Now perhaps the largest single cause of system failures.

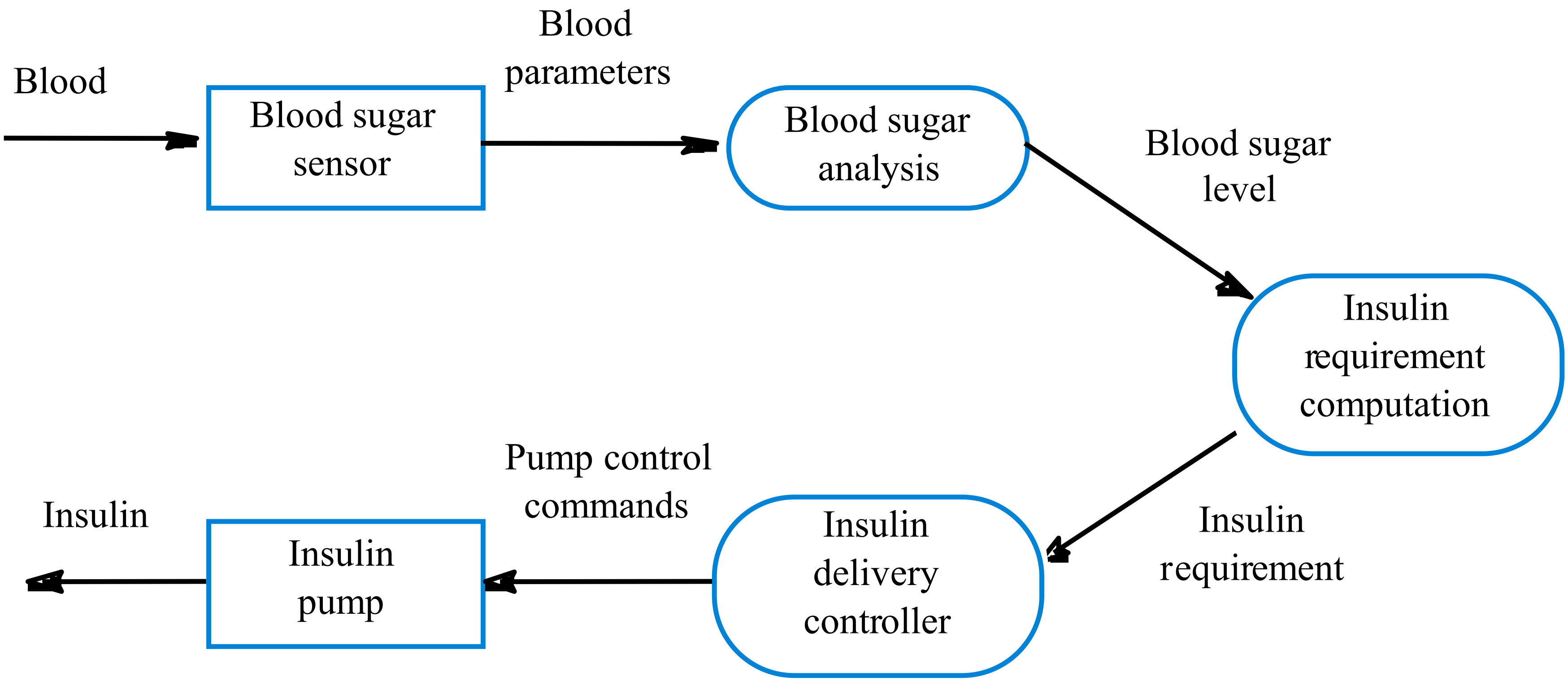
An example to critical systems: a SW-controlled insulin pump

- Used by *diabetics*
- *Simulates the function of the pancreas*
  - Manufactures *insulin*,
    - an essential hormone that *metabolises blood glucose*.
  - *Measures blood glucose* (sugar) using a micro-sensor and
  - *Computes the insulin dose required* to *metabolise* the glucose.

Insulin pump organisation



Insulin pump data-flow



Dependability requirements

- ...shall be *available* to *deliver insulin when required* to do so (*availability requirement*)
- ...shall *deliver the correct amount of insulin* to counteract (neutralize) the current level of blood sugar (*reliability requirement*).
- ...shall *never deliver excessive doses of insulin* as this is potentially *life threatening* (*safety requirement*).

Dependability

- *Dependability = Trustworthiness*.
- A *dependable* system is one *trusted by its users*.
- *Principal dimensions of dependability* are:
  - *Availability*
  - *Reliability*
  - *Safety*
  - *Security*

Principal dimensions of dependability

- *The (prob)ability of the system to ...*
  - *... deliver services* ***when requested*** *(Availability)*
  - *... deliver services* ***as specified*** *(Reliability)*
  - *... operate without catastrophic failure (Safety)*
  - *... protect itself against accidental or deliberate intrusion (Security)*

Other dependability properties

- *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*;
- *Survivability*
  - Reflects the *extent to which the system can deliver services whilst under hostile attack*;
- *Error tolerance*
  - Reflects the *extent to which user input errors can be avoided and tolerated*.

Maintainability

- concerned with the ease of
  - … *repairing the system* after a failure has been discovered or
  - … *changing the system* to include new features;
- very important for critical systems as faults are often introduced into a system because of maintenance problems
- different than other dimensions of dependability because it is a *static and not a dynamic system attribute*.

Survivability

- The ability of a system *to continue to deliver its services to users under attack*
- This is an *increasingly important attribute* for distributed systems whose security can be compromised
- Survivability subsumes the notion of ***resilience*** (flexibility, *fault tolerance*) - the ability of a system to *continue in operation in spite of component failures*

Dependability vs performance

- Untrustworthy systems may be rejected by their users
- System failure costs may be very high
- It is very difficult to tune systems to make them more dependable
- It may be possible to compensate for poor performance
- Untrustworthy systems may cause loss of valuable information

Dependability costs

- Dependability costs tend to increase exponentially as increasing levels of dependability are required
- Two reasons:
  - The *use of more expensive development techniques and hardware* required to achieve higher levels of dependability
  - The *increased testing and system validation* required to convince the system client that the required levels of dependability have been achieved

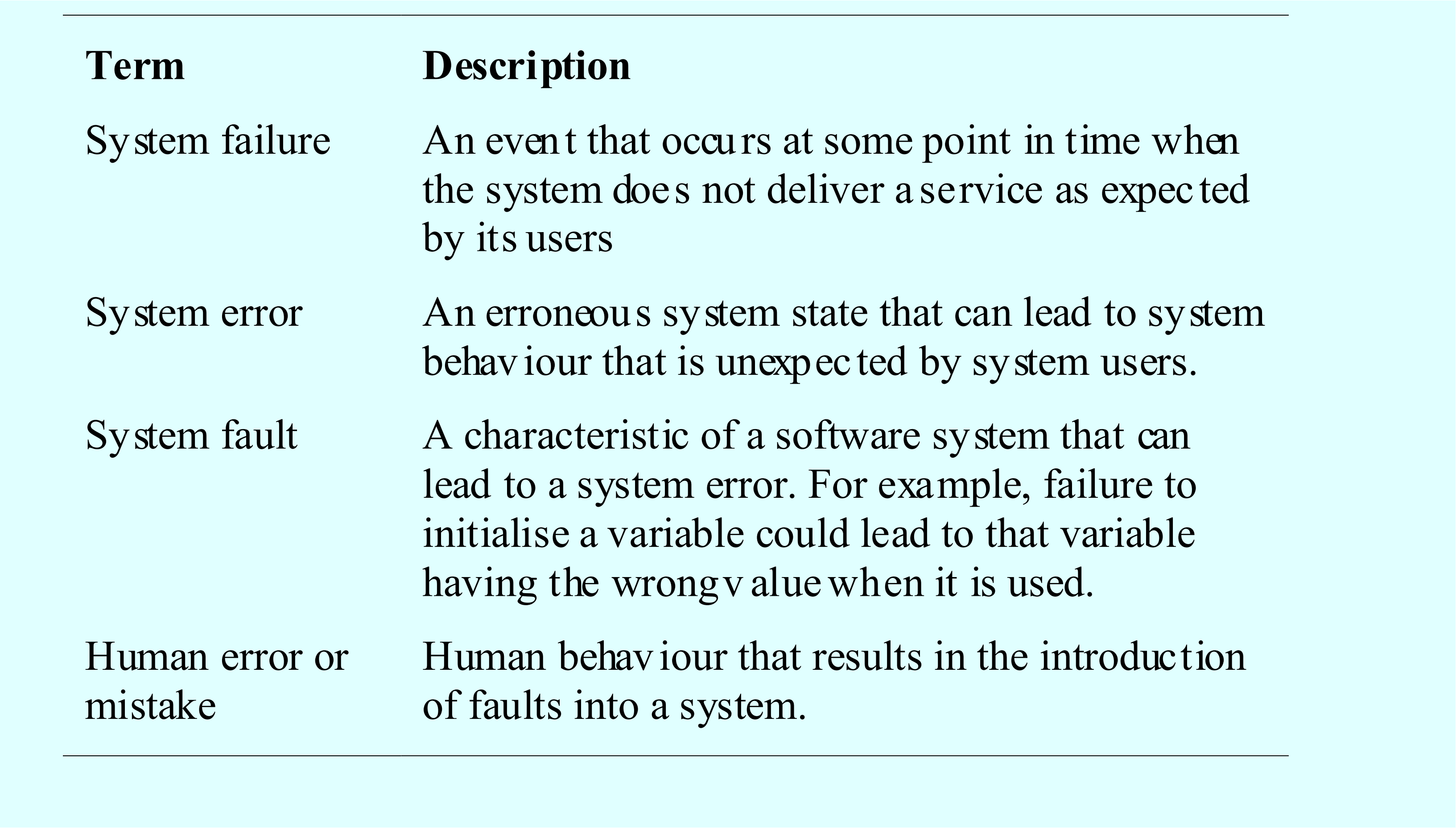
Dependability economics

- *Tradeoff*: Achievement of dependability requirements costs extremely high ⇒ *more cost-effective to accept untrustworthy systems* and pay for failure costs?
- *Social and political factors*: A bad reputation for products may lead to the loss of future business.
- *System type*: for *business systems* in particular, *modest levels of dependability* OK; for *medical systems highly dependable systems* are crucial.

Availability and reliability: expressible as stochastic (probabilistic) quantity

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

Reliability terminology



Term	Description
System failure	An event that occurs at some point in time when the system does not deliver a service as expected by its users
System error	An erroneous system state that can lead to system behaviour that is unexpected by system users.
System fault	A characteristic of a software system that can lead to a system error. For example, failure to initialise a variable could lead to that variable having the wrong value when it is used.
Human error or mistake	Human behaviour that results in the introduction of faults into a system.

Faults and failures

- *Failures* are usually a result of *system errors* that are derived from *faults* in the system
- However, *faults* do not necessarily result in *system errors*
  - The faulty system state may be *transient* (temporary) and ‘corrected’ before an error arises
- *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

Perceptions of reliability

- The formal definition of reliability does not always reflect the user's perception of a system's reliability
    - The assumptions that are made about the environment where a system will be used may be incorrect
        - Usage of a system in an office environment is likely to be quite different from usage of the same system in a university environment
    - The consequences of system failures affects the perception of reliability
        - Unreliable windscreen wipers in a car may be irrelevant in a dry climate
        - Failures that have serious consequences (such as an engine breakdown in a car) are given greater weight by users than failures that are only inconvenient

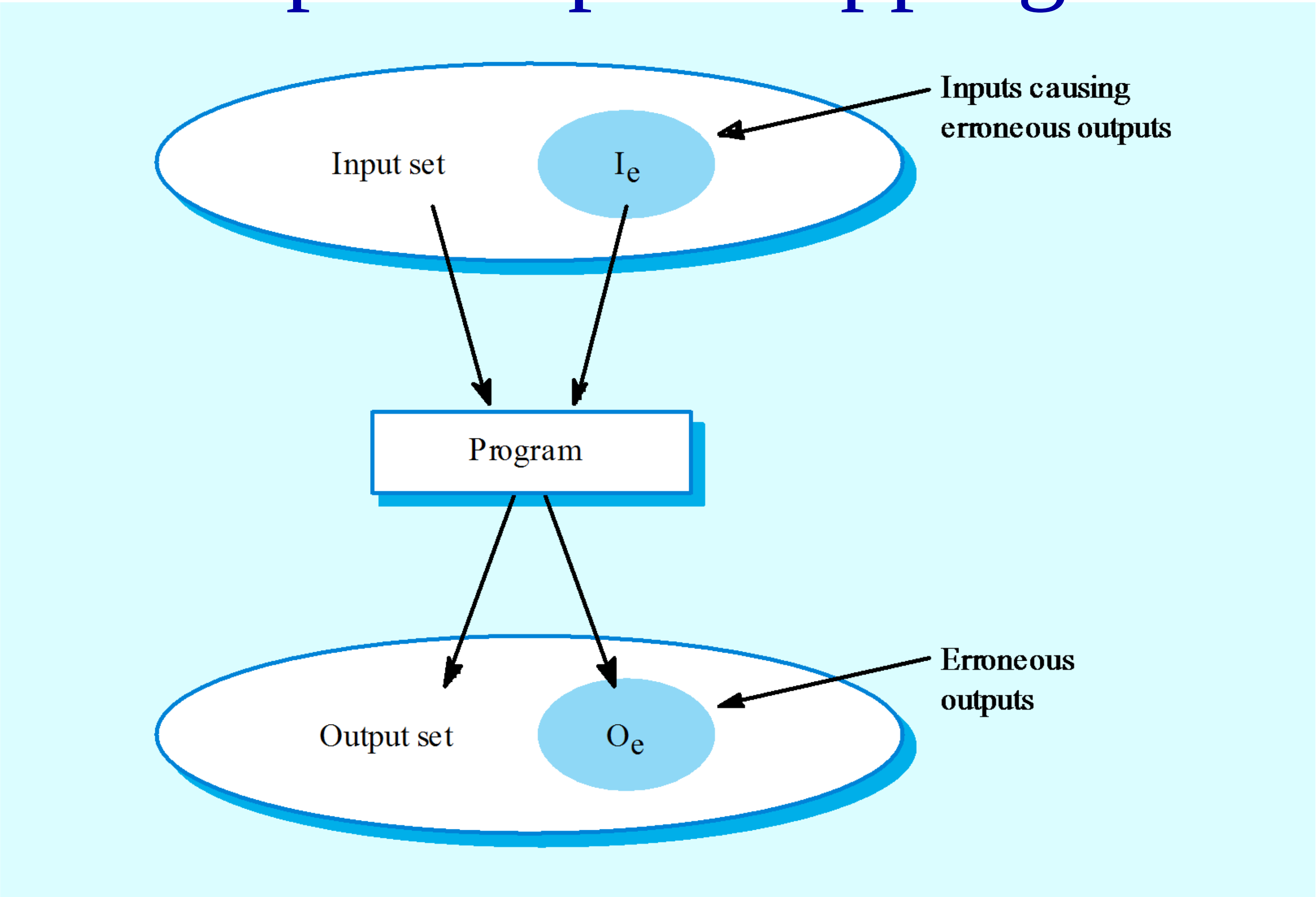
Reliability achievement

- ***Fault avoidance***
  - Development technique are used that either *minimise the possibility of mistakes* or trap mistakes before they result in the introduction of system faults
- ***Fault detection and removal***
  - Verification and validation techniques that increase the probability of *detecting and correcting errors before the system goes into service* are used
- ***Fault tolerance***
  - Run-time techniques are used to ensure that *system faults do not result in system errors and/or that system errors do not lead to system failures*

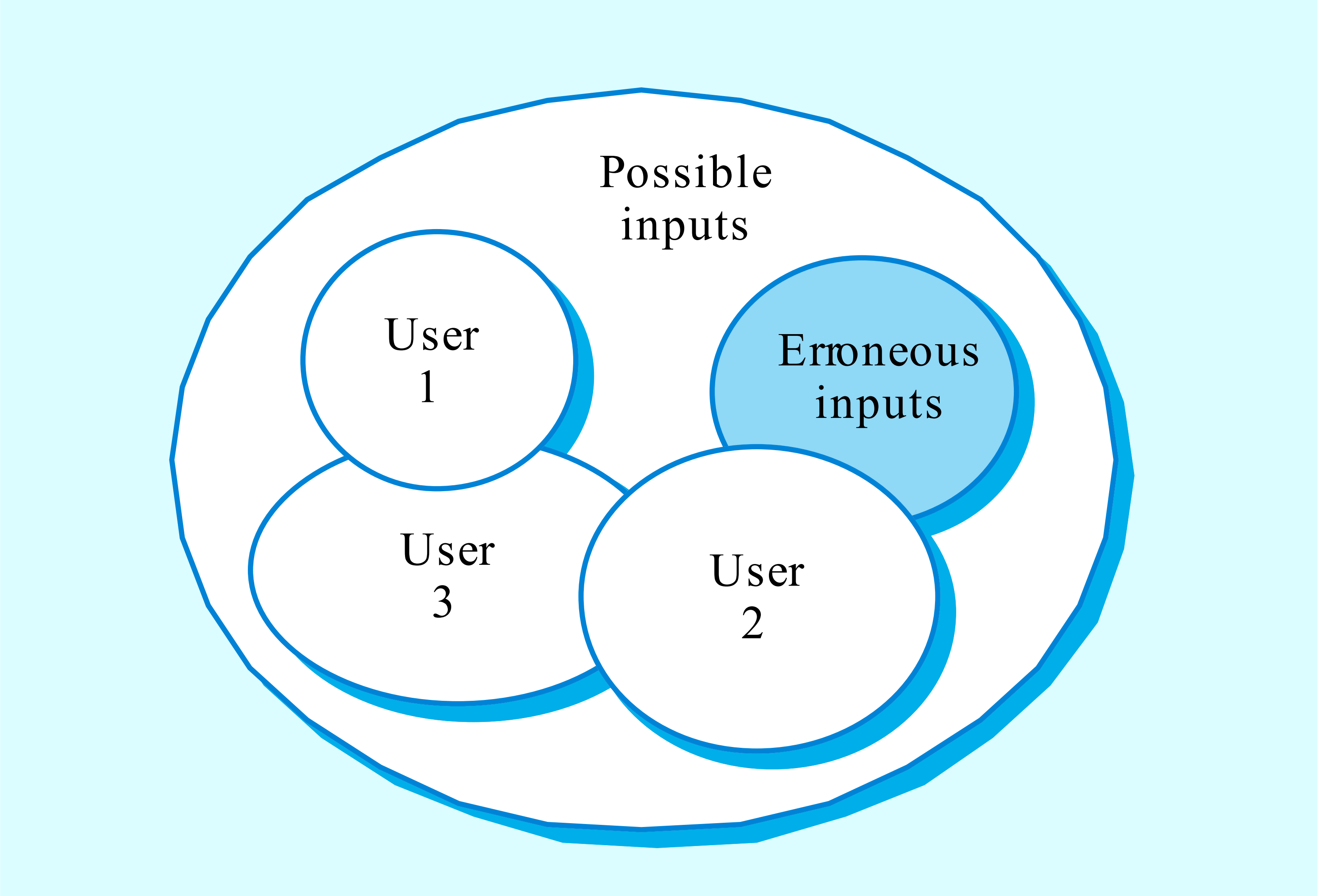
Reliability modeling

- You can model a system as an *input-output mapping* where some inputs will result in erroneous outputs
- The reliability of the system depends on the *probability that a particular input will lie in the set of inputs that cause erroneous outputs*
- Different people will use the system in different ways so this probability is not a static system attribute but depends on the system's environment

Input/output mapping



Reliability perception



Reliability improvement

- Removing X% of the faults in a system will not necessarily improve the reliability by X%. A study at IBM showed that removing 60% of product defects resulted in a 3% improvement in reliability
- Program defects may be in rarely executed sections of the code so may never be encountered by users. Removing these does not affect the perceived reliability
- A program with known faults may therefore still be seen as reliable by its users

Safety

- Safety is a property of a system that *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*
- It is *increasingly important* to consider software safety *as more and more devices incorporate software-based control systems*
- Safety requirements are *exclusive requirements* i.e. they *exclude undesirable situations* rather than specify required system services

Safety criticality

- *Primary safety-critical systems*
  - Embedded software systems whose failure can cause the associated hardware to fail and directly threaten people.
- *Secondary safety-critical systems*
  - Systems whose failure results in faults in other systems which can threaten people
- Discussion here focuses on primary safety-critical systems
  - Secondary safety-critical systems can only be considered on a one-off (every-once-in-a-while) basis

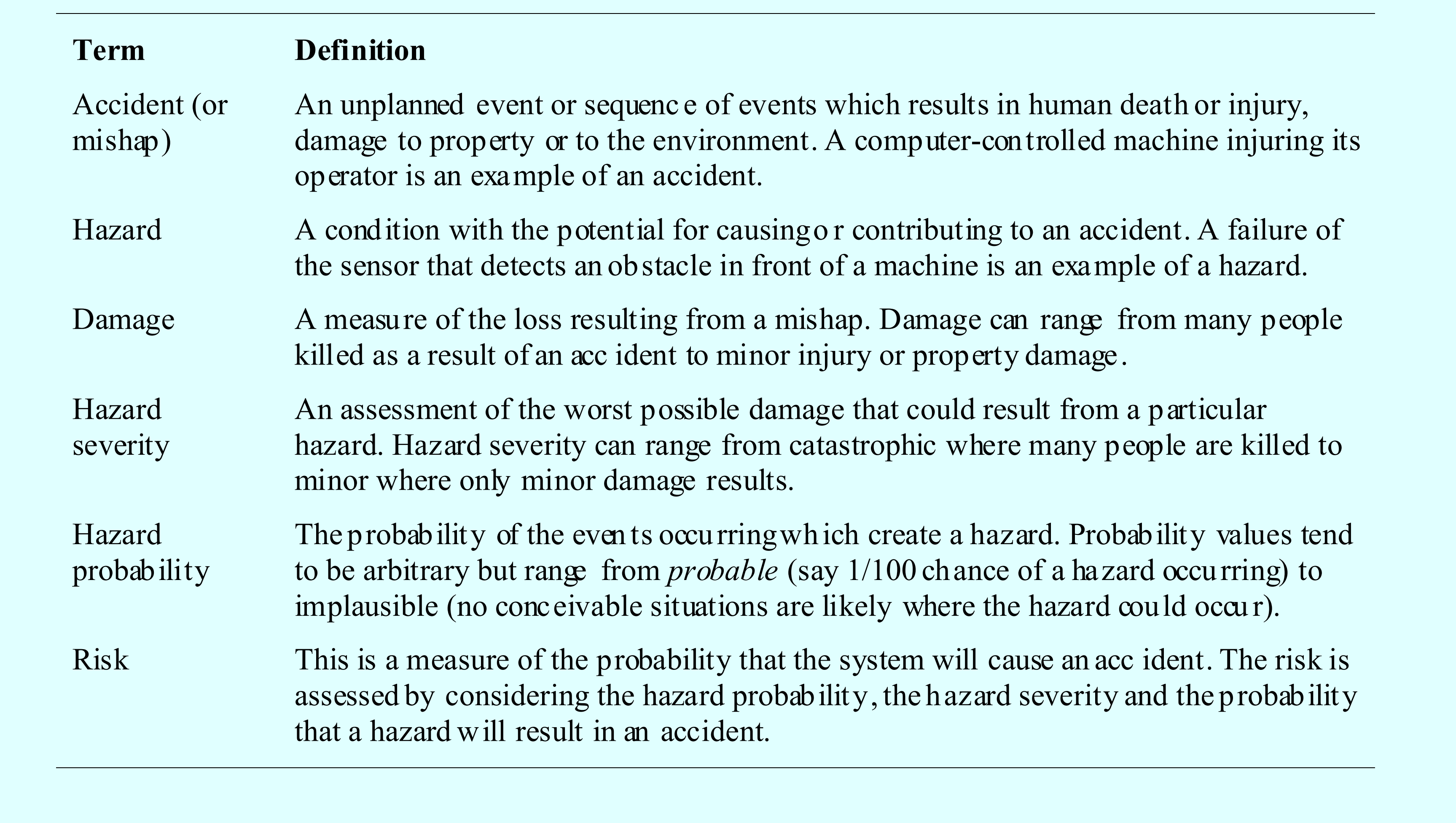
Safety and reliability

- Safety and reliability are related but distinct
  - In general, reliability and availability are necessary but not sufficient conditions for system safety
- *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*

Unsafe reliable systems

- *Specification errors*
  - If the system *specification is incorrect* then the system can *behave as specified but still cause an accident*
- Hardware failures generating *spurious inputs*
  - Hard to anticipate in the specification
- Context-sensitive commands i.e. issuing the right command at the wrong time
  - Often the result of operator error

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. A computer-controlled machine injuring its operator 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 detects an obstacle in front of a machine is an example of a hazard.
Damage	A measure of the loss resulting from a mishap. Damage can range from many people killed as a result of an accident to minor injury or property damage.
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.
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 where the hazard could occur).
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 a hazard will result in an accident.

Safety achievement

- *Hazard avoidance*
  - The system is designed so that some classes of hazard simply cannot arise.
- *Hazard detection and removal*
  - The system is designed so that hazards are detected and removed before they result in an accident
- *Damage limitation*
  - The system includes protection features that minimise the damage that may result from an accident

Normal accidents

- Accidents in complex systems rarely caused by single cause as these systems are *resilient* (i.e., durable, resistant) to a single point of failure
  - Designing systems so that *a single point of failure does not cause an accident* is a fundamental principle of safe systems design
- Almost all accidents are a result of *combinations of malfunctions*
- Anticipating all problem combinations, especially, in software controlled systems is next to impossible so *achieving complete safety is impossible*

Quiz 2 – Part 2 - Week 4

Send answers to **cse3044marmara@gmail.com**

(20pts + 10pts + 10pts)

3. *In what different levels may the relability be achieved?*
4. *Reliability may be achieved in only one of these levels. Reliability achievement may not occur at more than one of these levels. (True/False)*
5. *Removing 3% of the faults in a system may improve the reliability by 60%. (True/False)*

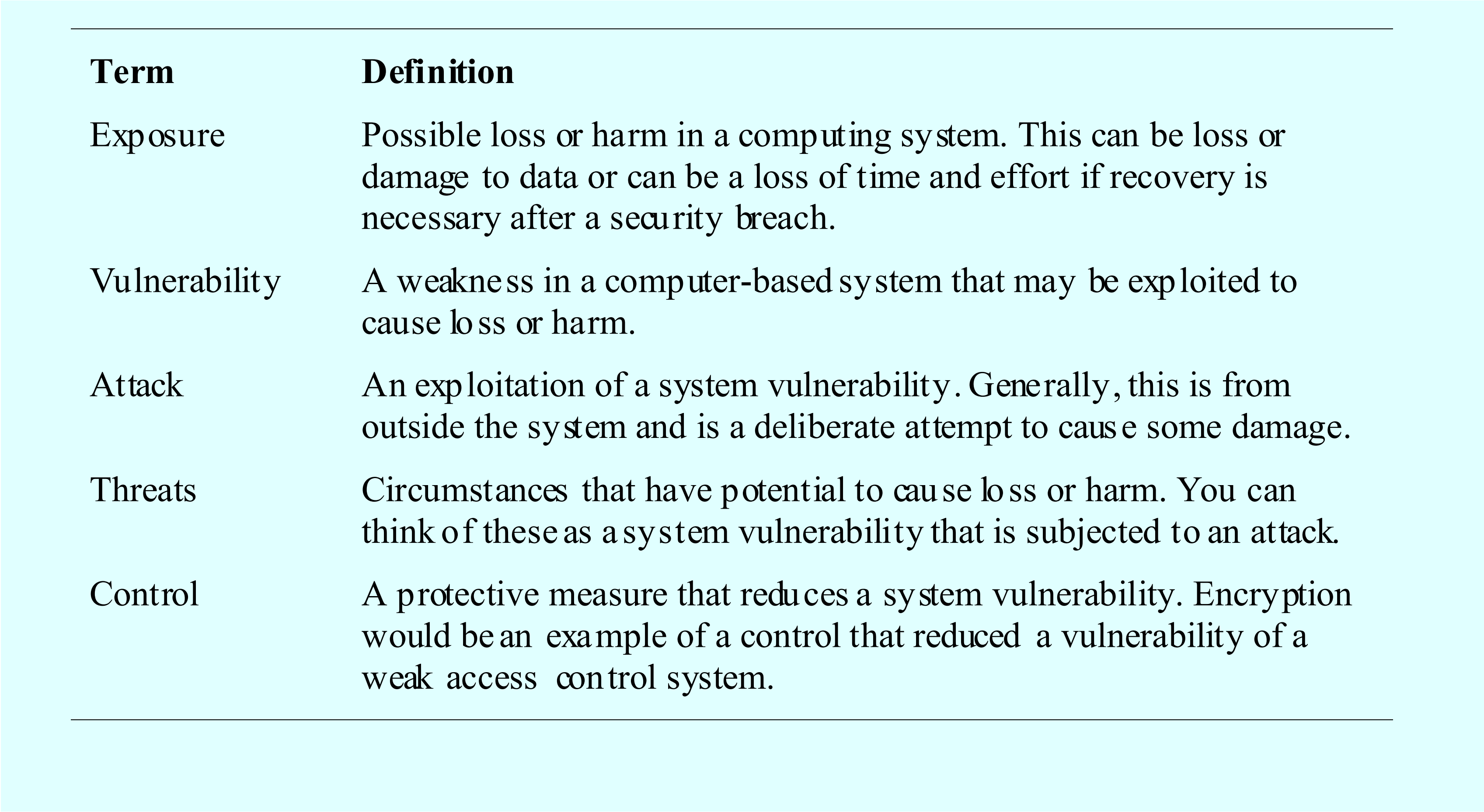
Security

- ... a system property that *reflects the system's ability to protect itself from accidental or deliberate external attack*
- Security is becoming *increasingly important* as 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

Fundamental security

- *If a system* is a networked system and *is insecure then statements about its reliability and its safety are unreliable*.
- These statements depend on the executing system and the developed system being the same. However, intrusion can change the executing system and/or its data.
- *Therefore, the reliability and safety assurance is no longer valid.*

Security terminology



Term	Definition
Exposure	Possible loss or harm in 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 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 vulnerability. Encryption would be an example of a control that reduced a vulnerability of a weak access control system.

Damage from insecurity

- *Denial of service*
  - The system is forced into a state where normal services are unavailable or where service provision is significantly degraded
- *Corruption of programs or data*
  - The programs or data in the system may be modified in an unauthorised way
- *Disclosure of confidential information*
  - Information that is managed by the system may be exposed to people who are not authorised to read or use that information

Security assurance

- *Vulnerability avoidance*
  - The system is designed so that vulnerabilities do not occur. For example, if there is no external network connection then external attack is impossible
- *Attack detection and elimination*
  - The system is designed so that attacks on vulnerabilities are detected and neutralised before they result in an exposure. For example, virus checkers find and remove viruses before they infect a system
- *Exposure limitation*
  - The system is designed so that the adverse consequences of a successful attack are minimised. For example, a backup policy allows damaged information to be restored

Dependability requirements

- *Functional requirements* to define error checking and recovery facilities and protection against system failures.
- *Non-functional requirements* defining the required reliability and availability of the system.
- *Excluding requirements* that define states and conditions that must not arise.

Quiz 2 – Part 3 - Week 4 (by 10:31)

Send answers to **cse3044marmara@gmail.com** **(25pts + 15pts)**

6. *How can an unsafe system be reliable?*

7. *Which SW process model is the most appropriate model for critical SW development?*

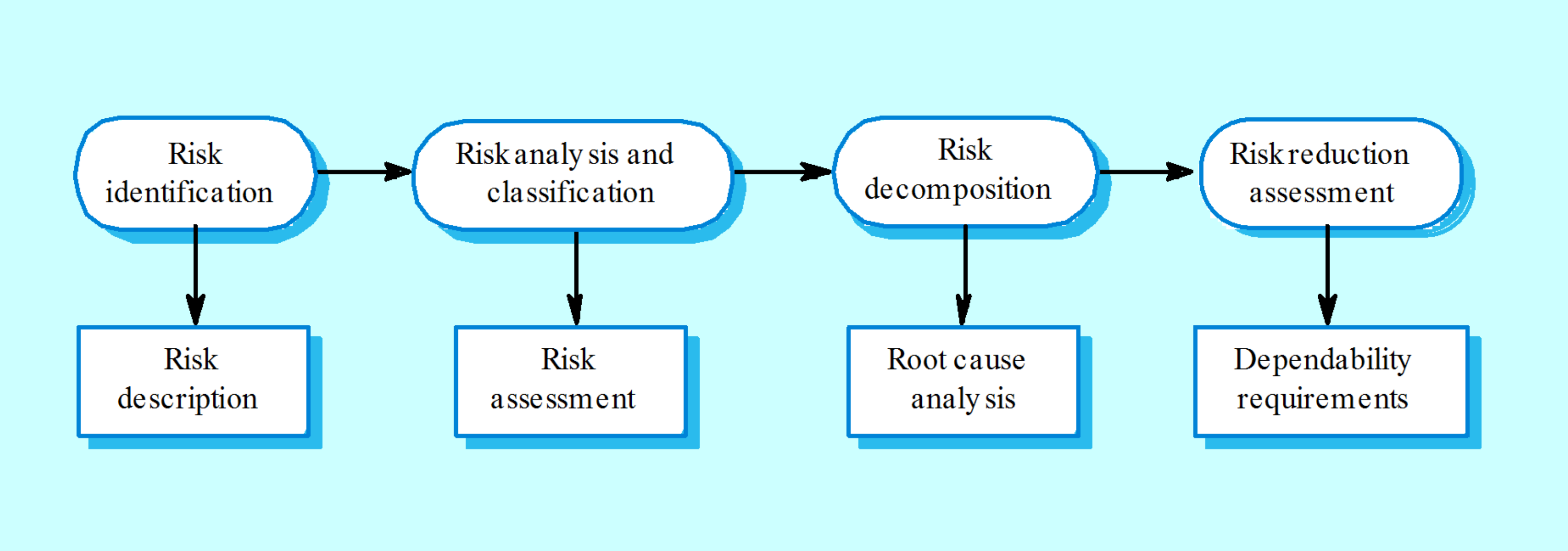
Critical systems specification...

- ...***risk-driven***.
- ...widely used in safety and security-critical systems.
- Aim of specification: to *understand the risks* (safety, security, etc.) faced by the system and to *define requirements reducing these risks*.

Stages of risk-based analysis

- *Risk identification*
  - Identify potential risks that may arise.
- *Risk analysis and classification*
  - Assess the seriousness of each risk.
- *Risk decomposition*
  - Decompose risks to discover their potential *root causes*.
- *Risk reduction assessment*
  - Define how each risk must be eliminated or reduced when the system is designed.

Risk-driven specification



Risk identification

- Risks for
  - Safety-critical systems: *hazards that can lead to accidents*.
  - Security-critical systems: *potential attacks on the system*.
- First: *identify risk classes*, and
- Second: *position risks in these classes*
  - Service failure;
  - Electrical risks;
  - …

Insulin pump risks

- Insulin overdose (service failure).
- Insulin underdose (service failure).
- Power failure due to exhausted battery (electrical).
- Electrical interference with other medical equipment (electrical).
- Poor sensor and actuator contact (physical).
- Parts of machine break off in body (physical).
- Infection caused by introduction of machine (biological).
- Allergic reaction to materials or insulin (biological).

Risk analysis and classification

- ...concerned with understanding the likelihood of possible risks and the potential consequences in case of any accident or incident.
- Risks categories:
  - ***Intolerable***. Must *never arise* or result in an accident
  - ***As low as reasonably practical(ALARP).*** Must *minimise the possibility of risk* given cost and schedule constraints
  - ***Acceptable.*** The *consequences of the risk are acceptable* and no extra costs should be incurred to reduce hazard probability

Risk assessment

- Estimate
  - risk probability and
  - risk severity.
- Normally not possible to do this precisely so relative values are used such as ‘unlikely’, ‘rare’, ‘very high’, etc.
- The aim must be to exclude risks that are likely to arise or that have high severity.

Risk assessment - insulin pump

Identified hazard	Hazard probability	Hazard severity	Estimated risk	Acceptability
1. Insulin overdose	Medium	High	High	Intolerable
2. Insulin underdose	Medium	Low	Low	Acceptable
3. Power failure	High	Low	Low	Acceptable
4. Machine incorrectly fitted	High	High	High	Intolerable
5. Machine breaks in patient	Low	High	Medium	ALARP
6. Machine causes infection	Medium	Medium	Medium	ALARP
7. Electrical interference	Low	High	Medium	ALARP
8. Allergic reaction	Low	Low	Low	Acceptable

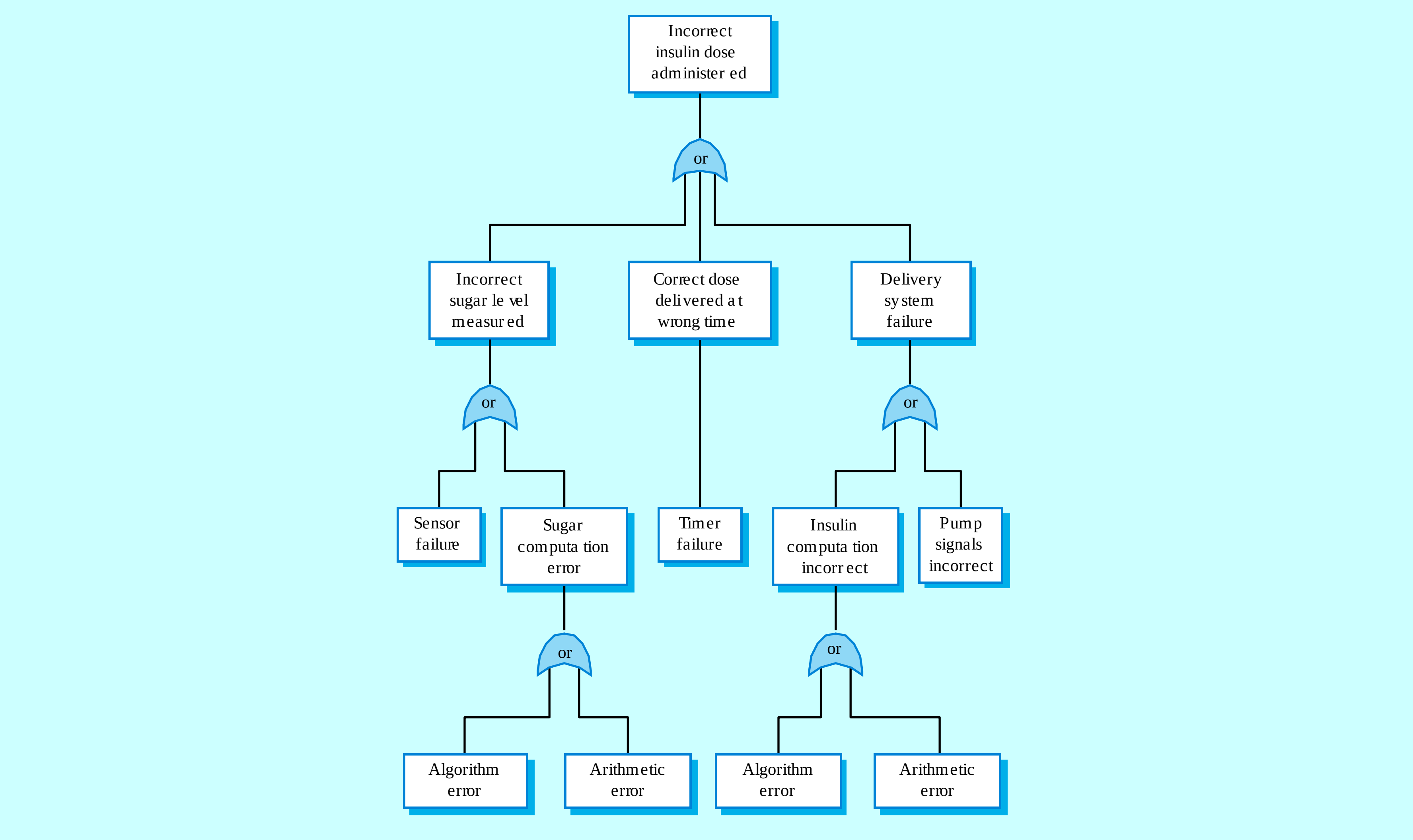
Risk decomposition...

- ...concerned with discovering the *root causes* of risks in a particular system.
- Two techniques...
  - *inductive, bottom-up techniques*.
    - start with a proposed *system failure*, and
    - assess the hazards that could arise from that failure;
  - *deductive, top-down techniques.*
    - start with a hazard, and
    - deduce what the causes of this could be.

Fault-tree analysis

- A *deductive top-down* technique.
- Put the risk or *hazard at the root of the tree* and *identify the system states that could lead to that hazard*.
- Where appropriate, *link these with 'and' or 'or'* conditions.
- Goal: *minimise the number of single causes of system failure*.

Insulin pump fault tree



Risk reduction assessment

- Aim:
  - to identify dependability requirements that
    - specify how the risks should be managed, and
    - ensure that accidents/incidents do not arise.
- Risk reduction strategies
  - *Risk avoidance*;
  - *Risk detection and removal*;
  - *Damage limitation*.

Strategy use

- Normally, in critical systems, a *mix of risk reduction strategies* are used.
- In a chemical plant control system, the system will include sensors to detect and correct excess pressure in the reactor (*Risk detection and removal*).
- However, it will also include an independent protection system that opens a relief valve if dangerously high pressure is detected (*Damage limitation*).

Insulin pump - software risks

- Arithmetic error
  - A computation causes the value of a variable to overflow or underflow;
  - Maybe include an exception handler for each type of arithmetic error.
- Algorithmic error
  - Compare dose to be delivered with previous dose or safe maximum doses. Reduce dose if too high.

Safety requirements - insulin pump

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 for a system user*.

SR3: *The system shall include a hardware diagnostic facility that shall be executed at least 4 times per hour*.

SR4: *The system shall include an exception handler for all of the exceptions that are identified in Table 3 (an imaginary table stating exceptions in Req. Spec. Doc.)*.

SR5: *The audible alarm shall be sounded when any hardware or software anomaly is discovered and a diagnostic message as defined in Table 4 should be displayed*.

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

Safety specification

- The safety requirements of a system should be ***separately specified***.
- These requirements should be ***based on an analysis of the possible hazards and risks*** as previously discussed.
- Safety requirements usually ***apply to the system as a whole*** rather than to individual sub-systems. In systems engineering terms, the ***safety*** of a system ***is an emergent property***.

Safety requirements

- *Functional safety requirements*
  - … define the safety functions of the protection system i.e. ***define how the system should provide protection***.
- *Safety integrity requirements*
  - … define the reliability and availability of the protection system. They are based on expected use and are classified using a safety integrity level from 1 to 4.

Security specification

- *Similarities to safety specification*
  - Not possible to specify security requirements quantitatively;
  - The requirements are often ‘shall not’ rather than ‘shall’ requirements. (similarity in being ***excluding*** requirements.)
- *Differences*
  - No well-defined notion of a security life cycle for security management; *No standards*;
  - *Generic threats* rather than system specific hazards;
  - *Mature security technology (encryption, etc.)*. However, there are *problems in transferring this into general use*;
  - The *dominance of a single supplier* (Microsoft) means that huge numbers of systems may be affected by security failure.

Stages in security specification

- Asset identification and evaluation
  - The assets (data and programs) and their required degree of protection are identified. The degree of required protection depends on the asset value: e.g., a password file (say) is more valuable than a set of public web pages.
- Threat analysis and risk assessment
  - Possible security threats are identified and the risks associated with each of these threats is estimated.
- Threat assignment
  - Identified threats are related to the assets so that, for each identified asset, there is a list of associated threats.

Stages in security specification

- Technology analysis
  - Available security technologies and their applicability against the identified threats are assessed.
- Security requirements specification
  - The security requirements are specified. Where appropriate, these will explicitly identify the security technologies that may be used to protect against different threats to the system.

LIBSYS security requirements

SEC1: All system users shall be identified using their library card number and personal password.

SEC2: Users privileges shall be assigned according to the class of user (student, staff, library staff).

SEC3: Before execution of any command, LIBSYS shall check that the user has sufficient privileges to access and execute that command.

SEC4: When a user orders a document, the order request shall be logged. The log data maintained shall include the time of order, the user’s identification and the articles ordered.

SEC5: All system data shall be backed up once per day and backups stored off-site in a secure storage area.

SEC6: Users shall not be permitted to have more than 1 simultaneous login to LIBSYS.

System reliability specification

- *Hardware reliability*
  - What is the probability of a hardware component failing and how long does it take to repair that component?
- *Software reliability*
  - How likely is it that a software component will produce an incorrect output? Software failures are different from hardware failures in that software does not wear out. It can continue in operation even after an incorrect result has been produced.
- *Operator reliability*
  - How likely is it that the operator of a system will make an error?

Functional reliability requirements

- A predefined range for all values that are input by the operator shall be defined and the system shall check that all operator inputs fall within this predefined range.
- The system shall check all disks for bad blocks when it is initialised.
- The system must use *N-version programming (Fault tolerant SW)* to implement the braking control system.
- The system must be implemented in a safe subset of Ada and checked using static analysis.

Non-functional reliability specification

- The level of system reliability required *should be verifiable*, hence, *expressed quantitatively*.
- Reliability is a *dynamic system attribute*- reliability specifications *related to the source code* are *meaningless such as the example below:*
  - *No more than N faults/1000 lines*;
  - This is only useful for a post-delivery process analysis where you are trying to assess how good your development techniques are.
- An appropriate reliability metric should be chosen to specify the overall system reliability.

Reliability metrics

- ***Probability of failure on demand (POFOD):*** *The likelihood that the system will fail when a service request is made.* A POFOD of 0.001 means that 1 out of a thousand service requests may result in failure.
- ***Rate of failure occurrence (ROCOF):*** *The frequency of occurrence with which unexpected behaviour is likely to occur.* A ROCOF of 2/100 means that 2 failures are likely to occur in each 100 operational time units (tu). This metric is sometimes called the *failure intensity*.
- ***Mean time to failure (MTTF):*** *The average time between two consecutive system failures.* An MTTF of 500 means that 1 failure can be expected every 500 tus.
- ***Availability (AVAIL):*** *The probability that the system is available for use at a given time.* Availability of 0.998 means that in every 1000 tus, the system is likely to be available for 998 tus.

Steps to a reliability specification

1. For each sub-system: *analyse consequences of possible system failures*.
2. From *system failure analysis*: *partition failures into appropriate classes (regarding the seriousness of their consequence)*.
3. For each *failure class* identified: *set out the reliability using an appropriate metric*.
   - *Possible to use different metrics for different reliability requirements*.
4. To reduce the chances of critical failures: *identify functional reliability requirements*

Bank auto-teller system

- Each machine in a network is used 300 times a day
- Bank has 1000 machines
- Lifetime of software release is 2 years
- Each machine handles about 200, 000 transactions
- About 300, 000 database transactions in total per day

Reliability specification for an ATM

Failure class	Example	Reliability metric
Permanent, non-corrupting.	The system fails to operate with any card that is input. Software must be restarted to correct failure.	ROCOF 1 occurrence/1000 days
Transient, non-corrupting	The magnetic stripe data cannot be read on an undamaged card that is input.	ROCOF 1 in 1000 transactions
Transient, corrupting	A pattern of transactions across the network causes database corruption.	Unquantifiable! Should never happen in the lifetime of the system

Specification validation

- It is impossible to empirically validate very high reliability specifications.
- No database corruptions means POFOD of less than 1 in 200 million service requests.
- If a transaction takes 1 second, then simulating one day's transactions takes 3.5 days.
- It would take longer than the system's lifetime to test it for reliability.

Quiz 1 – Week 5

Send answers to **cse3044marmara@gmail.com**

(20pts + 10pts + 20pts)

1. Why is a *reliability requirement* such as «No more than N faults/1000 lines» is meaningless?

2. List at least three reliability metrics!

3. What type of requirements are safety and security requirements in general?