

# НАДЕЖДНОСТ И СИГУРНОСТ НА СОФТУЕРНИ СИСТЕМИ

---



СУ „Св. Климент Охридски“  
Факултет по Математика и Информатика  
Увод в Софтуерното Инженерство

# 70s: Mainframes

## Systems/users:

$\sim 10^4$

## User competency:

Engineers



## Integration complexity

- Close systems
- Highly custom designs
- Hardware and software fully controlled by vendors



- Hardware

# 80s: Workstations

## Systems/users:

$\sim 10^6$

## User competency:

Basic knowledge



## Integration complexity

- Mostly close systems
- Network connectivity
- Standard interfaces exported for users



- Hardware
- Network

# 90s: Personal Computers

## Systems/users:

$\sim 10^7$

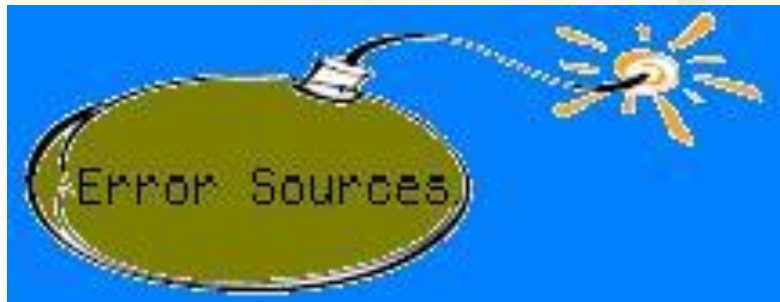
## User competency:

Computer Literacy



## Integration complexity

- Open systems
- Wide network access
- Commercial OS
- Third party software and hardware



- Hardware, Software
- Network
- Human mistakes

# 2000s: Mobile Devices

## Systems/users:

~10<sup>9</sup> and more

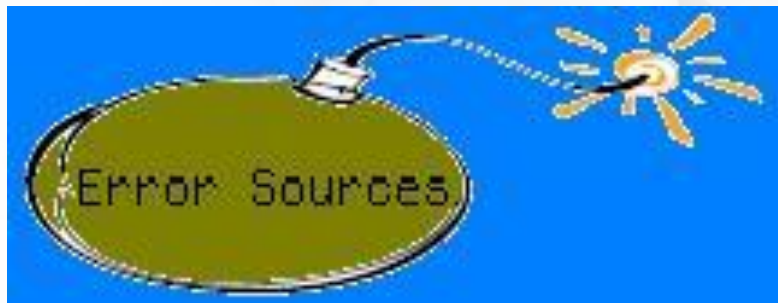
## User competency:

Undefined



## Integration complexity

- Open systems
- COTS/proprietary OS
- Highly integrated computer systems
- Wider range of networks



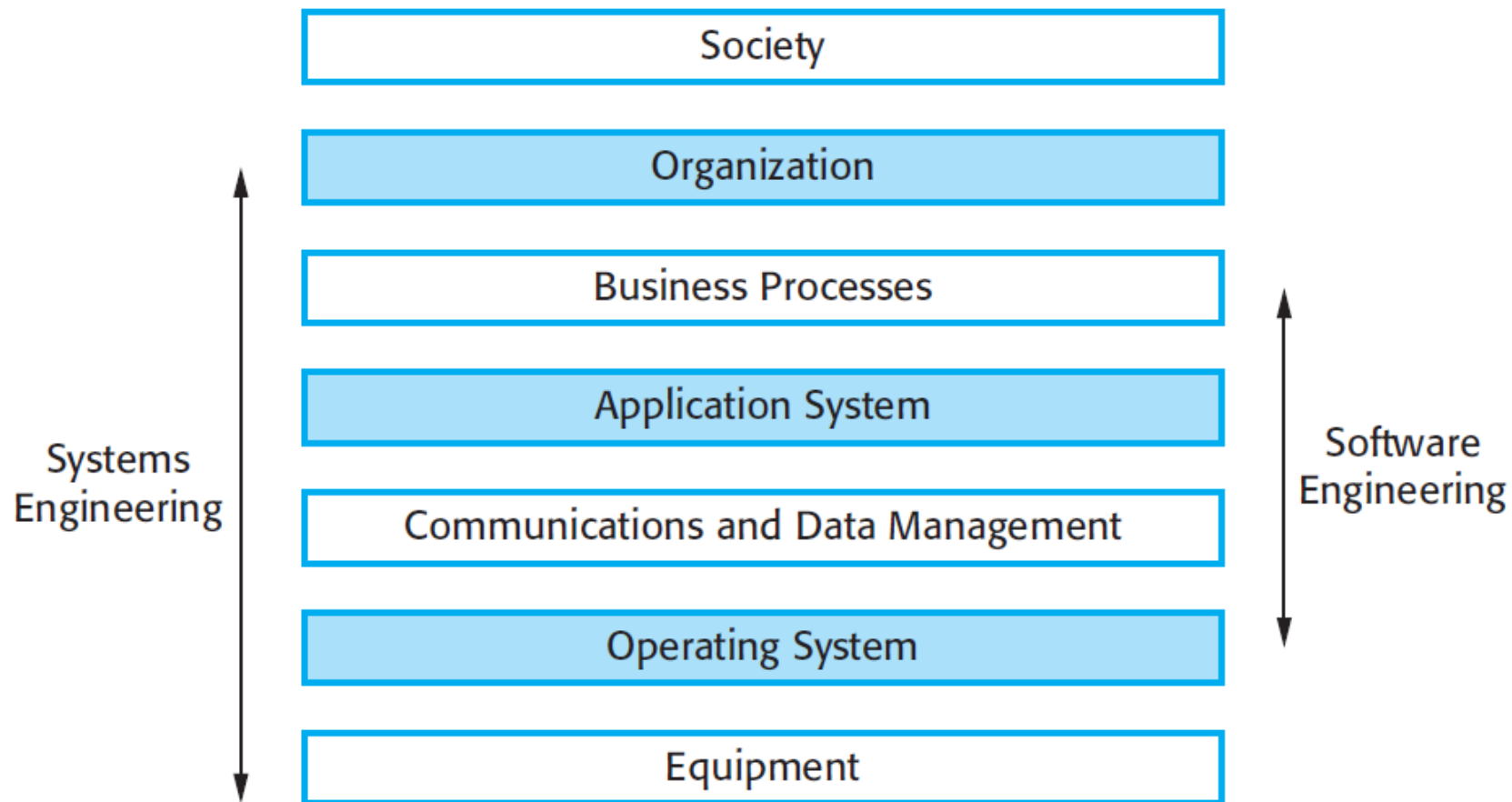
- Hardware, Software
- Network (wired/wireless)
- Human mistakes
- Malicious faults

# Industrial trends

- Newer application domains
- Increase in complexity of systems
- Increase in interactions among them
- Increase in volume of units
- Shift in error sources
- Reduced user tolerance levels

**Underline the growing importance of building dependable systems**

# Socio-technical system



# Layers in the STS stack

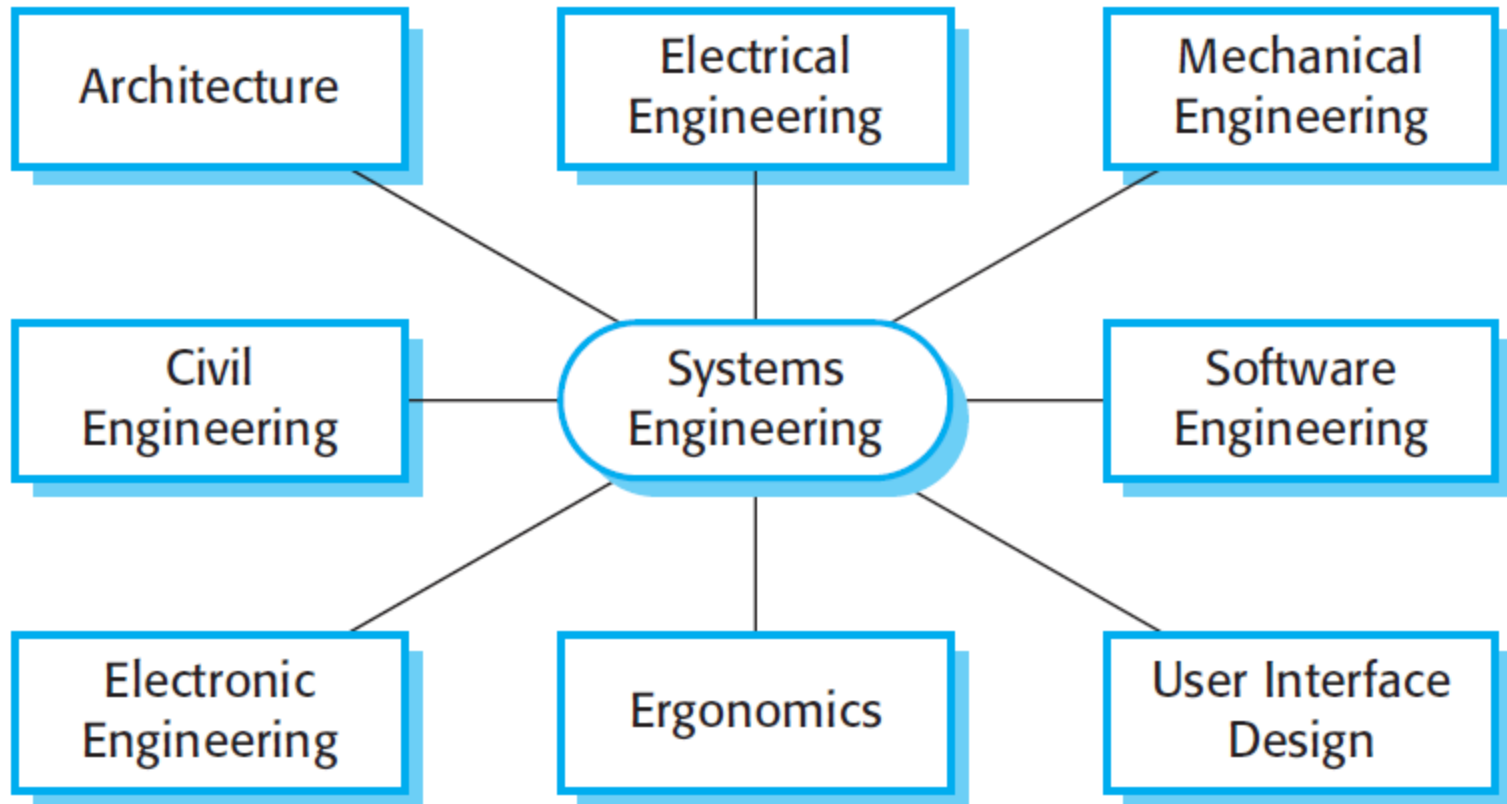
- **Equipment**
  - Hardware devices, some of which may be computers. Most devices will include an embedded system of some kind.
- **Operating system**
  - Provides a set of common facilities for higher levels in the system.
- **Communications and data management**
  - Middleware that provides access to remote systems and databases.
- **Application systems**
  - Specific functionality to meet some organization requirements.



# Layers in the STS stack

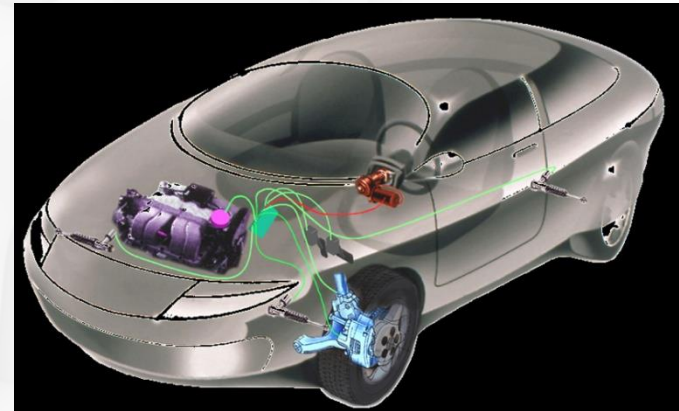
- **Business processes**
  - A set of processes involving people and computer systems that support the activities of the business.
- **Organizations**
  - Higher level strategic business activities that affect the operation of the system.
- **Society**
  - Laws, regulation and culture that affect the operation of the system.

# Systems engineering



# Nonfunctional requirements

- Define HOW software should perform its functionality
- Also known as “-ilities”
  - Dependability
  - Testability
  - Usability
  - Modifyability
  - Etc.



# Dependability

- Computer systems are characterized by many fundamental properties:
  - Functionality
  - Non-functional characteristics
    - Performance
    - Cost
    - Reliability
    - Integrity
    - Availability
    - Etc..

# Definition of dependability

“Dependability of a computing system is the ability to deliver service that can justifiably be trusted”

The service delivered by a system is its behaviour as it is perceived by its users

A user is another system  
(physical, human)

# HW Reliability vs. SW Dependability

## Hardware

- Deterioration over time
- Design faults removed before manufacture
- No new faults enter during life-cycle
- Initial use & end of life failures common
- No need of time to correct fault

**WARRANTY**

## Software

- No deterioration over time
- Faults removed after build
- Faults possibly enter during fault correction
- Failures during Initial test period, early use common, then stable
- Time needed to correct faults

**DISCLAIMER**

# Classes of dependable systems



# Classes of Dependable Systems

- **Safety-critical**
  - Airplanes
  - Cars
  - Nuclear power stations
  - Traffic control systems
  - Energy supply and distribution systems
  - Telecommunication systems
  - Etc...
- **Mission-critical**
  - Shuttles
  - Airplanes
- **Business-critical**
  - Industrial systems
  - Information systems
  - Traffic control systems

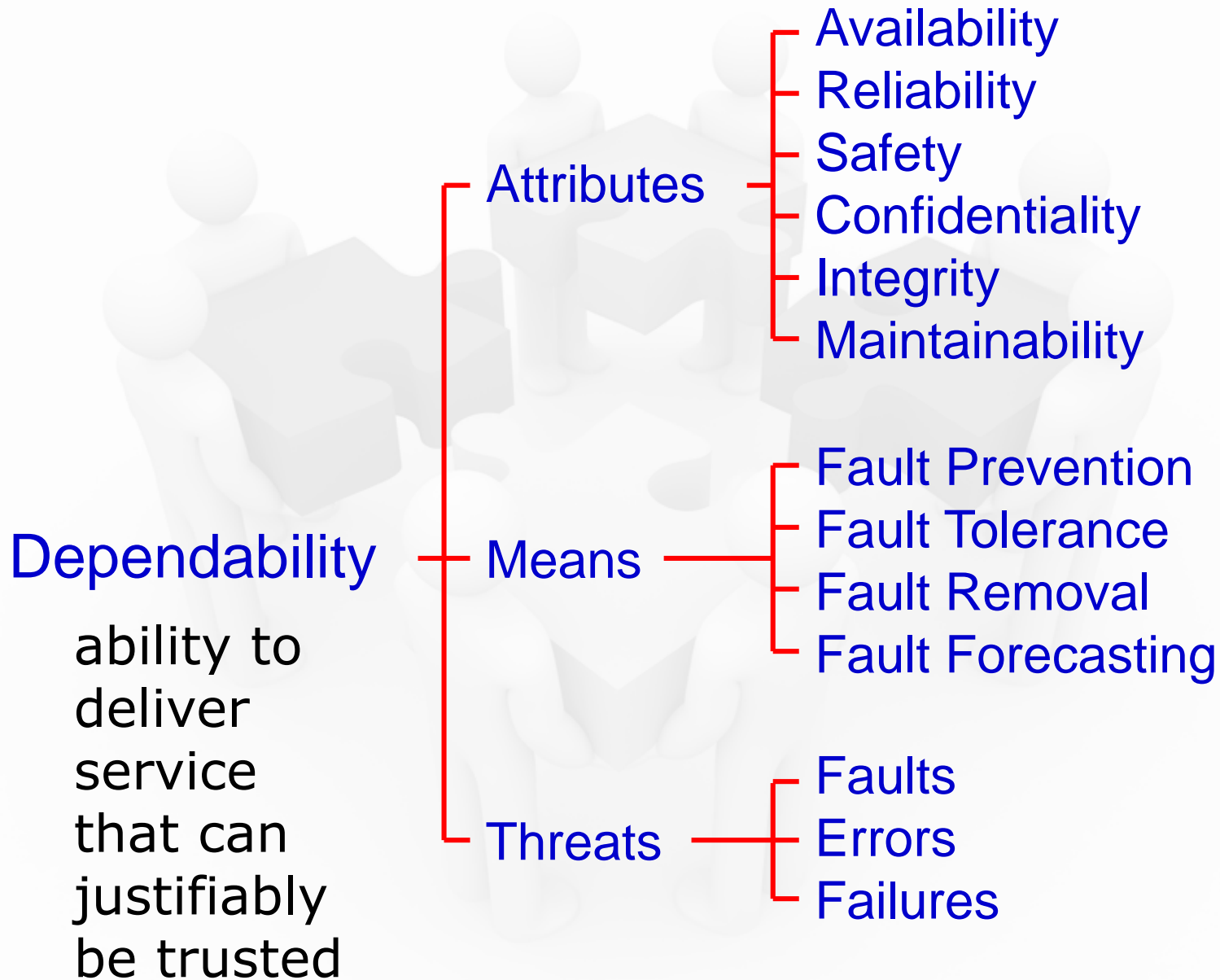


# Importance of dependability

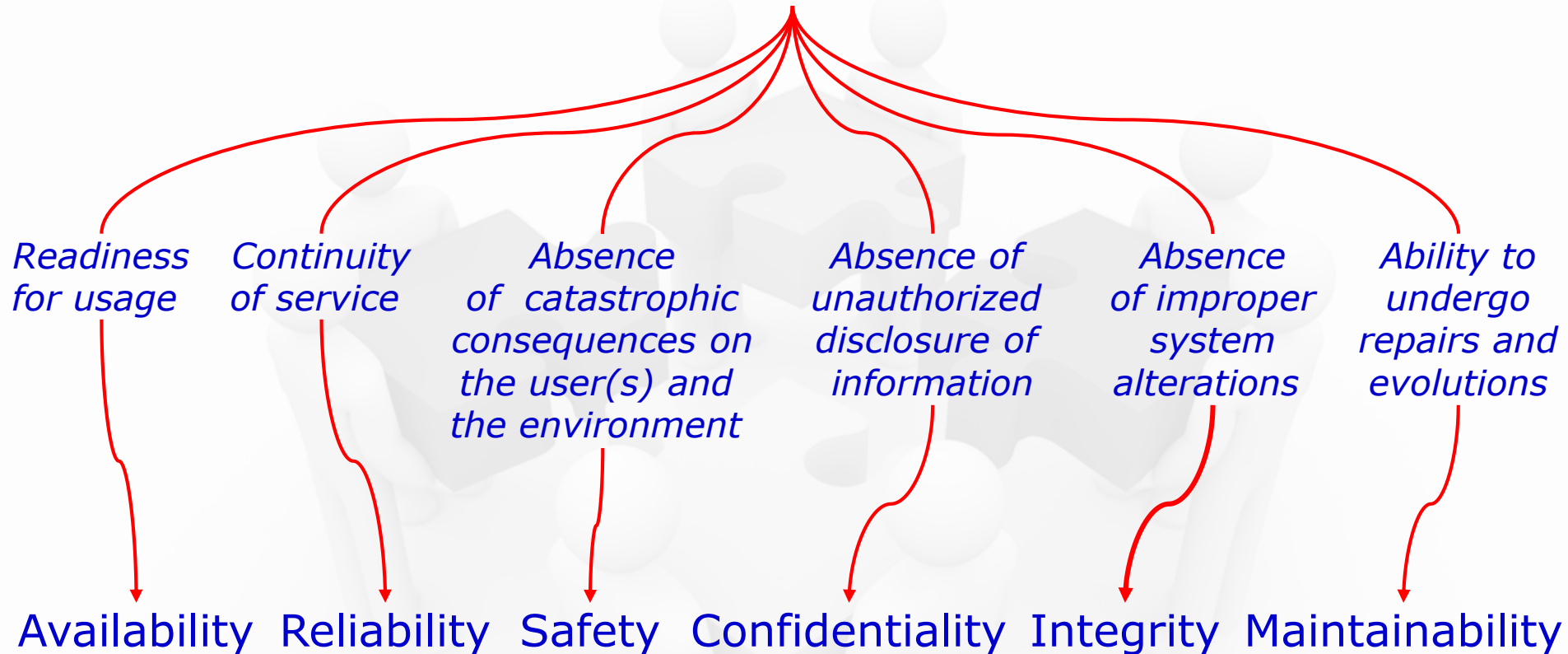
- System failures may have widespread effects with large numbers of people affected by the failure.
- Systems that are not dependable and are unreliable, unsafe or insecure may be rejected by their users.
- The costs of system failure may be very high if the failure leads to economic losses or physical damage.
- Undependable systems may cause information loss with a high consequent recovery cost.

# Causes of failure

- **Hardware failure**
  - Hardware fails because of design and manufacturing errors or because 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 in socio-technical systems.



# Dependability



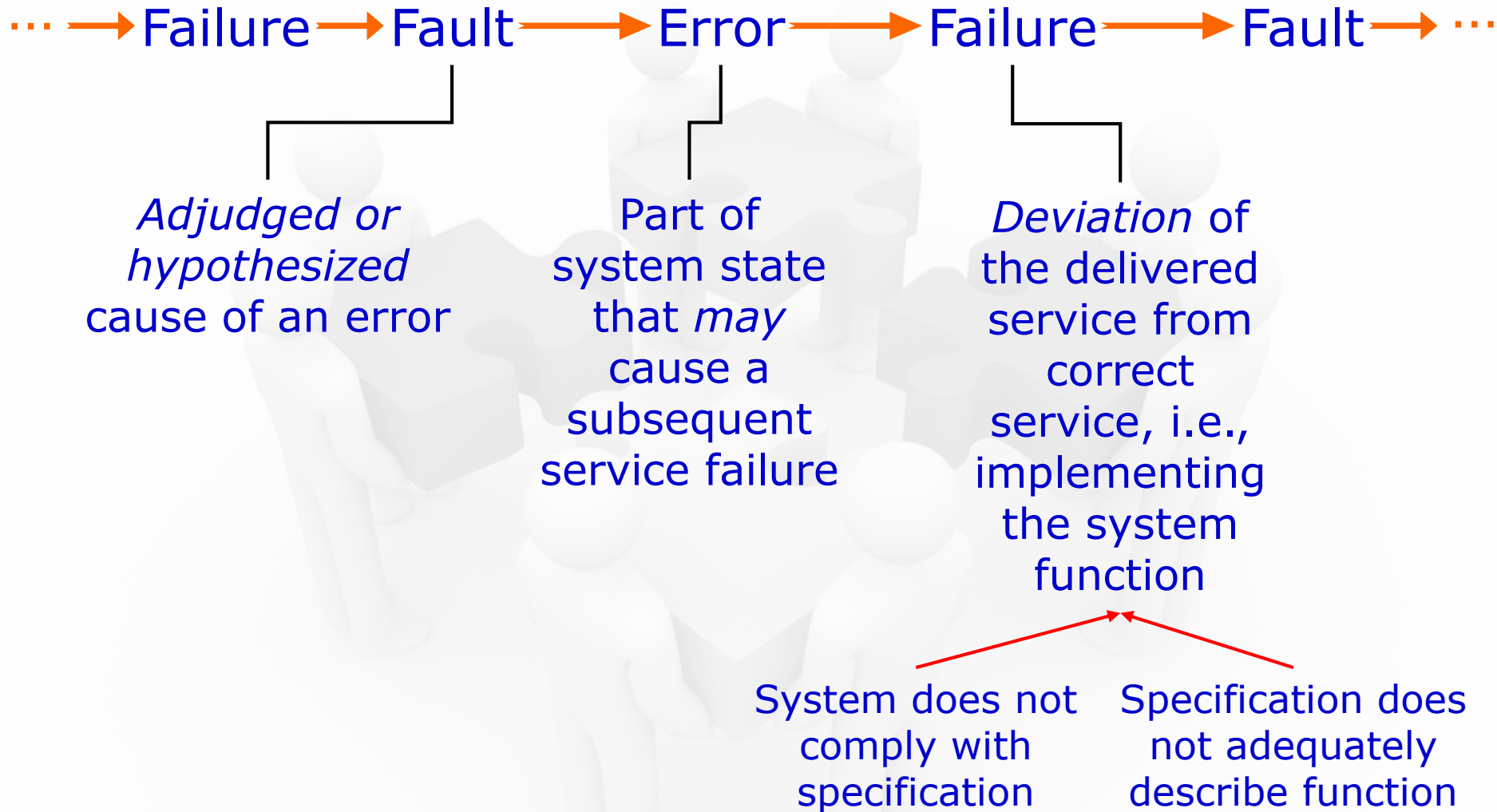
**Dependability:** Ability of the system to provide service that can justifiably trusted

# Strict definitions

- **Reliability:** The probability of failure-free (as per specification) operation over a specified time, in a given environment, for a specific purpose.
  - Depends on the environment
- **Availability:** The probability that a system, at a point in time, will be operational and able to deliver the requested services.
  - 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.

# Availability vs. Reliability

- Availability and reliability are not the same.
- Can a system be highly available but unreliable?
  - If a system goes down for a millisecond every hour, it has an availability of over 99.9999 percent, but it is still highly unreliable.
- Can a system be highly reliable but not available?
  - A system that never crashes but is shut down for two weeks every August has high reliability but only ~96 percent availability.

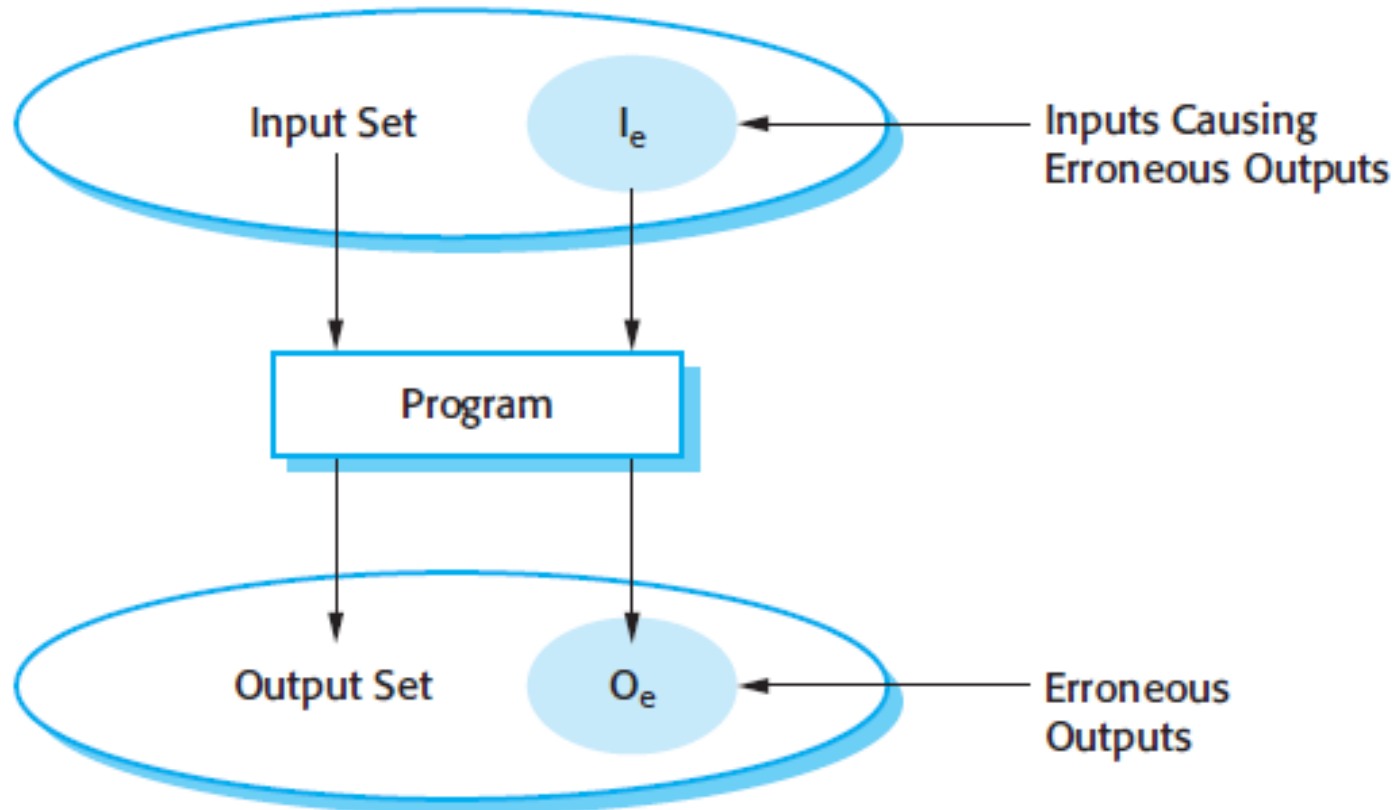


# Fault, Error, Failure - Example

- A Fault:
  - ```
int increment (int x) {  
    x = x+11; // should be x = x +1;  
}
```
- An Error – fault activated
  - `Y = increment(2);`
  - Can be propagated.
- A Failure – Error exposed to interface
  - `Print(Y);`



# Failures vs Input “space”



# Faults nature

- 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 may be 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 behavior is discovered and corrected before the system services are affected.

# Faults and failures

- 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

# Strict definitions

- **Reliability:** The probability of failure-free (as per specification) operation over a specified time, in a given environment, for a specific purpose.
  - Depends on the environment
- **Availability:** The probability that a system, at a point in time, will be operational and able to deliver the requested services.
  - 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.

# Availability vs. Reliability

- Availability and reliability are not the same.
- Can a system be highly available but unreliable?
  - If a system goes down for a millisecond every hour, it has an availability of over 99.9999 percent, but it is still highly unreliable.
- Can a system be highly reliable but not available?
  - A system that never crashes but is shut down for two weeks every August has high reliability but only ~96 percent availability.

# Software reliability

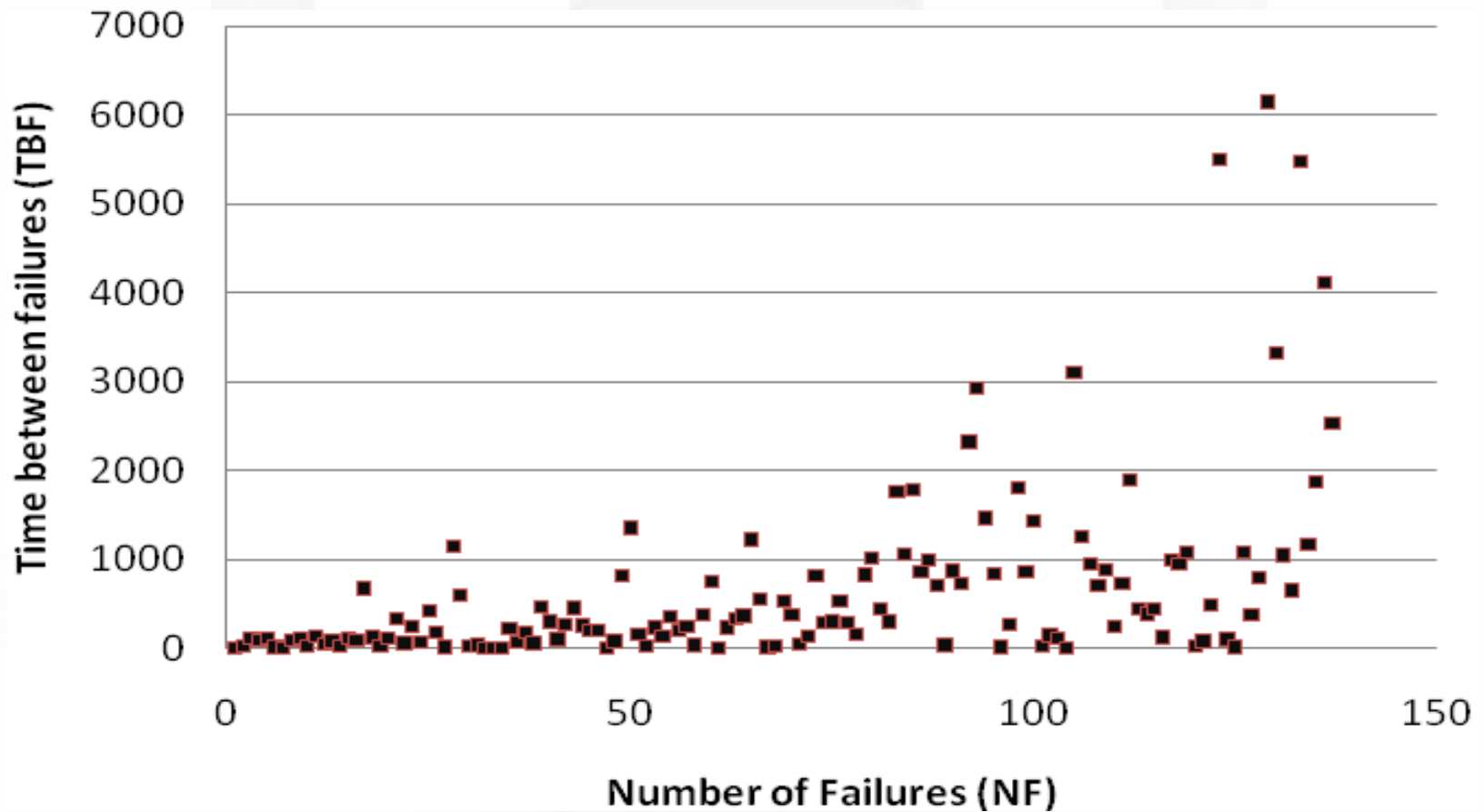
- Slightly different from traditional (hardware) reliability theory
  - Failure is deterministic, user behaviour is not
- Probabilistic value
  - Probability of failure (success)
  - Mean time to failure ( $\mu$ )
  - Failure rate  $\lambda$ 
    - $\lambda = 1/\mu$



# Reliability modeling data

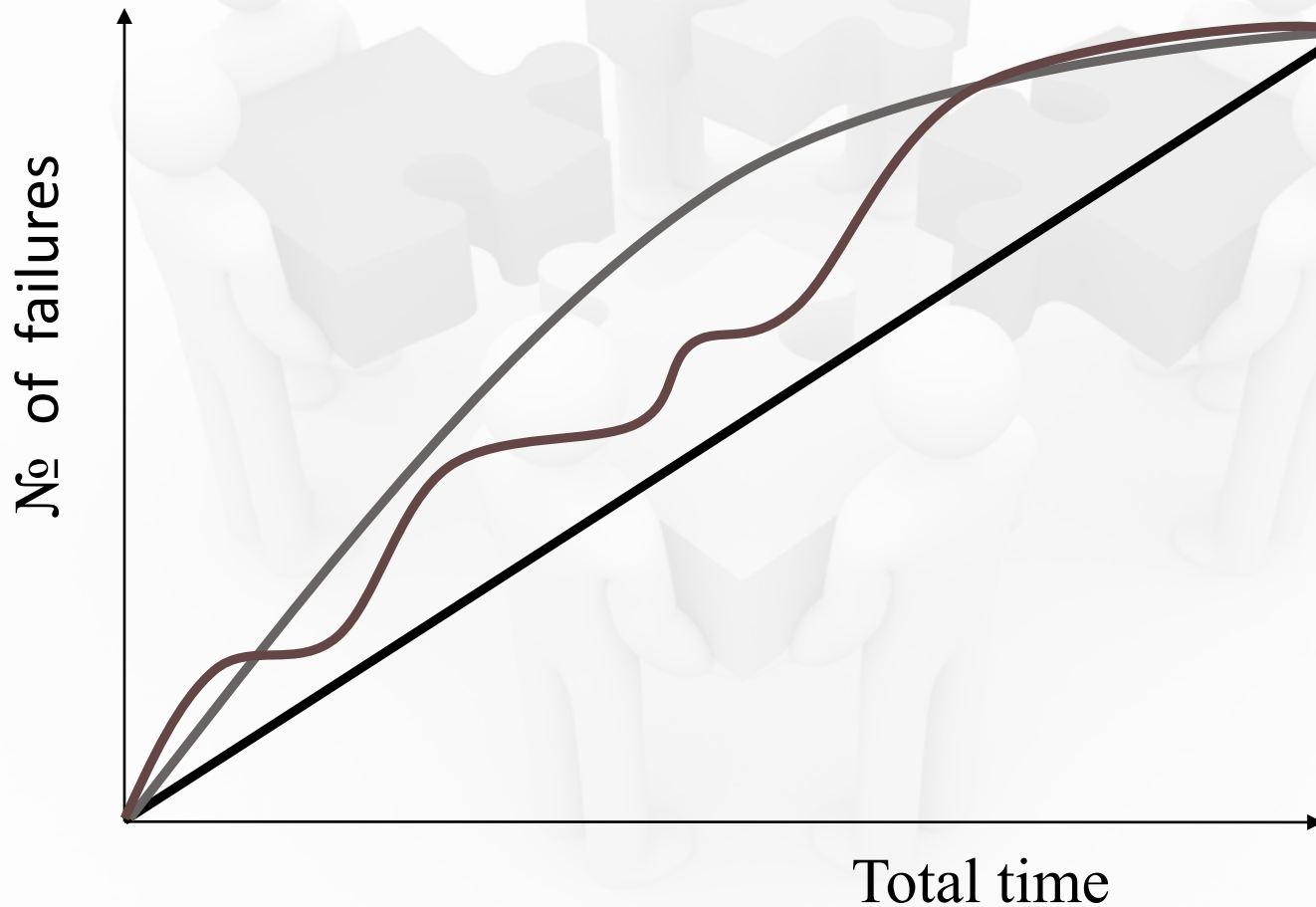
- Reliability is a probabilistic value, which may be calculated using statistical methods over some datasets
- Such datasets may be collected using different methods, like:
  - Testing
  - Users feedback
  - Experts opinion
  - Simulation

# Typical failure data set





# Kinds of software system failure behaviours



# Reliability estimation models

- **White box models**
  - Build an architectural model of the system
  - Integrate failure behaviour of individual components with architectural model
- **Black box models**
  - Statistic processing of data
  - Software Reliability Growth Models (SRGMs)



# Software Reliability Models

- Failure rate

$$\lambda(\tau) = \lambda_0 e^{-\frac{\lambda_0}{v_0} \tau}$$

$$\lambda(\tau) = \lambda(\mu) = \lambda_0 \left(1 - \frac{\mu}{v_0}\right)$$

$$\mu(\tau) = v_0 \left(1 - e^{-\frac{\lambda_0}{v_0} \tau}\right)$$

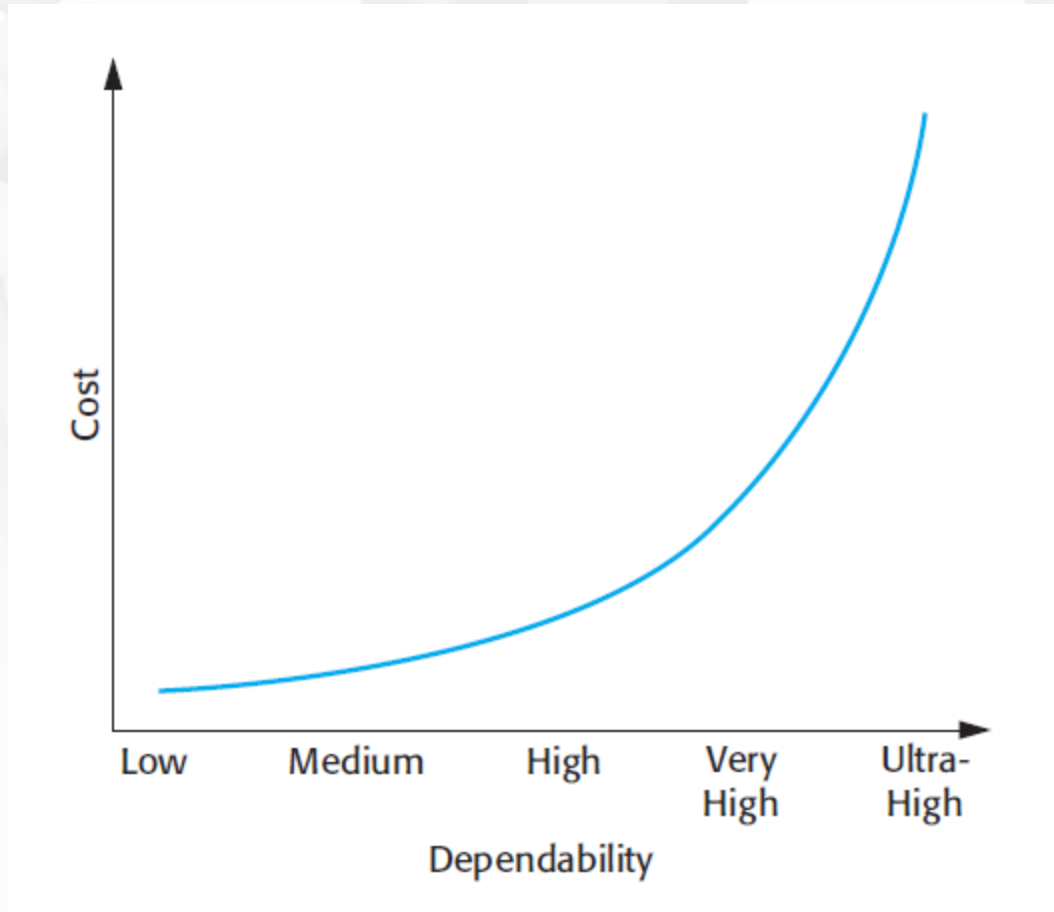
# Reliability in use

- 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.
- Users adapt their behaviour to avoid system features that may fail for them.
- A program with known faults may therefore still be perceived as reliable by its users.

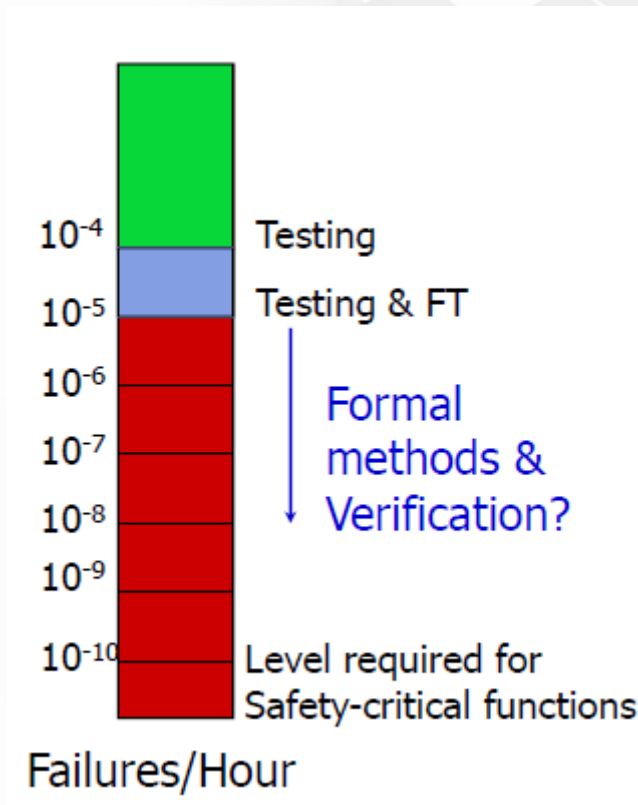
# The problem with testing

- Reliability of life-critical and real-time software is infeasible to be quantified by testing [Butler & Finelli 1993]
- Only small reliabilities (99,999%) are possible to be estimated in a obtainable period of time for testing
- For example assuring that a program has failure rate of about  $10^{-7}$  per hour may require thousands (and even more) years of testing
- There may not exist effective oracle to carry out statistical testing

# Cost of dependability/reliability



# Attainable levels of SW reliability



- FAA (Federal Aviation Administration) & NASA safety-critical requirement is less than  $10^{-10}$  failures per 10 hrs of flight.

# 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 two 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 1 long outage. Long repair times are a particular problem.



# Reliability/availability 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 00XX) 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.                                                                                                                                                                                             |

# 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 important to consider software safety as most devices whose failure is critical now incorporate software-based control systems.
- Safety requirements are often exclusive requirements i.e. they exclude undesirable situations rather than specify required system services. These generate functional safety requirements.

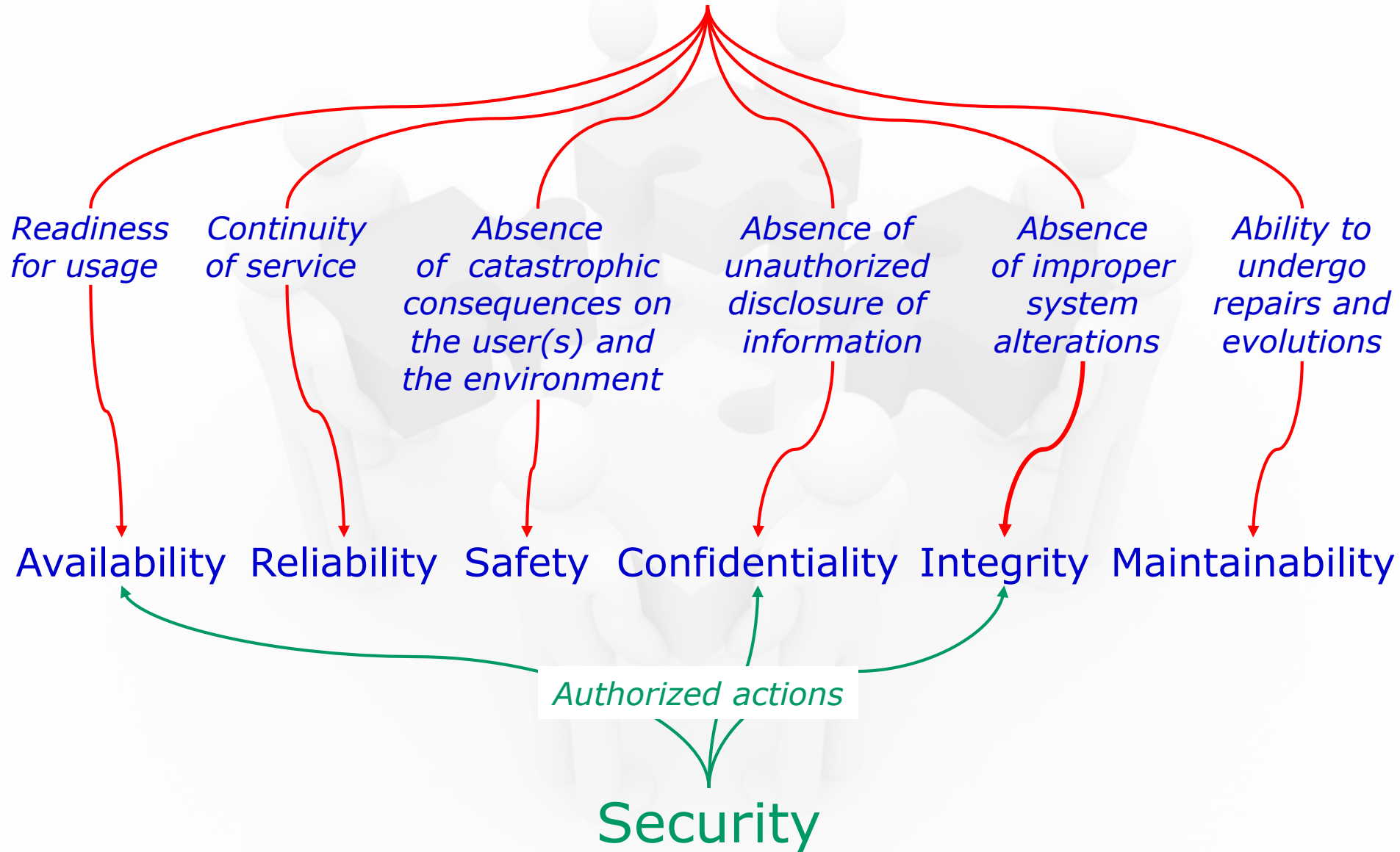
# Safety and reliability

- 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

# Safety terminology

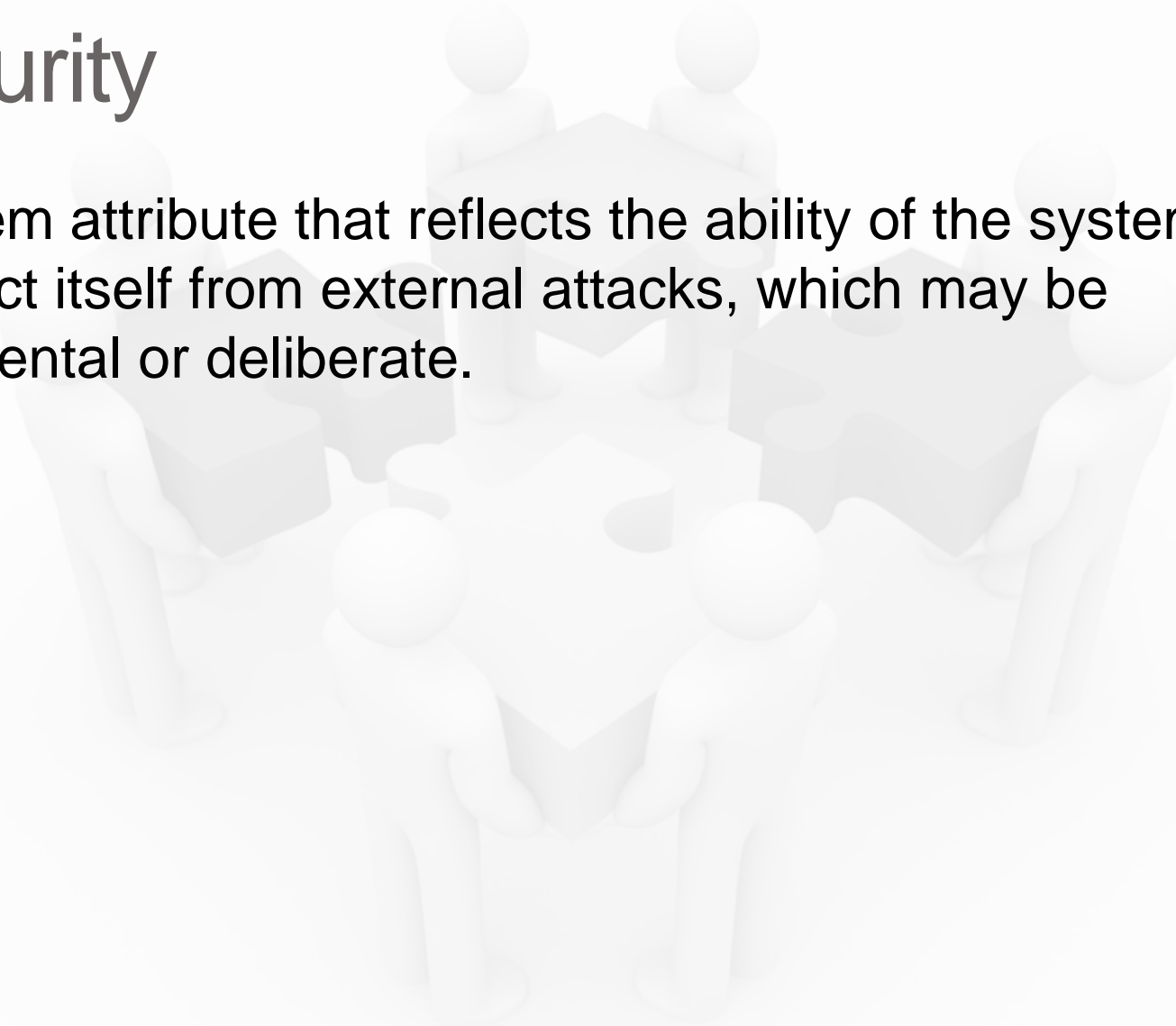
| Term                 | Definition                                                                                                                                                                                                                                                                                                                                                              |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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.                                                                                  |

# Dependability & Security



# Security

- System attribute that reflects the ability of the system to protect itself from external attacks, which may be accidental or deliberate.



# Security terminology

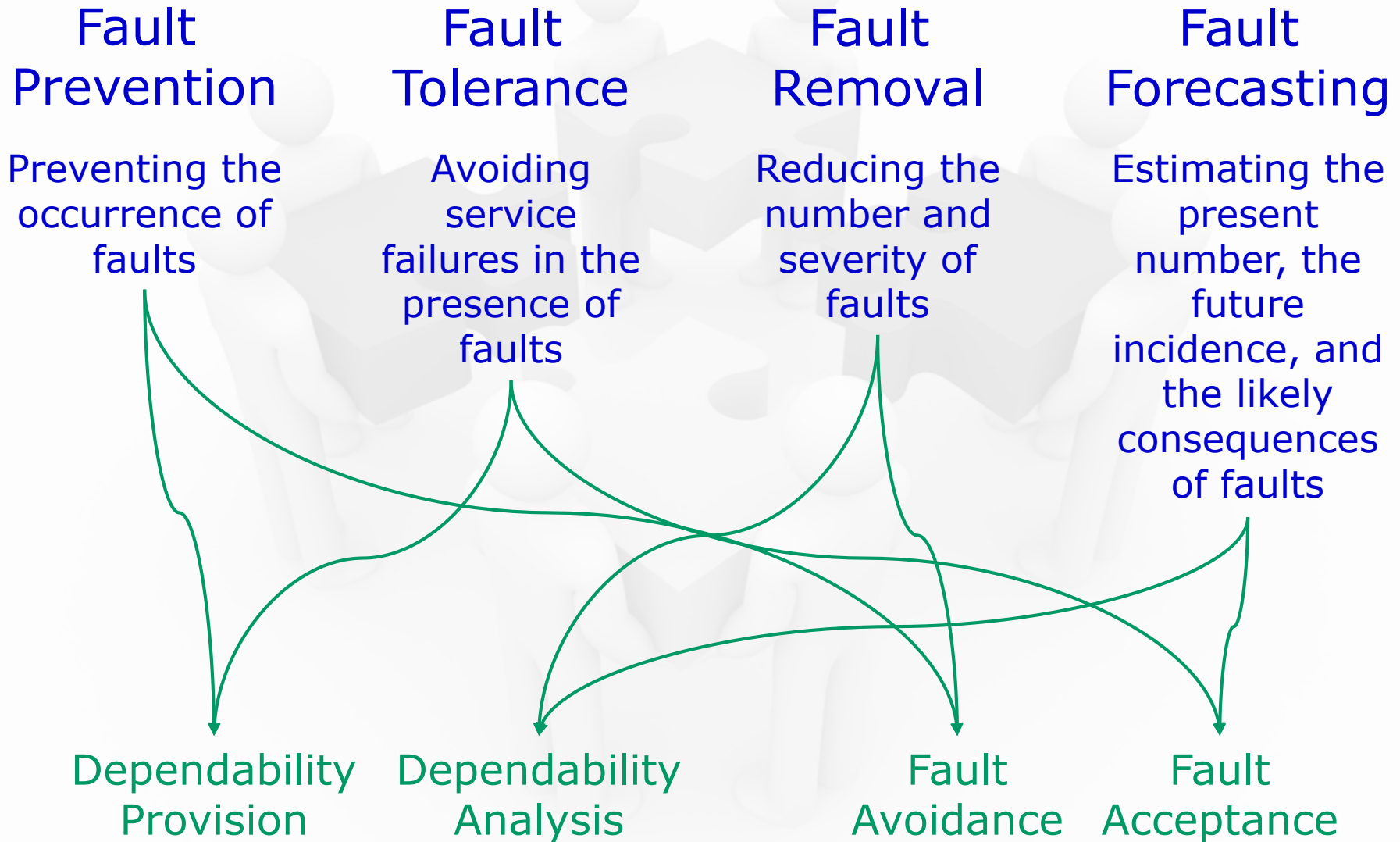
| Term          | Definition                                                                                                                                                             |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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.        |

# Example

| 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.                                                                           |



# Means to attain dependability



# Fault Prevention

- **Fault avoidance**
  - Techniques that prevent introduction of faults during development
    - Hardware: use reliable components, packaging
    - Software: formal specs, use of proven design methods.
- **Fault removal**
  - System testing is most important
  - Reviews, verifications, code inspections
- 'A test can only show presence of faults, but never prove its absence'

[Dijkstra]

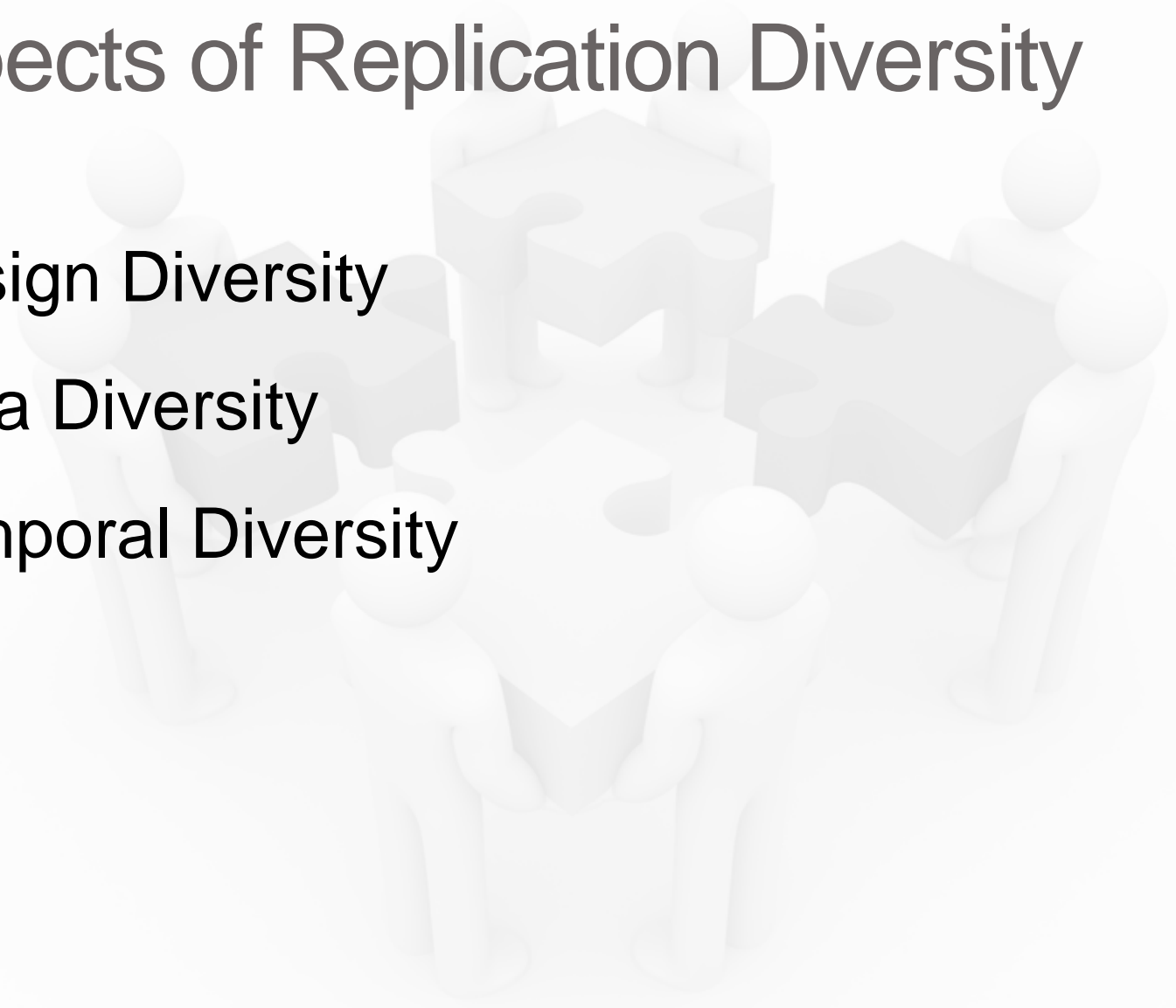
# Fault Tolerance

- The ability of the system to continue functioning irrespective of the presence of faults
- Levels of fault tolerance
  - Fail operational ( Full fault tolerance)
  - Fail soft (Graceful degradation)
  - Failsafe
- **Redundancy is the** key for fault-tolerance
  - Physical (Space), Information (data), Time, Analytical...

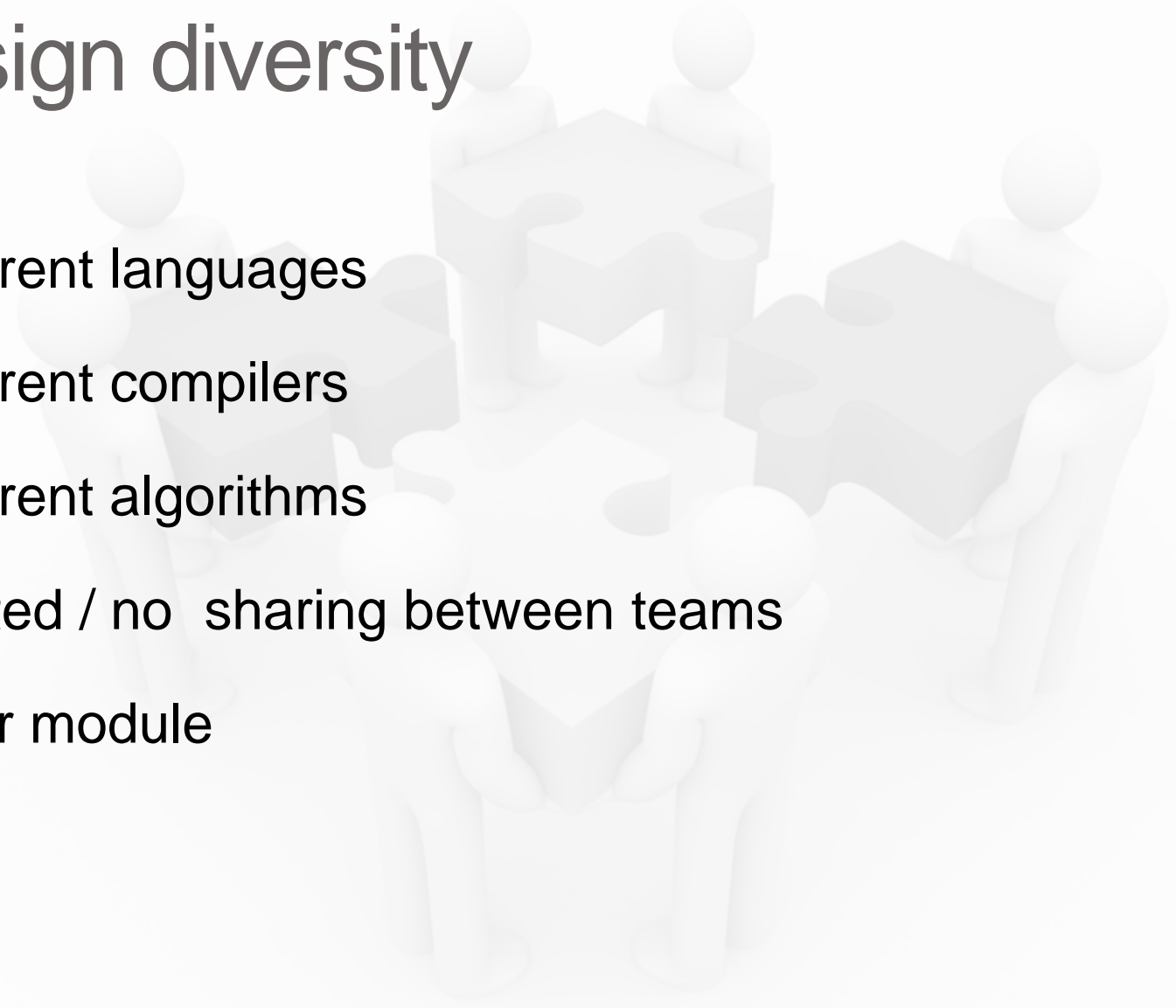
# Replication Diversity

- Software faults are usually caused by design error
- Multiplying a design error by redundancy is not a good idea
- Simple replication of identical software units is not the solution
- The key is to introduce diversity into software replicas

# Aspects of Replication Diversity

- Design Diversity
  - Data Diversity
  - Temporal Diversity
- 

# Design diversity

- Different languages
  - Different compilers
  - Different algorithms
  - Limited / no sharing between teams
  - Voter module
- 

# Data Diversity

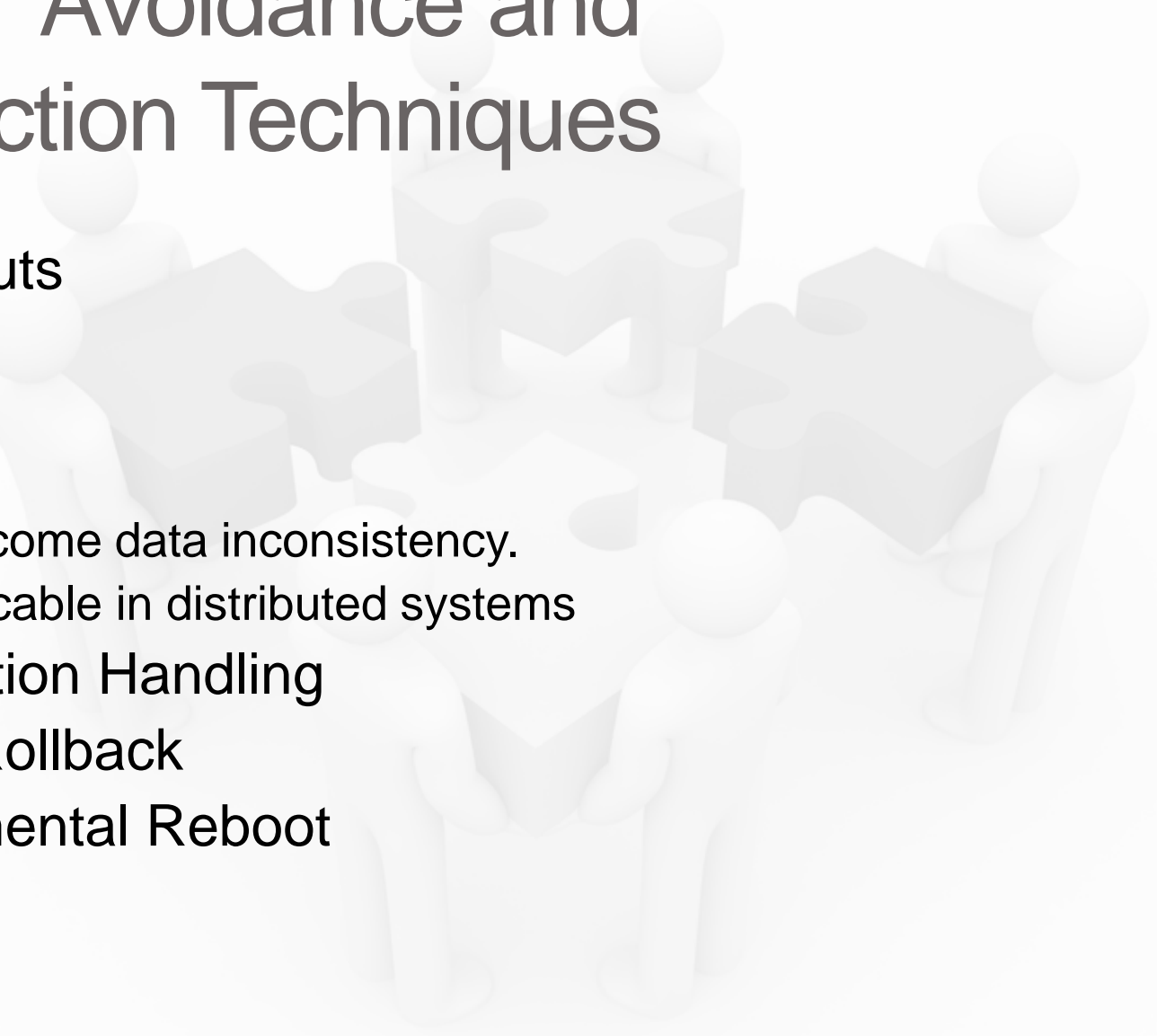
- Diversify input data for software to detect and tolerate software faults.
- Uses Data Re-Expression
  - Data Re-expression Algorithm (DRA) produces different representations of a module's input data.

# Temporal Diversity

- Involves the performance or occurrence of a certain event at different times
- Implementation of temporal diversity
  - By beginning software execution at different times
  - Using inputs that are produced or read at different times



# Fault Avoidance and Detection Techniques

- Timeouts
    - Retry
    - Abort
  - Audits
    - Overcome data inconsistency.
    - Applicable in distributed systems
  - Exception Handling
  - Task Rollback
  - Incremental Reboot
- 

# Fault forecasting

- Evaluation of system behavior
  - How to estimate the present number, the future incidents, the probability of different consequences
  - Qualitative (identify, classify, rank the failure modes, the event combinations, environmental conditions that would lead to system failures)
  - Quantitative (probabilistic)

# Re-thinking definitions?

- ‘Failure is defined as a ‘system not performing up to its specification’
- Good enough for piece of hardware, software, a micro processor...
- How about complex systems/services built using components?
  - With either no specs or vague specs
- How about socio-technical systems?
  - Customer satisfaction going to decide failure or otherwise ?

# Summary

- Mastery over control of physical faults over the years have drastically improved the availability of computing systems
- Presently, design and human made errors dominate as sources of failures
- Improving the state of the art in mastering these error sources is going to be on top of research agenda
- Appropriate end-to-end error models need to be developed and incorporated into the analysis and assurance

# References

- <http://www.cs.st-andrews.ac.uk/~ifs/Books/SE9/Presentations/index.html>
- A. Avizienis, J. Laprie, and B. Randell. Fundamental concepts of dependability. In Proceedings of the 3rd Information Survivability Workshop, 2000.
- Sasikumar Punnekkatt Dependability slides, [http://www.idt.mdh.se/kurser/computing/DVA403/DVA403-2012/Lectures/Dependability\\_DVA403\\_talk.pdf](http://www.idt.mdh.se/kurser/computing/DVA403/DVA403-2012/Lectures/Dependability_DVA403_talk.pdf)
- Sommerville, Ian, Software Engineering, 9th edition (2010), Addison-Wesley Pub Co;