Software Engineering

JUNBEOM YOO

Dependable Software Laboratory
KONKUK University

Chapter 10 – Dependable Systems

Topics Covered

- Dependability properties
- Sociotechnical systems
- Redundancy and diversity
- Dependable processes
- Formal methods and dependability

System Dependability

- For many computer-based systems, the most important system property is the **dependability** of the system.

- The dependability of a system reflects the user's degree of trust in that system.
  - It reflects the extent of the user's confidence that it will operate as users expect and that it will not 'fail' in normal use.

- Dependability covers the related systems attributes such as **reliability**, **availability** and **security**.
  - These are all inter-dependent.

Importance of Dependability

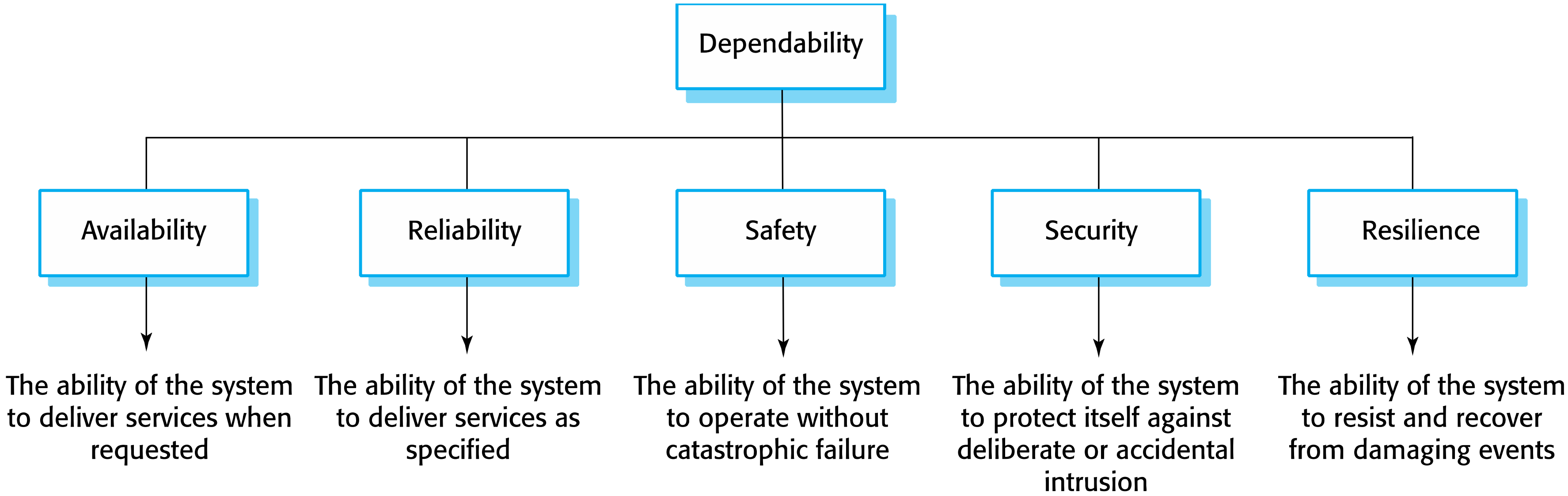
- System failures may have widespread effects with large numbers of people affected by the failure.
- Systems that are not dependable and are unreliable, unsafe or insecure may be rejected by their users.
- The costs of system failure may be very high if the failure leads to economic losses or physical 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 Properties

The Principal Dependability Properties



Principal Properties

- **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 cyberattacks.

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.

- Error tolerance
  – Reflects the extent to which user input errors can be avoided and tolerated.

Dependencies Among Dependability Attribute

- There are many kinds of dependencies among dependability properties.
  - Safe system operation depends on the system being available and operating reliably. (safety ↔ availability, reliability)
  - A system may be unreliable because its data has been corrupted by an external attack. (reliability ↔ security)
  - Denial of service attacks on a system are intended to make it unavailable. (security ↔ availability)
  - If a system is infected with a virus, you cannot be confident in its reliability or safety. (security ↔ reliability, safety)

Dependability Achievement

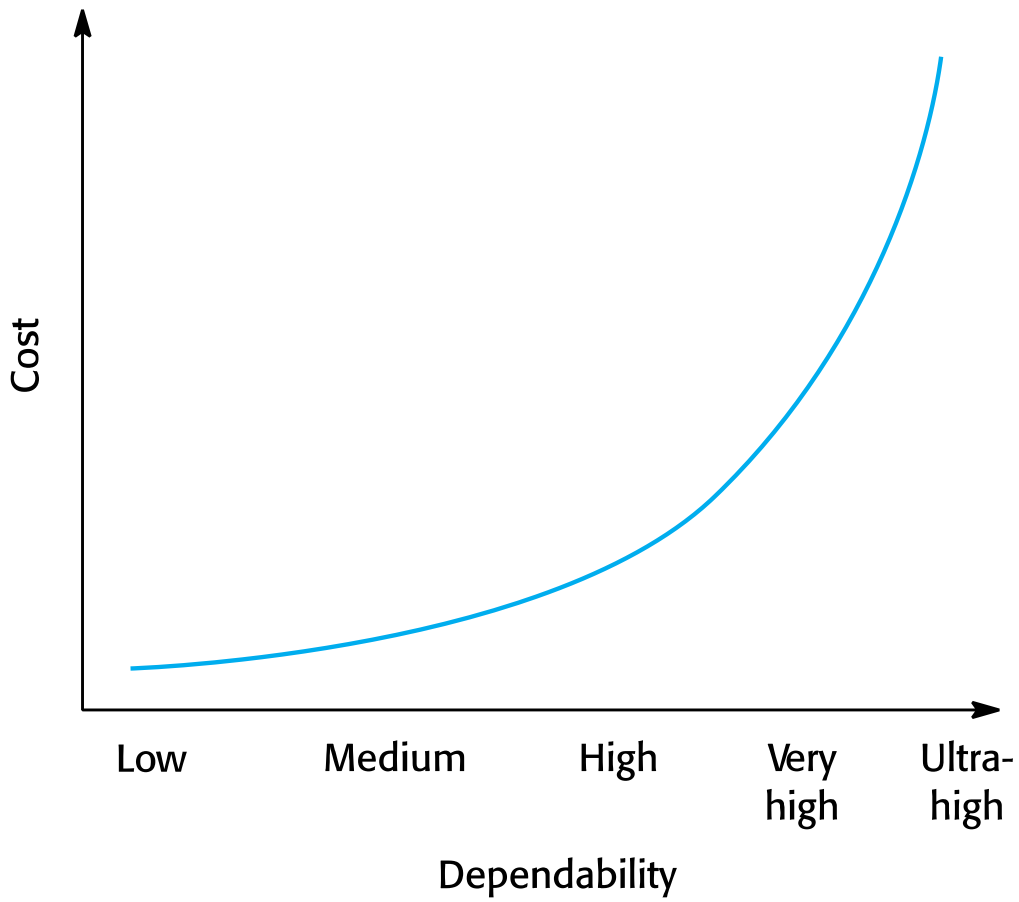
- Avoid the introduction of accidental errors when developing the system.
- Design V & V processes that are effective in discovering residual errors in the system.
- Design systems to be fault tolerant so that they can continue in operation when faults occur.
- Design protection mechanisms that guard against external attacks.
- Configure the system correctly for its operating environment.
- Include system capabilities to recognize and resist cyberattacks.
- Include recovery mechanisms to help restore normal system service after a failure.

Dependability Costs

- **Dependability costs** tend to increase exponentially as increasing levels of dependability are required.

- There are two reasons for this
  - The use of more expensive development techniques and hardware that are required to achieve the higher levels of dependability.
  - The increased testing and system validation that is required to convince the system client and regulators that the required levels of dependability have been achieved.

Cost/Dependability Curve



Dependability Economics

- It may be more cost effective to accept untrustworthy systems and pay for failure costs
  - However, this depends on social and political factors. A reputation for products that can't be trusted may lose future business
  - Depends on system type - for business systems in particular, modest levels of dependability may be adequate

Sociotechnical Systems

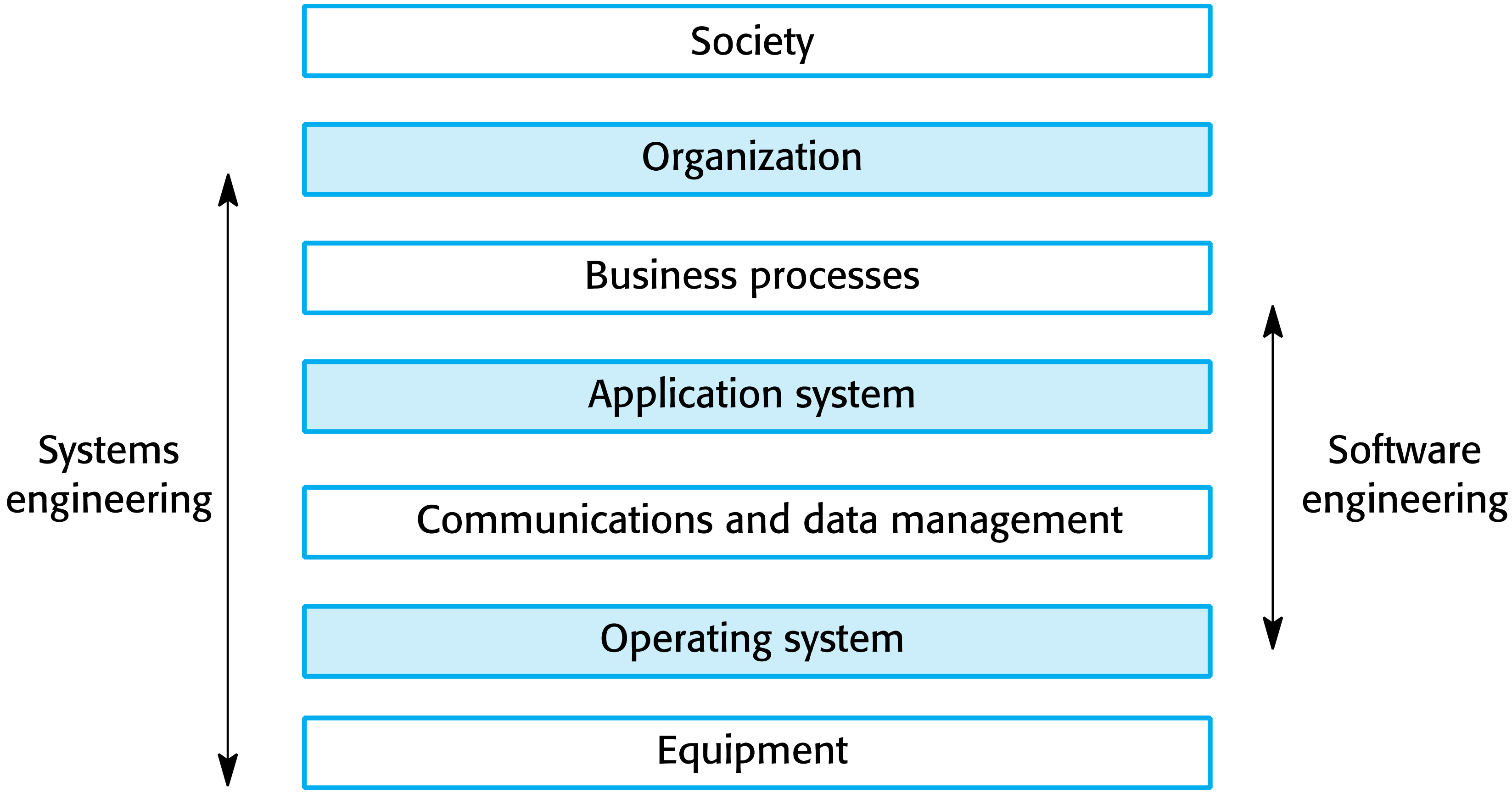
Systems and Software

- Software engineering is not an isolated activity but is part of a broader systems engineering process.

- Software systems are therefore not isolated systems but are essential components of broader systems that have a human, social or organizational purpose.

- Example
  - The wilderness weather system is part of broader weather recording and forecasting systems
  - These include hardware and software, forecasting processes, system users, the organizations that depend on weather forecasts, etc.

The Sociotechnical Systems Stack



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.
- 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.

Holistic System Design

- There are interactions and dependencies between the layers in a system and changes at one level ripple through the other levels
  - Example: Change in regulations (society) leads to changes in business processes and application software.

- For dependability, **a systems perspective** is essential.
  - Contain software failures within the enclosing layers of the STS stack.
  - Understand how faults and failures in adjacent layers may affect the software in a system.

Regulated Systems

- Many critical systems are **regulated systems**, which means that their use must be approved by an external regulator before the systems go into service.
  - Nuclear systems
  - Air traffic control systems
  - Medical devices

- A **safety and dependability case** must be approved by the regulator.
  - Critical systems development has to create the **evidence** to convince a regulator that the system is dependable, safe and secure.

Safety Regulation

- Regulation and compliance (following the rules) applies to the sociotechnical system as a whole and not simply the software element of that system.
- Safety-related systems may have to be certified as safe by the regulator.
- To achieve certification, companies that are developing safety-critical systems have to produce an extensive safety case that shows 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

Redundancy and Diversity

- **Redundancy**
  - Keep more than a single version of critical components so that if one fails then a backup is available.
- **Diversity**
  - Provide the same functionality in different ways in different components so that they will not fail in the same way.
- Redundant and diverse components should be independent so that they will not suffer from **'common-mode' failures (CCF)**.
  - For example, components implemented in different programming languages means that a compiler fault will not affect all of them.

Diversity and Redundancy examples

- Redundancy
  - Where availability is critical (e.g. in e-commerce systems), companies normally keep <u>backup servers</u> and switch to these automatically if failure occurs.

- Diversity
  - To provide resilience against external attacks, <u>different servers</u> may be implemented using <u>different operating systems</u> (e.g. Windows and Linux)

Process Diversity and Redundancy

- V&V activities should not depend on a single approach such as testing.
  - Redundant and diverse process activities are important.

- Multiple different (V&V) process activities complement each other.
  - Allow for cross-checking help to avoid process errors, which may lead to errors in the software.

Problems with Redundancy and 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 V&V as a more effective route to software dependability.
  - Airbus FCS architecture is redundant/diverse.
  - Boeing 777 FCS architecture has no software diversity.

Formal Methods and Dependability

Formal Specification

- **Formal methods** are approaches to software development that are based on mathematical representation and analysis of software.
  - Significantly reduce some types of programming errors
  - Can be cost-effective for dependable systems engineering

- Formal methods include
  - Formal specification
  - Specification analysis and proof
  - Transformational development
  - Program verification

Formal Approaches

- **Verification-based** approaches
  - Different representations of a software system (specifications) and a program implementing the specification are proved to be equivalent.
  - This demonstrates the absence of implementation errors.

- **Refinement-based** approaches
  - A representation of a system is systematically transformed into another, lower-level representation.
    - E.g., a specification is transformed automatically into an implementation.
  - If the transformation is correct, the representations are equivalent.

Use of Formal Methods

- The principal benefits of formal methods are in reducing the number of faults in systems. (fining defects)

- Formal method’s main application area are dependable systems engineering.
  - Several successful projects where formal methods have been used
  - The use of formal methods is most likely to be cost-effective because high system failure costs must be avoided.

Classes of Errors

- Errors and omissions in specification and design
  - Developing and analysing a formal model of the software may reveal errors and omissions in the software requirements.
  - If the model is generated automatically or systematically from source code, analysis using model checking can find undesirable states that may occur, such as deadlock in a concurrent system.

- Inconsistences between specifications and programs
  - If a refinement method is used, mistakes made by developers that make the software inconsistent with the specification are avoided.
  - Program proving discovers inconsistencies between a program and its specification.

Benefits of Formal Specification

- Developing a formal specification requires the system requirements to be analyzed in detail.
  - This helps to detect problems, inconsistencies and incompleteness in the requirements.
- As the specification is expressed in a formal language, it can be automatically analyzed to discover inconsistencies and incompleteness.
- If you use a formal method such as the B method, you can transform the formal specification into a 'correct' program.
- Program testing costs may be reduced if the program is formally verified against its specification.

Acceptance of Formal Methods

- Formal methods have had limited impact on practical software development.
  - Problem owners cannot understand a formal specification and so cannot assess if it is an accurate representation of their requirements.
  - It is easy to assess the costs of developing a formal specification but harder to assess the benefits. Managers may therefore be unwilling to invest in formal methods.
  - Software engineers are unfamiliar with this approach and are therefore reluctant to use formal methods.
  - Formal methods are still hard to scale up to large systems.
  - Formal specification is not really compatible with agile development methods.

Key Points

- System dependability is important because failure of critical systems can lead to economic losses, information loss, physical damage or threats to human life.
- The dependability of a computer system is a system property that reflects the user's degree of trust in the system. The most important dimensions of dependability are availability, reliability, safety, security and resilience.
- Sociotechnical systems include computer hardware, software and people, and are situated within an organization. They are designed to support organizational or business goals and objectives.
- The use of a dependable, repeatable process is essential if faults in a system are to be minimized. The process should include verification and validation activities at all stages, from requirements definition through to system implementation.
- The use of redundancy and diversity in hardware, software processes and software systems is essential to the development of dependable systems.
- Formal methods, where a formal model of a system is used as a basis for development help reduce the number of specification and implementation errors in a system.