Software Security and Dependability

ENGR5560G

Lecture 11

Dependable systems

Dr. Khalid A. Hafeez

Spring, 25

Outline

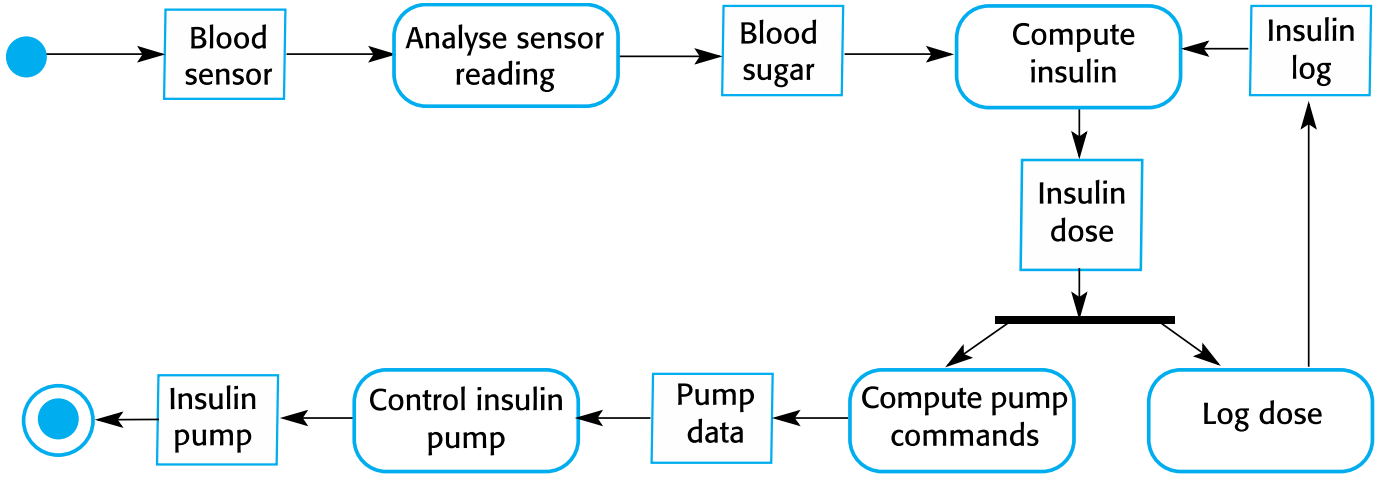
- Case studies
- Dependability properties
- Sociotechnical systems
- Redundancy and diversity
- Dependable processes
- Formal methods and dependability
- Security and Dependability

Case Studies

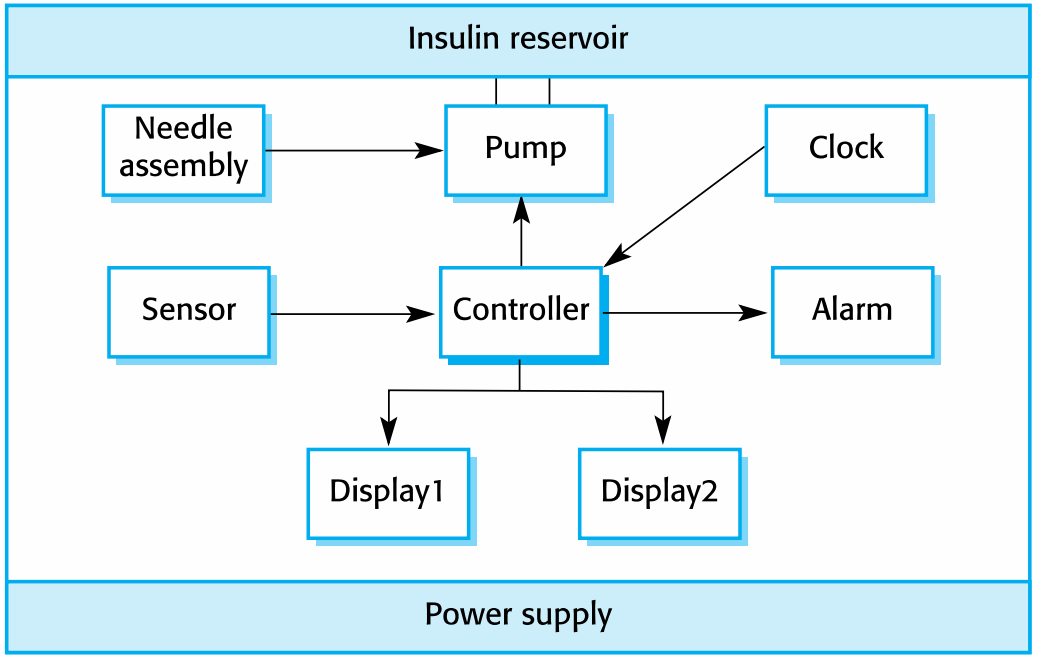
- A personal insulin pump
  - An embedded system in an insulin pump used by diabetics to maintain blood glucose control.
- Mentcare: mental health case patient management system
  - A system used to maintain records of people receiving care for mental health problems.
- A wilderness weather station
  - A data collection system that collects data about weather conditions in remote areas.

Insulin Pump Control System

- Collects data from a blood sugar sensor and calculates the amount of insulin required to be injected.
- Calculation based on the rate of change of blood sugar levels.
- Sends signals to a micro-pump to deliver the correct dose of insulin.
- Safety-critical system as low blood sugars can lead to brain malfunctioning, coma and death; high-blood sugar levels have long-term consequences such as eye and kidney damage.



Activity Model of the Insulin Pump



Hardware Architecture

Insulin Pump Control System

- **Essential High-Level Requirements**
  - The system shall be available to deliver insulin when required.
  - The system shall perform reliably and deliver the correct amount of insulin to counteract the current level of blood sugar.
  - The system must therefore be designed and implemented to ensure that the system always meets these requirements.

Mentcare: A Patient Information System for Mental Health Care

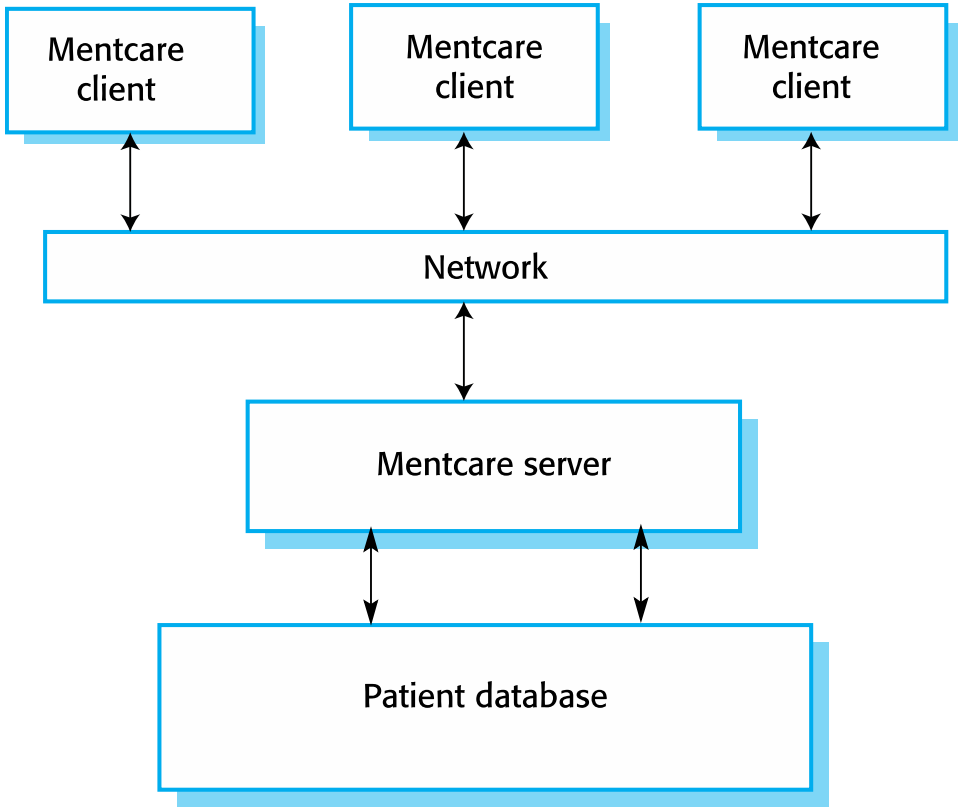
- A patient information system to support mental health care is a medical information system that maintains information about patients suffering from mental health problems and the treatments that they have received.
- Most mental health patients do not require dedicated hospital treatment but need to attend specialist clinics regularly where they can meet a doctor who has detailed knowledge of their problems.
- To make it easier for patients to attend, these clinics are not just run in hospitals. They may also be held in local medical practices or community centres.

- **Mentcare**
  - Mentcare is an information system that is intended for use in clinics.
  - It makes use of a centralized database of patient information but has also been designed to run on a PC, so that it may be accessed and used from sites that do not have secure network connectivity.
  - When the local systems have secure network access, they use patient information in the database, but they can download and use local copies of patient records when they are disconnected.
  - **Goals:**
    - To generate management information that allows health service managers to assess performance against local and government targets.
    - To provide medical staff with timely information to support the treatment of patients.

Mentcare: A Patient Information System for Mental Health Care

- **Key Features of the Mentcare System**
  - **Individual care management**
    - Clinicians can create records for patients, edit the information in the system, view patient history, etc. The system supports data summaries so that doctors can quickly learn about the key problems and treatments that have been prescribed.
  - **Patient monitoring**
    - The system monitors the records of patients that are involved in treatment and issues warnings if possible problems are detected.
  - **Administrative reporting**
    - The system generates monthly management reports showing the number of patients treated at each clinic, the number of patients who have entered and left the care system, number of patients sectioned, the drugs prescribed and their costs, etc.

The Organization of the Mentcare System



Mentcare: A Patient Information System for Mental Health Care

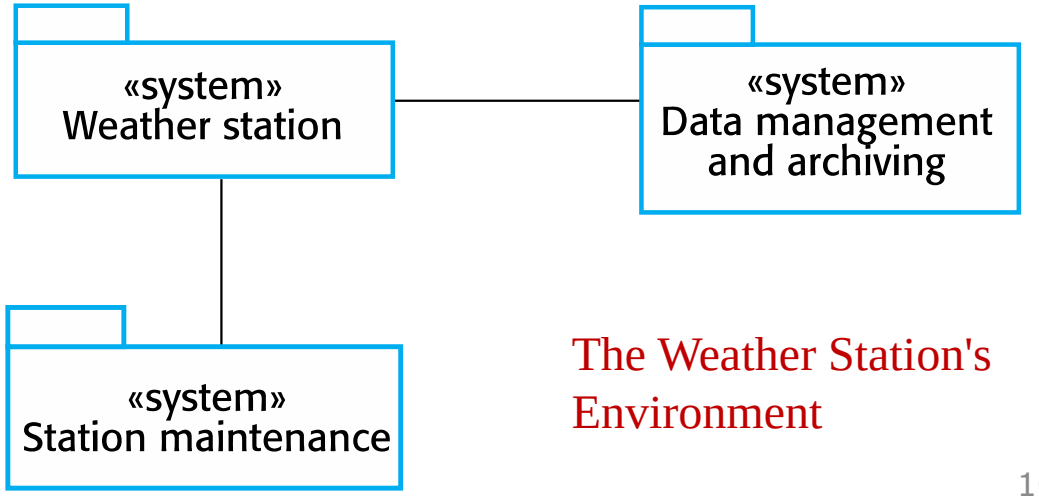
- **Mentcare System Concerns**
  - Privacy
    - It is essential that patient information is confidential and is never disclosed to anyone apart from authorized medical staff and the patient themselves.
  - Safety
    - Some mental illnesses cause patients to become suicidal or a danger to other people. Wherever possible, the system should warn medical staff about potentially suicidal or dangerous patients.
  - Availability
    - The system must be available when needed otherwise safety may be compromised and it may be impossible to prescribe the correct medication to patients.

Wilderness Weather Station

- The government of a country with large areas of wilderness decides to deploy several hundred weather stations in remote areas.
- Weather stations collect data from a set of instruments that measure temperature and pressure, sunshine, rainfall, wind speed and wind direction.
  - The weather station includes a number of instruments that measure weather parameters such as the wind speed and direction, the ground and air temperatures, the barometric pressure and the rainfall over a 24-hour period.
  - Each of these instruments is controlled by a software system that takes parameter readings periodically and manages the data collected from the instruments.

Wilderness Weather Station

- **Weather Information System**
  - The weather station system
    - This is responsible for collecting weather data, carrying out some initial data processing and transmitting it to the data management system.
  - The data management and archiving system
    - This system collects the data from all of the wilderness weather stations, carries out data processing and analysis and archives the data.
  - The station maintenance system
    - This system can communicate by satellite with all wilderness weather stations to monitor the health of these systems and provide reports of problems.



The Weather Station's Environment

Wilderness Weather Station

- **Additional Software Functionality**
  - Monitor the instruments, power and communication hardware and report faults to the management system.
  - Manage the system power, ensuring that batteries are charged whenever the environmental conditions permit but also that generators are shut down in potentially damaging weather conditions, such as high wind.
  - Support dynamic reconfiguration where parts of the software are replaced with new versions and where backup instruments are switched into the system in the event of system failure.

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 of **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.

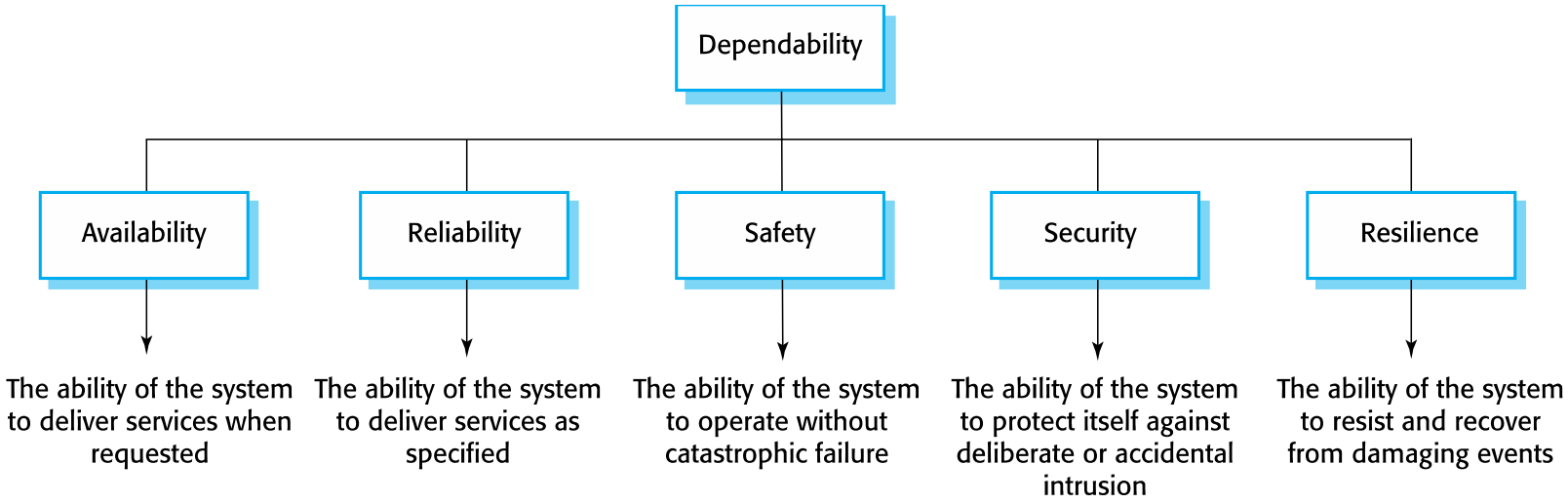
System Dependability

- **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

- **Main properties:**

1. **Availability**: The probability that the system will be up and running and able to deliver useful services to users.
2. **Reliability**: The probability that the system will correctly deliver services as expected by users.
3. **Safety**: A judgment of how likely it is that the system will cause damage to people or its environment.
4. **Security**: A judgment of how likely it is that the system can resist accidental or deliberate intrusions.
5. **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.



Dependability Properties

- **Other properties:**

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

Dependability Properties

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

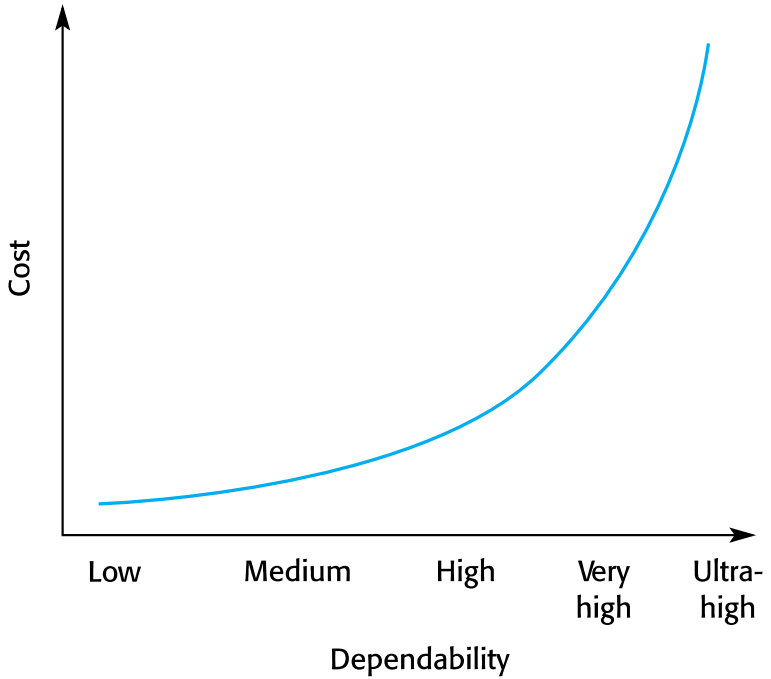
Dependable Software

- **How to develop dependable software:**
  1. Avoid the introduction of accidental errors when developing the system.
  2. Design verification and validation processes that are effective in discovering residual errors in the system.
  3. Design systems to be fault tolerant so that they can continue in operation when faults occur
  4. Design protection mechanisms that guard against external attacks.
  5. Configure the system correctly for its operating environment.
  6. Include system capabilities to recognize and resist cyberattacks.
  7. Include recovery mechanisms to help restore normal system service after a failure.

Dependable Software

- **Cost to develop a dependable software:**
  - 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.

- Because of very high costs of dependability achievement, 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

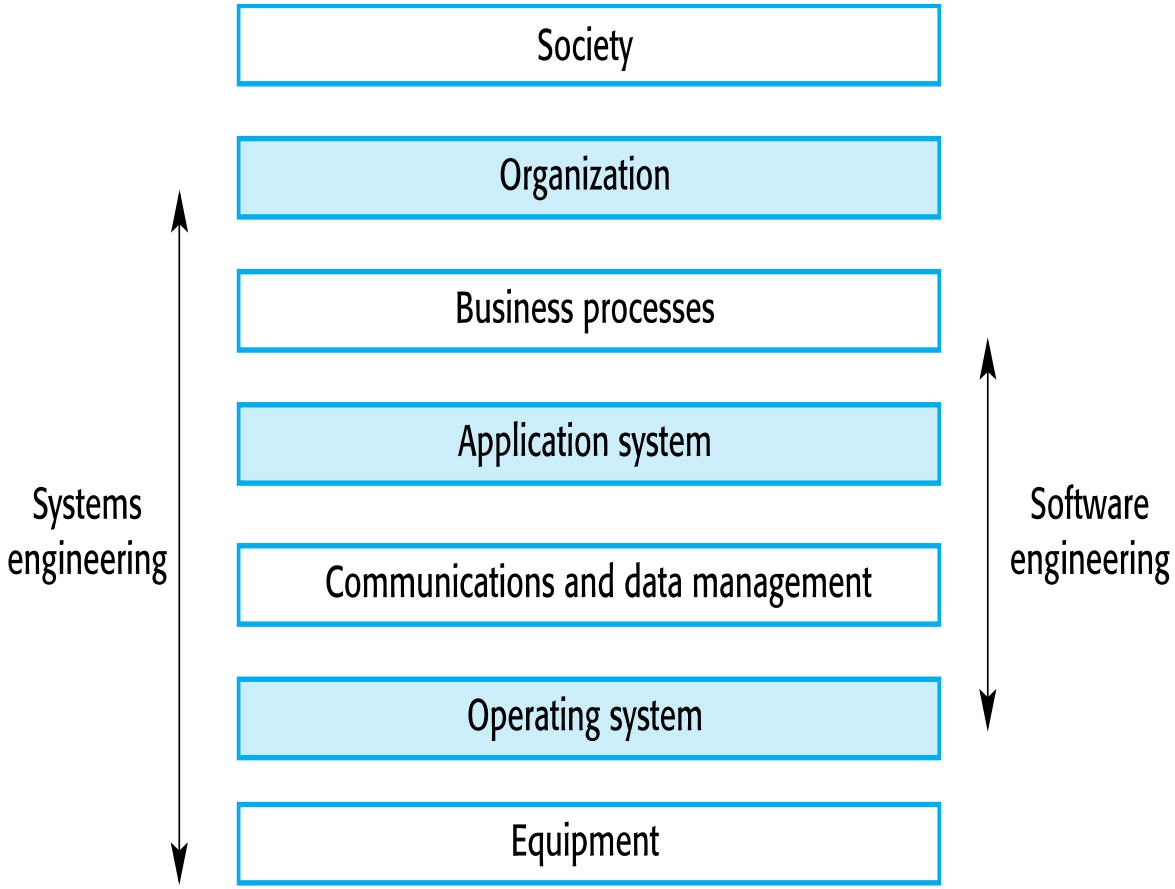


Sociotechnical Systems

- **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:

- Sociotechnical systems are so complex that it is impossible to understand them as a whole. Rather, you have to view them as layers.
- **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.



The Sociotechnical Systems Stack:

- **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.

Redundancy and Diversity

- **Strategies to achieve and enhance dependability rely on both redundancy and diversity**
  - **Redundancy**: Keep more than a single version of critical components so that if one fails then a backup is available.
    - Example: Where availability is critical (e.g. in e-commerce systems), companies normally 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.
    - Example: To provide resilience against external attacks, different servers may be implemented using different operating systems (e.g. Windows and Linux)
  - Redundant and diverse components should be independent so that they will not suffer from 'common-mode' failures
    - For example, components implemented in different programming languages means that a compiler fault will not affect all of them.
  - Adding diversity and redundancy to a system **increases** the system complexity which **increases** the chances of error.

Dependable Processes

- To ensure a minimal number of software faults, it is important to have a well-defined, repeatable software process.
- A well-defined repeatable process is one that does not depend entirely on individual skills; rather can be enacted by different people.
- Regulators use information about the process to check if good software engineering practice has been used.
- For fault detection, it is clear that the process activities should include significant effort devoted to verification and validation.
- **Dependable Process Characteristics:**
  - **Explicitly defined**
    - A process that has a defined process model that is used to drive the software production process. Data must be collected during the process that proves 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.

Dependable Processes

- **Attributes of Dependable Processes:**

Process characteristic	Description
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

- Examples of activities that might be included in a dependable process:
  1. Requirements reviews to check that the requirements are, as far as possible, complete and consistent.
  2. Requirements management to ensure that changes to the requirements are controlled and that the impact of proposed requirements changes is understood.
  3. Formal specification, where a mathematical model of the software is created and analyzed.
  4. System modeling, where the software design is explicitly documented as a set of graphical models, and the links between the requirements and these models are documented.
  5. Design and program inspections, where the different descriptions of the system are inspected and checked by different people.
  6. Static analysis, where automated checks are carried out on the source code of the program.
  7. Test planning and management, where a comprehensive set of system tests is designed.
     - The testing process must be carefully managed to demonstrate that these tests provide coverage of the system requirements and have been correctly applied in the testing process.

Dependable Processes

- Dependable Processes and Agility:
  - Dependable software often requires certification so both process and product documentation has to be produced.
  - Up-front requirements analysis is also essential to discover requirements and requirements conflicts that may compromise the safety and security of the system.
  - These conflict with the general approach in agile development of co-development of the requirements and the system and minimizing documentation.
  - An agile process may be defined that incorporates techniques such as iterative development, test-first development and user involvement in the development team.
  - So long as the team follows that process and documents their actions, agile methods can be used.
  - However, additional documentation and planning is essential so ‘pure agile’ is impractical for dependable systems engineering.

Formal Methods and Dependability

- Formal Specification:
  - Formal methods are approaches to software development that are based on mathematical representation and analysis of software.
  - Formal methods include
    - Formal specification;
    - Specification analysis and proof;
    - Transformational development;
    - Program verification.
  - Formal methods significantly reduce some types of programming errors and can be cost-effective for dependable systems engineering.

Security and Dependability

- **Security and reliability**
  - If a system is attacked and the system or its data are corrupted, then this may induce system failures that compromise the reliability of the system.
- **Security and availability**
  - A common attack on a web-based system is a denial of service attack, where a web server is flooded with service requests from a range of different sources. This attack will make the system unavailable.
- **Security and safety**
  - An attack that corrupts the system or its data means that assumptions about safety may not hold. Safety checks rely on analysing the source code of safety critical software and assume the executing code is a completely accurate translation of that source code. If this is not the case, safety-related failures may be induced and the safety case made for the software is invalid.
- **Security and resilience**
  - Resilience is a system characteristic that reflects its ability to resist and recover from damaging events. The most probable damaging event on networked software systems is a cyberattack of some kind so most of the work now done in resilience is aimed at deterring, detecting and recovering from such attacks.

Secure Systems Design

- **Design Compromises**
  - 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.

Design Guidelines for Secure Systems Engineering

1. Base decisions on an explicit security policy
   - Define a security policy for the organization that sets out the fundamental security requirements that should apply to all organizational systems.
2. Avoid a single point of failure
   - Ensure that a security failure can only result when there is more than one failure in security procedures. For example, have password and question-based authentication.
3. Fail securely
   - When systems fail, for whatever reason, ensure that sensitive information cannot be accessed by unauthorized users even although normal security procedures are unavailable.
4. Balance security and usability
   - Try to avoid security procedures that make the system difficult to use. Sometimes you have to accept weaker security to make the system more usable.
5. Log user actions
   - Maintain a log of user actions that can be analyzed to discover who did what. If users know about such a log, they are less likely to behave in an irresponsible way.

Design Guidelines for Secure Systems Engineering

6. Use redundancy and diversity to reduce risk
   - Keep multiple copies of data and use diverse infrastructure so that an infrastructure vulnerability cannot be the single point of failure.
7. Specify the format of all system inputs
   - If input formats are known, then you can check that all inputs are within range so that unexpected inputs don't cause problems.
8. Compartmentalize your assets
   - Organize the system so that assets are in separate areas and users only have access to the information that they need rather than all system information.
9. Design for deployment
   - Design the system to avoid deployment problems
10. Design for recoverability
    - Design the system to simplify recoverability after a successful attack.