Chapter 1

Introduction

Software Engineering

Software engineering is the discipline of designing, implementing, and maintaining software by applying engineering principles to the software development process.

Why do we need software engineering?

Software engineering is needed

- to predict time, effort, and cost
- to improve software quality
- to improve maintainability
- to meet increasing demands
- to lower software costs
- to successfully build large, complex software systems
- to facilitate group effort in developing software
- More and more systems are software controlled make it more readable

Why apply Software Engineering to Systems?

Applying software engineering to systems ensures they are reliable, maintainable, and scalable.It helps manage resource constraints and integration complexity, leading to better system performance and usability.

Challenges in Software Development

1) Software development struggles to keep pace with hardware advancements and rising expectations.
2) Aging software poses challenges in terms of maintenance and updates.
3) Estimating software costs and schedules is difficult.
4) Many software projects fail to meet their objectives. Examples are:
    -> Arianne Missile
    -> Denver Airport Baggage System
    -> Therac

Software Engineering Fundamentals

- Managed development process: Use a defined process to guide development.
- Dependability and performance: Ensure systems are reliable and efficient.
- Requirements management: Clearly understand and manage software requirements.
- Software reuse: Utilize existing software components to save time and effort.

Software Engineering Diversity

- No universal set of software engineering techniques applies to all types of systems.
- Methods and tools used depend on:
  -> Application type
  -> Customer requirements
  -> Development team background

Software Cost

The cost of software includes the expenses involved in designing, developing, testing, deploying, and maintaining software. Factors influencing cost include complexity, hardware, labor, and quality demands. Software engineering aims for cost-effective development.

Software Project Failures

- Software projects can fail due to increasing system complexity and the lack of software engineering methods.
- Complex systems require faster delivery and new capabilities, while companies without software engineering expertise may develop unreliable and expensive software.

Essential Attributes of Software

Essential attributes of good software include:

- Maintainability: Easy to update and modify.
- Correctness: Free from defects and performs as intended.
- Usability: Easy to understand and use.
- Efficiency: Doesn't waste resources.
- Reliability: Operates consistently and predictable

Software Products

Generic products	Customized products
• Stand-alone software sold to any customer • Software developer owns the specification and makes decisions on changes **Examples:** PC software, CAD software, dentist appointment systems	• Software developed for specific customer needs • Customer owns the specification and makes decisions on changes **Examples:** Embedded control systems, air traffic control software, traffic monitoring systems

Software Process Activities

- **Software specification**: defining the software to be produced and its constraints
- **Software development:** designing and programming the software
- **Software validation:** checking the software to ensure it meets customer requirements
- **Software evolution:** modifying the software to reflect changing customer and market requirements

Application Types

1. **Stand-alone applications:**
    - Run on local computers and do not require network connectivity.
2. **Interactive transaction-based applications:**
    - Accessed remotely from user PCs or terminals.
    - Include web applications like e-commerce platforms.
3. **Embedded control systems:**
    - Software systems that control and manage hardware devices.
    - Widely used in various industries.

4. **Batch processing systems:**

    - Business systems that process large batches of data.
    - Create outputs corresponding to individual inputs.

5. **Entertainment systems:**

    - Designed for personal use and entertainment.

6. **Systems for modeling and simulation:**

    - Developed by scientists and engineers to model physical processes or situations.

7. **Data collection systems:**

    - Collect data from sensors and transmit it for processing.

8. **Systems of systems:**

    - Composed of multiple interconnected software systems

Internet Software Engineering

1. Web-based systems are complex distributed systems.
2. Fundamental software engineering principles still apply.
3. Key techniques:

    - Software reuse: Assembling systems from pre-existing components.
    - Incremental and agile development: Delivering systems in stages.
    - Service-oriented systems: Using stand-alone web services as software components.
    - Rich interfaces: Creating user-friendly interfaces using AJAX and HTML5.

Case Studies

1. **Insulin Pump Control System**

    - Embedded system that delivers insulin to diabetics based on blood sugar levels.
    - Critical systems with incorrect insulin dosage can be life-threatening.

2. **Mentcare: Patient Information System for Mental Health Care**

    - Maintains patient records and provides treatment information.
    - Used in clinics with secure networks or on PCs with local copies of records.
    - Key concerns: privacy, safety, and availability.

3. **Wilderness Weather Station**

    - Collects weather data from remote areas.
    - Includes instruments that measure temperature, pressure, rainfall, wind speed, and direction.
    - Additional software functionality: monitoring, power management, and reconfiguration.

4. **iLearn: Digital Learning Environment**

    - Framework for learning tools and applications.
    - Service-oriented system with replaceable components.
    - Includes utility, application, and configuration services.
    - Services can be integrated or independent.