Week 1 Notes: Introduction to Software Engineering

Learning Objectives:

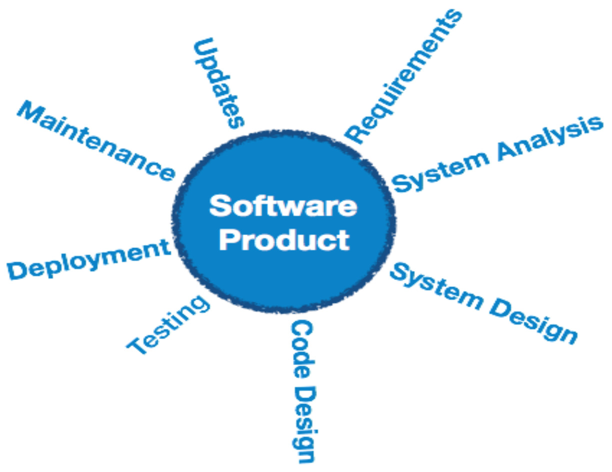
1. Understand the role and importance of software engineering in modern society.
2. Recognize the diversity of software systems and the need for tailored engineering practices.
3. Identify ethical considerations and responsibilities in software development.

1. What is Software Engineering?

Definition:
Software engineering is an engineering discipline concerned with all aspects of software production, including the design, development, validation, deployment, and maintenance of software systems.

Software is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called **software product.**

Engineering on the other hand, is all about developing products, using well-defined, scientific principles and methods.



Software engineering is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable software product.

- **Key Attributes of Software Engineering:**
    - **Systematic Approach:** Software engineering employs structured methods and processes to ensure quality outcomes.
    - **Focus on Quality:** Aims to produce reliable, efficient, maintainable, and secure software systems.
    - **Lifelong Maintenance:** Involves software evolution to meet changing user and market demands.

---

2. Importance of Software Engineering

Software systems are integral to nearly every aspect of modern life, underpinning industries such as healthcare, transportation, education, and communication.

Examples of software-dependent systems:

- **Critical Systems:** Control systems in airplanes or medical devices.
- **Interactive Systems:** Web applications, mobile apps.
- **Embedded Systems:** Devices such as microwave ovens or vehicle control systems.

---

3. Characteristics of Good Software

High-quality software exhibits the following attributes (Figure 1.2 from Sommerville's *Software Engineering*):

1. **Dependability and Security:**
    - Software must operate reliably, ensuring no harm or loss occurs due to failures.
    - Security is essential to prevent unauthorized access or damage.
2. **Efficiency:**
    - Should optimize resource use, including processing time and memory.
3. **Maintainability:**
    - Must be easy to adapt for new requirements or fix errors.
4. **Usability:**
    - Must be user-friendly and compatible with other systems.

---

4. Software Engineering vs. Programming

- **Programming** focuses on writing code to solve specific problems.
- **Software Engineering** involves:
  - Understanding user needs.
  - Designing a scalable and maintainable solution.
  - Testing, deployment, and long-term system evolution.
  - Managing teams, costs, and project timelines.

---

5. Types of Software Systems

Software systems vary greatly in complexity and purpose. Sommerville categorizes them as follows:

1. **Stand-Alone Applications:** Run on personal devices without network dependencies (e.g., word processors).
2. **Interactive Systems:** Include web applications accessed remotely.
3. **Embedded Systems:** Operate within hardware devices like vehicles or appliances.
4. **Batch Processing Systems:** Process data in bulk, such as payroll systems.
5. **Entertainment Systems:** Games and multimedia software.
6. **Data Collection Systems:** Gather data from sensors, often in remote environments.
7. **Systems of Systems:** Combinations of multiple independent systems working together.

---

6. Software Development Challenges

Despite advancements, software projects face persistent challenges:

- **Increasing Complexity:** Modern systems require more functionality and faster delivery.
- **Changing Requirements:** User needs and market demands often evolve during development.
- **Security Risks:** Software must be robust against malicious attacks.
- **Cost and Time Constraints:** Projects often overrun budgets or miss deadlines due to poor planning or unforeseen issues.

---

7. Ethical Considerations in Software Engineering

As a professional discipline, software engineering operates within ethical and social frameworks. Engineers have a responsibility to ensure their work benefits society and minimizes harm.

Key Ethical Principles (ACM/IEEE Code of Ethics):

1. **Public Interest:** Prioritize user safety and welfare.
2. **Client and Employer:** Act in their best interests while adhering to ethical standards.
3. **Product Quality:** Ensure high professional standards in all deliverables.
4. **Judgment:** Maintain integrity in decision-making.
5. **Colleagues:** Collaborate fairly and respect team members.
6. **Lifelong Learning:** Commit to continual skill development and ethical practices.

---

8. A Brief History of Software Engineering

The discipline emerged in the late 1960s in response to the "software crisis" where large systems often failed due to complexity, cost overruns, and reliability issues. Key developments include:

- **1970s and 1980s:** Introduction of structured programming and object-oriented development.
- **1990s and 2000s:** Agile methods, web-based systems, and open-source software became widespread.
- **Recent Trends:** Cloud computing, AI integration, and DevOps emphasize continuous delivery and scalability.

---

9. Case Studies in Software Engineering

To contextualize software engineering principles, Sommerville introduces the following real-world examples:

1. **Insulin Pump Control System:** Embedded software ensuring precise delivery of medication.
2. **Patient Information System:** Used in mental health care to manage records and monitor treatments.
3. **Wilderness Weather System:** Collects and analyzes data from remote weather stations.
4. **Digital Learning Environment:** Supports student learning in schools through online platforms.

---

10. Key Takeaways

- Software engineering is vital for creating robust, reliable, and maintainable systems.
- Ethical considerations and societal impact are central to professional software practices.
- Understanding the diversity of software systems helps engineers tailor their methods and tools effectively.

Discussion Questions

1. Why is software engineering critical in modern society?
2. How do ethical principles apply to real-world software projects?
3. Compare the challenges faced by embedded systems and web-based applications.

Practical Activity

- **Objective:** Explore the diversity of software systems.
- **Task:** In groups, identify five software systems you interact with daily. Classify them into the types listed in this lecture (e.g., stand-alone, interactive, embedded) and discuss their unique engineering challenges.