

Foundations of Software Engineering

(CPSC 362)

An Overview of Software Engineering

Computer Software

Categories of Software

- **System software**
 - Operating systems
 - Database management systems
 - Other server systems
- **Embedded software**
 - Software embedded in hardware components for control
- **Application software**
 - Product-line software
 - Business/Engineering/scientific software
- **Internet applications**
 - Web sites, smartphone applications
 - **Unique characteristics of Internet applications**
 - Network intensiveness
 - Concurrency and Unpredictable load
 - Performance (fast response to many users) and availability (24/7 uptime)
 - Data driven and content sensitive
 - Continuous evolution

Modern Computing Trend

- Internet computing
- Ubiquitous computing
- Cloud computing – Service oriented computing
 - Software as a service
 - Data as a service
 - Platform as a service
- Social computing
- Open-world computing
 - Open-source projects
 - Open platform

Software Quality and Challenges

Common Industry's Concern about Software

- Why does it **take so long** to get software finished?
- Why are **development costs** so high?
- Why can't we find all **errors** before we give the software to our customers?
- Why do we **spend so much time** and effort **maintaining** existing programs?
- Why can't we easily **measure the progress** as software is being developed and maintained?

Software Myths

- **Management myths**

- Don't we already have books with full of standards and procedures for building software?
- If we get behind schedule, we can add more programmers to catch up.
- If we outsource the software project to a third party, we can just relax and let them build it.

- **Customer myths**

- A general statement of objectives is sufficient to begin writing programs. We can fill in the details later.
- Project requirements continually change, but change can be easily accommodated because software is flexible.

- **Practitioners' myths**

- Once we write the program and get it to work, our job is done.
- Until I get the program running, I have no way of assessing its quality.
- The only deliverable work product for a successful project is the working program.
- Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.

Software Evolution and Its Impacts

- Legacy software - Hundreds of existing software
- **Common reasons for evolution in software industry**
 - Must be enhanced to implement new business requirements
 - Must be adapted to meet the needs of new systems, computing, network environment, and technologies, requiring re-design and – architecture
- **Software crisis** due to large and complex software systems with increasing demand, changing technologies, and market, causing:
 - Project running over-budget and/or over-time (missing the deadline)
 - Issue of software quality (questions about what is delivered)
 - Un-manageable and un-maintainable

Challenges of Software Quality

- **The key characteristics of quality software**
 - **Meeting the requirements** (functional and non-functional)
 - **Easy to maintain** (easy to understand, modify, and extend)
 - **Reusable** (easy to integrate, related to extension and maintenance)
- **Challenging factors for software quality** (Manny Lehman 1997)
 - The law of continuing change
 - The law of continuing growth
 - The law of increasing complexity
 - The law of declining quality
- **How can we improve the software quality?**

Experience from Manufacturing Industry

- Manufacturing industry experienced the same problems
 - Improved product quality using good process for quality control (e.g., six sigma)
- **Characteristic differences in Software**
 - Software is developed, not manufactured.
 - Most software continues to be custom built.
 - Customer requirements are changing even after deployment.
- Some people argue that “Software” is different!
- But software is now almost everywhere!

Software Quality Improvement Initiatives

- **Advancements**

- Improved software processes
- Advanced software engineering methodologies
- Use of tools for automation

- **Process improvement initiatives**

- CMMI (Capability Maturity Model Integration) by SEI (Software Engineering Institute) operated by Carnegie Mellon University sponsored by Depart of Defense (DoD)
- ISO, IEEE standards
- Enterprise Architecture framework (EAF)

Software Engineering

Definitions of Software Engineering

- **“Software engineering** is the establishment and use of **sound engineering principles** in order to obtain **economically** software that is **reliable** and works **efficiently** on real machines.” (Fritz Bauer 1969)
 - “sound engineering principles” – development method
 - “economically” – cost
 - “reliable”, “efficiently” – quality
- **“Software engineering: (1)** The application of a **systematic, disciplined, quantifiable** approach to the **development, operation, and maintenance** of software; that is, the **application of engineering** to software **(2)** The study of approaches as in (1).” (IEEE 1993)
 - “systematic, disciplined, quantifiable” – development method
 - “development, operation, and maintenance” – development process
 - “application of engineering” – quality and cost

The Goal of Software Engineering

- The **Ultimate goal**

- To deliver **quality software** (customer value) **timely** (within the deadline) and **economically** (within the budget) delivered to customers

- The **characteristics of quality software**

- Meeting the requirements (functional, non-functional)
- Easy to maintain (easy to understand, modify, and extend)
- Reusable (easy to integrate and maintain)

- **How to achieve the goal**

- **Process** – A “framework” of software development
- **Methods** – Engineering techniques “how” to build software
- **Tools** – Automated “support” for the process and the methods to improve the quality and productivity

Software Process

- **Definitions of process**

- A system of operations in producing something; a series of actions, changes, or functions that achieve an end or a result (Webster)
- A sequence of steps performed for a given purpose (IEEE)
 - A process defines who is doing what, when, and how to reach a certain goal (Pressman)

- **Definition of software process**

- A set of activities, methods, practices, and transformations used to develop and maintain software and the associated products (e.g., project plans, design documents, code, test cases, user manuals, etc.) (SEI)

Core Software Engineering Activities

- Feasibility analysis of the problem
- Planning and initiation
- Requirement collection, analysis, and specification
- Design and architecture
- Implementation
- Testing
- Deployment
- Operation and maintenance

Software Development Methods

- **Evolution of software development**

- Machine codes and assembly codes
 - Difficult to write even for a small function
- Programs written in high-level language
 - For easier to write, understand, and maintain, compared to assembly codes
 - Modularization using functions and procedures
- Structured programming
 - A systematic way of modularization
- Object-oriented programming
 - Improved modularization, maintenance, integrate, reuse
- Component-based software development
 - Advanced modularization, maintenance, integrate, reuse
- Service-oriented software
 - Even more advanced modularization, maintenance, integrate, reuse
- Cloud computing
 - Reuse (share) of services, platform, and infrastructure

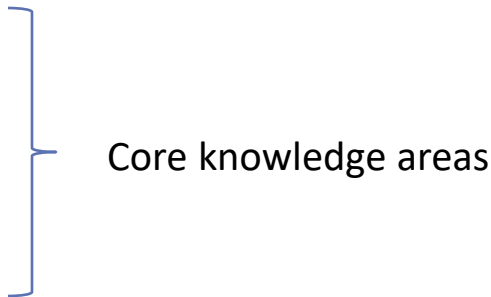
- **Most modern software systems are large scale.**

Software Development Tools

- IDE such as Visual Studio Code, PyCharm, etc.
- Application frameworks
 - React, Express, Django, Flask, ASP.NET, Spring, etc.
- Version control system such as Git, GitHub
- Software process and management tools such as Jira
- Third-party libraries
- Open-source projects
- Cloud infrastructures
- Deployment tools

Software Engineering Body of Knowledge

(SEBOK) areas, IEEE

- Software Requirements
 - Software Design
 - Software Construction
 - Software Testing
 - Software Maintenance
 - Software Engineering Process
 - Software Quality
 - Software Configuration Management
 - Software Engineering Management
 - Software Engineering Tools and Methods
- 
- Core knowledge areas

Areas of Software Engineering and Courses Offered by Our Department

- **Courses**

- Foundations of Software Engineering (CPSC 362)
- Software standards and requirements (CPSC 541)
- Software process (CPSC 466, 544)
- Software verification and validation (CPSC 463, 542)
- Software design and architecture (CPSC 462, 464 and 545)
- Software measurement (CPSC 547)
- Project management (CPSC 546)
- Software maintenance (CPSC 543)

- **M.S degree programs**

- M.S in Software Engineering (MSE) – a highly ranked online program
- Accelerated M.S in Software Engineering

Key Topics to Discuss in this class

- **Software process**
 - Concepts of iterative process
 - Agile process (Scrum, Kanban)
 - Use of toolchains
- **Requirements** (analysis, modeling, and writing requirements)
 - Domain modeling (requirements analysis and modeling)
 - Use case and user story (writing requirements)
- **Software design and architecture** (for quality and maintenance)
 - Design principles
 - Design patterns
- **Implementation**
 - Object-Oriented programming
 - Programming design principles and patterns
 - Refactoring and continuous integration
- **Testing**
 - Test-Driven Development (TDD)
- **Deployment**
 - Continuous release and deployment
- **Other software engineering topics**
 - Configuration management, project management, quality control, and process improvement

References

- Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 7th edition, McGraw-Hill High Education, 2010.
- M. Page-Jones, *Fundamentals of Object-Oriented Design in UML*, Addison-Wesley, 2000.
- Booch, G., Rumbaugh, J., and Jacobson, I., *The Unified Modeling Language User Guide*, Addison-Wesley, 1999.
- E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Boston, MA, 1995.
- Partha Kuchana, *Software Architecture Design Patterns in Java*, Auerbach Publications, 2004.
- C. Larman, *Agile and Iterative Development: A Manager's Guide*, (The Agile Software Development Series), Addison-Wesley, 2003.
- J. Highsmith, *Agile Project Management*, 2nd edition, Addison-Wesley, 2010.