Software Engineering

Software Engineering

Definition 3.2760 from ISO/IEC/IEEE 24765:2010(E)

1. the systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software. ISO/IEC 2382-1:1993, Information technology – Vocabulary – Part 1: Fundamental terms.01.04.07.
2. the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

The Definition Expanded

the systematic application of . . . methods . . . [and] disciplined, quantifiable approach to the development, operation, and maintenance of software

- Software development life cycle
- Software development process models

application of scientific and technological knowledge

- Software design principles
- Programming languages
- Software development tools

the systematic application of . . . experience

- Process improvement frameworks

Software Development Life Cycle (SDLC)

All software development projects go through identifiable phases:

- Planning
- Requirements Analysis
- Design
- Implementation
- Integration
- Testing
- Deployment
- Maintenance

Process models differ in how they approach these phases and organize them into a complete software development project

Planning and Requirements Analysis

Planning

- Identify the need for a software system
- Allocate resources (people, budget, equipment)
- Set a timeline for development

Requirements Analysis

- Identify the users and other stakeholders of the system
- Elicit requirements from the stakeholders: features, performance characteristics, usability requirements

Requirements and planning usually interleaved - requirements drive timelines, resources constrain requirements

Design and Implementation

Design: how the software will be structured to meet the requirements

- High-level architecture, e.g., client-server, desktop application, web application
- Component design using object-oriented design, entity-relationship modeling, etc

Implementation: writing the code to realize the design in a working system

- Programming
- Building
- Art and UI (icons, style sheets, dialog layouts, etc)

Integration and Testing

Integration: putting the components together

- Make sure software components work together
- Make sure software integrates with host operating system

Testing: verifying that the software works as expected

- Some tests done by developers (unit tests, some functional tests)
- Some tests done by quality assurance engineers and customer (functional tests, acceptance tests)

Deployment and Maintenance

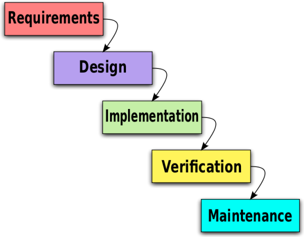
Deployment: putting the software in the hands of its users

- How to deploy

Maintenance: fixing bugs and adding enhancements or new features after the software has been deployed

- Enhancements and bug fixes for current release
- Development of new version

Software Development Process Models



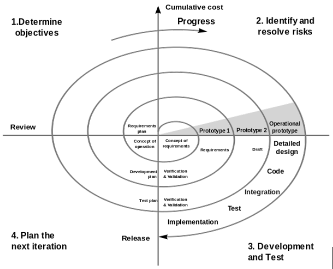
[^1]

Two stereotypical process models: waterfall and iterative

- Waterfall processes, a.k.a. sequential processes, finish each phase of the SDLC before moving on to the next
    - Sometimes called "big bang" development, since in classic waterfall the system under development is not released until the end of the project
    - Incremental waterfall processes include intermediate releases in the implementation phase

[^1] https://en.wikipedia.org/wiki/File:Waterfall_model_(1).svg

Iterative Development



[^1]

- Divide the project into short (typically two-week) iterations
- Each iteration progresses through each of the SDLC phases
  - Each iteration accomplishes a subset of the requirements and releases a working product

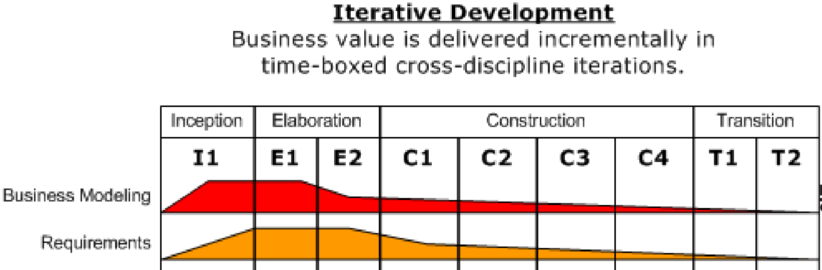
[^1] https://en.wikipedia.org/wiki/File:Spiral_model_(Boehm,_1988).svg

Rational Unified Process

Developed by Rational Software and acquired by IBM in 2003

Unifies waterfall and iterative process models with four life-cycle phases:

- Inception: feasibility - vision, scope, rough estimates
- Elaboration: most requirements, more detailed estimates, implementation of core architecture and highest risk features
- Construction: implementation of remaining features, iterative refinement of requirements and estimates
- Transition: beta tests, deployment



Software Design Science

Programs = data structures + algorithms

What software engineers learn in school

- Computer science
  - Data structures and algorithms
  - Programming languages
  - Object-oriented programming, Functional programming
  - Systems, networks, HCI, AI (threads)
- Software design and implementation
  - Design patterns
  - Modeling approaches and languages (like UML)
  - Programming

Software Development Practice

Programming tools

- Editors, debuggers, profilers

Build tools

- Make, SCons, Ant, Maven, SBT, Gradle, Buildr, Rake

Integration tools

- Test runners, installer software, continuous integration servers

Deployment and maintenance tools

- Software configuration management (CVS, Subversion, Git)
- Bug trackers (Bugzilla, Trac, GitHub)
- Application/web servers (Apache httpd, Tomcat, . . .)
- Containers (Docker, . . .)
- Cloud

ISO 9001

A generic quality management standard with a process-based management approach

- International standard based on british standard dating back to 1987 (current version is 2000)
- Adopted by many industries: aviation, automotive, software
- Based on 8 quality principles from ISO 9000:
- Customer focus
- Leadership
- Involvement of people
- Process approach
- System approach to management Continual improvement
- Factual approach to decision making
- Mutually beneficial supplier relationships

Capability Maturity Model (CMM/CMMI)

Developed by Carnegie Mellon's Software Engineering Institute - originally for software engineering, now generically covers acquisition development, and services (and people)

- Models include goals, practices organized into practice areas
- Appraisals grade organizations for capability levels (0 through 3) in each process area, and maturity levels (1 through 5)
- CMMI documented in zillions of pages of engagingly written documents and books. Consult them if you're fortunate enough to be implementing CMMI.

Conclusion

The engineering of software encompasses process and practice

- Process - documentation, project management
- Practice - software architecture, design, implementation, tools and technologies

In this class we focus on

- Agile software development process
- Object-oriented and functional design
- State of the art technologies and tools