Software Engineering

- Software is everywhere nowadays
- SW performs functions
- SW solves certain tasks, problems
- Needs to be written ---> “program”

- Programming - preparing a sequence of instructions for a machine/computer
- Programming went from an “art” to an industrial production process (really?)
- The result has become an industrial product!
In the beginning there is the "problem"
Software Engineering

Total Process consists of two separate phases:
- Analysis phase - what really is the problem, can we break it into sub-problems?
- Synthesis phase - design a solution and put it all together.

Some Definitions:
- Method/Technique - formal procedure for producing some result
- Tool - an instrument or automated system for accomplishing something in a better way
- Procedure - a combination of tools and techniques, a recipe, to produce a resultant
- Paradigm - a particular approach, “cooking style”,
SWE Goal

- Designing, developing, producing “high quality software”
- What is “high quality”? 
- Most obvious: freedom from “bugs”:
  - human error, mistake (logical, accidental)
  - fault - design does not match actual intent
  - failure - is caused by a fault when this piece of code is executed
- Zero-defect software - a fine goal but lastly not achievable on a large scale (unfortunately)
- High probability of correctness - “safety critical”
  - e.g. low probability of error
Consequences

High probability of correctness - "safety critical"
- e.g. Probability of error < $10^{-9}$/hour ~ 114,000 years

--- exhaustive testing is impossible !!!
- A successful test is one that finds an error (fault)
- A test that does not detect an error is what?

--- need methods and techniques that eliminate errors and faults during the construction process and with a high degree of probability!

This should result in "Good Software"

Perspectives on Quality

Garvin (1984) describes quality from 5 different perspectives:
- user view - fitness for purpose
- manufacturing - conformance to specifications
- product - inherent product characteristics
- value-based - price to the customer, how much
- transcendental - recognize it but cannot define it

Crucial is the handling of complexity
The Five Attributes of Complex Systems - 1


1. Frequently, complexity takes the form of a hierarchy, whereby a complex system is composed of interrelated subsystems that have in turn their own subsystems, and so on, until some lowest level of elementary components is reached.

2. The choice of what components in a system are primitive is relatively arbitrary and is largely up to the discretion of the observer of the system.
3. Intracomponent linkages are generally stronger than intercomponent linkages. This has the effect of separating the high-frequency dynamics of the components - involving the internal structure of the components - from the low-frequency dynamics - involving interaction among components.

4. Hierarchical systems are usually composed of only a few different kinds of subsystems in various combinations and arrangements.

5. A complex system that works is invariably found to have evolved from a simple system that worked. . . A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over, beginning with a working simple system.

Handling Complexity

High Level of Abstraction
Low Level of Detail

Analysis (What)

“divide et impera”
Divide and Conquer
specification process

Design (How)

Low Level of Abstraction
High Level of Detail
Building Complex Systems

Total System
User Interface

Inter-component connectivity
(large)

assembly process synthesis

Intra-component connectivity
(small)

Subsystem, Components

The Development Process

☞ How to approach a specific problem
  • The artists approach expressive-impressive
  • The architects approach attractive functionality
  • The scientists approach understand why
  • The engineers approach know how to build it

☞ Process:
  • a series of steps involving activities, resources and constraints that produce an intended output or result.
  • A set of ordered tasks, involving methods, techniques and tools.
  • In software frequently called a “life cycle”.

CS/IT 328 Secure Software Construction
Process Phases

- **Phase 1**: Analysis
- **Phase 2**: Design
  - Activities: architectural design, data structures, classes, interface design, module design, program design
- Implementation
- Test & Integration
- Delivery - Modification

Process Characteristics

- Prescribes all major process activities
- Composed of linked subprocesses, defined as a hierarchy, organized so that each subprocess has its own process model
- Uses resources subject to a set of constraints
- Produces intermediate and final products
- Each activity has entry and exit criteria
- Activities are organized in a schedule specifying when one activity ends relative to the others
- Every process has a set of guiding principles that explain the goals of each activity
- Constraints or controls may apply to an activity, resource or product.
Process Phases

Cost of an error:
- Analysis: 1
- Design: 5
- Implementation: 10
- Test & Integration: 20
- Delivery - Modification: 200

Parallel activities and phases
Process Phases

Parallel and overlapping process phases

<table>
<thead>
<tr>
<th>Analysis</th>
<th>Design</th>
<th>Implement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>Design</td>
<td>Implement</td>
</tr>
<tr>
<td>Analysis</td>
<td>Design</td>
<td>Implement</td>
</tr>
</tbody>
</table>

Life Cycle - Process

- So far all models have been “linear” where one phase leads to the next in sequence (“waterfall model”)
- This is typical for well understood engineering disciplines
- Usually not directly applicable to the construction of software products
- In typical software projects we find typically frequent iteration and refinement
- Several life cycle models attempt to grasp this repetitive process characteristic
The V-Model

The V-Model is a lifecycle model for software development. It consists of two V-shaped diagrams, one for the development phase and the other for the verification phase. The development phase includes:

- Requirements Analysis
- System Design
- Program Design
- Coding, Implementation

The verification phase includes:

- Acceptance Tests
- System Testing
- Operation & Maintenance

The development phase starts with Requirements Analysis, which leads to System Design. System Design leads to Program Design, which in turn leads to Coding, Implementation. Coding, Implementation leads back to System Design, completing the V-shape.

The verification phase starts with Acceptance Tests, which leads to System Testing. System Testing leads back to Acceptance Tests, completing the second V-shape. The V-Model validates design and implementation throughout the software lifecycle.

Life Cycle Model for Module Development

The water sprout model is a life cycle model for module development. It consists of:

- Module Specification
- Module Design
- Coding
- Testing
- Generalization
- Aggregation

The model progresses through these stages, with each stage leading to the next, forming a water sprout-like structure. The model iterates through these stages to develop a real-world entity.
Life Cycle model for real-world system

The Spiral Model

Figure 4.3 A spiral model.
Iterative Phases

Parallel and overlapping process phases

Analysis

Evaluate

Design

Test&Integ

Implement

Distribution of Effort

Effort vs. Time

Basic classes

Specific classes

Implementation

Detailed design

Broad design

Software requirements

Analysis of user requirements

Parallel and overlapping process phases
Total Development Effort

Observed distribution of effort of building a large system. More effort should go into the analysis and the design phases.

Figure 4.4 Total division of efforts over time between different activities.

From I. Jacobson
The generic tasks and phase steps

- User requirements analysis
- User requirements specifications >**
- Software requirements specifications >*
- System architecture design - Logical Design
- Detailed design - Program Design >*
- Implementation - Coding
- Unit / Module testing
- Integration testing >*
- System / Acceptance testing >**
- Maintenance / Modification

The question is “How to arrange these individual tasks into a realistic, implementable development process - and what are the intermediate results?”

---

Tacoma Narrows Bridge

An example of a faulty design which lead to a systems crash
Resume

There are innumerably many ways of arranging the individual activities, steps, tasks, subprocesses.

Well understood traditional engineering processes are mostly linear ("Waterfall Model")

More complex processes require incremental cycles and iterative refinement.

Iteration should involve the customer and future user to evaluate the intermediate performance against the user expectations!

This is a strong argument for building a prototype.

But it may result in a change of the requirements - source of many failed systems!

Process Modeling Techniques and Tools - desirable properties

1. Facilitate human understanding and communication - easily understandable representation, necessary and sufficient information
2. Support process improvement - identify essential components, allow reuse, estimate impact of change
3. Support process management - allow project-specific process, support planning, forecasting, monitoring and measuring process characteristics
4. Provide automatic guidance - define the development environment, provide guidance, retain reusable representations
5. Support automated process execution - support cooperative work, capture relevant data, enforce rules
Phases of the Development Cycle

- **Inception**: define the scope of the project
- **Elaboration**: plan the project, specify features, create system baseline
- **Construction**: build the product
- **Transition**: transfer the product to the user community

Phases and Milestones

<table>
<thead>
<tr>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Objective milestones</strong></td>
<td><strong>Architecture milestones</strong></td>
<td><strong>Operational capability milestones</strong></td>
<td><strong>Product release</strong></td>
</tr>
</tbody>
</table>

“Who is doing what and when” - activities
Workflow Activities/Disciplines

- Engineering
- Requirements
- Analysis and Design
- Implementation
- Test and Integrate
- Management
- Configuration and Change Management
- Project Management

Use Case Model Design
Analysis, Design Model
Implementation Model
Test Model
Project Repository
Schedule

Two-dimensional Model

<table>
<thead>
<tr>
<th>Workflow Activities</th>
<th>Life Cycle Phases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>Inception</td>
</tr>
<tr>
<td>Analysis and Design</td>
<td>Elaboration</td>
</tr>
<tr>
<td>Implementation</td>
<td>Construction</td>
</tr>
<tr>
<td>Test and Integrate</td>
<td>Transition</td>
</tr>
<tr>
<td>Management Disciplines</td>
<td></td>
</tr>
<tr>
<td>Configuration and Change Management</td>
<td></td>
</tr>
<tr>
<td>Project Management</td>
<td></td>
</tr>
</tbody>
</table>
Software -Process and -Life Cycle

Key words:

Software life-cycle
- Phases
- Development cycle and evolution cycle

Software process
- Discipline, activities, roles, and artifacts

Iterations
- Milestones

Understanding the Development and Evolution of Software

- Defining the Software Life Cycle and the Software Process
- Defining Elements of the Software Process
- Integrating the Software Life Cycle and the Software Process
- Customizing the Software Process
Software Process and Software Life Cycle

- Understanding the Development and Evolution of Software

- Defining the Software Life Cycle and the Software Process
  - The Software lifecycle
  - The Software process

- Defining Elements of the Software Process

- Integrating the Software Life Cycle and the Software Process

- Customizing the Software Process

---

Phases of the Development Cycle

- Inception - Define the scope of project
- Elaboration - Plan project, specify features, baseline system
- Construction - Build the product
- Transition - Transition the product into end user community
Each Phase is Marked by Milestones

<table>
<thead>
<tr>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development cycle</td>
<td>Objective Milestone</td>
<td>Initial Operational Capability Milestone</td>
<td>Product Release</td>
</tr>
</tbody>
</table>

Cycles are not Strictly Sequential

- An evolution cycle includes one execution of all four phases and produces a software generation

- Most software systems require multiple cycles
  - An initial development cycle leading to the initial delivery
  - Subsequent evolution cycles to maintain and enhance the system

- Cycles may be sequential, but more commonly overlap
Who is doing What, When, and How

New or changed requirements → **Software Process** → New or changed system

- Provides guidelines for efficient development
- Reduces risk and increases predictability
- Captures and presents best practices
- Promotes common vision and culture

Engineering and Management Disciplines

*Discipline: a set of activities performed by a role upon artifacts.*

**Engineering**

- **Requirements**
- **Analysis & Design**
- **Implementation**
- **Test**

**Management**

- **Configuration & Change Mgmt**
- **Project Mgmt**

- Use-Case Model, Specifications
- Analysis -Design Model
- Implementation Model, Code
- Test Model, T&I

---

CS/IT 328
Secure Software Construction
45

CS/IT 328
Secure Software Construction
46
Software Process and Software Life Cycle

- Understanding the Development and Evolution of Software
- Defining the Software Life Cycle and the Software Process

Defining Elements of the Software Process
- Roles, Activities, Artifacts
- Disciplines

Integrating the Software Life Cycle and the Software Process

Customizing the Software Process

A software process is a collaboration between abstract active elements called **roles**.

Roles perform operations called **activities**.

Activities are performed on concrete, tangible entities called **artifacts**.

The fundamental conceptual model makes use of the Unified Modeling Language (UML) notation.
- A role denotes one of several roles which may be played by an individual
- In the process roles perform activities; an activity is a piece of work executed by one role
- An artifact is any piece of information or physical entity used or produced by the activities of the SW development process.
Diagrams Represent Process Disciplines

- **Role**: A role that may be played by an individual or a team in the development organization.
- **Activity**: A unit of work an individual may be asked to perform.
- **Artifact**: A piece of information that is produced, modified, or used by a process.

Fundamental Conceptual Model

- **Role** 1
- **IsResponsibleFor** 0..*
- **Artifact**
- **Performs** 1
- **Activity**
- **Consumes** 0..*
- **Produces** 0..*
Dividing the Work by Activities

- Find Actors and Use Cases
- Capture a Common Vocabulary
- Manage Dependencies
- Develop Vision

Guideline: Process Discriminants

Software Process and Software Life Cycle

- Understanding the Development and Evolution of Software
- Defining the Software Life Cycle and the Software Process
- Defining Elements of the Software Process

- Integrating the Software Life cycle and the Software Process
  - Disciplines and Phases
  - Iterations
  - Opportunistic and Systematic Activities

- Customizing the Software Process
2D Model of Process and Life Cycle

Engineering Disciplines
- Requirements
- Analysis & Design
- Implementation
- Test

Supporting Disciplines
- Configuration & Change Mgmt
- Project Management

Weight of Disciplines in Iteration Varies

Engineering Workflows
- Requirements
- Analysis & Design
- Implementation
- Test

Supporting Workflows
- Configuration & Change Mgmt
- Project Management

ITERATION A
ITERATION E
Artifacts Evolve in the Development Cycle

Inception | Elaboration | Construction | Transition
---|---|---|---
M | R | D | I

- **M**: Management artifacts
- **R**: Requirements artifacts
- **D**: Design artifacts
- **I**: Implementation artifacts

An iteration is a distinct sequence of activities with an established plan and evaluation criteria resulting in modified artifacts.

Systematic or Opportunistic Iterations

Possible opportunistic sequences:

Systematic sequences: M R D I
Software Process and Software Life Cycle

- Understanding the Development and Evolution of Software
- Defining the Software Life Cycle and the Software Process
- Defining Elements of the Software Process
- Integrating the Software Life cycle and the Software Process

Customizing the Software Process
- Influential Factors in Customizing Software Process
- Software Process Standards and Models

Software Process Standards

- **IEEE std 1074-1995 processes**
  - Project management
  - Pre-development
  - Development
  - Post-development
  - Integral
Wrap up

- Software process versus Software life-cycle
- The phases of a software life-cycle
- The disciplines of a software process
- The components of a discipline
- Development cycle vs evolution cycle
- Interaction between the disciplines and the phases
- Iterations and software processes
- Various models and standard on software process and software life-cycle