2. Analysis, Design and Implementation

Subject/Topic/Focus:
- Software Production Process

Summary:
- Software Crisis
- Software as a Product: From Programs to Application Systems Products
- Software Development: Goals, Tasks, Actors, Issues
- Software Development Models
- Objetory: The UML Software Development Process

Literature:
- Sommerville
- Brooks
- Fowler

Software Crisis

- declared in the late 60’s at the NATO Conferences “Software Engineering Techniques”, Oct. 1968 and Oct. 1969
- expressed by delays and failures of major software projects that resulted in unmaintable software and unpredictable costs
- was obviously driven by uncoordinated and unstructured programming (“hacking”)
- lead to a new research and engineering discipline: Software Engineering

Hacking

Software Engineering
- Methods
- Models
- Tools and concepts
Your Experiences, Please!

- Which software development **methods** and **models** do you know?

- Which **steps** would you take if a customer, say the boss of 100 (german-wide) warehouses like Karstadt, comes to you, saying that he wants you to develop a business information system that helps him to control his business?

- How would you organize a **team** of 100 developers programming a banking application?

---

From Programs to Application Systems Products

![Diagram](attachment://diagram.png)

- A Program → An Application System: x3
- A Program Product → An Application Systems Product: x3
- Complexity: x9
**A Program**

- developed by a single programmer
- complete in itself
- one customer & one user: the author
- operational only on the author’s system for which it was developed

Example: My address manager written in VisualBasic.

---

**An Application System**

- contains many programs for different tasks
- components coordinated in function
- programs disciplined in format
- well defined interfaces between components
- one customer with many users

Example: Information system for all departments (stock, accounts, management, ...) of firm X
A Program Product

- developed by a developer team
- thoroughly tested and well documented
- many single customers (= users)
- specialized to one task
- available for different environments

Example: Microsoft Word

An Application Systems Product

combines the attributes of
- program
  - complete in itself
- application system
  - programs disciplined in format
  - coordinated components
  - many users
- program product
  - developed by a team
  - thoroughly tested
  - usable on different platforms
  - many customers

Example: SAP
Software Systems Characteristics

- Software is an **immaterial** product.
- Software does **not** underlie an **aging** process.
- For software there are **no spare parts** as there are for machinery etc.
- Software does **not wear off** and therefore it does **not need maintenance** in the common sense.
- Software is **easier changed** than other technical products.
- Software has to be **adapted** to changes of requirements or environments.
- For software there is no production process but **product development**.

Software Product Attributes

Essential attributes of **well-engineered** software:

- **Maintainability**
  - possibility to evolve software to meet the changing needs of a customer
  - well defined interfaces to third party products
- **Dependability**
  - reliable systems which do not cause physical or economical damage in case of system failure
  - security and safety
- **Efficiency**
  - no wasteful use of resources like memory, storage, or processor cycles
- **Usability**
  - appropriate user interface
  - adequate documentation
Goals and Tasks of Software Development

Main Goals

- Product related:
  - Usability
  - Productivity
  - Quality
- Process related:
  - Schedules and costs

Main Tasks

- Analysis
- Design
- Implementation
- Test
- Introduction
- Maintenance

Mission: Delivering a product
- in time
- that is useful and used
- at the predicted costs.

Meeting the Industrial Requirements

- The software product has to meet the specification.
  - Track changed requirements during development process.
  - Prepare for changes of hardware platforms and/or software environments.

- The software product has to be produced in time.
  - Apply project management and project organization.
  - Employ qualified experts.

- The software product should not exceed the estimated costs.
  - Develop in cycles and evaluate early using prototypes.
  - Plan the installation of the software system and the education of users.
**Actors**

- **Customer and Users**
  - Requirements
  - Using
  - Training

- **System engineers**
  - Problem definition
  - Solution analysis
  - Process planning
  - Process control
  - Product evaluation

- **Project managers**
  - Planning
  - Organizing
  - Staffing
  - Directing
  - Controlling

- **Software engineers**
  - Software design
  - Coding
  - Unit testing
  - Subsystem integration

**Communication**

- **capability & competence**

---

**Issues in Software Development**

**Characteristics**
- complex product
- complex development process
- many developers

**Communication**
- users, domain experts & developers
- different developers in different development phases
- developers on different platforms
- development & maintenance

**Problems**
- representation of complex domains
  - graphically & textually
  - prototypes
- dividing large problems into small manageable systems
- accuracy → verification contra flexibility → creativity
- teamwork
  - distribution
  - sharing & coordination
  - communication
Abstraction Levels

Requirements Analysis: Why?
- real-world model
- requirements
- domain knowledge
- goals

Design: What?
- system design
- abstractions
- models
- structures
- architecture

Implementation: How?
- system implementation
- algorithms
- generic services
- platform-specific services

Software Development Models (1)

1. Waterfall model
- requirements definition, analysis
- system and software design
- implementation and unit testing
- integration and system testing
- operation and maintenance

The development phases are performed sequentially.

[Ian Sommerville; Software Engineering, Addison Wesley, 1982]
Software Development Models (2)

1. Waterfall model (modified)

- **Requirements definition, analysis**
- **System and software design**
- **Implementation and unit testing**
- **Integration and system testing**
- **Operation and maintenance**

... but there are **backward loops** in case of changing requirements, error corrections, ...

[Ian Sommerville; Software Engineering, Addison Wesley, 1982]

Software Development Models (3)

2. Evolutionary development

- **Concurrent activities**
  - Specification
  - Development
  - Validation

- **Prototypes**
  - Initial version
  - Intermediate versions
  - Final version

[Ian Sommerville; Software Engineering, Addison Wesley, 1982]
Software Development Models (4)

3. Boehm’s spiral model

[Ian Sommerville; Software Engineering, Addison Wesley, 1982]

Objectory: The UML Software Development Process

- Inception establishes the business rationale for the project and decides on the scope of the project.

- Elaboration is the phase where you collect more detailed requirements, do high-level analysis and design to establish a baseline architecture and create the plan for construction.

- Construction is an iterative and incremental process. Each iteration in this phase builds production-quality software prototypes, tested and integrated as subset of the requirements of the project.

- Transition contains beta testing, performance tuning and user training.
First Step: Inception

- Inception can take many **forms**:
  - For some projects it is a **chat** at the coffee machine.
  - For bigger projects it is a full-fledged feasibility **study** that takes months.
- During the inception phase you work out the **business case** for the project:
  - Derive how much the project will **cost**.
  - Estimate how much **profit** it will bring in.
- Some **initial analysis** is required to get a sense of the project’s **scope** and **size**.
- Inception should be a **few days** of work to consider if it is worth doing a few months of work of deeper investigation during elaboration.
- At the point of inception the project **sponsor** agrees to no more than a serious look at the project:

  **Do we go ahead with the project?**
Second Step: Elaboration

- Starts after you have received the “go-ahead to start the project” agreement.
- At this stage you typically have only a vague idea of the requirements.
  “We are going to build the next generation customer support system for the Watts Galore Utility Company. We intend to use object-oriented technology to build a more flexible system that is more customer oriented - specifically, one that will support consolidated customer bills.”
- Elaboration is the point where you want better understanding of the problem:
  - What is it you are actually going to build?
  - How are you going to build it?
  - What technology are you going to use?
- Elaboration includes to have a careful and thorough look at the possible risks in your project:

Elaboration: Systems Analysis

Rationale: Finding and fixing a fault after software delivery is 100 times more expensive than finding and fixing it during systems analysis or early design phases.

- The goal of analysis is to develop a model of what the system will do.
- The analysis should include information required to understand what is meaningful from a real world system.
- The client of a system should understand the analysis model.
- The analysis phase delivers a base from which further details are derived in the design phase.
- Analysis provides the requirements and the real-world environment in which the software system will exist.

Object-oriented analysis forces a seamless development process with no discontinuities because of continuous refinement and progressing from analysis through design to implementation.
Analysis: Actors, Steps, Deliverables

- Users
- Developers
- Managers
- Users interviews
- Domain knowledge
- Real-world experience

Generate requests → Build models → Problem statement
→ UML Diagrams
- Use Cases
- Classes
- Interactions
- Packages
- States
- Activities
...

Elaboration: Requirements Capture

Identify typical use cases (see chapter 2.1) of the system you are going to build.

- For a person using a database a typical use case would be:
  - “list all customers who have ordered a certain product”
  - “create a list with my top 10 customers”
  - “I want fax-letters to be sent automatically”
- A developer responds with specific cost estimates:
  - “The top 10 customer list can be developed in a week.”
  - “Creating the auto-fax function will take two months.”
- User and developer negotiate about the priorities:
  - Developer: “I could start with the sold - products list.”
  - User: “I definitely need the top 10 customers list first.”
Elaboration: Planning

The users should indicate the level of priority for each use case.

- “I absolutely must have this function for any real system.”
- “I can live without this function for a short period.”
- “It is an important function, but I can survive without it for a while.”

The developers should consider the architectural risk.

- Do not omit use cases which later cause you to do a lot of rework to fit them in.
- Concentrate to the use cases which are technologically most challenging.

The developers should be aware of the schedule risks.

- “I’m pretty sure I know how long it will take.”
- “I can estimate the time only to the nearest man-month.”
- “I have no idea.”

Schedule use cases to specific iterations and dates of delivery.

Planning: Estimate

Once the use cases are assigned, the length of each iteration should be estimated to the nearest person-week.

In performing this estimate assume you need to do:
- analyzing
- designing
- coding
- unit testing
- integration
- documentation

The estimates should be done by the developers, not by the managers.
Third Step: Construction

Construction builds the system in a series of iterations. Each iteration is a project in itself.

During each iteration you go through a cycle of analyzing, designing, coding, debugging, integration and demonstration of the implemented use case by a prototype.

Construction: Iterations

- Finish the iteration with a demo to the user and perform system tests to confirm that the use cases have been built correctly.

- Iterations within a construction are both, incremental and iterative.
  - Iterations are incremental in function. Each iteration builds on the use cases implemented in the previous iteration.
  - They are iterative in terms of the code base which will be rewritten to make it more flexible.

- Do not underestimate the testing phase.
  - Write test code.
  - Separate the test into unit and test code.
  - Unit tests should be written by the developers.
  - Apply all tests after each iteration.