Software Engineering Lecture Notes WS 24/25

Igor Dimitrov

2024-10-14

Table of contents

Preface

Lecture notes for the course “Software Engineering” at Heidelberg University WS24/25.

1 Introduction

1.1 Chapters

1. Introduction
2. Communication in a Project link
3. Requirements Engineering (Communication with the Users) link
4. Design (Communication with Developers) link
5. Quality Management link
6. Evolution link
7. SWE-Process (Summary and Project Management) link

2 Communication in a Project

Following topics relate to and determine communication within a project:

1. Number of participants and their roles in the project
2. Type of the contractual relationship
3. Team Organization: The way developers communicate within the project
4. Collaborative Coding

2.0.1 Projects and Participants

terms relating to project and process:

- process:
- project
- process model:

characteristics of a project:

- limited time
- creator
- purpose
- client
- results
- means and tools
- organization and planning

participants:

- client
- user
- manufacturer

2.0.2 Contractual Relationship

project types:

- EP (Entwicklungsprojekt) -> development project
- AP (Auftragsprojekt) -> Commissioned Project
- EDP (EDV-Projekt, EDV = Elektronische Datenverarbeitung) -> IT Project
- SP (Systemprojekt) -> System Project

2.0.3 Team Organization

types of team organization:

- single person
- 2-person team
- anarchic team
- democratic team
- hierarchical team
- chief-programmer team
- agile team

types of secondary organization:

- functional
- project-based
- matrix

2.0.4 Collaborative Coding

- Pair programming
- Distributed development

3 Requirements Engineering

requirements engineering can be understood as corresponding to the communication with the users / clients. Deals with the following topics

1. Introduction to communication with users/clients.
    1. Clients and Requirements
    2. Description and specification of requirements
    3. Defining Requirements Engineering
    4. Outcome of Requirements Engineering
    5. Benefit of specification
    6. Complexities of RE
2. Usage modelling / description
    1. Introductory Example
    2. Introduction
    3. Tasks, Roles, Persona
    4. Domain Data
    5. Functions, UI-Structure
    6. GUI
3. Documentation Quality
    1. Introduction and Templates
    2. Characteristics and style guide
4. Usability
5. Quality assurance with the client
    1. acceptance test
    2. usability test
6. Quality requirements
    1. Motivation
    2. Quality attributes
    3. QR-description
    4. QR-test
7. Use-cases (not relevant to the exam)

1. Description of Uses Cases
2. Use for system testing

8. RE procedure

    1. Introduction
    2. Gathering requirements
    3. Specificying requirements

3.0.1 Communication with Users / Clients

- requirements engineering:
  - collection of the requirements from the client
  - specification / formalization of the requirements
  - testing / examination of the requirements
  - management of the requirements
- requirements engineering result:
  - document:
    - description using system functions
  - prototype
- Advantages and Uses of Specification:
- Disadvantages of a missing specification
- Difficulties related to RE:

3.0.2 Usage Modelling

- Task-oriented Requirements Engineering:
  - Task level: tasks, roles, persona
  - Domain level: subtasks (as-is & to-be), domain data
  - interaction level: system functions, ui-structure
  - system level: gui, screen-structure (virtual window)

Roles, Persona, Tasks

3.0.2.1 Roles, Persona

UDC (User-centered design)

Roles and Persona

- User role:
- User profile:

How are personae described:

- name
- biographic facts: age, gender, etc
- knowledge and attitude with respect to the tasks and technology:
- needs: main use-cases in which the user wants to apply the software -> Tasks
- frustrations:
- ideal features:

3.0.2.2 Tasks

How tasks are described:

- Goals
- Decisions
- Causes
- Priority
- Execution profile (frequency, continuity, complexity)
- Precondition
- Info-in (input)
- Info-out (output)
- resources (means, participating roles)

sub-tasks:

Persona-task correspondence:

- needs <=> sub-tasks => combination of system-functions
- frustrations <=> problems in sub-tasks
- ideal features <=> ideas, system functions

3.0.2.3 Domain Data

Domain data explain the terms used in the task descriptions.

Goal: describe/model the entities of real world, including their relationships / associations to each other, in order to understand the tasks.

- domain data describe **entities** that are relevant within the context of the sofwtare. They correspond to the terms used in the task description => independent from the software.
- described with simple **class diagrams** => **domain data diagram** (no operations, no aggregation, no inheritence, only associations)
- sometimes **glossary** is sufficient.

sometimes not sufficient, because there additional data necessary on the UI level => **interaction data**. (not relevant in our MMAP case)

3.0.2.4 Functions & UI-Structure (Interaction Level)

Goal: implementation of the User/Machine boundary with respect to the task descriptions.

consists of 2 parts:

1. System functions: which functions are provided by the system?
2. UI-Structure:
    - in which context can the user call which functions,
    - which data is available/visible in those contexts?
    - how are functionalities divided among the sub-parts?

> ⚠ Warning
>
> UI-Structure is **not** GUI-structure, i.e. the concrete layout is not yet determined.

3.0.2.4.1 System Functions

how is a system function described:

- name: nomenclature verb-object, describe what will be achieved on the user data (e.g. unlinkMovie => movie will be unlinked)
- input:
    - context (i.e. workspace) in which the SF accessible

- concrete input: data that is gathered during the interaction with the user. Such data is not yet determined when the SF is first called on the GUI, but first provided by the user during the interaction.
- output: changes of the UI
- description:
- exceptions: cancel/discard by the user
- rules:
- quality requirements:
- precondition:
- postcondition:

3.0.2.4.2 UI-Structure

Consists of

- workspaces:
  - bundle related system functions and data similar to a class (but only from an abstract user point of view. Actual structure in code can be completely different)
  - only system functions that can be triggered by the user are listed
- navigation links between workspaces

UI-structure abstracts from a concrete screen-layout. Logical represents a logical view of the interaction structure.

> **!** Important
>
> UI-structure is created concurrently with the System functions, because their close interrelation. (Workspaces contain System functions and data)

3.0.2.5 Design of System Functions and UI-Structure

- how is the system function specification template filled in?
- how are ui-structure decision made concurrently to SF specifications?
- initial test considerations?

There's still lots of wiggle room for specific design decisions.

4 Design

Desing can be understood as communication with and wthin the developers. Deals with the following topics:

1. Introduction to Modelling
2. Class diagrams
3. Interaction diagrams (sequence diagrams)
4. State Diagrams
    1. UML State diagrams
    2. Dialog models
5. Class design with OOAD
    1. OOAD introduction
    2. OOAD: Analysis Class diagram
    3. OOAD: Design Class Diagram
6. Design Patterns
    1. Introduction
    2. Creational patterns
    3. Structural patterns
    4. Behavioral patterns
7. Rationales (Communication of decisions)
8. Summary of modelling techniques

5 Quality Assurance

Quality: Software satisfies the requirements

topics:

1. Introduction
2. Organizational quality assurance
3. Testing:
    1. Intro
    2. Test-case specification
    3. Black-box component testing
    4. White-box component testing
    5. System testing
    6. Integration testing / overall-component testing
4. Static testing
    1. Static Analysis
    2. Metrics
    3. Inspection
5. Analytical Quality assurance at large

6 Evolution

All activities that facilitate re-use and further development. (All activities that take place after the initial development phase)

topics:

1. Intro
2. Architecture
3. Re-use
4. Further development and change management
5. DevOps & IT-Governence
6. Re-engineering

7 SWE Process & Project Management

Making sure that the Software system is developed withing the time money constraints.

topics:

1. Project management
2. SWE-process models & methods