Technische Universität München
Fakultät für Informatik

Bachelorarbeit/ Masterarbeit in Informatik

Titel der Bachelor/Masterarbeit

Vorname Nachname

Technische Universität München
Fakultät für Informatik

Diplomarbeit in Informatik

Titel der Bachelor/Masterarbeit

Deutscher Titel

Vorname Nachname

Abgabedatum:
15.MM.2010

Aufgabensteller:
Prof. Bernd Brügge, Ph.D

Betreuer:
Dipl.-Inf. Univ. Damir Ismailović

Ich versichere, dass ich diese Diplomarbeit selbständig verfasst habe und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 15.MM.2010, ...
(Vorname Nachname)

Acknowledgments

Eigener Text ...

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The aim of this thesis is ...

1.1 Problem Statement

In a software project a developer is usually working on several tasks simultaneously, using various tools appropriate for different phases of development. In the context of software development a task can be defined as *an atomic well-defined work assignment for a project participant or a team* [1] or as *a usually assigned piece of work often to be finished within a certain time* [4].

1.2 Outline

- **Chapter 2** provides a brief overview of ..... We document the requirements elicitation and the analysis model of the framework by describing the high-level functionality (functional requirements) and requirements that are not directly related to functionality (nonfunctional requirements). The chapter also describes the analysis model in form of use cases, object models, and dynamic models for the system. It contains the complete functional specification and therefore it can be seen as the **Requirement Analysis Document** (RAD).

- **Chapter 3** provides an overview of the software architecture and the design goals and presents a survey of current architectures for similar systems. The purpose of this chapter is to make explicit the background information of the system architecture, their assumptions, and common issues the new system will address. The main part of this chapter documents the system design model with subsystem decomposition, hardware/software mapping, persistent data management, access control and security and global software control. This chapter represents the **System Design Document** (SDD).

- **Chapter 4** describes the detailed decomposition of subsystems into packages and classes. Class interfaces will be introduced and additional classes and interfaces which elaborate the model will be presented. This chapter is organized by packages and is considered as the **Object Design Document** (ODD).

- In **Chapter 5** we will present the prototype implemented during the thesis. After presenting the prototype we will review the requirements and evaluate the implementation. Furthermore we will present informal discussions with developers and their feedback.
- **Chapter 6** will show the conclusion and the preview to future work. We will summarize in this chapter some interesting issues which we encountered during the development process for the future development of the present approach.

1.3 Document Conventions

We use the following conventions throughout this document:

- Typographical Conventions
  - A new term is written in **bold** when it appears for the first time. These terms will be explained in the glossary in Appendix C.
  - The names of system and of modelling elements such as packages, classes, attributes and methods are written in `Courier font`.
  - Source code appears in `Courier font` too.
  - Citations will be shown in *italic*.
- Notation and Diagrams
  - We will make use of the **Unified Modeling Language** (UML) version 2.1.2 for all diagrams in the thesis as defined by [5].
  - We will use the following notation for sequences of events presented in UML sequence diagrams which are referenced by other sequence diagrams.

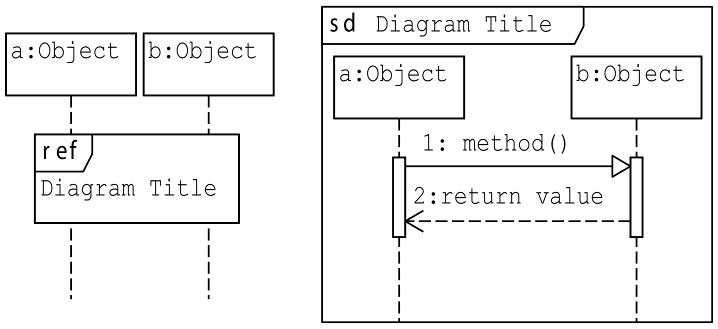

Figure 1.1: Referencing sequences of events in UML sequence diagrams.

Diagrams which are referenced will be framed and described with a name in a compartment in the upper left corner.

Chapter 2

Requirement Specification

In the first chapter we have initially described the focus of this thesis. This chapter provides a brief overview of the purpose and the scope of this framework we have developed. In the objective section the main goals of the framework are described. After discussing the current state of the art, a few scenarios which describe the problem situation are shown. This discussion results in the functional and nonfunctional requirements the framework has to meet, which is presented in the following sections. On the basis of those results in the section regarding the system model the use cases, the dynamic model and the object model will be specified.

2.1 Objectives

As already mentioned the goal of this thesis is ....

2.2 Related Work

This chapter describes the current state of affairs. If the new system will replace an existing system, this section describes the functionality and the problems of the current system. Otherwise, this section describes how the task supported by the new system are accomplished now.

2.3 Scenarios

We will first present some problem scenarios, which will reinforce problems described in the problem statement in Chapter 1. Afterwards we introduce the visionary scenarios. We will focus here on a single developer.

2.3.1 Problem Scenarios

Scenario A

Developer XXX is working on the ABC for the project DEF to include the functionality ....

Problem: A shortly described problem in the Scenario A

Scenario B

The company GAH tries to IJK ....

Problem: A shortly described problem in the Scenario B

2.3.2 Visionary Scenario

Scenario C

This scenario shows the vision of this thesis ....

2.4 Functional Requirements

This section describes the functionality of the system. ....

2.4.1 Functional Requirement 1

Text ...

2.4.2 Functional Requirement 2

In every activity a developer is doing, he is intending to accomplish some goals. This framework should ....

2.5 Nonfunctional Requirements

This section describes user requirements that are not directly related to functionality. These criteria that can be used to judge the operation of a system, rather than specific behaviors, are usability, reliability, security, privacy, performance, supportability, implementation, interface and packaging.

2.5.1 Usability

The common understanding of the term *usability* is the ease with which a user can interpret and respond to information, thus the ease of the interaction and operability with the system. Usability can be measured via performance errors and productivity, and also via user preferences and interface characteristics. Providing an user interface (UI) that is easy to understand is an essential criterion in this context. This includes an adequate representation of information even within complex systems.

2.5.2 Reliability

Reliability is an important facet of software quality. The high complexity of software is the major contributing factor of software reliability problems.

2.5.3 Security

In the area of software engineering security can be defined as a process of system screening that denies access to unauthorized users and protects data from unauthorized uses. The major security aspect concerning our target framework is the system's stability when performing critical operations with the user's permission.

2.5.4 Privacy

Privacy is the ability of an individual or group to seclude themselves or information about themselves and thereby reveal themselves selectively. Considering this issue we focus on the following aspects of the user's privacy: A, B and C.

2.5.5 Performance

A major factor in determining the overall productivity of a system, performance is primarily tied to availability, throughput and response time. Our framework should ....

2.5.6 Supportability

We define supportability as all the actions related to the inherent quality of the system or rather framework, including all maintenance procedures required to facilitate detection, isolation and repair of system anomalies. Examples of features that facilitate supportability include, e.g. , system documentation or software update.

To deal with fast changing development requirements, the framework has to be extensible. This extensibility could be considered as a part of supportability, thus successful dealing with risks and changes in technical surroundings is a major part of the system quality. Therefore, the framework should be able to easily integrate current and future technologies without major modifications of the existing system.

2.5.7 Implementation Requirements

Implementation requirements [3] are constraints on the implementation of our target framework which specify tools and hardware platforms to be used. ....

2.6 System Models

In this section we describe the analysis model of the framework. We begin with presenting use the case model inferred from the functional requirements. In what follows we introduce the conceptual monitoring model, which constitutes the result of the preliminary analysis of the elicited requirements.

2.6.1 Use Case Model

Working through the requirements described in Sections 2.4 and 2.5, we identified the use cases, which specify how the actors are interacting with the system. In this section we will show the use case model, and describe the specific use cases. A complete flow of events for each use case can be found in Appendix A. Examining the objectives and functional requirements, we identified the actors `A` and B. Figure 2.1 shows the main use cases from the A's and B's perspective.

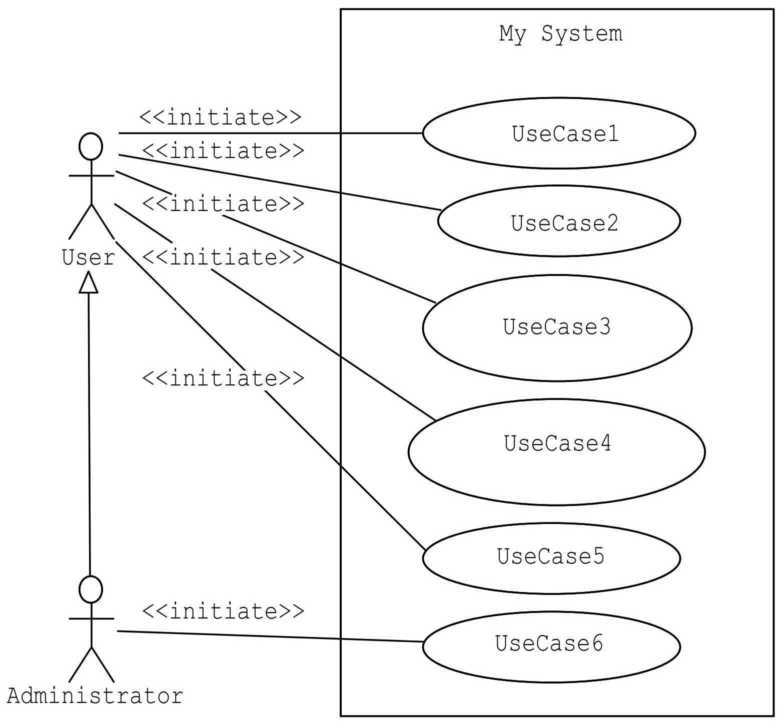

Figure 2.1: Main use cases identified for the actors `A` and `B` (UML use case diagram).

2.6.2 Object Model

In this section we describe the conceptual model of the framework. We will describe in detail all the objects we identified during the requirements elicitation. Each object will be described with textual definitions and relationships among objects will be illustrated with class diagrams. Because we identified many objects, we will describe the models based on the following structure....

2.6.2.1 Some Model

xxxxxx

2.6.2.2 More Models

xxxxxxx

2.6.3 Dynamic Model

This section documents the behaviour of the object model in terms of statechart diagrams and sequence diagrams. Although we here describe some use case information redundantly, dynamic models enable us to specify the behaviour more precisely.

2.7 User Interface

According to the use case model described in Subsection 2.6.1 this Section explains the user interfaces.

Chapter 3

System Design

This chapter will show the transformation of the analysis model described in 2.6 into a system design model. It presents the preliminary architecture along with the design goals that influenced it.

First, in Section 3.1 we discuss the qualities of the system that should be optimized in the development. Section 3.2 will show a survey of the enabling technology for the solution. Then, we describe in sections 3.3 and 3.4 the subsystem decomposition in terms of subsystem responsibilities, dependencies among subsystems and subsystem mapping to hardware. The following next 3 sections 3.6, 3.5 and 3.7 give an outline of the major policy decisions such as persistent data management, access control, security and global software control. Finally, we discuss the boundary conditions of the developed system.

- Design Goals (Section 3.1)
- Enabling Technology (Section 3.2 on the next page)
- Subsystem Decomposition (Section 3.3 on the following page)
- Hardware/Software Mapping (Section 3.4 on page 11)
- Persistent Data Management (Section 3.6 on page 11)
- Access Control and Security (Section 3.5 on page 11)
- Global Software Control (Section 3.1 on page 12)

3.1 Design Goals

Obviously, there is a potential conflict between some of the defined requirements which we should consider in the system design. For example, using large-grain components improves performance, and using fine-grain components improves maintainability. To make decisions in the system design easier we analysed the requirements prior to developing this framework. From this analysis we formulated a number of design goals, which represent qualities that enable us to prioritize the development of the system. Thus, we have settled those goals as three major design goals which are:

- A
- B
- C

A

The most important ....

B

We are trying to ...

C

Text...

3.2 Enabling Technology

This section presents a survey about the enabling technology. We will focus on the most important technologies which allow us to provide the functionality mentioned in the requirements. One of those technologies is XXX.....

3.3 Subsystem Decomposition

In this section we introduce the initial decomposition of the system and describe the responsibilities and boundaries of each subsystem. First, based on the functional requirements and the models shown in the previous chapter, we identified N main subsystems: A, B, C and D subsystem. The subsystems are presented in detail in the following sections.

3.3.1 Proposed Software Architecture

In the analysis model described in Section 2.6 we introduced N main concepts for our target framework. First of all there is .....

3.3.2 A Subsystem

The `A` subsystem consists of ....

Services:

- Service A1
- Service A2

3.3.3 B Subsystem

The `B` subsystem is ...

Services:

- Service B1
- Service B2

3.3.4 C Subsystem

The `C` subsystem provides the

Services:

- Service C1
- Service C2

3.4 Hardware/Software Mapping

describes how subsystems are assigned to hardware and off-the-shelf components. It also lists the issues introduced by multiple nodes and software reuse.

3.5 Access Control and Security

This section describes access control and security issues, such as ....

3.6 Persistent Data Management

This section describes the persistent data stored by the framework and the data management infrastructure required for it.

3.7 Global Software Control

This section describes how the global software control is designed. Figure 3.1 illustrates the main control flow involving the subsystems `xxx, xxx, xxx, xxx`.

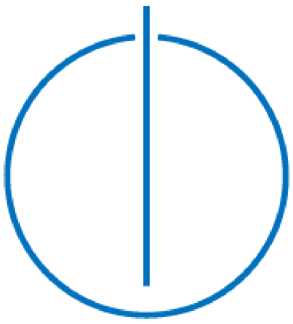

Figure 3.1: Main control flow involving the subsystems xxx, xxx, xxx, xxx (UML sequence diagram).

Chapter 4

Object Design

In Section 3.3 we divided the classes presented in the analysis model into different subsystems. In this section we will focus on packages and present additional classes to refine the model when necessary. Furthermore, attributes and methods are added to the existing classes. The object design is structured according to the subsystem decomposition presented in the system design in Section 3.3. We will show the packages in this chapter organized by subsystems.

- Package A (Section 4.1)
- Package B (Section 4.2)
- Package C (Section 4.3)
- Package D (Section 4.4 on the following page)
- Package E (Section 4.5 on the next page)

The diagrams for each package also show classes related to the package that are not part of the package for readability reasons.

4.1 Package A

Initial starting point for the design of the A package is the ...

4.2 Package B

As already described the aim of this package is ....

4.3 Package C

The aim of this package is ....

4.4 Package D

The aim of this package is ....

4.5 Package E

The aim of this package is ....

Chapter 5

Evaluation

In this chapter we will present the prototype. We will show how this prototype works and we will also present the graphical user interface. Furthermore we will evaluate the implementation by reviewing the requirements. In the next step we will present evaluation goals and methods......

5.1 Prototypical Implementation

This section presents the implemented prototype with its graphical representation. In the actual version XXX we introduce a XXX application. It is capable of .....

Table 5.1 shows the complexity of the implementation by using a software metric **Source lines of code** (SLOC) which is typically used to estimate programming productivity or effort once the software is produced.

Language	files	blank	comment	code	scale	3rd gen. equiv
Objective C	65	1510	1371	5024	x 2.96	14871.04
C	5	563	782	1965	x 0.77	1513.05
C/C++ Header	71	729	1158	1198	x 1.00	1198.00
SUM:	141	2802	3311	8187	x 2.15	17582.09

Table 5.1: Count of source lines of code (SLOC) for the prototypical implementation.

5.1.1 Some Functionality

The main functionality of the presented framework is ....

5.1.2 More Functionality

Text...

5.1.3 Achievement of Objectives

In this section we will verify the implemented prototype against the requirement specification presented in Chapter 2.

5.1.3.1 Realized Functionality

Here, we will review the functional requirements described in Section 2.4 and evaluate how they are realized.

Functionality A The main functional implementation of the ....

Functionality B Text. ....

Functionality C Text. ....

5.1.3.2 Nonfunctional Requirements Accomplishment

In this subsection we will evaluate the nonfunctional requirements.

Usability One of our goals in system design was to design usable interfaces. ...

Reliability We tried to make the framework extensible, but without losing the focus on reliability. In the prototypical implementation we „„

Security Text ....

Privacy Privacy Text....

Performance Performance Text...

Supportability In the requirement specification we have defined supportability as all the actions related to the inherent quality of the target framework required to facilitate detection, isolation and repair of system anomalies. In system and object design we have demonstrated how our target framework can be extended. Furthermore, we have designed the system to support future technologies without major modifications by decomposing the system and using design patterns. Beyond the extensibility to other technologies, we also considered the portability of the system to other platforms......

5.2 Evaluation Goals

5.3 Evaluation Methods

5.3.1 Case Study

Case Study Description

5.3.2 Qualitative Research

- Questionnaire
- Interviews
- etc.

5.3.3 Experimental Study

Experiments Description

5.3.4 Evaluation Tools

5.3.5 Data Analysis

5.4 Results

5.5 Validity

Chapter 6

Conclusion and Future Work

- Reference: [1]

Conclusions are often the most difficult part of the thesis to write, and many writers feel that they have nothing left to say after having written the thesis. A writer needs to keep in mind that the conclusion is often what a reader remembers best. Your conclusion should be the best part of your thesis.

A conclusion should:

- stress the importance of the thesis statement,
- give the essay a sense of completeness, and
- leave a final impression on the reader.

Suggestions

- **Answer the question "So What?"**
  Show your readers why this paper was important. Show them that your thesis was meaningful and useful.
- **Synthesize, don't summarize**
  Don't simply repeat things that were in your thesis. They have read it. Show them how the points you made and the support and examples you used were not random, but fit together.
- **Redirect your readers**
  Give your reader something to think about, perhaps a way to use your paper in the "real" world. If your introduction went from general to specific, make your conclusion go from specific to general. Think globally.
- **Create a new meaning**
  You don't have to give new information to create a new meaning. By demonstrating how your ideas work together, you can create a new picture. Often the sum of the thesis is worth more than its parts.

[1]http://leo.stcloudstate.edu/acadwrite/conclude.html

Conclusion Strategies

- **Echoing the introduction**: Echoing your introduction can be a good strategy if it is meant to bring the reader full-circle. If you begin by describing a scenario, you can end with the same scenario as proof that your essay was helpful in creating a new understanding.

  – Example

    ∗ *Introduction*
    From the parking lot, I could see the towers of the castle of the Magic Kingdom standing stately against the blue sky. To the right, the tall peak of The Matterhorn rose even higher. From the left, I could hear the jungle sounds of Adventureland. As I entered the gate, Main Street stretched before me with its quaint shops evoking an old-fashioned small town so charming it could never have existed. I was entranced. Disneyland may have been built for children, but it brings out the child in adults.

    ∗ *Conclusion*
    I thought I would spend a few hours at Disneyland, but here I was at 1:00 A.M., closing time, leaving the front gates with the now dark towers of the Magic Kingdom behind me. I could see tired children, toddling along and struggling to keep their eyes open as best they could. Others slept in their parents' arms as we waited for the parking lot tram that would take us to our cars. My forty-year-old feet ached, and I felt a bit sad to think that in a couple of days I would be leaving California, my vacation over, to go back to my desk. But then I smiled to think that for at least a day I felt ten years old again.

- **Challenging the reader**: By issuing a challenge to your readers, you are helping them to redirect the information in the paper, and they may apply it to their own lives.

  – Example
  Though serving on a jury is not only a civic responsibility but also an interesting experience, many people still view jury duty as a chore that interrupts their jobs and the routine of their daily lives. However, juries are part of America's attempt to be a free and just society. Thus, jury duty challenges us to be interested and responsible citizens.

- **Looking to the future**: Looking to the future can emphasize the importance of your paper or redirect the readers' thought process. It may help them apply the new information to their lives or see things more globally.

  – Example
  Without well-qualified teachers, schools are little more than buildings and equipment. If higher-paying careers continue to attract the best and the

brightest students, there will not only be a shortage of teachers, but the teachers available may not have the best qualifications. Our youth will suffer. And when youth suffers, the future suffers.

- **Posing questions**: Posing questions, either to your readers or in general, may help your readers gain a new perspective on the topic, which they may not have held before reading your conclusion. It may also bring your main ideas together to create a new meaning.
  - Example
    Campaign advertisements should help us understand the candidate's qualifications and positions on the issues. Instead, most tell us what a boob or knave the opposing candidate is, or they present general images of the candidate as a family person or God-fearing American. Do such advertisements contribute to creating an informed electorate or a people who choose political leaders the same way they choose soft drinks and soap?

Appendix A

Use Case Descriptions

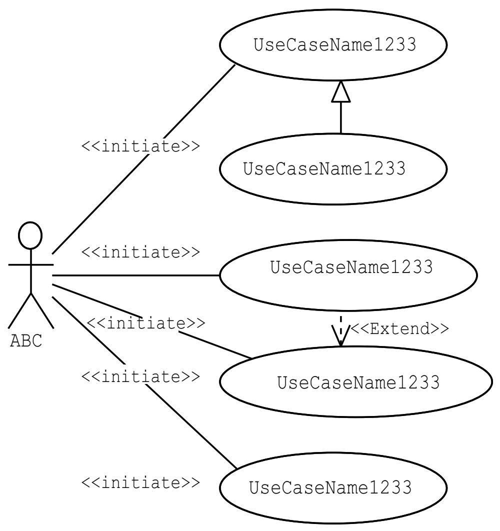

Figure A.1: All use-cases identified from the `User`'s and `Administrator`'s perspective (UML use-case diagram).

Table A.1: Use cases identified from the A's perspective.

Use Case Name	`UseCaseName`
Participating Actors	Initialized by `A`
Flow of Events	1. The `A` decides to xxx; 2. He opens yyy 3. He selects the "ZZZ" tab. 4. The System presents a list of ... 5. After finishing the User closes xxx

Use Case Name	`UseCaseName`
Participating Actors	Initialized by `A`
Flow of Events	1. The `A` decides to xxx; 2. He opens yyy 3. He selects the "ZZZ" tab. 4. The System presents a list of ... 5. After finishing the User closes xxx

Table A.4: Use cases identified from the B's perspective.

Use Case Name	`UseCaseName`
Participating Actors	Initialized by `A`
Flow of Events	1. The `A` decides to xxx; 2. He opens yyy 3. He selects the "ZZZ" tab. 4. The System presents a list of ... 5. After finishing the User closes xxx

Use Case Name	`UseCaseName`
Participating Actors	Initialized by `A`
Flow of Events	1. The `A` decides to xxx; 2. He opens yyy 3. He selects the "ZZZ" tab. 4. The System presents a list of ... 5. After finishing the User closes xxx

Use Case Name	`UseCaseName`
Participating Actors	Initialized by `A`
Flow of Events	1. The `A` decides to xxx; 2. He opens yyy 3. He selects the “ZZZ” tab. 4. The System presents a list of ... 5. After finishing the User closes xxx

Appendix B

Used Software

B.1 Software Development

Eclipse http://www.eclipse.com/
Description

XXXXX http://www.xxxx.com/
Desription

Smultron http://smultron.sourceforge.net/
Smultron is an open source text editor for Mac OS X. This tool supports syntax colouring for different languages. We used this tool to develop the D scripts.

JavaFX-SDK http://argouml.tigris.org/
ArgoUML is an open source UML modeling tool. It supports all UML 1.4 notations. We used ArgoUML for designing the system during the analysis and system design.

B.2 Authoring this Document

MacTEX http://www.tug.org/
MacTEX is a complete TEX system for Mac OS X, supporting TEX, LATEX, AMSTEX, ConTEXt, XeTEX and many other packages.

LYX 1.6 http://www.lyx.org/
LYX is an open source document processor. LYX is a front-end to the LATEX typesetting system. It allows writing a document based on the structure of the document, not the appearance. This document is written entirely with LYX.

Visual Paradigm for UML http://www.visual-paradigm.com/
Visual Paradigm for UML CASE Tool supporting UML modeling with UML 2.1. All diagrams in this thesis are designed with this tool.

Appendix C

Glossary

Adapter Pattern A design pattern that let objects from unrelated packages collaborate by adapting one interface to another.

ANSI C The standard for the C programming language published by the American National Standards Institute (ANSI).

Artifact One of many kinds of tangible byproducts produced during the development of software, such as functional and non-functional requirements, use cases, user interface designs, object oriented analysis etc.

Authorization The act of granting a right. For example, a user asks for the right to perform an operation. The Security Server grants authorization after the user fulfills the rules specified in the policy database—such as providing a credential or authenticating.

Boundary Objects Objects that represent the interface between the system and the actors.

Bridge Pattern A design pattern which decouples an interface from the implementations so that implementations can be substituted.

C A programming language described by the ANSI C standard.

C++ Object oriented programming language.

Causality Bidirectional relationship between a cause and the effect, as the effect is the result of the cause.

Composite Pattern A design pattern which represents a hierarchy of different objects.

Design Pattern Defined and repeatable design solution to a common problem. Design patterns are proven concepts for creating the code to solve a given problem. A common description of several design patterns can be found in [2].

Developer's Action Behaviour caused by the developers in particular situations. Executing an action results to an interaction between the developer and a working tool.

Entity Objects Objects used to describe analysis object model that represent the persistent information of the system.

Facade Pattern A design pattern for providing an interface to a subsystem.

Framework An extensible collection of classes providing a standardized interface to a particular service or application which can be then instantiated to build a concrete application.

FURPS A mnemonic for several kinds of software requirements: Functionality, Usability, Reliability, Performance and Supportability .

Graphical User Interface (GUI) Visual interface of any application or tool. It serves as a bridge between the user and an application/tool and allows the user to input information into the application/tool in a graphic method.

Integrated Development Environment (IDE) Computer software that assists developers in developing software.

Java An object-oriented programming language with portable binaries. It can be used to program applications and applets and is portable to a large number of operating systems.

Model / View / Controller (MVC) A software system architectural style which separates the entity objects and controllers from boundary objects.

Object Design Document (ODD) Documents the object design by describing the decomposition of subsystems into packages, classes and class interfaces.

Observer Pattern A design pattern which maintains the consistency across the states of one publisher and many subscribers.

Requirements Analysis Document (RAD) Documents the requirements elicitation and analysis by describing the requirements of the proposed system.

Ruby A reflective, dynamic, object-oriented programming language.

Shortcut A key press combination that sends a command.

Strategy Pattern A design pattern allowing objects to use different algorithms and change them at runtime.

System Design Document (SDD) Documents the system design by describing the design goals, subsystems, hardware/software mapping, persistent data management, access control, control flow and boundary conditions.

Task An atomic well-defined work assignment for a project participant or a team and usually assigned piece of work often to be finished within a certain time.

Unified Modeling Language (UML) An Object Management Group (OMG) standard for modelling software artifacts. It is widely-used industry-standard language for the specification, visualization, construction, and documentation of the software system components or software systems. It contains a set of symbols for creating diagrams to model software systems.

Bibliography

[1] Bernd Bruegge and Allen H. Dutoit. *Object-Oriented Software Engineering Using UML, Patterns and Java.* Prentice Hall, 2nd edition edition, 2004.

[2] Erich Gamma, Richard Helm, John Vlissides, and Ralph E. Johnson. *Design Patterns Elements of Reusable Object-Oriented Software.* Addison-Wesley, Massachusetts, 2000.

[3] Robert B. Grady. *Practical software metrics for project management and process improvement.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

[4] MA: Merriam-Webster. *Merriam-Webster's collegiate dictionary.* Springfield, 11th edition, 2003.

[5] Object Management Group, Framingham, Massachusetts. *UML Superstructure Specification, v2.1.2*, November 2007.