Farrukh Kholmukhamedov
Rustam Tukhvatov

Innopolis University

myCrashVarian.G01: iCrash system improvements Design Document - v 0.1 -

Friday 14th April, 2017 - 23:22

Springer

Contents

List of Figures

Listings

Chapter 1
Introduction

1.1 Overview

This software system is aimed for car crash crisis management.

Excerption from the analysis document: the *iCrash* is a simple system dedicated to any person who wants to inform of a car crash crisis situation in order to allow for crisis handling. At anytime and anywhere, anyone can be the witness or victim of a car crash and might be in a situation allowing for alerting this crisis. The *iCrash* system has for objectives to support crisis declaration and secure administration and crisis handling by the *iCrash* professional users.

As any software system, *iCrash* requires some improvements. This document covers two new features: biometric authentication and information diffusion.

1.2 Purpose and recipients of the document

This document is a design document. The aim of this document is to provide an example of how the design of a particular software system should be documented.

The recipient of this document is the development company (ADC) in charge of delivering the software system. The company's developers are expected to use this document as the basis for carrying out the actual development and deployment of the product (i.e. implementation, testing and maintenance).

1.3 Definitions, acronyms and abbreviations

Biometrics or **biometry** — the measurement and analysis of unique physical or behavioral characteristics (as fingerprint or voice patterns) especially as a means of verifying personal identity

1.4 Document structure

This document is organised as follows: Section 2 provides a general overview of the main concepts gathered during the analysis phase, in particular those concerning the software system abstract types, as well as the actors that interact with the software system through their interfaces.

The technologies used not only during the design and development phase, but also those required to make the software system runnable are presented in Section 3.

The architecture of the software system to be implemented and deployed is described in Section 4. This section presents the components of the software system architecture along with their interactions, both from the static and dynamic viewpoints.

The detailed design of each System Operation is given in Section 5, whereas Section 6 enumerates the current limitations of the software system at the writing time of this document.

Next, Section **??** presents the different test cases used to verify the correct behaviour of the software system's functional and not functional requirements.

Finally, Section 7 draws the conclusion achieved during the design and implementation of the software system.

Chapter 2
Analysis Models

This chapter provides a general overview of the main concepts gathered during the analysis phase, in particular those concerning the software system types (i.e. classes, datatypes, and enumerations), as well as the actors that interact with the software system through their interfaces. Figures included in the Messir Requirement Document that correspond to the the Concept Models and the Environment Models could be also included in this chapter, as a means of synthesizing what are the requirements to which the design is supposed to sketch a solution.

2.1 Concept Model

2.2 Environment Model

Chapter 3
Technological frameworks

This chapter provides a description of the technological frameworks used during the development phase of **iCrash** system.

3.1 Java Programming Language

Java is a programming language. Java is general-purpose multi-paradigm, class-based language with high portability of the software. Java is a high-level programming language originally developed by Sun Microsystems and released in 1995. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

3.2 MySQL

MySQ is an open-source relational database management system (RDBMS). MySQL is the world's most popular open source database. With its proven performance, reliability, and ease-of-use, MySQL has become the leading database choice for web-based applications, used by high profile web properties including Facebook, Twitter, YouTube, and all five of the top five websites*. Additionally, it is an extremely popular choice as embedded database, distributed by thousands of ISVs and OEMs.

3.3 Eclipse

Eclipse is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages.

3.4 Git

labelsec:Git Git is a version control system (VCS) for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for software development,but it can be used to keep track of changes in any files. As a distributed revision control system it is aimed at speed, data integrity, and support for distributed, non-linear workflows.

3.5 LaTeX

LaTeX is a document preparation system for high-quality typesetting. It is most often used for medium-to-large technical or scientific documents but it can be used for almost any form of publishing.

3.6 Excalibur

Excalibur is a tool supporting the Messir methodology, a Scientific Method for the Software Engineering Master, used in Software Engineering Lectures at bachelor and master levels.

Excalibur tool covers the phase of Requirements Analysis and its main features are requirements analysis specification (its own DSL), requirements report generation (latex/pdf) and requirements simulation (prolog). It relies on Eclipse technologies as XText for textual specification and Sirius for graphical views of the textual specifications

Chapter 4
System Architecture

This chapter presents information concerning the architecture of the software system both from the static and dynamic viewpoints. The static viewpoint focuses on the physical architecture (hardware) required to deploy and run the software system along with the manner in which the components that make such software system are grouped. On the other hand, the dynamic viewpoint focuses on the behaviour of the software system at runtime.

The static information is presented through the Deployment View and the Implementation View. The dynamic aspects of the system are presented by means of the UI Processing View.

4.1 Deployment view

The aim of the Deployment View is to describe the different processing nodes that compose the deployment infrastructure and how they are interconnected. A processing node corresponds to a piece of hardware aimed at executing either the whole software system or a sub-part of it.

4.2 Implementation view

The Implementation View describes each software system component and how they are organised and combined to make the targeted software system.

4.2.1 Component xx.yy.zz.c1

TODO

4.2.2 Component xx.yy.zz.c2

TODO

4.2.3 Component xx.yy.zz.c3

TODO

4.3 UI Processing view

A UI Processing View is aimed at explaining the required message exchanges to achieve the launching of a system operation (specified in the Messir Analysis Document). These required message exchanges (which are not specified in the Messir Analysis Document) make part of the user interface (UI). Thus, the main interest of a UI Processing View is to describe the design choices made at the UI level, such that a system operation is launched. The description of a UI Processing View is given by means of a UML Sequence Diagram.

A complete Design Document should contain a UI Processing View for each non-proactive system operation specified in the Messir Analysis Document, as such kind of system operations are launched by actors through UIs that allows them to make so.

4.3.1 UI Processing view for system operation oeSystemOperation1

TODO

4.3.2 UI Processing view for system operation oeSystemOperation2

TODO

4.3.3 UI Processing view for system operation oeSystemOperation3

TODO

4.4 Non-functional runtime concerns

The description of the runtime processes should be complemented with free textual information regarding concurrency, distribution, performance and scalability aspects.

4.4.1 Performance

TODO

4.4.2 Concurrency and Parallelism

TODO

4.4.3 Scalability

TODO

Chapter 5
Detailed design

Provide an introduction to the proposed design. This introduction should help the reader to understand the design choices made.

5.1 Interaction Model

An Interaction Model describes how each System Operation (that appears in the Operation Model of the Messir Requirements Analysis Model) is designed to meet its specification. The design description of each system operation must be focused on the messages exchanged between the different first-class objects (i.e. instances of classes either included in Concept Model or introduced as result of a design choice). An Interaction Model is modeled as a UML Sequence Diagram.

5.1.1 oeSystemOperation1

TODO

5.1.2 oeSystemOperation2

TODO

5.1.3 oeSystemOperation3

TODO

5.2 Design Class Model

The Design Class Model is composed of the contents of all design classes (i.e. every class appearing in at least one Interaction Model), all the navigable associations between design classes, and the inheritance structure. The description of each class must contain its attributes and operations. The Design Class Model is modeled as a UML Class Diagram.

It is advised to split the Design Class Model into multiple views as such model may become pretty large.

5.2.1 Design Class Model view1

TODO

5.2.2 Design Class Model view2

TODO

5.2.3 Design Class Model view3

TODO

Chapter 6
Known limitations

All known and non solved issues (like bugs, missing functionalities, abnormal behavior, etc.) should be precisely stated and described.

6.1 Issue 1

TODO

6.2 Issue 2

TODO

Chapter 7
Conclusion

TODO

References