SOFTWARE REQUIREMENTS SPECIFICATION AND TECHNOLOGY NEUTRAL PROCESS DESIGN

for

COS 301 Mini Project

Prepared by
Team Lima

University Of Pretoria

February 20, 2016

Contents

1 Introduction **4**
- 1.1 Purpose . . . 4
- 1.2 Document Conventions . . . 4
- 1.3 Intended Audience and Reading Suggestions . . . 4
- 1.4 Project Scope . . . 4
- 1.5 References . . . 4

2 Vision **5**
- 2.1 Product Functions . . . 5
- 2.2 User Classes and Characteristics . . . 5
- 2.3 Operating Environment . . . 5

3 Background **6**

4 Architecture requirements **7**
- 4.1 Access channel requirements . . . 7
- 4.2 Quality requirements . . . 7
- 4.3 Integration requirements . . . 7
- 4.4 Architecture constraints . . . 7

5 Functional requirements and application design **8**
- 5.1 Use case prioritization . . . 8
- 5.2 Use case/Services contracts . . . 8
- 5.3 Required functionality . . . 8
- 5.4 Process specifications . . . 9
- 5.5 Domain Model . . . 9

6 Open Issues **10**

1 Introduction

1.1 Purpose

Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.

1.2 Document Conventions

Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.

1.3 Intended Audience and Reading Suggestions

Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.

1.4 Project Scope

Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.

1.5 References

List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.

2 Vision

A short discussion of the project vision, i.e. what the client is trying to achieve with the project and the typical usage scenarios for the outputs of the project.

2.1 Product Functions

Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.

2.2 User Classes and Characteristics

Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.

2.3 Operating Environment

Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

3 Background

A general discussion of what lead to the project including potentially

- business/research opportunities,
- opportunities to simplify/improve some aspect of life/work or community,
- problems your client is currently facing, ss
- ...

4 Architecture requirements

The software architecture requirements include the access and integration requirements, quality requirements and architectural constraints.

4.1 Access channel requirements

Specify the different access channels through which the system's services are to be accessed by humans and by other systems (e.g. Mobile/Android application clients, Restful web services clients, Browser clients, . . .).

4.2 Quality requirements

Specify and quantify each of the quality requirements which are relevant to the system. Examples of quality requirements include performance, reliability, scalability, security, flexibility, maintainability, auditability/monitorability, integrability, cost, usability. Each of these quality requirements need to be either quantified or at least be specified in a testable way.

4.3 Integration requirements

This section specifies any integration requirements for any external systems. This may include

- the integration channel to be used,
- the protocols to be used,
- API specifications in the form of UML interfaces and/or technology-specific API specifications (e.g. WSDLs, CORBA IDLs, . . .), and
- any quality requirements for the integration itself (performance, scalability, reliability, security, auditability, . . .).

4.4 Architecture constraints

This specifies any constraints the client may specify on the system architecture include

- technologies which MUST be used,
- architectural patterns/frameworks which must be used (e.g. layering, Services Oriented Architectures, . . .)

...

5 Functional requirements and application design

This section discusses the application functionality required by users (and other stakehodlers).

5.1 Use case prioritization

Consider a simple three-level prioritization with

Critical: A use case which is absolutely essential (ask whether the project should be canceled if that functionality could not be provided).

Important: The system would still be useful without some of the important use cases, but the client would get quantifiably less value from the system.

Nice-To-Have: Its a requirement but the value to the client/business is insignificant/not quantifiable.

5.2 Use case/Services contracts

For each use case/service specify

Pre-Conditions: the conditions under which the service may be refused (usually there is an exception associated with each pre-condition).

Post-Conditions: the conditions which must hold true after the servcie has been provided.

Request and Results Data Structures: Use class diagrams to specify the data structure requirements for the request and result objects (i.e. the inputs and outputs). You will find an example for the UML documentation of a service contract in chapter 10 of "Technology-Neutral Analysis and Design using UML and URDAD".

5.3 Required functionality

Use for each concrete use case a use case diagram with the required functionality in the form of includes and extends relationships to lower level use cases – this may be specified across levels of granularity.

// DIAGRAM - refer to requirements_doc.pdf //

An example UML use case diagram specifying the functional requirements for a use case is show in Figure 1. We show the required functionality and the services domains represented by interfaces from which these services will be sourced.

5.4 Process specifications

For some of the use cases there may be requirements around the process which needs to be followed. If so, these requirements are typically specified via activity and/or sequence diagrams or alternatively via state charts.

// DIAGRAM - refer to requirements_doc.pdf //

An example UML of an activity diagram for a process specification is shown in Figure 2. The outer activity is the activity of the service for which we are doing the process design. It starts with a request. For each pre-condition there is a path leading to an exception being thrown and the service being aborted. If all pre-conditions are met, the service will return the return value. Note that each inner activity is a UML call operation, requesting a particular service from a particular interface, i.e. ultimately from some class realizing the service contract represented by that interface. This ensures decoupling of clients and services, i.e. that we can plug in any service provider implementing a services contract and that we are not locked into using a particular service provider (class).

5.5 Domain Model

Use UML class diagrams to specify the data structure requirements in a technology neutral way. These can ultimately be mapped onto different technologies like ERD diagrams/relational databases, XML schemas, Python/Java/C++/. . . objects, paper based or UI forms, . . . but those are just different technology mappings and this would not be part of the requirements specification.

// DIAGRAM - refer to requirements_doc.pdf //

An example UML class diagram for a domain model is shown in Figure 3. The domain model will show the domain objects, their attributes and potentially methods and the relationships between them (e.g. association, aggregation, composition, specialization, realization and containment). Dependency relationships are usually not explicitly shown.

6 Open Issues

Discuss in this section

- any aspects of the requirements which still need to be specified,
- around which clarification is still required, as well as
- any discovered inconsistencies in the requirements.