A Template for A Final Project in Software Engineering Application Requirements Document[1]

Last revised on October 21, 2018

[1]Derived from a previous documents authored by *Dr. Eliezer Kaplansky* and *Dr. Mayer Goldberg.*

Abstract

This document is the first of three documents describing the final project in the software engineering program. Specifically, this document describes the requirements of the software project *from the standpoint of the customer.* [here you should place a short summary of the project]

Contents

1 Introduction **3**
- 1.1 The Problem Domain . . . 3
- 1.2 Context . . . 3
- 1.3 Vision . . . 3
- 1.4 Stakeholders . . . 3
- 1.5 Software Context . . . 4

2 Usage Scenarios **5**
- 2.1 User Profiles — The Actors . . . 5
- 2.2 Use-cases . . . 5
- 2.3 Special usage considerations . . . 5

3 Functional Requirements **6**

4 Non-functional requirements **7**
- 4.1 Implementation constraints . . . 7
- 4.2 Platform constraints . . . 8
  - 4.2.1 SE Project constraints . . . 8
- 4.3 Special restrictions & limitations . . . 8

5 Risk assessment & Plan for the proof of concept **9**

Guidelines

This document formalizes your understanding of what your customer has requested. In writing this document, you should maintain a global view of the proposed software system, its purpose, use, and relation to the rest of the world. Avoid low-level details; Those will go in the application design document (ADD).

You should analyze the project description and the requirements that the customer expressed and come up with your own suggestions for additional and/or improved and/or more general functionality. Do not worry that you will need to implement all these extensions. After this document (ARD) is submitted and accepted, we will define what functionality needs to be implemented for the prototype, what will be implemented in Version 1.0 (which will be delivered by the end of the year), and what functionality will be reserved for future versions.

Chapter 1

Introduction

This chapter provides an overview of the entire requirements document. The goal is to provide the "big picture" into which your software projects should fit. Describe the systems that surround your project. Very few industrial projects will be entirely self-contained. The following sections document the context of your software system and how it should interact with the rest of the world.

1.1 The Problem Domain

Provide an overview of the domain you project is related to.

1.2 Context

Describe in detail the context in which your system operates. If needed, include a block diagram showing the major components of the larger system, how they inter-connect, and what external interfaces are maintained. Do not dive into design details. Your system should usually be a single block in the diagram.

1.3 Vision

Describe the overall goals and objectives of your software project.

1.4 Stakeholders

Who are the people that are relevant to the product? Who are your customers? Who has a say on the requirements?

In many systems, there are people, beyond the end customers, that use the system to, e.g., produce reports, design internal rules, etc. List them as stakeholders.

1.5 Software Context

Provide a high-level description of the software system you are planning. Describe the major inputs, functionality, processing, and outputs. Omit implementation and low-level details. Describe several use-cases of common usage scenarios that illustrate how users, the environment, and other systems will interact with your software.

Chapter 2

Usage Scenarios

In this chapter you will describe the main usage scenarios of your software system. You are to distill these scenarios from the information collected during the requirements elicitation process.[1]

2.1 User Profiles — The Actors

Describe the profiles of the main user categories. Describe the relevant characteristics of the intended users of the system. Two things to keep in mind about actors are that

- Actors are external to the system
- Actors exchange data with the system

Humans that use your system are certainly good candidates for actors. Actors can be non-human, but keep in mind that actors must be external to your system. Another way of saying this is that *actors begin where the system ends.* A reasonable candidate for a non-human actor could be, e.g., a radioactivity sensor that triggers some emergency system.

2.2 Use-cases

Describe the main use-cases for the software system.

2.3 Special usage considerations

Describe any special requirements associated with the use of this software system.

[1] **Hint:** Think about the relationship between the usage scenarios and the test cases you will use to test your system.

Chapter 3

Functional Requirements

- Describe each of the major functions to be handled by the software.
    - By *functions* we mean the capabilities and functionality of the software, itemized as individual operations. We do not mean functions in the programming languages sense of program units.
- The description of the functional requirements should be understandable to your customer, and indeed, to anyone else reading the document for the first time.
- It will be helpful to organize the functions into some hierarchical taxonomy.
- You may use charts of various kinds (e.g., use-case and sequence diagrams) to help the reader grasp the logical relationship between actors, functions, and data.

Chapter 4

Non-functional requirements

- Describe any constraints that affect the way your software system is to be specified, designed, implemented, tested, etc.
- These constraints may be related to the nature of the business of your customer, the nature of the product line, etc.
- Make sure you quantify these requirements so that they are measurable, demonstrable and verifiable.
- Here are three examples of how to state a requirement, the first acceptable, the second unacceptable, and the third even worse:

  Acceptable: 95% of all transactions will complete in under one second.

  Unacceptable: Transactions should complete quickly.

  Worse: The user should not have to wait for a transaction to complete.

- You may use as guidelines the following divisions; Feel free to delete some and/or add others.

4.1 Implementation constraints

Performance (Speed, Capacity, Throughput, etc.). Quantify the performance requirements for your software system.

Reliability & Stability. Is your system required to withstand certain hardware, software, network failures? Is your system required to support data recovery, error-correction, etc? describe and quantify the factors that effect the reliability & stability of the software system.

Safety & Security. Is confidentiality a constraint in your system? Does your data need to be encrypted? Are different views of your data accessible to users with different security/authority levels? Etc.

Portability. Should your system be deployable from different operating systems? If your system is a web service, does your system depend on a specific browser? Should your system support text in different languages and character sets (e.g., using Unicode)? Etc.

Usability: What is the level of expertise that the users of your system are expected to have? What is the level of training required to begin using the application?

Availability: Describe and quantify the level of availability required of the system. Describe what factors bear on the availability of the system.

4.2 Platform constraints

Describe any constraints that limit/restrict your choice of development tools, software packages, platforms, etc. E.g., some projects may require you to use Visual Basic and ASP, restrict you to .NET, etc.

4.2.1 SE Project constraints

These constraints are clearly artificial in the context of an industrial software project, but nevertheless, the reality of having to develop your software within a university course will impose certain rather rigid constraints on your system:

- You will need to give a demo of your system. Is the system interactive? Where will its inputs come from? If the inputs arrive naturally from some device, will you have access to this device? Will you have to simulate the device? Will you use random data or samples of actual data in your simulation?
- If the software system requires special hardware, special operating conditions, special access rights, access to proprietary or secret data, etc, how and where will you develop the software? How will you test it? How will you present the final system?

4.3 Special restrictions & limitations

Describe the assumptions that underly and affect the requirements and constraints listed herein. For example, the availability of certain tools or hardware, special operating conditions, etc. Some of the design constraints are derived from these assumptions and dependencies.

Describe any additional constraints not covered by previous sections. For example, users with specific disabilities, conformance to various standards, etc.

Chapter 5

Risk assessment & Plan for the proof of concept

In this chapter you will plan the next step of the project in which you will develop a proof-of-concept prototype.

The main objective of a proof-of-concept is to determine the solution to some technical problem, such as how two systems might be integrated or that a certain throughput can be achieved with a given configuration.

A proof-of-concept can be a partial solution that involves a limited amount of functionality and scope to establish whether the system satisfies some aspect of the requirements and to test various implementation options. It is also meant for you to become acquainted with the technologies and form opinions on how to use them best in the final system design. Specifically, when describing the planned prototype (and when presenting it later), make sure to convince the reader how the prototype is expected to give:

- Better understanding of requirements;
- Understanding of the capabilities and limitations of new technologies;
- The ability to assess design decisions early in the process;
- The ability of the customer to visualize the look-and-feel of the final solution;
- And, most impotently: **reduction of the risk of project failure.**

Appendices

Include any information that is relevant to the project, and that supplements the requirements specification. Such information includes, but is not limited to:

- I/O format information, such as file formats, etc.
- Reports of cost analysis studies, user surveys, descriptions and/or surveys of similar products, etc.
- Supporting and/or background information to help readers of this ARD.
- Glossary: List of all the domain-specific and technical terms with a brief definition.

Your customer is a likely source for such information.