A Template for
A Final Project in Software Engineering
Application Requirements Document*

Mayer Goldberg†

Last revised on September 8, 2005

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 client.* Please consult the supplementary document *Protocol On Written Reports* for detailed guidelines on how to use this template.

Guidelines

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

You should analyze the PD 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 the 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. While part of your work will involve planning the next major version, you will not need to implement it, unless of course, your client is excited enough about the Version 1.0 that they are ready to hire you to implement the next version! In any event, studying and extending the problem domain will provide a larger selection of features from which to choose those that you will actually implement.

*Derived from a previous document authored by *Mr Eliezer Kaplansky.*

†Dr Mayer Goldberg, Office 58:312, Department of Computer Science, Ben Gurion University, Beer Sheva 84105. Office phone: (08)647-7873. Email: gmayer@cs.bgu.ac.il.

1 Introduction

This section provides an overview of the entire requirements document. The goal is to provide the "big picture" into which your software projects should fit. Very few industrial projects will be entirely self-contained. If you happen to have come upon such a project, state and justify this claim clearly. Otherwise, the following subsections document the context of your software system and how it should interact with the rest of the world.

1.1 Vision

Describe the overall goals and objectives of your software project.

1.2 The Problem Domain

Describe in detail the problem domain in which your system operates. Include a block diagram showing the major components of the larger system, how they inter-connect, and what external interfaces are maintained.

1.3 Stakeholders

Who are the people that are relevant to the product? Who are your customers? Who has a say on the design? Some examples could include:

- Experts
- Users
- Sponsors

1.4 Software Context

Provide a high-level description of the software you are planning. Describe the major inputs, functionality, processing, and outputs. Omit implementation and low-level details.

1.5 System Interfaces

Describe the main system interfaces. Describe the functionality your software will need to implement in order to match the requirements specified and implied by the interfaces.

1.5.1 Hardware Interfaces

Describe the interface between your software and the hardware components of the underlying system. Describe the devices that are supported, what interface protocols are needed, and how your software will support them.

1.5.2 Software Interfaces

Describe the interface between your software and other software components of the underlying system. Describe their use, the interfaces they require, and how your software will support them.

1.5.3 Events

Detail and describe the events that control the behavior of the system.

2 Functional Requirements

Describe each of the major functions to be handled by the software[1].

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., sequence diagrams) to help the reader grasp the logical relationship between actors, functions, and data.

3 Non-functional requirements

Describe any constraints that effects the way your software is to be specified, designed, implemented, tested, etc. These constraints may be related to the nature of the business of your client, 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: Ninety five percent 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:

[1] 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.

Performance constraints

Examples of such constraints include

Speed, Capacity & Throughput. Quantify the performance requirements for your software. Such requirements may include:

- How long does it take to process a transaction?
- How long does it take for critical conditions to be detected?
- The number of users that can access the system simultaneously
- The maximum number of open input sources the system can handle simultaneously

Reliability. Describe and quantify the factors that effect the reliability of the software.

Safety & Security. Is confidentiality a constraint in your system? Does your data need to be encrypted? What cryptographic protocols will you be using? 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 Web browser? Should your system support text in different languages and character sets (e.g., using Unicode)? Etc.

Reliability. Is your system required to withstand certain hardware, software, network failures? Is your system required to support data recovery, self-stabilization, error-correction, etc?

Usability: Describe and quantify the effort required of the user to learn and operate the system, prepare input for it, and interpret its output.

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

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.

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 its 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 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?

Special restrictions & limitations

Describe the assumptions that underly and effect 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.

4 Usage Scenarios

In this section you will describe the main usage scenarios of your software. You are to distill these scenarios from the information collected during the requirements elicitation process.[2]

4.1 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, and 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.

4.2 Use-cases

Describe the main Use-cases for the software.

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

4.3 Special usage considerations

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

5 Appendices

Include any information that is relevant to the project, and that supplements the requirements specification. Such information includes:

- 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- and technical terms with a brief definition.

Your client is a likely source for such information.