SO1: Software Engineering 2008-9

User Requirements Document outline

This outline is adapted slightly from section 5.6 of the European Space Agency's "Guide to the user requirements definition phase" ESA PSS-05-02. The text in italics provides guidance on the contents, and should NOT be included in any real requirements document.

You're welcome to change the style but not the structure! Feel free, for example, to put a nice title page here.

Note the use of Word Styles which enable easy alterations to style of headers, and production of table of contents.

An example of requirements created with this document is available on the module website

If there is no material for a section then the phrase 'Not Applicable' should be inserted and the section numbering preserved.

Material unsuitable for the contents list above should be inserted in additional appendices.

DOCUMENT STATUS SHEET

DOCUMENT STATUS SHEET			
Document Title :			
ISSUE	**REVISION**	**DATE**	**REASON FOR CHANGE**

Sample entries would be something like:

1 0 12/06/07 First Issue

1 1 29/08/07 Section 3.1 Requirement number 23 added after discussion with X

Etc etc

1 INTRODUCTION

1.1 Purpose

What is the product? Who are the readers?

1.2 Scope

Identify the software products by name, outline what the software will do (and what it will not do, if necessary), goals, objectives and benefits.

1.3 Definitions, acronyms and abbreviations

For each, define or reference

1.4 References

Any linked URDs, any other relevant documents – possible none for group project, unless used to assist in refining the specification.

1.5 Overview

Contents and organization of the rest of the URD

2 GENERAL DESCRIPTION

Purpose of this section – to make specific requirements easier to understand. Readable by all.

2.1 Product perspective

How does this product relate to other systems, existing or under development. Could be 'standalone'

2.2 General capabilities

Main section containing high level descriptive functional requirements in natural language, supported by tables and diagrams. UML Use cases, activity or highlevel sequence diagrams should be included here.

2.3 General constraints

Explain what constraints there are and why they exist. Natural language version of non-functional requirements

2.4 User characteristics

What level of experience, such as educational level, language, experience and technical expertise do the users, operators and maintenance personnel have, that will impose constraints on the software? How many users will be frequent users who will become experts, and how many infrequent users who will stay novices.

2.5 Operational environment

How does this system fit with other systems – any interfaces? Could use context or system block diagrams.

2.6 Assumptions and dependencies

List any assumptions that the specific requirements are based on, or any other systems being developed whose existence this system depends on.

3 SPECIFIC REQUIREMENTS

Sections 3.1 and 3.2 are the core of the URD. The acceptability of the software will be assessed with respect to the specific requirements.
Follow these guidelines in each section:

- *Each requirement must be uniquely identified.*
- *If a requirement is essential, it must be clearly flagged. Non-essential requirements should be marked with a measure of desirability (e.g. scale of 1, 2, 3).*
- *The priority of a requirement measures the order, or the timing, of the related software becoming available. If the transfer is to be phased, so that some parts of the software come into operation before others, then each requirement must be marked with a measure of priority.*
- *Unstable requirements, which are dependent on feedback from later design stages, should be marked 'TBC'*
- *The source of each requirement must be stated. The source may be defined using the identifier of a system requirement, a document cross-reference or even the name of a person or group.*
- *Each requirement must be verifiable. Clarity increases verifiability. Each statement of user requirement should contain one and only one requirement. A requirement is verifiable if some method can be devised for objectively demonstrating that the software implements it.*
- *The user must describe the consequences of losses of availability and breaches of security, so that the developers can fully appreciate the criticality of each function*

3.1 Capability requirements

Here are the functional requirements, precisely defined. The organisation of the capability requirements should reflect the problem, and no single structure will be suitable for all cases.

Each capability requirement should be checked to see whether the inclusion of capacity, speed and accuracy attributes is appropriate.
Details of data requirements can go here as part of the functional requirements.

3.2 Constraint requirements

Her are the non-functional requirements, precisely defined. Constraint requirements may cover any topic that does not directly relate to the specific capabilities the users require.

Constraint requirements that relate to interfaces should be grouped around the headings:

- *communications interfaces;*
- *hardware interfaces;*
- *software interfaces;*
- *human-computer interactions (user interfaces).*

Requirements that ensure the software will be fit for its purpose should be stated, for example: adaptability; availability; portability; security; safety; standards.