Software Engineering Group Projects – Requirements Specification Standards

Author: H.R. Nicholls
Proj. Ref.: Software Engineering Quality Assurance
Client: Department of Computer Science, UWA
Config. Ref.: SE.QA.04
Date: 02 Oct 1995
Version: 3.4
Status: Release

Department of Computer Science,
University of Wales,
Aberystwyth,
Dyfed SY23 3DB,
U.K.

CONTENTS

DOCUMENT CHANGE HISTORY

Version	*CCF No.*	*Date*	*Sections changed from previous version*	*Change d b y*
1.1	N/A	27 Nov 92	initial development	HRN
1.2	N/A	27 Nov 92	3.1.1, 3.1.4, 3.2.2, 3.2.5	HRN
2.1	N/A	27 Nov 92	first draft	HRN
3.1	N/A	27 Nov 92	first release	HRN
3.2	29	12 Nov 93	sccs style incorporated	FWL
3.3	37	8 Oct 94	revised section 3.2.1.3	MBR
3.4	40	1 Oct 95	revised section 3.2.1	GCC
			revised section 1	
		3 Oct 95	moved change history	

1 INTRODUCTION

1.1 Purpose Of This Document

The purpose of this document is to describe the format of, and information which must be supplied in, requirements specifications produced in software engineering group projects.

The requirements specification is one of the most important project documents, since it specifies what the end system will do. Lack of clarity or completeness in the specification will lead to uncertainty and errors later in the project. This document aims to specify the main topics which must be covered by a requirements specification, and to thereby help in the elimination of uncertainty. The requirements specification must be written in close consultation with the client.

1.2 Scope

This document specifies the standards for writing software requirements specifications. It describes the necessary layout and content of a requirements specification. This document should be read by all project members.

It is assumed that the reader is already familiar with the QA Plan [1].

1.3 Objectives

The main objective of this document is to aid the production of a specification which is:

- unambiguous;
- complete;
- verifiable;
- consistent;
- modifiable;
- traceable;
- usable.

This objective is to be achieved by providing an outline structure of a model requirements specification, and describing in detail the components of this structure.

2 RELEVANT QA DOCUMENTS

The Requirements Specification must be produced in accordance with the quality standards specified in the QA Plan [1]. In particular, it must be produced as part of a specified task in the Project Plan [2] and maintained within the configuration management system according to the appropriate procedures [4].

The basic layout and information content must conform to the general documentation standards [3], which, amongst other items, specifies that there must be an introductory section and a references section. The remainder of this QA document (i.e., SE.QA.04), is concerned with the other sections which should be included in a requirements specification.

3 OUTLINE STRUCTURE

An outline structure for a requirements specification is presented in Figure 1. All requirements specifications must have this general structure.

It is recognised, however, that the structure of section 3 in the outline may not be appropriate for some software projects. Thus, all requirements specifications *must* have the same structure as the outline for sections *other than* the Specific Requirements section. The Specific Requirements section should address the same issues as those indicated in the outline but in an order or format appropriate to the particular system being specified. By default, the structure of section 3 in the suggested outline should be used.

A description of the outline structure is presented below.

3.1 General Description

The General Description section should describe the *general* factors that affect the product and its requirements.

3.1.1 Product Perspective

This should put the product into perspective with other products or projects. For example, it should be stated whether the product is to be a totally self contained program or a part of a larger system, and whether it replaces/upgrades an existing system or is entirely novel.

3.1.2 Product Functions

This should provide a *summary* of the functions that the software will perform. The functions should be organised in a way that makes the list of functions understandable to anyone reading the document for the first time.

```
1 INTRODUCTION
    (structure as defined in QA Document SE.QA.03)
2 GENERAL DESCRIPTION
    2.1 Product Perspective
    2.2 Product Functions
    2.3 User Characteristics
    2.4 Constraints, Assumptions and Dependencies
3 SPECIFIC REQUIREMENTS
    3.1 Functional Requirements
          3.1.1 Functional Requirement 1
             3.1.1.1 Introduction
             3.1.1.2 Inputs
             3.1.1.3 Outputs
             3.1.1.4 Mapping
          3.1.2 Functional Requirement 2
          . . .
          3.1.n Functional Requirement n
    3.2 External Interface Requirements
          3.2.1 User Interfaces
          3.2.2 Hardware Interfaces
          3.2.3 Software Interfaces
    3.3 Performance Requirements
    3.4 Design Constraints
          3.4.1 Standards Compliance
          3.4.2 Hardware Limitations
          . . .
    3.5 Other Requirements
4 REFERENCES
    (structure as defined in QA Document SE.QA.03)
APPENDICES
```

Figure 1: *Outline structure for a requirements specification.*

3.1.3 User Characteristics

This should describe those general characteristics of the eventual users of the product that will affect the specific requirements. For example, if the eventual users will be computer novices, then this might result in a requirement for online help information; if the eventual users are computer experts, then a resulting requirement might be for 'shorthands' to be provided for commonly executed commands.

3.1.4 Constraints, Assumptions and Dependencies

This should list each of the factors that affect the requirements. Constraints might include interfaces to other software items, or security and safety considerations. An

assumption might be that the Unix operating system is available for the product to run under.

3.2 Specific Requirements

The Specific Requirements section should contain all the detailed information the software developer needs to create a design; it should therefore be a clear statement of *what* the system is to do, enabling the designer to decide later *how* to do it.

3.2.1 Functional Requirements

This should describe how the inputs to the software product are to be transformed into outputs. It describes the fundamental actions which must be performed by the software. For each class of function, the requirements can be specified under the following headings:

3.2.1.1 Introduction

This should provide a description of the purpose of the function.

3.2.1.2 Inputs

This should contain a detailed description of all data input to the function, including: the source of the inputs (e.g., the operator, a sensor, another software module), quantities (e.g., two integers representing an (x, y) coordinate), units of measure (e.g., millimetres), the ranges of valid inputs, etc.

3.2.1.3 Outputs

This should contain a detailed description of all data output from the function. It should include destination of the output (e.g., the screen, a piece of equipment, another program), quantities, units of measure, the ranges of valid outputs, and error messages.

3.2.1.4 Mapping

This should define the relationships, that are relevant to the user, between the inputs, outputs (and side effects). In order to do this it may be necessary to outline the algorithms that are likely to be used in the design, but it is important to avoid this as far as possible. *How* the product works is a matter for the Design Specification. However, validity checks on the input, and responses to abnormal situations (e.g., response to overflow, error handling) should be covered.

3.2.2 External Interface Requirements

This should describe the interface between the software product and all other items. The main categories of interface specification are described below.

3.2.2.1 User Interfaces

This should specify the characteristics of the human-computer interface. If the human operator is to interact with the system through a display, then screen layouts, menus, forms, etc. should be specified here. This subsubsection should also include aspects of usability, such as specifying whether long or short error messages are to be supplied, and whether 'shorthands' for commands will be made available.

3.2.2.2 Hardware Interfaces

This should specify the hardware characteristics of each interface between the software and the hardware components of the system. For example, in the case of the product using a terminal, it should be specified whether full-screen cursor movement control or simple line-by-line output is required, or whether a colour or a monochrome display is required.

3.2.2.3 Software Interfaces

This should specify the use of other required software products (e.g., a database system, an operating system, a graphics tool). For each required software product, the following should be specified:

- name;
- version number;
- source or supplier.

The interfaces with other application packages (e.g., to an Ada data structure package in a publically-accessible library) should be identified, and reference made to the document defining such interfaces.

3.2.3 Performance Requirements

This should specify the static and dynamic performance requirements placed on the software. Static requirements may include: the number of terminals to be supported, the number of simultaneous users to be supported, the number of files and records to be handled, the sizes of tables and files. Dynamic requirements may include, for example, the amount of data to be processed within certain time periods. These requirements should be expressed in measurable terms; e.g., *95% of the user operations shall be processed within 1 second*, rather than *the operator shall not have to wait long for the operation to complete.* Note that these are general performance

requirements, if a different limit applies to a specific function, then that limit should be specified as part of the subsubsection on 'Processing' for that function.

3.2.4 Design Constraints

Design constraints may be imposed by other standards, hardware limitations, etc. and should be listed under this subsection.

3.2.4.1 Standards Compliance

This should specify requirements derived from existing standards if appropriate. For example, a new timetabling program might be required to produce timetables in the same format as those produced by the existing system.

3.2.4.2 Hardware Limitations

This could include requirements for software to operate inside various hardware constraints. For example, a program may have to run inside 640 Kbytes of memory, or a database program might have to fit all its files on to a 20 Mbyte disk.

3.2.5 Other Requirements

This should specify requirements that aren't covered by any of the other sections. An example is specifying the security of the system with respect to its protection from accidental or malicious access and use. Specific security requirements might include the use of passwords and encryption, or the keeping of logs of operations performed.

3.3 References and Appendices

The reference list must be presented according to the guidelines given in QA document QA.SE.03 [3]. Any background information which is not directly applicable to the requirements, but which needs to be presented, should be placed in the appendices.

REFERENCES

[1] H.R. Nicholls. Software engineering group projects – quality assurance plan. Technical Report SE.QA.01, University of Wales, Aberystwyth, 1997.

[2] H.R. Nicholls. Software engineering group projects – project management standards. Technical Report SE.QA.02, University of Wales, Aberystwyth, 1997.

[3] H.R. Nicholls. Software engineering group projects – general documentation standards. Technical Report SE.QA.03, University of Wales, Aberystwyth, 1997.

[4] H.R. Nicholls. Software engineering group projects – operating procedures and configuration management standards. Technical Report SE.QA.08, University of Wales, Aberystwyth, 1997.