Software Requirements Specification

The sub-sections :

What is a Software Requirements Specification

A software requirements specification is a document which is used as a communication medium between the customer and the supplier. When the software requirement specification is completed and is accepted by all parties, the end of the requirements engineering phase has been reached. This is not to say, that after the acceptance phase, any of the requirements cannot be changed, but the changes must be tightly controlled. The software requirement specification should be edited by both the customer and the supplier, as initially neither has both the knowledge of what is required (the supplier) and what is feasible (the customer).

Why is a Software Requirement Specification Required

A software requirements specification has a number of purposes and contexts in which it is used. This can range from a company publishing a software requirement specification to companies for competitive tendering, or a company writing their own software requirement specification in response to a user requirement document. In the first case, the author of the document has to write the document in such a way that it is general enough as to allow a number of different suppliers to propose solutions, but at the same time containing any constraints which must be applied. In the second instance, the software requirement specification is used to capture the users requirements and if any, highlight any inconsistencies and conflicting requirements and define system and acceptance testing activities.

What is Contained in a Software Requirements Specification

A software requirement specification in its most basic form is a formal document used in communicating the software requirements between the customer and the developer. With this in mind then the minimum amount of information that the software requirement specification should contain is a list of requirements which has been agreed by both parties. The types of requirements are defined in section 3.4. The requirements, to fully satisfy the user should have the characteristics as defined in section 3.5. However the requirements will only give a narrow view of the system, so more information is required to place the system into a context which defines the purpose of the system, an overview of the

systems functions and the type of user that the system will have. This additional information will aid the developer in creating a software system which will be aimed at the users ability and the clients function.

Types of Requirements

Whilst requirements are being collated and analysed, they are segregated into type categories. The European Space Agency defined possible categories as [ESA '87]:

- Functional requirements,
- Performance requirements,
- Interface requirements,
- Operational requirements,
- Resource requirements,
- Verification requirements,
- Acceptance testing requirements,
- Documentation requirements,
- Quality requirements,
- Safety requirements,
- Reliability requirements and
- Maintainability requirements

Functional Requirements

Functional or behavioural requirements are a sub-set of the overall system requirements. These requirements are used to consider trade-offs, system behaviour, redundancy and human aspects. Trade-offs may be between hardware and software issues, weighing up the benefits of each. Behavioural requirements, as well as describing how the system will operate under normal operation should also consider the consequences and response due to software failure or invalid inputs to the system

Performance Requirements

All performance requirements must have a value which is measurable and quantitative, not a value which is perceptive. Performance requirements are stated in measurable values, such as rate, frequency, speeds and levels. The values specified must also be in some recognised unit, for example metres, centimetre square, BAR, kilometres per hour, etc. The performance values are based either on values extracted from the system specification, or on an estimated value.

Interface Requirements

Interface requirements, at this stage are handled separately, with hardware requirements being derived separately from the software requirements. Software interfaces include dealing with an existing software system, or any interface standard that has been requested. Hardware requirements, unlike software give room for trade-offs if they are not fully defined, however all assumptions should be defined and carefully documented.

Operational Requirements

Operational requirements give an "in the field" view to the specification, detailing such things as:

how the system will operate,
what is the operator syntax,
how the system will communicate with the operators,
how many operators are required and their qualification,
what tasks will each operator be required to perform,
what assistance/help is provided by the system,
any error messages and how they are displayed, and
what the screen layout looks like.

Resource Requirements

Resource requirements divulge the design constraints relating to the utilisation of the system hardware. Software restrictions may be placed on only using specific, certified, standard compilers and databases. Hardware restrictions include amount, percentage or mean use of the available memory and the amount of memory available. The definition of available hardware is specially important when the extension of the hardware, late in the development life cycle is impossible or expensive.

Verification Requirements

Verification requirements take into account how customer acceptance will be conducted at the completion of the project. Here a reference should be made to the verification plan document. Verification requirements specify how the functional and the performance requirements are to be measured and verified. The measurements taken may include simulation, emulation and live tests with real or simulated inputs. The requirements should also state whether the measurement tests are to be staged or completed on conclusion of the project, and whether a representative from the client's company should be present.

Acceptance Testing Requirements

Acceptance test requirements detail the types of tests which are to be performed prior to customer acceptance. These tests should be formalised in an acceptance test document.

Documentation Requirements

Documentation requirements specify what documentation is to be supplied to the client, either through or at the end of the project. The documentation supplied to the client may include project specific documentation as well as user guides and any other relevant documentation.

Quality Requirements

Quality requirements will specify any international as well as local standards which should be adhered to. The quality requirements should be addressed in the quality assurance plan, which is a core part of the quality assurance document. Typical quality requirements include following ISO9000-3 procedures [Ince '94]. The National Aeronautics and Space Administrations software requirement specification - SFW-DID-08 goes to the extent of having subsections detailing relevant quality criteria and how they will be met. These sections are[NASA]:

- Quality Factors
- Correctness
- Reliability
- Efficiency
- Integrity
- Usability
- Maintainability
- Testability
- Flexibility
- Portability
- Reusability
- Interoperability
- Additional Factors

Some of these factors can be addressed directly by requirements, for example, reliability can be stated as an average period of operation before failure. However most of the factors detailed above are subjective and may only be realised during operation or post delivery maintenance. For example, the system may be vigorously tested, but it is not always possible to test all permutations of possible inputs and operating conditions. For this reason errors may be found in the delivered system. With correctness the subjectiveness of how correct the system is, is still open to interpretation and needs to be put into context with the overall system and its intended usage. An example of this can be taken from the recently publicised 15th point rounding error found in Pentium(processors. In the whole most users of the processor will not be interested in values of that order, so as far as they are concerned, the processor meets their correctness quality criteria, however a laboratory assistant performing minute calculations for an experiment this level or error may mean that the processor does not have the required quality of correctness.

Safety Requirements

Safety requirements cover not only human safety, but also equipment and data safety. Human safety considerations include protecting the operator from moving parts, electrical circuitry and other physical dangers. There may be special operating procedures, which if ignored my lead to a hazardous or dangerous condition occurring. Equipment safety includes safeguarding the software system from unauthorised access either electronically or physically. An example of a safety requirement may be that a monitor used in the system will conform to certain screen emission standards or that the system will be installed in a Faraday Cage with a combination door lock.

Reliability Requirements

Reliability requirements are those which the software must meet in order to perform a specific function under certain stated conditions, for a given period of time. The level of reliability requirement can be dependant on the type of system, i.e. the more critical or life threatening the system, the higher the level of reliability required. Reliability can be measured

in a number of ways including number of bugs per x lines of code, mean time to failure and as a percentage of the time the system will be operational before crashing or an error occurring. Davis states however that the mean time to failure and percent reliability should not be an issue as if the software is fully tested, the error will either show itself during the initial period of use, if the system is asked to perform a function it was not designed to do or the hardware/software configuration of the software host has been changed [Davis '90]. Davis suggests the following hierarchy when considering the detail of reliability in a software requirement specification [Davis '90]:

- Destroy all humankind
- Destroy large numbers of human beings
- Kill a few people
- Injure people
- Cause major financial loss
- Cause major embarrassment
- Cause minor financial loss
- Cause mild inconvenience

Maintainability Requirements

Maintainability requirements look at the long term lift of the proposed system. Requirements should take into consideration any expected changes in the software system, any changes of the computer hardware configuration and special consideration should be given to software operating at sites where software support is not available. Davis suggests defining or setting a minimum standard for requirements which will aid maintainability i.e. [Davis '90]:

- Naming conventions
- Component headers
- In-line document style
- Control constructs
- Use of global/common variables

Characteristics of a Good Software Requirements Specification

A software requirements specification should be clear, concise, consistent and unambiguous. It must correctly specify all of the software requirements, but no more. However the software requirement specification should not describe any of the design or verification aspects, except where constrained by any of the stakeholders requirements.

Complete

For a software requirements specification to be complete, it must have the following properties:

- Description of all major requirements relating to functionality, performance, design constraints and external interfaces.
- Definition of the response of the software system to all reasonable situations.
- Conformity to any software standards, detailing any sections which are not appropriate
- Have full labelling and references of all tables and references, definitions of all terms and units of measure.
- Be fully defined, if there are sections in the software requirements specification still to be defined, the software requirements specification is not complete

Consistent

A software requirement specification is consistent if none of the requirements conflict. There are a number of different types of confliction:

- Multiple descriptors - This is where two or more words are used to reference the same item, i.e. where the term cue and prompt are used interchangeably.
- Opposing physical requirements - This is where the description of real world objects clash, e.g. one requirement states that the warning indicator is orange, and another states that the indicator is red.
- Opposing functional requirements - This is where functional characteristics conflict, e.g. perform function X after both A and B has occurred, or perform function X after A or B has occurred.

Traceable

A software requirement specification is traceable if both the origins and the references of the requirements are available. Traceability of the origin or a requirement can help understand who asked for the requirement and also what modifications have been made to the requirement to bring the requirement to its current state. Traceability of references are used to aid the modification of future documents by stating where a requirement has been referenced. By having foreword traceability, consistency can be more easily contained.

Unambiguous

As the Oxford English dictionary states the word unambiguous means [Hawkins '88]: "not having two or more possible meanings". This means that each requirement can have one and only one interpretation. If it is unavoidable to use an ambiguous term in the requirements specification, then there should be clarification text describing the context of the term. One way of removing ambiguity is to use a formal requirements specification language. The advantage to using a formal language is the relative ease of detecting errors by using lexical syntactic analysers to detect ambiguity. The disadvantage of using a formal requirements specification language is the learning time and loss of understanding of the system by the client.

Verifiable

A software requirement specification is verifiable if all of the requirements contained within the specification are verifiable. A requirement is verifiable if there exists a finite cost-effective method by which a person or machine can check that the software product meets the requirement. Non-verifiable requirements include "The system should have a good user interface" or "the software must work well under most conditions" because the performance words of good, well and most are subjective and open to interpretation. If a method cannot be devised to determine whether the software meets a requirement, then the requirement should be removed or revised.

How Requirements are Specified

When detailing how requirements are specified, two factors need to be taken into consideration. Firstly the notation used to describe the requirements and secondly the way in which the notation is to be presented to the reader of the requirements specification document.

Notations

There are problems associated with specifying the requirements for a system, in any representation that is not natural language. This is due to the problem of the client not being able to understand the contents of the document, and therefore would not being able to agree that the requirement specification correctly describes their needs. With this in mind the requirements engineer must know the limitations of the clients knowledge and their ability to understand any modelling technique that is used to describe the software system. Advantages to using modelling techniques as opposed to natural language include a higher level of specification validation which is not available for natural language, a large reduction in the likelihood of ambiguity which gives the specification higher integrity and in most modelling techniques consistency checking is simplified and even automated when the modelling technique is supported by a software tool. Examples of modelling techniques used in software requirements specifications include Z schema's, data flow diagrams, entity relationship diagrams, state transition diagrams, flow charts and story boards.

Representation of Requirements

When diagrammatic techniques such as data-flow diagrams are used to model requirements, they inherently have a predefined method of presentation. With a data-flow diagram you would expect to see a context diagram, followed by the level zero data-flow diagram. Following the level 0 data-flow diagram you would expect to see the level one diagrams, followed by the level two diagrams, etc. Not all representations have this implied order, for example Z schema's are autonomous specifications, so a suitable method of linking the specifications is required to add cohesion.

Software requirement specification standards

In the following sections, two widely used software requirement specifications are presented, one is a military standard, the other is a recommended practice put together by commercial organisations

Department of Defence - DI-MCCR-80025A

The United States Department of Defence has defined a group of software development standards called DOD-STD-2167A [DoD '88]. All software systems which the Department of Defence procure must be written to this standard. Within this standard are a number of data item descriptions which contain a standard outline and instructions for creating deliverable documentation. One of the data item descriptions is DI-MCCR-80025A which corresponds to the software requirement specification. Each deliverable system is called a computer software configuration item (CSCI).The specification has a rigid format for its contents, as detailed below [DoD '88]:

1 Scope
1.1 Identification
1.2 CSCI overview
1.3 Document overview

2 Applicable documents
2.1 Government documents
2.2 Non-Government documents
3 Engineering requirements
3.1 CSCI external interface requirements
3.2 CSCI capability requirements
3.2.X Capability X
3.3 CSCI internal interfaces
3.4 CSCI data element requirements
3.5 Adaption requirements
3.5.1 Installation dependant data
3.5.2 Operational parameters
3.6 Sizing and timing requirements
3.7 Safety requirements
3.8 Security requirements
3.9 Design constraints
3.10 Software quality factors
3.11 Human performance/human engineering requirements
3.11.1 Human information processing
3.11.2 Foreseeable human errors
3.11.3 Total system implications (e.g. training, support, operational environment)
3.12 Requirements traceability
4 Qualification requirements
4.1 Methods
4.2 Special
5 Preparation for delivery
6 Notes
Appendices

The specification emphasises the points that are most important to the Department of Defence. These include safety and security requirements, human interaction and the consequence of possible human errors, interfaces to external systems and software quality (qualification requirements).

Institute of Electrical and Electronic Engineers - Std. 830 - 1993

IEEE 830 - Recommended Practice for Software Requirements Specifications

This American standard gives an outline to the contents and a detailed description of the contents of a good software requirements specification. The specification, unlike the military standard gives alternative layouts for the requirement specification section (section 3) of the software requirement specification document as well as the overall contents of the other sections.

Outline of the Software Requirement Specification [IEEE '93]:

1 Introduction
1.1 Purpose of the Software Requirement Specification
1.2 Scope of the Product
1.3 Definitions, Acronyms and Abbreviations
1.4 References
1.5 Overview of the Rest of the Software Requirement Specification
2 General Description
2.1 Product Perspective
2.2 Product Functions
2.3 User Characteristics
2.4 General Constraints
2.5 Assumptions and Dependencies
3 Specific Requirements
Appendices
Index.

There are a number of different layouts provided for the specific requirements section of the software requirements specification. The templates provided cover organised by mode, organised by class, organised by object, organised by feature, organised by stimulus, organised by functional hierarchy and showing multiple organisations [IEEE '93].

Comment on Software Requirement Specifications

The IEEE standard is a very flexible document, allowing the user to select the most appropriate format for the type of

system they are trying to specify. Because of the possibly infinite number of possible cases that the specification may be used in this is an appropriate approach. Having a rigid format, like the Department of Defence standard would dissuade companies from adopting the standard because it may be an inappropriate format for some types of software development. Large organisations, like the American Department of Defence have the might and will to create their own format of software requirement specification. The Department of Defence have one thing in mind when developing software systems - safety criticality and possibility of human error. In either case, if wrong could have dire consequences.

Specifying Multimedia Requirements

When describing multimedia systems, there is more emphasis on the data than on the functionality of the application. Data can be in the form of audio information such as voice or music, as well as visual data, such as animation and video. This changes, quite dramatically how the requirements are entered into the specification as well as the emphasis in the specification, as it is no longer on functions, but data and data interconnections. Currently there is little information about how multimedia systems should be specified and as stated by Jones, "further work in this area is badly needed" [Jones and Britton '95].

Return to Chapter Index
Return to Project Index

Issue: December 1996
E-Mail: spa@city.ac.uk