Table of Contents

1. Introduction

The following subsections of the Software Requirements Specifications (SRS) document should provide an overview of the entire SRS. The thing to keep in mind as you write this document is that you are telling what the system must do – so that designers can ultimately build it. Do not use this document for design!!!

1.1 Purpose

Identify the purpose of this SRS and its intended audience. In this subsection, describe the purpose of the particular SRS and specify the intended audience for the SRS.

1.2 Scope

In this subsection:

(1) Identify the software product(s) to be produced by name
(2) Explain what the software product(s) will, and, if necessary, will not do
(3) Describe the application of the software being specified, including relevant benefits, objectives, and goals
(4) Be consistent with similar statements in higher-level specifications if they exist

This should be an executive-level summary. Do not enumerate the whole requirements list here.

1.3 Definitions, Acronyms, and Abbreviations.

Provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendices in the SRS or by reference to documents. This information may be provided by reference to an Appendix.

1.4 References

In this subsection:

(1) Provide a complete list of all documents referenced elsewhere in the SRS
(2) Identify each document by title, report number (if applicable), date, and publishing organization
(3) Specify the sources from which the references can be obtained.

This information can be provided by reference to an appendix or to another document. If your application uses specific protocols or RFC's, then reference them here so designers know where to find them.

1.5 Overview

In this subsection:

(1) Describe what the rest of the SRS contains
(2) Explain how the SRS is organized

Don't rehash the table of contents here. Point people to the parts of the document they are most concerned with. Customers/potential users care about section 2, developers care about section 3.

2. The Overall Description

Describe the general factors that affect the product and its requirements. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in section 3, and makes them easier to understand. In a sense, this section tells the requirements in plain English for the consumption of the customer. Section3 will contain a specification written for

the developers.

2.1 Product Perspective

Put the product into perspective with other related products. If the product is independent and totally self-contained, it should be so stated here. If the SRS defines a product that is a component of a larger system, as frequently occurs, then this subsection relates the requirements of the larger system to functionality of the software and identifies interfaces between that system and the software. If you are building a real system,compare its similarity and differences to other systems in the marketplace. If you are doing a research-oriented project, what related research compares to the system you are planning to build.

A block diagram showing the major components of the larger system, interconnections, and external interfaces can be helpful. This is not a design or architecture picture. It is more to provide context, especially if your system will interact with external actors. The system you are building should be shown as a black box. Let the design document present the internals.

The following subsections describe how the software operates inside various constraints.

2.1.1 System Interfaces (if any)

List each system interface and identify the functionality of the software to accomplish the system requirement and the interface description to match the system. These are external systems that you have to interact with. For instance, if you are building a business application that interfaces with the existing employee payroll system, what is the API to that system that designer's will need to use?

2.1.2 Hardware Interfaces

Specify the logical characteristics of each interface between the software product and the hardware components of the system. This includes configuration characteristics. It also covers such matters as what devices are to be supported, how they are to be supported and protocols. This is not a description of hardware requirements in the sense that "This program must run on a Mac with 64M of RAM". This section is for detailing the actual hardware devices your application will interact with and control. For instance, if you are controlling X10 type home devices, what is the interface to those devices? Designers should be able to look at this and know what hardware they need to worry about in the design. Many business type applications will have no hardware interfaces. If none, just state "The system has no hardware interface requirements" If you just delete sections that are not applicable, then readers do not know if: a. this does not apply or b. you forgot to include the section in the first place.

2.1.3 Software Interfaces

Specify the use of other required software products and interfaces with other application systems. For each required software product, include:

(1) Name
(2) Mnemonic

(3) Specification number
(4) Version number
(5) Source

For each interface, provide:
(1) Discussion of the purpose of the interfacing software as related to this software product
(2) Definition of the interface in terms of message content and format

Here we document the APIs, versions of software that we do not have to write, but that our system has to use. For instance if your customer uses SQL Server 7 and you are required to use that, then you need to specify i.e.

2.1.4.1 Microsoft SQL Server 7. The system must use SQL Server as its database component. Communication with the DB is through ODBC connections. The system must provide SQL data table definintions to be provided to the company DBA for setup.

A key point to remember is that you do NOT want to specify software here that you think would be good to use. This is only for **customer-specified systems** that you **have** to interact with. Choosing SQL Server 7 as a DB without a customer requirement is a Design choice, not a requirement. This is a subtle but important point to writing good requirements and not over-constraining the design.

2.1.4 Communications Interfaces (if any)

Specify the various interfaces to communications such as local network protocols, etc. These are protocols you will need to directly interact with. If you happen to use web services transparently to your application then do not list it here. If you are using a custom protocol to communicate between systems, then document that protocol here so designers know what to design. If it is a standard protocol, you can reference an existing document or RFC.

2.2 Product Functions

Provide a summary of the major functions that the software will perform. Sometimes the function summary that is necessary for this part can be taken directly from the section of the higher-level specification (if one exists) that allocates particular functions to the software product.

For clarity:
(1) The functions should be organized in a way that makes the list of functions understandable to the customer or to anyone else reading the document for the first time.
(2) Textual or graphic methods can be used to show the different functions and their relationships. Such a diagram is not intended to show a design of a product but simply shows the logical relationships among variables.

AH, Finally the real meat of section 2. This describes the functionality of the system in the language of the customer. What specifically does the system that will be designed have to do? Drawings are good, but remember this is a description of what the system needs to do, not how you are going to build it. (That comes in the design document).

2.3 User Characteristics

Describe those general characteristics of the intended users of the product including educational level, experience, and technical expertise. Do not state specific requirements but rather provide the reasons why certain specific requirements are later specified in section 3.

What is it about your potential user base that will impact the design? Their experience and comfort with technology will drive UI design. Other characteristics might actually influence internal design of the system.

2.4 Constraints

Provide a general description of any other items that will limit the developer's options. These can include:

(1) Regulatory policies
(2) Hardware limitations (for example, signal timing requirements)
(3) Interface to other applications
(4) Parallel operation
(5) Audit functions
(6) Control functions
(7) Higher-order language requirements
(8) Signal handshake protocols (for example, XON-XOFF, ACK-NACK)
(9) Reliability requirements
(10) Criticality of the application
(11) Safety and security considerations

This section captures non-functional requirements in the customers language. A more formal presentation of these will occur in section 3.

2.5 Assumptions and Dependencies

List each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption might be that a specific operating system would be available on the hardware designated for the software product. If, in fact, the operating system were not available, the SRS would then have to change accordingly.
This section is catch-all for everything else that might influence the design of the system and that did not fit in any of the categories above.

3. Specific Requirements

This section contains all the software requirements at a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system and all functions performed by the system in response to an input or in support of an output. The following principles apply:

(1) Specific requirements should be stated with all the characteristics of a good SRS
- correct
- unambiguous
- complete
- consistent
- ranked for importance and/or stability
- verifiable
- modifiable
- traceable

(2) Specific requirements should be cross-referenced to earlier documents that relate

(3) All requirements should be uniquely identifiable (usually via numbering like 3.1.2.3)

(4) Careful attention should be given to organizing the requirements to maximize readability (Several alternative organizations are given at end of document)

Before examining specific ways of organizing the requirements it is helpful to understand the various items that comprise requirements as described in the following subclasses. This section reiterates section 2, but is for developers not the customer. The customer buys in with section 2, the designers use section 3 to design and build the actual application.

Remember this is not design. Do not require specific software packages, etc unless the customer specifically requires them. Avoid over-constraining your design. Use proper terminology:

The system shall… A required, must have feature

The system should… A desired feature, but may be deferred til later

The system may… An optional, nice-to-have feature that may never make it to implementation.

Each requirement should be uniquely identified for traceability. Usually, they are numbered 3.1, 3.1.1, 3.1.2.1 etc. Each requirement should also be testable. Avoid imprecise statements like, “The system shall be easy to use” Well no kidding, what does that mean? Avoid “motherhood and apple pie” type statements, “The system shall be developed using good software engineering practice”

Avoid examples, This is a specification, a designer should be able to read this spec and build the system without bothering the customer again. Don’t say things like, “The system shall accept configuration information such as name and address.” The designer doesn’t know if that is the only two data elements or if there are 200. List every piece of information that is required so the designers can build the right UI and data tables.

3.1 Performance Requirements

This subsection specifies both the static and the dynamic numerical requirements placed on the software or on human interaction with the software, as a whole. Static numerical requirements may include:

(a) The number of terminals to be supported

(b) The number of simultaneous users to be supported
(c) Amount and type of information to be handled

Static numerical requirements are sometimes identified under a separate section entitled capacity.

Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.

All of these requirements should be stated in measurable terms.

For example,

95% of the transactions shall be processed in less than 1 second rather than, An operator shall not have to wait for the transaction to complete.

(Note: Numerical limits applied to one specific function are normally specified as part of the processing subparagraph description of that function.)

3.2 Logical Database Requirements

This section specifies the logical requirements for any information that is to be placed into a database. This may include:

- Types of information used by various functions
- Frequency of use
- Accessing capabilities
- Data entities and their relationships
- Integrity constraints
- Data retention requirements

If the customer provided you with data models, those can be presented here. ER diagrams (or static class diagrams) can be useful here to show complex data relationships. Remember a diagram is worth a thousand words of confusing text.

3.3 Design Constraints

Specify design constraints that can be imposed by other standards, hardware limitations, etc.

4. Appendix