Software Requirements Specification (SRS)

Software Requirements Specification

for

<Name of Project>

<authors>

<date>

Version	Release Date	Responsible Party	Major Changes
0.1			Initial Document Release for Comment

Table of Contents

Built the table of contents here. Insert it when you finish your document.

1. Introduction

The following subsections of the SRS should provide an overview of the entire SRS.

1.1 Purpose

Identify the purpose of this SRS and its intended audience.

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. As a portion of this, it should:

(a) Describe the relevant benefits, objectives, and goals as precisely as possible

(b) Be consistent with similar statements in higher-level specifications if they exist.

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 an appendix or other document(s).

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.

1.5 Overview

Describe the rest of the SRS and how it is organized.

2. The General Description

Describe the general factors that affect the product and its requirements. This section usually consists of the five subsections that follow. This section does not state specific requirements; each of its subsections makes those requirements easier to understand; they do not specify design or express specific requirements. Such detail is provided in section 3.

2.1 Product Perspective

This subsection of the SRS relates the product to other products or projects.

(1) If the product is independent and totally self-contained, it should be stated here.

(2) If the SRS defines a product that is a component of a larger system or project:

(a) Describe the functions of each component of the larger system or project, and identify interfaces

(b) Identify the principal external interfaces of this software product (not a detailed description)

(c) Describe the computer hardware and peripheral equipment to be used (overview only)

A block diagram showing the major components of the larger system or project, interconnections, and external interfaces can be very helpful.

2.2 Product Functions

Provide a summary of the 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. 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. Block diagrams showing the different functions and their relationships can be helpful. Such a diagram is not a requirement on the design of a product itself; it is simply an effective explanatory tool.

2.3 User Characteristics

Describe those general characteristics of the eventual users of the product that will affect the specific requirements.

Many people interact with a system during the operation and maintenance phase of the software life cycle. Some of these people are users, operators, and maintenance and systems personnel. Certain characteristics of these people, such as educational level, experience, and technical expertise impose important constraints on the system's operating environment.

2.4 General Constraints

Provide a general description of any other items that will limit the developer's options for designing the system. 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) Criticality of the application

(10) Safety and security considerations

2.5 Assumptions and Dependencies

List and describe each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but any changes to them can affect the requirements in the SRS. For example, an assumption might be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change.

3. Specific Requirements

This section of the SRS should contain all the details the software developer needs to create a design. This is typically the largest and most important part of the SRS.

(1) The details within it should be defined as individual specific requirements, following the guidelines for sound requirements (verifiable, unambiguous, etc.)

(2) Specific requirements should be organized in a logical and readable fashion.

(3) Each requirement should be stated such that its achievement can be objectively verified by a prescribed method.

(4) Sources of a requirement should be identified where that is useful in understanding the requirement.

(5) One way to classify the specific requirements is as follows:

(a) Functional Requirements

(b) Performance Requirements

(c) Design Constraints

(d) Attributes

(e) External Interface Requirements

The organization of this section of the SRS should be chosen with the goal of properly specifying the requirements in the most readable manner.

3.1. Functional Requirements

This subsection of the SRS should specify what is to be done by the product, to what level or specific requirement, what inputs should be transformed to what outputs (not **how** this is done), what specific operations are required. Where the rationale for a requirement is not obvious, provide an explanation. Where issues need to be resolved, cite those.

For each function, specify requirements on inputs, processing, and outputs. These are usually organized with these four subparagraphs:

(1) Purpose of the function: Provide rationale to clarify the intent of the function.

(2) Inputs: sources, valid ranges of values, any timing concerns, operator requirements, special interfaces

(3) Operations to be performed: validity checks, responses to abnormal conditions, types of processing required

(4) Outputs: destinations, valid ranges of values, timing concerns, handling of illegal values, error messages, interfaces required

3.2. External Interface Requirements

This should specify:

(1) The characteristics that the software must support for each human interface to the software product. For example, if the user of the system operates through a display terminal, the following should be specified:

(a) Required screen formats

(b) Page layout and content of any reports or menus

(c) Relative timing of inputs and outputs

(d) Availability of some form of programmable function keys.

(2) All the aspects of optimizing the interface with the person who must use the system. This may simply comprise a list of do's and don'ts on how the system will appear to the user

Specify the logical characteristics of each interface between the software product and the hardware components of the system. Include such matters as what devices are to be supported, how they are to be supported, and protocols.

Specify the use of other required software products (for example, a data management system, an operating system, or a mathematical package), and interfaces with other application systems .

For each required software product, the following should be provided:

(1) Name

(2) Mnemonic

(3) Specification Number

(4) Version number

(5) Source

For each interface:

(1) Discuss the purpose of the interfacing software as related to this software product.

(2) Define the interface in terms of message content and format. It is not necessary to detail any well-documented interface, but a reference to the document defining the interface is required.

Specify the various interfaces to communications such as local network protocols, etc.

3.3. Performance Requirements

This subsection should specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software, as a whole.

(1) Static numerical requirements may include:

(a) The number of terminals to be supported

(b) The number of simultaneous users to be supported

(c) Number of files and records to be handled

(d) Sizes of tables and files

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

(2) 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 s, rather than, 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.

3.4. Design Constraints

Design constraints can be imposed by other standards, hardware limitations, etc.

Specify the requirements derived from existing standards or regulations. They might include:

(1) Report format

(2) Data naming

(3) Accounting procedures

(4) Audit Tracing. For example, this could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum government or financial standards. An audit trace requirement might, for example, state that all changes to a payroll data base must be recorded in a trace file with before and after values.

Identify the requirements for the software to operate inside various hardware constraints

3.5. Quality Characteristics

There are a number of quality characteristics that can apply to software. Pick the ones most important to this product and develop a section for each one. Definitions of the quality characteristics follow.

- Correctness - extent to which program satisfies specifications, fulfills user's mission objectives
- Efficiency - amount of computing resources and code required to perform function
- Flexibility - effort needed to modify operational program
- Integrity/security - extent to which access to software or data by unauthorized people can be controlled
- Interoperability - effort needed to couple one system with another
- Maintainability - effort required to locate and fix an error during operation
- Portability - effort needed to transfer from one h/w or s/w environment to another
- Reliability - extent to which program performs with required precision
- Reusability - extent to which it can be reused in another application
- Testability - effort needed to test to ensure performs as intended
- Usability - effort required to learn, operate, prepare input, interpret output

Describe the rationale for including this characteristic for this product.

Describe how the presence, absence, or level of this characteristic will be measured; identify ways to test the characteristic once the product is complete.

3.6. Other Requirements

Certain requirements may, due to the nature of the software, the user organization, etc., be placed in separate categories such as those below.

This could specify the requirements for any data base that is to be developed as part of the product. This might include:

(1) Types of information

(2) Frequency of use

(3) Accessing capabilities

(4) Data element and file descriptions

(5) Relationship of data elements, records and files

(6) Static and dynamic organization

(7) Retention requirements for data

Note: If an existing data base package is to be used, this package should be named under Interfaces to Software and details of using it specified there.

This could specify the normal and special operations required by the user such as:

(1) The various modes of operations in the user organization; for example, user-initiated operations

(2) Periods of interactive operations and periods of unattended operations

(3) Data processing support functions

(4) Backup and recovery operations

Note: This is sometimes specified as part of the User Interfaces section.

This could:

(1) Define the requirements for any data or initialization sequences that are specific to a given site, mission, or operational mode, for example, safety limits.

(2) Specify features that should be modified to adapt the software to an installation.

4. Supporting Information

The supporting information; that is, the Table of Contents, the Appendices, and the Index, make the SRS easier to use. The Appendices are not always considered part of the actual requirements specification and are not always necessary. They might include:

(a) Sample I/O formats, descriptions of cost analysis studies, results of user surveys.

(b) Supporting or background information that can help the readers of the SRS.

(c) A description of the problems to be solved by the software.

(d) The history, background, experience and operational characteristics of the organization to be supported.

(e) A cross-reference list, arranged by milestone, of those incomplete software requirements that are to be completed by specified milestones.

(f) Special packaging instructions for the code and the media to meet security, export, initial loading, or other requirements.

(3) When Appendices are included, the SRS should explicitly state whether or not the Appendices are to be considered part of the requirements.