# Software Requirements Specification – Outline

## Introduction

The introduction serves to orient the reader. It describes both the system and the SRS itself.

### *Purpose*

This section describes the purpose of the document. Typically, this will contain a brief two- or three-sentence description, including the name of the project. For example: "The purpose of this document is to serve as a guide to designers, developers and testers who are responsible for the engineering of the *(name of project)* project. It should give the engineers all of the information necessary to design, develop and test the software." This is to ensure that the person reading the document understands what he or she is looking at.

### *Scope*

This section contains a brief description of the scope of the document. If the SRS is a complete description of the software, then it will state something similar to: "This document contains a complete description of the functionality of the *(name of project)* project. It consists of use cases, functional requirements and nonfunctional requirements, which, taken together form a complete description of the software." For complex software, the requirements for the project might be divided into several SRS documents. In this case, the scope should indicate which portion of the project is covered in this document.

### *System Overview*

This section contains a description of the system. This is essentially a brief summary of the vision and scope of the project.

### *References*

Any references to other documents (including the vision and scope document) should be included here. These may include other documents in the organization, work products, articles, and anything else that is relevant to understanding the SRS. If there is an organizational intranet, this section often includes URLs of referenced documents.

## Definitions

The definitions section contains any definitions needed to understand the SRS. Often it will contain a glossary, defining terms which the reader may not be familiar with (or which may have a specific meaning here that differs from everyday use). This section may also contain definitions of any data files that are used as input, a list of any databases which may be needed, and any other organizational or workflow-related information that is needed to understand the SRS.

## Use Cases

This section contains a set of use cases that describe the external behavior of the software. A use case is a description of a specific interaction that a user may have with the software. Use cases are deceptively simple tools for describing the functionality of the software.

A use case is a simple, straightforward tool that can be used to completely describe all of the behavior of a piece of software. It contains a textual description of all of the ways that the intended users could work with the software through its interface. Use cases do not describe any internal workings of the software, nor do they explain how that software will be implemented. They simply show how the steps that the user follows to use the software to do his work. All of the ways that the users interact with the software can be described in this manner.

A typical use case includes these sections, usually laid out in a table:

| **Name** | **Use case number and name** |
|---|---|
| Summary | Brief description of the use case |
| Rationale | Description of the reason that the use case is needed |
| Users | A list of all of the categories of users which interact with this use case |
| Preconditions | The state of the software before the use case begins |
| Basic Course of Events | A numbered list of interactions between the user and one or more users |
| Alternative Paths | Conditions under which the basic course of events could change |
| Postconditions | The state of the software after the basic course of events is complete |

This section will contain multiple use cases, enough to define all of the interactions the users will have with the software. The following sample use case describes a simple search-and-replace function in a word processor:

| **Name** | **UC-8: Search** |
|---|---|
| Summary | All occurrences of a search term are replaced with replacement text. |
| Rationale | While editing a document, many users find that there is text somewhere in the file being edited that needs to be replaced, but searching for it manually by looking through the entire document is time-consuming and ineffective. The search-and-replace function allows the user to find it automatically and replace it with specified text. Sometimes this term is repeated in many places and needs to be replaced. Other times, only the first occurrence should be replaced. The user may also wish to simply find the location of that text without replacing it. |
| Users | All users |
| Preconditions | A document is loaded and being edited. |
| Basic Course of Events | 1. The user indicates that the software is to perform a search-and-replace in the document.<br>2. The software responds by requesting the search term and the replacement text.<br>3. The user inputs the search term and replacement text and indicates that all occurrences are to be replaced.<br>4. The software replaces all occurrences of the search term with the replacement text. |
| Alternative Paths | 1. In step 3, the user indicates that only the first occurrence is to be replaced. In this case, the software finds the first occurrence of the search term in the document being edited and replaces it with the replacement text. The postcondition state is identical, except only the first occurrence is replaced, and the replacement text is highlighted.<br>2. In step 3, the user indicates that the software is only to search and not replace, and does not specify replacement text. In this case, the software highlights the first occurrence of the search term and the use case ends.<br>3. The user may decide to abort the search-and-replace operation at any time during steps 1, 2 or 3. In this case, the software returns to |

| Name | UC-8: Search |
|---|---|
| | the precondition state. |
| Postconditions | All occurrences of the search term have been replaced with the replacement text. |

# Functional Requirements

Functional requirements define the internal workings of the software: that is, the calculations, technical details, data manipulation and processing, and other specific functionality that shows how the use cases are to be satisfied.

The name, summary and rationale of each functional requirement are used in the same way as those of the use cases. The behavior that is to be implemented should be described in plain English in the "Requirements" section. Most requirements are only relevant to a small number of use cases—these should be listed by name and number in the "References" section. (Some requirements are not associated with use cases.)

The core of the requirement is the description of the required behavior. It is very important to make this clear and readable. This behavior may come from organizational or business rules, or it may be discovered through elicitation sessions with users, stakeholders, and other experts within the organization. Many requirements will be uncovered during the use case development. When this happens, the requirements analyst should create a placeholder requirement with a name and summary, and research the details later, to be filled in when they are better known.

The following table shows a template for a functional requirement:

| Name | Name and number of the functional requirement |
|---|---|
| Summary | Brief description of the requirement |
| Rationale | Description of the reason that the requirement is needed |
| Requirements | The behavior that is required of the software |
| References | Use cases and other functional and nonfunctional requirements which are relevant to understanding this one. |

This section will contain multiple functional requirements, enough to define the complete behavior of the software. The following table shows an example of a requirement that might be discovered during the development of the search-and-replace use case (above):

| Name | FR-4: Case sensitivity in search-and-replace |
|---|---|
| Summary | The search-and-replace feature must have case sensitivity in both the search and the replacement. |
| Rationale | A user will often search for a word that is part of a sentence, title, heading or other kind of text that is not all-lowercase. The search-and-replace function needs to be aware of that, and give the user the option to ignore it. |
| Requirements | When a user invokes the search-and-replace function, the software must give the option to do a case-sensitive search.<br>By default, the search will match any text which has the same letters as the search term, even if the case is different. If the user indicates that the search is to be done with case-sensitivity turned on, then the software will only match text in the document where the case is identical to that of the search term.<br>During a search and replace, when the software replaces original text in the document with the replacement text specified by the user, the software retains the case of the original text as follows:<br>• If the original text was all uppercase, then the replacement text must be inserted in all uppercase. |

| Name | FR-4: Case sensitivity in search-and-replace |
|---|---|
| | • If the original text was all lowercase, then the replacement text must be inserted in all lowercase.<br>• If the original text had the first character uppercase and the rest of the characters lowercase, then the replacement text must reflect this case as well.<br>• If the original text was sentence case (where the first letter of each word is uppercase), then the replacement text must be inserted in sentence case.<br>• In all other cases, the replacement text should be inserted using the case that was specified by the user. |
| References | UC-8: Search |

# Nonfunctional Requirements

Nonfunctional requirements impose constraints on the design or implementation (such as performance requirements, quality standards or design constraints).

Users have implicit expectations about how well the software will work. These characteristics include how easy the software is to use, how quickly it executes, how reliable it is, and how well it behaves when unexpected conditions arise. The nonfunctional requirements define these aspects about the system. (The nonfunctional requirements are sometimes referred to as “non-behavioral requirements” or “software quality attributes”.)

The nonfunctional requirements should be defined as precisely as possible. Often, this is done by quantifying them. Where possible, the nonfunctional requirements should provide specific measurements which the software must meet. The maximum number of seconds it must take to perform a task, the maximum size of a database on disk, the number of hours per day a system must be available, and the number of concurrent users supported are examples of requirements that the software must implement but do not change its behavior.

This section will contain multiple nonfunctional requirements, enough to define all of the performance and quality attributes of the software. Nonfunctional requirements can use the same template as functional requirements (above). The following table shows an example of a nonfunctional requirement:

| Name | NF-7: Performance constraints for search-and-replace |
|---|---|
| Summary | The search-and-replace feature must perform a search quickly |
| Rationale | If a search is not fast enough, users will avoid using the software. |
| Requirements | A case-insensitive search-and-replace performed on a 3MB document with twenty 30-character search terms to be replaced with a different 30-character search term must take under 500ms on a 700mhz Pentium III running Microsoft Windows 2000 at 50% CPU load. |
| References | UC-8: Search |