

Lifecycle

Functional Specification Standard

By [Dino Fancellu](#)
May 20, 2002

URL of this document:

<http://www.softwarereality.com/lifecycle/FunctionalSpecStandard.jsp>

Introduction

In general terms, the functional specification states what the proposed system is to do, whereas design is how the system is to be constructed to meet the functional specification. However in writing it, some consideration of design issues must take place, to ensure a realistic system is specified.

The functional specification should be clear, consistent, precise and unambiguous. The user requirement may mean that the user interface should be included in this document for some projects, whereas for others this will be done at the design stage either within a document or developed via a prototype.

It is important that there is a draft functional specification before the design stage on any project is started and that the functional specification is agreed and issued normally within a week of the final quality review. There must be a milestone on the project plan for the issue of the functional specification. The functional specification must be kept up to date as this is the communication with the world outside the development staff.

The following should be used as a standard for a functional specification with some mandatory sections. The layout itself is at the discretion of the author except for Chapter 1. The document should have a standard front page, document authorisation page containing the title, issue, author and quality controller and

contents page. Use diagrams where appropriate.

Do not be afraid of examples! Use them copiously throughout, as a brief, concrete example often illustrates a point much more succinctly than a normative explanation. Also remember to keep the examples interesting, as this is a useful way of keeping the reader's interest - this is just as important in a functional specification as in any other type of document.

1. Introduction

An introductory sentence or two about the project as this is probably the first document written on the project.

1.1. Summary

A few sentences summarising the project: what it is, who it is for (customer or internal), is it a bespoke project, a product, a demo.

1.2. Requirements

This section should state the requirements the functional specification is attempting to fulfil. This may be an understanding of a customer's requirement or a statement given as an internal starting point, e.g. produce a comprehensive mail tool in minimum time. Normally requirements are by their nature unstructured with high and low level statements intermingled. This section should refer to a separate requirements document if it exists. If there is anything else clarifying the requirement such as faxes these should also be referred to and probably a copy put into an Appendix.

1.3. Numbers

This section should detail the number of users expected to use the system, how often, expected number of transactions (per minute/hour/day), peak usage times etc.

The question that should be asked of project stakeholders up-front is: "What numbers are we looking at?"

Capacity/response time needs have to be outlined so that we don't come up with a slow/tiny system, or don't totally over-do it and come up with a n-tier EJB solution

costing £500k, when the system will only ever have 20 simultaneous users.

Such information will make a big difference to the architecture, i.e. the eventual design specification. This is why it is vital to establish these figures early in the project.

These figures are such an overarching issue that they do not belong in any one section. In fact the issue is expanded upon in several sections, such as User Community, Performance and Expandability.

1.4. Existing System

This section should include an explanation of the system we are replacing, even if it's an old manual system.

What problems does the current system have? Which of these problems do we solve?

What useful functions of the current system will we not provide (Constraints)?

Depending on the depth of analysis required, this section may also describe the root causes of each problem. "Root cause" analysis is a systematic way of uncovering the underlying cause of an identified problem:

"It's amazing how much people do know about the problem behind the problem; it's just that no-one - by which we usually mean management - had taken the time to ask them before. So, ask them and then ask them again."

Source: *Managing Software Requirements: A Unified Approach* by Dean Leffingwell, Don Widrig - Chapter 4, "The Five Steps in Problem Analysis"

1.5. Terminology

This section should contain all words or phrases having a special meaning for this project with a clear, concise, unambiguous statement on their meaning.

1.6. References

List any document references with numbers, remembering to include issue numbers and/or dates so that the actual version is identified and refer to them as ref[n] in the rest of the document.

2. Functional Description

The rest of the document may be divided into individual sections or chapters depending on the size and complexity of the system. Avoid forward references as the flow of the document is lost; consider re-ordering of the document in such circumstances.

Whereas requirements tend to be unstructured, the functions provided to fulfil the requirements must be structured. All statements as to functionality, should be written clearly using consistent terminology such that a test could be written to ensure the final system, performs as described and also that a design should fall naturally with no interpretation being necessary. It should be possible to draw up a table of functions within full system and product tests and incorporate a test for each function. To this end all functional statements should be numbered.

It may be that basic functionality could be identified such that some items are mandatory whereas some are highly desirable which should be clear from the requirements. If this is so, then identify these in this specification.

The functions should be grouped where possible under sub-headings to make an easily readable and understandable system.

All the following headings must be included somewhere in the document, not necessarily in the order given here. If it is not relevant or we are not addressing it for this system, then say so.

Use Cases

Most likely these will be kept in a separate document or CASE tool, referenced from the functional specification. Development of the use cases and functional specification should happen in parallel, where information from one feeds the other incrementally.

Always avoid repetition. The amount of detail in the rest of the functional specification will depend on the number of use cases that have been written.

Although important, use cases do not capture all functional requirements: this is why we need an encompassing functional specification. The availability of a separate document also discourages use case authors from putting too much detail

in the use case (e.g. functional requirements instead of usage scenario text) or the wrong detail (e.g. boundary conditions), which are both common mistakes.

(Note this is a similar approach to the Unified Process "Supplementary Spec" which captures additional detail that should be kept separate from the use case).

Where the functional specification references a use case, always use the unique use case name (e.g. "Perform Order Entry"). Depending on the size of the system being modelled, you might also need to include the package name.

Similarly, if the use case references an item in the functional spec, always use the section and number of the functional item (e.g. "User Community, item 1.2.3.4"). If possible (given the constraints of the word processor or CASE tool being used) provide a hyperlink that takes the reader directly to the referenced item.

User Community

Identification of who the system is aimed at. There may be more than one group of people and each group may have slightly different requirements. Are we providing different functions to fulfil these or not?

These groups of people are normally identified as use case roles (i.e. actors), and the functions assigned to each role as individual use cases. Where this information does not fit into the use case model, it should be captured in the main functional specification instead.

Administration Functions

How will the system be administered? Are there separate functions for an administrator? Is there any security built in to stop others using administrative functions? Passwords?

Error Handling

How errors should be handled should be stated. Identify the different types and reasons for the classification.

Security

Security considerations are an important part of any project. This section should detail possibilities of abuse of the system.

Along with error handling, the specification has to handle "the negative path". There is no point in having a system that does lots of good things if it also does

lots of bad things.

Help

What type of help is to be provided?

Printing

Ensure any printing to be provided is stated.

Interfaces

User:

This could be a chapter in its own right if it is a full definition. If it is deferred to the design specification stage, this should be stated.

Software:

We may be interfacing to existing software. This should be stated, e.g. toolkits, back ends of existing packages. State versions. Do interface documents exist?

Boundary Conditions

It should be clear what are the extremes to be taken into consideration. These items may have come up. This will vary with different systems but it could be items such as number of users, size of forms, number of forms.

Constraints

All other constraints not specified under particular headings. For example design constraints, e.g. it must be a client server architecture.

Platforms

We should list which platforms we will be supporting. Name a reference platform or platforms plus appropriate operating system versions.

Internationalisation

Is this to be included in the product now or in the future?

Performance

Capacity

Response times

Portability

Although we may only be supporting one platform initially, we almost certainly will want to be able to port developments to other platforms. This should be stated

here.

Expandability

State the likely expansion requirements. Some of the items may have been considered earlier in the document. These should be referred to from this section and any additional items put in.

Customisation

Are we allowing the user to customise the system? If so, what are we going to provide?

Support & Maintenance

Are any functions to be included to make maintenance and support easier, e.g. internal monitoring of traffic flows.

Configuration Management

How are we proposing to manage the various software versions?

Documentation

List the documents that will be produced. This could refer to the project plan if that exists and contains such a list, otherwise it should be stated here.