

# Requirements Fundamentals: the basis for effective testing

Suzanne Robertson  
The Atlantic Systems Guild Ltd

[suzanne@systemsguild.com](mailto:suzanne@systemsguild.com)  
<http://www.systemsguild.com/>

## Abstract

We accept that testing the software is an integral part of building a system. However, if the software is based on inaccurate requirements then, despite any amount of testing, the software will be unsatisfactory. Instead of limiting our testing to code, we should start testing as soon as we start work on the requirements for a product. The problem is that requirements are often written in loose and inconsistent ways that make them impossible to test. If we have a consistent and well-defined way of specifying requirements then we can involve people with testing expertise early in the project. This paper describes a quality gateway that defines testpoints that a requirement has to pass in order to enter the requirements specification. These gateway tests are concerned with ensuring that the requirements are accurate, and do not cause problems by being unsuitable for the design and implementation stages later in the project.

The set of requirements tests cover relevance, coherency, traceability, completeness, consistency, connectivity and other qualities that successful requirements must have. The starting point of the tests is that each requirement must have a number of components one of which is a fit criterion. Fit criteria are used to test whether it is possible to classify any given solution into one that satisfies or does not satisfy, the requirement. The aim is to trap requirements-related defects as early as they can be identified and to prevent incorrect requirements from being incorporated in the design and implementation where they will be more difficult and expensive to find and correct.

## Contents of the Requirements Specification

A tester can test anything provided he has a set of criteria against which to test. Given a well-specified definition of the content of a requirements specification, then a tester can establish whether or not a given specification meets the standard for requirements specifications. In other words we can test a specification to establish whether or not it is a specification.

A good place to start is to consider what we expect to find in a requirements specification; here we have a problem because “requirements specification” is one of the most elastic terms in current use. Every specification that I review has variations in the components that it contains and the level of detail of each of those components. Some specifications contain purely functional requirements, others include non-functional or quality requirements, some contain all the project constraints, some specify the stakeholders, some include business goals, some concentrate on the solution – and all these things are at differing levels of detail - the variations are endless. However without some consistent agreement on what you intend your specification to contain it is impossible to test the quality of its contents.

Figure 1 is the table of contents of a requirements specification template [Volere] that we have built as a result of working on many different projects. The template is intended as a guide for the contents of a requirements specification.

## Requirements Specification Table of Contents

### PROJECT DRIVERS

1. The Purpose of the Product
2. Client, Customer and other Stakeholders
3. Users of the Product

### PROJECT CONSTRAINTS

4. Mandated Constraints
5. Naming Conventions and Definitions
6. Relevant Facts and Assumptions

### FUNCTIONAL REQUIREMENTS

7. The Scope of the Work
8. The Scope of the Product
9. Functional and Data Requirements

### NON-FUNCTIONAL REQUIREMENTS

10. Look and Feel Requirements
11. Usability Requirements
12. Performance Requirements
13. Operational Requirements
14. Maintainability and Portability Requirements
15. Security Requirements
16. Cultural and Political Requirements
17. Legal Requirements

### PROJECT ISSUES

18. Open Issues
19. Off-the-Shelf Solutions
20. New Problems
21. Tasks
22. Cutover
23. Risks
24. Costs
25. User Documentation and Training
26. Waiting Room
27. Ideas for Solutions

*Figure 1: The Volere requirements template identifies the components of a requirements specification.*

For instance item 1 is the Purpose of the Product. A requirements specification must contain measurable goals for building the product. Unless there is an unambiguous

understanding of the goals then subjective judgements will lead to an unsatisfactory and in many cases, irrelevant solution. Sections 2 and 3 of the template are concerned with identifying all the people who are stakeholders in the project and agreeing their roles and involvement. This is much more than saying who the eventual end users might be. Here we are talking about all the sources of business and technical information necessary to build the product. Some of the stakeholders might be external organisations, some will be business people, others will be the members of the project team.

Sections 7 and 8 make the distinction between the business problem (the scope of the work we need to understand) and the product we are planning to build (the scope of the product). These two things are very different. The scope of the product identifies the boundaries between the end users and the product. The scope of the work is wider - it deals with the part of the world that we need to investigate in order to identify the most advantageous boundary for the product.

Sections 9-17 contain the detailed specification of individual functional and non-functional requirements.

## Contents of Individual Requirements

A requirement is a testable statement of some function or quality that the product must have. The question is can you give the requirement a fit criterion such that it will be possible to put any solution into one of two sets – solutions that fit the requirement, solutions that do not fit the requirement. The amount of work involved in writing the fit criteria varies tremendously depending on the understanding of the project and the sources of knowledge. We have found that it helps to have a staged approach to identifying a fit criterion by exploring a number of aspects of the requirement. The requirements shell [Robertson] in Figure 2 illustrates the aspects that we are looking for.

The *Requirement #* is a unique identifier for the requirement. The *requirement type* cross-references the requirement back to the list of types on the template. For example if the requirement is type 12 then it is a performance requirement. The *business event* and *product use case* numbers are there to identify all the business events and use cases to which the requirement relates.

The *description* might be very ambiguous, very vague or anything in between depending on who or where it came from. The *rationale* is the first step to dealing with ambiguity because it says why this requirement is considered to be important – what does it do to contribute to the goals of the product? Neither the description nor the rationale are testable, but they help to explore the requirement and lead to the ability to define *the fit criterion*. Once a requirement has a fit criterion then you can test the fit criterion to see whether it contains any ambiguity, whether it uses defined terminology, and whether it is possible to write a test that is capable of testing any eventual solution. Bear in mind that you are not attempting to design the test, instead you are testing the requirement to see whether it actually is a well-defined requirement.

Another aspect of an individual requirement deals with the *customer value* [Pardee] - how satisfied will the customer be if the solution fits the fit criterion versus how dissatisfied will he be if the solution does not fit the fit criterion. This is the first attempt to understand the relative importance of the requirement. *Dependencies*, *Conflicts*, *Supporting Material* and *History* all serve to make the requirement coherent, to

make it possible to discuss the details of the requirement and to understand how it fits into the whole project.

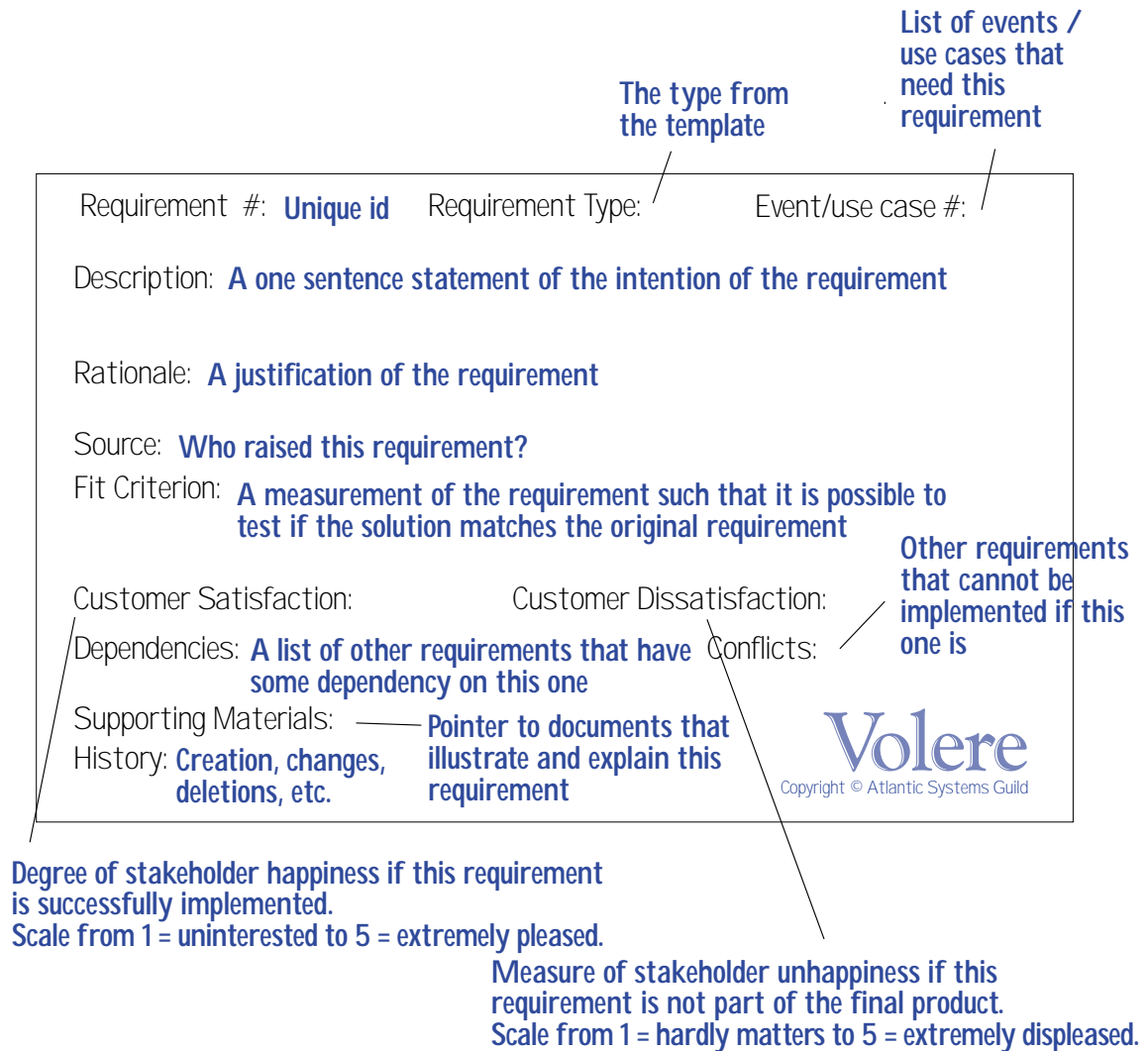


Figure 2: The requirement shell identifies the components of an individual requirement

Other sections of the template deal with individual functional and non-functional requirements and constraints and a variety of project issues. I do not intend to discuss the complete contents of the template in this paper. You can find the complete template on our web page at <http://www.systemsguild.com> where it is available for you to download. The point of introducing the template is to illustrate that there are a number of components that you can expect to find in a requirements document. You might have a number of phases, each of which contains different components at different levels of detail. For instance many organisation produce a feasibility study or initial document that might contain sections 1-7, some of section 8. Then the next version of the requirements document contains the detailed requirements in section 9 to 17. Some organisations

decide that sections 18-27 will not appear in their requirements documents because strictly speaking they are not requirements but are more related to project management. It all depends on how your organisation works and who needs to see what. The point is that the project management decisions should be based on the content of the requirements and in order for this to happen the parts of the specification need to have well defined connections.

## Keeping Track of Connections

Look back at the components listed on the requirements specification table of contents on Figure 1. There are many connections between the components and, in order to build the product and deal with changes, we need to be able to understand these connections. For example which requirements are to be implemented by which product use cases and which product use cases are necessary because of which business events. To test the completeness and traceability of the requirements it must be possible to identify these connections. Figure 3 is a model that shows the connections between the components of the requirements specification.

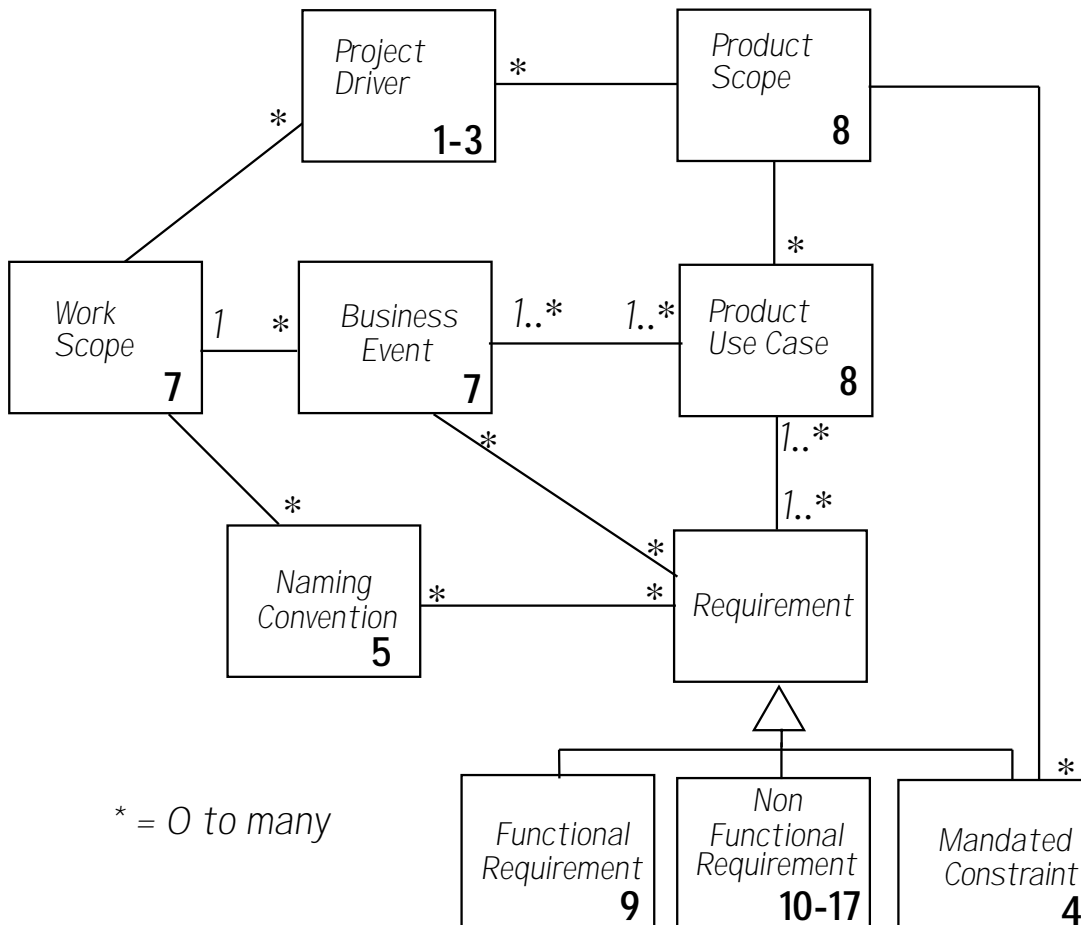


Figure 3: One test of a requirements specification is to verify that the connections between the components are traceable.

Each class represents one or more of the components listed on the requirements specification table of contents. The numbers in the boxes correspond to the numbering scheme used on the table of contents. The lines between the classes indicate that there is a connection between them. For example the line between *Product Use Case* and *Requirement* is there because each Product Use case can have many connected Requirements and each Requirement might be connected to many Product Use Cases. It does not matter how these connections are implemented within your requirements specification (there are many tools available to help with this); what matters is that it is possible to keep track of the connections between components and the quality of content of each component.

## The Quality Gateway

The purpose of the quality gateway is to have a checkpoint where you apply a series of tests to each requirement in order to determine whether it should be accepted into the specification. The aim is to discover problems like ambiguous requirements, missing requirements, inconsistent requirements – and to raise questions early rather than later on in the development process when it takes much more effort to deal with them. Provided the requirements have been defined in a consistent and testable way then people with testing expertise can put the requirements through the quality gateway tests. Here are some testpoints that you can build into your quality gateway:

- Does the specification contain all the requirements components?
- Does the specification contain all the relationships between components?
- Are all types of requirements defined?
- Is each requirement uniquely identifiable?
- Does each requirement contain all components on the shell?
- Is each goal measurable?
- Is the work context defined?
- Does the event list agree with the work context?
- Are all the event inputs and outputs defined?
- Can you quantify why each constraint exists?
- Is each constraint measurable?
- Are the stakeholders identified by name?
- Are the users' characteristics defined?
- Are the stakeholders' responsibilities defined?
- Can you trace the product use cases back to the business events?
- Are the terms used in the fit criteria defined in the requirements specification?
- Are the terms used consistently?
- Is it possible to write a cost-effective test to prove or disprove any solution to this requirement according to its fit criterion?
- Is each requirement really a requirement or is it a solution?
- Are conflicts between requirements identified?

If your project team is aware of the testpoints, then this awareness will help requirements engineers/developers/systems analysts to improve the quality of the

requirements. The quality gateway test is a point (or points) in the project where you say let's use testing expertise to see how well we are doing before we issue a document and allow errors to pollute the product.

## References

Gilb, Tom with Susannah Finzi. *Principles of Software Engineering Management*. Addison-Wesley, Wokingham, England, 1988.

Jackson, Michael. *Software Requirements & Specifications: a Lexicon of Practice, Principles and Prejudices*. Addison-Wesley Longman, Wokingham, England, 1996.

Leffingwell, Dean and Don Widrig. *Managing Software Requirements - a Unified Approach*. Object Technology Series, Addison-Wesley, 2000

Pardee, William. *To Satisfy & Delight Your Customer*. Dorset House, New York, 1996.

Robertson, James & Suzanne. *Mastering the Requirements Process*. Addison-Wesley, London, 1999.

Volere The Volere requirements template can be found at <http://www.systemsguild.com>

## Suzanne Robertson

Suzanne is co-author of *Mastering the Requirements Process* (Addison-Wesley 1999) a book that provides guidance on finding requirements and writing them so that all the stakeholders can understand them.

Suzanne is a principal of The Atlantic Systems Guild, a systems think tank. She has more than 30 years experience in systems specification and building. Her courses on requirements, systems analysis, design and problem solving are well known for their innovative workshops and business games.

Suzanne has varied experience as a manager, programmer, analyst, and designer. Since 1978, she has consulted, done research and taught in Europe, Australia, the Far East and the United States. She specialises in helping organisations to adapt modern systems development techniques to fit specific projects.

Current work includes research and consulting on patterns and the specification and reuse of requirements and techniques for assessing requirements specifications. The product of this research is *Volere*, a complete requirements process and template for assessing requirements quality, and for specifying business requirements.

Apart from her books, Suzanne is author of many papers on systems engineering. She is a member of IEEE and the Australian Computer and the British Computer Society's Requirements and Reuse Groups.