PROCESS AUTOMATION 02

16 Tips for Writing Better Functional Requirements

Business Management brought to you by

nimblex

One system. Countless possibilities

16 Tips for Writing Better Functional Requirements

Our Nimblex Platform offers unprecedented flexibility, which means even if you get some of the requirements wrong, we can still fix it later on. As long as we get the big picture correctly defined, then we can always make changes with the smaller aspects of your solution as you use it.

Make no mistake, however, that it is important to get requirements right from the start. Requirements are the primary means of communication between users and eBMS configurators. They should be prepared as carefully as if you were writing out a contract. For example, it will contain the conditions governing whether the configured solution will be acceptable or not.

Crafting requirements is not difficult. EBMS are capable of crafting your requirements for you, however, in most cases, especially with large organisations, it is best that you do this 'in-house'. It's not necessary for you to be a highly-qualified Business Analyst to do this. All that is required for you to do is to have:

- Credibility with the users and their area, as users will be more likely to share their thoughts with someone they trust. This will avoid a 'them and us' attitude which will not be helpful.
- Excellent listening skills and the ability to communicate with clarity. This will help to analyse, to validate, to double check and to clarify requirements as received from users.
- Firm, yet wise, management of a project plan.

The good news is that you will most likely improve through continuous learning and practice.

However, there are no substitutes for good requirements. It would be very beneficial for analysts to pay attention to every little detail ranging from the style, structure, and presentation to content; this will increase the chances of delivering successful systems.

What defines a good requirement document?

There is no silver bullet or 'one-size-fits-all' approach to writing requirements, but if you allow yourself to be guided by the following structure, then it will certainly improve the clarity of your requirements.

Ideally, every requirement statement (written from the user's perspective) should contain:

- a user role that benefits from the requirement
- a desirable state that the user role wants to achieve
- a metric that allows the requirement to be tested, where applicable

Documenting your Requirements

When documenting your functional requirements, think of who you're talking to and what level of technical knowledge they understand. Then when you writing it down, you're really just capturing the information and then presenting it back to your audience for validation. Remember it is not just about words on paper, requirements are also drawings, such as a workflow diagram and the like.

In addition to the above structure, the following are some useful tips on what to do, and what not to do, which should help guide you.

16 Tips for Writing a Good Requirement

1. Define one requirement at a time; each requirement should be independent. Try to avoid the use of conjunctions like ‘and’, ‘or’, ‘also’, ‘with’ and the like. This is vital because words like these may cause configurators and testers to miss out on some requirements. One way to avoid missed requirements is by splitting complex requirements until each one can be considered a discrete test case.

2. Avoid using loophole clauses like ‘but’, ‘except’ and ‘if necessary’.

3. Each requirement must form a complete sentence with no jargon, buzzwords, or acronyms.

4. Each requirement must contain a subject (user/system) and a desired outcome (intended result, action or condition).

5. Avoid the ‘how to fix it mode’, i.e. describing how the system will do something. Only debate what the system will actually do, and try to avoid system design thinking. Generally, if you catch yourself mentioning field names and database entities in the Requirements Specification Document, you’re in the wrong place.

6. Avoid ambiguity caused by the use of acronyms like ‘etc’, ‘approx’, and the like.

7. Avoid the use of indefinable terms like ‘user-friendly’, ‘versatile’, ‘robust’, ‘approximately’, ‘minimal impact’, etc. Such terms often mean different things to different people, making it difficult to define their test cases.

8. Avoid rambling, and using unnecessarily long sentences or making references to unreachable documents. As a general rule: Keep it short and simple.

9. Do not speculate; instead try to be realistic. Don’t waste time drawing up wish lists of features that are impossible to achieve.

10. Avoid duplication and contradictory statements. Be clear and to the point.

11. Do not express suggestions or possibilities. You can identify these wherever you see statements with ‘might’, ‘may’, ‘could’, ‘ought’, etc. Apply the principle of: “It is either black or white, but no grey areas, please”.

12. Simplify your requirements documentation so that it is a practical and easy to reference document.

13. Use positive statements such as “The system shall…”, instead of “The system shall not…”

14. “Shall“ should be used where requirements are being stated, “Will” should be used to represent statements of facts; and “Should” to represent a goal to be achieved.

15. Explore software options to capture your requirements.

16. Create the following template in MS Excel to help you gather requirements:

Client: ABC Council

Solution Title: The Better Contract Management Solution

Process name: Procurement and Contract Management

	User Stories					
Category	**Name & Description**	**As a(n) <actor>**	**I would like to <description>**	**I should ... <description>**	**So that <outcome>**	**Priority**
Functional Requirements for Contract Management	Subcontractors: This section relates to sub-contractors working with the contractor.	Contract Manager	See and report on the following information: Sub-contractor's business name, key contact, phone number, address, safety orientation (Y/N)		I can better control and track the sub-contractors on site	