Best Practices for writing Requirements

By: Rami Azulay
ALM Master @ Orcanos.com

Rami Azulay brings to this article on writing Requirements, 20 years of experience with a Medical, Government, Telecommunication, Finance, Startups and Enterprise organization, working on their requirements and testing. Rami now work in Marketing for ALM Systems for both Regulated and Nonregulated organizations. In this piece, he shares best practices and lessons learnt.
The Importance of Good Requirements

Quoting baseball player Yogi Berra who said, “If you do not know where you are going, you will wind up someplace else”, Rami begins his discourse by pointing out that Berra’s statement is true of Requirements. Rami continue that, many people are very aware of the impact Requirements managements has on programs. While this is so, he however is clear that this discussion is not about managing Requirements, but rather talking about writing the Requirement itself.

It is not how well one manages their Requirements. However, if one doesn’t start with high quality Requirements, it really is not going to matter. It is therefore imperative to practice writing good Requirements, as these will have a great impact on your business.

The Purpose of Requirements

Requirements and Requirements management need to do two things:

1. Validate: Making sure you are building right products. In other words, are we doing the right thing? This is the upfront portion. When working with stakeholders, it is important to understand their Requirements, and ensure we are developing the right product.

2. Verify: Secondly, make sure the product is built right. Are we doing the thing right? Verifying is ensuring that what we have put together, what we have designed, has actually been done correctly. Although validation is at the beginning of the process, and verification is at the end, it is really important to appreciate that Requirements management continues throughout the entire ALM development process.

To do this well, a good environment and good processes to analyze and develop Requirements is needed, for the complete development process.
The Challenges of writing Good Requirements

While it should be simple, writing good Requirements can be truly difficult. To start off, there are different teams, and different groups of people who meet different things, for different sets of Requirements. Different industries have different needs. One example Rami recalls is being on an elevator with his 6 years old son, who was on his iPad. He asked him, “Which would you rather fail, your iPad, or the engine of the elevator?” You can imagine what his answer was.

It very important to understand that different industries need different type of Requirements processes. Additionally, Engineers like to solve problems. It makes a lot of sense that if you think before you act it saves time. For some reason when an Engineer is given a bunch of Requirements, all of that goes out the window. But it's actually very important for us as Engineers to really stop and think before we start developing.

As well, Requirements are always changing. There could be changes in the environment, changes in technology, changes in what the stakeholders would like, and changes in the budget. So one thing we can always be sure of is that Requirements will always be changing. One should also keep in mind that the longer the development cycle, the greater chance Requirements are going to be changing.

Other things that make Requirement management complex, is that Requirements come in from everywhere. There are government regulations, and even customer requirements that can also come in form of complaint. It is very important that one is able to understand from all of these requests coming in, what the actual Requirements are.

Quite often, we end up accepting things that are really not Requirements. However, we must gather these possible Requirements from these different places, put them together and do an analysis of what really is a Requirement, and what is not.
Over Designing:

Being an engineer is something we are great at, but over designing is something we should often take a step back for, and really truly look at. Getting the solution you need without over engineering it by adding extra electronics, or over complicating things. When it comes to software that can go real wild, too many gadgets or complicated flow to perform simple task can make the whole difference. Here is one small example, looking at the implementation of the Silent mode in different smartphones shows how simple and very commonly used task was implemented differently. if you are iPhone owner then this will be as simple as switching mechanical switch to turn your phone into silent mode, on all other phones that will take you 3 - 4 steps on software menus to get to the same result. Until today no other vendor adopted this method. Sometimes it reflects about the engineering thinking of such company. The simplest solution is often the best solution.

Getting Agreement:

It is not always easy when you go through Requirement reviews to get everybody to agree on them. Requirement writing is not exactly an exact science. The fact that there are many many different types of Requirements; we have customer Requirements, structural Requirements, functional, and non-functional designs among others we have to be aware of, and manage. So a documented discussion about the requirements and the decision taken based on that discussion are imported to stay focus and tuned to those changes.

Writing for what is not visual:

Another factor which complicates writing Requirements, is having to write Requirements for things you don’t see. For example, you may be receiving messages, but you can’t see them. This makes it very difficult to visualize them and to write them. You do have to get into the practice of not just writing on what you can see, but really understanding what is out there. When it comes to system validation, one’s understanding of the Requirement is failure #1. Therefore, any effort made to adequately describe the
Requirement, is a good investment. What will be ensured, is that the chance of getting what was requested, will be greatly increased. There will be no need to rework what needs to be done at the ATP stage of your product.

Multiple Requirements Set:

On occasion, you and the organization might be dealing with multiple Requirements sets. Also, a large number of Requirements, different level of Requirements, version control, change control, product line, distributed teams and different teams, different parts of your organizations, or partnering with other organizations, might have different processes. To keep track on the flow of the requirement from one area of the ALM to the other, you need traceability. The traceability tool that is mandatory by regulation, but also part of the Good Manufacturing Practice (GMP), provides needed control over the change that may happen in every level of Requirements. So if you have a requirement tool that can create those traceability tables for you, and at the same time release the constraints that may come from the layout of those requirements as they appear in your document, then you are on the right track.

There are also systems of systems, and one that seems to be becoming more important, is the convergence of new and old technologies. What this means is that there is often times for example Rami says, instances when the Navy has a large ship which could be 50 years old, but it must be fitted with newer technologies. It is situations like this one, which makes writing Requirements complicated.

It is also important to know that working with Requirements is a lifecycle activity, going from validation, all the way to verification, and also in between. It is not something you
write in the beginning, and at the end you mark off. These Requirements require ongoing developing, and changing and improving over the lifecycle. So whether one is in the analysis and design phase, or the testing phase, there will be continuous tweaking and updating.

**Writing Requirements**

The art of writing Requirements takes great skill, and like writing code, the end result will be clearer and more consistent, if there’s a single author. Since there are likely going to be multiple authors writing and developing them, it helps to have rules or a system that facilitates discussions on a topic in your Requirement, and at the end merges all the ideas into one consensus.

One of the most important things to understand is that the Requirements will never be 100% correct. What you may think is 100% correct, will likely not be thought of the same way by the person next to you. Requirements therefore need to be written in a way that you know they will be constantly improving over time. You need to allow for human nature, and use the same language as your client.

The following guidelines provide 15 tips, or best practices. They are lessons learnt on how to write better Requirements, by following simple rules for word selection and sentence structure. Some are simple and obvious, while others are more difficult. Some will apply to you in your business, and others will not.

Most of these guidelines only apply to the Requirement statement, not the complete specifications.

1. **Use the simplest words** to state the complete Requirement. A quickly written Requirement is probably not a good one. However, the easier and more limited the vocabulary used, the better the set of Requirements will be. One way to do it, is to break the Requirements into short descriptions, such as a heading (in an office document) and then have more elaborated description. Whenever I see a requirement that is described in one block of text, that's the place where interpretation becomes possible and other ideas begin to take form.

   Additionally, there will be less chance to misinterpret the meaning, and the chance of more people really understanding what it is you are trying to get across in the requirement, will be increased.
2. **Use requirement imperatives correctly.** In many Requirements there are imperatives such as ‘shall’, ‘will’ and ‘should’. Often persons say a ‘shall’ is a requirement, others that a ‘shall’ and a ‘must’ is a requirement. It is extremely important to understand which of these is going to be contractually obligated to deliver on.

Sometimes ‘will’ is a Requirement, but you don’t have to verify. There should be a company standard which outlines what each of these mean, and just be able to use them correctly when writing Requirements, or updating Requirements. One should ensure one is defining what the requirement is, and what the goals are.

3. **Don’t use weak phrases and subjective words.** This is key, as often persons see a word and the meaning is totally misinterpreted. For example, if one writes, “…Something bad happened to the system”. Does “something bad” mean that it got a little ding, or does something bad means being obliterated down to the siliconic particle? It means different things to different people, so it's extremely important not to use these words. Or if these words are used, it is important that they are going to be interpreted the same way by everyone.

4. **Use continuations carefully**, as they make traceability and verification extremely difficult. Use continuations when all items are to be verified by the same method, and at the same time.

   **For example:**

   The system shall report software status to the host interface under the following conditions:
   - At system initialization
   - When the status of an external interface has changed
   - When a report has been requested

   This example is not one requirement, it is three Requirements. It would be important to rewrite this requirement for such reasons as traceability and verifications.

   Say one was in the verification process, and in a particular version of a release, testing at system initialization. What happens here is that the Requirement of
being verified becomes marked off, leaving the other two parts untested. Therefore, though the end of the product might actually look like verification is complete, quality will break down, there will be errors and bugs. Later on, this will prove to be more costly and difficult. The same applies for traceability. So there must be a single requirement in a statement.

5. **Tables and Figures:**
   Table and figures are a great source of information and clarification. Research shows that the more figures and tables you have in a document, the higher the quality of that document. It is therefore recommended that tables and figures be used. As they are applied, it must be clear whether they are there for reference or as Requirements.

   Said another way...Sometimes you put tables and particular cells might contain information that actually limit Requirements. One should therefore be very careful and make sure they are really specifying if the tables and figures are just for information, or if there are parts which contain Requirements, or requirement information.

6. **Consistency:**
   If there is a large group of people writing Requirements, there may be some confusion. It is therefore key to always call the same entity by the same name. For example, if in some Requirements the subject is called ‘the system’, and in others “the PR-65”, the name is not consistent. Ensure that everybody is using the same nouns, same verbs, when writing the requirement set. Always use the correct name for the level of specification. You cannot verify a sub capability at a systems level.

7. **Using TBD or TBR:**
   When you are writing and developing Requirements, it is likely that you will get to a point where you just don’t know the answer...and that’s ok. You won’t always know everything. Sometimes needed information will not be available, or you will need to consult an expert.

   You should have a plan. As Requirements are written so as to develop over time, to become a higher quality; you will want to ensure that if you put something in a
TBD, that it is not ignored. You must ensure you have a plan which outlines who is responsible for getting the data, who is responsible for the information, and when it will be completed. If you have a system that manages those requirements over a database then you may want to use an entity such as Action Items to list all the actions and their owner which are related to that TDB.

**Make sure a Requirement contains all the requirements of a good Requirement**

Ensure that your requirement contains qualities of good Requirements writing, such as:

A. **Concise**: Minimal, easily understood, a complete expression of a single thought, non-ambiguous, only one possible interpretation.
B. **Correct**: An absence of errors of fact
C. **Consistent**: No conflicts between individual Requirements, parts of a single requirement complement each other. Connectivity exists between the Requirements, consistent words and terms.
D. **Traceable**: Know source of requirement and be able to allocate it. Uniquely identified for life. Never reused identification on project.
E. **Verifiable**: Method (Test, inspect, demonstration, Analysis, Certification) Understand how requirement can be verified, and determine criteria for acceptance.
F. **Necessary**: Can the system be complete without this requirement?
G. **Attainable**: Is this requirement technically feasible within given time and cost?
H. **Modular**: Will a change to this requirement have a big impact on the system? Can this requirement be easily used and monitored by other programs if needed.
I. **Restrictive**: The requirement should be written in such a way as to not limit implementation. Make sure the requirement states what needs to be done, not how.

8. **Conjunctions:**

Earlier we talked about the need to have one requirement per statement. One way of ensuring this, is to limit the use of ‘ands’ and ‘ors’. They are not great words to have in Requirements. They should be broken out into separate
Requirements. With good Requirement management tools, your concern about mass data, created by many little requirement, is reduced to near zero. Single Requirement can be broken up into as many Requirements as you wish.

9. References:

As well, make sure that if a Requirement references other documents, make sure it is done correctly. What this means is that whereas sometimes in Requirements there is an upfront section that says, ‘shall be compliant to some type of standard’. Maybe this standard its pointing to is 10,000 pages long. You would not want to have those 10,000 pages being Requirements. If something is a reference, you want to make sure you have pointed out if its Requirements or they are just particular parts of references. Here is the place again where you can enjoy the benefit of Document Management System (DMS), a tool that can archive, manage and control electronically the changes in the course of document lifecycle. You can create traceability and references so that any change in the Requirement, will be highlighted, reducing conflicts with other documents and vice versa. When it comes to Medical Devices most documents reference many standards such as IEC 62306, ISO 14971, ISO 13485, IEC 60601, IEC 62366 etc.

10. Acronyms:

Make sure acronyms are used correctly.
  a. Place the acronym n the acronym list in the specification.
  b. Spell out the complete phrase followed by the acronym in parentheses the first time used
  c. The next time just use the acronym
  d. Example, first use: “The Automatic Test Equipment (ATE) shall…” Second use “The ATE shall…”
  e. It is also suggested that you have an acronyms list at the end of your document, should persons want to know what that acronym is for and its specific specification, they’ll be able to easily find that information (Pressing Ctrl-End).
11. **Over-specification:**

This leads to unfunded Requirements, and can add duplicate Requirements.

a. The length of a requirement should not be excessive.
b. Keep them as simple and complete as possible.

12. **Use a requirement template:**

This is one of the best practices to employ when there is the need to have consistency, and there are multiple persons, complete teams, or distributed teams writing or updating Requirements.

There are four major parts to a Requirement

a. **Entitles:**
   i. Subject of the Requirements (noun)
   ii. Object of action (noun)

b. **Actions** – What the subject does, contains imperative (verb)

c. **Conditions** – What must be in place in order for this action to take place

d. **Constraints** - Qualifies the action, performance

*For example:* When signal X is received (Condition), the system (Subject) shall set (Action) the signal X received bit (Object) within 2 seconds (Constraint).

When everyone on your team is using the same template, at least when you now put the Requirements together it would look like a consistent, higher quality overall document. The better Requirement management systems allow you to create such templates for each Requirement type; so whenever the user is creating new Requirements, the system automatically will generate the template text for that specific Requirement type. The more sophisticated system even allow you to create different templates of the same work item type between different projects so if you have complex project that include software on the hardware module and software on the client each software requirement will contain different template although both are of the same type.

13. **Design for Asset Reuse:**
The reuse of assets is a strategy when writing Requirements. Sometimes product components need to be designed with its ability to fit into several products, not just one. This is what makes it difficult. What seems to be best for a single configuration may not be as useful in a family of products. Good ALM systems can allow you to modelize the components of your product, and at the same time, the Requirements. Each module and its Requirement can be then managed once in different line of products (ALM solution) and version. Such capability increases your ability to reuse data, knowledge across products and control the implication of one change on many solutions.

14. Simulations and modelling:

Writing textual Requirements using simulations and modelling helps you to determine design Requirements, what’s smart and what’s not smart. So when building Requirements, you might want to actually look into some of these other techniques.

15. Testing:

Testing is also beneficial, as it helps us to know the behavior of our system. Often times when developing Requirement, and it is discovered that a task can be done multiple ways, tests can be run to see how the system works, or to decipher more efficient ways of implementing something. So testing requirements to be part of the requirement definition will assure to the stakeholder implementing them, how the requirement shall be used by the end user.

Conclusion

So it is very important when writing textual Requirements that it is seen as a development process, which takes place over time. They will be changing between validation and verification. The tips and techniques shared can help in the writing and improving of Requirements over time.

Rami Azulay - ALM Master @ Orcanos.com