Requirements Management In Action

A beginners guide to Requirements Management

Psoda

Table of Contents

Introduction

Capturing requirements is what I call a swan task – on the surface it looks really easy and not much effort but underneath is a heck of a lot of work. It's also one of those jobs that if it isn't done properly it will have a significant negative impact on your project, and normally by the time it's discovered it's too late to do anything about it without major time and cost impacts. Before we get into the tips its worth noting what requirements actually are.

What is a requirement?

A requirement is a **single** documented physical or functional need that a particular design, product or process must be able to perform. Note the emphasis of single - I've lost count of the number of requirement documents that have multiple needs listed as one requirement. This makes it almost impossible to ensure that all the requirements are met in the final product, whether this is a piece of software, a bridge or a power station. Requirements are normally categorised into two distinct types – functional and non-functional.

So, what's the difference?

Functional requirements describe what the product should do while **non-functional** requirements describe success criteria.

For example:

A functional requirement for a software system might be that if a person registers for an account, an email will be automatically sent to the creator.

A functional requirement for a power station might be the level of power that can be generated at a given time.

A non-functional requirement for a software system might be that it has to have an up time of 97%.

A non-functional requirement for a power station might be that it should not have an environmental impact on the water that passes through the power station.

Chapter 1

How to Capture Requirements

The process I'm about to outline looks intensive but it can be tailored to meet any size and type of project. Also, each step can be as detailed or as high level as you need. The objective is to ensure you get as detailed and accurate a set of requirements as possible to minimise pain later on in the process. This process can be used with both agile and waterfall methodologies, the only thing that changes is the level of detail at each step.

Identify your stakeholders

This list of people should be more than just the users and developers of the solution. It should include representatives from all areas of the business that are likely to be impacted by the potential solution. It can also be useful to get senior management involved at this stage.

Gather requirements

Requirements can be captured in any number of ways and there is no right or wrong answer to this. Your particular project methodology will determine the level of detail required. For example a standard waterfall development will require extremely detailed requirements while an agile methodology will not, as the requirements will be refined at each review cycle.

Regardless of the method used it's worth developing a standard template to capture the information, that way you are sure that the information is in a consistent format and that it is as detailed as possible.

Whatever way you capture requirements it is best to be as collaborative and open as possible. A series of requirements gathering workshops is normally the most efficient way of doing this. If you do choose this route make sure that there is someone in the room whose sole responsibility is to be the scribe. This will ensure that important information is not lost.

Documenting requirements

When you are documenting the requirements it is worth capturing them in a central repository, whether that is a document or some type of requirements database.

Some organisations still use manual processes for requirements management, while they have their place it is far better to automate this as much as possible to ensure requirements are not overlooked or updated as things (inevitably) change.

The key outputs of this process will be a requirements document (or database) that has been agreed to by all stakeholders and a **requirements traceability matrix** that shows the relationships or dependencies between requirements.
while they have their place it is far better to automate this as much as possible to ensure requirements are not overlooked or updated as things (inevitably) change.

The key outputs of this process will be a requirements document (or database) that has been agreed to by all stakeholders and a **requirements traceability matrix** that shows the relationships or dependencies between requirements.

Requirements feasibility

Once all the requirements have been defined it is time to assess their **priority and feasibility**. There are a number of different techniques that can be used to analyse requirements. They include:

- Mandatory and Optional
- Ranking
- Value Analysis
- Requirements Risk Analysis
- Kano Analysis (delighting the customer)
- Stakeholder impact analysis
- PERT Analysis (Sequencing)

The most important part of this process is to determine the costs of each requirement and compare that to the cost of how things are currently being done, or if nothing is in place then the cost of doing nothing.

The key output of this process will be a signed-off, prioritised list of requirements and well as a draft budget.

Design & build the solution

Now you have a full set of requirements and a budget, it's time to **design** the solution. At this point the requirements manager's role becomes more about overseeing and monitoring the design to ensure that all of the requirements are incorporated correctly. At this point you will really begin to benefit from an automated requirements traceability matrix.

The key outputs of this process will be a detailed design, any applicable interface specifications and a more accurate breakdown. The design process may also uncover additional (more detailed) requirements or may identify requirements that are unfeasible.

Test the solution against the requirements

At this point the original requirements will be handed over to the test team so that they can build their **test cases** against the requirements. This is where you really need to have good requirements traceability. As requirements are tested and functions change it is important to ensure that the original requirements are not diluted or lost as can have a significant impact on the benefits of the solution.

Chapter 2

What is Traceability?

At its simplest, traceability allows you to find your way around your project information.
For example one high-level requirement may be broken into a number of detailed requirements. Traditionally the high-level requirement lives in a different document to the detailed requirements.

To keep the relationship, or traceability, between these requirements you need to have some type of connection or trace between them.

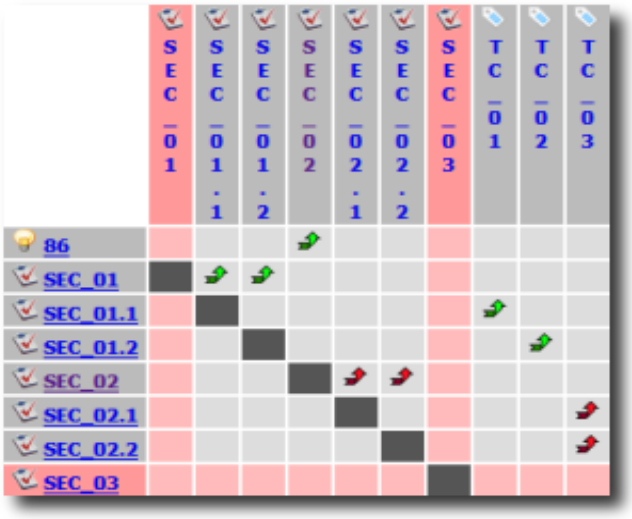
You could do this by adding a column showing the high-level requirement associated with each of the detailed requirements in the detailed requirement specification.

So what happens when things, inevitably, change? When a high-level requirement changes, you need to be able to find all of the related detailed requirements and make sure they're still valid. This can mean manually trawling through every single detailed requirements document to find those that specifically relate to the amended high-level requirement.

If you had only one detailed requirements specification this would be okay but what happens when you've got ten or more derived specifications? It becomes a bit of a pain.

Traceability Matrix

The next step is to create a traceability matrix. It should look something like this:



So how do you create a traceability matrix?

First, place all of your requirements along both axes of a table. Then mark the relationship between each requirement.

Using SEC_01.1 in the matrix above as an example, to show that requirement SEC_01.1 is derived from requirement SEC_01 you do the following:

1. Find the row with SEC_01 in the left-

hand column
2. Follow that row to the right until you find the column with SEC_01.1 at the top

In that cell you place and arrow to show the relationship or trace

You now have the beginnings of a traceability matrix.

Using SEC_01.1 in the matrix above as an example, to show that requirement SEC_01.1 is derived from requirement SEC_01 you do the following:

1. Find the row with SEC_01 in the left-hand column
2. Follow that row to the right until you find the column with SEC_01.1 at the top

In that cell you place and arrow to show the relationship or trace

You now have the beginnings of a traceability matrix. To find all of the requirements that are derived from a high-level requirement you now only have to find the row for the high-level requirement and follow that row to the right. Each cell with an arrow in it will indicate a derived requirement.

What about finding any high-level requirements that have been forgotten and do not have any detailed requirements defined yet? If your requirements live in different documents as I described earlier you'd end up having to make a checklist of all the high-level requirements and then go through each of the detailed requirements and tick of all of the high-level requirements that have detailed requirements defined for them.

With the traceability matrix it is a simple matter of finding any rows that have no arrows in them at all. In the image above you can easily see that SEC_03 has no derived requirements. In Psoda the row is automatically coloured rose to highlight that this requirement has no derivatives for it. Alternatively, you can easily find requirements that have no determinants or parent requirements associated with them. Just look out for columns with no arrows in them. Once again in Psoda these columns are automatically highlighted in rose to make it even easier to spot them.

Test-cases

Of course you don't have to stop with requirements. Depending on the type of project that you are running, you may be creating test-cases to test the functionality as defined by the requirements. In this case you want to extend the traceability matrix to include those test-cases. In the image above the test-cases are identified with a TC_ prefix.

All the same principles we had just been discussing with regard to requirements now also applies to the relationship between requirements and test-cases. For example to find any requirements that do not have test-cases derived yet you just find rows with no arrows underneath the test-cases in the matrix.

Change

What happens when things change and your customer changes their mind about some of the high-level requirements half-way through the project and you don't have a traceability matrix?

If you're running an agile project and haven't analysed those specific requirements yet then you're okay. But it's more likely that they will change their mind about the requirements that you've already done detailed analysis for and that you have the design and test-cases written for.

In this case you will have to trawl through every project document trying to find all of the derived requirements, designs, test-cases and other project artefacts and then update those to reflect the changes to the higher-level requirements. But updating the derived requirements may impact other requirements, designs, test-cases, etc. again. You then have to start again and find those secondary impacted items and update them accordingly.

This process will have to be repeated until you're sure that all the possible implications of the high-level changes have been identified.

With a traceability matrix this whole process is made easier. Psoda takes it one step further by automating the traceability matrix, so that if one requirement changes it automatically shows you all of the other requirements and test cases that are impacted.

Chapter 3

Tips to Capture Better Requirements

Here are a few tips to help you capture better requirements:

1. Understand that any requirements gathering will never be 100%.

2. Keep the number of stakeholders involved to one representative from each area of the organisation that will be impacted by the project. Any more than this and it becomes too complicated to manage.

3. Spend as much time as possible on gathering and refining the requirements. The more time spent at this stage getting the requirements clearly articulated the less time will be spent on rework during the design & development stage.

4. Make the requirements gathering process as collaborative as possible. Ideally there will be requirements gathering workshops with all of the key stakeholders in one room. Where this is impossible or impractical make sure that a master list of requirements is circulated often.

5. Make sure that there is only one feature or need per requirement.

Conclusion

In this eBook we covered the basics of requirements management, walking you through a process that can improve requirements capture and traceability. As you begin requirements management keep in mind that all of these efforts are directly linked to increased project success.

Seamlessly manage your requirements

Psoda provides a suite of cloud-based tools to capture, manage and visualise your project requirements. Sign up today for your 30 day free trial and see the benefits of a centralised requirements database.

SIGN UP TODAY