The Atlantic Systems Guild Ltd.

Simplicity and Requirements

By Suzanne Robertson

In today's world of system development we are increasingly concerned with being more agile, with having a lean approach, with delivering value more quickly. We are consciously looking for ways of avoiding spending time on anything that is unnecessary. We are looking for simple, elegant, spare solutions that fit seamlessly into the worlds of the people using them and so help them to do their work more effectively. We want to deliver software and have people say " thank you, this really helps me. It is so simple I am able to understand it and use it immediately". But this kind of user delight with the product always involves careful thinking and requirements gathering in order to make the software simple.

In Search of Simplicity

When someone tells you their requirements for a new or changed software product, they usually state those requirements in a way that defeats immediate and unambiguous understanding. Naturally enough people look at the world in terms of their own concerns. They probably assume that you share their contextual knowledge. Your challenge, as a requirements

specialist, is to uncover the assumptions, omissions and contextual bias to discover the real needs. You must be able to state the requirements in a simple manner so that all stakeholders have the exact same understanding.

The path to a simple understanding is usually strewn with complexities and irrelevancies. These often lead you in the wrong direction and cause unnecessary complications, features and pure bloat in your software. I have found from experience that the best way to avoid these time wasting side trips is to encourage early and continuous feedback throughout the process of requirements discovery.

Make a Mess Early

At the beginning of the project, I try and expose my understanding of the problem in a way that encourages people to give me their specific (rather than general) feedback. A technique I use for doing this is the Scope, Goals, Stakeholders (SGS) approach. You can find a detailed description of this and other approaches mentioned in this article at http://www.volere.co.uk

Scope: start by drawing a first-cut diagram of your understanding of the problem/work scope. This diagram should show all the systems adjacent to the work you are studying. The adjacent systems are the people, organisations, software systems and other technology that interfaces with your area of study. The diagram also shows the interfaces between these adjacent systems and your area of study.

Goal: Write the umbrella goal of the project along with the rationale for why it is important and a measurement for how you will know if you have succeeded in meeting the goal. The goal must address some benefit, and the measurement is usually a quantification of that benefit.

Stakeholders: do a quick stakeholder analysis to identify the people who are sources of requirements along with their roles and which requirements knowledge you expect them to contribute.

Now ask stakeholders with different viewpoints what changes they would like to make to your Scope, Stakeholders, Goals. Encourage feedback and different interpretations. Here you are taking a problem and exposing the complexity to identify the differences, conflicts and misunderstandings. By making these things visible you can identify where choices and decisions

need to be made before going into details that might be irrelevant or provide low benefit.

One side effect of this approach is that you are helping stakeholders to be aware that other people have different opinions about what is important. These need to be resolved before building your solution.

Prioritise the Pieces

Encourage more feedback by taking your understanding of the scope of the problem and attempting to break it into functionally related pieces. Then you can identify, encouraging more input from the stakeholders, which of those pieces provides the highest benefit/value for this project. In the preliminary Requirements Study in Figure 1 you decide which pieces (functions, features or components) you plan to deliver first. Take those pieces individually through your detailed development process using whatever combination of requirements stories and models that you prefer to use in order to arrive an implementation of your detailed requirements for that piece. For each implementation unit, encourage feedback from the appropriate stakeholders. Keep iterating, always encouraging feedback, until you have implemented all of the requirements within your scope.

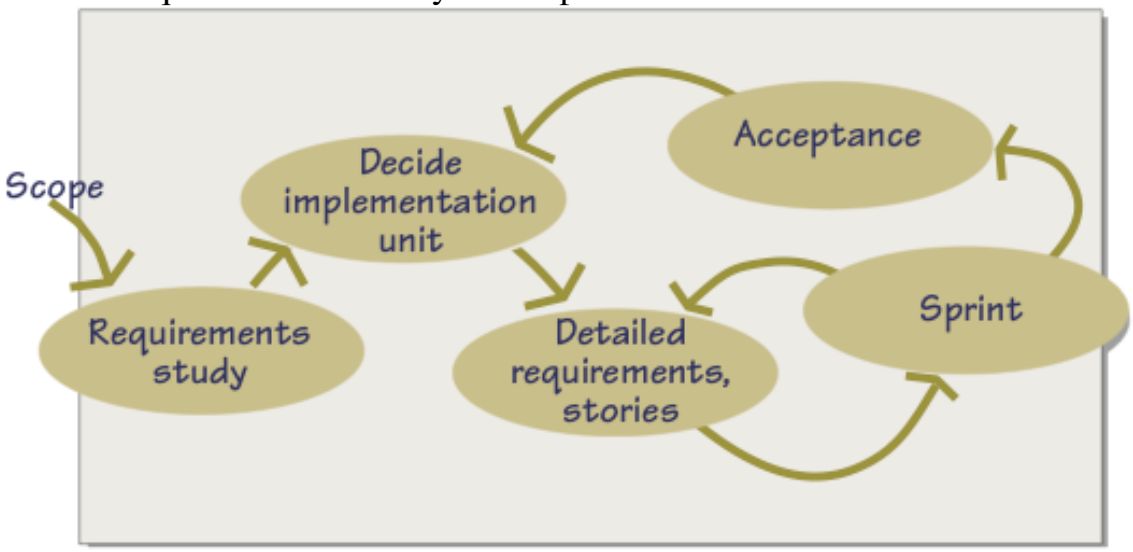

Figure 1: The requirements study at the start of a project helps you to identify the highest benefit implementation units so that you can deliver those first.

Throughout your iterative loops the requirements will tend to become more complicated. When people understand what you are doing then some of the feedback that they give you will be other possibilities and bells and whistles

that introduce new requirements. You are in a cleft stick: you want to make sure that you get all the requirements but you want to give priority to those that provide a simple, usable solution to the real problem.

Thoughtful Reduction

John Maeda, author of *The Laws of Simplicity* [1]says that "the simplest way to achieve simplicity is through thoughtful reduction". In other words we can often make a simpler product by taking something away rather than adding more complexity.

Some requirements will clearly be essential to meeting the umbrella goal of the project. It seems reasonable that an insurance claims system must be able to record claim decisions; a television remote controller must make it possible to switch the television on, a school enrolment system must accept applications from potential students. However there are almost always requirements that are unnecessary and make the product needlessly complex.

For example the claims system could in addition ask the user to enter the precise time that the claim was made down to the minute and second. The television controller could have a number of different buttons, one for each member of the family to turn the TV on,. The school enrolment system could be available on the web or mobile phone. You could take each one of these systems and add lots more additional requirements to it and while it might appear to be better because it is more powerful it is not necessarily better in terms of fitting into the world of the problem you are trying to solve.

Try applying Maeda's reduction law to requirements by assessing the effect of each requirement within the scope of the problem. If we include this requirement does it make the system more complex? If it does make it more complex does the accompanying benefit outweigh the effect of omitting it? Consider the person who is doing this work. If you omit this requirement will it make it simpler for that person to do his work?

"Perfection is reached, not when there is no longer anything to add, but when there is no longer anything to take away." Antoine de Saint-Exupéry.

[1] http://lawsofsimplicity.com

More information is available:

- http://www.volere.co.uk
- in three books written by James Robertson & Suzanne Robertson, the most relevant to this article is *Mastering the Requirements Process – second edition*.
- in Volere seminars and consulting
- on the Volere Requirements Linked In group http://www.linkedin.com/e/vgh/2491512/

Previous articles are available at http://www.volere.co.uk

Suzanne Robertson is a principal and founders of The Atlantic Systems Guild http://www.systemsguild.com and joint originator of the **Volere** requirements process, template, checklists and techniques http://www.volere.co.uk

You can contact her at
suzanne@systemsguild.net

