The Atlantic Systems Guild Ltd.

Use Cases for Useful Points of View

This is number seven in a series that explains the thinking behind the *Volere*[1] requirements techniques. Previous and future articles explore various aspects of applying these techniques in your environment.

By Suzanne Robertson & James Robertson
The Atlantic Systems Guild — August 2010

What is a Use Case?

If you ask this question in the wider community, you will come up with at least 40 different answers. This is hardly surprising – use cases are written a by many people and used by many people in many different situations, at many different levels of detail. However, there is one thing common every effective employment of use cases: they are helpful because they provide a way of identifying and communicating a cohesive chunk of functionality.

This tells us why there are so many interpretations about what should, or should not be, included in a use case. To decide the relevant details one needs to know who will be involved in writing, reviewing and using the use case. To know this helps identify the necessary point of view of the use case. In other words, the decision on what to include depends on the source, the intended consumers and the feedback providers of the use case.

If you accept this argument then it is true to say a use case can be as large or small a chunk of functionality as you need, and can contain whatever details you decide. The important point is that the intended participants arrive at a

[1] *Volere* is the Italian verb – to wish or to want

common understanding of the functionality covered by the use case, and are able to provide feedback and take appropriate action.

However, it helps to have some guidelines about what is appropriate to include in a use case in some typical situations. To start with, consider the *point of view* you want the use case to provide.

Points of View

To identify the appropriate point of view for a use case you need to decide what sort of knowledge you are looking for and how the source/s of that knowledge looks at the world. You are trying to identify the mental model of the people with whom you are communicating.

Business point of view

Software is built to support a business, and so it is appropriate (and obvious) that some of your use cases should focus on the business without paying too much attention to the technology that might be deployed to do the work. That is, the business point of view is an abstraction that is concerned solely with a statement of what the business does, and not how it does it. Your use cases here are chunks of functionality that are part of the larger business.

For example, suppose you are talking to librarians and library managers about the the business of library loans. In this event you might have a *business use case* (BUC) called, 'Business Decides Loan Extension'. This use case would contain, from the point of view of the library's business, all the decisions the library has to make in order to decide whether to extend the loan to a book borrower. You would also include what the business records about the loan extension and whether they keep a record of loans that they decide not to extend. To take this point of view you focus on talking about the business of running a library and you use terminology that is familiar to the librarians. You take the point of view that helps the librarians to give you early feedback about their business, and for the moment ignore any current or future solutions to this business problem.

User's point of view of the product

Now suppose that you want to get feedback from librarians and library clerks about how they could use an automated library loan system to support their daily work. Then you might have a *product use case* (PUC) called, 'Product Determines Extension Eligibility'. This use case is from the point of view of a user—in this case a librarian—using a software product. The

use case focuses on what information the librarian has to provide to the product, what the product will do for the librarian and what information the product will provide for the librarian. This use case focuses on an intended interface with the intention of discovering whether the product will fit what the librarian needs. The point of view avoids details that are internal to the design of the software because they are irrelevant to the librarian and will only distract him from the real requirements.

System point of view

Another point of view is that of the software system. The use cases within the software system are concerned with how the software will be partitioned, organised and connected in order to best solve the problem and satisfy the functional requirements, non-functional requirements and constraints. The audience for use cases from this point of view are developers, testers and system architects. These are people whose expertise makes a difference to the internal design of the product. Depending on your development environment these *system use cases* (SUC) will incorporate the includes and extends concepts. They might also make use of established design patterns and software packaging particular to your environment. They will add your systems architecture and implementation terminology that would make no sense to, and should not be the concern of, the business people or the users of the system.

These are just three examples of different use case points of view. There are a limitless number of variations depending on factors like your stakeholders' knowledge, their availability and degree of involvement, your implementation environment, the size of the project, knowledge of the domain and so on.

What comes first?

So which point of view should you start with? The practical answer is whatever helps you to make the most progress in each particular situation.

For example, if you are starting a project and you want to experiment with the most tricky part of the intended product, then you might choose to build some user product use cases and prototype them to see if you are on a reasonable track. If you are making a change to an existing system then you might sketch out some system use cases to identify the technical impact of the parts that you need to change. If a business person is telling you about a new piece of business policy then it makes sense to do a business use case to

help you identify the functional chunks that are important to the business, regardless of software interfaces.

You don't need to take every point of view every time.

What about traceability?

Every use case that you work with can stand alone as a point of view of a chunk of functionality. The use case encourages and facilitates feedback and helps people who share a point of view to communicate, understand and provide feedback. If you want to make sure that a use case from one point of view has been accurately translated to another point of view (something that happens many times during systems development) then you need some way of tracing the translation.

The least procedural and most effective way of providing traceability between points of view is to have a dictionary that defines the terminology within the scope of the part of the world you are trying to understand. The idea is for everyone; no matter what point of view they are taking, to use the definitions in the dictionary in order to provide traceability. Whenever they discover a term that is not yet defined the team will add its definition and those of its component parts to the dictionary. This progressive creation and use of a project wide dictionary provides an effective communication tool for the entire project team.

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 and James Robertson are principals and founders of The Atlantic Systems Guild http://www.systemsguild.com and joint originators of the **Volere** requirements process, template, checklists and techniques http://www.volere.co.uk

You can contact them at
suzanne@systemsguild.net
james@systemsguild.net

