INTERVIEW with Suzanne Robertson

Suzanne Robertson is a principal of The Atlantic Systems Guild, an international think tank producing numerous books and seminars whose aim is to make good ideas to do with systems engineering more accessible. Suzanne is particularly well known for her work in systems analysis and requirements related activities including the development of the Volere requirements techniques.

HS: What are requirements?
SR: Well the problem is that 'requirements' has turned into an elastic term. Requirements is an enormously wide field and there are so many different types of requirements. One person may be talking about budget, somebody else may be talking about interfacing to an existing piece of software, somebody else may be talking about a performance requirement, somebody else may be talking about the calculation of an algorithm, somebody else may be talking about a data definition, and I could go on for hours as to what requirement means. What we advise people to do to start with is to look for something we call 'linguistic integrity' within their own project. When all people who are connected with the project are talking about requirements, what do they mean? This gets very emotional, and that's why we came up with our framework. We gathered together all this experience of different types of requirements, tried to pick the most common organization, and then wrote them down in a framework.

HS: Please would you explain your framework? (**The version discussed in this interview is shown below. The most recent version may be downloaded from www.systems-guild.com.**)
SR: Imagine a huge filing cabinet with 27 drawers, and in each drawer you've got a category of knowledge that is related to requirements. In the very first drawer for example you've got the goals, i.e. the reason for doing the project. In the second drawer you've got the stakeholders. These are roles because they could be played by more than one person, and one person may play more than one role. You've got the client who's going to pay for the development, and the customer who's making the decision about buying it. Then you've got stakeholders like the project leader, the developers, the requirements engineers, the designers, the quality people, and the testers. Then you've got the less obvious stakeholders like surrounding organizations, professional bodies, and other people in the organization whose work might be affected by the project you're doing, even if they're never going to use the product.

HS: So do you find the stakeholders by just asking questions?
SR: Yes, partly that and partly by using the domain model of the subject matter, which is in drawer 9, as the driver to ask more questions about the stakeholders. For example, for each one of the subject matter

areas, ask who have we got to represent this subject matter? For each one of the people that we come across, ask what subject matter are we expecting from them?

Drawer 3 contains the end users. I've put them in a separate drawer because an error that a lot of people make when they're looking for requirements is that the only stakeholder they talk about is the end user. They decide on the end user too quickly and they miss opportunities. So you end up building a product that is possibly less competitive or relevant. I keep them a bit fuzzy to start with, and as you start to fix on them then you can go into really deep analysis about them: What is their psychology? What are their characteristics? What's their subject-matter knowledge? How do they feel about their work? How do they feel about technology? All of these things help you to come up with the most competitive non-functional requirements for the product.

HS: How do you resolve conflict between stakeholders?

SR: Well, part of it is to get the conflicts out in the open up front, so people stop blaming each other, but that certainly doesn't resolve it. One of the ways is to make things very visible all the way through and to keep reminding people that conflict is respectable, that it's a sign of creativity, of people having ideas. The other thing that we do is that in our individual requirements (that is atomic requirements), which end up living in drawers 9 to 17 of this filing cabinet, we've got a place to say "Conflict: Which other requirement is this in conflict with?" and we encourage people to identify them. Sometimes these conflicts resolve themselves because they're on people's back burners, and some of the conflicts are resolved by people just talking to one another. We have a continuing process of cross-checking requirements and looking for conflicts and if we find some that are just not sorting themselves out, then we stop and have a serious negotiation.

In essence, it's bubbling the conflicts up to the surface. Keep on talking about them and keep them visible. De-personalize it as much as you can. That helps.

HS: What other things are associated with these atomic requirements?

SR: Each one has a unique number and a description that is as close as you can get to what you think the thing means. It also has a rationale that helps you to figure out what it really is. Then the next component is the fit criterion, which is, "If somebody came up with a solution to this requirement, how would you know whether or not it satisfies the requirement?" So this means making the requirement quantifiable, measurable. And it's very powerful because it makes you think about the requirement. One requirement quite often turns into several when you really try and quantify it. It also provides a wonderful opportunity for involving testers, because at that point if you write the fit criterion you can get a tester and ask whether this can be used as input to writing a cost-effective test. Now this is different from the way we usually use the testers, which is to build tests that test our solutions. Here I want to get them in much earlier, I want them to test whether this requirement really is a requirement.

HS: So what's in drawers 18 through 27?

SR: Well here you can get into serious quarrels. The overall category is 'project issues,' and people often say they're not really requirements, and they aren't. But if the project is not being managed according to the real work that's being done, in other words the contents of the drawers, then the project goes off the rails. In project issues we create links so that a project manager can manage the project according to what's happening to the requirements.

In the last drawer we have design ideas. People say when you're gathering requirements you should not be concerned with how you're going to solve the problem. But mostly people tell you requirements in the form of a solution anyway. The key thing is to learn how to separate the real requirements from solution ideas, and when you get a solution idea, pop it in this drawer. This helps requirements engineers, I think, because we are trained to think of solutions, not to dig behind and find the real problem.

HS: How do you go about identifying requirements?
SR: For too long we've been saying the stakeholders should give us their requirements: we'll ask them and they'll give them to us. We've realized that this is not practical—partly because there are many requirements people don't know they've got. Some requirements are conscious and they're usually because things have gone wrong or they'd like something extra. Some requirements are unconscious because maybe people are used to it, or maybe they haven't a clue because they don't see the overall picture. And then there are undreamed-of requirements that people just don't dream they could ever have, because we've all got boundaries based on what we think technology is capable of doing or what we know about technology or what our experience is. So it's not just asking people for things, it's also inventing requirements. I think that's where prototyping comes in and scenario modeling and storyboarding and all of those sorts of techniques to help people to imagine what they could have.

If you're building a product for the market and you want to be more competitive you should be inventing requirements. Instead of constricting yourself within the product boundary, say, "Can I push myself out a bit further? Is there something else I could do that isn't being done?"

HS: So what kinds of techniques can people use to push out further?
SR: One of the things is to learn how to imagine what it's like to be somebody else, and this is why going into other fields, for example family therapy, is helpful. They've learned an awful lot about how to imagine you might be somebody else. And that's not something that software engineers are taught in college normally and this is why it's very healthy for us to be bringing together the ideas of psychology and sociology and so on with software and systems engineering. Bringing in these human aspects—the performance, the usability features, the 'look and feel' features—that's going to make our products more competitive.

I always tell people to read a lot of novels. If you're having trouble relating to some stakeholders, for example, go and read some Jane Austen and then try to imagine what it would have been like to have been the heroine in *Pride and Prejudice*. What would it have been like to have to change your clothes three times a day? I find this helps me a lot, it frees your mind and then you can say, "OK, what's it really like to be that other person?" There's a lot to learn in that area.

HS: So what you're saying really is that it's not easy.
SR: It's not easy. I don't think there's any particular technique. But what we have

done is we have come up with a lot of different 'trawling' techniques, along with recommendations, that can help you.

HS: Do you have any other tips for gathering requirements?

SR: It's important for people to feel that they've been heard. The waiting room (drawer number 26) was invented because of a very enthusiastic high-level stakeholder in a project we were doing. She was very enthusiastic and keen and very involved. Wonderful! She really gave us tremendous ideas and support. The problem was she kept having ideas, and we didn't know what to do. We didn't want to stop her having ideas, on the other hand we couldn't always include them because then we would never get anything built. So we invented the waiting room. All the good ideas we have we put in there and every so often we go into the waiting room and review the ideas. Some of them get added to the product, some are discarded, and some are left waiting. The psychology of it is very good because the idea's in the waiting room, everyone knows it's in there, but it's not being ignored. When people feel heard, they feel better and consequently they're more likely to cooperate and give you time. ■

The Template

PROJECT DRIVERS

1. The Purpose of the Product
2. Client, Customer and other Stakeholders
3. Users of the Product

PROJECT CONSTRAINTS

4. Mandated Constraints
5. Naming Conventions and Definitions
6. Relevant Facts and Assumptions

FUNCTIONAL REQUIREMENTS

7. The Scope of the Work
8. The Scope of the Product
9. Functional and Data Requirements

NON-FUNCTIONAL REQUIREMENTS

10. Look and Feel Requirements
11. Usability Requirements
12. Performance Requirements
13. Operational Requirements
14. Maintainability and Portability Requirements
15. Security Requirements
16. Cultural and Political Requirements
17. Legal Requirements

PROJECT ISSUES

18. Open Issues
19. Off-the-Shelf Solutions
20. New Problems
21. Tasks
22. Cutover
23. Risks
24. Costs
25. User Documentation and Training
26. Waiting Room
27. Ideas for Solutions

The Volere Requirements Specification Template © Atlantic Systems Guild.