The Atlantic Systems Guild Ltd.

Models or natural language – which is best for requirements?

This is the ninth in a series that explains the thinking behind the *Volere*[1] requirements techniques—previous and future articles explore aspects of applying these techniques in your environment. In this article we start a discussion about combining models and text to ensure a better understanding of the requirements.

By Suzanne Robertson & James Robertson

The Atlantic Systems Guild

The Best Medium?

You have ideas for a new design for your garden and you want to communicate the details to your partner. Keep in mind that a garden is a multi-dimensional system, with both intangible and functional considerations. You could draw a sketch of the new layout of the garden, take pictures of similar gardens or make a list of the Latin names of all the plants that you propose for the new garden. But which of these is the best way to capture and communicate your ideas? By itself, each of these approaches has its own particular strength; however if you combine them in a coherent way you will transfer a much richer depth and breadth of knowledge about the subject of the new garden. The same thinking applies when communicating the requirements for software systems—something else that also has intangible and functional considerations. To explain all the details, dimensions and viewpoints of the requirements you need a

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

combination of models, pictures and text. You also need some way of connecting them.

Natural Language Pitfalls

Natural language is the most common way of writing requirements. We use natural language to specify atomic requirements, to define goals, to write scenarios, to summarise levels of requirements. Natural language is known to everybody and is therefore common ground between the business-oriented stakeholders, and the technicians who are to build the final product. However, despite its convenience, there is a potential downside to using natural language.

Suppose you are working in a team where everyone has the same native language, and let's also suppose that your language is English. Your fellow team members have grown up speaking English and have learnt its grammar (more or less) and its vocabulary since they were small children. It is therefore reasonable to expect that everyone in the team can use English to communicate the details of their work, on the assumption they will understand each other. But, and this is a big "but", does everyone understand the language in exactly the same way? The answer is obviously no—you know this if a colleague has ever misunderstood you when you are certain that you have spoken or written what you consider to be a perfectly accurate and complete description. When you look someone in the eye and use words whose meaning you think is shared, and that person nods in agreement, you still run the risk that your correspondent's mental model of meaning could be different from your own.

Figure 1 is an illustration, from the work of family therapist Virginia Satir, that illustrates the difficulty of successfully communicating an idea from one human brain to another and having both parties end up with an identical interpretation of the idea.

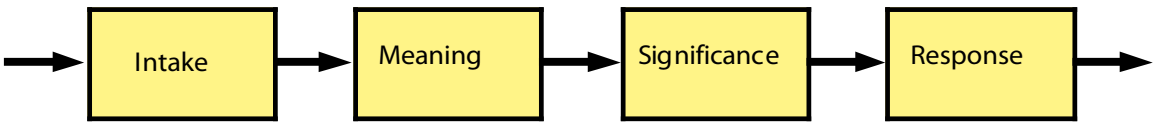

Figure 1: Virginia Satir's interaction model illustrates the different stages of thinking when people observe and take action.

During *Intake* someone receives data. They then make their own interpretation of the *Meaning* and the particular *Significance* to that person's experience and attitude. In the final stage the person makes his or her own

Response. Keep in mind that we are not consciously aware of this progression, but it is useful for us to discuss the problems of communicating using natural language. During the Intake activity, noise in the form of expectations, body language, culture, and other factors can interfere with the receiver actually hearing or reading the intended message. The Interpretation is affected by culture and experience, both influencing the message the receiver is formulating in his or her mind. The Significance depends on the receiver's priorities at the time, and how much importance the receiver attaches to the incoming message. Finally, the response is formulated taking into account the three preceding activities, and by this stage the intended meaning of the original message could be significantly corrupted.

We also have to consider the difficulty of using natural language to define the scope of the problem so that everyone involved interprets it to be exactly the same. A textual description of a business area cannot possibly hope to be precise enough to differentiate which functionality is in or out of the scope.

Natural language is often the best way to define atomic requirements. However, if you have thousands of atomic requirements then you cannot manage them without some intermediate levels of detail. You need some unambiguous way of defining each level of detail and navigating between them.

It turns out that all these problems of misinterpretation, scope definition and levels of detail can all be addressed by defining requirements using a combination of models, natural language and informal sketches or pictures.

A Cornucopia of Models

There are a number of models that we can use to specify requirements: models that focus on processing, others that are more concerned with data and other that concentrate on state. We have business process models, UML activity diagrams, swim lane diagrams, class diagrams, entity relationship models, state transition diagrams, SYSML diagrams.....the list is practically endless. Given all the options, it is beneficial for you to consider which models are most useful (in conjunction with natural language) to help you specify the requirements in your environment.

Defining the Scope

As already discussed, you need a way of defining the scope of your study so that everyone involved makes the same interpretation of it. At the highest level you need to identify what functionality is included, and what is specifically excluded. While functionality is hard to describe or model, data

is easy. Any function must processes its input data and produce its output data, so by modelling the data and material that enters and leaves your scope of study, you unambiguously define the included functionality. Also for each input and each output, you should specify where it is coming from or where it is going. An easy way of doing this is to draw a work context diagram as illustrated in Figure 2. This example comes from the Library Loans case study that we have used in previous articles in this series.

The arrows indicate data and material that is coming into the work and the squares (often referred to as adjacent systems) indicate where the inputs and outputs come from and go to. The circle containing the work indicates the functionality that you are to study, and hopefully improve, by writing requirements for your eventual product. This work context diagram defines the highest level of the requirements. You can make this model more acceptable to people who are not used to models if you do two things: use terminology that is familiar to the people with whom you are trying to communicate; and draw the model interactively.

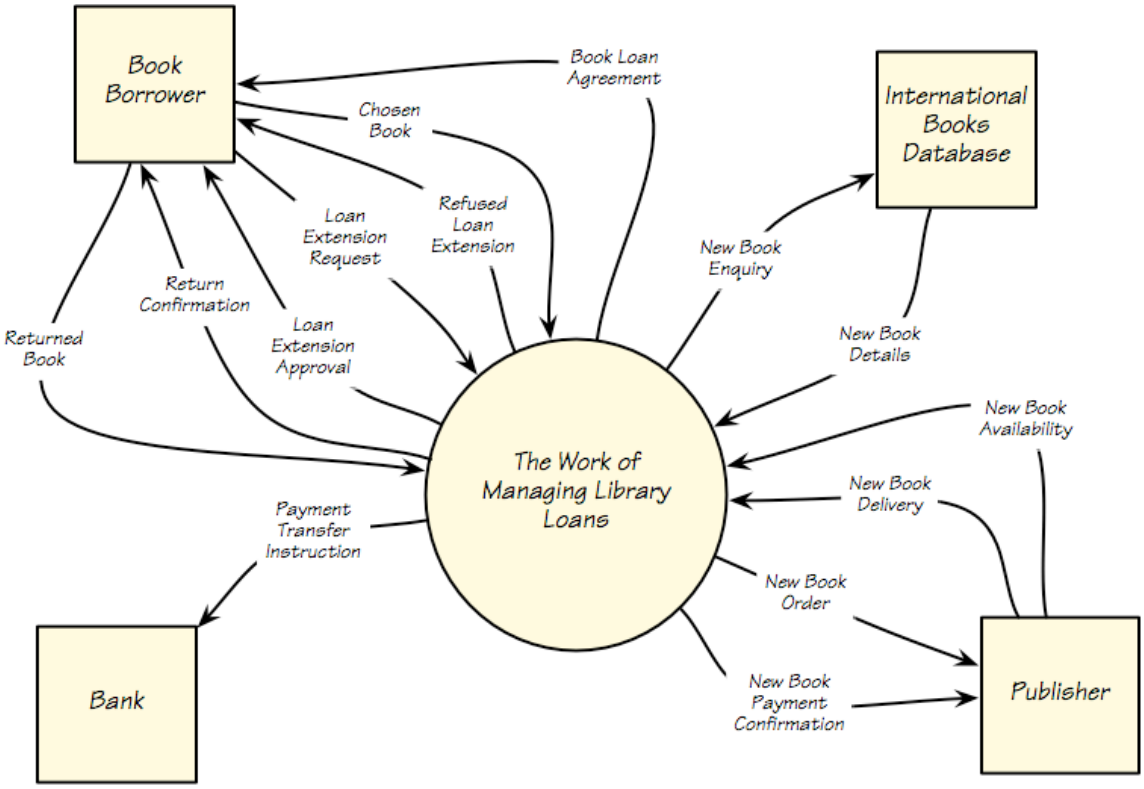

Figure 2. This work context diagram defines the scope of the work. Your investigation of this work means that you come to understand it, and eventually improve it by building an automated product. It is for this product that you write the atomic requirements.

Of course the work context diagram can only be unambiguous if each of the flows has a single, and well-defined meaning. This then is the role of the *data dictionary*.

Data Dictionary Provides the Glue

The data dictionary is a repository for the meaning of each of the inputs and outputs on your work context diagram. Let's look at the input Loan Extension Request that comes from the borrower into the work. You can have a conversation about Loan Extension Request and have a reasonable idea of what details it might contain. But to be sure that everyone has the same understanding you define the meaning in your data dictionary:

Loan Extension Request – an input from the borrower containing
Borrower Id
+ Book ISBN
+ Book Title
+ Extension Request Date

Then, to go to the most detailed level you define, in the data dictionary, each of the components of the input, for example:

Book ISBN – a unique identifier for a book, assigned by the agency responsible for ISBN in each particular country.

By carefully defining the content, and thereby the meaning, of the data that is input and output to the work, you are building a common understanding of precisely what the data is and through that, what the functionality of the work is. When you reach the stage of writing the atomic requirements, you continue to use the terms that you have defined in your data dictionary.

Back to Natural Language

As your study of the work progresses, you reach the stage of determining the product that will best improve that work, and writing the atomic requirements for it. Natural language is usually the most convenient way to write your requirements, but it is important to keep using the terms that you have defined in your data dictionary. For example, here is an atomic requirement that you have discovered. This uses the usual Volere conventions:

Description: Determine if the borrower has any outstanding book loans.

Rationale: We do not grant a **Loan Extension Request** if the borrower has any loans that are currently overdue.

Fit Criterion: Find all the loans for this **Borrower Id** where the **Loan Expiry Date** is before today's date.

Notice that the requirement uses terms that are defined in the data dictionary. And to complete the loop, the terms that are used in the requirements are identical to those used on the work context diagram.

Last Word

In this article we have discussed the pitfalls of exclusively using natural language to define requirements. Similar pitfalls exist if you limit yourself exclusively to the use of models. The idea is to combine the use of models and natural language and to link them by employing the formalism of a data dictionary.

Future articles in this series will discuss how to use other types of models to define other abstractions and levels of requirements.

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 – third edition*.
- in Volere seminars and consulting
- on the Volere Requirements Linked In group http://www.linkedin.com/e/vgh/2491512/

Previous articles in this series 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

