Requirements Engineering

An Interview with Ian Alexander at REFSQ 2010

Dear Readers, welcome to the IT-Radar interview taking place at "REFSQ 2010 – The International Working Conference on Requirements Engineering: Foundation for Software Quality." We talk with Ian Alexander, an independent consultant and trainer, specialized in Requirements Engineering. He is lead author of a number of books in this discipline and the most recent one is titled "Discovering Requirements". Ian Alexander is currently chairman of the BCS Requirements Engineering Specialist Group.

When we visualize Software Development as a big puzzle – where do the Requirements Engineering pieces belong to?
Intuitively, we assume that Requirements Engineering takes place all up front in the beginning, but as your question implies this is not correct. Requirements Engineering can be done iteratively, you can have a rapid prototyping or agile life cycle which involves a bit of requirements, design, testing, demonstration and a bit of usage as well which results in another bit of requirements. In many ways, this is more likely to work because we give a better overview to the users and stakeholders what the requirements are and what their implications will be in practice.

Is Requirements Engineering still part of the puzzle "Software Development", or is it somehow an independent puzzle?
I think it is part of bigger pictures. Requirements are only useful if they are followed by design and implementation. In fact, the purpose of requirements is to guide design, testing and documentation to ensure that the work meets the customer's demands.

When we talk about stages of the Requirements Engineering process, how many stages do generally exist and do we have to observe rules of order precisely?
There are certain things that probably should always be done to a certain degree. It should always be obvious, who the stakeholders are you get requirements from and whose views need to be represented. Additionally, you should always know what those stakeholders want, eventually you need to know the scenarios of usage and there-

fore should be imagined future scenarios be done to find out how the requirements will fit together. We probably always need some definitions and priorities but we certainly always need to measure our requirements in order to verify and test them.

Your recent research was concerned with Requirements Discovery, which is apparently the first stage of Requirements Engineering. Is this stage terminated in the beginning or does it have to be repeated during the process of Requirements Engineering?
Fundamentally, discovery has to be accomplished at the beginning, but if you have a staged life cycle, work iteratively or agile, there will be opportunities for discovering requirements later. Nevertheless, even if discovery work is carried out precisely, some requirements will still be found during the process. With change, development and people's expectations growing, new requirements will continually arise, in any case.

How should Requirements Discovery be accomplished successfully?
It should be done flexibly because requirements certainly vary in different industries. Actually they critically vary with the projects size, time and money available and critically how novel the project is.

What is most challenging in Requirements Discovery?
We can make out many things that are practical challenges. The difficulty in industry is that people do not have time and sometimes do not take time to practice Requirements Discovery, because it appears to be simple. People think that this is easy because they simply have to write down some words in natural languages, like English or German. All these facts can be responsible for producing weak, ambiguous and incomplete requirements which are poor specifications. To avoid this, research provides a wide range of techniques that could improve Requirements Discovery fundamentally and could be adopted by industry, if it were clear how different techniques can be consolidated, which is not the case yet.

Assuming that Requirements Engineering is domain-specific because, for example, safety critical domains and application systems underlie different preconditions, do you believe that this has to be considered in Requirements Engineering?
The techniques of safety and security analysis are discovery techniques in those specialized areas. Hence, to a certain degree we know that discovery varies with the type of project, but some tasks should always be fulfilled, like finding out the stakeholder's goals, modeling scenarios or making the requirements verifiable. Other techniques, such as stakeholder analysis, seem to be much more useful in some kinds of projects than others. Therefore, my intuition is that there are things you always need to know, but the amount of what you need to know varies widely.

As far as we know you train people in Requirements Engineering. Do you teach standard techniques that should be acquired generally to do Requirements Engineering successfully?
The first dimension of the problem is that there is a battery of techniques to model the elements of Requirements Engineering. There are questions of modeling the stakeholders, goals and scenarios as well as defining measurements or doing prioritization. The other dimension is to find out how to answer these questions in particular contexts. Sometimes people think requirements is all about going to speak to one or two individuals, interview the expert and the domain specialist or talk to a user, but few requirements can be found by talking

to individuals. For other requirements you actually need to have people in the context of a system, in order to discover what is wrong. In cases where prototypes exist, it can be discovered what people like about it and what could be improved. Other aspects, like scenarios, are much more easily discovered in workshops or in groups, than from individuals. Therefore, the requirements engineer needs good interviewing skills, knowledge of facilitating a workshop, be able to find unconsidered facts and noticing or resolving conflicts. Quit a wide range of people skills are needed.

Do you believe universities impart those skills to students? If yes, could it be improved or are universities already prepared for that problem?
It certainly varies very widely in different universities, but many universities teach Software Engineering or programming courses with very little content of soft skills. Some treat "domains" entirely as a technical matter, simply design the code and at least test if it works. It is important to learn various analysis skills which can be applied using the Unified Modeling Language (UML), but that is still a long way from doing Requirements Discovery it seems to me.

Switching to tool support in Requirements Engineering, DOORS[1] still leads the field. Why is that so?
That is because there is a good match between DOORSes capability and the real engineering and human problem. DOORS can cover any number of documents of any size with any number of relationships between them, with any number of attributes documenting each item or object. The human and engineering problem is that people have different opinions, which have to be processed through various stages and documented for software or system developers in a way they can eventually work on. Accordingly, there is not just one large body of information, but many large bodies of information with complex relationships of different types and therefore a lot of description is needed in texts, graphics or tables. DOORS provides all these capabilities, what few of the other tools really do.

Thinking about tool support, alternative tools might be Wikis, Feature Trackers, Ticketing-Systems etc. These tools are increasingly used for Requirements Engineering. Do you consider it a good idea?
I think, they are being used in different corners of requirements. I suspect that people who plan to do requirements engineering, do not use them. Among people who do not intend to do requirements and believe they simply document the project's needs or the customer's outline, have a more agile view and Wikis have taken off to some degree. They provide some of the features needed for Requirements Engineering, for example they record the time, so we can have some kind of tracking of history. We normally do not have any Baselining, so the change control is weak. So called Wiki wars, where people undo each others changes, need to be avoided. Wikis do not provide mechanisms for converging ideas and generally, tools like Wikipedia provide users with additional mechanisms, like discussion groups or chat rooms, to converge ideas. Probably, a combination of these new tools should be used and controlled to achieve good requirement results.

Do you believe there are domains where Wikis or Feature Trackers etc. are especially useful?
I suspect that at the smaller and more informal end of the market it is a very possible approach, but when things are mission- or safety critical, it

[1]Editor's note: DOORS („Dynamic Object Oriented Requirements System") is a requirement tracking tool offered by the Sweeden-based company Telelogic AB.

should not be handled like that. For a little piece of business software within an organization, it can probably work very well

Talking about Requirements Discovery again: do we have adequate tools supporting this special field as well?
First of all, a lot of requirements discovery is concerned also with soft skills, human interaction, pencil and paper, flip board and pens, white boards, blankets or pieces of colored felt, furthermore with acting skills, theatrical improvisation, video and photography. These things play a certain role, but you cannot expect to find the perfect tool. There are tools and I have even made tools which do things like defining dictionaries given very simple inputs, but tooling is a relatively small part of discovery. When we move over to the latest stages of requirement analysis in management, tooling becomes more important. A wide range of analysis tools already exist and data base like DOORS manage predefined requirements very effectively, but to discover and define them, is rather a soft task.

In industrial projects we have experienced that people commonly use no specialized tool for Requirements Engineering, but rather spread-sheets or word-processing tools. Do you consider this a good alternative to specialized tools or are these tools misused?
This is potentially dangerous because people know how to edit simple documents in WORD or EXCEL and because of the fact that requirements are made of words they believe that they can be put into such a document, which is potentially very dangerous. Firstly, there are very poor facilities for traceability between documents. Secondly, there are very limited facilities for identifying individual requirements which is one of the most basic things that a database does, which is to assign each record in the database to a unique identifier and DOORS can do that as well. In WORD or EXCEL you do not really even get that. You can put a number or something to the text, but it can be easily separated from the text to which it is attached. At first sight, simple tools look attractive, but fail in supporting requirements properly.

Is there a topic that we have not yet covered and might be important for our readers?
I think that many of the new tools enable to handle requirement in more open way. Older tools tend to provide a proprietary database and sometimes predefined formats, whereas DOORS is highly customizable, rather than dictating which format you must use. Oftentimes, new tools work on a server which can be anywhere in the world allowing people in different parts of the world, time zones and from different companies to log in securely and revise documents or models edited by other people. Therefore, these tools support collaboration and discovery, what older tools do not do.

Do you believe that tools are the most important topic to regard when we think about Global Software Engineering, Off-Shoring or Off-Sourcing?
It is also about people, organizations and the policies people have. How do companies with different standards of Software Engineering work together, when people are used to different document formats or write in different languages? Some people are used to codes, others to designs, or to various customer domains. We have big cultural gaps, intellectually and socially, and bridging these gaps is a human and managerial task. I think this is far more important than any technical issue.

Is Requirements Discovery especially challenging when people with different cultural background work together because, for example, they make different implicit assumptions?
It certainly makes it considerably more difficult and relying on groupware to get over problems of culture will not succeed. People have to be collocated. Therefore, people have to go to India if projects are outsourced to Bangalore, for example. It will be necessary for either team to visit the other team in order to work with them, to study the local issues and find out how to handle them. Human managerial contacts are absolutely vital to make distributed projects work.

That is certainly true for projects building systems for different markets. What about cases where steps of developing processes are outsourced? For example, an implementation is done in India, but the system is built for the US market and Requirements Engineering is done in the USA, will this procedure work out?
Maybe it might work out, but it depends upon whether the Indian developers understand the assumptions and explicit requirements, and they have been imparted to them. If the assumptions are unstated and the requirements are expressed in terms of these unstated assumptions which are rooted in the American culture, then the question emerges, on which basis the Indian developer can be able to understand the requirement. Either he has been traveling to the USA himself, which is often unlikely, or he has watched American television which may or may not have given him a realistic idea of what goes on there. This is potentially a large gap to overcome and I recall an Indian developer working for a supermarket, who had to cope with words like “aisle”. When foreigners look up the word “aisle” they find it defined as “a side passage in the church”, which might not make it any easier for them to guess what is supposed to be happening in a supermarket. Unless foreign developers have some domain knowledge and unless they have a very good command of the language, they may have difficulties in interpreting requirements.

In contrast, if a program or system is built for multiple markets focused on different customers, like a software product line, are there techniques helping to stay on focus and enables that variability will be identified as variability and common features will be identified as common?
This is a huge question. Meanwhile, there are whole conferences only on Product-Line-Engineering, but requirements engineering for different market areas involves a little bit of market testing, survey work and prototyping, hence there is a long way of discovering whether things are going to work. There was a car called NOVA, which sounds to an English or German person as if it means NEW, which was the intended meaning, but the Spanish people interpreted it as NO VA - meaning “does not go”, and the car became an object of fun. Cultural assumptions vary in different parts of the world, and that is why people may simply read things differently or interpret colors positively or negatively. Unwittingly choosing the colors of death and funerals of another culture or using unusual screen layouts or printing methods is decisive for the acceptability or unattractiveness of a product. These things have to be approached very carefully and sensitively.

Ian, thank you for this interesting interview!

Redaktion: Michaela Trebing und Katharina König