Requirements Quarterly

The Newsletter of the Requirements Engineering Specialist Group of the British Computer Society

© 2008 RESG http://www.resg.org.uk RQ49 (October 2008)

Contents

RE-soundings

From the Editor

I was once told that if volunteering for anything, be sure to take over when things are going well. Thanks to Ian I have inherited a newsletter that is going very well indeed, and I hope that with your continued support and contributions we can maintain this high standard. Regular readers will be pleased to see that Ian continues to be involved in his role as RESG Chair.

I first became involved in the world of requirements as a user involved in submarine sea trials and acceptance, and sponsoring improvements to in-service Command Systems. Those were the days when modifications could be made by persuading the person who actually did the work, often with the encouragement of a bottle of scotch. Not an approach found in many configuration management plans, but it did open my eyes to the benefits of getting the requirements right in the first place!

After completing an MSc during 2000 (and joining RESG!) I spent my last uniformed job within Defence Procurement as a 'proper' Requirements Manager. I enjoyed it so much that I have been plying a trade as a requirements engineer since then, primarily working with stakeholder requirements.

This 49th edition of RQ has been relatively easy to compile, primarily through the efforts of others. Ian shares his diary from RE08 in Barcelona – his comprehensive review offers a useful resume of the trends and issues affecting our profession to-day.

Another feature of this edition is an entertaining look at traceability by Dr Olly Gotel of Pace University. Traceability is one of the value-adding benefits of requirements engineering, enabling the full impact and risk of changes in context, need or design to be fully assessed. But as Olly points out it tends not to receive the attention it deserves, and she offers some realistic ideas to improve your approach to traceability.

And finally I include what I claim to be the longest review of the shortest publication in the history of RQ.

I shall be satisfied if no-one notices that RQ has a new editor - Enjoy!

Simon Hutton, Headmark Analysis

Chairman's Message

Hallo everyone, and welcome to RQ and to the new-look RESG committee. Simon Hutton is the new editor of RQ, bringing a wealth of practical expertise from his consultancy, not to mention his time in the Navy and his stint editing the INCOSE newsletter, so we could hardly have a better pair of hands on deck here.

RE'08 has just gone by, in sunny Barcelona. It was an exceptionally lively conference in a vibrant city. The RE community was buzzing with new ideas, notably for teaching but also for research; both the tutorials and the workshops, including REET (that's Education and Training) which we co-sponsored, were well attended and a lot of fun, too.

We are planning a diverse mixture of events for the coming year, not only in London but around the country. We hope to see you at some of them. If you have ideas for future events, or would like to help bring an event down your way, let any of us on the committee know, and we will do our best to serve you.

Ian Alexander, RESG Chair

RE-treats

For further details of all RESG events, see www.resg.org.uk

RESG - Ubiquitous Requirements

2.00pm, 15th October 2008.

Walton Hall Campus, The Open University

(http://www3.open.ac.uk/contact/maps/wh-campus.gif),

Computers are becoming ubiquitous and so are the requirements for the applications that they run. To date, related areas of ubiquitous, pervasive and ambient computing have been technology-led. The technology is maturing very fast, however, and we are starting to see real applications.

But is RE ready? Can we, for example, understand, model and reason about requirements for context-sensitivity? Do our methods and tools allow us to cope with systems that are self-adapting to changes of context? This event will explore what is special about "ubicom" for which RE needs to find the answers. To explore this question, we have invited talks by two of the leading researchers in ubiquitous computing. We hope to make the event highly interactive so that the audience can posit questions and challenge the speakers and, we hope, come away with a better understanding of what challenges emerging ubiquitous technologies pose the RE.

We hope you will join us for what promises to be a fascinating few hours. To help us organize venue and refreshments, please email Pete Sawyer to let us know you'll be coming:

sawyer@comp.lancs.ac.uk.

The event will be free to RESG members and £5 for everyone else.

Yvonne Rogers is professor of Human-Computer Interaction in the Computing Department at the Open University and directs the Pervasive Interaction Lab. She is also a visiting professor of Informatics at Indiana University (where I was from 2003-2006). From 1992-2003 I was at the former School of Cognitive and Computing Sciences at Sussex University. I have also spent sabbaticals at Apple, Stanford University, University California San Diego, and the University of Queensland.

Prof. Rogers researches and teaches in the areas of HCI, ubiquitous computing and CSCW. A particular focus is augmenting and extending everyday learning and work activities with novel technologies. This involves designing enhanced and engaging user experiences through using a diversity of technologies, including mobile, wireless, handheld and pervasive computing.

Gerd Korteum is a Senior Lecturer in the Computing Department at Lancaster University. He does research in the areas of ubiquitous computing, collaborative computing and business computing and is the principle investigator of the NEMO project, an EPSRC-funded collaboration between Lancaster's Departments of Computing, Management Science and Psychology, which has the goal to investigate ubiquitous computing technologies for industrial workplaces.

He obtained his PhD in 2002 from the University of Oregon with a dissertation on development support for applications supporting wearable communities. He also holds an MSc from the University of Oregon and a Diplom in Computer Science from Stuttgart University.

Prior to coming to Lancaster he held several positions in the software industry and at various research labs. From 2000 – 2002 he was leading the development efforts at Internet start-up Livingnetworks. Before that he developed mobile computing solutions at Apple Computer's Advanced Technology Group in Cupertino, USA, wrote 3D architectural software for Artifice Inc., researched knowledge-based systems at both the (now defunct) IBM Science Center in Germany and Technical University of Berlin, and developed production management software at the Fraunhofer Institute for Industrial Engineering in Stuttgart, Germany.

IIBA UK - October Event

IIBA™ United Kingdom Chapter

Wed 29th October 2008
Registration from 6:00pm

Balls Brothers Minster Pavement Restaurant,
Minster Court, Mincing Lane,
London EC3R 7PP

The International Institute of Business Analysis (IIBA) is an independent non-profit professional association serving the growing field of Business Analysis. The **IIBA UK Chapter** mission is to promote and develop the profession of business analysis and communicate standards for best practice within the UK.

The next UK IIBA event on 29th October includes a presentation on Joined-up Requirements: Business Goals to System Tests by John Cheesman**.** John is a Principal Consultant with Strata Software Ltd, and specialises in business analysis, software specification and process-improvement, with a particular focus on pragmatic tool support.

Getting test requirements right is a difficult business. This is partly due to the fact that the business, analysts and testers have different perspectives on requirements and how they should be organised and mapped, and partly due to the fact that requirements change so the organisation, mappings are priorities are constantly changing.
This presentation summarises an approach to structuring requirements which provides traceability between the business objectives of a project all the way through to the detailed test requirements on the software. This traceability structure enables multi-level project status checks, prioritisation and scope management, and provides a basis for risk-based testing and agile application development.
The approach is illustrated with a case study from a major UK bank using Compuware's requirements management tool (Optimal Trace) and HP's test management tool (Test Director).
Further details and registration information can be found on the IIBA UK Chapter web site at http://**uk.theiiba.org**

RESG - Making your requirements knowledge count - Working in RE

2-5pm, 5th November 2008

The Pavilion Room, Westminster Uni, London

Maybe you are thinking about becoming a requirements analyst. Do you know what kind of life can you expect? What your biggest joys and deepest sorrows will be? How you will spend your days? What skills you will need? Are the things that are taught in universities and described in books actually useful? During this event, several practising requirements analysts will tell you what their job is really like.

Contact Emanuel Letier, University College London

RESG - PhD Student Event

December 2008

London

Details and date to be confirmed, but the basic outline is to have a series of student talks in the morning and afternoon discussions with academics. Lunch will be provided, and it is hoped that the best student talk will be awarded a prize. Details will be available at: www.resg.org.uk/events.html, or Contact Dalal Alrajeh, Imperial College

International Workshop on Requirements Analysis

6 & 7 December 2008, London

More than 70% of the IT projects in the UK fail every year and over 80% of them fail in the requirements analysis phase. A number of methods have been developed as a response to this problem without causing significant improvement to failure rates.

Inadequate or non-existent models, approaches and methodologies are not the only cause of failure at the requirement analysis stage. Poor project management, bad organizational politics, false business priorities, lack of commitment and other issues can also cause a project to fail. However, most IT experts do not use requirements analysis approaches and methods, and are often unfamiliar with them, so that the UML diagrams that the minority of designers are using can appear alien to business stakeholders.

The aim of this workshop is to bring together relevant academics, researchers and industry experts to discuss potentials solutions to the problem of poor requirements analysis. Professionals are invited to share their experience on conducting requirements analysis in the IT industry. Discussion on unified methodologies or approaches will be particularly welcomed, with a view to jointly developing a unified approach. Details at **http://palab.dcs.kcl.ac.uk/iwra/**

RESG - Creativity Tutorial

March 2009, London

Further details on the planned Creativity Tutorial scheduled for March 2009 will be announced when the date and venue are confirmed. Contact Neil Maiden.

RE-member

RESG AGM 2008

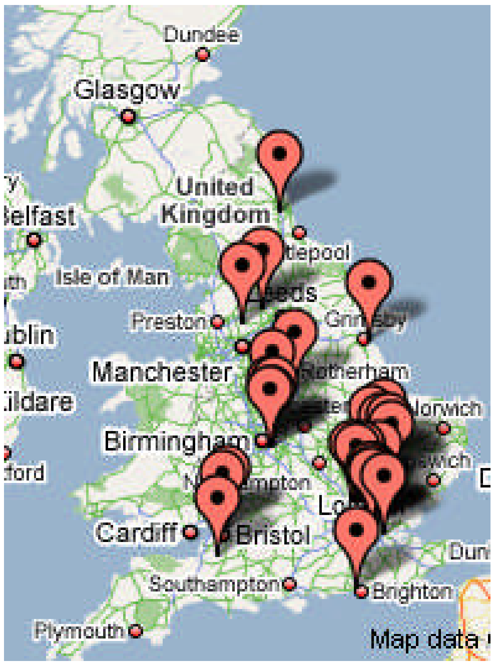
The RESG Annual General Meeting was held at Imperial College, London, on 10th July 2008. Although only attended by a small number of the membership, it was a great opportunity for the new members of the committee to be introduced. Ian and James were there to discuss their roles as Chair and Secretary, and even your new editor managed to venture into the capital, without getting lost!

Pete Sawyer reviewed his tenure as Chair, and in reflective mood looked at events, successes and some lessons learnt. One even detected a small tear, although I am not sure if that was because he realised he will continue to be heavily involved as Past-Chair and by providing welcome support to the Newsletter!

Pete Sawyer delivers his final State of the Nation

Yijun Yu, our membership secretary, gave a short overview of where we are with the membership, especially with recent changes to the mechanism for joining the Group. All members must be BCS members, and can opt for affiliation to any specialist group. As a result, the membership jumped from 168 last year to 320 this year – a warm welcome to all!

Yijun also shared some of his analysis of the membership figures, looking at e-mail addresses, titles and geographical dispersion. The majority (90%) are from UK, but we do have members from all over the world, including Canada, Australia, Sri Lanka and Tokwawan (I had to google it too – I think it is in Hong Kong!). It would be a pleasure to hear about the experiences of our global community in future editions of RQ.



RESG Membership – UK By Postcode

Of course, no AGM report can ignore the hospitality that is a feature of these events. The wine and beer flowed freely, the buffet was a real feast, with platters bearing the grand names of Summer Sensation, Taste of India and Vegetarian Vitality. And last, but by no means least, the string quartet that serenaded us with a range of classics from Brahms to Danny Boy.

Elana, Rosie, Beccy and Maddie entertain us in style!

The AGM is a rare opportunity to chat to fellow practitioners and researchers, renew old or discover new acquaintances, and to be a part of an active and sociable group. If you only attend one event over the next year, I would certainly recommend the AGM!

Simon Hutton, Headmark Analysis

RE-writings

Traceability – Problems in a Word

Dr Olly Gotel, Pace University

ogotel@pace.edu

Traceability is a topic that has been discussed and written about within software engineering circles for many years. While there has been much progress since the so-called traceability problem was studied in the early 1990s [1], traceability is still perceived to be problematic today [2]. Make a casual mention of the word ‘traceability’ when amongst practitioners or researchers and you’ll hear a few sighs. The topic is one that most of us prefer to sweep under the proverbial carpet.

I was asked to speak on traceability and its issues in software development as a panelist at RE’06 (The International Requirements Engineering Conference). I used that forum as an opportunity to air, in a somewhat light-hearted manner, some of the interrelated reasons for the continued challenges in this area. The talk itself was an excellent example of one of the main issues that compounds traceability – if no permanent record is created, what is there to subsequently trace? Traceability often comes down to people and their fading memories.

So, drawing upon my now faded memory, I have written down the intentionally provocative reasons I suggested as to why traceability problems linger. It is not meant to be a rigorous academic treatment of the topic; rather, it is more an aide-mémoire for practitioners, a mnemonic of important issues to consider when putting traceability into practice. Each issue also presents unique and interesting challenges for researchers to get their teeth into.

T **Traceability is perceived as TEDIOUS.** Discovering the requirements for a system can be difficult, but it is usually interesting work. When undertaking such a task, you interact with others as an explorer of the requirements space – you are a ‘requirenaut’[1]! Writing the requirements in a testable manner, then structuring them as an organized whole, is clearly an intellectual activity. It is now widely recognized how critical our descriptions are to the quality of all the development work that follows. Working with the requirements to design a system is also an engaging activity that taxes the grey matter. First-rate designers get respect. However, linking chunks of information content to establish traceability is (quite frankly) dull work and universally regarded as such. Who wants to be setting up traceability when they’d rather be exploring or designing? The inevitable upshot is that any task that is considered monotonous to undertake is put to the very end of the to-do list and is liable to result in errors when eventually undertaken. I would be so bold as to claim that the quality of the resulting traceability graph is invariably associated with the mood and boredom threshold of the person(s) doing the task. Worse, we often only find out about the quality of the traceability days, months or even years later when it is finally needed. Traceability is either in need of a serious image makeover or we have to make it invisible. Negative perceptions can cripple all good traceability intentions on a project. It is an issue to be tackled head-on.

R **RESPONSIBILILITY for traceability is blurred.** Ask someone whose responsibility is it to establish the traceability on a project and they will probably use the words of Douglas Adams – it is ‘Somebody Else’s Problem’[2] [5]. While traceability needs it champions, it is unlikely to scale in the hands of an individual, unless you are talking about a one-person project. When traceability is assumed to be the shared responsibility of all team members, as something extra they are expected to do as part of their regular job, what do you (realistically) expect to see achieved? Firstly, do we reward those who do a superb job with setting up traceability and penalize those who jeopardize its potential?

[1] A little bit of history about this term: When the Requirements Engineering Specialist Group of the British Computer Society was established in 1994, the first committee brainstormed potential titles for its flagship newsletter. It arrived at the title ‘Requirenautics Quarterly’ (see issue 1 [3]). The term ‘requirenaut’ was proposed to reflect the fact that requirements engineers need to explore an unknown and seemingly unbounded requirements space, and the newsletter was created to share resources to help requirements engineers in this pursuit. The newsletter was renamed ‘Requirements Quarterly’ in 2005 (see issue 36 [4]). The subtle change was intended as a simplification, but perhaps it was also a consequence of untraceable rationale being locked away in the fading memories of a few individuals?

[2] “An SEP is something we can't see, or don't see, or our brain doesn't let us see, because we think that it's somebody else's problem.... The brain just edits it out, it's like a blind spot. If you look at it directly you won't see it unless you know precisely what it is. Your only hope is to catch it by surprise out of the corner of your eye.” [5]

Secondly, are they all on the same page anyway? To distribute the responsibility and move towards scale, we need to establish a framework for all to work within, and ensure agreed roles and responsibilities are undertaken. But, if we demand exacting processes and the policing of activities, the negative perceptions mentioned earlier are simply going to explode. We could learn much from how traceability is achieved in the food industry here [6]. End-to-end traceability is achieved via a simple guiding policy that all parties subscribe to. It affords the freedom to implement traceability in independent ways so long as it complies with a standard protocol. They have distilled the essence of the traceability need in their industry, and they share both the implementation responsibility and risk of failure across the entire food chain.

A **The ARTIFACTS to trace do not come pre-packaged.** We deal with many different types of artifact in software development (e.g., interview transcripts, UML models, code, etc.) The information content is therefore commonly represented in a variety of media (e.g., natural language text, diagrams, sound, etc.), and at differing levels of formality and granularity. Traceability thus needs to account for information chunks in many formats and at different levels of abstraction. Unfortunately, these artifacts are not always so neatly structured and packaged within an integrated environment to enable tracing. It is all to easy to set off on a traceability venture without paying sufficient prior attention to what exactly may need to be rendered traceable and in what ways. How are trace relations to be established between radically disparate artifact types anyway? The term 'traceability meta-model' may populate the academic literature [7], but it is not clear how pervasive the use of such a concept is in practice. We also talk about media-based traceability in the literature [8], but many practitioners are still stuck trying to do a good enough job with plain old text. We need to step back and start by thinking about the materials we are likely to be dealing with and how these can be integrated, interrelated and handled. Elaborate schemes may be overkill, but a little bit of prior thought will go a long way.

C **The CREDIBILITY of traceability can be debatable.** How do we know whether we can trust the results of any traceability provided? What is the coverage of the traceability graph, how up to date is it, what is included and what is excluded? If we lack confidence in the traceability we are unlikely to use it and very unlikely to maintain it. What metrics do we have to tell us about the quality of the traceability, for individual traceability relations, extended traceability paths and for a project as a whole? What do we do to monitor and convey these measures on a project? The whole point of this enabling mechanism we call 'traceability' is to help people to make more informed decisions. We can't just establish traceability on a project and expect everything to be perfect. We need to find a way to communicate confidence levels so that people can understand the basis upon which they are taking their decisions, so think about this quality dimension when instituting traceability. You are going to need it to help you with the unavoidable issue that follows below …

E **Traceability ENTROPIES.** Artifacts normally evolve as a project progresses and as learning takes place. Any traceability relations that have been established therefore have a finite shelf life and the overall traceability graph tends towards entropy without dedicated ongoing maintenance. Given it is so burdensome to put traceability in place initially, one would hope that the investment be sustained. Rather, it is more likely that the traceability is left to decay. Who is responsible for traceability maintenance anyway? Who would want to be? A more significant problem is, given that we rarely monitor the quality of the traceability, who knows if we are taking important decisions based upon data that is far from credible? Maintenance has always been the very last thought in software engineering and consequently incurs immense cost. Why should it be any different with traceability? Regrettably, this is an issue that complicates any attempt to understand the return on investment from putting traceability in place. If you are going to bother with traceability, then don't bother if you are not going to ensure the results of your endeavors remain timely. Plan your maintenance strategy well, because the researchers have got a long way to go before they find a way to make traceability maintenance fully automatic.

A **Unrealistic expectations are placed on traceability AUTOMATION.** The traceability promise associated with automated tools and techniques is seductive. Many organizations purchase requirements management tools to store their traceability artifacts and relations, but these tools don't help with any of the issues mentioned above … yet. You still need to do all that thinking and planning. The ability to recover traceability relations automatically may be the current Holy Grail, but such techniques still demand a high quality set of artifacts to begin with and the manual filtering of results. Rather than relying on an elusive point solution, we need to place more emphasis upon traceability strategy, figuring out how to blend heterogeneous automated and human approaches, and leveraging tools to best advantage.

B **Traceability should be a BY-PRODUCT, but it just gets in the way.** Traceability should not be the goal of software development. We are in the business of delivering systems that address customer needs, and everything else we do in software development supports this primary goal. Traceability really needs to be achieved as a by-product of other engineering activities, rendering it invisible for all extents and purposes. When we develop our software engineering practices and techniques such that support for traceability is built in as a natural by-product of other tasks, we will make progress with the traceability problem. We cannot stop what is perceived to be the 'real work' on a Friday afternoon to ensure the traceability is put in place on a project – it has to be integral and built in from Monday morning.

I **Ambitious INTENTIONS for traceability go unproven.** Traceability is expected and even claimed to support many development-related tasks (e.g., impact analysis, validation and verification, regression testing, trade-off analysis, change management, etc.), so the intended users and use for traceability is extremely broad. However, very little serious attention has been paid to understanding the traceability stakeholders and their tasks, and even less attention has been paid to gathering data to validate how well their goals are actually being met by whatever traceability is put in place. This is a significant issue and a somewhat embarrassing one to highlight within a community that prides itself on 'doing' requirements. We must do our requirements for traceability, keep on doing our requirements and follow through to see how well we are actually satisfying them.

L **Little sharing of traceability LESSONS.** This issue is the obvious result of the point made above. Where are the traceability exemplars and case studies that people can learn from? What works well and what doesn't work well, and in what contexts? Do software engineers take traceability training prior to embarking on a project? Do they consult a little red book? We are a long way from instituting best practices in traceability because we really don't know what these are. Or, if we do know what these are, we certainly don't share them. There are professional certifications for testing, quality assurance and project management, so why not a competence for traceability? Given that traceability is so essential for each of these three disciplines, this situation is surprising. While the research community busies itself with sophisticated proposals, very simple benchmark practices go unshared. This is a plea for practitioners to contribute to a body of practical knowledge for others to build upon.

I **Traceability breaks-down with the slightest INTERRUPT.** It is inevitable that there will be gaps in the traceability record due to missing artifacts and missing relations. Rather than stumble at these junctures, we need to find ways to deal with a broken traceability graph and to work with incompleteness, inconsistency and inaccuracy. The World Wide Web works despite broken links as the search algorithms have been designed to work with imperfection. We need to pay as much (if not arguably more) attention to how we can handle problems with the traceability graph as we do for building it in the first place. If you are planning on an ideal world, plan again.

T **We can't trace the TACIT 'stuff'.** Even if we can create an environment in which every communicative exchange is recorded, we would not be privy to tacit information, those private thoughts that go unsaid and are the basis for hidden assumptions and rationale[3]. Exhaustive tracing demands a way to incorporate the very edges of the information network, but attempting to do this would be a Sisyphean task. A large part of the record in software development will always remain intangible and so not available for traceability purposes. One way to address this issue is to provide links into the social structure that contributed to the tangible artifacts and forged the relations. Admittedly we may be relying on faded memories again, but at least this would provide a way to track down those memory holders. Related to the issue with interrupted traces, we need to think beyond tangible artifacts when setting up traceability and consider the necessary fallbacks.

Y **Traceability? You aren't gonna need it (YAGNI)!** A practice of eXtreme Programming is to only implement that

[3] Science fiction writers would have us believe otherwise, so we should certainly keep an open mind for the future!

functionality that is needed for the current project iteration and not to put scaffolding in place for the future. Named after the justification – 'You Aren't Gonna Need It' [9]. When it comes to traceability, there can be a YAGNI feeling amongst software practitioners. Perhaps this arises when elaborate traceability systems are proposed, with huge demands on people's time, with no conception as to when and how the results will actually be used? This is further understandable when the cost / benefit equation is largely unknown, especially for those individuals doing the work. With this issue we come full circle. We are not going to make progress with addressing yet another human perception unless we make progress with many of the issues that are outlined above.

Practitioners – many of the challenges you will face in putting traceability into practice are hidden in the very letters of 'TRACEABILITY' itself. Figure out how you plan to address each of these twelve issues on your projects:

- **T**edious – how will you overcome this image?
- **R**esponsibility – whose is it?
- **A**rtifacts – what are they?
- **C**redibility – how will you determine this?
- **E**ntropy – how will you deal with it?
- **A**utomation – do you recognize its limits?
- **B**y-product – what do you expect people to do?
- **I**ntentions – who needs it and why?
- **L**essons – are you capitalizing on these?
- **I**nterrupts – what is your contingency plan?
- **T**acit – how will you reclaim some of this stuff?
- **Y**AGNI – how will you convince others you need it and it is worth it?

Researchers – there are lots of underlying and interrelated reasons as to why traceability problems linger. To escape traceability Catch-22 we demand systems solutions from you.

References

[1] Gotel, O.C.Z. and Finkelstein, A.C.W. An Analysis of the Requirements Traceability Problem. In *Proceedings of the 1st IEEE International Conference on Requirements Engineering*, IEEE Computer Society Press, Colorado Springs, CO (April 1994), pp.94-101.

[2] Cleland Huang, J., Dekhtyar, A. and Huffman Hayes, J. (Eds.) *Grand Challenges in Traceability*. Center of Excellence for Traceability Technical Report COET-GCT-06-01, University of Kentucky, September 2006.

[3] *Requirenautics Quarterly: The Newsletter of the Requirements Engineering Specialist Group of the British Computer Society*. Issue 1 (October 1994).

[4] *Requirements Quarterly: The Newsletter of the Requirements Engineering Specialist Group of the British Computer Society*. RQ36 (June 2005).

[5] Adams, D. *Life, the Universe and Everything*, Pan Macmillan, 1982. See also: http://en.wikipedia.org/wiki/Somebody_Else's_Problem (accessed August 2008).

[6] Gotel, O.C.Z. and Morris, S.J. From farm to fork or a bite of the unknown: Learning from the food industry. In *Proceedings of the International Symposium on Grand Challenges in Traceability (GCT'07), Traceability in Emerging Forms of Software Engineering*. Lexington, Kentucky, 22-23 March 2007.

[7] Ramesh, B. and Jarke, M. Toward Reference Models for Requirements Traceability. *IEEE Transactions on Software Engineering, 27, 1* (January 2001), pp.58-93.

[8] Gotel, O.C.Z. and Morris, S.J. Macro-Level Traceability via Media Transformations. In *Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'08)*. Montpellier, France, 16-17 June 2008.

[9] *Extreme Programming Practices: You Aren't Gonna Need It.* See: http://c2.com/cgi/wiki?YouArentGonnaNeedIt (accessed August 2008).

RE-flections

Requirements Engineering 2008 Barcelona

Ian Alexander, Scenario Plus

Ian Alexander shares his diary from RE08 in Barcelona, giving a fascinating overview of the key events and a flavour of the key issues in Requirements Engineering.

Monday

I gave a tutorial on **Rationale Modelling**, a neglected Cinderella sitting in a dark corner of Requirements practice. It was clear that, despite the varied backgrounds of the audience, everyone agreed that rationale was scarcely being captured on many projects.

Some projects (about 1 in 6, from this very small non-random survey) do model their goals; rather more model scenarios in some form; and most make some use of traceability, which documents rationale at least implicitly.

Everybody enjoyed trying their hand at modelling rationale, making use of a remarkably wide range of techniques on the way. We all agreed that rationale analysis ought to follow work on identifying stakeholders, analysing goals and documenting scenarios; the presentations made by the three teams variously included storyboards, lists of stakeholders, goal models, lists of assumptions and graphical rationale models.

At lunch the SOCCER team (**Service-Oriented computing**…) was discussing services of different types, and the need to be clear on the difference between software services and business services implemented using software. The former were definitely a solution mechanism; the latter certainly closer to the problem to be solved.

In the afternoon the **Requirements Visualization** workshop run by Brian Berenbach of Siemens and Olly Gotel of SUNY looked at ways of visualizing requirements, and then moved into a discussion phase, before splitting into working groups to brainstorm ideas for future work. The two working groups were given different agendas, but both came up with quite similar results: attributes by which requirements visualizations could be described, and ultimately measured and compared. For instance, a visualization could have purpose = validation, notation = X-Y graph and so on. I didn't write much as the workshop was fully hands-on, but the findings will be written up by the workshop organizers.

Flamenco Dancers and Spanish Guitars at the Conference Banquet

Tuesday

The **Requirements Engineering Education and Training** (REET) workshop resumed for the third year running.

Tony Gorschek from Blekinge Institute of Technology, Sweden reflected on what was taught and whether it was the right stuff. He said it was hard because he saw industry "doing a really bad job of requirements", and in the last ten years he had hardly seen any improvements. Should he be teaching languages like Z, he asked rhetorically, or trying to address the problem?

Ljerka Beus-Dukic of the University of Westminster was teaching requirements to third-year students, so they had at least had some experience from their projects of identifying requirements, developing software from them, and then getting feedback on how well the software met the requirements. You couldn't just pick up and write down the requirements; they had to be approached slowly in small steps – identifying the stakeholders, writing scenarios and so on. These were the essential elements in building up practical skill in requirements work. Students were taught to approach each element through a discovery cycle, so they learnt the place of documenting and validating as well as the various discovery techniques. All of this makes for a very different style of course for the students: it is not a matter of teaching one modelling technique after another.

G. Auriol of INSA in Toulouse, France said it was hard to get students to see the need for managing requirements, as their projects were generally so small that they could hold all the knowledge in their heads. He gave the students the task of building a scale model radio-controlled boat, conforming to the very strict rules of the IOM racing class. If they made the mast strut too short – through machining it down, and so on – then they had to lower the deck to avoid creating a hole which would admit water. Then they found they had an illegally large clearance between the boom and the deck. They could solve this by shortening the mast. Perhaps this is a practical answer to the challenge of presenting requirements concepts to inexperienced students: you start from design, and through that you move into design constraints, and then into conflicts created by such constraints. And perhaps this hints at the solution to the pedagogic problem posed by RE: you have to ditch the formal presentations, and gently coax the students into thinking about the issues by grappling with small but realistic problems. Even if those are design problems.

Raymond Barnes of Binghampton University, NY, USA spoke about (un)Knowability, and agreed with previous speakers that the critical competency gaps in engineering students included the ability to structure problems, rather than just applying specific modelling techniques. Students had difficulties with problem solving, being mature enough to take the initiative, and leadership, as you might expect. So, the task is to move the students out of the easy and comfortable path. Classroom techniques offered included using Toulmin argumentation, assessing medical-style levels of evidence (randomized controlled trials at one end, expert opinion at the other), and "humble decision-making" as an attitude that might be helpful.

Salvador Dali with Dolphin

S. Faily of the Computing Lab, University of Oxford, spoke about teaching RE practices to "end-users", ie software developers, working at ESOC. They had had bad experiences with CASE tools back in the 1990s (or maybe earlier): some of them had been working in the domain for 40 years, ie far longer than the time that Software Engineering (let alone RE) has existed; and they had their own language for software development concepts as well as for the domain (space and spacecraft). They were therefore quite intolerant of novelties such as UML and use cases, for instance. The use of a Wiki however took off well, with the developers building up (and thus taking ownership of) the Wiki pages to cover guidelines with examples on different topics.

Sascha Konrad of Siemens spoke about the use of pedagogical patterns for effective teaching in industry, where the constraints (as Faily implied) are different and in some ways stricter than those in universities. Reinforcement (a pattern) in particular is essential, or people soon forget a lesson; this can be applied through repetition via questioning, interactive exercises, feedback on the exercises and so on.

Barbara Paech (Heidelberg University) spoke about RE as a profession, and in particular the development of certification, including the preparation of a syllabus and an exam. The exam is a multiple-choice test. The preparation for it is meant to take a 3-day course. A book covering the syllabus is being written. The International RE Board (IREB) is a non-profit venture with a freely available syllabus, developed by volunteers, unlike some certification bandwagons. The exam fees go mainly to the licensed certification authorities, who of course have the work of marking the papers. The training community is not accredited: anyone can use it, and the market can determine if they are any good. Only if trainers want to appear on the IREB website do they have to agree to follow the syllabus. There is some overlap with the IIBA's syllabus, which is however focused on one domain, and which tries to define process in fine and immediately-usable detail, whereas the IREB believes that applicability to any domain implies the need to adapt and customise the basic material.

Zhi Jin (Peking University) spoke about balancing academic and industrial needs in RE courses. The materials need to integrate RE methods with processes in the software life-cycle; and to be suitable for examination by a combination of exam, project, and interview (viva).

Mike Alexander (Seilevel) spoke about using games to teach RE. It is essential to explain why games are used.

- *Q&A&Q* has a participant write an answer to a question written at the top of a sheet of paper. They then fold the paper over to hide the question. The next participant writes a question to match the visible answer, and folds the paper over to hide the answer. The next participant writes an answer, and so on. Sometimes the result is the same question repeated many times down the sheet; sometimes there is wild variation. The game can show something about how definite requirement knowledge is or isn't: perhaps everyone has the same question for some answers ("1:1 interviews").

- *ReGo* (RE-Bingo) gives the ability to test a little knowledge (you can only get a few words in a 4x4 grid on a single sheet of paper) in the format of a bingo game, in a more acceptable way than a formal test. You can use it to test before and after a course is taught.

- *Guess What We Want* has the students draw a picture of a solution, based on a spoken list of requirements. We had a lot of laughs playing this game. It demonstrated the variety of real-world responses, variability, misunderstandings, the effect of adding more requirements (sudden changes), beliefs about design, the need for different types of requirement, and more.

Tailoring games to learning objectives is vital: and can be hard work. People have to be willing to play the game. You have to reinforce the game – people have to see why they did it, and realise what they learnt, or it's no use. The balance of lecture and games is a challenge: 50/50 might be right.

The final session of REET was devoted to trying out some activities and then working on how these could be documented, perhaps on a Wiki – ie there would be some metadata to describe each one. It was remarkable how varied the activities were: a facilitated prioritisation-by-voting exercise; the presentation of a suitable case study for a master's course; a goal clustering exercise; NFR graphical modelling and analysis (a la i*); conjectural explanations of why an airplane owner says his plane has been in the repair shop for 3 days, but the mechanic states the plane has only been in the repair shop for 1 day. The workshop created 8 plausible reasons for this! The students who had been given this activity had uniformly felt that the reason was obviously that the mechanic was a liar. Perhaps the workshop participants were more experienced as engineers, or just more worldly wise.

Wednesday

In the morning two of the RESG committee (Mav and me) went and had a look at Barcelona's Barrio Gothico, a most beautiful quarter of this wonderful city between the mountains and the sea.

Clustered Columns, Stone Vaults, and Wrought Ironwork in Barcelona Cathedral

Antonio Monzon of EADS spoke on requirements reuse in product families of on-board systems. A basic concept is the (oddly-named) Derived Requirement, which means one that is derived from the design (of the next level up of system), rather than as usual derived by tracing to the requirements of the next level up. This kind of requirement turns out to be important in reuse. In the FITS product family there are 300 requirements documents in a repository, from a total of 11 projects. The documents include interface control documents (ICD) containing interface elements, requirements documents containing requirements, V&V documents containing V&V actions, and change requests (ECR) containing change proposals. There is a typical cascade from high to low level system requirements, and then down to high and low level software requirements, with satisfaction traces going back up all the way. References can be made within documents.

For reuse, the common requirements are identified from legacy projects and promoted to the product family (PF) master (for each type of document just mentioned) in the repository. Then the PF master is reused in new products.

Reuse can be strong (by synchronizing the product with the master) or weak (by copying). An average project has 31 documents, each with about 500 requirements and about 500 other paragraphs. The reuse rate is higher for Derived Requirements. It takes some man-months to reuse each document, but always less than the effort to start afresh.

Reuse rates are about 40-60% - higher for lower-level requirements, especially if those are derived. Since lower-level requirements are also more numerous, the savings are also larger in absolute terms at the lower levels.

Frank Houdek of Daimler asked if some types such as HMI requirements were harder to reuse: Monzon said no, but he would look at it.

Ian Alexander asked if they had found that requirements had to be specially instrumented for reuse, by pulling out definitions, rationale, comments, parameters and so on. Monzon said yes, it was necessary, and an effort was under way to improve the requirements in this way. Airbus in particular had been very helpful in pushing for better requirement structure, making the requirements "designed for reuse".

Dominik Schmitz of the German ministry for education and research spoke about requirements for engine control systems; in Germany, these are mainly produced in small and medium enterprises. Such systems are strongly dependent on hardware, including the processing power and memory available in the controller. Different control systems were therefore not easy to reuse. But for proposals – quick specifications, in effect! – there are two possible strategies: try to capture requirements faster; or try to reuse requirements better.

The i* goal modelling approach is model-based; it captures both functional and non-functional aspects; and it permits combined, inter-disciplinary (eg hardware and software) investigation. The i* model can be used to represent both the intentions of the stakeholders (inside large circles) and the design options available (with a set of implemented-by links for each item, eg petrol or diesel as fuel options). Then when writing a proposal, if diesel is not an option, the model can quickly be simplified just by striking out the boxes and lines that are not relevant.

If reusable artefacts cannot be found (in time) then the proposal will contain too many new artefacts, and will be too expensive (so the proposal will be rejected). Or, if the artefacts found are in fact not reusable, the project will make a loss as it will have to develop new ones when it didn't expect to do so.

An i* repository with a query language makes it easier to locate options for items, eg the number of cylinders, the position of cylinders (V versus in-line, etc) and the fuel used. Queries return results with a percentage similarity, so the projects are ranked for the engineer writing the proposal.

Antonio Monzon asked why he had used i* rather than UML or SysML. Schmitz replied that even SysML was heavily text-based for requirements; it offered none of the advantages of i*. i* is quite simple – it just offers stakeholders and goals; you don't have to go into the mathematical detail of say MatLab where each block of a diagram has equations associated with it.

Alistair Mavin of Rolls-Royce spoke on using scenarios to discover jet engine control system requirements. There is a harsh environment with widely varying temperatures. There are many sensors, often dual or triple with voting to handle failures; the controller too has dual channels, leading to up to 100,000 lines of code. ART-SCENE relies on people being better when prompted, so it asks what-if questions based on the structure of already-defined scenarios. Actions have types such as communication; agents have types such as system. What-if questions too have types, such as "What if x is malfunctioning?"

ART-SCENE was specialized to the aero engine domain by identifying relevant classes of failure. Many were similar to traditional HAZOP safety threats, but HAZOP did not cover all the classes, so the list was extended. The output was a long list of "alternative course" (i.e. exception) events, e.g. "What if the temperature signal between temperature sensor and control system is missing?" all generated mechanically.

A Variable Stator Vane (VSV) scenario was chosen as a specimen. Four events were picked to represent a range of types. These were examined as usual against HAZOP threats. The same scenario and events were put into ART-SCENE. The (25-years) experienced safety engineer took 4 hours, producing about 20 recommendations (from which requirements could be inferred, with additional work by a different engineer) per event. E.g. possible loss of temperature signal led to a recommendation to detect the loss and find another way to estimate or measure the temperature. Using ART-SCENE, a systems engineer took 3 hours and found 30 requirements – with more detail – directly. Further, the systems engineer used the prompting to go further, and consider further alternatives within the situations suggested by the tool.

Clearly the burden of safety analysis – hours for just a few events in just one scenario – is very large for a complex system. If some of this burden can be relieved it should be well worth while. In addition, the tool encourages multi-disciplinary review, involving systems engineers in safety analysis. Further extension of the tool to suit it better to aero engines would probably make it more useful in many other engineering areas as well. For example the tool ought to consider multiple failures, not just one (a major change to how it works); and it should consider all life-cycle phases, not just normal operations. A graphical front end would make it much friendlier to use, too.

http://www.rolls-royce.com/education/schools/default.jsp allows interested readers to take a virtual tour through a jet engine.

Masks in a Shop Window in the Barrio Gothico

Thursday

Ivan Jureta, John Mylopoulos and Stephane Faulkner won the Best Paper Award for Revisiting the Core Ontology and Problem in RE – essentially taking another look at the famous paper by Pamela Zave and Michael Jackson a decade ago.

David Callele and colleagues won the Best Poster Award. The poster was on their work on emotional requirements for video games – terrifying the player is *de rigueur*.

Michael Jackson (of the Open University, and our Patron) gave the Keynote Talk on Problems, Solutions, and Requirements. (He gave the Keynote 13 years ago at RE'95in York, too.)

"RE is a new label attached to an old problem that has been with the SW profession since its inception" – Michael Harrison and Pamela Zave – foreword to RE'95.

David Caminer of Lyons tea-shops was, said Jackson, the first requirements engineer. Lyons started building their own computers (!) called Leo. Leo 1 first ran in 1951, and went fully live for the staff payroll in 1954.

Is RE like other engineering?

Social, clinical, financial, psychological, food, fashion, electoral, political engineering: these terms are each on the cover said "*engineering is the design and construction of any artefact which transforms of a journal! But these terms are too general: engineering is just "getting what you want*". But we're interested in something more established. CFC Rogers the physical world around us to meet some recognised need." E.g. the 1992 Toyota Camry is a pre-computerisation era car which moves people and baggage on good roads, protects from weather and in a collision, using portable power, and so on. Artefacts include cars, ships, disk drives, power plants, arch bridges, dams, aircraft, bullet trains, chemical plants. Jackson suggested that "our kind of engineering" is like this. "Our artefacts" include software, software, software, software, and software, he joked. Perhaps software people have not realised the importance of specialising. But requirements people know better. Our artefact is the system-machine + world + requirement, suggested Jackson, referring to his Problem Frames. There are phenomena shared between machine and problem world, and then (even further out) there are requirements connecting the real world to the problem world.

(This seems to be a new strand in Jackson's thinking. I would like to imagine that having Jackson write the foreword to my forthcoming book had the side-effect of attracting his attention to this new angle on requirements: requirements are not the intersection of the domain and the machine's specification, but something closer to people and the world.)

What can go wrong? How can we fail?

There are many ways we can fail.

We can misunderstand the requirements. E.g. Does the same priority scheduling apply to all lifts?

We can misunderstand the problem world. E.g. Do # call requests at floor F = # requesting users at floor F?

The machine can fail to ensure the requirement. E.g. On reversing car direction all outstanding requests are lost.

Development can totally fail, making an unusable system. E.g. multiple unreliabilities and knock-on effects.

How do we learn from other engineers?

Software engineering, said Jackson, has ignored progress in all other engineering disciplines for 40 years. Other engineers mainly carry out normal design, once radical innovation is over.

Established engineering branches use maths, science, technology, production control, and so on – they are all important. But crucially, learning from experience. And you can only learn from experience if a) you are specialised; and b) you move into normal design, so you can gently improve your own practice.

Walter Vincenti is an academic aeronautical engineer. He wrote the absolutely indispensable book "*What engineers know and how they know it*". He defines radical design as work where engineers largely do not know how the device should be arranged or even how it works. The designer has never seen such a device before, and has no presumption of success. The problem is to design something that will function well enough to warrant further development.

Benz's 1886 car only had 3 wheels, because he didn't know how to solve the problem of steering: the inner wheel has to lean in more than the outer wheel to track around the curve. So, he began with radical design. After this initial success, normal engineering took over. The driver always faces forwards, sitting in the front. There are always 4 wheels, and so on. The design has a standard structure, and engineers can specialised to improve efficiency in many different areas. By 1888 cars had 4 wheels. By 1910 the cab was closed. By 1919, 4-wheel brakes. By 1933 independent front suspension. By 1940, automatic gearbox. By 1954, tubeless tyres. By 1956, disk brakes. By 1976, fuel injection. Normal design had totally taken over; everyone agrees the names of the component types – fuel lines, radiator hose, catalytic convertor: this is a crucial symptom of normal design. This allows engineers to learn to avoid failures.

Henry Petroski, in his Design Paradigms, says "engineering advances by proactive and reactive failure analysis, and at the heart of the engineering method is an understanding of failure in all its real and imagined manifestations."

Jackson illustrated the engineering understanding of failures with the Comet 1 airliner, and the Tacoma Narrows bridge. The Comet had rectangular windows, and metal fatigue created cracks from the window corners. The bridge was exceptionally long for its width (72 times the width), which was 40% more than the Golden Gate bridge which already had the greatest span/width ratio of any normally designed bridge. The insurer argued that the roadway should be widened as it was outside the parameters of normal design. The engineers replied they had done their calculations: but these only covered horizontal oscillation. They had failed to consider vertical oscillation, which broke the bridge.

Specialisation and Artefacts

It is necessary for engineers to have Artefact-specific knowledge. Aero engineers do not just have general principles to avoid fatigue; they have an iron rule not to have any sharp corners to apertures in aircraft fuselages. Specialisation gives an enormous advantage to normal design.

Vincenti gives some wonderful examples of specialisation of knowledge around artefacts. One is the design of a wooden aircraft propeller (airscrew) between 1915 and 1926. Another is the development of flush (smooth, aerodynamic, load-bearing) riveting of metal aircraft skin in the 1930s. Given that the skin was often only 1mm thick, the rivets had to have conical heads, and the machines to fasten them had to be specially made for aircraft manufacture. No other branch of engineering knew how to carry out the task.

It takes a village to produce a child, says the African proverb. It takes more than a village to produce an artefact. There are elaborate specialisations for civil engineering including soil mechanics, for instance. Nuclear, mining, petroleum, railway, structural, mechanical, chemical, electronic, electrical power engineering each have their own knowledge, journals, skills, training.

Component structure in software-intensive systems

Capers Jones in Computer magazine, July 1995 listed specialisations in software engineering. Cost estimating, human factors, configuration management and so forth: but they are all general to all projects!

They select parts of the same process. All right, admitted Jackson, there are some artefact specialisations: Internet or LAN infrastructure; compilers, model checkers and other tools; GUIs, databases and other universal components in “system” software. And indeed there is some specialisation in application artefacts: medical systems, lift software and so on. But not enough. This he illustrated with the Therac-25 medical disaster, which fatally irradiated its patients. Or Ariane-5 which had an inappropriate rocket control system. Or the London Ambulance System which had poor user interfaces and an inaccurate vehicle location system.

Lessons have to be attached to components, specifically and precisely. The parts have to be the same as in other systems, or you can’t fasten lessons to them. So, what are the components in software-intensive systems? Requirements (responses to stimuli)? Use Cases? Desired System Properties?

Jackson suggested that a component must be a sub-problem or system function: the ability to manage book loans, the ability of a lift to brake on danger, when a fault is found. He illustrated the problem decomposition with (sub)problem frames. For instance, a sub-problem is to make a model of the lift gear, to help detect faults. Another model is to run the lift display and buttons. These components are crucially (he argued) specialised sub-problems. Editing the rules for service priority is a *work-piece* problem: the building manager decides what priorities he would like, and the output has to be a correctly edited text. Modelling the lift gear is a *dynamic model building* problem. This kind of problem has to obey constraints, has to be initialised, has to diagnose faults, and so on. In other words, each class of sub-problem has its own special skills and techniques, like an established engineering discipline.

In the Therac-25, it was possible for the human operator to set the radiation amount without seeing any change on the screen! Even the excellent formal report on the accident did not draw attention to this critical fact, because it was not obviously associated with any design component that the authors could identify. Jackson said it was clearly a *work-piece* problem: the operator edited the text.

Manfred Broy told the tale of the “improved” parking brake control of a famous German car-maker. The parking brake was to be released automatically when the driver depressed the accelerator pedal. A test driver stopped the car, applied the parking brake, and got out to open the factory doors. The car started driving towards him. The warm air from the inside of the factory had triggered the air conditioning to run harder. This demanded more engine power, so the controller activated the accelerator. This released the handbrake… Of course, WE would like to imagine that we would not have allowed this in our requirements. It isn’t true, said Jackson. There are too many interactions, it is too tricky. The only answer is normal design. Brake designers just need to *know* they must measure the position of the accelerator pedal, not the power demand. Success depends on specific, artefact-related, experience built up over time and many different products.

“It may be engineering – but is it requirements?”

The extreme of pure RE is ab initio, free of all implementation bias, the prerequisite of design. It’s impossibly hard, assuming totally radical design. The other extreme is purely design-based: you want Dual Core 2GHz PC with XP Pro, 160 GB hard disk, 4GB RAM, DVD+R-DL, 24” 1920x1200 screen, WiFi 802.11n,… it is much easier. It is normal design.

Back in 1995 Jackson had spoken, he said, on Problems and Requirements. He had thought more or less pure RE. He had been young and naïve. Today he hopes he has moved rather closer towards the other end of the spectrum.

He said thank you to VIncenti, JE Gordon, Levy and Salvadori, Henry Petroski, Eugene S Ferguson and GFC Rogers, the wonderful authors who stimulated him to think in this direction. The details of their books are on the website.

Jian Ren (of Microsoft, working with colleagues at UCL) spoke on Fairness Analysis in Requirements Assignments. Which requirements should go in the next release? You try to minimise cost, maximise revenue. But is that enough? In particular, is it fair to your customers? People’s priorities obviously conflict. Experience and observation is not enough to achieve fairness. A multi-objective optimisation approach, adopted here for the first time, tries to address the issue of fairness objectively.

The model assumes multiple customers, N requirements each with a cost. Each customer assigns an importance value to each requirement. What then is fairness?

1) The number of fulfilled requirements to each customer should be the same
2) The value of fulfilled requirements should be the same for each customer
3) The investment of cost (effort) for each customer should be the same.

You can map solutions in a space with revenue on the X-axis, and cost on the Y-axis. Clearly, if solution A has lower cost AND higher revenue then it is better than solution B; but if A has higher cost AND higher revenue than B, then you can’t say if it is better or worse: it’s just different. The Pareto Front is the curve on the graph which is optimal: you have a set of points with the lowest attainable costs and highest attainable revenues. This could be a straight line but is more likely a convex curve. It is difficult to find this analytically, but using a genetic algorithm you can let a population evolve until you have seen many possible points on the graph, and the Pareto Front becomes visible when you plot them all. Given three axes (as numbered above) rather than just cost vs benefit, you need a 3-D graph.

(It isn't clear that fairness can be attained: you may need to choose a solution that is workable for various reasons, and that may force you to drop some requirements even if stakeholders want them very much. Perhaps in Microsoft software products, the requirements can be considered to be more or less independent; perhaps in other domains they are not?)

What if requirements conflict? Could conflict analysis be added? Ren said they were looking at it.

You can read more about the work in the SEBASE Repository, http://www.sebase.org.sbse/publications

Andrea Herrman and Maya Daneva (University of Twente, The Netherlands) spoke about the research agenda for prioritization based on benefit and cost prediction. They searched systematically for credible, original, relevant, cited papers on methods for prioritising individual requirements. The search turned up 240 papers.

Results include methods of Benefits estimation – but there are none on primary requirements. Business value is analysed subjectively, and at system level. For secondary requirements, there are risk-reduction approaches, and contribution-oriented approaches (ie, how far do the secondary requirements contribute to satisfying primary requirements). Size estimation methods look mainly at functional requirements; Cost estimation methods look only at functions: there are none for NFRs yet. Importance methods look at cost, at benefit, or both at once. There are 15 such methods documented.

These results suggest several research questions: how do functions relate to NFRs for priority? How should benefit estimation techniques from other areas be borrowed for requirements? How should secondary requirements be used to prioritise primary requirements? Can you prioritise NFRs quantitatively or by cost estimation? And what about dependencies?

Chih-Wei Ho of North Carolina State University spoke on the relationships between performance requirements and "not a problem" defect reports. Performance can be described by defining the subject, the measure, the environment, and the workload. In a case study, firmware for an embedded control module, a team of 80 programmers and 20 testers had 300 pages of requirements, containing 33 performance requirements. 1500 defects were found, of which only 3% related to performance, and a third were declared to be "not a problem" – for whatever reason. It takes, however, almost as long to investigate defects that are eventually declared to be "not a problem" as the others. 29 of the performance requirements were functional; 22 were quantitative (so 11 were not, in a real-time system). The only area which showed statistical significance was workload: all quantitative defects were confirmed as problems, whereas qualitative ones were half-and-half.

Eero Uusitalo of the Helsinki University of Technology spoke about linking requirements and testing in practice. He looked at 5 practices: early tester participation (in writing requirements); tester participation in requirement reviews; tracing tests to requirements; getting testers to meet requirement owners personally; and having testers suggest requirements.

These all have their pros and cons in terms of cost, people's time and availability, and benefits such as making requirements and tests better. Tracing, curiously, was not fully implemented on any of the half-dozen projects surveyed: the only practice which was, was having testers meet requirements owners. Perhaps this reflects a bias in the sample towards smaller, more agile projects, where the frequent iterations encouraged personal contact over tool use. Certainly the emphasis on getting testers involved early is very welcome.

Ian Alexander spoke about evaluating design options against requirements: in other words, trade-offs, using a statistical analysis method (Principal Components Analysis, PCA). Along with prioritisation and rationale – both covered by papers and tutorials in this conference – it was a neglected corner of RE. The paper can be found on his website (www.scenarioplus.org.uk).

Martin Feather of NASA spoke about guiding technical decisions by quantitative analysis: in other words, helping the team choose which projects to run by finding quick, cheap short-cuts to identifying the best ones. He said that the process still depended on humans, "Just as Ian Alexander's PCA leaves the final decision to humans".

The full treatment consisted of getting 10 people together for about 2 days to evaluate each proposal. Spending 20 man-days on each of 21 proposals was way over budget, even if people had been willing to spend so long on evaluation. So the goal was to find a way of picking the best in about the time it would take to do one single evaluation: a tall order.

Feather went about this by summarily rejecting 14 of the 21 proposals because they had no eager "Champion" to shepherd them through to success. For the rest, the team agreed to skip the usual scrutiny of the merits of the proposals: it was assumed that each was a worthy case. And they also skipped scrutiny of standard practices, which would probably not help to separate the sheep from the goats.

Instead, the effort was focused on listing the Obstacles to each proposal; scoring the damage each obstacle would probably do to the proposed mission; and then scoring the cost of the mitigating actions needed to overcome the obstacles. Feather then calculated all the combinations of these things, plotting the results to give a Pareto curve: number of obstacles remaining (on the Y-axis) versus cost to overcome the obstacles (on the X-axis). The curve naturally starts high up on the left, drops sharply with a well-chosen initial expenditure, and then falls off towards the right in further small steps. The optimal expenditure for any given amount of money is a point on the Pareto curve, which also indexes the specific actions to achieve the most bang for your buck. (Points above the curve are sub-optimal.) A "prudent" reduction on one project was to spend $2M to remove 80% of the obstacles, for instance.

In 6 of the 7 cases, the experts not only found near-optimal (Pareto) choices but also picked points on the Pareto curve that were very sensible – being at "corners" where spending a little less would sharply cut the benefit, and where spending a little more would bring little improvement. The 7th project's choices were optimisable.

The effort eventually expended was about 30 man-days: 21 projects evaluated for the price of one-and-a-half.

In conclusion…

As always at conferences, there was a huge wealth of ideas and experiences. What was really nice about Barcelona '08 was the freshness of the ideas for teaching RE – using all kinds of fun and games; and the attention finally being paid to real-world requirements problems like prioritisation, rationale, trade-offs and making requirements testable. All of these concern human skill and human choices; and if they don't have much to do with research into formal methods, modelling notations, and theorem provers, well, so much the better.

© Ian Alexander 2008

From the Horse's Mouth

What do requirement companies think of themselves, and what are they trying to offer? Ian Alexander interviewed the exhibitors at RE'08 to find out. Each was limited to a single page of his notepad.

José Manuel Muñoz of VISURE: We are a requirements company – our tool is IRQA – and offer services in RE. What is distinctive is our focus only on requirements, not selling a suite of products. So we interface with many other products and have an open API. And we are not just a tool but an Engineering tool, so we offer something that goes a step further than Requirements Management. For instance we support elicitation of requirements with Use Cases, concept models, testing via traceability. Engineers find it easy after just a two day course.

Gert Bikker of EXTESSY: "Experts for TESt and Systems Engineering". We are an integration company, because tools must be connected. For instance, DOORS and VISURE may need to work together. EXTESSY pioneers the implementation of the RIF, the Requirements Interchange Format, in its EXERPT (requirement interchange) tool. Its success is because our automotive and other customers need to interchange specifications without loss of structure. It works equally well in railway, financial, aerospace and other domains. The interface is open, the standard is open, so we believe we can work in any area. RIF is to become an OMG standard next year.

Gabriela Zornoza of TELELOGIC (an IBM company): Our tools are the best choice when you have complex projects, hierarchies of information, and it is critical to conform to customer requirements and standards. This is because we offer the best traceability – which makes the difference between our products and the rest. Ours is the best way to see information links between documents. Traceability is the key to doing requirements: where they come from, where they go. Our tools are easy to learn and to manage: DOORS for requirements life-cycle management; CHANGE and SYNERGY for configuration management: the three offer a complete integrated solution for upper level requirement down to lines-of-code traceability.

Enrique Castillo of POLAR CONSULTORES: We offer here a tool (iTestMan) which is 100% web-based, good for requirements management, covering all cases of system development from specification through to test, all kinds of problem-handling. You can customise workflow for your project. Problems such as fault reports and change requests trace to requirements and test results.

We always work on international projects, so having all functions on the Internet makes it specially suitable. E.g. email notification: when you log in you see all the problems that concern you – it's a fully automated workflow mechanism. It's very strong on reporting – both built-in and customisable. You can access iTestMan via Excel, MS Project, and so on. There is a real-time integration with DOORS and SYNERGY.

Roel Lucassen of SYNERGIO (with the TopTeam tool): Our primary focus is consultancy services with requirements. We feel we have a unique and innovative approach with requirements. We see this as a capability or skill, not a discipline. Requirements are everywhere. Based on the domain, we advise on the strategy to bring you success. We ask people to invest time in stating requirements as they'll define your end result; so, requirements are key to tracking and monitoring your projects.

"Start at the end" is our principle. From there, we help people plan their activities to meet those needs. We help to ensure projects balance requirements, work breakdown and product breakdown. We train people how to actually use requirements, which become part of the negotiating scheme between requestor and contractor.

© Ian Alexander 2008

RE-verberations

The Customer is Always Right – Sometimes!

Simon Hutton, Headmark Analysis

"Someone calling themselves a customer says they want something called service."

An article in the Times last month looked at ways of resurrecting Gordon Brown's premiership through the eyes of six 'experts'. One of the experts, Mary Portas, took a hard retail marketing point of view, as one would expect from a former creative director of Harvey Nichols and presenter of BBC2's *Mary Queen of Shops*. Comparing the Brown Brand with the washing machine department of John Lewis, she decided that the Brand needed energising – nothing about substance, more about packaging. What caught my eye was this comment made by Ms Portas:

> "The most successful brands are the ones that lead and don't explain themselves. When I'm re-branding businesses I never ask customers what they want, because they don't know. If you give them something, lead with it and tell them that it is fabulous, they follow."

Never ask customers what they want, because they don't know – something I'll bet you never thought you would read in RQ! Perhaps we should apply the Portas approach to requirements engineering. Don't bother with requirements – make something then convince the customers that it is the answer to their problems. Hopefully the cheque will clear before they discover the truth!

Unfortunately it is a common view, particularly in the fast moving consumer goods world. Marketing is not about discovering customer requirements before designing the product. It is about branding, packaging and advertising technology in the hope it will catch on and sell. Any requirements are gathered through sales figures and feedback from consumers, resulting in changes to the product.

There is some hope. Sir Richard Needham, Sir James Dyson's right hand man at Dyson, recently presented at a University of Northampton business networking and exhibition event. He explained the need for constant innovation, using failure as much as success as a valuable input to the innovation process. What was interesting was his take on understanding customer needs:

> "The most important lesson I have learnt is worry about what the customer wants. Too often in business you can think internally about what your organisation requires and don't put yourself in the shoes of the customer or your competitor"

Needham went on to illustrate these wise words with an example of when Dyson sold rights to their technology to an American company called Phantom. Phantom hid the unique design inside the machine, because they didn't want their customers to "see the muck and filth". But Dyson recognised that people did want to see what was being picked up, to satisfy themselves that the thing was working – it was a feature of the product that met a user need that only Dyson recognised at the time.

Needham points out that you have to go on innovating because the competition catches up. First it was the clear bin, then multi-cyclones, then they looked at manoeuvrability and switched to a ball. It is vital to constantly think through your design strategy and where you are going next.

Of course one could argue that this constant need for innovation is an example of what happens without good requirements at the start. The need to innovate to meet emerging customer demands is there because the customer needs were not clearly defined early enough to influence the design.

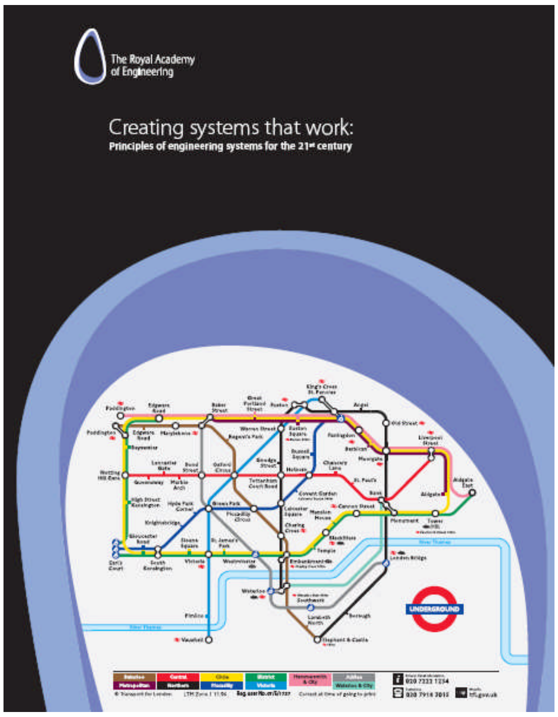
Innovation is also driven by the need to keep in front of the competition, and this is where we come back to the Portas view that customers don't know what they want until we put it in front of them. How many times has the requirements engineer hear "I don't know what I want, but I'll know when I see it"! The role of marketing is to convince the customer that they do have a need after all, they just didn't realise it.

Requirements through manipulation and persuasion – at least it would ensure some customer satisfaction!

Simon Hutton, Headmark Analysis

RE-readings

Creating Systems That Work


The Royal Academy of Engineering
Creating systems that work:
Principles of engineering systems for the 21st century

Principles of Engineering Systems for the 21st Century.

Royal Academy of Engineering, June 2007

I suspect it is not often that a book review considers a report that is freely available and only has 31 pages, but this report published by the Royal Academy of Engineering during 2007 does merit a read by anyone involved in requirements engineering. I would also recommend it as a quick read for a project manager or project sponsor, to give a broad overview of the benefits of discovering requirements early in any project lifecycle.

The RAE is Britain's national academy for engineering, and aims to promote excellence in the science, art and practice of engineering. This report was prompted by the recognition that the problems that engineering systems seek to solve are becoming more complex, and that it is not possible to design part of the system in isolation without considering the problem and its solution as a whole. Music to the ears of the requirements community!

Although written by committee – the Academy's 12 strong Working Party on Integrated System Design – the report is succinctly presented through the use of six principles for integrated system design:

1. Debate, define, revise and pursue the purpose.
2. Think Holistic.
3. Follow a systematic procedure.
4. Be creative.
5. Take account of the people.
6. Manage the project and the relationships.

Without doubt the first principle is of interest to the requirements engineering community, as the basis for the principle is that systems are created to satisfy a need, and as the expression of that need has to determine every step of the system's life it is essential to get it right. It is reassuring that the authors recognise that stakeholders rarely want a system – they want the ability to do something, and eliciting these requirements can be harder that engineering the system that meets them. I enjoyed the examples used to show that stakeholders usually think in terms of solutions – "they might specify four wheel drive when they want off-road capability, or antenna gain when what they want is reliable communications at a specified distance". A familiar issue with defining stakeholder requirements, but a pleasure to see it written down so clearly!

I even approve of the definition of a requirement:

> "A requirement is an unambiguous statement of the capability that the system must deliver. It is expressed in operational terms (what the system will do) rather than solutions (how the system will do it). The statement of requirement must also define how it is to be tested – if it can't be tested or measured, it isn't a requirement."

This first principle is developed further, albeit limited to a meagre 2 pages, to cover iteration, trade-off, and the need to constantly review requirements as context and priorities change over time.

The second principle to think holistically considers the benefits of looking across all parts of the system and along all of the time line. Anticipating what can go wrong is as important as planning what should go right, as is accommodating change. Few systems are built on a green field site which implies legacy, constraining possible solutions but also bringing experience and standards. Accommodating unforeseen future needs is hard to specify (*the system shall be upgradeable*!) but it is one of the requirements, and may prevent short-term thinking.

Another aspect of thinking holistically that is raised is the concept of understanding boundaries, and recognising that one of the first tasks when designing a system is to decide where to stop. Of course, this is an area where requirements really start to add value – helping designer understand where the boundaries lie.

Principle Three encourages us to follow a disciplined procedure, and the report uses a V-Model to illustrate the main features of a well-planned engineering life cycle. Although fairly traditional - no doubt the agile community will be reaching for their blackberries as they read – it does highlight the importance of understanding stakeholder needs, driving down to the functional requirements and the system architecture, maintaining coherence through iterative trade-off, and finally using the requirements as the basis for test and acceptance.

The fourth principle to be creative again emphasises the importance of working with the customer and other stakeholders to tease out and define the capability that the system must deliver, and to translate the top level design or system architecture into the requirements of each element.

The theme of requirements and the importance of defining and managing requirements through-life to enable successful system design runs through all six principles, and is a key feature of the useful examples used to illustrate the principles.

The introduction of a national MOT testing system is a great example of what can happen if the requirements are taken back to basics, and identify what is needed rather than how is should be delivered. Instead of implementing the system on existing equipment as originally required, the supplier provided new, basic computers and a dial-up system to share data. This bypassed legacy and interoperability problems, and whilst slow it was adequate to meet the actual requirements. And the marginal cost of the new PC was negligible compared to the costs that would have been spent solving integration issues.

Although the Principles of Engineering Systems for the 21st Century probably won't appear on any best-seller list, I would recommend you invest some time reading this report – it contains a lot of useful ideas and does support the use of requirements in successful projects.

And it is free!

The report can be downloaded from www.raeng.org.uk/education/vps/pdf/RAE_Systems_Report.pdf

Simon Hutton, Headmark Analysis

RE-partee

Predicting the Future – Or Maybe Not!

"Computers in the future may weigh no more than 1.5 tons." -- Popular Mechanics, forecasting the relentless march of science, 1949

"I think there is a world market for maybe five computers." -- Thomas Watson, chairman of IBM, 1943

"But what ... is it good for?" -- Engineer at the Advanced Computing Systems Division of IBM, 1968, commenting on the microchip.

"640K ought to be enough for anybody." -- Bill Gates, 1981

"I have travelled the length and breadth of this country and talked with the best people, and I can assure you that data processing is a fad that won't last out the year." -- The editor in charge of business books for Prentice Hall, 1957

"This 'telephone' has too many shortcomings to be seriously considered as a means of communication. The device is inherently of no value to us." -- Western Union internal memo, 1876.

"So we went to Atari and said, 'Hey, we've got this amazing thing, even built with some of your parts, and what do you think about funding us? Or we'll give it to you. We just want to do it. Pay our salary; we'll come work for you.' And they said, 'No.' So then we went to Hewlett-Packard, and they said, 'Hey, we don't need you. You haven't got through college yet.'" -- Steve Jobs on attempts to get Atari and H-P interested in his and Steve Wozniak's personal computer, shortly before they set up Apple Computer Inc.

"The wireless music box has no imaginable commercial value. Who would pay for a message sent to nobody in particular?" -- David Sarnoff's associates in response to his urgings for investment in the radio in the 1920s.

"There is no reason anyone would want a computer in their home." -- Ken Olson, president, chairman and founder of Digital Equipment Corp., 1977

RE-sources

Books, Papers

RQ archive at the RESG website:
http://www.resg.org.uk

Al Davis' bibliography of requirements papers:
http://www.uccs.edu/~adavis/reqbib.htm

Ian Alexander's archive of requirements book reviews:
http://easyweb.easynet.co.uk/~iany/reviews/reviews.htm

Scenario Plus – free tools and templates:
http://www.scenarioplus.org.uk

CREWS web site:
http://sunsite.informatik.rwth-aachen.de/CREWS/

Requirements Engineering, Student Newsletter:
www.cc.gatech.edu/computing/SW_Eng/resnews.html

IFIP Working Group 2.9 (Software RE):
http://www.cis.gsu.edu/~wrobinso/ifip2_9/

Requirements Engineering Journal (REJ):
http://rej.co.umist.ac.uk/

RE resource centre at UTS (Australia):
http://research.it.uts.edu.au/re/

Volere template:
http://www.volere.co.uk

DACS Gold Practices:
http://www.goldpractices.com/practices/mr/index.php

Software Requirements Engineering Articles (India):
http://www.requirements.in

Media Electronica

RESG Mailing List
http://www.resg.org.uk/mailing_list.html

RE-online
http://discuss.it.uts.edu.au/mailman/listinfo/re-online

ReQuirements Networking Group
www.requirementsnetwork.com

RE Yahoo Group
http://groups.yahoo.com/group/Requirements-Engineering/

RE-actors

The committee of the RESG

Patron:
Prof. Michael Jackson,
Independent Consultant,
jacksonma @ acm.org

Chair:
Ian Alexander,
Scenario plus,
iany @ scenarioplus.org .uk

Vice-chair:
Dr Kathy Maitland,
University of Central England,
Kathleen.Maitland @ uce.ac.uk

Treasurer:
Steve Armstrong,
The Open University,
S.Armstrong @ open.ac.uk

Secretary:
James Lockerbie
City University,
ac769 @ soi.city.ac.uk

Membership secretary:
Dr Yijun Yu
The Open University
Y.Yu @ open.ac.uk

Publicity officer:
William Heaven
Imperial College,
wjh00 @ doc.ic.ac.uk

Newsletter editor:
Simon Hutton
Headmark Analysis Limited
simon.hutton @ headmark-analysis.co.uk

Newsletter reporter:
Ljerka Beus-Dukic
University of Westminster,
L.Beus-Dukic @ westminster.ac.uk

Student liaison:
Dalal Alrajeh
Imperial College
dalal.alrajeh @ imperial.ac.uk

Immediate past chair:
Dr Peter Sawyer,
Lancaster University,
sawyer@comp.lancs.ac.uk

Member without portfolio:
Prof. Bashar Nuseibeh
The Open University
B.Nuseibeh @ open.ac.uk

Member without portfolio:
Prof. Neil Maiden
Centre for HCI Design, City University,
N.A.M.Maiden @ city.ac.uk

Member without portfolio:
Emanuel Letier
University College London,
e.letier @ cs.ucl.ac.uk

Member without portfolio:
Sara Jones
City University,
saraj @ soi.city.ac.uk

Industrial liaison:
Suzanne Robertson,
Atlantic Systems Guild Ltd,
suzanne @ systemsguild.com

Industrial liaison:
Alistair Mavin
Rolls-Royce,
alistair.mavin @ rolls-royce.com

Industrial liaison:
Dr David Bush
NATS,
David.Bush @ nats.co.uk

Contributing to RQ

To contribute to RQ please send contributions to Simon Hutton (simon.hutton@headmark-analysis.co.uk). Submissions must be in electronic form, preferably as plain ASCII text or rtf. The deadline for the next issue is 7th January 2009.

Joining the RESG

Visit **http://www.resg.org.uk/** for membership details, or email **membership-RESG@open.ac.uk**