BCS™ Requirenautics Quarterly

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

©2004, BCS RESG http://www.resg.org.uk Issue 34 (December 2004)

Contents

RE-Soundings

From the Editor

In this issue, RQ is happy to offer coverage of a remarkably diverse set of recent events, along with a rich Christmas mix of articles and book reviews.

We give space to a team of keen-as-mustard, raring-to-go PhD Students, organized capably by Carina Alves, our Student Liaison Officer.

We also have the Defence community engaging with our discipline: they have to cope with the regular staff turnover required in their environment, so people have little time in which to pick up enough knowledge to start running with.

We have a fascinating and original article from Norah Power on the vexed question of Customers, Clients,

and Requirements Agreement.

We take a look at a selection of new books, including the Robertsons' latest book (I think it's their best) and a fresh perspective on requirements from Robin Goldsmith.

And of course now's the time to be thinking which events to attend or submit a paper to in 2005. Have a very Happy Christmas, and a prosperous and fruitful New Year.

Ian Alexander,
Scenario Plus

Chairman´s Message

Last week I had the pleasure of helping to run a workshop for PhD students working in RE (see Carina Alves' report in this issue). Much of the workshop was concerned with the mechanics of the research process. However, ten of the participating students presented brief overviews of their work. From interest, we mapped the topics being tackled by the students onto a set of outstanding issues in RE culled from Nuseibeh and Easterbrook's "Requirements Engineering: A Roadmap" (Proc. ICSE 2000) and Anthony Finkelstein's "Unsolved Problems in RE" talk given at this year's RESG AGM. There are various ways you can cut this and of course we only had a snapshot of current research in RE, so the results have to be interpreted with a bit of care. However, our snapshot revealed clusters of work on handling the results of contextual enquiry and on tracing, some attention being paid to extending formal models to systems' environment but nothing on handling NFRs, the relationship between requirements and architecture or the reuse of requirements.

This prompts the question, should we be worried? Does this indicate a disconnect between academic research and industry need? The first thing to note of course is that these are all hard problems and it can be hard to carve out a research agenda for a PhD student, with all the resource limitations that implies, for such problems. The second thing to note is that sometimes the most pressing problems are not necessarily the most interesting from the point of view of a PhD student. Finally, of course, our snapshot was a small one and certainly doesn't mean that because some topics aren't being covered by the group we saw they are being neglected everywhere. I know of a number of projects tackling aspects of each topic – and my overview is by no means all-seeing.

The PhD workshop was just the most recent of the year's RESG events. I do hope that you managed to come to some of them and that you take advantage of next year's programme of events and help make them a success. These kick off on the 26th of January with an RE for medical informatics event on my home turf at Lancaster. At present, events are decided by the committee where we try to anticipate their appeal to the membership. It would be good to hear directly from members of ideas for future events – please don't be shy in putting forward your views on this or any aspect of the RESG.

With that, I'll wish you a happy Christmas and a very prosperous 2005. On a final note; if you're an existing RESG member you'll be receiving a membership renewal request with this RQ. An early renewal will ensure you continue to receive RQ in '05 and help us continue to service the RE community.

Pete Sawyer
Computing Department, Lancaster University

RE-Treats

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

Forthcoming events organised by the RESG:

RE for Medical Informatics

26th January 2005

To be held in Lancaster.

Medical applications demand high performance and dependability, as they are often safety-related. How should their requirements be handled? Speakers include: Rob Proctor; Oliver Wells; Ashok Gupta.

Registration is required: contact Pete Sawyer (sawyer@comp.lancs.ac.uk).

The *i** Conference: Goal modelling with the *i** approach: 3 days of events

20 April 2005, Tutorial and 4 Talks, City University, London

21-22 April 2005, Invited Speaker Workshop, University College London

Concerned about how to model the goals of diverse actors in your organisation or project?

Unsure how to explore complex goal trade-offs during your requirements process?

This conference contains events for both practitioners and researchers. It will introduce, tutor and investigate the *i** approach for modelling and reasoning about the goals of heterogeneous systems.

i* (pronounced eye-star) is a powerful approach for modelling and reasoning about the goals of heterogeneous actors in business and socio-technical systems, and for choosing systems architectures that best meet these goals.

The events will be in 2 parts.

- On Wednesday 20th April there will be a half-day tutorial on *i** followed by 4 presentations of the use of i* on industrial projects.

- On Thursday and Friday 21st and 22nd April there will an invited speaker workshop to investigate and extend the *i** approach.

For registration, contact Neil Maiden (N.A.M.Maiden@city.ac.uk).

An Audience with ……

One of the big names in software engineering.

May 2005. (Date to be announced)

To be held in London.

Contact Bashar Nuseibeh (B.Nuseibeh@open.ac.uk)

Tool Vendors Day and AGM

Wednesday 6th July 2005 (Provisional Date)

Each Vendor will speak on how their tool meets the following challenge:

You are acting as requirements management consultant to a client who wants to automate his existing multi-storey car park with time-stamped ticket-issuing machines, payment machines, closed-circuit television cameras to deter both theft and non-payment, and automatic barriers operated by validated (paid-up) tickets. The client' s systems engineer has advised that the requirements should be organised into a list of stakeholders, a set of stakeholder requirements, a system specification, a project dictionary, and a list of references, with traces between these (the dictionary and references both receive traces from all the other documents; the specification traces to the requirements, which trace to the list of stakeholders). The requirements will certainly need to be prioritised, approved (or rejected), and then have their status tracked through to final acceptance of the automated car park.

Show (without spending time restating the problem) how your tool handles this challenge. Illustrate briefly the steps you would go through to structure the requirements, traces, priorities, and status in your tool.

Present slides on each of the following topics:

- setting up the information structure skeleton;
- importing the requirements from Word or text files;
- setting up the traceability;
- prioritising and approving the requirements;
- tracking the status of the requirements;
- checking the completeness of the traceability;
- any special features of your tool that assist with these tasks.

Registration: contact David Bush (David.Bush@nats.co.uk)

RE-Calls

Recent Calls for Papers and Participation

RE'05: 13th IEEE International Requirements Engineering Conference

August 29th - September 2nd, 2005, Paris, France

http://www.re05.org

Important dates

Paper abstracts (technical, evaluation, reflection papers): 07 February 2005
Paper submissions (technical, evaluation, reflection, practice papers): 14 February 2005
Notifications sent to authors: 22 April 2005
Camera-ready papers received: 03 June 2005
Workshop, tutorial, and panel proposal submissions: 11 March 2005
Doctoral symposium submissions: 28 April 2005
Poster and research demonstrations submissions: 28 April 2005

ICSE 2005: 27th International Conference on Software Engineering

15-21 May 2005, St Louis, Missouri, USA

http://www.icse-conferences.org/2005/

http://www.cs.wustl.edu/icse05/Downloads/ICSE05_CFP_General.pdf

Other events and calls that might possibly be of interest to RESG members:

IADIS 2005: International Conference Applied Computing

22-25 February 2005, Algarve, Portugal

http://www.iadis.org/ac2005

ICEIS 2005: International Conference on Enterprise Information Systems

24-28 May 2005, Miami, Florida, USA
http://www.iceis.org

CITSA'05 and ISAS'05

11th International Conference on Information Systems Analysis and Synthesis: ISAS´05 and 2nd International Conference on Cybernetics and Information Technologies, Systems and Applications

July 14-17, 2005, Orlando, Florida, USA

http://www.infocybernetics.org/citsa2005

CaiSE'05

13-17 June 2005, Porto, Portugal

http://www.fe.up.pt/caise2005

REFSQ'05: Requirements Engineering: Foundation for Software Quality Workshop

will be held in connection with CaiSE'05.

http://www.refsq.org

SERP'05: International Conference on Software Engineering Research & Practice

June 27-30, 2005, Monte Carlo Resort, Las Vegas, Nevada, USA

http://www.cs.und.edu/~reza/SERP05.htm

SCI 2005: 9th World Multi-Conference on Systemics, Cybernetics and Informatics

10-13 July, 2005, Orlando, Florida, USA

http://www.iiisci.org/sci2005

EISTA'05:

3rd International Conference on Education and Information Systems: Technologies and Applications

14-17 July 2005, Orlando, Florida, USA

which this year will have "emphasis on the area of Software Engineering, mainly Requirements Engineering."

http://www.confinf.org/eista05

International Journal on Software Engineering and Knowledge Engineering

Special issue on software traceability

Contacts: Dr. George Spanoudakis gespan@soi.city.ac.uk, Dr. Andrea Zisman a.zisman@soi.city.ac.uk

Organizations and Society in Information Systems (OASIS) 2004 Workshop

http://www.ifipwg82.org/calls/oasis2004call.php3

RE-Readings

Reviews of recent Requirements Engineering events.

RE for Defence

19th October, 1 pm, Defence Procurement Agency, Abbey Wood, Bristol

Writing Better Requirements – The Good, the Bad and the Ugly

This was our annual regional event and was made more interesting by the diversity of the audience, which consisted of people who had just started writing requirements, to authoritative practitioners and lecturers. The presentations were pitched to the lower end of knowledge pertaining to requirements engineering, but all attendees that I spoke to came away with food for thought.

Gordon Woods kicked the event off with a very apt PowerPoint presentation given the secure military environment (a bombproof lecture theatre), which included graphical gun shots complete with realistic sound effects. It was a miracle that nobody ducked for cover!

Gordon gave a brief overview of the types of requirements before launching into defects to avoid and a list of words that should be banned from requirements specifications. Too much was said to describe it all here, but it is well worth looking at his slides.

Gordon Woods' slides can be found at http://www.resg.org.uk/events.html

The second presentation was given by Michael Goom who discussed the difference between a common view and a realistic view of defence procurement. Michael reminded the audience of the importance of the users/human perspective in the elicitation of system requirements and it was interesting to see some of Gordon's 'banned' words in some of the examples given. The presentation ended with Michael highlighting the role of internal and international standards in the documentation of system requirements.

The final presentation was given by Colin Ingamells who started his presentation with an overview of reasons for project failure and the cost committed to projects at the concept stage. The audience was then given a verbal 'walkthrough' of the V life cycle model from the perspective of Validation and Verification; using examples from his person experiences and well known case studies.

The afternoon ended with a brief panel session. One discussion revolved around the difference between systems requirements and software requirements. I walked away from the event wondering whether the RESG should run some basic sessions on what is requirements engineering, as well as our normal sessions on state of the art requirements engineering research and practice.

Another point of view on the Defence event:

The Requirements seminar at the DPA was well attended and well organised. The presenters were all excellent, and they covered a great deal of material in a short time. I am sure that nearly everybody learnt something useful and came away with some ideas or knowledge. Those who are just starting out on a DPA tour of duty or are new to Requirements will have derived the greatest benefit. More experienced hands, who have for instance attended the 2 day Requirements Engineering Symposium at Shrivenham, are hoping to find answers to deeper and more difficult questions in follow-up seminars.

This was a very worthwhile event and I look forward to the next one.

(based on comments from) Rod Chidzey, Requirements Project Manager, Military Airborne Communications and Homing Systems (MAC) IPT, MoD Abbey Wood

Doctoral Workshop

8th December, University College London

See also the students' abstracts for this workshop, which are presented in full in the special **RE-search** *section below.*

The workshop was a full day event organised by RESG as a forum for PhD students doing research in Requirements Engineering. The idea was to provide an environment for students to expose their ideas to other students and a small, carefully-chosen panel of experienced academic facilitators. Juan Ramil and Pete Sawyer from the RESG were on hand to provide RE-specific input. However, we were lucky to have Ita Richardson (from the Department of Computer Science at the University of Limerick) and Marian Petre (from the Computing Department at the Open University) to provide guidance on more generic issues relating to doing a PhD and on crucial methodological issues for RE researchers. Ita and Marian proved to be very skilful facilitators who have deep insights into what it is to do a PhD while also being notable experts in their own areas of software engineering and computer science.

Following a welcome from Carina Alves who organised the event and is herself in the final stages of her PhD, Ita presented practical guidance on how to write a PhD thesis based on her own experience. She explained the usual parts that a PhD thesis might have and discussed the importance of deciding the structure of your thesis before starting to write it. Ita provided interesting advice on the importance of knowing your own writing requirements such as choosing the right environment, deciding if you prefer long stretches or short writing times, and most important of all, avoiding distractions that may divert the focus from your thesis. In fact, during discussions over the course of the day, students pointed out that keeping focused is one of the greatest challenges they face in their PhDs.

Following Ita's talk (and punctuated by lunch), ten students (nine from the UK and one from the Netherlands) squared up to the daunting task of presenting their research in ten minutes flat. It was striking that, even though some of the students were presenting their work for the first time, and even though presenting anything worthwhile in ten minutes is very hard, the presentations were of high quality. At the end of each presentation, there was five minutes for the audience to ask questions. Questions came from students and facilitators in about equal measure. While students' questions tended to focus on the technical issues, those of Ita and Marian in particular tended to probe process, motivation and methodological issues. This was very effective at surfacing common themes such as the absolute need to be able to articulate a clear question that your research addresses.

Following the student presentations, Pete led a discussion session which he began by flagging the other stakeholders in the PhD process; supervisors and examiners. A table was then presented that mapped the participating students' research onto open challenges in RE culled from [Nuseibeh and Easterbrook 2000, Finkelstein 2004], such as: handling non-functional requirements, bridging the gap between requirements and architectures, requirements reuse, etc. The interesting thing about this was how little correspondence there was; maybe indicating that other issues are more interesting or (perhaps more plausibly) that the big problems aren't tractable to PhD-scale projects. Interesting though this might have been, it proved not to be fit for purpose since the subsequent discussion focused on more generic PhD issues. In particular, we talked about success and exception scenarios and the consequent requirements for and constraints on a PhD.

Instead of presenting a traditional talk, Marian Petre conducted an exciting dynamic exercise which she started by asking everyone to name the biggest obstacle to progress that they currently faced. A number of common themes emerged that had a remarkable correspondence with the stage in their PhD that the students had reached. For example, finding resources for evaluation emerged as a common headache for students close to or already embarked upon their write-up. Following this, Marian got everyone to line themselves up against a wall according to how close they were to finishing their PhD. After much shuffling about and reorganisation, while everyone found their right position, each student was asked to give what they considered to be the most important single piece of advice to the person standing next to them on the "less close to finishing" side. This proved to be an amusing but effective way to share experience and disseminate wisdom.

Next up was a mock viva where each facilitator played a role: Marian as supervisor and narrator, Ita as

internal examiner, Pete as external examiner and the heroic Juan as student. Counting his real viva, this was the fourth such ordeal that Juan had subjected himself to! The main goals of this experiment were to get students to understand that vivas require good – not perfect – performances and to reveal some of the "hidden mechanisms" behind vivas. The mock viva was very amusing as our "actors" delivered a superb performance. Students agreed that it was a good way to demystify the fear everyone has about the viva.

To round off the day and tie up the loose ends, Juan presented some useful tips towards a successful viva and wrapped up with some final conclusions.

In the end, the event turned out to be a day packed with activities, wisdom, advice, and sharing of concerns and everyone left feeling pretty tired. But it was also a really enjoyable and stimulating event in which the flow of wisdom was by no means one-way.

[Nuseibeh and Easterbrook 2000] Nuseibeh, B., and Easterbrook, S. : "Requirements Engineering: A Roadmap", Proc. International Conference on Software Engineering (ICSE-2000), Limerick, Ireland, 2000

[Finkelstein, A. 2004] Finkelstein, A.: "Unsolved Problems in Requirements Engineering", 10th anniversary address to RESG AGM, 2004

RE Books Event / Birds of a Feather Meeting

12th November at 6p.m.
City University, London

The event went very smoothly (at least, the drinks appeared to be going down that way after a hard day's preparation and committee meeting). We had the enormous and comfortable boardroom at the top of City University.

All the publishers who attended seemed to be very happy with the amount of interest they received from a knowledgeable audience of all ages. There were bookstalls from Wiley, Pearson's (Addison-Wesley), Springer, Artech House, Dorset House (unmanned), and the BCS itself which is just moving into publishing. They all deserve our thanks for the energy they put into this event.

There was a plentiful supply of wine, beer, fruit juice and soft drinks, accompanied by a wide range of things to nibble. Conversations ranged from how security requirements could not be framed in terms of probability (as human stupidity constituted a common-mode point of weakness everywhere), through current and future projects as birds-of-a-feather put their heads together, to which books people liked and what they wanted to see in print but couldn't find.

We stumbled out tired but happy into the night.

Autonomy and quality in distributed software systems: A dichotomy? Professor Wolfgang Emmerich's Inaugural Lecture

13th December 2004
University College, London

Wolfgang Emmerich was inaugurated as a Professor at University College, London, on 13th December 2004. He is a long-time member of the RESG committee, and although he is now very busy with his academic duties, he remains on our Industrial Liaison committee. We extend him our congratulations. The abstract and biography for his inaugural lecture are reproduced below.

Abstract:

Organisations increasingly rely on specialist suppliers to perform non-core activities. These activities are often supported by IT systems. This leads to distributed software architectures that require the integration of IT services across the boundaries of otherwise autonomous organisations. These organisations increasingly depend on their suppliers and IT services therefore need to be delivered to an appropriate level of quality. We review whether autonomy and quality are a dichotomy and show that they can indeed be reconciled using service level agreements. We present a novel approach to precisely define the meaning of formal languages for service level. We demonstrate how compliance to such formalised service level agreements can be monitored automatically. We conclude by showing a number of application areas that would benefit from the use automated SLA monitoring.

Biography:

Wolfgang Emmerich is Professor of Distributed Computing at University College London. He received his undergraduate degree in Informatics from the University of Dortmund in 1990 and went on to conduct research into process-centred software engineering environments. He received a PhD in Computer Science from University of Paderborn in 1995. After a brief post-doctoral appointment at the Software Verification Research Centre of the University of Queensland in Brisbane, he joined The City University as a Lecturer in 1996. He was appointed as a Lecturer at UCL in the Department of Computer Science in 1997 and co-founded the Software Systems Engineering Research Group, which he currently heads.

In parallel to his academic career, he worked for the Central European OMG representation on the CORBA middleware specification and co-founded three start-up companies for which he currently serves as a non-executive director. He has an active interest in Grid computing and serves on the UK e-Science Architecture Taskforce and the OMII Technical Advisory Board.

RE-Papers

Customers, Clients and Requirements Agreement

Norah Power, University of Limerick, Ireland

It is generally agreed that there is no such thing as an ideal requirements engineering process, that it depends on the situation. In this article I look at different requirements situations and discuss the different types of agreement might be reached among the stakeholders in each type of situation.

One thing that is widely agreed in RE is that requirements gathering or elicitation begins with the stakeholders. Stakeholders are people or organisations that have an interest in the outcome of a development project and therefore want to have or should have some say or influence over the system/software requirements. Stakeholders include the end-users of the software, and other parties, such as their customers, who are only involved indirectly. Stakeholders often include government departments and regulatory bodies. They always include the developers of the software, and their managers or employers. In fact, the term 'stakeholder' is often used as an umbrella term for all of the participants who are concerned, either directly or indirectly, with the requirements for a project. Stakeholders are generally defined in relation to a particular project, that is, they exist as stakeholders because they have a stake in the outcome of a particular project.

What is not often realised or made explicit is that stakeholders have two very different roles in relation to requirements. These are:

1. As sources of requirements
2. As parties to the agreement that comes out of the requirements process (requirements agreement)

Let us take an example to illustrate the difference. A large hospital wants to acquire a system for its personnel, payroll and related functions. The hospital decides to purchase the Human Resource Management (HRM) module of a well-known ERP (Enterprise Resource Planning) software package. As a customer of this ERP vendor, the hospital could well be regarded as a stakeholder, along with other hospitals and several other similar organisations. The ERP software vendor is aware of this and gathers input to the requirements for the HRM module from these market sources. Typically, these stakeholders are represented in the requirements gathering process, their views are taken into account to some extent, but they have no involvement in the agreement that comes out of the requirements process. That agreement is an internal matter for the vendor company itself. There are many other sources of requirements besides the customers, and each customer, however large, represents only a small portion of the market in a situation like this. The HRM module has other sources of requirements that are related to the vendor company's business plan, its business partners, its software architecture, previous and future versions of this product and its other products.

The HRM package will to need to be adapted to suit the specific requirements of the hospital. This is known as configuring or integrating the package. The hospital will typically decide to contract a software services or consultancy company to help it with the configuration project, which typically could take several months. The hospital's specific requirements for payroll deductions and personnel administration functions, such as staff leave and rostering, will need to be documented and agreed before the new system can be implemented. These requirements will have to be within the constraints imposed by the HRM package, but as the ERP vendor's marketing department will assure you, any reasonable requirements that the hospital might have within the scope of HRM can be satisfied, eventually, by the HRM module.

In this project, the hospital, its management and its personnel are stakeholders who again are sources of requirements, but they will have a much greater say in the specific requirements for the system that they are going to get than they had in relation to the package that they bought.

The hospital in this example has two suppliers, the package vendor and the service company, and two very different relationships with them. The relationship with the package vendor is as a customer. Individual customers in a market have some or very little influence on the requirements for a software product that they later on may or may not choose to buy. As stakeholders, they may have an input to requirements, but they are not party to the agreed requirements as determined by the software company. Their contract with the software vendor does not begin until the software has been delivered.

The hospital's relationship with the service company is as a client. Clients are organisations or people who commission software or services from a supplier. The integration services supplier undertakes to establish the client organisation's requirements and to fulfil them within the constraints imposed by the chosen package.

Unfortunately, the terms customer and client are rarely distinguished clearly in requirements terminology (there are some exceptions) but are often used interchangeably in discussions about the requirements process. For example, the IEEE-830 'Guide to the Software Requirements Specification,' states that the most important role of the software requirements specification is to "*Establish the basis for agreement*

between the customers and the suppliers on what the software product is to do."

This is a pity, because, as argued above, while the customers may have an input, they do not always have a say in the agreed requirements. In the case of the ERP software vendor, it is likely that the requirements for a specific release of the product are agreed within the organisation following a formal process of consultation and review meetings. This type of agreement has the significance of endorsement by the hierarchy of the organisation, represented by the multiple reviewers of a requirements document. It has no legal significance, unlike the agreement between the hospital and the service company that configures the software.

The requirements document agreed between the hospital and the service company has the force of a legal contract, or a formal agreement between two legal entities. It protects each party from some of the consequences of lack of satisfaction when the system is delivered. For example, if it can be shown that the supplier has fulfilled the requirements specified in the document, then it does not matter whether the client is satisfied, the bill has to be paid. If the agreed requirements are not fulfilled, then the client has a case for withholding some or all of the agreed payment.

This type of agreement applies in many situations where software is developed or configured for a specific user organisation. Each party needs to ensure that it can control the outcome in some way:

– The client organisation needs to know that it can 'get what it is paying for' by having an explicit contract for the development.

– The supplier organisation needs to know that it can control its costs by estimating and charging for the cost of the work and by being in a position to negotiate increased revenue if the client decides to expand the scope or radically change the requirements.

The two types of situations are independent of the type of software being developed or deployed. Software product companies often contract out the work to other companies, and user organisations such as hospitals may commission their own software. Requirements engineering takes place in a wide variety of situations, yet it lacks a theory to explain how different techniques are appropriate to different situations.

$\subseteq \in$

This discussion is based on research presented at REFSQ04 in June. My thanks to the REFSQ discussants, the reviewers and all the participants at Riga.

The research is funded by the Science Foundation Ireland (SFI) Investigator Programme, B4-STEP (Building a Bi-Directional Bridge Between Software ThEory and Practice).

Requirements Engineering is……

by Kathy Maitland
RESG Committee Member, and
Lecturer, University of Western England

Methods, Techniques, & Tools?

At the Bristol event I was asked the question 'What is requirements engineering?' I replied with some trite answer along the lines, 'a collection of methods, techniques and tools that are used to elicit and analyse the requirements of a system to meet the needs of the organisation commissioning or operating it.' In simple terms, a variation of the Avison and Fitzgerald definition of a systems development methodology (Avison and Fitzgerald 1995). This was my definition, but at the recent BOF event, my colleagues on the RESG committee suggested that I accosted people and ask for their thoughts on a definition of requirements engineering. I had an interesting evening asking people to complete the following sentence, 'Requirements Engineering is…..' As a band of people from a variety of industrial and academic backgrounds and away from the other lecturers who gave a similar response as I had and students who said it was whatever their Prof said; I did received some interesting replies.

An Art?

My first reply came from James Robertson, who immediately said that it is not an 'engineering' discipline. This was interesting to me as I had recently argued that it was an engineering discipline. In my opinion, requirements engineering is all about a structured, systematic approach to the elicitation and analysis of system requirements.

In contrast, James Robertson argued that requirements elicitation was an art. It was about finding, 'what is the problem in the real world' and what was needed to solve it. To him, requirements engineering was about finding the boundary of the solution and tracing between the two, i.e. the organisational problem and the boundary of the solution.

Such comments are similar to those of uttered many times by the RESG patron – Michael Jackson, who suggests that systems developers rarely solve an immediately recognisable and well understood problem and that developers should start by describing and structuring your problem (Jackson 2001).

Therefore, we could say that requirements engineering is both an art and an engineering discipline: there is a need for creative processes in the elicitation of system requirements, and an engineering approach in the form of methods and techniques that enable developers to describe and structure the problem.

A Good Title for a Book

I then accosted our RESG Chairman – Pete Sawyer – who suggested that Requirements Engineering was a good title for a book, and I guess he should know; as

he has written one (Sommerville and Sawyer 1997). Pete was of the opinion that requirements engineering was the pump primer of the development process of all systems life cycles and that requirements were hard to establish. This remark led to discussions on problems of requirements elicitation, especially when you have a variety of experts who think that they know what everyone else knows and how to do it. David Bush's interjection was, 'Requirements Engineering is easier if you call in a consultant.' Well, I guess that is one solution.

A Pragmatic Process?

Another response came from one of the publishers who said that pragmatic approaches were being requested in this growth area of computing. He went on to suggest that 'if more people understood the process of requirements engineering the world would be a better place.' These two comments provide an interesting view of requirements engineering and suggests there is a lack of pragmatic approaches within the discipline. If you read books pertaining to the subject of requirements engineering there are few requirements methodologies, but many individual techniques and generic methods that could be used in a requirements engineers toolkit.

Assuring a Benefit?

From group discussions there was a general consensus that requirements engineering is about capturing 'what an organisation wants and what they do not want' and it required information about the organisation and the processes that it performed. It was to address these general points that Jeremy Dick suggested that requirements engineering is the negotiation, definition and the assurance of benefit. If there is no benefit in making proposed changes to a current system, why change it.

For Life, not just for Christmas

There are no definitive answer to the statement 'Requirements Engineering is……', but as Jeremy Dick jokingly commented, 'Requirements are for life and not just for Christmas' and of course he is right; requirements are for life of the system, they might be changed, evolved or even dismissed but must still be traceable and they will all will have impacted upon the development and evolution of a system.

References

Avison, D. E. and G. Fitzgerald (1995). Information Systems Development Methodologies, Tools and Techniques, McGraw-Hill.

Jackson, M. (2001). Problem Frames: Analyzing and structuring software development problems. London, Addison-Wesley.

Sommerville, I. and P. Sawyer (1997). Requirements Engineering: A Good Practice Guide. Chichester, John Wiley & Sons.

What is an Exhaustively Satisfied User Requirement Anyway?

In classical RE theory, the system specification (let's call it the SRD, and an individual system requirement an SR) is said to "satisfy" the so-called "User" Requirements (URD, and UR). Stakeholder or Business Requirements (neither are exact synonyms of UR, by the way) might be better, but that's another story. It satisfies it exhaustively, if we're to believe the theory: every atomic UR is accounted for in all its meaning. No SR that is not shown to be mandated by the UR is supposed to exist. Either a "satisfies" trace must be added to show which UR(s) the untraced SR exists to satisfy, or else the SR must be deleted. Or so the theory says.

Intentionally sexual language? Drawing by William Stukeley (1687-1765), showing the greek goddess Artemis seated on the sphere of the stars, and holding a painting of Isaac Newton, adorned like a greek hero

Now this is a strong claim. Leaving aside the engagingly sexual language – reminiscent of Isaac Newton's (intentional) choice of technical terms such as "the attraction of heavenly bodies" [Newton 1687], to much learned sniggering over the claret – the claim is that an SR is valid if and only if it is called out by a UR, and that a URD is satisfied only when all the URs are exhaustively satisfied by suitable SRs:

$$\text{URD} \leftrightarrow \text{SRD}.$$

Really? It may be a neat and tidy theory, but what about the law, international standards, and the like? Does the URD have to contain all of those? Surely not: the poor old stakeholders just need to say what they want, and leave it to the systems people to say which parts of the law are applicable in their case. That's where the knowledge probably is, and there is a sensible division of labour. Standards can be thought of as reusable requirements, and in mature domains they cover many of the required qualities of a system (whereas the functionality is presumably unique, and less easy to recycle).

Secondly, a whole lot of system needs come from the process of analysis, and indeed from trading-off design options. But do such requirements have to be justified *post hoc* by pretending that they came from the URD?

Suppose that the metallurgy guy comes up with some constraints on how the metal components are to be worked. Do we have do go away and invent a matching UR

"All metal parts shall be bashed appropriately"?

There's something wrong here! Users can't be expected to know about such things, and even the existence of metal within the chosen solution may not have been known in advance: the level is wrong. This is a system issue, and apart from a completely general requirement for safety and reliability, the users should not concern themselves with it. The truth, surely, is that the specialist disciplines relevant to the system solution are stakeholders in their own right, though in no sense "users".

If so, we should either abandon the UR/SR distinction, or the idea that there should be complete traceability between the two. A "user" statement of need – Business Requirement, Problem Statement, whatever – is clearly valuable and in a sense natural. We *do* want to describe the problem before specifying a solution. And we *should* check that we haven't forgotten to account for any of the individual URs in the specification.

So, the thing that has to give is any feeling or hope that a URD can be comprehensive. It describes what some of the stakeholders want – hopefully, a properly representative set of people. They can only speak from their own backgrounds and experience. But is it The Requirements? No. There'll be plenty of others. The SRD does have to satisfy the URD – where it's appropriate, affordable, and well-founded. After that, the SRD is on its own.

As for the individual SR-UR relationship, it may happen that an SR on its own sometimes satisfies a UR completely. But if the URD is genuinely at a higher level of abstraction than the SRD, it's likely that several SRs will be needed per UR: each of them "contributes to satisfying" the UR. That must be the general case – even if it doesn't sound quite so snappy.

References

Isaac Newton, 1687. *Philosophiæ Naturalis Principia Mathematica (The Mathematical Principles of Natural Philosophy).* Imprimatur S. Pepys, Reg. Soc. Praeses Julii 5. 1686. Londini Jussu Societatis Regiae ac Typis Josephi Streater. Prostat apud plures Bibliopolas. Anno MDCLXDXXVII.

RE-search: Doctoral Workshop Students' Abstracts

The Doctoral Workshop (for PhD Students), held on the 8th December 2004 at University College London (see the Report in **RE-Readings** *above), is an opportunity for research students starting out on the road to a Doctorate to develop confidence in talking about their work. Writing a brief abstract that people working in other areas can understand is an essential part of this. Here is how the students describe their proposed theses.*

Lindsay Smith: *Retro-methodology*

Supervisor: Dr. Tracy Hall

The solution of a technical problem: Placing a computerised artefact in any given operational environment potentially changes that environment in unpredictable ways. Once implemented any underlying change caused has the potential to invalidate any developmental decision taken prior to that implementation. The computerised artefact is 'in situ'. The developer can attempt to anticipate change but application of developmental techniques retrospectively is problematic.

Prior research has incorporated work originating in the social sciences to improve the context in which the requirements of stakeholders are processed. This work has gone some way towards a solution of the problem outlined above. A consensus exists that a developer's interpretation of an existing environment should include stakeholder views. Less agreement exists as to how this reduces the risk of undesirable events happening after implementation, e.g. the 'system' is inadequate for stakeholder requirements.

Formalising (to make machine-executable) an informal (e.g. human-activity) system remains problematic. You cannot rule out unpredictable change even if the scenario includes correctly identified stakeholders making perfect sense of a developer's interpretation. Some people would indeed consider this 'best case' scenario highly unlikely in the first place. Certainly, the issue of *how* to include stakeholder views in requirements engineering continues to be open to argument. Interestingly the argument as to *why* include such views appears much less open.

The research hypothesis: That further work is possible on 'social issues' in requirements which can improve representation of stakeholder context in computerised solutions. A more socially sophisticated representation, of stakeholder requirements, could be used to 'off set' any negative effects of stakeholder environment change due to computerised implementation.

Sketch of proposed solution: Evaluate social science techniques that have previously been 'adopted' into the requirements engineering process. Identify 'workable' aspects of this type of technique for the requirements process. Investigate incorporating such

techniques into the requirements engineering process, using existing models/methods if possible. Apply 'post-operational' requirement 'smoothing' using the 'social issues' identified during the 'pre-operational' part of the requirements process.

The expected contribution: The nature of this problem area is essentially 'paradoxical' and unlikely to be entirely resolved. The introduction of new technology has the potential to alter stakeholder environment/view unpredictably by default. That said addressing 'social issues' with such an approach should, on balance of probabilities, give further insight into stakeholders role in the requirements process.

Evaluation of results: It would be desirable to 'prove' this with a computerised solution in an operational environment. Scalability is a likely problem factor with this type of evaluation. A direct comparison, if possible, with a more traditionally developed solution could 'prove' increased compatibility with stakeholder views within the operational environment. Other methods of evaluation may be necessary for interim results etc.

Mark Nicholas Elkins: *Requirements from Marketing?*

Supervisors: Professor Margaret Ross, Geoff Staples, and Ian Tromans all of Southampton Institute

Abstract: Technical problem to be solved: Marketing is concerned with the identification and satisfaction of customer needs. Therefore can the methods that it offers be of significant use for [large UK*] organizations in the identification of initial needs for software for internal use? Identification of initial need is the foundation on which all software projects are based and hence the importance this has on the quality of software produced. An extensive literature review has revealed that there appears to be no previous research on this specific problem.

Research hypothesis: Marketing methods are of significant use for [large UK*] organizations in the identification of initial needs for software for internal use?

Sketch of the proposed solution: This positivistic study intends to use survey questionnaires and interviews to gather quantitative and qualitative data from a sampling frame of [large UK*] organizations [, which have over 1000 employees*]. [The population sample being chosen by taking a quota of one, by computer programme generated random selection, from each county within the UK where such an organization exists.*] Sample size will be decided by considering what is achievable within a research project being undertaken by a PhD research student over 3 years within given resource and time constraints. Collected data and variables will be analysed using statistical methods in an attempt to disprove the hypothesis. There will be an ongoing search and review of secondary data to provide information on and keep up to date with external research within the problem boundaries.

Expected contributions of this research: To add knowledge that improves the quality of software used and produced through the better identification by [large UK*] organizations of initial needs for software for internal use.

Explanation of planned evaluation of results: Respondent validation, Negative case analysis, Reflexivity and Replication. Such methods to be used to attempt to check the Reliability, Validity, and Generalisability of the research.

* Provisional Research Boundaries

Waraporn Jirapanthong: *A Rule-based Approach for Traceability of Product Family Systems*

Supervisor: Andrea Zisman

Requirements Traceability (RT) has been recognized as an important activity in software system development. Traceability relations can improve the quality of the product being developed, and reduce the time and cost associated with the development. In particular, traceability relations can support evolution of software systems, reuse of parts of the system by comparing components of the new and existing systems, validation that a system meets its requirements, understanding the rationale for certain design and implementation decisions in the system, and analysis of the implications of changes in the system.

However, automatic generation and maintenance of traceability relations is not an easy task. Very few approaches have been proposed in order to support automatic generation of traceability relations. The majority of the approaches assume that traceability relations should be established manually, which is error-prone, difficult, time consuming, expensive, complex, and limited on expressiveness. Therefore, traceability is rarely established.

In this work, we propose a rule-based approach to allow automatic generation of traceability relations between documents created during the development of product family systems. These traceability relations can be used to facilitate identification of *common* and *variable* functionality in the product members of the family, and to support reuse of core assets that are available under the product family architecture.

We are interested in creating traceability relations for documents generated in feature-based methodologies. We also believe that object-oriented methodology is important to support product family development and we concentrate our work in an extension of the FORM methodology that combines object-oriented documents and documents proposed in the FORM methodology.

Our work focuses on eight different types of documents and assumes the documents represented in XML, in order to allow interchange of documents produced by different tools. We have identified nine different types of traceability relations between these documents. We propose to use XQuery to represent the traceability rules. Currently we are developing a prototype tool for enabling automatic generation of traceability relations for product family systems. We plan to evaluate our approach in real case studies in terms of recall and precision of the traceability rules.

Amit Thakur: *Perception of image by its sense in Content-Based Image Retrieval*

Supervisors: Dr. Lynee Dunkley and Dr. Amer Salman Thames Valley University

Problem To Be Solved: As we know that until now there is no proper standards for developing image description. This research for solving the problem of image description in content-based retrieval. The content in this system is how to recognize the image according to user's sense and retrieve the image by that sense from the database.

In this research, we need to develop a concrete set of metadata that facilitates content based image search by its sense. We also need to develop a system that recognizes the image automatically and store it under its proper datatype in the database. The datatypes should be categorises according to image senses. Sense classification for images are common sense, emotionality and sense of humour. There should be a hierarchical approach maintained between these senses i.e. hyponymy (the semantic relation of being subordinate or belonging to a lower rank or class).

Develop a computer vision system, which is capable of image description and can recognise the complex scenes rapidly like a human visual sense does.

This research could benefit:

- The heritage sector, including art galleries, museums and libraries
- Newspapers and other media organizations maintaining image archives
- Multimedia developers who need to re-use content
- Engineering firms with extensive ranges of spare parts
- Software developers in the image or multimedia database area
- Medical applications.
- Academic research groups.
- Home entertainment
- Web searching.

Yun Chen: *Designing the User-Interface for Effective Interaction with E-planning Systems Using a Human-Centred Approach*

Supervisor: Andy Hamilton

With the development of Information Technology, more and more digital systems are applied in urban planning process, which refers to e-planning systems.

Although the usability of products developed for e-planning use has improved immensely in recent years, they still require users to have or acquire considerable technical knowledge to operate them. The major obstacle to non-expert users is navigating an interface that embeds a language, world view and concepts that support the system's architecture rather than the user's work view.

Human Computer Interaction (HCI) received attention in the first part of the 1990s to solve this kind of problem; however it seems that within urban planning research, little attention has been paid to the influence of HCI on research and practice.

Furthermore, the type of current users that are being exposed are very different from those who have been at the centre of the earlier research on HCI issues, who are specialists using the system to accomplish a specific work-related task. With the varying level of computer skills and literacy, the general public may use the e-planning system in one of a large number of application areas.

Accommodating such a wide spectrum of needs is a challenge and the interface is generally the key. A Human Centred Approach (HCA) opens new avenues for understanding users' expectations from an e-planning system, the ways in which they use, understand and value the system, and the role of e-planning systems within the wider societal context, so that it can provide valuable information in the designing of effective user interface architectures.

As a result, HCA is supposed to be a proper way to make a complex computer technology accessible to a wide range of users, who are bringing a diversity of knowledge, technical capabilities and cultural perspectives.

In addition, finding an appropriate balance of theoretical rigour and practical applicability in HCA is also worth to be addressed, which will benefit future researchers in related field, as well as contribute to the reality of the Inclusive Knowledge Sociality in 2010.

The whole research will follow the 'activity line' as 'Investigate – Generate –Develop – Test – Evaluate'. After the literature investigation, a conceptual model will be generated based on HCA, which involves two ideas, i.e. 'User as Nucleus' and 'User as Refiners'.

This model will provide a sound base for developing a prototype interface in a project called 'IntelCities'.

During the process of IntelCities project, the prototype will be tested repeatedly and redeveloped.

And finally, evaluation methods will be adopted to evaluate four aspects of the research, i.e.

- using Cognitive Dimension Framework (CDF) to evaluate HCA theory applied in urban area (theoretical evaluation),
- using HCI usability evaluation to evaluate the final prototype developed (prototype evaluation),
- using comparative methodology to evaluate whether or not HCA could increase the interactivity with e-planning systems (hypothesis evaluation) and
- using triangulation methodology to evaluate whether or not research methodologies adopted is correctly used (methodology evaluation).

Zhi Li: *A Semantics of Problem Frames*

Supervisors: Dr Jon G. Hall, Dr. Lucia Rapanotti, Prof. Darrell Ince, The Open University

Problem Frames [3] provide a useful way for people to understand and solve software problems. They contain a concrete set of syntaxes – graphical notations (e.g., labelled boxes representing domains, annotated arcs representing shared phenomena, dotted ovals with dotted arcs representing requirements) for communication and informal reasoning in RE practice.

It is a widely recognised problem in software development that a graphical notation (syntax) without an unambiguous semantics can cause difficulties in communication [5] – a typical example is UML [1].

Although graphical notations aid communication, they do not necessarily facilitate formal or rigorous reasoning. A formal semantics can help with rigorous reasoning and avoid ambiguities. Not enough research has been focused on rigorous reasoning from domain modelling [4] to software solutions, especially in the context of problem frames.

Hall et al [2] have established a semantic framework for Problem Frames – a set-based logical characterisation of solutions that map to a problem diagram, which should work in all cases in RE because it is a textual language framework that can reason about both formal and informal descriptions.

However, it has not given constructive methods to find the solutions. We have begun investigating a restricted form of CSP that can provide this constructive method.

I expect that my PhD will facilitate RE practitioners in representing problem diagrams, defining equivalent problems, allowing problem transformations, and underpinning correctness arguments for frame concerns. The results will be evaluated by a series of case studies.

References

[1] A. Evans, J.-M. Bruel, R. France, K. Lano and B. Rumpe, Making UML precise, *Proc. OOPSLA'98 Workshop on Formalizing UML. Why? How?,* 1998.

[2] J. Hall, L. Rapanotti and M. Jackson, Problem Frame Semantics for Software Development, *Journal of Software and Systems Modelling* 2004 (to appear).

[3] M. Jackson, *Problem Frames: Analyzing and Structuring Software Development Problems*, Addison-Wesley, 2001.

[4] M. Jackson and P. Zave, Domain Descriptions, *Proc. IEEE International Symposium on Requirements Engineering*, IEEE CS Press, 1993, pp. 56-64.

[5] M. Petre, Why looking isn' t always seeing: readership skills and graphical programming?, *Communications of the ACM*, 38 June 1995, pp. 33-44.

Paul Arkley: *Traceable Software Development*

Supervisor: Dr S Riddle, Newcastle University

The Technical Problem: Modification of complex computer based systems requires a detailed understanding of their functionality. To gain this understanding it is necessary to traverse the development artefacts looking for common threads of development reasoning.

Requirements Traceability is a technique which helps the engineer to find these threads. Previous requirements traceability research has concentrated on structuring information, rather than considering how traceability relates to the main development process.

In practice, traceability is performed as a separate quality-proving process by a different set of engineers from those who developed the product.

Thesis Hypothesis: We argue that the lack of direct benefits to main development process from traceability is the cause of the above problem, leading to information that is incomplete, inaccurate and out of date. Integrating traceability into the development process will significantly improve the quality of the information. To succeed, this integration must provide immediate, tangible benefits for the main development process.

Proposed Solution: We propose an inter-phase Traceable Development Contract in order to provide these benefits.

The TDC defines the actions to be taken by related development phases, such as requirements definition and software design, in response to changes to their common information artefacts.

The TDC will consist of three parts:

- the common information artefacts;
- traceability information structures which record

how the common information artefacts are related to the development phases; and

– a behaviour protocol.

The protocol will define the behaviour of each development phase depending on changes in the common information artefacts and traceability information structures. The TDC is beneficial to the development process as it, improves the quality of traceability information, coordinates inter-phase development and provides a means of assessing and negotiating development changes.

Contributions of PhD: This thesis builds upon previous work by establishing a means for the integration of traceability techniques into the main development process. This integration and the formalisation of the relationship between development phases will result in an increased level of traceability information correctness. This will, in turn, result in greater product understanding and reduced upgrade costs.

Evaluation of Results: The thesis hypothesis will be tested by:

- Determining the empirical evidence from traceability practices that supports or denies the hypothesis.
- Determining a theory from the empirical evidence that describes the factors involved in the recording of traceability relationships.
- Conducting experiments where student teams apply either an integrated or separate traceability recording process to a simple development case study. Comparison of the results from these groups will show whether that an integrated method produces a richer set of recorded relationships.
- Implementing the TDC in an industrial setting and recording the observation of the engineers taking part in the development process.

David Nutter: *A Self-Organising Awareness System for Distributed Software Engineering*

Supervisor: Cornelia Boldyreff, University Of Lincoln

Software engineers and other collaborative disciplines rely on informal "out-of-band" communication for effective coordination of their activities, especially in agile methods. This type of communication is lost when development is distributed, with consequent deleterious effects on engineering effectiveness. In order to effectively support distributed software engineering, a replacement for this informal communication must be found.

Much previous research focussed on either synchronous awareness such as radar views and shared editors, where participants were distributed in space not time, or asynchronous awareness such as change notification, which did not explicitly support concurrent activities. A unified approach is necessary to support software engineering.

Furthermore, requiring co-location of engineering teams is not possible in today's marketplace where development is often outsourced, consequently a definite requirement for awareness tools to replace informal communication exists. To implement an awareness tool capable of providing awareness of activities distributed both in time (asynchronous awareness) and space (synchronous awareness). The tool will not rely on a centralised reflector; instead information will be distributed over a peer-to-peer network arranged using a self-organisation algorithm.

Consequently awareness information need not travel more than a few hops from its originating peer, reducing network load and increasing relevance of information received. Unlike reflector-based CSCW systems, the network will scale and will not have a single point of failure in the reflector. Furthermore, without the need to setup a reflector, there is the capability for ad-hoc awareness, using low-complexity peer discovery by local broadcast for example.

The tool will be integrated with the Eclipse development environment. The files a user is currently editing will determine the data they are interested in and fuzzy similarity metrics will be used to compare the collections of each peer in the network in order to drive the self-organisation process.

To evaluate the success of self-organisation, a simulation approach will be used before deploying the algorithms in the wild. To evaluate the effectiveness of the awareness provision, initial deployment and controlled experiments will be conducted within the Distributed Software Engineering group at the University of Lincoln and a later version of the tool will be trialled with existing Eclipse users.

Marco Lormans: *Structuring Requirements Evolution in Embedded Systems Development*

Supervisor: Arie van Deursen, Delft University of Technology

Current requirement engineering tools support the evolution of requirements insufficiently for embedded systems development practice.

The multidisciplinary environment (ranging from mechatronics to human machine interfacing) and the ongoing evolution of these systems cause inconsistencies in the set of requirements, which in turn lead to error-prone, time consuming, and costly repairs.

An explicit requirements management environment incorporating both structured and semi-structured data, supported by tools, and tailored towards the embedded systems domain, is needed to improve requirements

evolution with respect to

1) the interaction with stakeholders,
2) the presentation of the requirements and
3) the processing of changes in the set of requirements.

A Requirements Engineering System (RES) is a conceptual framework that provides a structured requirements management environment, which explicitly defines the process of evolution by identifying all three important aspects of requirements evolution.

Our RES captures the needs of the embedded systems domain, characterized by its multidisciplinary nature, product families, and product evolution. It provides a metastructure (RE meta-model) incorporating both structured and unstructured requirements data which supports reuse of requirements for different product families as well as processing changes to the set of requirements consistently.

Furthermore it provides flexible interaction between the various stakeholders involved in developing embedded systems making it easier to tailor it for a specific industrial situation. Finally, it provides a set of guidelines to set up a RES in practice using state-of-the-art RE technologies, improving the adoption of these technologies in practice [1].

The RES framework will be applied in a number of industrial and academic case studies. In each case study, selected aspects of the framework are investigated, resulting in improvements for the meta-structure and prerequisites for successful tool application. In [2] we investigated the complete process of requirements evolution in an outsourcing context. Other case studies include the generation of forms and views from a semi-structured set of requirements and the recovery of traceability links from an unstructured set of requirements – the latter in the form of a structured view.

References

[1] Bas Graaf, Marco Lormans, and Hans Toetenel. Embedded software engineering: state of the practice. *IEEE Software*, 20(6):61–69, November–December 2003.

[2] Marco Lormans, Hylke van Dijk, Arie van Deursen, Eric Nöcker, and Aart de Zeeuw. Managing evolving requirements in an outsoucring context: An industrial experience report. In *Proceedings International Workshop on Principles of Software Evolution*, Kyoto, Japan, 2004. IWPSE04.

Paul Adams: *A Collaboration Environment to Support Distributed eXtreme Programming*

Supervisor: Professor Cornelia Boldyreff, University of Lincoln

The Problems Posed by eXtreme Programming: The combination of a lightweight process and important interactions in XP creates a strong implication for the requirement of collocated collaborators. This is not always practical, in particular the end-user may not be located near the software engineering team. Previous attempts to distribute XP have failed to adequately solve the problem in one of two ways.

– Some systems, such as MILOS [1] have failed because they are based as an ad. hoc. integration of existing tools rather than a system specifically designed for the purpose.

– Other systems, such as Joto and Rito-Silva' s "adaptive workflow" [2] have failed because they do not support all the crucial features of XP, such as pair programming.

Hypothesis: The key principle of this research is that it is both desirable and possible to create a system to support the distribution of XP so that there is no degradation of productivity.

Proposed Solution: The proposed solution for this problem shall be based on development of support for the key interactions within distributed XP: daily meetings involving the customer, pair programming and continuous integration. The solution must also support the communication and awareness overheads created by distributing XP. The entire support environment for distributed XP shall be developed as a plug-in for the Eclipse IDE.

Evaluation: The goal of this research is to develop a system that allows the distribution of XP without a degradation of productivity. Within XP the best metric for productivity would be project "velocity' " (the rate of conversion from desired features to delivered features). The two key areas of success for a system of this nature are communication and awareness, both of which are quantifiable and whose contribution to project velocity can be assessed.

References

[1] F. Maurer and B. Dellen and F. Bendeck and S. Goldmann and H. Holz and B. Kötting and M. Schaaf. Merging project planning and web-enabled dynamic workflow technologies. In IEEE Internet Computing, May 2004

[2] Ricardo Jota and António Rito-Silva. Supporting Distributed Extreme Programming with Adaptive Workflow. In Automated Software Engineering: Proceedings of the Workshop on Cooperative Support for Distributed Software Engineering Processes, September 2004

RE-flections

Old English Requirements Met

… on him byrne sc•n,
séaronet séowed smíÞes orÞáncum
'…on him a mail-coat shone,
armour woven by a smith's skill…'

Beowulf, lines 405-6

When on holiday it is pleasant and sometimes helpful to read and reflect a little. I enjoyed some Old English in the form of an account of a hero's deeds by an unknown, but wonderful, poet. In those days – the poet was writing in about AD850 of wilder times a few centuries earlier – basic requirements for survival included swords and armour. A Byrnie or coat of mail may seem an unremarkable piece of equipment today – its modern equivalents are perhaps flak jackets and body armour. But in a time when people had few possessions and little in the way of technology, a shirt that could save your life in a fight must have been a prized object indeed.


Image © Nathan Tudor Armoury 2004

English has changed quite a bit in twelve centuries, but most of the words in the lines quoted above should need little explanation. By the way, the 'y' in ' byrne' is pronounced 'ü', and the 'sc' in ' scan' is pronounced 'sh' as in its modern equivalent, 'shone'.

'searo' may be recognised by Tolkien readers as the root word in 'Saruman', the cunning man or wizard. 'searo' seems always to be used in the context of knowledgeable use of technology – not necessarily magical, but certainly clever and possibly dangerous.

Describing a piece of armour as a 'searo-net' says that it is a crafted mesh, valued for its properties. As the poet writes, it is

'héard hóndlocen, hélpe gefrémede'
'hard, hand-linked [mail-shirt] –– help afforded'.

Beowulf, line 551

We may notice that hardness (a required quality), hand-linking (a design, and means of production), and affording help (a function) are here elegantly and tersely combined.

We can also enjoy, perhaps, the strong rhythm of the contrasted half-lines, noting that alliteration of the stressed syllables takes the place of rhyme; and that intentional obscurity of word-coinage was appreciated by the audience. For instance, the poet refers to a mail-coat only by its properties – being hard and hand-linked (line 551), and talks of 'weaving' or perhaps even 'sewing' an armoured shirt (line 406), though word-meanings shift over the centuries – and sometimes much more quickly: it is dangerous to guess meanings from apparently familiar words. Obscurity and guessed meaning are features of poetry that we do not want to reproduce in our requirements.

'smiÞ / smith' (the letter Þ is the modern 'th', as in 'Þe olde tea shoppe' by the way) is plainly the Old English for engineer, the person who designs and makes things.

'orÞancum' contains the root of the modern word 'think'. Tolkien named Saruman's tower and fortress Orthanc or 'cunning mind'. In Beowulf it seems to mean ingenious use of knowledge, applied skill or artifice. The use of a specialised vocabulary, which on the surface appears to consist of ordinary words, is just as much a challenge for analysts trying to understand a new domain as for readers of poetry. And reading someone else's interpretation or translation, even a good one, is no substitute for seeing the original text or stakeholder.

The finished product, the hero's shining Byrnie, the 'séaronet séowed — smíthes ortháncum' is an ancient vision of requirements put perfectly into practice. Safety-critical ones, at that.

© Ian Alexander 2004

Special Guest Proverb

This issue's proverb (one of my favourites) may need to be prefaced by explaining that in stone-masons' yards and gardening centres, almost any kind of rock except slate, white limestone, flint, clay or soft sand (yes, geologists think clays and sands are rocks) is called 'granite'.

That is properly a crystalline igneous rock composed entirely of quartz, feldspar, and mica. It is (of course) not to be confused with quartz-syenite, granodiorite, rapakivi-porphyry, plagioclase-porphyry, or (naturally) any kind of metamorphic rock such as granite-gneiss or mica-schist. I hope that's clear.

So…

'Geologists take NOTHING for Granite'
(T-shirt slogan)

And of course, we shouldn't take anything for granite when trying to understand people's requirements either. Been there, done that, got the T-shirt.

RE-Creations

To contribute to RQ please send contributions to Ian Alexander (ian @ scenarioplus.org .uk).
Submissions must be in electronic form, preferably as plain ASCII text or rtf. Deadline for next issue: 15th March 2005

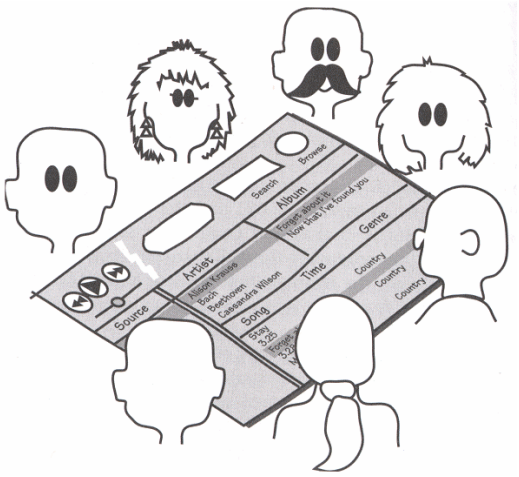
RE-Publications

Requirements-Led Project Management

Discovering David's Slingshot

Suzanne & James Robertson
Addison-Wesley, 2004
ISBN 0-321-180629

This elegant and informative book is the follow-up to the Robertsons' wildly successful Mastering the Requirements Process (1999) (MRP). It isn't very often that a review finds he can't put a technical book down, but it happened this time. Every page is full of wisdom gained by experience: and more than that, it comes from "observing our clients, participating in project teams, and listening to the wiser people in our industry."



"Never go to a meeting without a prototype"

This book is the product of seriously good consultancy over "more than a quarter century"; and that is supported by beautifully clear writing, not to mention James Robertson's fresh and witty illustrations.

Pundits and textbook authors have been arguing for well over a decade that projects absolutely must straighten out their requirements, or they're bound to go wrong. So why haven't project managers listened?

Two obvious reasons spring to mind:

- we're preaching to the choir, not to the sinners outside: i.e. managers don't read requirements books;
- our sermons haven't spelt out to managers and decision-makers what they'll get back if they do their requirements better.

The Robertsons address managers directly in their own language. What is the Return on Investment (RoI) of requirements? Chapter 2 talks all about it. What should managers do, given that time and resources are short? Each chapter ends with practical sections

- What Do I Do Right Now?
- What's the Least I Can Get Away With?

That doesn't sound like High Priest-speak, with the swish of flowing robes barely audible through the jargon. It's the sound of helpful companions who know that the job of running a project is stressful.

Chapter 3 looks at Project Sociology. I had better confess that I get a mention for having worked with Suzanne on the Onion Model of project stakeholders. But the chapter does far more than explain the layers of the onion and the roles involved; the discussion ranges effortlessly over the team structure needed for decision-making, like the Botswanan Kgotla, and Belbin's wonderfully practical analysis of team roles. The point is that nobody can do everything, but a team can. Teams are built, says Belbin, of people such as creative and imaginative 'plants'; confident 'coordinators'; cooperative 'teamworkers'; conscientious 'completers'; single-minded 'specialists', and others. You need to choose people who are not only eligible for a job, but suitable, having the right personality attributes. And the chapter shows how to put all this knowledge to good use.

Chapter 4 turns to the core of requirements work: discovering what people need. This touches on the processes described in their MRP book, but with a different slant: what do you need to ensure happens on your project. "Requirements are free if you pay for them" is one of the paradoxical pieces of advice and wisdom.

Chapter 5 has the provocative title 'Inventing Requirements'. Of course it's something you're not supposed to do; Colin Potts wrote a famous paper on 'Invented Requirements and Imagined Customers'. But the authors argue convincingly that analysts should invent; engineering is after all about shaping the world we live in. Nobody, they argue, asked for the printing press, or spreadsheets, or the Web, or PDFs or MP3s, but these things have found markets for themselves. As usual, the arguments are lively, and made with well-chosen anecdotes and illustrations (in words, photographs, and drawings).

Chapter 6 is called Requirements Simulations, but it's about something much wider than simulation modelling: the use of prototypes, scenarios, hi-tech and ultra-low-tech ways of giving people an idea of what you might build -- always with a view to catching more and better requirements. Prototypes can be drawings on a whiteboard; playfulness is encouraged. Talking like this takes courage, and doing it successfully demands experience.

Chapter 7 manages yet another oxymoron: Requirements for Existing Systems. Most projects are upgrades of existing systems, and most requirements are for enhancements. Yet, most textbooks have blithely assumed simple 'green-field' developments, unconstrained by awkward facts like existing

interfaces, legacy code, and user expectations. So it's really nice to see a detailed and clear process for handling change projects.

Chapter 8 looks at requirement metrics -- a vital tool for management. Whether you believe in Function Points or not, you can hardly avoid having to estimate the effort and cost of your projects, and there's usually nowhere else to start from than the requirements.

Chapter 9 looks at managing the requirements themselves; and the people who deal with them. This is certainly a much broader look than is usually hidden in the acronym 'RM': but it's probably far more realistic. What's the least that can be got away with? A list of requirements deliverables, suggest the authors -- all the classes of knowledge involved.

Chapter 10 is about 'meta-management', such as of the interfaces between projects. This higher-level stuff is crucial, but rarely mentioned; it's the bigger, systems engineering picture that seems to be left out of management school courses.

Finally, Chapter 11 focuses on 'Your Requirements Process': again, covering some of the ground of MRP, but from a different perspective: not doing the tasks, but choosing which set of tasks, which process to use.

This is a book with a grand scope but a practical purpose. It is well-conceived and beautifully produced. Let us hope that managers read it and adopt some of its many suggestions: they should.

© Ian Alexander 2004

Book Review: Discovering Real Business Requirements for Software Project Success

Robin F. Goldsmith
Published by: Artech House, 2004
ISBN 1580537707

"You folks start coding.
I'll go find out what the requirements are." (page 31)

Ah yes. With humour along the lines of Fred Brooks' *Mythical Man-Month*, it is at once clear that Goldsmith knows what he's up against: requirements are seen as obvious, trivial, small bits of documentation that can be left for an idle moment... and projects break time after time as a result. The challenge isn't to do anything terribly complicated; it's to get something simple done properly. As Richard Stevens used to say, "Making good wine is simple, but not easy".

Goldsmith, an experienced consultant, enters a crowded market with a distinctively different requirements book: on discovery. There are very few other books on requirements discovery as such, though many touch on it. Ellen Gottesdiener's *Requirements by Collaboration* is certainly a discovery phase book, though it largely focuses on use case workshop techniques.

But perhaps discovery (despite the title) isn't the half of it. Much of the book is about what Goldsmith calls "testing" the requirements (perhaps "validating" would have been a happier choice):

Moe: I know 20 ways to test business requirements.
Joe: Well, there's the regular way...
Moe: 21 ways! (page 17)

The two things are of course tightly coupled -- there is no such thing as a tidy waterfall going from discovery to documentation to validation: these things go along together. In fact Goldsmith eventually lists no fewer than 64 techniques, of which 11 are specifically about "revealing overlooked requirements" (which itself is about validating a requirement set), and the rest are about "testing" or validating the requirements, though in the process they'll often help to discover requirements and fix requirement weaknesses.

Chapter 1 looks at the disconnect between what everybody says: that requirements matter a lot; and what everybody does: they work on projects doing the wrong things and doing things wrong because the requirements were poor.

Chapter 2, The Regular Way, looks at traditional review techniques. The hint in the chapter title is, plainly, that there is a whole world of exciting and better alternatives.

Chapter 3, Real requirements, goes into the "as is" versus the "as should be" (what Michael Jackson calls the indicative and optative moods), and takes a critical look at the Business vs System requirement distinction.

Chapter 4 looks at evaluating the form rather than the content of requirements. Goldsmith rightly points out that rules don't apply evenly across the board; he'd surely agree with Søren Lauesen's *Software Requirements - Styles and Techniques* that different forms are needed for different situations -- a security requirement looks very unlike a reliability one, for instance. So the different "tests" -- magic words, ambiguity, being positive, etc, are guidelines rather than absolutes.

Chapter 5, Discovering real requirements, begins by observing the commonplace, that users never know what they want. Requirements, in other words, cannot be captured, but must be invented, hunted down as Detective Columbo hunts villains by piecing together the available evidence.

Chapter 6 seems in some ways to be the heart of the book: the Problem Pyramid. This is a simple but powerful technique - something like a template - identifying the problem, the measurable benefit before and after, the cause (as is) of the problem, the what (should be), and the how (system spec, design). This is good stuff, simple and effective, and not I think quite like anyone else's approach (though it contains hints of Tom Gilb's Gist and so on, and of Suzanne Robertson's Fit Criteria).

Chapter 7 is on Applying the techniques.

Chapter 8 is called Data gathering, but actually it' s much better than that: it' s about requirements discovery, elicitation, detective work, whatever. It covers a wide range of techniques from surveys and literature searches to prototyping, JAD, observation and work experience, and a detailed chunk on interviewing. The advice is supplemented by solid warnings (surely based on experience) and a worked example. An excellent chapter.

Chapter 9 is called Formats for analyzing requirements, but it' s about the need to find out what we don' t know, which means becoming aware of it, a paradoxical thing at best. This is very much a fresh take on the familiar types of analysis diagram, plus some that might be less familiar, such as cause-and-effect graphing (a simple and elegant way of drawing business rules). The different techniques each give another angle on a problem, an approach that Goldsmith calls "taking a CAT scan" from the complex medical imaging technique that reveals hidden details in three dimensions.

Chapter 10, Key to completeness, looks at that most difficult attribute, whether a set of requirements is in fact good enough to go. One way is to document end to end scenarios using swimlanes to give a "customer view" (he means operational stakeholder). This is obviously essential (in some form): omitting it is a major cause of project failure.

Chapter 11, Formats for documenting requirements, dives into IEEE Std 830-1998 (which you might think a not particularly wonderful standard); then looks at Use Cases ("today' s most commonly favored format for documenting requirements"); and then the more traditional "hierarchical itemized deliverables" that constitute the "as should be" part of the Problem Pyramid. These are supported by a dataflow model of the high-level conceptual design.

Chapter 12, Finding overlooked requirements, suggests a list of "tests" such as whether you' ve forgotten any interfaces or quality requirements, whether you' ve checked to see if you conform to laws and regulations, and so on. These are all sensible, corresponding to the use of templates for NFRs, and stakeholder modelling.

Chapter 13, Checking requirements accuracy and completeness, proposes another twenty "test" methods, many of which are Sommerville & Sawyer-like good practice guidelines. The chapter covers a great deal of ground in a few pages, and some of the "tests" scuh as "balancing conflicting requirements trade-offs" deserve much more than the half-page that they get.

Finally, Chapter 14, Measuring [the] proof of the pudding, goes into cost-benefit analysis and other checks of whether you have got, or will get, what you want. One is to "define system requirements". Of course whole volumes have been written about that; but perhaps that' s the point: people dive into specification and design before the problem is properly characterised. The hard bits are the easy bits, and vice versa. A little more attention to discovery and validation would be a good thing.

All in all, this is a very welcome book. It is practical and down to earth, to the extent that there is no bibliography, though in fact a score of references are scattered through the chapters. It is not that Goldsmith is not aware of the literature; rather that from a practitioner' s viewpoint, he mostly sees little need to refer to other sources. The exceptions are to make a specific point, and they have an industrial feel – the Standish Group (in the shape of the much-quoted Chaos report), Fagan (inspections), and the IEEE software engineering "body of knowledge".

If you are involved in creating or reviewing requirements in industry, this book is essential homework. It' s not too long, and every page contains practical and worthwhile advice. Buy a copy now.

Choosing and Using Statistics

A Biologist' s Guide
by Calvin Dytham
Published by Blackwell 2003
ISBN 1405102438

As I said in my review of Mathematics Handbooks (RQ 33), I' ve felt a need for some time for a clear and up-to-date guide to the maze of statistical techniques that one is confronted with when trying to analyse data – such as the results of a questionnaire or an experiment.

There *are* some statistics books meant for engineers, but all of them that I' ve looked at so far have been forbiddingly unhelpful: basically you have to know what you want to do before you start, and that' s precisely the problem that needs to be solved. The ' here' s a method, and these are the equations for it' sort of engineering approach is no use. This is firstly because the reader needs to know the criteria for preferring one method over another in a given situation; and secondly, because there is now so much excellent software on the market (much of it available as well-crafted freeware or shareware – see e.g. http://members.aol.com/johnp71/javasta2.html) that the task is hardly likely to consist of following a complicated set of equations with a spreadsheet, programming language, calculator, or pen-and-paper as the older books implicitly suggest.

Therefore it comes as a breath of fresh air to read

> "most students do not really care how or why the test works. They do care a great deal that they are using an appropriate test and interpreting the results properly. I think that this is a fair aim to have for occasional users of statistics."

That is the voice of an expert who realises, albeit with sorrow in his heart, that users have a different and valid point of view: their specifications concern the

what, not the how, the results not the mechanisms. Of course there's a lesson in there for requirements people too. It's perhaps illuminating that this degree of insight comes from a book intended for the relatively technophobic biology student or researcher, rather than for engineers.

Happily, the truth is that a statistical technique remains invariant regardless of who applies it. In the old days, technical authors were advised to write for the Geologist, someone who was well-informed in his own field, but only an intelligent layman in whatever was being written about. So perhaps a book for biologists should have just the right tone for non-statisticians of any profession.

The heart of Dytham's book is a Key to statistical techniques. It occupies the whole of Chapter 3. Each question takes about 8-12 lines of text, sometimes even more, and there is often simple advice about whether the relevant technique is any better than similar options.

Each leaf node of the tree of questions leads to a section in the rest of the book. A typical statistical method, such as the t-test, gets several pages of coverage. First the purpose of the method is described. Then an example with dummy data is provided. Then instructions are given for running the test using three popular statistics packages: SPSS, MINITAB, and Excel. The introduction explains that this was the same set as in the first edition; for the second edition, the author considered Systat, Genstat, SAS, Statistica, S/S-plus-R and GLIM, but "there was surprisingly little consensus on the packages to add" – so he didn't add any. This was a wise choice: the book would have become unwieldy (and expensive) without becoming any easier to use.

I tried some of the techniques with Excel (as I had it already) and then looked around for a suitable package for the more advanced techniques that I needed. The main commercial packages work out at over $1000 per seat, so I soon found myself looking at cheaper alternatives. One that is easy to use, free, and remarkably powerful is PAST, aimed at Palæontologists! – but none the worse for that. The truth is that once you know what you're trying to do and which methods you should use, getting a tool to work is mostly a matter of shovelling the data into the right-shaped heap (and no equations in sight).

The book is introduced (Chapter 1) with a simple 'Eight steps to successful data analysis' recipe, which consists of planning, planning, and planning -- it's much nicer to design your survey or experiment to be easy to analyse with a good chance of getting a clear result, than to pore over a heap of possibly-meaningless data afterwards. Of course, human nature being what it is, statisticians and stats books often have to face up to the latter situation.

Chapter 2, 'The basics', explains in a non-patronising way what essential concepts such as observations, hypothesis-testing, P-values, sampling, experiments and even statistics really mean.

There follow chapters on: Hypothesis Testing, sampling and experimental design; Statistics, variables, and distributions; Descriptive and presentational techniques. These contain simple and good advice for beginners who want to get clear results and to present them plainly.

The body of the book is essentially a list of descriptions of techniques, indexed by the Key. The chapters are: tests to look at differences; tests to look at relationships; and tests for data exploration.

The book closes with a list of statistical symbols and letters, a glossary, the assumptions made in the tests, and some hints and tips (like not using 3-D graphics effects to tart up your graphs). There's a summary classified table of tests, and a decent index.

This book met my needs (and ended quite a long search). I can recommend it to anyone who's trying to design a way of collecting evidence or experimental proof but has not much idea how to do the statistics. Much more advanced texts exist; Dytham recommends Zar's *Biostatistical Analysis* or Sokal and Rohlf's *Biometry*. No doubt there exist excellent advanced texts for engineers too. But for the rest of us, Dytham is a splendid companion.

RE-Sponses

RQ welcomes comments and reactions to articles and reports published in its pages.

RE-Sources

Books, Papers

For a full listing of books, mailing lists, web pages and tools that have appeared in this section in previous newsletters, see the 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:
http://www.cc.gatech.edu/computing/SW_Eng/resnews.html

IFIP Working Group 2.9 (Software Requirements Engineering):
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:
http://www.volere.co.uk

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

Mailing lists

RE-online (formerly SRE):
http://www-staff.it.uts.edu.au/~didar/RE-online.html

The RE-online mailing list acts as a forum for requirements engineering researchers and practitioners. To subscribe to RE-online mailing list, send e-mail to majordomo@it.uts.edu.au with the following as the first and only line in the body of the message:

subscribe RE-online <your email address>

LINKAlert:

http://link.springer.de/alert

A free mailing service for the table of contents of the *International Journal on Software Tools for Technology Transfer*.

RE-Actors: the committee of the RESG

Patron:

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

Chair:

Dr Pete Sawyer, Lancaster University, Computing Department,
sawyer @ comp.lancs.ac.uk.

Vice-Chair:

Dr Kathy Maitland, University of Central England,
Kathleen.Maitland @ uce.ac.uk.

Treasurer:

Prof. Neil Maiden, Centre for HCI Design, City University,
N.A.M.Maiden @ city.ac.uk.

Secretary:

David Bush, National Air Traffic Services,
David.Bush @ nats.co.uk.

Membership secretary:

Dr Juan Ramil, Computing Department, The Open University,
J.F.Ramil @ open.ac.uk.

Newsletter editor:

Ian Alexander, Scenario Plus Ltd.,
ian @ scenarioplus.org .uk

Publicity officer:

William Heaven, Department of Computing, Imperial College,
su2 @ doc.ic.ac.uk

Regional officer:

Steve Armstrong, Computing Department, The Open University,
S.Armstrong @ open.ac.uk.

Student Liaison Officer:

Carina Alves, University College London, Department of Computer Science,
c.alves @ cs.ucl.ac.uk

Immediate Past Chair:

Prof. Bashar Nuseibeh (The Open University)

B.Nuseibeh @ open.ac.uk

Industrial liaison:

Dr. Wolfgang Emmerich, University College London,
W.Emmerich @ cs.ucl.ac.uk.

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

Gordon Woods, Independent Consultant,
Gordon @ cigitech.demon.co.uk

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