BCS™ Requirements Quarterly

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

© 2006 BCS RESG http://www.resg.org.uk RQ40 (June 2006)

Contents

RE-Soundings

From the Editor

In this issue, we report on the RESG's Problem Frames event. We can claim to be the spiritual home of the Problem Frame, as Michael Jackson is our Patron and both our ex-chairman and several of our committee are at the Open University.

The excellent REFSQ workshops are graduating into conferences in their own right: another sign of the increasing maturity of RE as an established field.

We continue the series on requirements at different institutions with an article by Neil Maiden on Requirements at City University, London. You'll be left in no doubt that City is serious about RE and deserves its reputation as one of the world's leading RE universities.

RQ is delighted to welcome three scholars from Carnegie Mellon, demystifying customer requirements.

We also have some treats in store for you on Safety Requirements, books new and old, and the curious question of whatever DID people do when they didn't do requirements? RQ hopes you will be provoked into writing to (or for) the next issue, and is pleased that the last issue triggered a response for the letters page. Keep the letters coming!

Ian Alexander,
Scenario Plus

Chairman´s Message

One of the regular highlights of my year is going to REFSQ - the annual workshop on RE and quality. At least, it used to be a highlight but this year it turned into a working conference, which turned out to be just as much fun. In many ways, REFSQ is RE's (as in the IEEE International Conference on RE) younger sibling. It comes earlier in the year and provides a forum for presenting innovative work-in-progress. As such, it's a good barometer of what people are obsessing over.

Of all the themes that emerged at REFSQ this year, the most pervasive was the absolute need to make sure our innovative tools, notations, processes, etc. can deal with industry-scale problems. This theme is hardly new

and perhaps the reason it keeps recurring is that it's hard.

It's not usually hard at a technical level - with a bit of mutual sympathy, academic researchers and practitioners can generally get along and communicate quite effectively. It's the other stuff that tends to get in the way. This typically boils down to two things. The first is the time and effort commitment needed by practitioners for whom the research benefits are typically tangential to their core task. This isn't peculiar to RE and where there's a will, it can be solved. The second thing does appear to be a particular issue for RE. It is the sensitivity of the data that the academics want to examine and analyse.

The guiding rule appears to be common sense: case studies should never be run on confidential or controversial issues. Of course, where people work in sensitive or regulated domains, such as health, it might be hard to isolate the stuff that isn't controversial but even here, a way can usually be found through the trust and confidentiality minefield. By the end of the two days of REFSQ'06, I think we all felt uplifted by the mutual understanding of this issue that had emerged between the academic and industrial people participating.

Finally, make sure you mark the 12th July 2006 in your diaries so that you can come along to hear Bashar Nuseibeh give our distinguished speaker talk at Imperial College (details inside). It gives all of us on the committee enormous pleasure to invite Bashar to give this talk, not least because he was one of RESG's founders and Chairman for its first 10 years. Now Professor of Computing at the Open University and internationally known for his work on RE and the relationship between RE and architecture, and latterly on security requirements, it promises to be a really fascinating talk.

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:

RESG Student Poster Competition: Win an iPod!

12th July, 2006, Imperial College, London

The Requirements Engineering Specialist Group of the British Computer Society will be holding a poster competition on the 12th of July at Imperial College, London.

We're looking for undergraduates, PhD students and RAs alike to demonstrate their requirements engineering related work in the form of an A0 poster. The posters will be judged by everyone attending the event : The winner will receive a 2GB iPod nano and the runner up will receive an iPod shuffle.

The competition will be held alongside the Distinguished Speaker Event of the RESG. You are invited and encouraged to attend this also. Details can be found at the RESG website, http://www.resg.org.uk/

To enter the competition please send your completed posters, in PDF format, to a.stone@comp.lancs.ac.uk by Monday the 3rd of July. This will give us a clear indication of the number of people attending, as well as allowing us to check the posters are suitably RE-related. Exact details of the time, room number and location of the event will be sent to you later, although it is expected that poster demonstrators should arrive by approximately 1pm.

The layout of the poster is up to you - do whatever best demonstrates your work. Remember to keep your posters clear and concise and use a variety of layout tools, such as colours, numbering and spacing to make it easy to understand. Advice, including suitable font sizes, can be found here :

http://lamswww.epfl.ch/conference/re06_poster/poster_guideline.pdf

Please note that YOU are required to print and bring your poster to this event.

Your target audience is a mix of both academic researchers and industrial practitioners; this competition represents an excellent way to inform the UK requirements engineering community about your work.

We look forward to seeing you on the 12th of July.

- Andrew Stone, RESG Student Team

AGM and Distinguished Speaker Event

12th July 2006, Imperial College, London

2:00pm AGM and Election of Committee

2:30pm Distinguished Speaker Event: Professor Bashar Nuseibeh (Computing, The Open University) “The Problems of Security”

Abstract of Professor Nuseibeh’s talk:

The proliferation of computers in society has meant that valuable business and mission critical assets are increasingly stored and manipulated by computer-based systems. The scale of misuse of those assets has also increased, because of their worldwide accessibility through the Internet and the automation of systems. The Security Engineering community has therefore developed a variety of techniques for protecting computer-based information. What this community has recognised as necessary, but still lacking, is a systematic process of eliciting, specifying, and analysing system and software security requirements.

Security requirements describe the need to protect assets from harm, typically in the form of constraints on other functional requirements. However, they are often left imprecise, or ignored entirely in favour of specifications of the mechanisms, such as encryption or access control, for implementing security in particular systems. This prevents early yet rigorous analysis of security goals, and can lead to the development of systems that miss some key security concerns and that are therefore vulnerable in the face of unexpected threats.

The Open University's Computing Department has an active research programme in Security Requirements Engineering, who focus is on analysing security problems, with a view to eliciting and documenting security requirements, and relating those requirements to architectural specifications and designs. This talk reviews some of the key challenges in security requirement engineering, and provides an overview of a range of research projects in the area. These include:

- modelling access policies by relating roles to their organisational context,

- analysing security threats using abuse frames,

- using satisfaction argumentation to validate security requirements, and

- managing privacy requirements in ubiquitous computing.

Bashar Nuseibeh is Chair and Director of Research in Computing at The Open University (OU), and a Visiting Professor at Imperial College London. His research interests are in software requirements engineering and design, software process modelling and technology, and technology transfer. Bashar is currently holder of a Royal Academy of Engineering and Leverhulme Trust Senior Research Fellowship. http://mcs.open.ac.uk/ban25/

IEE/RESG Requirements Events For Systems Engineers

Introduction to Requirements

2nd October 2006, IEE, Savoy Place, London: 1-day training course (Ian Alexander)

http://conferences.theiet.org/itr/course.htm

Answering the Six Questions about Requirements

3rd October 2006, IEE, Savoy Place, London: 1-day Seminar of 8 talks covering all the essentials of requirements work:

- Who: stakeholder analysis (Ian Alexander)
- What: goal modelling (Nadia Amin)
- Where: scenario modelling (Neil Maiden and Alistair Mavin)
- Why: rationale analysis (Simon Buckingham Shum)
- How: making requirements verifiable (Suzanne Robertson)
- When: triage and prioritisation (Andrew Farncombe)
- With What? Requirements tools (Kathleen Maitland)

http://conferences.theiet.org/itr/seminar.htm

Members' discount for RESG members at both events

Early Aspects

October 2006, Lancaster

PhD Day

6th December 2006, Lancaster University

A whole day specially for PhD Students researching any aspect of Requirements. Previous events in this series have been a riotous success! – Ed.

The RESG Student representative writes:

Dear All,

Following on from the success of our previous PhD day the RESG are pleased to announce a successor event to be held at Lancaster University on the 6th of December 2006.

This is a unique opportunity for requirements engineering researchers from all over the UK to meet and share advice, stories from the trenches and the results of their research to date. The event is divided into three sections :

1) Current research - an opportunity for students to showcase their work to date. For students early on in their research this session will provide ideas for issues such as how to scope your work, where to find ideas and supporting evidence and how to present these ideas concisely.

2) Mechanisms of PhD assessment - invited speakers will present an overview of the assessment process, advice about writing up and the main attraction : a mock PhD viva. This is an ideal chance to see what really goes on in the assessment room and the hidden mechanisms that examiners use.

3) Share the pain - Attendees state something that has been a real problem for them during their research. Common themes are explored and discussed. To end on a positive note everyone provides one piece of advice to put into the communal knowledge-share.

Registration for this event will open shortly.

- Andrew Stone, RESG Student Team

Contact Andrew Stone a.stone1 @ lancaster.ac.uk

or Pete Sawyer sawyer @ lancaster.ac.uk

RE-Calls

Recent Calls for Papers and Participation

RE'06

14th IEEE International Requirements Engineering Conference

Minneapolis/St Paul, Minnesota, USA

11-15 September 2006

The world's leading RE event is nearly upon us. This year the conference is taking place in the heart of the USA's automobile country, so we hope there will be a strong and practical industrial flavour. Along with that are academic papers selected ruthlessly from an increasingly large number of submissions, and an excellent programme of invited and keynote talks.

The conference programme is accompanied by a varied mix of tutorials and workshops, both research and industrial. Come along and join in!

www.re06.org

INCOSE EuSEC 2006

European Systems Engineering Conference 2006

18-20 September 2006, Edinburgh Conference Centre, Heriot -Watt University, Edinburgh, Scotland

The next International Council on Systems Engineering (INCOSE)'s 5th biennial EuSEC conference focuses on the effectiveness of interdisciplinary and multicultural collaboration.

Systems engineering has proved its ability to add value to enterprises seeking to cope with increasingly omplex and dynamic environments and in particular the integration of diverse technologies and disciplines to create breakthrough products and dependable systems, and more effective collaboration in international multi-cultural enterprises and projects. This conference will explore how these objectives can be achieved, and what are the pitfalls to be avoided. The program will provide unique networking opportunities with world-class experts.

www.incose.org/eusec2006

Inquiries to the General Chair, Paul Davies
paul.davies@uk.thalesgroup.com

RE-Readings

Reviews of recent Requirements Engineering events.

Problem Frames

The Open University, Milton Keynes, 10th May 2006

The symposium was perfectly framed - sunny and pleasantly warm weather and clear directions to a brand new Michael Young building where our welcoming hosts were waiting for us.

The morning tutorial on Software Development Problem Frames was delivered by the man who came up with the idea in the first place – **Michael Jackson**. As the audience was a mix of practitioners and academics, Jackson explained Problem Frames in a way which did not assume much prior knowledge and it was well paced and easy to follow. For the uninitiated, a Problem Frame (PF) is a characterisation of a certain problem for which a software solution is required. A requirement is a condition on the problem world, not a condition on the machine. [Yeah, but engineers often use 'requirement' for the latter, as it's machines they're interested in. –Ed.]

We ask *what*, we ask *why* but we rarely ask *where* is the problem. PFs help to locate the problem in the physical world. Jackson strongly emphasised that the PF approach is not a cookbook but an intellectual framework for thinking about the problems and solutions in software engineering. In the second part, he focused on problem decomposition into subproblems (set of certain features) and subproblem composition concerns (how to compose certain subsets of features).

The afternoon session gave a snapshot of current Problem Frames research and practice.

Jon G. Hall (Centre for Research in Computing, The Open University) talked about "Practically perfect Problem Frames". Using the example of a sluice gate problem, Hall explained how to develop problems using transformations. He pointed out that a risk is a property of a transformation and needs justification.

The way we handle risk is influencing how we make progress with the problem we are trying to solve. Although PFs are an approach to early life-cycle software engineering, according to Hall, it is "generally accepted that choice of solution can influence problem development". He told us that moving into the solution domain, including human, software and organisational specification, will expand the applicability of PFs.

Christine Choppy (LIPN, Université Paris XIII) told us how the structure provided by PFs is used to develop UML descriptions. In her talk entitled "Problem Frames and UML description development" she explained how they are using PFs to match architectural styles, design patterns and code patterns. In this context, PFs are treated as problem patterns as they classify software development problems. In a

couple of case studies she described how five basic problem frames (transformation, commanded behaviour, commanded information, rich workpieces) are mapped onto UML description (Domain Model, Requirements Specification, Design Specification). She concluded that if the modelling of the domain and of the problem is accompanied by their "framing" this helps the developer to manage complexity and offers support to navigate the complex UML models.

Since questions were being raised throughout the tutorial and afternoon talks there seemed to be not much left for the panel session. Nevertheless, the captivated audience had more discussion topics in store for Christine, Jon and Michael and we had lively and provoking panel session which ended this successful and well attended RESG meeting outside London. Well done Lucia & the OU!

© Ljerka Beus-Dukic 2006

REFSQ'06

The twelfth *Requirements Engineering: Foundation for Software Quality* (REFSQ'06) was in Luxembourg this year. As usual, it was co-located with CAiSE but, in a departure from tradition, it was billed as a working conference rather than a workshop. In practical terms, this meant that attendance was opened up to all-comers so there was a mix of people presenting papers and interested but non-presenting participants. One of the nice things about REFSQ is that it seems to inspire loyalty from people, yet it isn't just a clique of the usual suspects every year. The opening-up helped pull in a good mix of people from those well embedded in the RE research community, to people new to RE research to people from industry with particular and very illuminating views on the problem.

Another innovation this year was a panel session that ran in parallel with the main paper track. This was on the relationship between RE and project management. I didn't attend the panel, but reports were good - unsurprisingly given the two panellists: Sjaak Brinkkemper of University of Utrecht (formerly at Baan) and Barbara Paech of Heidelberg University.

Given the heady pace of all these changes, it was reassuring to know that some REFSQ traditions remained. Most notable among these was that of the baffling first slide. Every year the organisers oblige speakers to instantiate a standard first slide designed to position their work with respect to the others in their session. Every year the speakers produce a range of interpretations of this requirement that the organisers never anticipated. This year, based on feedback from the previous eleven years (we're nothing if not reactive to users' needs), a new obligatory first slide was developed, with exactly the same results as the previous eleven years. REFSQ veterans hugged themselves with pleasure at recognising an old friend.

There's also an obligatory last slide designed so that every paper is concluded in a standard format. Each paper is assigned either two or three discussants who also summarise the paper using the same slide. This simple mechanism is REFSQ's way of gathering different viewpoints on the work, and it generally stimulates good discussion: the whole of REFSQ is designed to get people interacting. The proceedings are deliberately produced post-event so that authors can incorporate the referees' comments and the results of the discussion sessions in the final versions.

I've laboured the format. What about the content? Well, we had thirteen papers (there should have been fifteen but two of the authors were ill) spread over six paper sessions: One on *Quality requirements* and another on *Quality **of** Requirements*, and one each on *Case studies*, *Formal methods*, *Elicitation* and *Complex systems*. As always, they were a fascinating and eclectic mix. The session titles only capture half the story: there were papers on RE for product lines, dependable systems, new methods, old methods in new domains, and on new domains and the problems they pose. Underpinning them all was a concern for real-world scalability. Not everything looks like it will scale, of course, but at least everyone recognises the need to evaluate whether it would.

Two days of intense discussion and exploration of the fascinating, but hilly, old city of Luxembourg left everyone drained of energy but keen to resume the fun at REFSQ'07.

© Pete Sawyer 2006

RE-Papers

Requirements for Socio-Technical Systems: Research at City

Neil Maiden, Centre for HCI Design, City University, London

Requirements for Socio-Technical Systems

Rarely are software systems developed without considering their intended environment. We integrate such systems with other software systems, and people use them in the redesign their work and leisure. Yet the requirements and design processes for such systems have reflect this focus, instead using UML techniques to specify the software system, leaving the wider socio-technical system under-specified.

We are now seeing an increasing focus on the specification of requirements for socio-technical systems. One of the principle challenges facing such work is the need to adopt an inter-disciplinary approach, applying models and techniques not only from software and systems engineering but also from human-computer interaction and the social sciences.

City University's Centre for Human-Computer Interaction Design has been researching requirements engineering since the 1980s. Over last 15 years the Centre has undertaken basic and applied research that has been funded by the UK Research Councils, European Union and organisations including NATS, Eurocontrol and Dstl. It has also actively transferred results from this research into requirements practice. The research tackles 3 main challenges in socio-technical system specification:

1. How to specify new software systems;
2. How to specify the redesigned work resulting from new systems;
3. How to specify the complex interaction between the work and systems.

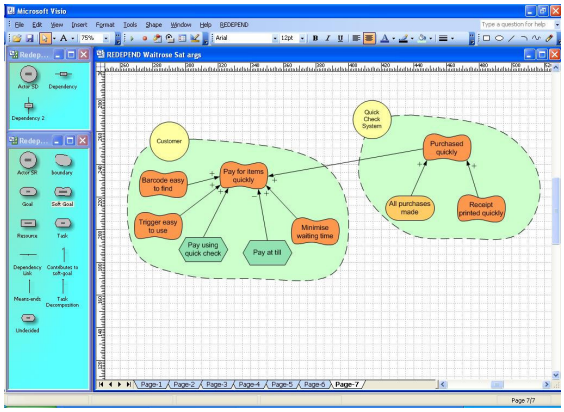
Particular challenges include determining system boundaries, arguing that software system specification can satisfy requirements on the redesigned work, and establishing complete requirements on the wider socio-technical system. Below we demonstrate techniques developed in face of these challenges, and report lessons learned that summarise why these are such neat techniques to use.

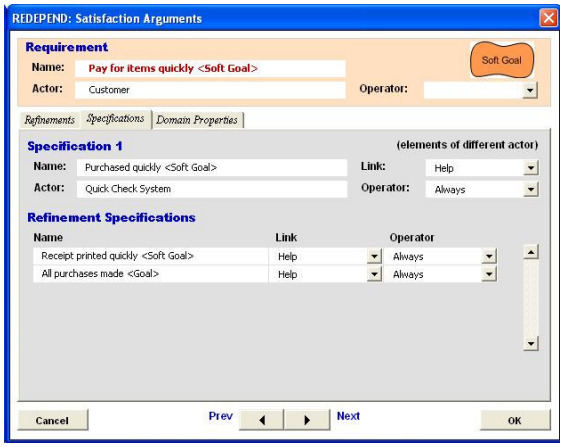
Analysing system boundaries with i*

*i** is a well-established requirements technique well-suited to modelling socio-technical systems. It models such systems as a network of actors that depend upon each to achieve different types of goals, undertake tasks and obtain resources. *i** models have proven to be very useful for thinking about and exploring boundaries of socio-technical systems. Each model expresses and tests boundaries as goal and soft goal dependencies that actors want to achieve. If the depender actor in a dependency relationship wants to attain a goal or achieve a soft goal, and the project will test to determine whether the depender actor attains the goal or achieves the soft goal, then the depender actor is part of your socio-technical system. Conversely, if the project is not interested if the depender actor attains the goal or achieves the soft goal, then the actor is not part of your system. We support such boundary analyses in REDEPEND, an application running on top of Microsoft's Visio application.

Embedding satisfaction arguments into requirements modelling

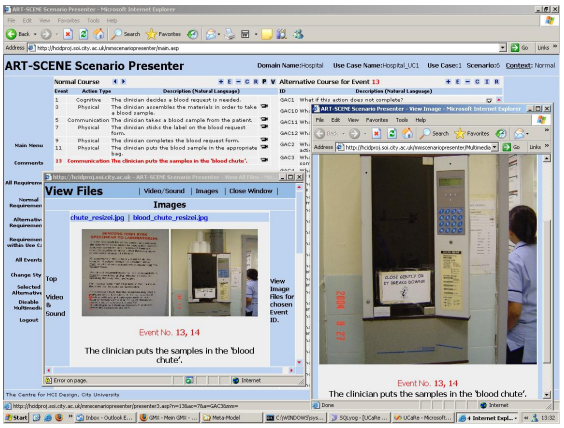
Another challenge is being able to argue that new software systems will bring about the required work redesign in a complex social environment. Jackson's satisfaction arguments are a useful technique here. In recent work funded by NATS, we have extended satisfaction arguments and integrated them with *i** socio-technical system modelling in our REDEPEND environment. This enables analysts to demonstrate, in terms of requirements on work redesign, specifications of new systems, and domain properties, how these new systems contribute to satisfaction of modelled socio-technical system requirements. The REDEPEND screenshots show a large *i** model for a complex socio-technical system for a supermarket payment system, and one structured satisfaction argument that demonstrates the arguments developed for one soft goal – *pay for items quickly*.





Social system requirements

One technique for modelling complete requirements for all actors of a socio-technical system is to establish and walkthrough scenarios of that system's behaviour. Our ART-SCENE environment allows analysts to generate and walk through scenarios that describe normal and abnormal behaviour of the software, human and organisational actors in a socio-technical system. Requirements patterns guide the specification of different requirements types, ranging from training and usability to performance, depending on type of actor's behaviour. Recent rich-media extensions enable ART-SCENE to depict complex work situations using graphics, images and video. ART-SCENE applied to specify a new accident-and-emergency hospital system is shown below.



Concluding

We have successfully applied these and other techniques to specify socio-technical systems in major air traffic management and other domains. We have also investigated these applications to determine how useful and usable these techniques are in these projects.

If you want to know more about these techniques and their application, do not hesitate to contact us via Neil Maiden (N.A.M.Maiden@city.ac.uk).

Demystifying Customer Requirements

Ram Narayan Seshadri, Sathviha RaviArunan, and Tharanian Mahendran are scholars pursuing Master of Science in Carnegie Mellon University at Pittsburgh. Contact them at {sramnarayan, sathviha, tharani.mahendran}@gmail.com

Understanding the customer requirements is a tough job. A simple misunderstanding of the requirements can not only lead to unsatisfactory product but also a project swipe. This is because; only from these understanding the requirements are derived.

This write-up elaborates the issues we faced while gathering requirements for a small academic project.

Our project team consisted a small group of fresher doing masters in software engineering. As part of our curriculum we have to do a 12 month long practicum project for our professor who is our customer. Even though we worked closely with our professor we faced lots of issues initially understanding our customer needs. But after enough ground work we understood the importance of correct requirements gathering.

Initially we had weekly meetings and interactions with our customer. We started with a well structured plan to elicit the requirements by having weekly meetings with our customer. The scribes from the first few meetings were used as the base and we started deriving the requirements from it. We made the vision document of our project based these requirements. Our initial understanding was our customer needed a tool to check the consistency in specifications. The tool must be able to take in specifications in terms of uses cases or any other formal specification language like OCL or Z notations. In case of use cases the tool must allow manual consistency checking by visually displaying the use cases in an interleaved manner. In case of formal specification the tool must automatically check for inconsistencies. A draft of our understanding of vision was made and forwarded to the customer for verification.

Few days later, our customer called for a discussion. The customer remarked that he wanted more than what is described in the vision documented. He said that our understanding of his requirements was incomplete. Then, we brainstormed several times within the team, had several meetings with customer and suggested many features, but the customer was not looking for any of those. There was confusion within the team since we were slipping from the schedule and the project was still in requirements stage.

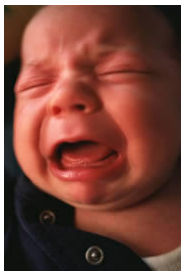
Then we tried to use other techniques like storyboarding and prototyping. We made a requirements prototype of the system that we understood which consisted of only the GUI. This prototype was explained to him in the manner of a storyboard. To our surprise, our customer wanted so many refinements and changes. We had to make six versions of the requirements prototype. Only then did we realize, that our customer needed an extensible requirements tool to support his research in requirements engineering. The tool must be able to incorporate plug-ins to extend its functionalities. For example if we develop a tool which can accept and analyze OCL specifications, it must be possible to develop a plug-in for other specifications like Z or state machines and add it to the tool without changing the code of the tool. Also the customer needed a configurable use case plug-in in which he must be able to change the structure of use-case as needed. For example if the structure of the use case is “Use Case Name”, “Actors”, “Actions” and “Pre conditions”, he must be able to add another element “Post conditions” to the use case structure. Also he must be able to remove elements from the structure.

Therefore, we decided to provide a tool that has Pluggable Environment and a configurable Use Case Plug-in for that environment. By the time we understood this we had already slipped more than fifteen percent of the total schedule. But we understood and learnt many things from this experience.

Brainstorming proved to be an excellent way of developing many creative and radical solutions to our problems. The team members were able to think more freely and move into new areas of thought and so create numerous new ideas and solutions. Each participant in the session shouted out their ideas as they occur to them and then build on the ideas raised by others. All the ideas were noted down by a scriber in our team and were not criticized. Only at the end of the brainstorming session the ideas were evaluated and we came to a consensus as to what is feasible and what is not. Even though brainstorming is a good approach for

eliciting requirements, other techniques are also needed. Although, 90 percent of projects today gather requirements only through brainstorming and meetings, not all the words the customer utters are requirements. The understanding of what he wants is more important.

If not for the storyboard approach we would not have elicited the correct requirements. We would have given the customer what we knew and not what he actually required. Storyboard is a technique borrowed from movie making. It represents the tasks to be done in a pictorial way with a story to drive the discussion. It not only helped the customer to understand what we were conveying but it helped us think like the customer. Prototyping gave the customer a feeling of what the system is going to be and what he can expect as an end result. It also helped in getting our requirements signed off.



A baby cries to announce it has a requirement.
Working out what that is, is your job

We understood the most important part, customer relationships. A customer is like a baby who can only cry when it wants anything. It is the responsibility of the mother (Requirements Engineers) to understand the meaning of the cry and give what it actually wants.

© Ram Narayan Seshadri,
Sathviha RaviArunan,
and Tharanian Mahendran 2006

Requirements Engineering in Software Product Lines

Mark Dalgarno, Software Acumen

(mark@software-acumen.com)

Introduction

Software Product Line Engineering is a systematic reuse approach that offers many benefits to organisations developing multiple software variants. Reported benefits include significant cost savings, faster time-to-market, improved quality, increased productivity and the ability to support a greater number of variants. Software Product Lines have usually made a return on their reuse investment when **at least** 2 - 3 variants have been developed within the Product Line [Clements & Northrop 2002].

This article outlines the approach and describes how Requirements Engineering activities are shaped by the need to engineer requirements that are reusable across the Product Line.

About Software Product Lines

A Software Product Line is a set of software **variants** aimed at a defined market segment. The demand to support multiple variants comes from different sources, for example the need to support target devices with

- different characteristics,
- different technology solutions,
- market differentiation (high vs. low-end products)
- different legal or environmental constraints.

The benefits of the approach arise because, by limiting the Product Line to a defined market segment, it is possible to develop core assets that are reusable in multiple variants in a cost-effective manner.

The Product Line's reusable assets are systematically developed to be configurable in predefined ways to satisfy product-specific requirements. The production of a single product variant involves selecting the relevant set of core assets, configuring these assets and then adding any new product-specific assets, if there are any, to this mix. This production process is often automated using bought-in or custom in-house tooling.

Product Line Scope

The *Scope* of a Product Line specifies what is *in* and what is *out* of the Product Line. Scope typically consists of some description of the features that are common to the products in the Product Line and the features that vary between the products in the Product Line. In this context a **feature** is

> *"a logical unit (for some stakeholder) of behaviour that is specified by a set of functional and quality requirements"* [Bosch 2000],

so a single feature usually captures a set of (related) requirements.

Scope is a key concept in Software Product Lines since it bounds the context in which the Product Line's core assets have to be reusable. Setting the Scope too small may limit the ability of the Product Line to grow or may limit the number of products over which the cost of developing reusable assets may be amortized.

Setting the Scope too large may make it costly to develop or reuse the core assets and the Product Line could disintegrate into a number of parallel one-of-a-kind software development projects. However, it is possible to start conservatively and expand the Scope over time to reduce the risks.

Ultimately, Product Line Scoping is an economic decision involving many trade-offs and varying approaches to this have been adopted depending on the scale or longevity of the Product Line and the nature of the development organisation.

Requirements Engineering in Product Lines

The Product Line's scope acts as a guide for all life-cycle activities including Requirements Engineering: requirements within scope are elaborated further to support architecture and design activities, and generally speaking, requirements out of scope are not considered further. The organisation may also proactively seek out new requirements within the scope or reshape requirements to be within scope as these may be supported more economically than out of scope requirements.

As each requirement is elaborated it is analysed to determine whether it represents a **shared** requirement for products in the Product Line or a **product-specific** requirement.

Shared requirements may further be divided into **non-variable** and **variable** requirements:

- non-variable requirements are reused "as is" across the Product Line,
- variable requirements are configured or parameterised in some way ion the specification of different Product Line variants.

Variable requirements need to be recorded with additional information specifying precisely how they may vary across Product Line variants.

Requirements capture has some additional complexities not found in Requirements Engineering for one-of-a-kind systems. Multiple stakeholders, domain experts and other participants can be involved and to avoid confusion and ambiguity it is important to agree a common vocabulary to describe domain concepts.

Another complexity is that dependencies between requirements are common and can be complex e.g. some requirements may require or exclude other requirements, or may restrict the possible configurations of other requirements (possibly based on some formula). When elaborating requirements, care is also needed to avoid describing them with too little or too much variability, as either can lead to problems later.

Feature modelling, a concise approach to modelling commonality and variability information, helps to capture and visualise Product Line requirements and may also be used to manage other Product Line life cycle activities. Feature models may be combined with transformative or generative approaches to automate the selection and configuration (making choices where there is any variability) of requirements when specifying individual products.

Automated support not only ensures that a valid configuration of requirements is produced, but that it is done quickly, and in some cases without expert configuration knowledge.

Variant-specific requirements can either be used directly in other life cycle activities e.g. to support architecture configuration or product validation, or, in a software supply-chain, they can be passed on to external software suppliers to drive their life cycle activities.

Product Line Evolution

Product Lines, like one-of-a-kind software, are subject to change. Customers, or the Product Line marketing team, may ask for new products or features. If these are within Scope then they may be produced economically already.

However, if they are out of Scope then either the new requirement needs to be negotiated to bring it within Scope or the organisation needs to charge more for development of the out of Scope requirement or the Scope may have to be revised. If the organisation repeatedly receives requests for an out of scope feature then it strongly suggests that the Product Line should evolve to incorporate that feature.

Closing Remarks

Requirements Engineering in Product Lines is more complex than in one-of-a-kind development, due to the need to develop multiple products with varying requirements. There are also more opportunities to get things wrong than in one-of-a-kind software development. However, many organisations have mastered the approach, and are using it successfully to reap the benefits described in this article.

References

[Bosch 2000] *Design & Use of Software Architectures*, Jan Bosch, Addison-Wesley 2000

[Clements & Northrop 2002] *Software Product Lines Practices and Patterns*, Paul Clements, Linda Northrop, Addison-Wesley 2002

Mark Dalgarno is the Lead Consultant for Software Acumen (http://www.software-acumen.com/), a specialist provider of tools and services for Software Product Lines based in Cambridge, England. Mark's experience spans almost twenty years working with a number of organisations, both large and small, primarily in software product development at all levels from programmer to development manager. Mark is a member of the BCS and a Chartered Engineer.

A Brief Note on Requirement Metrics

This was written at the request of a client, but as it's quite general in content, it may be of interest to RQ readers.

Questions

Project Managers often ask "What Metrics should I use to measure progress on my Requirements?"

Business Managers often ask "What is the Payoff from Writing Requirements? Why should we use them anyway?"

Both lines of questioning lead to Metrics on Requirements, but of different kinds.

Measuring Progress

Progress during a development project is not as easily measured as, for instance, maintenance work or productivity on a production line. Those can be measured by number of units produced per day, etc.

Development progress is harder to measure because the end point is not known in advance. When the requirements themselves are unknown, even the effort and time needed to complete the specifications is very uncertain: and haste at this stage is usually rewarded with severe delay and cost overruns later in the project. Therefore, naïve metrics like "Number of requirements written" are of little use.

Better progress metrics measure the rate of change of requirements: when the number of changes per month tails off, the project is approaching stability. Ralph Young, in his book *Project Requirements* (Management Concepts Inc, 2006) suggests that Volatility must be limited to 0.5% per month, ie you have to get the rate of change of requirements down to that level, and keep it there through the design and test phases, for a project to be a success.

Young also suggests other practical progress metrics, from a basic requirements count to the number of requirements validated, traced to design, traced to test cases, etc. These are essentially the "obvious" things to measure, and there is nothing complex about them: you just have to count them.

Payoff

The payoff from writing requirements is so large that Erik Simmons, Process Manager at Intel, states it is "essentially unmeasurable"[1]. A project that goes ahead without requirements is virtually guaranteed to fail: ie, to overspend, be late, and deliver a poor quality product (one that does not do what is wanted).

Simmons suggests that the improvement from even a simple effort to write the requirements down yields a payoff of 30 to 50x. That is a multiple, not a percentage: it means 3000 to 5000% improvement – off the scale of possible metrics, as far as almost any other management action is concerned.

How to measure such improvements? Record the costs of initial development and of rework; not forgetting the costs of cancelled projects, legal action, and the benefits from system operations (eg product sales). Clearly the measures of benefit vary widely depending on what kind of projects you run (a one-off in-house project versus a product line, for example).

Why these extraordinary figures? Requirements are critical because:

- They state what the stakeholders want
- They guide system specification, design, and test
- They indicate the priorities for development
- They drive the duration, cost, and success of development and operations
- They are what progress is measured against
- They are what quality is measured against
- They are what systems are accepted against, contractually
- They are the basis for operations and maintenance manuals and training
- They set the direction for every aspect of development.
- Requirements are the basic communication between customer and developer, whatever the scale of project (whole network of systems, down to individual pieces of equipment)
- Without requirements, no effective development (or maintenance) contract is possible.

Given these functions, reaching into every part of development, it is hardly surprising that requirements are critical. Simply put, without (good) requirements there is no control of development, and little chance of project success.

Good requirements are the key to everything else in a development project.

[1] *To do or not to do: If the requirements engineering payoff is so good, why aren't more companies doing it?* Discussion Panel at RE'05, The Sorbonne, Paris, France.

RE-verberations

This section is for items of news that have a bearing on requirements work. RQ would like to hear of such things from its readers.

Hartree's Constant

Software projects believe they are "90 percent complete", and just a few more months from completion, for more than half of their lifetime. So says Hartree's Law, obscurely attributed to the mathematician Douglas Hartree. [If any reader knows the reason for the attribution, RQ would be grateful to hear it.]

Hartree's constant can apply to Civil Engineering projects too. English readers will need no reminding of the sad timetable for the rebuilding of the home of Association Football, Wembley Stadium. Even after all the very public sorrows of the project, the firm responsible for the actual construction still claimed earlier this year that the Stadium was "95% complete" (apart from trivial details like testing and safety certification), and "6 months from completion". How are the mighty fallen.

© Wembley National Stadium Ltd

Similar ideas are found in Fred Brooks' famous and still current *Mythical Man-Month* (1975). Brooks' Law – one of numerous maxims in the book – is "Adding manpower to a late software project makes it later". Or, for the politically incorrect, "The bearing of a child takes nine months, no matter how many women are assigned."

Many of the accidental details that Brooks takes for granted (batch programming, creaky languages, etc) are now long obsolete, but the essence of the book is still very much alive.

The liveliness of Brooks' ideas, and the continuing popularity of the *Mythical Man-Month* (even the title would be a bit dodgy today…) are evidenced by even the quickest of trawls on the web.

Watts Humphrey wrote '*Why Big Software Projects Fail: The 12 Key Questions*" in CrossTalk at http://www.stsc.hill.af.mil/crosstalk/2005/03/0503Humphrey.html quoting Brooks several times: clearly a key witness to the SEI's line of thinking.

Ed Willis wrote enthusiastically and critically – yes, the two can go together – in *The Mythical Man-Month Revisited* in O'Reilly's OnLamp ("the open source web platform"). His really charming article is all new-software-guy-discovers-amazingly-wise-old-guru at http://www.onlamp.com/pub/a/onlamp/2004/06/17/mmm_revisited.html. It's great fun, and illuminating too.

A while back, Steve McConnell (of Microsoft fame) wrote *Brooks' Law Repealed?* in IEEE Software, at http://www.stevemcconnell.com/ieeesoftware/eic08.htm. Brooks' Law probably isn't always true, though I doubt Brooks thought it would be. Rather, it's an ominous possibility if you don't carefully work out how replanning is going to make things better. But the article quietly admits that a 3-week schedule slip can gently turn into "six months or more after that", which is more or less Hartree's constant all over again.

In mischievous mood, RQ had a quick Google for "90 percent complete" and "95 percent complete". And yes, it was easy to find some recent claims that big software projects were still on track, demonstrating the continued constancy of Hartree, and the continued truth of Brooks. See for yourself!

Wise After The Event

It takes a disaster for the public to learn the names of engineering structures like "Tank 192, Buncefield Oil Depot". That was, if you were on holiday on Neptune at the time, the storage tank whose safety gauge jammed at around 3am in the morning of Sunday 11 December 2005.

Thereafter, the gauge cheerfully reported that Tank 192 was safely only 2/3 full. The night operators started to fill the tank at 7pm on the Saturday. They continued to pump hundreds of thousands of litres of 95 octane petrol into the tank until it started to overflow through its breather holes in the roof, at about 5:20am. The pumping continued for a further 40 minutes.

Didn't the tank also have a MAX LEVEL REACHED sensor? Oh yes, of course, only it apparently wasn't working either. (It is at this moment that every engineer feels a dryness in the throat. I'm getting that dreadful dry feeling just writing this.) The maximum level sensor should both have triggered an alarm, and shut down the tank's inlet valves. Either the alarm didn't sound in the control room, or the operators were VERY sleepy. (Do you know how loud emergency klaxons are? Yes, that loud.) The inlet valves stayed open.

Didn't anyone notice what was happening outside? It seems they did, and they complained too, but apparently nothing was done about it. About 300,000 litres of petrol spilled onto the forecourt. It formed a huge vapour cloud.

Air forces use fuel-air mixtures as an explosive. You drop a ton or two of petrol to form a cone of vapour. Then you drop a match into it (if you take my meaning), and the mixture devastates the whole area that it was dropped over. It is one of the most powerful weapons in the armoury. The Buncefield fuel-air cloud was 30 times as large…

The bang at 6:01am on that Sunday morning woke sleepers in central London (including me), 25 miles away.

By a miracle, or by the good fortune that the disastrous sequence of failures occurred on a Sunday when the depot and neighbouring office buildings were nearly empty of workers, no-one was killed.

It's easy to be wise after the event. Surely even a passive ready-reckoner could have predicted that if a tank is meant to hold x litres and you pump at x/10 litres/hour then it will be full in 10 hours' time? That would give you a simple, robust, independent alarm.

Another thing: how safe is it to do such things at night? Chernobyl Reactor No. 4 (yes, another engineering unit whose name the public knows) was under the control of a night crew at the time of its disastrous experiment.

It's hardly news that people are pretty clumsy when they stumble downstairs for a glass of water at 4 in the morning. The Human Factors people have shown conclusively that being on duty in the wee small hours is comparable to trying to work after a couple of pints of liquid lunch in the pub. You can just about manage, as long as you're not trying to, ahem, operate machinery, drive a vehicle, or do anything at all fiddly with Microsoft Office.

There is an alternative: have the night crew work as if they're in a submarine: in what their bodies think is the daytime. Turn off the lights during our day, turn on some nice bright sunlamps in our night, convince the crew's Pituitary glands that they have traveled to the Antipodes. When they've got over the jet-lag after a few days, let them control our oil depots and nuclear reactors, whatever, in artificial daytime.

Mind you, checking that the overflow sensors are working properly might not be a bad idea, either.

Jargon and the General Public

While on the subject of safety requirements, the question of whether and how the public understand the delphic pronouncements of safety engineers came up this week at the 1st IET International Conference on System Safety. (That's the IEE at Savoy Place, London, to you, but Faraday's institution no longer has Electrical in its name now.)

RQ put its head in the lions' den, presenting a tutorial on the interface between requirements and safety (ie, teaching in an area where neither tutor nor audience felt entirely safe…), and escaped happily unscathed.

At the opening Keynote, C. Michael Holloway of NASA said that people misunderstood safety engineers because in popular usage:

- Safety = Security (indeed, in German the same word is used for the two -ilities!)
- Reliable = Safe
- Hazard = Risk
- Safety is absolute or non-existent (ie, Boolean, and hence not quantifiable).

He demonstrated with the help of a few dictionary entries just how hopeless the case is.

Why do specialists use terms with odd meanings? It could be:

- To denote real differences
- A claimed need for precision
- A desire to justify extra funding
- A way of asserting superiority (the swishing of priestly robes)

Obviously the last 3 reasons are post-hoc rationalisations (I guess this was a jibe at some safety case argumentation).

Of course, such problems are totally confined to safety engineering, and never occur in requirements engineering. We requirements engineers never form cults, follow heroes, mutter magic spells, or use complicated language to impress clients, get funding, or attract converts. Absolutely not.

RE-flections

What DID People Do When They Didn't Do Requirements?

Well, if requirements are so wonderful (see for instance *A Brief Note on Requirement Metrics*, above), then how could people possibly manage without?

Obviously they did, as RE in anything like its current form has only existed a short while:

- a decade or so under its current name;
- since 1965 or so in the form of books of rules at NASA and the DoD telling you to write *shall*-statements.

Design

One answer is given by Robert Dreyfuss in his extraordinary and pioneering *Designing for People* (1955). What comes before the details of engineering a product is Design. What comes after that is Production. Thus 'Design' covers the whole V-model from requirements to testing.

Dreyfuss designed the standard black bell telephone and the original Hoover vacuum cleaner. Were those designs meant to meet market needs? Of course. Did anyone write down those needs? No, the process of working out what exactly was needed and sketching a physical 'design' for the devices was one and the same. The designer's mind had to span the gap between function and engineering, or as we might say between problem and solution. But Dreyfuss would not have recognized that problem space (eg the realm of vacuuming a carpet) and solution space (eg the realm of, err, Hoovering a carpet) were distinct. Indeed, the distinction is often hard to maintain, being a relative matter of one's perspective, not anything absolute.

Business Case … Procurement

A quite different answer is given by the Office of Government Commerce (http://www.ogc.gov.uk). In-service use of computer systems is preceded by best-practice Procurement, which in turn is preceded by the preparation of a properly-evaluated Business Case.

In other words, you work out the costs and benefits of getting what you need in different forms, and then you buy the best-value set of kit you can find. The difficulty here is seen not as a matter of designing anything new, but in identifying something affordable that does the job. The requirement may be large or small, but on the whole there is no special difficulty in meeting it, at least as long as you have deep pockets. It's just a matter of putting out an invitation to tender, getting two or three quotes, and picking the best one in an above-board process.

Engineers may remark that this does rather gloss over any small local difficulty the IT department may have in actually getting the kit to work, but then the OGC certainly isn't looking at things from an engineering perspective.

Business Cases … and Screen Logic

I first came across the Requirements = Business Case view of life with a retail bank, longer ago than younger engineers can remember, probably. Having previously worked with engineering organizations, I had imagined that everyone developing a system followed the V-model (or perhaps a Waterfall), at least in general outline. It was startling to discover that there were quite other ways of thinking about getting a new information processing system into being.

Banks at that time met customers face-to-face, via handwritten cheques, or via telephone calls and faxes. If a bank could get an Internet channel to haul in a profitable few hundred thousand customers, at the cost of only a few tens of millions of pounds, then the benefits would far outweigh both the cost of the development, and indeed a few millions lost through confusion, systems not working, fraud or whatever. The bank frankly couldn't give a damn, my dear, whether the requirements were accurately met, or indeed whether they were written in a traceability database, on tracing paper, or old tin boxes.

While I was recovering from the shocking cynicism of this point of view, I made a second unpleasant discovery. The well-fed, relaxed stripy-suited City types in the comfort of Head Office did the Business Case. After that they handed the project over to the lean, hungry and overstressed analyst/programmer types, conveniently spaced a few hours' away in an anonymous grey building in an industrial city.

The analysts worked out the Screen Logic, the flow of information and control from one User Interface view to another. Then the programmers coded it all up and tested it. That was followed by large-scale Trials, typically in parallel with real operations using the previous system, and if all went well, by live Operations in banking service.

And if all didn't go too well, the analyst/programmer types could just work a bit harder to plug the gaps.

Bank Clerks may no longer stand, like Mr Cratchit, at tall wooden desks, but the us-and-them attitudes that Dickens portrayed have, perhaps, not entirely vanished.

Objectives and Business Benefits

The construction industry sees things differently again. Clients don't generally ask for the ability to sleep untroubled by wind, rain, and thieves: they ask for houses with windows, rooves, front doors and bedrooms. Even very large developments are rarely treated as systems.

Objectives are set - so much commercial space and rental income, so much housing, complete by such and such a date.

Business benefits are considered - improvements in access to shops, easier parking, and so on. These are added up, and if the planning rules are tight, they are weighed against costs such as the environmental impact, loss of amenity and the like.

Plans and maps are drawn up showing what will go where, but this is more like 'design' than 'requirements'. Then the whole thing is cleared through the planning authorities, or a public inquiry if it's really big or sensitive, with a mountain of documentation, and then the engineers and builders put the thing together, often after a design competition and/or a tendering cycle.

There is a striking sense of care and attention to detail, but it absolutely isn't a sequence of requirements, design and verification documents, and if you mentioned traces or attributes, people would give you funny looks.

Modification Engineering

By no means all engineering developments write requirements either. (Is this heresy?) The 1956 model is the 1955 model with enhanced fins and more chrome, and maybe you get a radio and a heater as well. Since the product worked fine last year, why change it? Just make it look a little better, add a few sales features, and off you go.

Modification engineering is the rational strategy as long as you're sure that the changes you are making won't take you over the cusp and down some precipitous drop into the unknown. Small physical changes usually don't do that; but small additions to software unfortunately can, as features interact in unpredicted ways. Then you need all the controls and safeguards systems engineers can devise, but that's another story.

User Stories

Finally, we should mention that even when a problem statement / solution specification split is seen to be needed, it isn't universally agreed that requirements are the answer. The Agile methods / extreme programming community writes brief User Stories (among many other practices), as ably explained by Mike Cohn in *User Stories Applied* (2004). The approach enables the software development team to focus effort on specific deliverable results, quickly, which can't be a bad thing. Critics point out that you still need an architecture, but where that is broadly known in advance (as with many Web services), it may make excellent sense to concentrate on the most variable elements in the equation, namely the functionality.

101 Ways to Swing a Cat

I fancy that it would be fair to say that while none of these people do or did Requirements Engineering (with Capital Letters), all of them engineer(ed) their requirements in one way or another. So, next time you feel the urge to rush out and decry someone else's approach, just ask yourself how you know that your way is better.

RE-partee

The Optimist, The Pessimist, and the Requirements Engineer

Once upon a time three friends, an optimist, a pessimist and a requirements engineer went into a pub. The engineer discovered a shared requirement for beer and ordered three. Wishing each other good health, they all supped the excellent ale.

Ah, my glass is half empty already, said the pessimist.

Never mind, it's still half full, said the optimist.

Hmm, the glass is too large for its contents, said the engineer.

Proverbs

We seem to be close to the heart of RE this issue, so it's time for some core thinking. Here is one serious positive proverb, along with one lightly negative one on our theme:

Ask and it shall be given unto you.
St. Matthew, VII, 7

If you don't know where you're going,
you're unlikely to end up there.
Forrest Gump

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 September 2006

RE-Publications

Weinberg on Writing

The Fieldstone Method

Gerald Weinberg, Dorset House, 2006

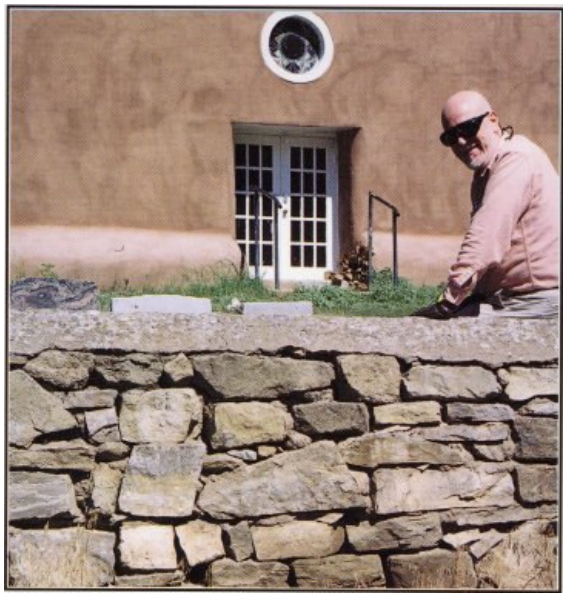
Weinberg should be known to older RQ readers as the co-author of one of the very first RE textbooks: Gause and Weinberg's *Exploring Requirements, Quality Before Design,* Dorset House, 1989. It's still an entertaining and stimulating read, by the way. Since then, Weinberg has somehow managed to pen more than 40 books, and consult into the bargain. So even a non-RE book may be of some interest: and of course, the art of written communication isn't a million miles from the art of writing requirements.

But how do you write about writing? It demands introspection, privileged observation, or close study. It is inevitably an over-the-shoulder sort of thing, a meta-level exercise. Indeed, since writing is always about something, writing about writing is a meta-meta-thing, like, well, writing about improving the processes for writing requirements … how many meta-levels is that?

To avoid vanishing into meta-space at warp speed, Weinberg wisely chooses a very concrete image to illustrate his approach: choosing stones from the field to build a dry-stone wall. He calls it the 'fieldstone method', and just in case you think he's bluffing, the cover sports a photo of Weinberg on a handsome wall that he's plainly built himself.

One aspect of building dry-stone walls is that you don't know in advance where each stone will go, or the exact design of the wall in terms of how the stones will fit together, until you build it. Instead, you gather stones, and then fit them in when and where they are needed. Authors, like magazine editors, get into the habit of collecting stones into a more or less orderly pile, and then pick from the pile when they are ready. Some of the stones will be rough and unfinished; others will be nicely smoothed, just waiting to be used.

As you can guess from all this about walling, Weinberg is full of ideas, metaphors, suggestions, parallels, and partial methods. Some of the metaphors may feel a bit of a stretch; some of the suggestions really basic. Of course, as an author myself, I'm probably not the intended typical reader. But as to the overall approach, Weinberg is surely right. Writing is a craft, like walling. It can be taught and learnt. True, some people will turn out to have a natural aptitude for it; but most people can become much better at it through instruction and practice.



Semplice, ma non facile…

The principles of writing are in the main simple, as with most crafts: but simplicity does not exclude skill, far from it. The opposite may be true: long detailed procedures can readily be described as algorithms, whereas to the terrible question 'How do I become a famous author?', one can only reply 'Write a good book' – not something likely to be automated in a hurry.

Somehow, Weinberg manages the trick of making all this captivating. The introductory chapter - usually the most boring bit of a textbook - is totally delightful (I won't spoil it for you, but it contains a true story).

The book looks in turn at the "fieldstone method", writer's block, gathering materials, recycling from literature, stealing stones safely ("when is it plagiarism?), tools, discarding stones that don't fit, several techniques for organizing your work, using your subconscious (sic), shaping stones to fit, filling the cracks, and not least, knowing when to stop.

These chapters are themselves shaped stones, and (taking Weinberg's own hint) you should feel free to dip into his stone-pile, gather and steal what is useful to you, and go shape your own (as he might say). Some of the chapters resonated with me as good things that I do,

some as excellent advice that I should try, and some didn't resonate at all. That seems fine: Weinberg's is a broad field, and few readers, perhaps, will need to cover every inch of it.

Any engineer faced with having to write an article or paper, to give a public talk, or even to write a book, will find Weinberg's advice immediately practical and relevant. He's been there.

-o0o-

Weinberg has written a weblog on the subject of this book; it gives a free taste of his approach.

http://weinbergonwriting.blogspot.com

Ralph Young: Project Requirements

A guide to best practices

Management Concepts, Inc, 2006

Our indefatigable colleague Ralph Young has just brought out his third requirements book. (He'll have to get a move on to catch up with Gerry Weinberg's quoted 'more than 40 books' – Ed.)

This one sets out its stall at once with *How the PM Will Benefit By Paying Attention To Requirements*, beginning how it means to go on.

Young's previous books *Effective Requirements Practices* (2001) and *The Requirements Engineering Handbook* (2004) described what projects and analysts (respectively) needed to do to get their requirements right.

This book is straightforwardly aimed at project managers, aiming to provide them with friendly, in-their-language advice on putting good practices in place on their projects.

RQ doesn't often quote cover blurbs, but Kathy Altizer's comment

> *"It puts the onus for requirements squarely where it belongs – with the PM"*

does echo the tone of the book. That isn't to say that Young believes that the analyst's skills are unimportant: far from it. But without informed direction, projects don't stand a chance.

This book sells RE by telling managers fairly and squarely what's in it for them: project success, saving time and effort, staying on schedule, getting a quality product, satisfied customers. No room here for treating engineers as oily rags!

The chapters cover Key requirement success factors, Partnering, Project Startup issues, Teamwork, Coaching the team, Clear Communication, Being Agile, Continuous Improvement, the PM's role on Quality, and finally Requirements, Risk, and the PM.

All of this is calculated to focus managers on the vital issues: does the project know what it is doing, and is that defined as well as possible. Risk, Young knows, never goes away; requirements never stand still; and processes never become perfect. On such shifting sands, some give up, keep their heads down, and plan for failure. Young stays put, and gives detailed practical advice on what to do (when Confusion reigns, Identify a champion, Hired experienced analysts, Write a project vision and scope…).

The book is enlivened with short cameo appearances by the RESG's Suzanne Robertson and Pete Sawyer (both very readable).

If you find yourself talking to management types, and you get the feeling they need a bit of persuasion, *Project Requirements* might be just what you need.

RE-Sponses

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

Toulmin Argumentation for Security Requirements

Good afternoon, Ian,

I enjoyed reading your review of Toulmin's book in the latest *Requirements Quarterly* [RQ39].

You might be interested to know that my security requirements work heavily uses Toulmin argumentation. One paper available online describing how the argumentation fits into a formal/structured security argument is:

Haley, C.B., Moffett, J.D., Laney, R., Nuseibeh, B.: *Arguing Security: Validating Security Requirements Using Structured Argumentation*. In Proceedings of the Third Symposium on Requirements Engineering for Information Security (SREIS'05) held in conjunction with the 13th International Requirements Engineering Conference (RE'05), Paris France, 29 Aug 2005.

http://mcs.open.ac.uk/cbh46/papers/SREIS05-Haley-SecArgs.pdf

I also understand that Chris Johnson at Glasgow bases much of his accident analysis argumentation on Toulmin. When queried, he responded as follows:

--- From Chris Johnson

If you're interested, there is a joint paper I did with Michael Holloway from NASA Langley on causal logics for safety-critical systems at:

http://www.dcs.gla.ac.uk/~johnson/papers/mishap.pdf

There's also a detailed study of a bank fraud (Allfirst in Boston) that I did for a group of US Banks and investment companies last year using a Toulmin style analysis on:

http://www.dcs.gla.ac.uk/~johnson/papers/V2.PDF

Best regards,

Charles B. Haley, The Open University.

c.b.haley@open.ac.uk

Charles kindly provided the following article for RQ on the theme of his letter.

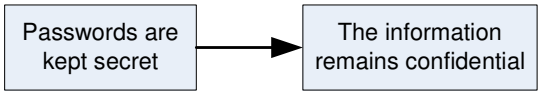
Toulmin Argumentation and Security Requirements

Charles Haley, The Open University

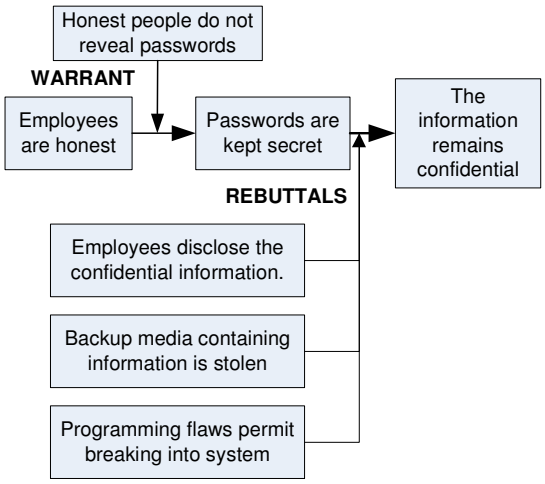
When considering security requirements, there is only one certainty: the bad guys won't follow the rules. They bend, probe, and subvert anything they can, from exploiting buffer overflows in software to 'post-it' surfing in offices looking for passwords. No assumption about how components of a system (system in the large, not just software) will behave is truly safe. Regardless, we cannot simply abdicate because the problem is too hard. Instead of giving up, or instead of mouthing platitudes like "the system shall be secure," we must instead find ways to reason in an uncertain world. One technique that we have been studying is the use of informal argumentation to bring assumptions to the surface, to challenge them, and to reason about the impacts if the assumptions are incorrect. We have found Toulmin argumentation to be very helpful in this process.

Issue 39 of the Requirements Quarterly reviewed Stephen Toulmin's classic book "The Uses of Argument," bringing out Toulmin's point that in real life, arguments are made up from "*practically certains*" and "*pretty sures*". One seldom finds trustworthy *always wills* or *can never happens*. Toulmin proposed a way of structuring arguments to show how what is known or believed (the *facts*) supports some *conclusion*. Especially important for security, his arguments include rebuttals that discuss what, if true, might cause the conclusion to be invalid.

To illustrate how we use Toulmin arguments, consider a system containing confidential information that uses password authentication. For such a system to be secure, i.e. for the information to remain confidential, one must assume that the passwords are kept secret. The only person who knows a password is the person authorized to use the password. Such an assumption would be diagrammed in Toulmin's notation as follows:



Seeing the assumption so baldly placed should provoke many questions. How do we know that passwords are kept confidential? Under what circumstances might passwords not be kept confidential? Are there ways to get the information even if passwords remain secret? Toulmin comes to our rescue again. The following diagram begins to expose these questions.



Beyond the content of the boxes, three important ideas can be seen in the above diagram. The first is that rebuttals are used to describe conditions under which the argument does not hold; in the situation under discussion this is where the system will not meet its security goals. The second is that warrants are used to establish how a fact relates to a conclusion; given that employees are honest, the fact that honest people do not reveal passwords allows us to conclude that passwords are kept secret. The third is that arguments are recursive; the fact "Passwords are kept secret" is itself a conclusion supported by its own argument.

As can be seen from the discussion above, warrants and rebuttals are just another kind of conclusion. Left unsupported, they are assumptions. They can, and in often should, be argued just like other conclusions. The diagram grows until the stakeholders (in this case stakeholders interested in security) are convinced that all of the arguments are sufficiently complete and reasonable, and that the risks presented by the assumptions are not excessive.

If by now you are thinking "Wait a minute. You can't stop here. Those rebuttals are serious!", you would be correct. We need some way to describe how to remove a rebuttal. However, we are again forced to deal with uncertainty - rebuttals (usually) cannot be 100% countered. Instead, they must be *mitigated*, and to this end we have added 'mitigated by' box to the Toulmin diagram, linked to the affected rebuttal(s). Mitigations describe how the effects of a rebuttal can be reduced to an acceptable level. Mitigations might require the introduction of new functionality into the system, which may introduce other security risks. For example, the effects of the rebuttal "backup media containing information is stolen" can be mitigated by encrypting

the backups (encryption and key management are new), introducing the possibility of key loss (an availability concern). "Employees disclose confidential information" can be mitigated by a combination of training and physical security (both are additions to the system), introducing the possibility of subverting the physical security. More arguments would be needed.

Our work has shown that each part of a Toulmin argument, including our newly added 'mitigated by', is either a bare fact or the conclusion of some other argument. As such, and as mentioned above, arguments are highly recursive. The recursion is terminated by unsupported facts, which we call *trust assumptions*. Trust assumptions represent the analyst's belief that something is true and need not be further justified. Whether this belief is or is not true is a matter of judgment. By making trust assumptions explicit, they can be debated and judgments agreed.

In conclusion, when reasoning about security, one needs a way to 'formalize' the informal and the uncertain. Toulmin argumentation helps us capture and reason about "well, what about …," "that won't work if …," and "you are making an assumption that …" The structure and loose semantics of Toulmin arguments facilitate some automated analysis, for example answering questions of the form "if this assumption isn't true, what breaks?" and "are all rebuttals mitigated to an acceptable level?" Finally, the artifacts (the diagrams & associated reasoning) serve as a record of the rationale for decisions, and as evidence of due diligence.

More details about our work in security requirements, including copies of papers providing more detail about how we use Toulmin argumentation, can be found in [1, 2, 3]. The papers are available on my web site http://mcs.open.ac.uk/cbh46.

[1] C.B. Haley, R.C. Laney, J.D. Moffett, and B. Nuseibeh, "The Effect of Trust Assumptions on the Elaboration of Security Requirements," *Proceedings of the 12th International Requirements Engineering Conference (RE'04)*. Kyoto Japan, IEEE Computer Society Press, 6-10 Sep 2004, pp. 102-111.

[2] C.B. Haley, J.D. Moffett, R. Laney, and B. Nuseibeh, "Arguing Security: Validating Security Requirements Using Structured Argumentation," *Proceedings of the Third Symposium on Requirements Engineering for Information Security (SREIS'05) held in conjunction with the 13th International Requirements Engineering Conference (RE'05)*. Paris France, 29 Aug 2005.

[3] C.B. Haley, J.D. Moffett, R. Laney, and B. Nuseibeh, "A Framework for Security Requirements Engineering," *Proceedings of the 2006 Software Engineering for Secure Systems Workshop (SESS'06), co-located with the 28th International Conference on Software Engineering (ICSE'06)*. Shanghai China, 20-21 May 2006, pp. 35-42.

RE-Sources

Books, Papers

See also 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:
www.cc.gatech.edu/computing/SW_Eng/resnews.html

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

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

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

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

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

Mailing lists

RESG Mailing List:

The RESG provides an email forwarding service to the requirements engineering community in the UK and worldwide. Calls for papers for RE-related conferences, workshops and special events are forwarded to the mailing list. Mailings also include reminders of coming RESG or related events and position openings for professionals in the RE field.

This mailing service is moderated to ensure that all messages are RE-related and conform to the aims of the service indicated above.

RESG members become part of the RESG mail forwarding service soon after they register.

If you are not an RESG member, but wish to benefit from to the RESG mail forwarding service, you can join by following the instructions below.

To register your email address on the RESG mail forwarding list send a message to admin-mail-list@resg.org.uk including the text "subscribe" anywhere in the subject line.

To update your email address on the RESG mailing list send a message to admin-mail-list@resg.org.uk including the text "change address" anywhere in the subject line. Please indicate in the body of the message the old email address that is to be replaced. The originating email will be replacing the old one on the list. Addresses to which

messages cannot be successfully delivered are removed from the list.

To remove your email address from the RESG mailing list send a message to admin-mail-list@resg.org.uk including the text "unsubscribe" anywhere in the subject line.

If you wish to forward a relevant message - a call for papers or participation - to the RESG mailing list, send it to admin-mail-list@resg.org.uk with the text "forward to list" anywhere in the subject line.

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>

RE-Actors: the committee of the RESG

Patron:

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

Chair:

Dr Pete Sawyer, Computing Department,
Lancaster University,
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, NATS,
David.Bush @ nats.co.uk

Membership secretary:

Dr Lucia Rapanotti, Computing Department, The Open University,
l.rapanotti @ open.ac.uk

Publicity officer:

William Heaven, Department of Computing, Imperial College,
wjh00 @ doc.ic.ac.uk

Newsletter editor:

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

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

Regional officer:

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

Student Liaison Officers:

Zachos Konstantinos, City University,
kzachos @ soi.city.ac.uk

Andrew Stone, Lancaster University,
a.stone1 @ lancaster.ac.uk

Immediate Past Chair:

Prof. Bashar Nuseibeh, The Open University,
B.Nuseibeh @ open.ac.uk

Industrial liaison:

Prof 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

The Requirements Engineering Specialist Group of the British Computer Society

Individual Membership Form for 2006

1. Membership Type

Please indicate type of membership:

BCS/IEE members, please

BCS / IEE member (£10) [ ] *indicate membership number*_____

Non-BCS / IEE member (£20) [ ]

Full-time student (free) [ ] *Studying at:*_____

If you need a receipt, please tick here [ ]

Corporate Membership also available – details at www.resg.org.uk or ask the Membership Secretary

Payment by cheque only. Please make it payable to "The BCS Requirements Engineering Specialist Group"

---

2. Your Details Title: Mr/Mrs/Ms/Dr/Professor/Other:_____(delete as appropriate)

First Name:_____Surname:_____

Address for correspondence:_____

_____Postcode:_____

Phone:_____Fax:_____

E-mail address: *Please write your e-mail address clearly using BLOCK CAPITALS*

As an RESG member your e-mail address will be added to the RESG mailing list

Optionally, indicate:

Your organisation's name:_____

Its type of business/domain:_____

3. Your Specific Interests

Areas of interest for the RESG given at *http://www.resg.org.uk/about_us.html*. Is there another area would you like us to add?

---

4. Your Participation Preferences

Please indicate what timings/duration for RESG events would suit you:

Whole day [ ] Half day [ ] Evening [ ] Other (please specify) [ ]_____

Please indicate whether you would be willing to help the RESG with:

Publicity [ ] Newsletter contributions [ ] Organisation of meetings [ ]

5. Your Mail Preferences

The RQ quarterly newsletter is regularly sent to all RESG members. It will be sent as a PDF e-mail attachment unless you prefer a hard-copy of RQ delivered by post. For hardcopy delivery, please tick here [ ]

I understand that the information supplied on this form will be held in a database and used for the purpose of distributing information relevant to the activities of the RESG.

6. Your signature:_____Date:_____

Please send this form with payment (if applicable) to: Juan F. Ramil
RESG Membership Secretary membership-RESG@open.ac.uk Fax +44 (0)1908-652140
Computing Dept., The Open University, Walton Hall, Milton Keynes, MK7 6AA, U.K.