Requirenautics Quarterly

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

http://www.cs.york.ac.uk/bcs/resg/

© 1999, BCS RESG Issue 19 (December 1999)

RE-Locations

RE-Soundings

Editorial

Welcome to the 19th, and millennial, issue of Requirenautics Quarterly. Despite intense lobbying, the RESG chairman has failed to secure an 'RE zone' in the dome. In spite of this setback, there's every reason to be optimistic about the prospects for RE.

The established RE conference, symposium and workshop series continue to flourish and it is rare nowadays to see a programme for a systems or software engineering conference that doesn't have at least one session devoted to RE issues. More significantly, perhaps, is the degree of industrial involvement that we are seeing in the RE research community. At a recent Finnish seminar on RE process improvement which I attended, the audience was almost entirely composed of people from Industry.

It is clear that many more organisations now recognise the strategic role of requirements engineering to their business. Similarly, many more researchers recognise the practical hurdles to adapting research ideas to an industrial setting. There are encouraging signs that RE practice as a whole is set on an upward path. There is much still to do, but the augurs, particularly in Europe, are good.

Regular readers may by now have noticed that the editorship of RQ has changed. Let me take this opportunity to thank George Spanoudakis for his excellent work as the last incumbent. Fortunately for myself and RQ's readership, Ian Alexander is our guarantee of continuity. He continues to shoulder a very large part of the burden of producing RQ. I hope we've got him well insured.

There is one cloud looming on the horizon, however. Geoff Mullery, who's regular 'CORE-Blimey!' column has been the highlight of many more issues than Geoff would care to remember, has served notice that he would like to get on with his life sometime soon. We owe an enormous debt to Geoff, both in the RESG and in RE as a whole, for being so generous with his experience, his ideas, and his wit. As a reading of this issue's *CORE-blimey!* will show, there is no more perceptive and articulate proponent of RE. Although Geoff has indicated that he will contribute a small number of further articles, we need to start looking for someone to fill his shoes. Any volunteers?

Pete Sawyer,
Computing Department, Lancaster University

Chairman's Message

The RESG has again been active around the country during the last few months. The group organised meetings on requirements engineering for e-commerce (London in July), on managing requirements change (Oxford in August), and on requirements and COTS (York in November). We have also lined up some more exciting events for the new year. One that I am personally looking forward to is a debate on "Extreme Programming" in London on 9th March 2000 - jointly organised by the BCS RESG and the Advanced Programming Specialist Group.

As the year draws to a close, I would once again like to thank the RESG Committee for all the hard work they have put into the group and its activities. Special thanks to George Spanoudakis for doing such a great job editing RQ over the last year, and a warm welcome to Pete Sawyer as he takes over the editorship. As you can see from this issue, Pete is a very fast learner!

Season's greetings to all the RESG membership, and my best wishes for a very happy, and bug-free, New Year.

Bashar Nuseibeh,
Imperial College, London

Industrial Liaison - aspirations, plans and an invitation

One of the fascinating aspects of RE is that while there are always innumerable paths to explore developing the theory, there is also a strong practical pull. Designing something of practical use, getting something done and getting it done better, is a major part of the subject's attraction.

To get things done in practice requires knowing some practitioners. Many forward-looking organisations are already members of RESG. We need to build on these links and establish more. The first step in this process is to raise awareness of RE in industry and government and to link RE to their immediate concerns.

Following a brainstorming session which helped organise our ideas, the committee has decided on a programme of activities which will, we hope, increase industrial involvement in RESG first and foremost by recruiting more industrial members to the group.

First we are identifying what the principal concerns of practitioners and mangers are in order to define more clearly where RE can help and why it is important. The current list includes e-commerce, extra-nets, procurement and extended enterprises.

Second, we are planning to raise the public profile of RE.

Third, we want to develop mini-tutorials which explain in practical terms what RE is and present some of the leading edge work, for the more sophisticated practitioners. The sub-text here is to bring RE people together so that the sense of community is strengthened. Following on from that is the idea of RESG acting as a broker, which will mean developing a list of experts, their expertise and how they can be contacted.

Finally we are planning to develop a European focus to our activity. The global economy may or may not be upon us, but it makes sense to be aware of the issues and concerns of the wider economy of which we are already a part.

To do all this we need the help of the membership, that is, you. If you have ideas on what's really bugging practitioners at the moment, a mini-tutorial in your back pocket, specialised expertise which you want to broadcast more widely or practical problems which need fundamental research to crack, please let us know.

Barbara Farbey
University College, London

RE-Treats

Next event organised by the group.

Dependable Distributed System Requirements

Date: 16th February 2000
Location: University College London
Contact: Wolfgang Emmerich, University College, London (W.Emmerich@cs.ucl.ac.uk)

Other forthcoming events.

Extreme Programming not Requirements Engineering - a debate

For the motion: Paul Dyson
Against: Ian Alexander
The meeting will be preceded by a (neutral) mini-tutorial on Extreme Programming by Susan Eisenbach (Imperial College).
Date: 9th March 2000
Time: 6:30pm - 8pm
Location: Imperial College, London
Contact: Bashar Nuseibeh, Imperial College, London (ban@doc.ic.ac.uk)

UML Mini Tutorial.

Presented by: Jim Arlow
Date: 17th May 2000
Location: Imperial College, London
Contact: Wolfgang Emmerich, University College, London (W.Emmerich@cs.ucl.ac.uk)

RE-News

Calendar

We have received the following notices of forthcoming events in which RQ readers may like to participate.

June 99

ICSE 2000 – The New Millennium: 22nd International Conference on Software Engineering, Limerick, Ireland, June 4-11, 2000.
http://www.ul.ie/~icse2000

CAiSE 2000 – 12th Conference on Advanced Information Systems Engineering, Stockholm, Sweden, June 7-9, 2000
http://www.ul.ie/~icse2000

ICRE 2000 – 4th IEEE International Conference on Requirements Engineering, Schaumburg, Illinois, USA, June 19-23, 2000
http://www.cse.msu.edu/ICRE2000

ICSR 2000 – The Sixth International Conference on Software Reuse, Vienna, Austria, June 27-29, 2000
http://icsr6.isys-e.uni-klu.ac.at/icsr6

RE-Readings

Reviews of recent Requirements Engineering events.
All reports by Ian F. Alexander

Managing Requirements Change

This meeting, co-located with ICSM'99 conference at Keble College, Oxford was held on the 31st August 1999. We were very pleased that during the holiday season and out of London, there was a full room of RESG members and visitors, with a good mixture of industrialists and academics.

Jane Searles (ICL, UK) spoke on Understanding IT requirements in a changing world - The role of Enterprise Architecture. She described herself as a process person, not a requirements engineer, but was interested in how the two worlds interrelated. She had initially studied Methods, but had a road-to-Damascus conversion, and has since looked not at data but at process.

Her thesis is essentially that we need to look at the purpose of a business: this is more stable than implementation requirements or architectures. Over the last 30 years, ICL has moved from being a mainframe manufacturer to being a global IT services company. Some things were remarkably stable: the VME operating system endured for 30 years, through large changes in computer hardware. An abstract architecture described system purposes, while a CM system (CADES) handled the actual changes.

An enterprise architecture does the same thing, but for a business: it describes the shared values and purposes for an enterprise, and is implementation independent. Searles, following Graham Prattens' Process Oriented Systems Design (a thinking technique), identifies behaviours of each group of stakeholders: some of these behaviours will be shared, dramatically simplifying the model of processes. Behaviours are adaptable things, and they influence each other while surviving for long periods, so they offer a structure for organizing requirements. Behaviours are implemented by People, Processes, and Technology, and are influenced by (an enterprise's) Direction Setting, as well as the External Environment. People have Roles (which have Accountabilities), and Interactions. This general structure leads to a set of views of how to manage change: the business view, the social system view, and the technology view all overlap around an end-to-end process model.

Searles believes there is no one way to manage change, it is a huge problem with no boundaries. Instead, an architecture of the business helps. Scoping is vital, using systems thinking. Modelling helps one to understand implementations, and Impact Analysis is useful when making decisions.

Searles agreed with Richard Stevens (QSS) that companies were reluctant to put down high-level management objectives; it was easier to get project requirements on to paper.

Dr. Jun Han (Monash University, Australia - visiting University College London) spoke on Managing Air Traffic Control system requirements and their change. UCL is working with NATS to improve its handling of requirements, starting with a pilot project and hoping to move to a live project soon.

The first study was a reality check on NATS, which had large long-lived systems with complex procurement procedures. Issues for the project included processes, requirements content, the domain, infrastructure, people and tools. Goals for improvement included establishing process definitions, corporate standards, a "tool pool", a corporate memory and learning mechanism, and process assessment and improvement. The first study produced a list of suggestions, including a project startup kit, a training and mentoring program, an intranet server for Requirements Engineering, a support group, and a demo project.

The second study looked at requirements management. It aimed to support requirements capture, evolution, and traceability. To do this, it set up a core information model and a process guide, along with templates for documents in the form of HTML structures and DOORS templates and menu commands. This work was based on a demo project using the existing NATS conflict alert system retrospectively.

The roots of the approach are goal-directed Requirements Engineering (such as van Lamsweerde's KAOS), Jackson's Machine/World distinction, and a careful analysis of the relationship of requirements to system architecture (e.g. Interfaces are described in the architectural design document, not the requirements).

A process guide is a description of one traversal through the core information model (an entity-relationship diagram of the development process). For instance, you can identify stakeholders, define goals, refine the goals, state your assumptions and associated risks, and so on – in that order – and make suitable templates for these tasks. Change is accommodated by allowing assumptions to be violated, removed, or added: the traceability of DOORS then allows you to see what has to be changed to restore consistency.

Jill Burnett (Innovation Manager, QSS, UK) spoke about SACHER, one of QSS' Esprit projects. The acronym stands for Sensitivity Analysis and CHange Management support for Evolving Requirements, and as always with Esprit represents a co-operative project between industry and european universities.

The tool support was based on QSS' DOORS requirements engine, with custom extensions written in DXL for change management.

She stressed that the approach was very simple and pragmatic. The basic idea was derived from the concept of traceability, as in ESA's PSS-05 standards: a sequence of document products connected by a thorough set of traces. QSS itself receives about 700 to 1000 customer support calls per year, and these usually contain new requirements. Similarly, each user group meeting ('Wolfpacks' and 'InDOORS') leads to one or two hundred new requirements, including a Top Ten priority list agreed by the meeting.

At the other end of the process, each new DOORS release considers about 1000 candidate requirements. The goal for Innovation is to pick the ones that "we really must address", she said – from that list of 1000, by assessing cost and benefit in each case.

The mechanism consisted of a review cycle based on priority for all change requests. This led to an assessment of impact on the product by development engineers, and then to an evaluation of cost and risk. The cost was rated as 1, 10, 30, 100, 300, or 1000 person-days per requirement. A re-estimate was then made of whether the change was justified (i.e. a second cycle of review). If approved, there followed the usual build-test-release process.

Metrics were kept of real and observed costs: QSS hopes to use these to improve its estimates in future. Many observations are logged on each project.

This description left several questions unanswered, she explained. How was one to assess the impact of a change? For instance, how many user requirements would be affected? The prototype DXL tool treats a baseline as a set of DOORS modules. An impact tree is gradually built up, showing the costs per individual requirement and per node in the requirement hierarchy. The tool can show rolled-up costs for higher-level nodes. Another issue was how to derive a cost function from a set of observations, which were always prone to error. Eventually a simple straight line was used (cost = mx + c). A goal was to provide a custom in/out Excel interface as everyone seemed to like handling data in spreadsheets.

The SACHER consortium consisted of QSS, CEFRIEL, and Objectif Technologie. Both Nokia and GEC Marconi Avionics were prototyping with SACHER tools. One could look at requirements change, assess impact (crucially), evaluate costs, and carry out sensitivity (what-if?) analyses. Metrics were central to the process if knowledge was to be built up.

In the break we walked to the Great Hall of Keble and enjoyed a cup of tea in splendid surroundings: so intense was the discussion that everyone remained standing between the long oak tables with their traditional wooden benches.

Prof. Spencer Rugaber (Georgia Institute of Technology, USA) spoke on "Coping with mission-oriented requirements changes". An extensive project, MORALE with the US Department of Defense, is looking at many aspects of system evolution, starting from the premise that existing systems need to be understood and then updated. He was interested in the evolutionary design of complex software: i.e. his focus was on how to evolve existing software.

Software evolution consisted, he argued, of four major processes: adaptive redesign, requirements determination, program (code) understanding, and architectural evaluation and resolution. The project had developed or recycled a method for each of these: MESA (model-based evolutionary software adaptation); SCENiC (scenarios and inquiry cycle); SR/MORPH (synchronized refinement and HCI re-engineering); and SAAM (software architecture analysis method) with Architectural Reconciliation. A set of tools supported all of these methods to differing extents.

Central to all of this was the use of scenarios. These were "stories with embarrassing questions" – what would the stakeholder want if …? The response was invariably more stories. Tool support helped with reverse engineering, as when SAAMpad recognised boxes and arrows sketched by engineers explaining existing software. Rugaber's colleague, Colin Potts, had developed the SCENiC method to generate concrete scenarios using a systematic inquiry process.

Rugaber himself has his students write scenarios, with rewards for the most humorous. They are encouraged to give specific irrelevant detail to give colour and life. A goal hierarchy is illustrated with a set of scenarios; goals may be thwarted by Obstacles, or satisfied with Objectives and Tasks (which are performed by Actors).

How does all of this deal with Evolution? Firstly, the approach helps with the reuse of requirements and architectures. Secondly, there is a lot of value in existing systems which might be useful in new systems; any requirements gathered up help to reuse that value. Thirdly, scenarios are valuable in their own right as they show directly what systems are trying to achieve, and help to prevent surprises in the form of "unpleasant shocks".

The meeting ended with a vigorous panel discussion and questions from the floor. If there was a common theme, it was perhaps that change can only be handled by moving up from requirements to an understanding of business processes, where things change more slowly. Even so, constraints like performance and reliability can drive any architecture over the edge: all systems have their limits of adaptability. Systems might be designed specifically to be adaptable: it is a research issue how to allow systems to adapt while at the same time continuing to meet safety requirements.

The First International Workshop on the Requirements Engineering Process (REP'99)

Innovative Techniques, Models, Tools to support the RE Process

This meeting was held at the University of Florence, Italy, on the 2nd and 3rd of September 1999 (preceding the DEXA Conference).

Although this was not an RESG meeting, we Brits were well represented. Several of the RESG committee either spoke or had author credits on a paper, and Neil Maiden helped to organize the workshop.

Reuse, Validation and Verification of System Development Processes
Peter J. Funk, Ivica Crnkovic - Sweden

Peter explained that reusing and adapting templates was hard work. His approach was to use case-based reasoning, which made it easy to recycle cases, with formal representation, which made it easy to analyse, to capture adaptations made to existing cases. A task hierarchy broke processes down into atomic tasks, consisting of formalized input, output, and selection criteria, with an informal work description, allowed him to match processes using simple criteria such as project size, skill and experience, cost, time, and degree of formality. Matching was just by set intersection, giving a straightforward % measure of accuracy of fit.

Reusing Scenario Based Approaches in Requirements Engineering Methods : CREWS Method Base
Jolita Ralyte - France

Jolita described her work on a framework for organizing methods into chunks to enable to eventual construction of a method base from which we could identify the right method for a job. Each chunk consisted of a descriptor (meta-knowledge), a chunk interface and the chunk body. From a product viewpoint a method is a set of chunks; from a process perspective, a chunk is based on a set of product models, and consists of a set of guidelines which might be strategic, tactical (consisting of a number of context statements) or informal. A number of reusable chunks have already been documented in this way; they can be accessed via an internal website. Future work would describe more methods and would provide efficient access and search mechanisms.

A pattern based approach for Requirements Engineering
Mounia Fredj – France, Ounsa Roudies - Morocco

Ounsa presented a simple language consisting of seven patterns from which she claimed one could describe Requirements Engineering processes.

A Process Model for Requirements Engineering of CCOTS
Bernhard Deifel - Germany

Bernhard described a model of the process of specifying the requirements for 'Complex COTS', based on a collaboration between the Technical University of Munich and a large industrial partner. As well as the usual developer roles, he identified the importance of sales roles such as technical and strategic marketing, requirements compiler and sales representative. (He didn't mention technical support, though – a valuable source of requirements.) In a useful diagram he illustrated the place of each role in different types of product, from single customer/one-off through to mass-market/long-term. The first phase of his COTS life-cycle is to define such a market architecture; then he defines a system view and finally a development view. He also took a realistically complex look at version control, where many versions in different states of realization may co-exist.

Adding a Systematic View to the Requirements Engineering Processes
Vito Veneziano, Sara Jones, Carol Britton - UK

Vito presented some ideas on how one might approach the handling of partial or wrong knowledge through a visual representation of the possibly conflicting views of different stakeholders about different parts of a specification. A user interface prototype illustrated the concept as well as the problem: how could Requirements Engineers cope with hundreds of pieces of conflicting information? The prototype showed many windows linked visually by arrows. Clearly a mechanism for classifying and filtering related items, and for identifying discrepancies, would be necessary.

Creation of an Automated Management Software Requirements Environment : A Practical Experience
J. Andrade, J. Ares, O. Dieste, R. Garcia, M. Lopez, S. Rodriguez, L. Verde - Spain

Oscar described a process improvement initiative at a Spanish bank, starting from CMM levels 1-2 (somewhere between nothing and getting started, as far as RE was concerned). The team introduced a simple waterfall model with templates for documents, with elaborate criteria for choosing a tool. Requisite Pro was used, apparently (despite the criteria) because of its interface to the SQA testing tool. The project has found some benefits, such as more clarity about deadline and cost control, and better quality as seen by the users. However, the requirements attributes and their types were awkward and difficult to modify, and the requirements tool had not in fact been successfully interfaced to the testing tool, which "would have the added benefit of assuring that requirements are considered when the system is tested". As well as this startling admission, development was not in fact getting faster and might be taking longer than before. Requirements were written in system not user language, and users did not like JAD-like sessions involving the tool which they probably found intimidating as it implied finality of specification. It was not clear whether Oscar was intentionally describing such a serious set of process breakdowns, but we can be grateful for what is a rare and frank account of difficulties in RE process introduction.

The Challenges of Requirements Engineering in Mobile Telephones Industry
Alessandro Maccari - Finland

Nokia's research center is in the difficult position of being both within and without the development of mobile telephony. The developers are quite suspicious of the researchers and of Requirements Engineering, which is "too often regarded as useless and time-consuming". Alessandro hopes for "a higher degree of co-operation". The statistics are interesting: a million lines of code in what the Italians charmingly call a telefonino, organized into 2000 objects. The infrastructure is even more complex: 25 million lines in the Lucent mobile switch, for instance. The requirements are complex because of "the lack of a global standard transmission protocol, the numerous product customisations and the variations due to local standards". Black-box reuse is rare; no-one manages to predict how requirements will evolve, so the components always end up being modified or rewritten instead of reused. The rest of us can count ourselves lucky.

Neil Maiden raised the intriguing question of why Scandinavia was so advanced in mobile communications. Perhaps the long history of participative design and concern for usability had paid off. High phone densities, early adoption, well-educated and rich population, and open attitudes were also claimed as reasons: but the connection of these things with successful design is not obvious.

Supporting a Co-operative Requirements Engineering Process
Ian Alexander - UK

Ian described, using the example of the PIPSEE (aerospace process improvement) project, how the Scenario Plus toolkit supports a range of RE processes – the tools could to some extent be used independently, so the suggested approach could be mixed and matched. The objective was to support co-operative RE by explicitly representing stakeholders as agents, and communications between them as messages; by providing a visual structure for goals; and by offering easily-understood feedback. These goals were approached by providing tools to construct, analyse, and transform three information structures: agent interaction models (from SOMA); typed goal models; and test scripts (specific sequences of tasks, representing paths through a goal model). All the tools were freely available in an add-on menu to DOORS. Feedback to users included metrics on each type of model and UML-style swimlanes diagrams (as output). The key tasks of a requirements toolkit were to enable interactive construction of models, and consistent and effective visualizations of those models. Stakeholders needed simple diagrams and animations; requirements engineers needed clearly presented metrics and traceability. Work was in progress on measuring the effectiveness of the Scenario Plus approach.

"But people always give sequences" said Neil Maiden. Ian agreed, explaining that a sequential scenario turned into a branching tree when you asked about possible exceptions and alternatives, and fitted the resulting scenarios into the primary structure.

Guiding the Process of Requirements Elicitation through Scenario Analysis : Results of an Empirical Study
Mustapha Tawbi, Camille Ben Achour, Fernando Velez - France

Camille related the results of trying out the CREWS L'Écritoire guiding rules in an empirical study. The approach was to ask the subjects to identify goals for a specific problem, and then to teach them the CREWS rules and have them identify goals that way. The result was that the rules proved to be good for verifying discovered goals, but were also useful for discovering new ones. Subjects were in particular much better at envisioning the future system when using the rules. There was also some improvement in finding alternative paths (though a majority still failed on this task). The theoretical possibility that the effects might have been due to practice rather than the rules themselves was largely excluded by allowing the subjects plenty of time before moving to the second part of the experiment.

Guidelines for Better Scenarios : supporting Theories and Evidence
N.A.M Maiden, G. Rugg, P. Patel - UK

Neil argued that different representations of scenarios were more important than we thought. Representation affected how people used scenarios; RE using scenarios was still a craft with adhoc approaches and little explicit design of scenarios for particular tasks. For instance, UML focuses heavily on graphics (activity and sequence diagrams) whereas CREWS chose text (mainly to allow more on a screen). His experiment compared 3 approaches, presenting subjects with a representation of a robot manufacturing system: CREWS text using the authoring guidelines; UML sequence diagrams; and graphical animation (little men, a bucket, etc). For the number of correct requirements found, the scores were respectively 21, 14, and 14.6 on average: not significant. For the number of alternative courses found, the results were more exciting: 10, 4, and 2: CREWS/SAVRE was very significantly better. Neil at once disclaimed any massive validity: the experiment wouldn't stand up in an HCI conference. But anecdotally, the subjects said they found both text and animation easier than UML, even though all of them had recently been taught a full university course of UML. There seemed good reason to suppose that scenario use should be driven by empirical evidence (much needed) instead of theories or guesswork, and constraints on scenario design and use should come from industrial experience. Tools and methods including UML probably needed urgent improvement.

Reliable Requirements Through the Quality Gateway Suzanne Robertson – UK

Suzanne unfortunately decided not to come and present her excellent paper; whatever her reasons, I sympathize, having paid for 2 full days of a database conference in which I had no interest (to the amusement of those present, I cut down my proceedings book to lighten my travel baggage…). Perhaps the next REP workshop will be independent.

Her paper argued that testing of requirements needed to start at once, using her concept of a "fit criterion", so as to tell whether each requirement would be satisfied by any proposed solution. It was no good waiting until testing of actual software to find out whether it could in principle meet its requirements. The requirements themselves therefore had to have testable fit criteria, and they had to pass through a quality gateway to check relevance, coherency, traceability, consistency and other attributes as soon as possible.

The Process of Inconsistency Management : A framework for Understanding
Bashar Nuseibeh– UK, Steve Easterbrook– USA

Bashar said that multiple views were OK, as they help to focus attention on design issues and problem areas. Classically, inconsistency was a disaster as it trivially enabled any conclusion to be drawn by formal logic. However one could encapsulate alternative views as distributed objects which if taken as truth might conflict, whereas if taken as partial knowledge about representations, development options and specifications were often useful and important contributions. A form of logic called hypothetical reasoning could handle this sort of thing. Inconsistency was caused by a spectrum of things, from mistakes (at the minor end) to total conceptual disagreement (at the heavy end). In between were things like mistaken positions, divergences of interpretation, and so on. Strategies ranged from ignoring inconsistency, circumventing it, deferring it (often a good idea) and ameliorating it.

Colette Rolland asked how a rulebase could be designed to allow inconsistency; Bashar replied that a wide range of logics from temporal and QC through to informal methods could be applied. Most of the approaches basically allowed a mapping of elements in one person's description to those in another's.

Experiences from the ESSI Project "Requirements Management in Alcatel Telecom Norway" (RMATN-24257)
Bjarne Tvete - Norway

Bjarne told us about what really happened when what is now Thomson-CSF Norway tried introducing Requirements Management. They went for a waterfall model with stress on traceability. They carefully worked out evaluation criteria and studied the available tools, recommending DOORS (with RTM second), but management chose Requisite Pro (rejected in Phase 1 for lack of functionality) on the grounds it was "easy to get started with". Perhaps price came into it, though Bjarne did not say so. There was plainly some tension in the team about this decision; the team had to repeat its tool evaluation but did not change its conclusions. One recommendation that came out was that management needed to be involved better from the start. Staff and managers lacked understanding of requirements, and the project was hampered by high staff workload and over 50% turnover. However there were clear benefits from managing requirements, which would come into play if the approach was introduced on a pilot project and "lighthouse" people with influence in the firm were used to get it accepted.

The impact on Staff of the implementation of PVCS Technology (SEPIE project – 27469)
Dermot Hore - Ireland

Dermot spoke about the social effect of using a requirements gathering technique in a small company. There had been a serious problem of Configuration Management as there was no traceability between code and changed requirements. The 2 tools chosen were a test planner and a configuration manager. The test planner was unusable in practice, being rejected by the staff, but the CM tool – despite a terribly old-fashioned and awkward user interface – proved very helpful and was well accepted. Management had failed to realize how necessary the tool was: they should have provided one per desk to enable employees to keep up with changes coming in from around the world.

The Labour Game Method
Vesa Torvinen - Finland

Vesa described the outline of a community-chest style game to explore aspects of the RE process. Unfortunately he did not illustrate his paper with actual examples of the game's materials, nor did he present any experimental findings.

Stakeholder Identification in the Requirements Engineering Process
Helen Sharp, Anthony Finkelstein, Galal Galal - UK

Helen said that little attention had been paid to finding out who the stakeholders were before we tried to get requirements from them. Four obvious "baseline" groups, she said, were users, developers, legislators, and decision-makers. One should go and observe (counsel of perfection!) work in its context, starting from the baseline stakeholders, and in each case find out which "satellite" stakeholders interacted with them. Interactions could be use, as in sharing data or products; but could also mean management/staff or customer/supplier relationships. By following such paths one could systematically chase down all stakeholders, exclude irrelevant actors, and also cross-check the list of relevant people. External agents and systems could also be stakeholders in some sense, as could people left out of proposed improvements (whose jobs might be at risk).

Ian Alexander asked whether developers were not poachers rather than gamekeepers, and Helen agreed that their stake was quite unlike that of the users, but still important.

The proceedings are available in the IEEE Computer Society book 'Tenth International Workshop on Database and Expert Systems Applications', PR00281, ISBN 0-7695-0281-4, 1999.

RE-Papers

An Integrated Framework for Requirements Change Impact Analysis

By Simon Lock and Gerald Kotonya
Lancaster University
{lock,gerald}@comp.lancs.ac.uk

Editor's note: This is a summary of the authors' paper presented at the Fourth Australian Conference on Requirements Engineering (ACRE'99)

1 Introduction

The introduction of a change to a single requirements may cause it to ripple through a system and impact on other requirements and broader organisational goals. The problem of identifying the total impact of a change is compounded by the size and complexity of relationships between require-ment artifacts. This can make the process of assessing the effect of change expensive, time consuming and error-prone.

Existing techniques for impact analysis can be problematic when applied to abstract requirement level entities. This paper outlines our work [1] in developing a novel new approach which integrates traditional impact analysis techniques with our own experience based approach. Our approach overcomes many of the problems associated with performing change impact analysis at the requirement level. This integrated approach is supplemented by features which take into account the uncertainty of the information on which analysis is based.

2 Techniques for impact analysis

A number of techniques are possible for identifying impact propagation paths between requirement level entities. These include existing techniques such as pre-recorded traceability analysis, dependency analysis and plain experience analysis. In addition to these, we have developed extrapolation analysis and an integrated hybrid approach. All these methods will now be discussed in more detail.

2.1 Pre-recorded traceability analysis

Pre-recorded traceability analysis is an existing technique which uses information recorded by developers, maintainers and end users to identify potential traceability links between requirements [2].

This information captures the relationships between the entities which make up the system and it's environment. Each recorded relationship has the potential to cause an impact to propagate from one requirement to another. Thus all relationship types must be considered when assessing a proposed change. The information used as a basis for performing pre-recorded traceability analysis must be recorded by the requirements engineer when the requirements are being specified.

2.2 Dependency analysis

Dependency analysis is an existing technique which aims to extract traceability information from behavioral models of the system in order to predict possible impact propagation paths [2]. The basis for extracting such information is normally a call graph which represents dependencies between the system entities. Both forward and backward calling relationships are extracted from the models to ensure that all possible impact propagation paths are identified. It is possible to determine potential impact propagation paths by tracing altered functional requirements (impact sources) to impacted functional requirements (impact targets) via the call graph. This is illustrated in Figure 1, where a change (C1) has a direct impact on the source requirement (FR.5) which can then result in indirect impacts on the target requirements (FR.1, FR.2, FR.3, FR.4, FR.8, FR.9) via the model calling graph.

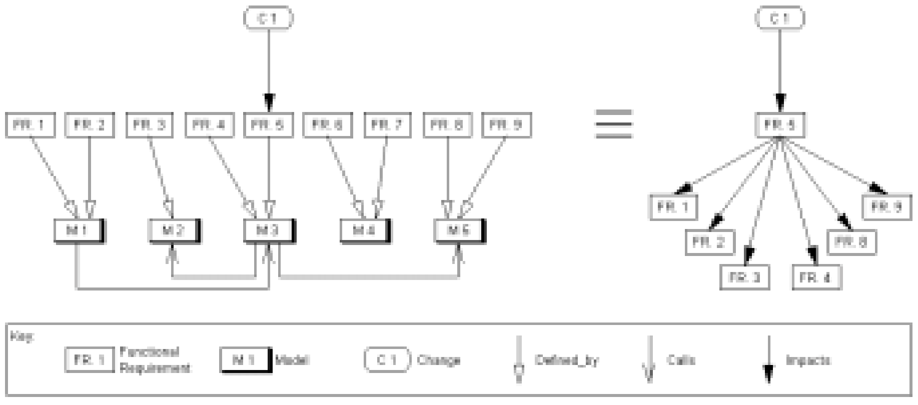

Fig. 1 Identifying propagation paths

A variety of models may be used to extract call graphs, ranging from structured textual descriptions through use-case scenario diagrams to pseudo code. The extraction of such call graphs may be performed manually, although improvements in efficiency and accuracy can be attained if automated analysis tools are available.

2.3 Plain experience analysis

Existing plain experience analysis approaches use a record of the effect of previous changes to the system as a basis for extracting traceability relationships. For a given change, traceability relationships can be assumed to exist between the change and requirements which it has previously effected. Thus each previous change provides us with a cluster of impacts which represent a small segment of the entire system propagation path structure.

2.4 Extrapolation analysis

Extrapolation analysis is a new technique which utilises past experience data to perform propagation identification [1]. The aim of this type of analysis is to project the small number of direct impacts specified for a proposed change into a complete set of both direct and indirect impacts. Extrapolation analysis is achieved by comparing the direct impacts specified for the proposed change with the actual total impacts of all previous changes. In this way it is possible to identify a number of previously experienced changes which contain the direct impacts of the proposed change as a sub-set of their total direct and indirect impacts.

It is possible to combine the total impact effect of a number of similar, previously enacted changes to develop a composite prediction of the impact effect of the proposed change. This not only maximises the set of projected impacts, but allows us to identify the most common, and therefore most likely impacts. The direct impacts of a proposed change are compared with the set of all impacts (direct and indirect) of previously enacted changes. The previous changes can then be ranked in order of similarity with the new change. The impact effects of most similar (i.e. highest ranking) previous changes are combined to produce a final set of impacts with which to extrapolate the effect of the proposed change.

2.5 Hybrid integrated approach

A number of problems exist when applying the above impact analysis techniques to requirement level entities. These problems include:

- Many methods are intended for use with concrete artifacts (e.g. source code) and are not suited to abstract requirements
- The limited focus of the individual methods leads to a limitation in system coverage
- Single approaches generally have a 'one shot' attempt at picking up propagation paths and are thus vulnerable to omissions in the requirement information
- The failure to distinguish between those potential propagation paths which actually propagate change, and those which do not
- The prediction of impact effects that are larger than would actually be caused by a change

Our proposed solution to these problems is an integrated analysis approach, which combines a number of individual techniques. This is supported by the inclusion of certainty measures to take into account the uncertainty of artifact information on which analysis is based. We believe that such an approach provides a basis for better impact analysis by:

- Providing a more sensitive and accurate analysis than the single paradigm methods
- Maximising completeness through improved system analysis coverage
- Counteracting the abstract nature of requirements
- Allowing a degree of contrast between potential impacts

- Providing an extensible approach to support change impact analysis across the entire system lifecycle

3 Combining individual techniques

If we wish to produce an integrated approach, we must combine the individual techniques in some way. This does not simply mean combining the results of each method for the entire system, but rather interleaving them so that they can work together in unison. A diagram showing the process used for this integration is illustrated in Figure 2.

We will first investigate the cyclic nature of the process and then move on to discuss the three key stages in more detail.

3.1 Analysis cycles

In order to identify the total potential impact of a change on a particular system, any analysis technique must be repeatedly applied to the system, forming a number of analysis cycles. This is true even for the hybrid technique and in fact forms an important aspect of the combination process.

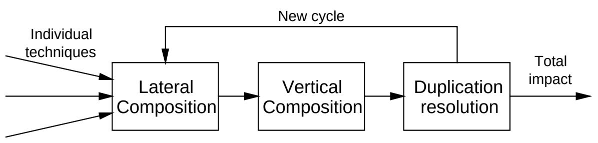

Fig. 2 Propagation structure generation

Each analysis cycle takes a set of impact sources (entities which must be altered to accommodate the change proposed) as inputs and produces a set of impact targets (entities which are impacted as a result of the alterations made to the sources) as outputs. Repeated analysis is required because the targets of the impacts can propagate the change impact to other entities and must therefore be fed into the next analysis cycle as impact sources [3]. This is illustrated in Figure 3.

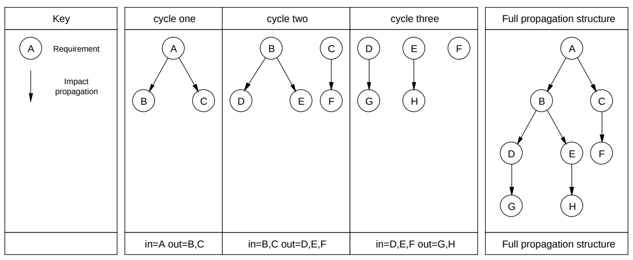

Fig. 3 Analysis cycles

3.2 Lateral composition

Lateral composition is the stage in which the outputs of individual methods are combined. This is achieved by summing all the propagation paths detected by the individual methods at each cycle of analysis. Figure 4 shows the combination of the results of two separate methods for a single cycle of analysis.

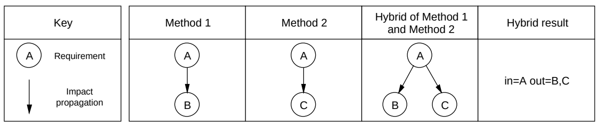

Fig. 4 Lateral composition

In order to allow propagation paths identified by different methods to be combined in this way, the results of each method must be weighted appropriately. This is done because the accuracy with which impacts can be predicted may vary depending on the reliability and focus of the method being used.

By weighting the results of each individual method it is possible to adjust them to reflect the relative accuracy of each approach. In this way the results of disparate techniques may combined while still ensuring approximate parity between predicted impacts.

3.3 Vertical composition

Vertical composition involves the chaining together of the results of each cycle of analysis in order to construct the complete propagation path structure. Each cycle of vertical composition adds an additional layer of predicted impacts to the propagation tree. This additional layer is the total set of impacts identified by the lateral composition stage of the current cycle. Figure 5 shows how lateral composition combines the results of each method and how vertical composition then takes these impacts to construct the next layer of the propagation structure.

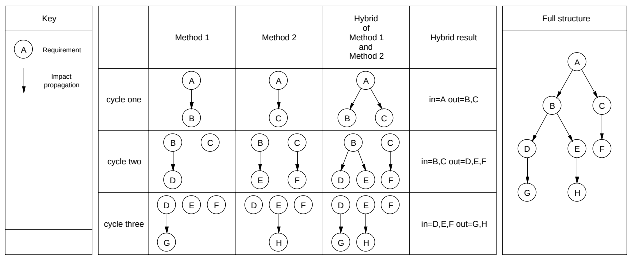

Fig. 5 Lateral and vertical composition

3.4 Duplication resolution

It is possible for duplicate propagation paths to be identified between two requirements. This results from the same impact being predicted by two different extraction mechanisms or multiple predictions from the same method. During duplication resolution these are converted into a single propagation path in order to minimise the size of the final propagation structure. Duplication resolution is done at the end of each cycle to minimise the number of inputs to the next cycle of analysis and prevent the replication of analysis.

4 Certainty analysis

We have developed a certainty analysis technique as part of this work to try to minimise the effect of the inherent incompleteness and uncertainty involved in performing impact determination with requirement level artifacts. Certainty analysis provides us with additional information to offset the deficiencies of the recorded information. To this end, the following additional items of information are collected for each requirement by the developers specifying the system:

- Degree of definition - the extent to which the requirement has been completely specified [4].
- Certainty of definition - the confidence with which the developers believe the entered information is correct

Using these two measures, it is possible to derive a quantitative estimation of the reliability of requirements and the traceability relationships between them. This additional information makes it possible to gain an appreciation of which impact predictions are most likely to be correct. Conversely, it is also possible to identify those predictions, which are less likely.

To produce a total certainty value for a particular impact, the two measures of the reliability of the target requirement are combined using the following formula:

$$C_{total} = DoD * CoD$$

Where C_{total} is the total certainty value, DoD is the degree of definition and CoD is the certainty of definition.

5 Conclusions

Our proposed solution to the problem of requirement level impact analysis achieves the composition of multiple traceability extraction approaches while maintaining approximate parity between predicted propagation paths. Our hybrid technique successfully integrates past experience based approaches with more standard dependency and pre-recorded traceability techniques. This integrated hybrid technique overcomes many of the problems of single paradigm approaches. The issue of uncertainty has also been addressed via the utilisation of certainty values to reflect the reliability of the requirement information.

To ensure the scalability of the approach a number of filtration mechanisms have been developed. These can be used to both minimise the set of impacts predicted by the process as well as providing important techniques for revealing different aspects of the system. In addition to this, the analysis of large systems is also eased by the availability of an integrated tool which supports all of the features described in this paper.

We are also currently investigating ways of extending our approach to support impact analysis in the later stages of system development. In this respect we are exploring ways of extending the technique to include artifacts from the design, implementation, testing and maintenance stages of the lifecycle.

References

[1] Lock S. and Kotonya G., *An Integrated Framework for Requirement Change Impact Analysis*, proceedings of the 4th Australian conf. on Requirements Engineering, September 1999

[2] Bohner S. A. and Arnold R. S., *Software Change Impact Analysis*, ix-xii, IEEE Computer Society, 1996

[3] Thompson, J., *What you need to manage requirements*, IEEE Software, pp115-118, March 1994

[4] Ecklund E. F. , Delcambre, L. M. L. and Freiling, M. J., *Change Cases: Use Cases that Identify Future Requirements*, SIGPLAN notices 31(10), pp342-357, October 1999

CORE-Blimey!

A regular column by Geoff Mullery, of Systemic Methods Ltd.

Political Requirements

There is a need in specification for support in handling the realities of politics – personal, institutional and cultural. In politics it is rarely tools which produce solutions to political problems – it is people interacting successfully. But tools can assist those interactions or ignore them – and in computing they appear largely to ignore them.

In this contribution I want to suggest some recurrent features of political interaction which are largely ignored or treated in an idealised fashion by specification support environments. They have their roots in the general problem of interactions of people in complex social structures and they certainly apply to system specification environments.

Regulation .v. Autonomy: There is always a conflict between forces of over-control (dictatorship) and over-freedom (anarchy). The gentler faces of these forces are regulation and autonomy. The practical implementation of politics seems to be that of balancing regulation against autonomy in such a way as to prevent the worst excesses of either influence.

Language and Culture: There is nothing better at exacerbating differences between people or groups than language and cultural barriers. One response is to attempt over-control by moving towards unification of culture and language – by annexation or media control. Another is to attempt separation via ultra-nationalism or xenophobia.

Threat and Response: Initially the majority of people do not much care about which of the minorities competing for power wins unless and until the victory starts to impinge adversely and severely upon their day-to-day lives. Hence power-seekers gain or keep power by making the "silent majority" perceive sufficient threat from rival power seekers to lead to active opposition.

Simplicity .v. Complexity: It appears a well established principle that if you want to carry popular support you should present a simple message. There appears to be a corresponding popular belief that there is a simple solution to every problem. The idea that some problems are complex and might require a complex solution is rarely popular.

Political Layering: Political interaction applies not only to relationships between people or between large organisations like nation states. They apply within and between many layers: people, shared interest groups, localised companies or cooperatives, nation states, international alliances and multinational groups.

Peer Rivalry: There is never a single instance of a ruling/ruled hierarchy. The existence of multiple instances leads to competitive confrontations. There are rival multinationals, rival international alliances, rival nations, rival companies and rival individuals.

Divided Loyalties: Any individual entity (a person or an organisation) may belong to several political groupings in the sense described here. A person may belong to a church, a profession, a political party and a company. An organisation may belong to an industry, a country, a trade body and a person or other organisation. The result is that the motives for a particular act or support for a particular policy can not always be regarded as the product of altruistic analysis. Rivalry in one area may stem from pre-existing or discovered rivalry in another.

Parasitic Groups: Where there are differences, rivalries or even just distinct identities there is always a place for individuals or groups which survive by virtue of a relationship with the problems (feeding them or easing them). For example there are often third party suppliers of goods, tools or services which sell to any of the rivals – either to facilitate the battles or reduce the differences.

If my past experience is any guide you will by now be asking what all this has got to do with methods or tools you develop and use. Technical specialists appear to me to be fanatically determined to ignore politics, which is why their efforts frequently succeed in a benign environment (one under their control or tutelage) and fail in a hostile one (a normal development environment).

It is the ignored and thence hidden nature of the problems deriving from politics which is a frequent major contributor to system development disasters. In the absence of some means of formally recognising and living with their existence the evidence for failure will always appear as something primarily technical – late or failed acceptance, invalid functionality or poor performance.

There are of course serious technical difficulties in developing large, complex, multi-company, multi-site systems. There are systems being built today which I certainly feel we are not competent to get right – and when they have safety implications I don't think we should be building them unless their absence has even worse safety implications.

But the technical difficulties are, in my experience, largely beside the point in all the major computer system development disasters I have encountered. The real reasons for dramatic failure have been failure to recognise and deal with inter-communication and political failures of the types I have described here. The influences I have mentioned are real and recurrent in the computer developers' world.

I must make clear what I am advocating here. I do not have a solution to the problems of politics. I am saying that they are guaranteed to be present on any large or long-running project. If that is true then it is fatuous to develop support environments which ignore their presence. More specifically it is fatuous to develop tools which attempt to allow only the expression of some close approach to perfection in specification.

I am saying that the overall specification support environment must actually support the presence of the various influences I described above. It must not ignore them or assume it can, by virtue of superior mathematics or logic conjure them away. What that is doing is worse than sweeping the problems under a metaphorical carpet. It is deception of ourselves and everyone else.

I assert that the idea, for example of a single consistent specification database (in the widest sense of the term database) is inappropriate. If you try to express only consistent high-quality details there is a mountain of inconsistent, low quality information which is relevant and which must be handled by some non-visible process.

I assert that the idea of a single controlling entity (a project manager, say) who totally controls what can be said and done in producing a specification is inappropriate. There will always be dissenting views and those views provide a perspective on the current official view of the specification which may provide a valuable warning to its later users – the development team.

I assert that the idea of a single standardised approach and language for system specification is inappropriate. It is almost invariably true that there is more than one way to perform a complex task and that there is more than one way to express a complex problem. The variations between people and groups means that forcing a single pattern on everyone will make significant numbers of those involved perform at least sub-optimally and sometimes badly.

I assert that imposition by a minority group – even if they are respected by management and/or researchers – of such a single approach/language combination is inappropriate. Not only will it cause sub-optimal performance by some –

it will also lead to active obstruction by others, who will then use the consequent problems to demoralise the rest of the team and discredit the approach with management for future projects.

I assert that the division of specification languages on an either/or basis between low content, low reading skill notations like bubble diagrams and high content, high reading skill notations like Z or VDM and the banning or down-grading of natural language and ad hoc expressions is inappropriate. There is an audience for each notation and there are overlaps between the information expressed in each.

In the area of natural language and ad hoc expressions there are things which need to be said which are naturally imprecise, but that does not negate their importance to those for whom the system is to be built. The fact that we can't make the requirements precisely contractual does not mean they have no part in a specification or even that they have no part in a contract. Reality in politics says we sometimes need to leave things vague to allow progress to be made.

I assert that there will always be divisions in the loyalties and technical priorities of those involved in development of systems. There will be personal, team and commercial rivalries which can not safely be suppressed in the name of a futile attempt to impose the constraints of non-redundancy, consistency, freedom from ambiguity and precision.

There will be a need to permit separation and duplication of information and transfer and translation between these separated locations. If you try to suppress it, the separation and duplication will only be achieved by some other, hidden means. By allowing for it and assisting it in the support environment you grant the possibility that the existence of the corresponding problems can be detected and that the possible locations of those problems can be recorded.

The overt existence of a problem which is certain to occur anyway is much more constructive than ignoring it and/or hiding it, leaving the uninitiated with the mistaken belief that there are no problems. One thing we have made some progress with in the technical area is in designing defensively so that where changes are likely they can be made with relative ease.

That is why knowing about problems or the possible need for future changes is easily as important as making a futile attempt to develop a system on the basis of a tightly defined, tightly controlled specification which has a number of impending but suppressed forces for change working away even as development and installation proceed.

RE-Publications

Book Reviews: Kovitz vs the Robertsons – a contrast in styles

Mastering the Requirements Process
Suzanne and James Robertson
Addison-Wesley 1999
£29-95 (hardback)
ISBN 0-201-36046-2

Practical Software Requirements
A Manual of Content & Style
Benjamin L. Kovitz
Manning 1999
£43-99 (paperback)
ISBN 1-884777-59-7

Reviewed by Ian Alexander

Two major Requirements Engineering texts have appeared this year, and both have been reviewed here in RQ (Kovitz in RQ 18, the Robertsons in RQ 17). It seemed worth asking what their opinions of RE were: whether there was a consensus, or if they held differing views, what these were and why.

Firstly, it is immediately clear that the books have different objectives: Kovitz limits himself to software, and stresses style, whereas the Robertsons are concerned specifically with process. In both cases the authors do a good job of keeping to, and meeting, the goals they have set themselves, so the two books to some extent address different markets – leaving aside the question of whether these are real or imagined.

Omitting a clear account of process is an obvious weak point in any book about a process such as RE. Kovitz has implicitly to answer the challenge: what is your process? He does, in fact, devote a chapter in the middle of the book (7) to the software development process (though he avoids the p-word in favour of talking about a 'division of cognitive labor'). He carefully maps tasks such as analysis and user-interface design to documents such as requirements and interfaces, and then in turn maps these to principal audiences. To some extent this focus on products sidesteps the p-question: Kovitz effectively challenges the reader to select any development process that includes the tasks he mentions, and communicates their results adequately.

Even this brief glimpse of Kovitz' approach may be enough to hint at where he is coming from, and the suspicion is confirmed by the cover blurb: "he has worked as programmer, tester, system analyst, user-interface designer, and technical writer". This bespeaks an impressive personal knowledge, but perhaps specifically not a process view, and maybe especially not a systems view. Kovitz aims at and achieves great clarity of human communication with other engineers, just as in his user-interface work (and there are excellent UI examples in the book, including a fully-worked Bug Log which, as he says, 'interfaces only to people') he certainly helped systems to communicate with their users.

Kovitz is extremely readable: in fact, you can even relax with it and laugh with him at pieces of Jargon (section 12.5) and Small Details (chapter 15, as entertaining as all those books on Plain Words, and more relevant). He is also the most quotable writer on systems engineering so far published: perhaps, unfortunately, this is not yet necessarily a high distinction. It is with a mixture of pain and the pleasure of recognition that we read that 'a sprawl of use cases is a terrible problem frame', or that 'many requirements documents … double their size by including software requirements that are identical to the user requirements, with just the wording changed'.

In short, Kovitz has written a fresh, lively, honest, funny, and provocative book on RE – not easy to do.

Like Kovitz, the Robertsons have written a book based solidly on their own experience. James is a management (i.e. requirements!) consultant, and Suzanne would have been called an expert Requirements Engineer quite a few years ago, if the term had been invented then. They write clearly and plainly, if without the Kovitz sparkle.

One of the Kovitz jargon-words, Non-functional Requirements (a requirement that doesn't work?) becomes a chapter heading, and a basic term used throughout the Robertsons' book. Evidently the approach is more traditional, at least as far as nomenclature is concerned. The chapter, like many in the book, reveals Suzanne's background in analysis, as Yourdon-style dataflow diagrams relate processes to products. Fortunately, stakeholders make an appearance, communicating their 'wants and needs' and (startlingly) receiving the 'rejects' (rejected requirements) from the quality gateway: the accepted ones make it into the 'requirements specification' (a mixture of terms thoroughly panned by Kovitz).

The Robertsons do live in the modern world, though: they have a clear idea of identifying objects as 'things' and 'work

RE-Bites...

Give poisonous houseplants to childless friends.

(Taken from: 'What to expect when you're expecting' by Eisenberg, Murkoff and Hathaway, Simon & Schuster; ISBN: 068481787X) areas', and they skilfully combine old-style analytical thinking about topics such as work context with a practical treatment of use cases. In fact, they could probably stake a stronger claim to being practical than Kovitz (who risks the word in his title), as they propose a fully-worked out method, VOLERE, which they explain in detail in the book.

The Robertsons are efficient and systematic engineers, and it is comforting for the reader to come across statements like 'This is how we intend to proceed:' – followed by, in the Event-driven Use Cases chapter, a straightforward bulleted list of steps leading neatly to 'Derive the requirements for each use case' at the end. Of course the list is a scenario, even if the reader does not know it at that stage: the Robertsons practise what they preach.

One of the real strengths of the Robertsons' book is the inclusion of templates for each of the work products that the book advocates. For example, a non-functional requirement is illustrated as a card, pre-printed with attributes for Requirement #, Type, Event/use case #, Description, Rationale, Source, Fit Criterion (this is a key concept), Customer Satisfaction, Customer Dissatisfaction (!), Dependencies, Conflicts, Supporting Materials, and History. It is at once clear that the approach is thorough, pays proper attention to people, is perfectly suitable for use with paper and card-indexes instead of computers, and may not suit every RQ reader. To be fair to Kovitz, his Small Details chapter does include perfectly serviceable templates, to the extent of recommending a page layout.

So, whereas with Kovitz the beginning Requirements Engineer may not be entirely clear what to do when (despite the excellent advice and examples), with the Robertsons the danger is rather that the sheer amount of detail on what has to be done may seem overwhelming. There are probably many topics on which they would agree: for instance, the need to identify stakeholders early on, and to find out from each of them personally what they need. The two books together make fascinating reading; both of them are essential background (and possibly foreground) reading for all would-be and practising systems engineers.

Editors' note: Individual reviews of both the Robertsons' (issue 17) and Kovitz's (issue 18) books can be found in previous issues of RQ.

RE-Calls

Recent Calls for Papers

ICSE 2000 – The New Millennium: 22nd International Conference on Software Engineering, Limerick, Ireland, June 4-11, 2000

http://www.ul.ie/~icse2000

Important dates:
The deadlines for submissions of papers and workshop and panel proposals has now passed. However, submissions for Research demonstrations and posters, doctoral symposia and teaching demonstrations may be submitted until 25th February 2000.

CAiSE 2000 – 12th Conference on Advanced Information Systems Engineering, Stockholm, Sweden, June 7-9, 2000

http://www.ul.ie/~icse2000

Important Dates:
The deadlines for submissions of papers and workshop, panel and tutorial proposals has now passed. However, submissions for posters may be submitted until 1st March 2000.

REFSQ 2000 – Sixth International Workshop on Requirements engineering: Foundation for Software Quality, Stockholm, Sweden, June 7-9, 2000 (Preceeding CaiSE 2000)

http://www.ifi.uib.no/konf/refsq2000/cfp2000.html

Scope:
One way to achieve the goals of RE is to enhance the methods used. Method Engineering (ME) has sought ways to develop new methods and tools for developing information and software systems. In this workshop we seek innovative method development approaches which aid in the early stages of systems development. Possible theme topics include new methods for RE, methodical approaches to distributed development and advanced methods for rapid application development. Of particular interest are methods that aid in end-user participation and facilitate building of shared understanding among all stakeholders. REFSQ'2000 also invites general submissions addressing a wider range of RE-issues, such as: understanding and improving RE-process; RE for Commercial Off-The-Shelf (COTS) systems; empirical evaluation of RE methods and tools; and empirical studies of RE practice in industry. However, papers addressing other areas of RE are also welcome.

Submissions:

FULL PAPERS (max 6000 words) Full papers should emphasise what is new and significant about the chosen approach and adequately compare it with similar work. Integration of the contributions with mainstream or other research approaches to SE and RE are especially encouraged. The maximum length of a full paper is 6000 words.

POSITION PAPERS (max 2000 words) Position papers should state the author's research position with respect to view of current RE practice, relations between current RE practice and RE research, and/or research methodology and ontological assumptions. Papers should emphasise which topics that are of particular concern to the RE community at present, and why. The maximum length of a position paper is 2000 words.

INDUSTRIAL PROBLEM STATEMENTS (approx 1 page) People from industry are especially encouraged to submit problem statements. Industrial problem statements should focus on perceived mis-matches between current RE practice and research and/or on emerging areas of concern for RE practitioners. An industrial problem statement may be about one page long, but more comprehensive statements will also be considered.

Papers should be submitted by e-mail in RTF- or PDF-format before **February 28th 2000** (hard arrival date) to:

Matti Rossi
Helsinki School of Economics
P.O. Box 1201
FIN-00101 Helsinki
Finland

Important Dates:

Submission deadline:	February 28th, 2000
Notification of acceptance:	April 10th, 2000
Camera-ready paper due:	May 2nd, 2000

RE-Sources

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: http://www.soi.city.ac.uk/homes/gespan/rq/rq.html

Web Pages

The web address of the RESG is:
http://www.cs.york.ac.uk/bcs/resg/

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

An interesting collection of 72 papers (!) and a description of an ESPRIT project on co-operative requirements engineering with scenarios. The CREWS project has developed two prototypical tool suites which can be employed, e.g., as extensions to the Use Case approach in object-oriented systems engineering. One shows traceable multimedia-based current-state analysis and animation of future scenarios, the other provides guidance for the creation and analysis of text scenarios and for the systematic generation of exception scenarios.

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/

Mailing lists

The SRE list

The SRE mailing list aims to act as a forum for exchange of ideas among the requirements engineering researchers and practitioners. To subscribe to SRE mailing list, send e-mail to listproc@jrcase.mq.edu.au with the following line as the first and only line in the body of the message:

subscribe SRE *your-first-name your-second-name*.

RE-Actors

The committee of RESG

Chair: Dr. Bashar Nuseibeh, Department of Computing, Imperial College, 180 Queen's Gate, London SW7 2BZ, UK. E-Mail: ban@doc.ic.ac.uk, Tel: 0171-594-8286, Fax: 0171-581-8024

Treasurer: Dr. Neil Maiden, Centre for HCI Design, City University, Northampton Square, London EC1V OHB, UK. E-Mail: N.A.M.Maiden@city.ac.uk, Tel: 0171-477-8412, Fax: 0171-477-8859

Secretary: Dr. Wolfgang Emmerich, University College London, Department of Computer Science, Gower Street, London WC1E 6BT, UK. E-Mail: W.Emmerich@cs.ucl.ac.uk, Tel: +44 171 504 4413, Fax: +44 171 387 1397

Membership Secretary: David Shearer, Department of Computer Science, University of Hertfordshire, Hatfield, AL10 9AB, UK.

RE-Creations

To contribute to RQ please send contributions to Peter Sawyer (sawyer@comp.lancs.ac.uk). Submissions must be in electronic form, preferably as plain ASCII text or rtf.