Requirenautics Quarterly

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

 Issue 11 (July 1997)

RE-Locations

RE-Soundings

Editorial

Welcome to the first RQ produced by the new editorial team, with Ian Alexander and George Spanoudakis contributing greatly to the quality of this issue (the typos, however, are all mine). It's getting pretty crowded in the editorial broom cupboard though! Which reminds me, there's two new books stuffed under my seat, free to the first person to volunteer to review them: Sommerville & Sawyers' RE Good practice guide, and Heitmeyer and Mandrioli's Formal Methods for Real-Time Computing. Any volunteers out there?

Please note: The copy deadline for the next issue has been brought forward to **12th September 1997**, to allow us time to get the issue out by RE-day.

Steve Easterbrook,
NASA IV&V Facility, Fairmont WV

Chairman's Message

Well, the academic year may be over, but we are in the first month of RESG's operational calendar. The group's AGM was held on 11th June, and I'd like to welcome three new elected members to the RESG Committee. Ian Alexander and George Spanoudakis have joined as Associate Editors of RQ, and Carol Britton is our new Events Officer. The RESG's activities and newsletter have grown steadily over the last three years, and the contributions of Ian, George and Carol will be particularly welcome.

The minutes of the AGM are included in this issue, and the contact details of all the committee have been updated. Please feel free to contact any of us if you have any questions, suggestions or if you would like to contribute to the group's activities or publications.

The group is actually taking a break during July and August, but we are back with bang in September! To warm up, we are holding a joint meeting with the BCS Software Reuse Specialist Group in Manchester on 10th September, but the main event of the RESG calendar has to be RE-Day on 30th September. A preliminary programme is included in this issue, but there are many more surprises planned. RE-Day will include a keynote, tutorials, workshops, tools & publishers exhibition and a variety of smaller activities to keep you entertained during coffee breaks. If you have any interest in requirements engineering, you really should not miss RE-Day. And the cost? It's FREE!! Just mark it in your diary and get down to the Holiday Inn, King's Cross, London at 9:30am on the 30th September. Enough said.

Best wishes for an enjoyable summer.

Bashar Nuseibeh,
Imperial College, London

RE-Treats

Forthcoming events organised by the group.

Requirements Acquisition and Modelling: The Impact of Reuse

Wednesday September 10, 1997

An afternoon panel session organized in conjunction with the BCS Reuse Specialist Group.

Location: UMIST, Manchester (TBD)

Time: 2:00pm to 5:00pm

Cost: free to members, £5 to others

Abstract: To many people the idea of reuse is the retrieval, composition and adaptation of program code. However, structured methods and CASE tools place greater emphasis on the earlier phases of systems development; more analysis and design artifacts are available for reuse. More recently there has been interest in reuse during the requirements engineering phase of systems development. In some organisations there is considerable reuse of requirements engineering artifacts such as existing specifications, domain models and data bases containing generic requirements. This seminar explores the potential for and impact of reuse during the requirements engineering process.

The seminar will bring together practitioners and academics with experience of reuse during requirements engineering. Particular emphasis will be placed on the potential for reuse, and techniques for best achieving this reuse.

Requirements Engineering: National Awareness Day

Tuesday, 30th September 1997, 9:30-5:30pm

Venue: King's Cross Holiday Inn (Accommodation Details)

Cost: free to all

Everything you needed to know about requirements - and would like to ask! 'Requirements Engineering - National Awareness Day' (RE-Day) is the UK's premier Requirements Engineering event of the year. If you are involved in any aspect of Requirements Engineering, academically or industrially, then this is the event that you cannot afford to miss!

The purpose of RE-Day is to showcase the currently available best practice in Requirements Engineering. The day will contain a number of timetabled events interleaved with open tool vendor and other display sessions.

Details for the finalised events are as follows:

- Keynote address “Requirements: Rough Sketch or Precise Specification?” by Suzanne Robertson (Atlantic Systems Guild)
- A selection of the best-selling Requirements Engineering tools will be on display
- A workshop programme covering a wide variety of practical topics, where you can try examples of the latest technology
- Tutorials on “Requirements Engineering for Systems” by Tom Docker, “UML for Requirements Engineering” by Jim Arlow, and “RAD and Requirements Engineering” by John Crinnion
- Research posters from leading European academic centres of Requirements Engineering excellence
- An interactive requirements elicitation exercise
- A 'Room 101' vote where attendees will have the chance to decide which current topic of academic Requirements Engineering research should be confined to the Room 101 dustbin, never too be considered again
- …And a 'Get on Your Soapbox' session!

The RE-Day Keynote address:
“Requirements: Rough Sketch or Precise Specification?”

Suzanne Robertson, Atlantic Systems Guild

Abstract: The role of a requirements engineer is to specify precise, consistent, testable requirements that provide a solid foundation for building the new system. However the source of the requirements, human beings, does not help to satisfy the goal of precision. Humans have different experiences, personalities, priorities and use of language, and these differences influence the way we state and hear requirements.

Given that precise statements of requirements are not the natural currency of human communication, it is folly for the requirements gatherer to assume that what he hears, especially the first version, is what is wanted.

We can, and we must, tailor our requirements gathering to expect imprecision. Each fragment of requirement can be captured in the form of a rough sketch, and progressively nurtured and refined until it can be given all the attributes of a measurable and testable requirement.

By fitting each requirement into a structure, it is far more convenient to progressively assess each requirement, and its interactions with other requirements, until it reaches its final stage of precision. By not immediately committing to a final version of the requirement, we are in a better position to react to the inevitable changes.

This talk discusses how to use the concepts of rough sketch, requirements template and requirements shell to mould imperfect wishes into a precise requirements specification.

RE-Day Mini-Tutorial 1: Requirements engineering for systems

Tutor: Tom Docker, Director, CITI Limited

Requirements engineering for systems is much more than capturing the problem. It is essentially managing the expectations of large numbers of stakeholders to deliver products with clear business benefits, when a system can be in development for a significant number of years (e.g., a new railway service).

A characteristic of many systems is that component subsystems have different rates of development and maturity, which introduces a potentially critical timing dimension to producing an adequate overall solution.

A relatively long development life frequently means that the profile of claimed benefits change and that, if the problem is not adequately described, the requirements themselves are liable to change.

A further area of concern is the likelihood that different subsystems will, in reality, be provided by different 'subcontractor' groups. Consequently, traceability and configuration management become key factors.

In this tutorial, key issues are discussed related to the identification and structuring of requirements and their mapping to adequate solutions, to ensure that monitoring of the products and benefits can be maintained throughout the product development lifecycle.

***About the tutor**: Dr Thomas Docker has worked in both business and academia. He has worked in the IT function of a number of commercial and government organisations. He was a business consultant for a computer manufacturer, where he assessed the impact of future systems in the commercial arena, and defined strategic and tactical products. He taught computing and IT at Massey University in New Zealand. Tom was also managing director of a consultancy company that helped determine management strategies and business processes.*

He is the author of numerous papers related to project management, the specification of products, and management education. He is director of the postgraduate programme in business process management, and is a visiting fellow of the University of Bradford, as well as being an invited speaker at conferences on project and product issues.

Tom supports organisations in the project delivery and management of products to realise business benefits. This includes business modelling, stakeholder management, management of change, and managing the relationship between projects and products. He has helped a number of large government and commercial organisations to review and modify the way that they specify and manage products, and has particular interest in the specification of requirements related to benefits realisation.

RE-Day Mini-Tutorial 2: UML for Requirements Engineering

Tutor: Jim Arlow

The Unified Modelling Language (UML) of Booch, Rumbaugh and Jacobson is defining the strategic direction for object-oriented methods. This tutorial focuses on those aspects of UML which may be effectively used in Requirements Engineering. In particular, it presents Use-Cases, Scenarios, Class Diagrams, Sequence Diagrams and Collaboration Diagrams which are the artefacts generated in the Analysis phase of the OO project lifecycle. Basic concepts and UML syntax will be discussed, and the mapping of high level system requirements (Use Cases) down to individual Classes will be demonstrated.

***About the tutor**: Dr. Jim Arlow has been involved in OO methods since 1990 and is currently constructing the Enterprise Object Model for British Airways. When not working for BA, he also works as a trainer and consultant for QA Training, Rational and LogOn Technology Transfer. Highlights of his career include winning an award at Object World Frankfurt for the best OO development environment and helping to introduce object technology to BA. Jim has also done training and consultancy for Mannesmann, Standard Life, DEC, IBM and Burlington Northern Santa Fe Railroad.*

RE-Day Mini-Tutorial 3: RAD and Requirements Engineering

John Crinnion, City University

This mini-tutorial will examine the principles and practice of rapid and evolutionary systems development methodologies, and will discuss how they address the requirements engineering aspects of business applications development. The tutorial will comprise three parts:

1. Principles and definitions: What is RAD, RAD & structured systems development, RAD & evolutionary development, RAD & O-O, The DSDM Methodology
2. Video, illustrating RAD in practice
3. The RE implications, both theoretical and practical: Converging ideas in systems development.

***About the tutor:** John Crinnion is a senior lecturer at the City University in London. He is an experienced practitioner, who has been writing and speaking on the subject of RAD and evolutionary development since the late 80's. He has worked extensively as a consultant on RAD projects of all sizes, and is the author of the book ' Evolutionary Systems Development' (published by Pitman 1991.*

Further details of RE Day can be found at:
http://www.cs.york.ac.uk/bcs/resg/reday.htm

You can't afford to miss it - it's FREE!!!

Quality Certification for the RE Process

Wednesday November 26, 1997

Location: Room 418, Huxley Building, Imperial College, London

Time: 2:00pm to 5:00pm

Cost: free to members, £5 to others

Industrial Experiences in RE

Wednesday February 4, 1998

Location: York University

Time: 2:00pm to 5:00pm

Cost: free to members, £5 to others

RE-News

Calendar

July 1997

Second IFIP International workshop on Formal Methods for Open Object-based Distributed Systems (FMOODS'97), Canterbury, UK, 21st-23rd July, 1997.
http://alethea.ukc.ac.uk/Dept/Computing/Research/NDS/FMOODS/

August 1997

Fifth International Conference On Conceptual Structures (ICCS'97), Seattle, USA, August 4 - 8, 1997.
http://www.cs.uah.edu/~iccs97/

The Twenty-First Annual International Computer Software and Application Conference (COMPSAC 97), Washington DC, August 13-15, 1997. *Info: paulra@acq.osd.mil*

Designing Interactive Systems: Processes, Practices, Methods, And Techniques, (DIS 97), Amsterdam, The Netherlands, 18-20 Aug 1997. *http://www.acm.org/dis97/*

Fourth ISPE International Conference on Concurrent Engineering: Research and Applications (ISPE/CE97), Troy, Michigan, USA, August 20-22, 1997.
http://www.secs.oakland.edu/SECS_prof_orgs/ISPE/CE97.htm

Second International Conference on Cognitive Technology, (CT'97) "Humanising the Information Age", University of Aizu, Japan, 25 - 28 August 1997.
http://www.u-aizu.ac.jp/news/news.html.

September 1997

Second International Conference on Coordination Models and Languages (COORDINATION'97), Berlin, Germany, September 1-3, 1997.
http://www.wins.uva.nl/research/coordination/

The 16th International Conference on Computer Safety, Reliability and Security (SAFECOMP'97), University of York, UK, September 8th-10th, 1997
http://www.cs.york.ac.uk/safecomp-97

Third IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'97) Villa Olmo, Como, Italy, September 8-12, 1997.
http://www.elet.polimi.it/iceccs97

The Fourth NASA Langley Formal Methods Workshop (Lfm97), Hampton, Virginia, U.S.A, 10-12 September 1997.
http://atb-www.larc.nasa.gov/Lfm97/

Fourth International FME Symposium (FME'97), Technical University of Graz, Austria, 15-19 September, 1997.
http://ist.tu-graz.ac.at/fme97

Sixth European Software Engineering Conference (ESEC), and Fifth ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE), Zurich, Switzerland, September 22-25, 1997.
http://www.ifi.unizh.ch/congress/esec97.html

Eighth Specification and Description Language (SDL) Forum, Evry, France, 22-26 September 1997. *http://alix.int-evry.fr/lor/SDL97/*

October 1997

OOPSLA '97 Workshop, Atlanta, Georgia, USA, October 6, 1997. *http://www.forsoft.de/~rumpe/oopsla-ws/*

10th European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW-97), Sant Feliu de Guixols, Catalonia (Spain), October 15 - 18, 1997.
http://www.acia.org/ekaw

The 1st International Enterprise Distributed Object Computing Workshop (EDOC'97), Marriott Resort, Gold Coast, Australia, October 24-26, 1997.
http://www.dstc.edu.au/events/edoc97/

The Second Australian Workshop on Requirements Engineering (AWRE'97), Macquarie University, Sydney, Australia, 27th Oct 1997.
http://www-comp.mpce.mq.edu.au/~didar/awre97.html

November 1997

The 8th International Symposium on Software Reliability Engineering (ISSRE'97), Albuquerque, New Mexico USA, November 2 - 5, 1997.
http://admin.one2one.com/issre97

IASTED International Conference on Software Engineering, San Francisco, USA, November 2-5, 1997.
http://www.iasted.com/sanfran/se97.html

International Workshop on (Situated) Method Engineering, Delhi, India, 3-4 November 1997. *Info: cvs@sms.utwente.nl*

12th IEEE International Conference on Automated Software Engineering (ASE'97) (Formerly the Knowledge-Based Software Engineering Conference [KBSE]), Hyatt Regency Lake Tahoe, Nevada; USA, November 3 - 5, 1997. *http://ic-www.arc.nasa.gov/ic/conferences/ASE97*

Fourth International Symposium on Software Metrics (Metrics '97), Albuquerque, New Mexico USA, November 5 - 7, 1997. *http://www.cs.pdx.edu/conferences/metrics97/*

Fourth International Conference on Information and Knowledge Management (CIKM'97), Baltimore, Maryland, USA, November 8-11, 1997. *http://enws396.eas.asu.edu/cikm97/cikm97.html*

4th International Conference on Object-Oriented Information Systems (OOIS'97), Brisbane, Australia, 10-12 Nov 1997. *http://www.cs.uq.edu.au/conferences/oois97*

Symposium On Current Trends In Software Engineering, Louvain-la-Neuve, France, November 15, 1996. *http://www.info.ucl.ac.be/event/CTSE.html*

The Second International Workshop on CSCW in Design, Bangkok, Thailand, Nov 26-28, 1997. *http://www.chinavigator.co.cn/edu-sci/cscwd97.htm*

December 1997

Joint 1997 Asia Pacific Software Engineering Conference (APSEC) and International Computer Science Conference (ICSC) Hong Kong, 2 - 5 December 1997. *http://www.comp.hkbu.edu.hk/~apsec*

January 1998

Engineering Complex Computer Systems Minitrack, part of the Emerging Technologies Track of the 31st Annual Hawaii International Conference on Systems Sciences (HICSS), Big Island of Hawaii, HI, January 6-9, 1998. *http://www.ce.unipr.it/hicss*

March 1998

Second Workshop on Formal Methods in Software Practice (FMSP98), Clearwater Beach, Florida, USA, 4-5 March 1998. *http://www.bell-labs.com/user/maa/fmsp98*

European Joint Conferences on Theory and Practice of Software, Lisbon, Portugal, March 30 - April 3, 1998. *http://www.di.fc.ul.pt/~llf/etaps98/*

April 1998

Third IEEE International Conference on Requirements Engineering (ICRE'98), Colorado Springs, Colorado, USA, April 5-10, 1998. *http://www.cs.technion.ac.il/~icre98/*The Eleventh Workshop on Knowledge Acquisition, Modeling, and Management (KAW'98), Banff, Alberta, Canada, April 18-23, 1998. *Info: gaines@cpsc.ucalgary.ca*

The 20th International Conference on Software Engineering (ICSE'98), Kyoto, Japan, April 19-25, 1998. *http://icse98.aist-nara.ac.jp/*

The 1st IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC '98), in conjunction with The 4th IEEE Workshop on Object-oriented Real-time Dependable Systems (WORDS '98) and The 4th International Workshop on Object-Oriented Real-Time Systems (WOORTS '98), Kyoto, Japan, April 20 - 22, 1998. *http://dream.eng.uci.edu/isorc/*

September 1998

5th International School and Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'98) Lyngby, Denmark, Sept 14-15 (School) and Sept 16-18 (Symposium) 1998. *http://www.it.dtu.dk/~ftrtft98*

RE-Readings

Reviews of recent Requirements Engineering events.

Safety-Critical Issues in Requirements Engineering

University of York, 18th September 1996

Report by Andy Vickers

On the grounds of better late than never, here is a short write up of a very interesting RESG event that took place at the University of York in September last year (!). The event was titled "Safety-Critical Issues in Requirements Engineering" and was given by three of the leading figures in the field: John McDermid (University of York), Barry Hebbron (University of Teesside), and Amer Saeed (University of Newcastle). The forty or so attendees were treated to a 30-minute presentation from each speaker with the afternoon concluded by a lively panel session.

John McDermid started the afternoon off by introducing the audience to some of the basics of safety and outlining why safety is an issue at the requirements stage of the life-cycle. Requirements can only be considered 'safe' or 'unsafe' (and even these terms are not cut and dry, depending upon the perception of the viewer) when viewed in context. There are therefore particular issues and activities that must be considered during the requirements phase for a safety critical project. Unfortunately, safety requirements are not only found at the requirements phase, they also emerge during subsequent phases of development. The safety engineer must therefore also be alert to the implications of requirements identified throughout the life-cycle. McDermid's introduction provided suitable context for the

next two speakers who concentrated on more technical aspects of the topic.

Amer Saeed explored some of the work in this area that has been undertaken by the University of Newcastle. Saeed discussed the issues involved in analysing the safety properties of requirements for a particular class of systems - process control systems - and suggested particular processes that could be followed to promote requirements engineering from a safety perspective. Barry Hebbron's talk discussed the application of the HAZOPS technique (originally from the Chemical Process Industry) to Ward-Mellor requirements models. HAZOPS is a very effective, though expensive, technique that has seen increasing application in the software design field. Hebbron has investigated the technique and found it to be useful in identifying hazardous requirements - although still expensive in application.

The afternoon was then concluded with a panel session, with most discussion focussing on the kinds of skills that would be most desirable for a requirements engineer working in the safety field. The question was whether requirements engineering skills were more important, or domain knowledge. As you can imagine, opinions were divided, though it was clearly an issue that practitioners should be aware of in choosing staff to undertake requirements engineering activities within this domain.

Requirements for Off-The-Shelf Systems & Software

The June meeting of RESG was a half-day seminar on engineering requirements for selecting off-the-shelf software systems. The guest speakers were Eddie Williams of Rolls-Royce & Associates, Gareth Somerset of Systems Engineering and Assessment Ltd., David Law an independent consultant, Cornelius Ncube of City University and Tim Kelly of the University of York.

Report by George Spanoudakis & Ian Alexander

Eddie Williams presented an industrial perspective over requirements for COTS, which are used in software development by companies for whom this development is not the main activity. Eddie pointed out that although the cost-saving benefits of using COTS are attractive and industry has been keen on using them for a number of years, in reality the adoption of COTS presents serious problems. The underlying reason for these problems is that the way of constructing software that a company has evolved over a number of years is often in disparity with the development processes implied by existing off-the-shelf components. As a consequence the success or failure of using COTS depends largely on the extent of these disparities. The least-pain approach is to seek out components that are sufficiently generic (such as word processors, spreadsheets and databases) both in functionality and implied development approach. These components can be easily integrated into existing development practices (as development tools) or systems (as parts of the final product). For non-sufficiently generic components a solution could be to "stick" them to developing systems using "glue" software. However this is not without problems. Most of these problems have to do with the lack of full confidence in a system constructed in this way, especially if it is a safety-critical system. In such cases, the only recipe Eddie had to offer was intensive testing, admitting that this is very expensive (usually it counts for half of the budget) and can never be exhaustive. The discussion which followed made apparent the concerns of the audience about the risks involved in this approach, the need to write "wrappers" (software components that conceal unwanted interfaces of COTS) which comply with specific standards, and the scope for further research and development in this area.

Having been concerned with similar problems, Tim Kelly pointed out that what usually disturbs the harmonious use of COTS in a "design and procure" approach to software construction are secondary features of off-the-shelf components. These features usually turn out to be at odds with the preferred design methodology. Clearly, the influence of an off-the-shelf component to the system development approach has to be fully appreciated from early on in the development lifecycle. This may only be based on detailed information about the design of components. Tim suggested the description of the characteristics of COTS (such as their UI concepts, design basis and implementation structure) using "patterns" which could then be used as the basis for assessing the "fit" of these components with the architecture of the overall development. In the subsequent discussion it was suggested that further research in this area ought to consider the applicability of techniques that have been developed for software reuse but keeping in mind that these techniques are often tuned for components of low-granularity.

Gareth Somerset focused on the dynamic nature of the life of requirements and how this affects procurement of equipment giving examples drawn from his experience as a procurement consultant for the MoD. Large organisations (like the MoD) have moved towards elaborated procurement activities embedded in their corporate psyche in response to the need to conduct procurement on a competitive and accountable basis. They have also paid attention to the birth and early life of requirements, which are generated as a part of the procurement process. However, they have been less concerned with the evolution and role of requirements after the award of the contract. Quite often during this period requirements are subjected to assaults from organisations and contractors who may have an interest in adding new requirements or having some of the existing ones modified or even deleted. Requirements must also survive to provide the customer with a basis for contractual acceptance. This is a messy rolling process that ties requirements to tests (in the broadest sense) in a tangle of interdependencies. In such settings requirements capture, specification development, decision support and continuous traceability of requirements right through to acceptance testing are vital. Somerset presented a procurement tool marketed by SEA Ltd. called

TMA, which supports requirements capture (Tracer), tender assessment (Marker) and contract acceptance (Acceptor) in a structured and traceable way. TMA is based on Microsoft's Access, Word, and Excel.

David Law presented a framework for expressing requirements for procurement influenced by "feature analysis". Feature analysis is a technique that may be used in the assessment of a variety of products and services. These may range from consumer to high technology products and services. Feature analysis builds upon the results of the DTI/SERC sponsored DESMET Project. The presentation focused on the use of the technique for evaluating methods and tools used in the development and maintenance of software-based systems. In such settings often hundreds of off-the-shelf products may appear to be available and relevant. The goal of the assessor is to use as many of the features which appear to be mandatory as possible to eliminate most of the field before the more costly process of comparing and evaluating similar compliant products. Features may include technical characteristics and effects of the products being evaluated as well as other factors such as quality, cost, experience of, and support available from the supplier. In some cases they may even include cultural factors. The importance of each feature has to be determined and the compliance of the products with it has to be assessed. This assessment may be a simple inspection of the literature, an informal test, or a carefully designed experiment with controls and in many cases it is inevitably subjective.

Cornelius Ncube reported on problems identified during the selection of a complex COTS software system for the MoD and suggested some solutions to these problems. The difficulty of assessing product-requirement compliance, the inability to generate measurable compliance tests in many cases and the biases in subjective evaluation were some of the most prominent problems. All of them arise from the complexity of the products to be evaluated. Equally problematic were the lack of effective product demonstrations by suppliers, the inconsistent assessment of the importance of requirements by the customer, the time spent on the acquisition and consideration of non-discriminating requirements, and the failure to record the rationale for decisions made during the procurement process. Cornelius suggested that there are ways and techniques that could be useful in dealing with at least some of these problems. These include the active involvement of customers all the way through the process and the use of benchmarks for product evaluation. The simultaneous acquisition of requirements and product evaluation, the discrimination of the core of a product from its peripheral parts and the exploration of its behaviour in the absence of some of these parts could also be useful. Finally, decision-making and design rationale recording techniques could be used to provide a more informative trace of the procurement process. Cornelius argued that the application of these techniques in a systematic way requires proper guidance based on heuristics and a normative model of the entire procurement process.

International Workshop on Representations in Interactive Software Development

Queen Mary & Westfield College, London, 2-4 July 1997

Report by: Ian F Alexander

There never seems to be much point writing up a conference or workshop in the traditional "Dr Jones discussed the importance of" style for the sufficient reason that the proceedings say all that anyway. In these days of the World Wide Web, it is easier than ever before to get hold of papers (and authors) to see what was formally said. But the life of a workshop consists precisely in what was NOT formally said: we can be academic about this, quoting Polanyi on Tacit Knowledge, or simply refer to the well-known conference-goers' adage, that the only real work takes place outside the meeting, in the corridors and coffee-rooms. The QMW Representations workshop was a proper workshop in this sense: well over half the time was allotted to discussions.

As the speakers and audience were lively and well-informed, this made for a valuable break from the routines of daily life. For RQ readers who were not there, I will try to convey a little of the taste of what happened. If the account seems to jump a bit, remember that the representation of three days of intense discussion in a limited amount of typescript is bound to break down somewhere!

Hilary Johnson (QMW) wryly kicked off the workshop by mentioning Aristotle's classification of representation: the Thing represented, the world of the Representation, the Objects (and their Properties) in that World, and the Correspondence between the representation and the thing. What is the nature of that correspondence? It is surely not an actual likeness, even if we say that a flat painting is "like" a flesh-and-blood person. Saying that it is a description or a small-scale model hardly catches our intention either. Craig's maxim, that thought actually consists of the manipulation of representations, seems closer to the mark, but perhaps it was William Palmer in 1976 who really opened up the question by discussing templates, prototypes, holograms, isomorphisms and more. Representations are certainly made to be manipulated: success comes when the manipulated results can be exported back to the real world. A representation is thus both a tool and the output of that tool's use (we need metaphors of machine-tool factories, as we do for software, it seems).

Sara Jones (Hertfordshire) asked the question, "how do you make a representation easy to use?" in her clear, light and direct way. She also risked starting a war, by equating Model with Representation. (Exercise for the Reader: think of a model you know, and list some representations you could use for it.) Sara's paper in fact considered just one criterion (ease of understanding) out of ten that she listed for models: for instance, models should be precise, easy to generate, and provide good coverage. As for ease of

understanding, Paul Dourish asked: can we separate understandability from the power of a notation? The audience was expected to spot the allusion to Wittgenstein here, that language is part of our world, not separate from it. Sara replied that what we say IS what we discuss, and admitted candidly that the challenge "leaves us pretty nonplussed".

Charles Brennan (BT) held the flag for Industry. Ordinary language was vital for communication but poor for modelling: in practice an industrial team had to use the least common denominator, fuzzy not formal. People such as Gilbert Cockton (Sunderland) referred to "lightweight" representations, which apparently come with equally lightweight definitions: nobody supplied one. Chris Roast pointed out the inertia of models once they are made: just the act of building one tends to hold up communication instead of being helpfully dynamic. Gilbert, who turned out to be the most talkative participant, said that models were used early on (in the system life-cycle), and as it turned out much of the workshop focussed on the use of representations in the early and fluid part of any development. Neil Maiden (City) rightly drew the distinction between user/problem and system/solution domains; Lisa Tweedie (Imperial College) rounding off the discussion with the perceptive idea of a hierarchy of representations, shared (among a whole team) at the highest level and descending through not-shared expert representations which are increasingly specialised.

Paul Dourish (Xerox PARC) talked about reflection. The coffee cup on his T-shirt did not contain Java, just as the shirt did not actually contain an American. He spoke with a humorous touch of Gaelic, fluent, persuasive, rich in gesture and combining philosophical astuteness with a distinctive Californian lushness. Computers are "intensional", abstract representations of things in the world; but they are also concrete things, embodied in wires and silicon. This is a lively source of confusion! Software embodies tasks as well as represents them, so it is both static and seductive: we tend to imagine that the world is somehow like our systems. Reflectiveness can be useful though: systems can be made to explain what they are doing, rather than blindly just doing it. Fabio Paterno (CNUCE, Italy) asked what could then be negotiated? Paul replied that we always see through representational sunglasses, no matter how hard we try to open the box and let users "see inside". After all, we are conscious only of very limited areas of our own workings. I pointed out a similarity with expert systems (such as the British Nationality Act) here: it was no use just explaining to the user why their application was being refused! Peter Johnson said that we have to choose what can be explained, and more importantly why: the reflectiveness must have a purpose, which must be to assist users with their goals (such as gaining citizenship). He was fascinated by this question of reflection, and asked how we could know when a notation ran out of expressive power (e.g. your LISP program ran into trouble because you were not using an object-oriented approach).

Neil Maiden (City) spoke about frameworks for evaluating representations in Requirements Engineering. He paced up and down, leonine with his flowing locks, elegantly sardonic in a dark suit and collarless shirt. RE was difficult, error-prone, and expensive (not news to RQ readers). He proposed a framework with four dimensions (layers?), namely Information needs, Media Type, Artefacts used, and Interaction Support. People were currently using everything from formal methods to video recordings, scenarios (as in City's CREWS project) to prototypes, with a trend towards mix'n'match. Neil distinguished between medium/artefact (such as paper) and representation (such as a dataflow diagram). In RE, inconsistency was allowed while models developed; conflicts needed to be identified later. As for how... work was in progress. Discussion centered on whether tools should be a random cupboard-full, or a neat toolkit with a prescribed order of application. Peter Johnson, professing "amnesia for names", relaxed, be-pullovered, spoke with the air of professorially chivvying along the students to higher things. We needed to consider the nature and use of representations, their usefulness in interactive software design, and to look both at current research and new areas. Some were only possible on some media: for instance, interactivity offered "functional affordances" on computers. Discussion returned (heatedly) to the issue of models; Paul Dourish described models as going from the world to abstract categories, whereas representations took such categories and displayed them (i.e. going almost in the opposite direction).

Ann Blandford made the crucial point that the process of describing helps the analyst (such as the requirements engineer) to understand: notation and models all contribute.

Lisa Tweedie, succinct as usual, said that externalising (as with a spreadsheet for a business plan) aids reasoning, and delightfully quoted Bertrand Russell: "A good notation has a subtlety and suggestiveness that at times feels like a live teacher".

Eamonn O'Neill (QMW) presented a simple framework, which posed 3 questions about representations: What is it? What is it for? Who is it for?, illustrating the theme with a Lascaux cave-painting of a large hairy bull impaled on a spear, but still knocking a man senseless. He couldn't answer the questions for the painting, and things weren't much better in RE either. Eamonn said he didn't like the word "co-operative" (did it just mean that users and developers talked to each other?); pale and close-cropped, with a rapid allusive delivery, he lifts his hands to put extra layers of inverted commas around his statements. The representation offered an external shared model, but only partially; viewers might always have slightly different ideas about the underlying model. (Shades of the linguistic philosopher W.V.O. Quine.) Representation thus consisted of a development activity, a communication, a collaboration, and a setting of boundaries (including the transformations needed to other representations, such as from a set of user requirements to a system's functional model). Eamonn stuck his head out with the claim that the representation was "almost another person

in the interaction structure". While the audience might agree that a whiteboard reproduced some functions such as memory, this provoked lively discussion. Tacit knowledge emerged as a theme again: problems arose in a prototyping project when the developers (naughtily) inferred additional (unstated) user tasks, and "it was these that gave usability problems". Discussion concentrated on ways that developers unconsciously take power away from users (who may not wish to stand with a pen at a whiteboard), and conversely on ways in which engineers might push power back to the users, consciously directing attention to possible gaps in models. Neil Maiden suggested that scenarios could help to do this.

Paul Beynon-Davies (Glamorgan) is mild, laid-back, and bespectacled, belying the rapid and comprehensive overview that he gave of the use of scenarios. His main point was perhaps that failures at any scale, hopefully small, are the opportunities for identifying gaps and then iterating designs to cover them. RAD, and indeed model development in general, can be treated as the deliberate search for such gaps. This has a radical ring to industrial ears, where failure is supposed to be avoided and denied at all costs. Obviously, engineers have always tried to improve their designs; what the academics are doing here is to attempt to articulate best practice, to make it more conscious, and hence (one hopes) ultimately to improve it. Discussion examined ambiguities: researchers like Susanne Bødker have used "breakdown" in at least two very different senses, namely the everyday experience of users, thrown by a perceived problem with a system, and the much more specific, Heidegger-type situation when the first-time user of a system finds his attention drawn to some object (such as a button or command) once it fails to behave as expected. Neil Maiden pointed out that industrial techniques such as hazard and fault analysis have long been used to construct trees of contingencies (which depend on other contingencies).

Michael Harrison (York), a plausibly business-like figure in continental-style checked jacket and striped shirt, presented a framework for scenario-based enquiry. "I'm a reductionist", he said, wanting to analyse scenarios objectively. His (interaction) framework consists of agents (usually people, though as Ann Blandford interjected, the Environment can helpfully be treated as an autonomous, possibly random, agent); system states and goals; events sequences, leading to goals; and the implied communication between agents. A scenario could be taken as an "interaction trajectory", i.e. any path through the events (one path might be canonical). There was an ambiguity as most speakers took a scenario to be a generalised pattern (induced from experienced instances), while Harrison seemed to take a scenario as a concrete instance, such as a protocol transcribed from videotape. The common purpose was to use scenarios to identify design weaknesses, so Harrison seemed to be concentrating on snapshots of the use of existing systems such as aircraft cockpits, rather than on possible future systems.

Ann Blandford (Middlesex), iconoclastic in an Ikat jacket, gave a lively and witty presentation of yet another framework for evaluating things, this time of usability. As soon as discussion moved on to HCI, the back row of the workshop started to become markedly noisy. The idea was to represent entities (what the user needs to know) rather than processes, actions, and relationships. Dismissing "the famous white rat of text editors", studied to death by HCI researchers, Blandford had a look instead at a simple sketching tool, comparing it with a graphics editor. Her students had to try to draw a whale using both tools, giving amusing results. Her evaluation model (OSM) was much quicker to learn than famous approaches like PUM or GOMS, neither of which seem to have been taken up by industry to any noticeable extent. But while the students discovered the obvious weaknesses of the sketching tool, they disappointingly didn't use the model to reason about the tool's viability. Further research was called for.

Darryn Lavery (Glasgow) appeared as the mild, centre-parted, long-suffering analyst of miles of video protocols while trying to represent predicted and actual usability problems. The basic issue: "Neilsen says his method found 50% of the problems found (later) by user testing": but that conclusion depends on matching what users found to what the analysts found, and any such matching is highly subjective, given the dire nature of software problem reports. Lavery had a mountain of logs generated from the protocols; what he wanted to do was to condense these (semi-automatically?) into "episodes". Discussion focussed on whether this was feasible. Lincoln Wallen (Oxford) said that calling a pattern "Recovery" simply because the user did something and then undid it, was dangerously misleading: it didn't mean the action sequence was a mistake. For instance, I might switch to outline view and back, gaining an insight into a document's structure. One needed to know the user's purpose to understand his actions! The circularity is just the same as in the debate about how scientific theories are made: do scientists make "pure" observations and induce theories from them as the Positivists used to claim? Wallen said that we just have to assume that everything users do makes sense, even (or perhaps especially) when it looks to be "off task". We want to know why! Peter Johnson said we needed either to be able to verify that things occurred, or to predict they would occur.

Hilary Johnson, summing up the issues of the workshop, said there was a debate between bottom-up construction of representations (à la Wallen) or top-down theory-driven analysis. The workshop did not agree that representations for software were expressive enough for prediction.

I came away from the workshop feeling stimulated to do more with scenarios, and to reflect more on what we do in RE, and why. The choice of representations is wide, and we tend to stick to what we know, but it is good to see that other options are open.

RE-Papers

English or Formal Language?

Recently (starting 12/3/97) there was a discussion in the SRE group concerning the relative merits of English and of formal languages for requirements specifications. Here we try to summarise some of the most interesting points.

Readers who would like the whole discussion in its original but confusing glory can fetch it from *ftp://ftp.mpce.mq.edu.au/pub/jrcase/sre-archive* under "English-vs-formal".

Norbert E. Fuchs of the University of Zurich argued that one could combine the advantages of natural language and formal languages. "In our Attempto project we unambiguously translate specifications in Attempto Controlled English (ACE) into first-order predicate logic, and optionally into Prolog. ACE is a controlled subset of English with a precisely defined grammar and an application-specific vocabulary."

See *http://www.ifi.unizh.ch/staff/fuchs.html* for full details.

Ian Sommerville, Lancaster University, asked two questions about ACE:

Question 1: What do you do about domain-specific terminology e.g. terms like 'trajectory', 'module' etc? Many language problems, in our experience, arise because different people interpret these terms in different ways.

Fuchs replied that this was a general problem of specification languages, not solely of natural language, or specifically of ACE. He wrote: Let's have a look at the Z specification 9.1 from your [Sommerville's] book on Software Engineering:

```
contents: N
capacity: N
contents =<< capacity
```

The meaning of the terms `contents` and `capacity` is not derivable from the specification alone, but needs ulterior knowledge. Also, further defining the terms will not help since some unexplained base terms would always remain.

Specifications are meant to be composed and read by people who understand the background of the pertinent application domain. These people use a terminology that may or may not be standardised. This standardisation process is outside of any specification language or tool, though once there is a standard there can be tools to enforce it. Examples of such tools are found in the field of controlled languages used for technical documentation.

Question 2: How do you convince people to write in ACE or similar languages?

Fuchs replied that Sommerville had alluded to AECMA Simplified English, a controlled language used in industry by Boeing, Caterpillar, Scania, Siemens, and taught to technical writers. Caterpillar claim that after a three-day course and six weeks of practical experience, technical writers can use the controlled language competently. We are highly confident that ACE can be learned in a short time, as it is much simpler that AECMA.

Andy Champ replied to Fuchs: This brings up the point I find problematical in formal methods. If I write a specification in English it may be vague and hard to match, but understandings of it may differ. Specifications in Z may be unambiguous but you have a much worse mismatch between understanding of the problem and its expression. The result is that you have replaced two small error sources - the difference between the customer's view of the requirement to the requirement document being the first, and the difference between the suppliers view and the document being the second - with one enormous error source...

Nancy Mead replied:

In using any formal language, there will be a problem of customer understanding. I was on a large project that used an Ada-based PDL for design. We had almost completed the design reviews when a mock-up of the displays was presented to the users, who realized that they did not have the display information that they desired. The data that they were looking for was not even being calculated, but they did not realize that in all the earlier reviews because they did not comprehend the design! We had at most 2-3 people in the customer set (of 60 or so) that could read and intelligently comment on the design.

Haim Kilov (haim_kilov@ml.com) replied to Andy Champ as follows:

I disagree. The "small" error sources are not small at all: a warm and fuzzy feeling (e.g. from a slide!) is not a definition, and the delivered system (if it will ever be delivered) will probably differ quite substantially from what the customer had in mind. You cannot rely on defaults, and "meaningful names" do not help either, as these meanings are quite different for different stakeholders. Having said that, I do NOT advocate showing the customer a Z specification for obvious reasons. It is possible to use a semi-formal (graphical) notation for relationships, combined with application-specific invariants and pre- and post-conditions for operations (all in pseudo-English).

Les Munday wrote: "A requirements document is written for a minimum of two groups of readers: the customer and the supplier. Write it in a language that will satisfy all readers. In this case, both English and Z..." (He did not say what to do when interpretations of the two differed.)

Norbert E. Fuchs chipped in: "Or in Attempto Controlled English (ACE) and in a semantically equivalent

representation in first-order predicate logic automatically derived from the ACE text..."

Soren Lauesen (slauesen@swin.edu.au In Denmark: slauesen@cbs.dk) thanked Andy Champ, expressing happiness with his explanation. "I have felt like you for a long time, but not been able to express it properly. And I certainly agree with you after studying many styles of requirements used in practice."

Dick Botting (*http://www.csci.csusb.edu/dick/signature.html*) thought things were getting far too academic.

"The pragmatic answer is that you would be best to include both pieces of the user's natural language and also precise mathematical formulae. Also the connections between the natural and the precise statements. Plus Jacksonian Designations of what the terms mean. But there is more... Also screen shots, storyboards,... Also charts, diagrams, and tables. And perhaps a video of a prototype, A recording of a key meeting with the users, ... You control variety by adding variety."

Geoff Mullery (gmul@sml.win-uk.net) sided with Haim Kilov, along the following lines: "Look at other industries and you see the example we should learn from. When planning a building we have an aesthetic (artist's) impression as one form of representation, a draughtsman's drawing as another, an engineer's (load/stress, etc.) calculations as another and an accountant's calculations as yet another.

Put the artist's impression alone in front of the builder and you may get a building which falls down too readily, or in which the proportions, though looking very pleasant, prevent practical use of the building..."

If the building trade can work out that they need multiple representations, with a defined scope of responsibility and relationship between the representations, why can not the computer industry do the same?

The argument here should not be whether we use English or a formalism, but how we use English and a formalism - and what other representations are also needed.

Dick Botting presented a little anecdote on "The customer is always right".

In 1974 (I think it was) I taught a CS1 class which also served the mathematics department majors. Two of the exercises were set and graded by mathematicians. They were to be coded in FORTRAN. The students were given a written specification of the algorithm complete with the formula that they were to use.

On the first exercise all the best students failed because they optimized the formulae that they were given to give a faster algorithm. Howls of protest! The real shock and learning experience occurred when I and the other CS department faculty told them that the mathematicians were right -- because they were the client/customer.

Andrew Gabb (Andrew.Gabb@dsto.defence.gov.au) replied to Dick Botting's anecdote as follows:

I guess it's another example of "if what you think they need is different from what they think they want, you (the developer) have a serious problem, and giving them what they need, OR what they want, usually won't solve it".

Which reminds me of another anecdote (see below under Martin Feather).

Now I don't know what form the requirements were in, formal or natural language or algorithm. They could have been written in a formal language, and been proved correct, even though they were incomplete. It also appears to me that finding the missing constraint would be more likely by reviewing natural language than in a formal specification. It also looks like a damn good advertisement for having test cases developed and validated by the customer.

However, in my mind this is a typical software requirements flaw and, as both a customer and a developer, I would

RE-Bites...

Here is a lesson from the requirements analysis for an aircraft, the name of which is lost in the mists of time.

For safety-critical applications, it is usual practice to put safety interlocks (both hardware and software) into the system, to prevent the wrong things happening at the wrong time. An interesting example is the use of reverse thrust on an aircraft. Clearly, one does not want this to come on in mid-air, but of course it is pretty essential for landing. An appropriate system requirement would be something along the lines of "Reverse thrust shall be disabled whenever the aircraft is not on the ground"

For the software, the real world state "aircraft on the ground" must be detected somehow. A number of possible surrogate states can be used, such as "weight on wheels" and "wheels turning", which can be detected with sensors on each wheel.

In one incident, on a wet runway, with a strong crosswind, both the brakes and the reverse thrust failed to engage, on the grounds that the plane was not on the ground. The plane was aquaplaning on the wet runway; hence the wheels were not turning. In addition, the pilot was banking the aircraft to some extent, to counteract the crosswind. There was no weight on one set of wheels, and neither set was turning.

The lesson? Requirements engineers need to ask the right questions. In many cases it is not enough just to ask whether the surrogate state is an adequate way of determining the real world state. Ask whether "weight on wheels" is suitable to determine if the aircraft is on the ground, and you'll generally get a positive response.

In this case, the right questions should have included an examination of the conditions under which the mapping between surrogate and real world states *does not* hold. For example, a pilot might tell you that it is normal practice to land on one wheel in a crosswind, but only if you ask the right question.

expect a good developer to pick it up, alert the customer, and fix it, particularly because the requirements were so simple to understand. And this applies even where the customer had supplied an algorithm. Simply saying "I do what I'm told" is not an excuse, in my opinion (although it *is* a defence), and won't help anyone in the long run. On the other hand, changing the requirements without agreement by the customer is also inexcusable, and not defendable in most cases.

Martin Feather (feather@jpl.nasa.gov) wrote: "Andrew Gabb posted an anecdotal problem, which I think lends itself nicely to a notational treatment:"

"As in many places in the world, the final year high school results here are used as one of the criteria for entry into university. The results of 5 subjects are combined in a simple formula, weighted towards the better results. ..."

Write this in some sort of pseudo-functional-notation:

```
final_result(stu) =
    weighted(resultsof(5subject&resultsof(stu)))
```

Then combine that with a definition of

"If a student repeated final year, it was decided to use the top 5 subjects from both years, which seems fair."

You've now got TWO sets of results for student stu, those in year yr1, and those in year yr2. In trying to write down the definition, we discover 3 alternatives. For the more formally inclined, defining each of the functions, and then attempting (perhaps failing) to prove that they yield the same answer, is a good way to uncover problematic aspects.

CORE - Blimey!

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

Darn Those Threads

In previous contributions I have mentioned the problems that there are many views of a requirement to be considered, that they inter-relate, involve varying cultures and terminologies, are frequently distributed among several teams and locations and require the storage and use of very much more information than many people are prepared to admit.

There are aspects of the data handling problem for which there are available solution technologies, but for which only inadequate solutions appear to exist. The problem requires efficient handling of diverse, unstructured documentation which forms the source for most requirement specification exercises and which is mirrored in much of the information used to support or even form the main product of the requirement specification process.

Most of this material is, for perfectly understandable historical reasons, in the form of word-processed documents or even older archived paper or microfiche documents. Almost all of this material has been produced on the basis of a linear model of how most effectively to present the necessary information to an audience. This means that users of the document almost invariably have to perform a continuous filtering process on what and where they read to get the information they need without being confused by (to them) extraneous information present for other purposes (other users of the document).

The introduction of hypertext showed a way of reducing the problem of linearity in traditional documentation, but it was limited in its usefulness because it would not easily cater for the fact that the desired starting point and topic thread through the document would often be strongly dependent on the purpose of the current document user.

The need for assistance in this area was evident from the start and a partial solution was introduced into the document reading service via bookmarks. The user of a hypertext document reader was permitted to mark specific points for quick access so that they could be accessed by selection from a pop-up or drop-down list. They were called bookmarks to emphasise the analogy with traditional paper book ownership. There were a number of problems remaining.

First, hypertext documents have numerous users (in the sense of people who have access to them for reading purposes). This means that the analogy is more properly with library book users than with personal book ownership, so bookmarks need association, not with the book or with the document reader service, but with the <book, reader service, user> triple.

Second, documents inter-relate directly via explicit references and implicitly via shared concepts and data - and those documents are not necessarily co-resident. This means that hypertext had to (and of course now does) allow for references between documents, possibly distributed among different physical locations.

Third, though a user might search for, find and mark specific locations of interest in a document, what they are frequently doing is finding a thread through the document(s) rather than looking for a specific set of individual locations. Remember, the document was often originally written in its linear form because that was the author's best shot at predicting the most efficient linear thread through the contents. A specific user may be using the document for another purpose - rendering the author's predicted thread (and explicit hypertext links) wholly or partly irrelevant. So bookmarks are inadequate - they require supplementary facilities for thread traversal.

Fourth, when a user finds a thread through a document, even if it can be re-created via a series of explicit bookmarks and hypertext links, the next time s/he comes to read the document and even start to traverse the same marked thread,

it is likely to be for a slightly different purpose. This means that users need to mark not just a thread, but a thread tree - just like that set up by the original author's hypertext links. The bookmark triple has become a quadruple: <book, reader service, user, purpose>.

Fifth, though the document's author sets up the hypertext linkages s/he can foresee, s/he is not omniscient, so there are likely to be linkages which other users need to add - and these may be specific to a particular purpose. This means that the document user needs the ability to set up other internal and external linkages in a manner that does not interfere with other users' use of the document.

Sixth, many documents exist in a form that does not readily allow for hypertext use. They exist as paper or hypertext archives that are frequently very opaque for many interested potential users, making the tracing of reference threads difficult, tedious and imprecise. Some improvement is possible by using paper copies to generate scanned images on computer and then manipulating and inter-referencing those.

Some elements of any document - particularly scanned documents - are in a form which make hypertext linking a little more difficult. In particular there are vector or bitmap images which must either be transformed (e.g. using OCR tools) or transparently treated (e.g. by using invisible buttons) to form hot spots for linking or other operations.

Finally (for this discussion) for some things it is most effective to use animation techniques to specify or illustrate a requirement. Possible examples are user interface styles, protocol definition animation and parametric illustration of mathematical behaviour. For these cases it may be relevant for some users to specify a thread which links to some part of an animated sequence, rather than the whole.

Most of the above problems are at least partially addressed now by facilities provided as a result of the explosion of interest in and use of the World Wide Web and co-operative working tools like Lotus Notes, but some are not yet fully supported for the very good reason that they lead to strong possibilities for chaotic interwoven networks of mutually interfering hypertext links.

A few extra facilities are needed to avoid the onset of chaos and they are not (to my knowledge) provided by anything on the Internet or in tools like Lotus Notes - though, as I implied in my introduction, the technology exists to permit their provision.

One key feature is the need for display of hypertext links to be based on a definition of the user's current purpose, so that the user need see only the links appropriate to that purpose. It should then be possible for the user to select one or more from any published (see below) purposes - and to re-select at any time, with the result that the links displayed would be those relevant to the currently selected purpose.

A necessary additional feature, given that several purposes may be selected, is to resolve which of multiple links to follow when two purposes link from the same point to different destinations. This may simply require a pop-up selection list, asking which purpose link to follow.

Given that features such as those described can be provided it is also necessary to permit users to define threads based on a purpose, itself defined by them. In other words, a hypertext-linked document set must no longer be considered as a static set of linkages defined by its originator. Even though they contain animated sequences, linkages are currently still almost entirely static - bookmarks being a crude exception.

This notion of a linkage thread is very important. A problem with current on-line documentation schemes is that they allow the user to step backward through the thread via which they reached the current point, but do not permit quick steps forward or back to any selected intermediate points. This is what marks on-line document readers as distinct from and inferior to the way people read and jump between pages of a paper document.

When a thread is defined and stored it is possible for a user to be provided with a facility to switch contexts quickly to exactly the location s/he now wishes to view, so s/he can read the thread, either for the exact purpose for which it was originally created or, later, to re-visit specific for confirmation of understanding or for some related purpose.

For most purposes, but particularly for specification, one user's purpose may be shared by other users, so there is benefit in allowing for users to publish purpose definitions for use by. It is also important for a purpose to be capable of being withheld from publication (e.g. to satisfy commercial or military security/access demands).

Given that multiple users may make use of a published purpose definition it follows that a situation may arise where a specific purpose may have several different start-points in threading through available documents - and any given user may choose to exclude selected starting points or selected alternative intermediate links from consideration. Hence a user interested in a specific purpose definition must be able to specify exclusions - perhaps by refinement of the purpose definition.

The availability of purpose definitions does not of itself mean that they will be of use to a user who has a need for that type of information. A definition of a purpose does not necessarily make itself stand out from other purpose definitions available when a user with a shared interest uses the document reader. Also, a purpose definition may apply across documents, disciplines and projects, so it will become necessary to separate, but inter-relate purposes from the documents and people who may use the document reader.

This diversity and possible distribution of associations leads to the need for users to be able to establish filters on what they see based on location, access rights, date interval, interests and purposes. Then there is a need for a search engine that enables them to find purpose definitions and

document sets relevant to them within the bounds of the filters they have defined.

This latter requirement goes beyond the notion of document linkages as I have described it here. It strays into other areas: Enabling Network Systems and database support systems, which I may deal with in later contributions.

The key point of all the above is that system development in general and requirements specification in particular necessarily makes use of a wide variety of information sources, most of which can be put into machine processable form and used more efficiently than is currently the case.

There is still a need for transforming some of this into a more formalised format - going into structured and/or mathematically formal notations, but with the techniques I have suggested here the source documents can be made less opaque, can be read and re-read much more efficiently than is currently possible and can be effectively and efficiently traced to and from the formalisms derived from them.

All of this can be done without the user having to be explicitly aware of the fact that, because of the potentially vast document base in use there are database facilities needed to provide the inter-relationship support which enables this efficiency. With current facilities which I have seen, means of tracing relevant documents is crude, the efficiency with which they can be processed is poor and the discovery and pursuit of inter-relationships between them is diabolically difficult. As the project size and distribution increases the size of the problem grows exponentially. With the type of tool support I suggest here the problems can be much reduced.

Geoff Mullery
geoff_mullery@dial.pipex.com

RE-Publications

Reviews of recently published books about requirements engineering.

Book Review: Bruno Latour, "Aramis, The Love of Technology", Harvard University Press, 1996. ISBN: 0674043235.

Reviewed by Richard Veryard (rxv@veryard.com)

Phenomenon 1: High reuse of case study material

It is often much easier and quicker to illustrate an argument using a well-known case study, than to present a new case study in sufficient detail to support the argument.

As a result, we see the same case studies used by many writers to make many different (and often contradictory) points. An extreme example of this is the Challenger disaster, which has been analysed in countless different ways, the failure of the O-Ring being traced to all kinds of flaws: in the engineering process, in the management structure, in the political environment, or in the unconscious psychology of the organization. Other case studies are not so much analysed as paid homage: mention the London Ambulance Service, for example, and the reader is expected to nod wisely and adopt a fixed critical stance.

It is therefore a delight to come across the book under review, which documents and analyses a large yet little-known R&D project in enormous detail. The book is witty, insightful, and is written in the style of a detective novel, following the progress of two academic socio-technologists as they try to conduct a postmortem on the Aramis project. Who killed Aramis, and why?

Aramis was an attempt to construct a new mode of public transport in Paris, which would combine the privacy and directness of the private car with the efficiency and speed of the metro. The project lasted from 1970 to 1987, absorbed many million francs, produced several impressive prototypes, gained political support from successive administrations, and was suddenly and summarily cancelled. Our two fictional academics are trying to explain on the one hand why the project lasted as long as it did, and on the other hand why it didn't complete. They interview the participants, wade through the files, and come up with a succession of hypotheses, each one modified or undermined by the evidence uncovered in the next chapter. Latour alternates the fictional discussions of the academics with extracts from the real-life material gleaned in his field research, tracing the technical complications, politically motivated half-truths and ever-shifting and competing visions with clarity, intelligence and rigor.

Many writers will be able to find illustrations of their pet theories in the book, although I suspect it will be difficult to come up with good explanations of the phenomena that are not already in the book. To give you a taste of the book, let me briefly mention a few of these phenomena.

Phenomenon 2: 20-20 hindsight

When interviewed retrospectively, many of the participants claim to be able to identify some fatal flaw in Aramis that caused its death, either in the technological concept itself, or in the socio-political environment in which Aramis was to have been implemented. However none of the participants expressed any reservations at the time, except perhaps in the standard small-print phrases that are used to evade liability. Even when the prototypes didn't actually work, or didn't demonstrate what they were supposed to demonstrate, these failures were always explained away in the interest of keeping the funds coming. All the participants apparently colluded in remaining optimistic.

The problem with the 'fatal flaw' type of explanation is that it explains too much. Similar 'fatal flaws' can be often found in projects that did run to completion. In this particular case, technology analysts are especially fortunate; Aramis had a

sister project called VAL, which involved many of the same participants at the same time, and shared many of the allegedly 'fatal flaws' to which the demise of Aramis is ascribed. VAL was successfully implemented in Lille, and has since been exported to several US cities.

Furthermore, the 'fatal flaw' type of explanation cannot explain why Aramis survived so long. It is certainly not the case that some flaw was initially hidden, and that it was the discovery and proof of such a flaw that triggered the cancellation of the project. Indeed, most of the flaws were known in 1984, when Aramis was given a political and financial commitment to go ahead.

Phenomenon 3: Minor components that take over the show

In the conventional view of technological development, the designer establishes a mapping between a set of demands (usually known as The Requirements) and a set of components or services that collectively satisfy the demands (usually known as The Solution). As is often the case, Aramis turns out to require several components that haven't been invented yet, although engineers are confident that they could develop leading edge technological solutions, given time and budget.

But the design logic of component-based development isn't compatible with the research logic of technology breakthrough. The components that don't yet exist create 'holes' or 'attractors' in the design structure, which have a distorting effect on the project as a whole.

For example, Aramis demands the invention of a new type of electric motor. The variable-reluctance motor is invented, and then this invention starts to be cited as one of the beneficial spin-offs of the project as a whole, although when pressed nobody can think of any other use for this motor. The design of Aramis now revolves around this motor.

Phenomenon 4: Killing by simplification

The development of Aramis was marked by compromise. The initial vision, apparently simple, turned out to contain several conceptually distinct parts. On the one hand, the simultaneous achievement of all these parts was thought to be politically impossible and/or prohibitively expensive. On the other hand, dropping any of these parts was regarded as an unfortunate watering down of the Aramis vision, to the point where the very identity of Aramis was compromised.

In other words, some engineers argued that THIS requirement makes Aramis unimplementable, while other engineers argued that without THIS requirement, there is really no point in continuing with Aramis at all.

Phenomenon 5: Autonomous technology

Like many complex technological projects, Aramis seems to have taken on a life of its own. There is too much at stake for any of the participants to make decisions that conform to textbook rationality. Perhaps this is why Aramis has a momentum, and takes as many years to stop as to start. In the fictional parts of the book, Latour reflects this by giving Aramis a voice, with tongue-in-cheek references to Frankenstein and other monsters. Particularly in these sections, the book hovers on the boundary between sociological analysis and literature.

The book is a marvellous and rich narrative, clear and thought provoking. Every engineer should recognize with wry pleasure the excitements and disappointments, the debates and distractions and diversions and difficulties of technological development. Strongly recommended.

RE-Calls

Recent Calls for Papers

Human-Computer Interaction A Journal of Theoretical, Empirical and Methodological Issues of User Science and of System Design

Special Issue on Representations in Interactive Systems Development

Various representations serve diverse purposes throughout interactive systems development. Representations may be used to support processes such as analysis of users' work and environment, envisioning and implementing new work situations and technology and evaluation of usability, work performance and quality.

In playing a part in such processes, a representation may provide, for example, an embodiment or model of the object of a particular development activity. It may provide an historical record of the development process. It may serve as a channel of communication within or between groups of stakeholders in the development process.

Representations may take many different forms. Dimensions along which types of representations may be considered include formal to informal, general to concrete, exclusive to participative, tool-based to ad hoc, graphical to propositional.

Questions of interest include: which processes and activities within systems development may benefit from or require particular representations? Which form of representation is most appropriate to meet which needs? What benefits has one form of representation over another, in what circumstances? Which properties of representations support particular processes, roles and activities? Who needs to be able to understand and work with a given representation?

What tools and environment must be provided to support construction and use of a particular representation?

Papers should present the authors' work and ideas on representations and their uses within systems development, addressing the above or related questions.

Five clear copies of a submission should be sent to Patricia Sheehan, HCI Administrative Editor, Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94303, USA. A note should be included that the submission is for the special issue. For further information see Information for Authors in the journal or at http://www.parc.xerox.com/HCI. Deadline for submissions is 19 September 1997. Guest editors for this special issue are Peter Johnson, Hilary Johnson and Eamonn O'Neill of Queen Mary and Westfield College, University of London, UK.

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://research.ivv.nasa.gov/~steve/resg/

Web Pages

The BCS RESG home page can be found at:
http://porta.cs.york.ac.uk/bcs/resg/

Back issues of Requirenautics Quarterly:
http://research.ivv.nasa.gov/~steve/resg/

The DTI/EPSRC Safety Critical Systems R&D Programme:
http://www.augusta.co.uk/safety/

The University of Calgary's Concept mapping tools page:
http://ksi.cpsc.ucalgary.ca/~lukose/cseic/

Books

Ian Sommerville and Pete Sawyer, Requirements Engineering: A Good Practice Guide. John Wiley & Sons Ltd, 1997. ISBN 0 471 966916

Richard H. Thayer, and M. Dorfman (Eds.) "Software Requirements Engineering, 2nd Edition". IEEE Computer Society, 1997. ISBN: 0818677384

Jeff Poulin, "Measuring Software Reuse", Prentice Hall. ISBN: 0201634139

Carma McClure "Software Reuse Techniques: Adding Reuse to the Systems Development Process", Prentice Hall. ISBN: 0136610005

Mailing lists

Requirements Engineering Newsletter

A Requirements Engineering Newsletter is published as an educational service by Prof. Anthony Finkelstein at City University. If you wish to contribute send your material to the moderator at: requirements@cs.city.ac.uk

Subscription (or removal) requests should be sent to: requirements-request@cs.city.ac.uk Send an email containing

subscribe <address>

or

unsubscribe <address>

The Requirements Engineering Newsletter is archived at:

http://web.cs.city.ac.uk/homes/acwf/rehome.html
or *ftp://ftp.cs.city.ac.uk/pub/requirements/*

Software Requirements Engineering Mailing List

To subscribe to the Software Requirements Engineering (SRE) mailing list, e-mail listproc@jrcase.mq.edu.au, with the only line in the body of the message:
subscribe SRE your-first-name your-second-name

Articles to the SRE mailing list should be sent to SRE@jrcase.mq.edu.au.

Tools

ObjecTime allows reactive type requirements to be modeled, validated and derived using the principle of executable models. The same executable requirements model is then passed to the design team to further refine into an architectural and then design model implementation. The model through automatic code generation then becomes the actual target implementation linked directly back to the requirements. See: *http://www.objectime.com*

Special feature this issue: Concept mapping tools…

Folks at the University of Hamburg have developed a multi-form editor for Conceptual Graphs, that allows manipulation in both linear (i.e. textual) and graphical form. It is based on a VisualWorks environment It is available at:
http://www.informatik.uni-hamburg.de/NATS/papers/moeller/cg.html

Mobal offers graphical presentation of: facts (concept instances); predicates (concepts w or w/o typed arguments); and rules (inferential relationships between concepts). Mobal was designed mainly for theory building and maintenance in a first-order logic formalism. It runs on. SunOS/Solaris, and is available at:
http://nathan.gmd.de/projects/ml/mobal/mobal.html

Smart Ideas is a commercial concept mapping tool produced by a Calgary company called Smart Technologies. It is available at: *http://www.smarttech.com/smartideas.htm*

RE-Actions

Minutes of the Third Annual General Meeting of the BCS Requirements Engineering Specialist Group

Location: Room 418, Department of Computing, Imperial College, London
Date: 11th June 1997
Time: 14:00 Local Time

1. Apologies for absence.

Steve Easterbrook (Newsletter Editor)

3. Minutes of previous meeting

The minutes of the previous AGM were approved.

4. Chairman's report (Bashar Nuseibeh)

Bashar Nuseibeh, the chairman of the RESG, presented his annual Chairman's report. A summary of this report is given below.

Membership (Membership Secretary : Sara Jones)

The current membership profile of the RESG is as follows:

Individual Members	241
of which students	42
Corporate Members	15
of which academic	12
New members since June '96	73

Events

The RESG has held a number of successful events during the last year:

10 July 96 (London): Why RAD is BAD! (Room 101)
25th October 96 (York): Safety-critical issues in RE (talks/panel)
16-17th December 96 (London): Formal methods (workshop)
19th February 97 (London): Clapping with one hand (Room 101)
26th March 97 (London): Business specification patterns (tutorial)
11th June 9 (London): Requirements for OTS Systems (talks/panel)

In addition to these events the RESG co-sponsored two events (an IEE residential course on SRE in April 1997, and a UNICOM Workshop) as well as producing four issues of the 'Requirenautics Quarterly' (RQ) newsletter.

Events planned for the future include:

10th September 97 (Manchester): Requirements Acquisition and Modelling: The impact of reuse (talk/panel)
30th September 97 (London): RE-Day (everything!)
26th November 97 (London): Quality Certification for the RE Process (talks/panel)
4th February 98 (York): Industrial Experiences in RE

Bashar concluded his presentation by encouraging members of the RESG to become involved in organising events.

5. Treasurer's report (Neil Maiden)

Neil Maiden, the Treasurer of the RESG, gave an overview of the group's financial situation:

Income:

Membership		
members	470.00	
non BCS members	219.60	
Corporate	400.00	
Academic	240.00	
Membership subtotal		*1329.60*
BCS Subvention	860.00	
Gen. Meetings	50.00	
Events		
Robertson Tutorial	433.33	
RESG/FACS	747.77	
Kilov Tutorial	1164.42	
Uncollected	223.00	
Total income:	**4808.12**	

Expenditure:

Gen Meetings	75.02
Newsletter	737.80
VAT Expenditure	863.97
RESG/FACS event	373.88
Total expenditure:	**2050.67**
Operating profit:	**£2757.67**

Savings Account approx £9700, subject to VAT payments

Neil noted that the healthy balance of the RESG Savings Account enabled the group to underwrite large events such as the forthcoming 'RE-Day'. He also predicted that the RESG would undertake even larger events in the future.

6. Election of Executive Committee

Bashar Nuseibeh introduced the current RESG committee and proposed three additional members:

Current Committee:

Steve Easterbrook (Newsletter Editor)
Olly Gotel (Industrial Liaison Officer)
Mike Bearne (Secretary)
Sara Jones (Membership Secretary)
Neil Maiden (Treasurer)
Bashar Nuseibeh (Chairman)
Andrew Vickers (Publicity Officer)

Proposed Additional Members:

Carol Britton (Events Officer)
Ian Alexander (Associate Editor - Events)
George Spanoudakis (Associate Editor - Features/Book Reviews)

The above committee was nominated by Simon Smith of the University of Durham, and seconded by Shailey Minocha from City University. The committee was duly elected.

7. A.O.B.

As there was no other busy to discuss, Bashar Nuseibeh declared the third AGM of the RESG closed.

RE-Actors

The committee of RESG

Chair: Dr. Bashar Nuseibeh, Department of Computing, Imperial College, 180 Queen's Gate, London SW7 2BZ. 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. N.A.M.Maiden@city.ac.uk, Tel: 0171-477-8412, Fax: 0171-477-8859

Secretary: Michael Bearne, Philips Research Labs, Cross Oak Lane, Redhill, Surrey RH1 5HA. bearne@prl.research.philips.com, Tel: +44 (0)1293 815428, Fax: +44 (0)1293 815500

Membership Secretary: Dr. Sara Jones, School of Information Sciences, University of Hertfordshire, Hatfield, AL10 9AB. S.Jones@herts.ac.uk, Tel: 01707 284370, Fax: 01707 284303

Industrial Liaison Officer: Dr. Orlena Gotel, Systems and Software Engineering Centre, Defence and Evaluation Research Agency, St. Andrews Road, Malvern, Worcestershire, WR14 3PS. olly@hydra.dra.hmg.gb, Tel: 01684 894674

Publicity Officer: Dr. Andrew Vickers, Department of Computer Science, University of York, Heslington, York Y01 5DD, andyv@minster.york.ac.uk, Tel: 01904 434727, Fax: 01904 432708.

Events Officer: Carol Britton, Department of Computer Science, University of Hertfordshire, College Lane, Hatfield, UK. AL10 9AB, c.britton@herts.ac.uk, Tel: 01707 284354, Fax: 01707 284303

Newsletter Editor: Dr. Steve Easterbrook, NASA/WVU Software IV&V Facility, 100 University Drive, Fairmont, WV 26554, USA. steve@atlantis.ivv.nasa.gov, Tel: +1 (304) 367-8352, Fax: +1 (304) 367-8211

Newsletter Associate Editor (Features and Book Reviews): Dr George Spanoudakis, City University, Department of Computer Science, Northampton Square, London EC1V OHB. gespan@cs.city.ac.uk, Tel: 0171 477 8000 ext. 3701, Fax: 0171 477 8587.

Newsletter Associate Editor (Features and Event Reviews): Ian Alexander, 2 Dornton Road, London SW12 9ND. iany@easynet.co.uk. Tel: 0181 675 6915

RE-Creations

How to contribute to RQ

Please send contributions to Steve Easterbrook (steve@atlantis.ivv.nasa.gov) before the publication deadline. Submissions must be electronic copy, preferably plain ASCII text. A list of the kinds of contributions we welcome can be found in the January 1996 newsletter, or on the web at:

http://research.ivv.nasa.gov/~steve/resg/rq5/ReCreations5.html

	Copy deadline
Issue 12 (October)	12th September 1997
Issue 13 (January)	19th December 1997
Issue 14 (April)	27th March 1998