Requirenautics Quarterly

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

 Issue 10 (April 1997)

RE-Locations

RE-Soundings

Editorial

Well, spring has arrived (although here in West Virginia, it's still snowing…), and so it must be time for another *RQ*. This issue features reviews of the workshop on formal methods and requirements engineering, held by RESG just before Christmas, and the IEEE Requirements Engineering symposium held in Annapolis just after Chrismas. Interestingly, both these meetings seemed to be putting across a strong message that it's finally springtime in formal methods land. Do I detect the rumblings of a bandwagon?

We also have two book reviews in this issue - the past year seems to have produced a rich crop of requirements engineering books, and they keep coming. More reviews still to come - watch this space! Also in this issue, a paper describing the results of a study into package selection for requirements management tools. Does that sound slightly recursive? Well, read the article…

Articles for the next issue are due by 27th June 1997.

Steve Easterbrook,
NASA IV&V Facility, Fairmont WV

Chairman's Message

It's been a busy few months for the RESG, with at least one RE event every month since December. The 2-day "Christmas Workshop" organised with the FACS SG was a great success, and I'd like to thank Sara Jones, David Till and all those who helped organise the event for their time and effort. A summary of the workshop proceedings is in this issue.

In January, many members of the UK RE community met in Annapolis, USA, to participate in the 3rd International Symposium on Requirements Engineering (RE '97). The symposium saw the highest number of attendees of any in the series of symposia so far (with just under 200 delegates). My personal highlight was Daniel Jackson's engaging mini-tutorial on model-checking: a topic I knew very little about, and from which I came out informed and interested. Having four (good) keynote talks was probably excessive, but I don't know which one of the four I would have missed: Hall, Potts, Harel or Rushby.

In February, the RESG organised the second in its series of "Topics Banished to Room 101" with Vic Stenning suggesting that Requirements Engineering was one such

topic! He argued that RE was an activity that could not be undertaken in isolation, which despite being an apparently uncontroversial assertion, provoked much discussion both during and after Vic's talk.

In March, Haim Kilov (formerly at IBM and now at Merril Lynch), flew in especially to London from New York (for a whirlwind 48 hours visit), to give a half-day seminar on "Business Process Patterns". Haim provided an entertaining and thought-provoking account of good and bad practices for eliciting, specifying and matching business patterns.

Finally, this month the RESG co-sponsored an IEE Residential Course on Systems Requirements Engineering in Cambridge. This 4-day event, aimed at practitioners, addressed a variety of RE-related topics: from analysis and modelling to documentation, traceability and tool support. There are plans for another similar course next year.

I don't know about you, but I need a break. Our next meeting is scheduled for 11th June - a half-day seminar/panel on "Requirements for Off-the-Shelf Software". It will be preceded by the RESG Annual General Meeting, which I encourage you to attend. All the members of the current RESG committee would like to continue in their current posts and ask for your support.

However, we also welcome new energetic volunteers who would like to serve on the committee. In particular, we are looking for an Events Officer and an Associate Editor for this newsletter. If you are interested, have a look at the brief "job descriptions" of these posts inside this issue of RQ and contact Mike Bearne, the RESG Secretary, or myself for more information.

For those who celebrated Easter, I hope you had a happy one; for those who celebrated Purim, Hag Sameach; and for those who are celebrating Eid, Eid Mubarak. Hmm... the last sentence was too implementation-oriented...what I meant to say was: Season's Greetings!

Bashar Nuseibeh,
Imperial College, London

Membership Matters

Many thanks to those who renewed their memberships promptly in response to my January request. Also a small reminder to those that didn't that this is the last newsletter you will receive if I hear nothing from you before the next issue is sent out in summer!

If, like me, you wish that this renewal business didn't crop up with such alarming frequency, you may be interested in a proposal discussed at our committee meeting in March. It was suggested that membership of the Group should, starting in 1998, cover a period of two years rather than one. Of course, it would then cost twice as much, but would last twice as long and mean one less thing in your in-tray in January 1999. I know this means thinking some way ahead (!), but if you have any thoughts on this, or any other matter in relation to your RESG membership, please do not hesitate to get in touch.

Sara Jones
RESG Membership Secretary

RE-Treats

Forthcoming events organised by the group

Requirements for Off-the-Shelf Systems and Software

Wednesday, June 11, 1997

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

Time: 2:00pm to 5:00pm

Cost: free to members, £5 to others

Abstract: Procurement is the process of purchasing complex software systems from suppliers. These systems might be commercial off-the-shelf systems or tailor-made for a customer. An increasing number of organisations are procuring software-intensive systems. For example the UK government has awarded contracts for over £8bn worth of software-intensive weapons systems to fit platforms including 25 patrol aircraft and 5 hunter-killer submarines. However, successful procurement of software systems is difficult. The London Ambulance Service fiasco in 1992 (Dowell & Finkelstein 1996) is one of the more well known examples of system failure due to inadequate procurement and requirements definition.

Procurement of complex software systems has had a fundamental impact on the processes by which such systems are developed. The requirements engineering process is no exception. This seminar will explore the impact of system procurement on practices, processes, methods and software tools for requirements engineering. Speakers from both industry and academia will explore current problems and propose new solutions to overcome these problems. The seminar will take a wide view of procurement, from procurement of large defence systems to purchasing small off-the-shelf systems.

Please note: *RESG's policy is that distribution of speakers' slides will be free of charge to those who attend the meeting and specifically request a copy. Those people who do not attend the meeting and request a copy of the slides will be asked to pay £5 to cover photocopying and mailing costs.*

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

Organised by the BCS RESG, Co-sponsored by BCS, RENOIR

Location: *To be Confirmed...*

Cost: Free.

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 include:

- A keynote address by Suzanne Robertson (Atlantic Systems Guild)
- Seminars by acknowledged experts
- Workshops where you can try examples of the latest technology
- Tutorials
- Publications and tools fair, and much much more!

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

Contact Information About:

1. Attending the meeting: Dr Andy Vickers, Department of Computer Science, York University, Heslington, York Y01 5DD. Fax: 01904 432788 (andyv@minster.york.ac.uk)
2. Participating in the meeting: Dr Bashar Nuseibeh, Department of Computing, Imperial College, 180 Queens Gate, London SW7 2BZ. Fax: 0171 5818024 (ban@doc.ic.ac.uk)
3. Exhibiting tools: Dr Neil Maiden, Centre for HCI Design, City University, Northampton Square, London EC1V OHB. Fax: 0171 4778859 (N.A.M.Maiden@city.ac.uk)

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

Method Engineering 1996

The proceedings of Method Engineering 1996 are still available. The costs for the proceedings are US$50 inc. shipping and VAT. Please e-mail us your postal address and credit card number with expiration date (or alternatively an invoice address) to:

Sjaak Brinkkemper, Department of Computer Science, University of Twente, P.O.Box 217, NL-7500 AE Enschede, The Netherlands. E-mail: brinkkemper@cs.utwente.nl http://www.cs.utwente.nl/~sjbr

Needless to say that these proceedings are essential reading for anyone active with IS development methodology and method support tools.

Brinkkemper, S., K. Lyytinen and R.J. Welke (Eds.), Method Engineering: Principles of Method Construction and Tool Support. Proceedings of the IFIP WG8.1/8.2 Working Conference on Method Engineering. 26-28 August 1996, Atlanta, USA. ISBN 0.412.79750.X, 324 pages.

Calendar

May 1997

Reasoning About Function: A Special Track to be held during the Tenth Florida Artificial Intelligence Research Symposium (FLAIRS '97) Daytona Beach, Florida, May 10-14, 1997. *http://huckleberry.sfsu.edu/*

1997 Symposium On Software Reusability (SSR'97), Boston, Massachusetts, USA, 18-20 May 1997, http://www.owego.com/~ssr97/

7th International Workshop on Software Configuration Management (SCM7), Boston, 19-20 May 1997. *Info: conradi@idt.unit.no*

International Conference on Software Engineering (ICSE 97), Boston, Massachusetts, USA, May 18-23, 1997. http://www.ics.uci.edu/icse97/

The 7th European - Japanese Conference on Information Modelling And Knowledge Bases, Toulouse, France, May 27- 30, 1997.
http://www.ladseb.pd.cnr.it/infor/Ontology/ontology.html

Tenth International Software Quality Week 1997 (QW'97), San Francisco, California USA, 27-30 May 1997. *http://www.soft.com/QualWeek/*

June 1997

Third International Symposium and Forum on Software Engineering Standards (ISESS 97), Walnut Creek, California, June 1-6, 1997. *Info: John.Harauz@hydro.on.ca*

Sixth International Conference On User Modeling, Chia Laguna, Sardinia, Italy, June 2-5 1997.
http://www.cs.uni-sb.de/UM97/

Fifth International Symposium on Assessment of Software Tools and Technologies (SAST'97), Pittsburgh, PA, USA, 3-5 June, 1997. *http://www.sei.cmu.edu/~case/sast97/*

4th Eurographics Workshop on Design, Specification and Verification of Interactive Systems (DSV-IS'97), Granada, Spain, 4-6 June 1997.
http://www.info.fundp.ac.be/~jvd/dsvis

International conference on Computer Ethics, Linkoping University, Sweden, 9-10 June 1997.
Info: ninni@tema.liu.se

ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'97), Seattle, WA, June 15-18, 1997.
http://www.research.att.com/conf

Third International Workshop on Requirements Engineering: Foundation for Software Quality, (REFSQ'97), Barcelona, Catalonia, June 16-17 1997.
http://alabi.upc.es/~caise97/REFSQ97.html

The 9th Conference On Advanced Information Systems Engineering (CAiSE'97), Barcelona, Catalonia, 16-20 June 1997. *http://www-fib.upc.es/caise97*

12th Annual Conference on Computer Assurance (COMPASS'97), Gaithersburg, MD, June 16-20, 1997. *http://hissa.ncsl.nist.gov/compass/*

IEEE Sixth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE'97), Cambridge, Massachusetts, USA, June 18-20, 1997.
http://www.cerc.wvu.edu/WETICE/WETICE97.html

Ninth International Conference on Software Engineering and Knowledge Engineering (SEKE '97), Madrid, Spain, June 18-20, 1997. *http://www.fi.upm.es/seke97*

Second International Workshop on End User Development at CAiSE*97: The 9th Conference on Advanced Information Systems Engineering, Barcelona, Catalonia, 18-20 June 1997. *http://www.hull.ac.uk/php/mssndm/workshop.html*

European Symposium on Validation and Verification of Knowledge Based Systems (EUROVAV-97), Leuven, Belgium, June 26-28, 1997.
http://www.econ.kuleuven.ac.be/congres/EuroVaV/eurovav97.htm

July 1997

Workshop on Representations in Interactive Software Development, Queen Mary and Westfield College, University of London, 2-4 July 1997.
http://www.dcs.qmw.ac.uk/research/hci/CfP.html

INTERACT'97 Workshop on User-Centred Requirements Engineering: What Techniques To Use? Sydney, Australia, Monday 14th July 1997.
http://i97.syd.dit.csiro.au/i97/i97wprg.html#w4

Software Technology and Engineering Practice (STEP'97) 8th International Workshop (incorporating CASE'97), Holiday Inn, King's Cross, London, 14-18 July 1997. *http://www.co.umist.ac.uk/STEP97*

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/

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

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

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://www.cs.colostate.edu/~malaiya/issre.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/*

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*

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

RE-Bites...

This month's RE-bite is an interesting illustration of the difficulty of getting user requirements right.

Cash machines are pretty similar the world over, and the user interfaces are relatively simple. Not that all of them are easy to use, though. In North America, the banks have a tremendous problem with people leaving their cards in the machines. Besides the security risk, there is a significant cost involved in reuniting customers with the swallowed cards. Why does this not happen in Europe? Are Europeans brighter?

Well, a little psychology is in order. The important concept here is *closure*. When humans perform any goal-oriented task, they experience closure when the main goal is satisfied. With a cash machine, the main goal is usually to get cash, and a user will experience closure once the cash is in his or her hands. There is a strong possibility that if the user is distracted or in a hurry, other actions after closure will be neglected.

So what's the difference between cash machines in North America and Europe? In Europe, the machines give you the card back before giving you your cash. In North America, they retain the card after giving you the cash, and ask if you want another transaction. Only then do they give you the card back. By which time, of course, you have already driven off.

Now the teaser: Originally, European cash machines were programmed the same way as their North American cousins. They were nearly all reprogrammed a couple of years after cash machines were introduced. So how did the banks discover the requirement concerning closure?

International Conference on Systems Sciences (HICSS), Big Island of Hawaii, HI, January 6-9, 1998. *http://www.ce.unipr.it/hicss*

March 1998

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/*

RE-Readings

Reviews of recent Requirements Engineering events.

Workshop on Formal Methods & Requirements Engineering: Challenges & Synergies

City University, 16th and 17th December 1996

Report by Sara Jones and David Till

The aim of this workshop was to encourage communication and understanding between academics and practitioners in the fields of requirements engineering and formal methods, and to identify areas where work in each field might contribute to the other. The workshop was jointly organised by RESG (Requirements Engineering Specialist Group) and FACS (Formal Aspects of Computing Science), both of which are BCS special interest groups.

Around 70 people attended the workshop to hear a distinguished panel of invited speakers presenting Requirements Engineering (RE) and Formal Methods (FM) perspectives on:

- Change in Software and System Requirements and Specifications
- Requirements Traceability
- Requirements Elicitation and Validation
- Non-functional Requirements
- Inconsistency in Software/System Requirements and Specifications
- Use of Multiple Notations
- Architecture
- Domain Knowledge

There were also group discussion sessions on each day in which delegates were divided into smaller groups to discuss issues arising from the presentations, as well as more specific questions about what FM could deliver for RE practitioners, and vice versa. In the final plenary session, delegates were invited to put any further questions directly to the speakers.

Michael Jackson began by Setting the Scene. His own abstract for the talk summarises best the main points:

> "The essence of formality is reasoning with symbols without considering what the symbols stand for. The essence of informality is the unboundedness of the set of considerations that may prove significant. This unboundedness can invalidate otherwise sound logic applied to an informal domain.
>
> Machines are formal. Requirements are informal. To deal with requirements we must explore the problem context far from the machine. We formalise requirements to understand how a formal machine behaviour at the interface affects the world. This formalisation is necessarily imperfect. We must minimise the approximation error by appropriate choices."

One can say the same thing in different ways. When we write software we are specifying some very definite behaviour on the part of the machine, and whether we like it or not the software system is a formal system. On the other hand, the real world is immensely complex and we cannot expect to describe it completely as a formal system. But unless we do formalise some properties of the real world that are of interest for the system concerned, to a sufficient degree of accuracy, there is no way that we can hope to demonstrate that the formal system we specify and build will, in combination with the relevant part of the real world, result in those desirable properties being realised.

The first RE/FM pair of speakers were Ken Eason and John Wordsworth who discussed aspects of Change in Software and System Requirements and Specification. Ken Eason set the scene here by presenting a case study scenario in which a major bank wanted help in specifying a large change programme. He pointed out that stakeholders do not have ready-made requirements at the beginning of a project, but generate them over time. Support for this generation of requirements must therefore be provided as, for example, in the ORDIT method, where increasingly realistic forms of scenario (involving, for example, static paper-based constructions, role play, or trial implementations) are used to assist clients in understanding possible solutions to their problems. He ended his presentation with a challenge to the formal methods community: could formal methods help in managing on-going changes in requirements through the course of a project?

John Wordsworth's claim was that formal methods encourage engineers to circumvent changes to requirements by promoting thorough exploration of requirements before they are agreed between client and supplier. He saw animation and execution of formal specifications as being important tools in allowing clients to see the effects of a

proposed system function in advance of development and claimed that formal methods in general were helpful in allowing project managers to assess the risk of changes to system requirements or functions as well as the effort required to make the changes. On a more cautionary note, however, he noted that the use of formal methods demands substantial intellectual investment, and that system specifiers are therefore likely to resist change once a specification has been developed.

Laurence James spoke about Requirements Traceability and addressed what he calls the higher-level principles of requirements traceability: closure (connections between requirements all the way through to delivered system); continuity (consistent information flow between different project viewpoints); logic (solutions developed are correct); emotion (this seems to be the requirements engineering version of the feel-good factor!) -- "traceability is emotion backed up by logic". At a more pragmatic level, he emphasised the importance within an organisation of knowing what are its requirements for a traceability tool, and especially understanding the likely impact of the use of such a tool on their development process. He also claimed that traceability tools can support the necessary bridges between informal requirements and formal specifications of components and their properties. He said that "logic and traceability are the two faces of requirements". Another way to express this is to say that formal languages can be used to specify precisely and unambiguously what is needed, while traces help to answer questions about why such precisely described components are needed and how they relate to the informally expressed requirements. Francisco Pinheiro showed how the requirements for traceability tools can themselves be formally defined, but pointed out that any such formal framework should not be too prescriptive because traceability needs should be guided by necessity rather than predefined structure. He introduced the TOOR approach to traceability as an example of an appropriately formal attack on the problem.

The topic considered after lunch was that of Requirements Elicitation and Validation. Lee McCluskey had the challenge of presenting a formal methods perspective on these issues. He began with some observations regarding the different sets of jargon used by those in the RE and FM communities. Although there is some overlap, with languages such as RML straddling the jargon divide to a certain extent, he noted that there are apparently also many differences in the concerns of the two communities, judging from keywords used in conference proceedings of the respective communities.

Lee's talk then focused on the validation of formal requirements specifications which he felt was an important and little researched area. He suggested that the design of formal specification languages should perhaps be influenced to a greater extent by the need to validate specifications, and by the characteristics of particular applications or domains. While validation of formal specifications is still so difficult, no single method is likely to be sufficient on its own, particularly in the development of safety critical applications where reliable validation procedures are crucial. The solution he proposed was to combine a number of different methods. He ended his talk by describing an approach which he had developed in projects funded by the UK Civil Aviation Authority and other bodies, and which involved combining manual checking with automated syntax and type checking, simulation or animation of specifications, and formal proofs.

John Dobson's talk adopted a wider perspective and presented a common framework for model building which was intended to show how sets of vocabulary items and composition rules for different representations relate to common conceptual models and architectural principles according to which a system is to be built. His framework consisted of five levels:

- the natural language level, at which requirements are articulated;
- the conceptual level, at which the concerns of a client are framed and alternative models of requirements are explored;
- the logical level, where appropriate calculi are identified and formal models are defined;
- the design level, where real-world interpretations of operations described in the calculi are defined;
- and finally, the descriptive level, at which the implications of these interpretations are explored.

The final session of the first day was devoted to so-called Non-Functional Requirements. Ian Sommerville began the session by declaring himself as someone who didn't believe in the existence of non-functional requirements as a kind of requirement that is fundamentally different from functional requirements. In fact, he claimed, assuming such a distinction could be made had been largely responsible for the lack of uptake of academic formal methods in industry. This was because formal methods research had focused exclusively on functional requirements, ignoring their inextricable links with non-functional requirements, while industry had to be very concerned with so-called non-functional aspects of the systems they produced. The solution he proposed was to forget trying to make any distinction between functional and non-functional requirements and to focus research on developing formal notations which are 'usable by real people working on real systems'. The emphasis should be on sending industry a simple message about what techniques and notations to use under what circumstances, and on providing good support, for example, in the form of tools, for notations which are already available.

David Robertson also discussed the fact that the term 'non-functional' can, at best, be only loosely defined. However, taking a definition encompassing ideas such as 'aspects of a system which don't influence its mechanics', 'descriptions of problems, not all aspects of which may influence directly the systems we build to solve them', and 'various other things to do with higher levels of design', he was able to point to several examples in which formal methods had been

used to help reveal argument and model structures, and to assist in developing formal interpretations of knowledge whose structure can not be easily accessed. In general, however, his feeling was that the use of formal methods in dealing with non-functional requirements was still at a 'pre-engineering' stage, so that examples of good practice are scarce, and reliable guidelines for their use in this context are non-existent.

After the two presentations, there was some discussion of the idea that it may not always be desirable to formalise non-functional requirements relating to, for example, the political implications of a new system. However one delegate described how her organisation had used FUSION to specify at least some non-functional requirements relating to users and stakeholders along with requirements relating more directly to the system required.

The two speakers on Inconsistency in Software/System Requirements and Specifications were Alfonso Fuggetta and Tony Hunter. Alfonso Fuggetta pointed out that though we ultimately would like a consistent and complete specification of the problem to be solved, the process of reaching such a happy state of affairs is likely to pass through intermediate states in which our knowledge is partial and even inconsistent. This is inevitable and indeed necessary because the detection of inconsistency is one of the major driving forces to discover more salient information about the domain and the problem to be solved within that domain. Of course if we want to define what we mean by inconsistency in a given context we will have to formulate formal consistency rules or rely on the inference mechanisms of a logical language.

However it is important to realise that often we will not detect this kind of formal inconsistency because our formal descriptions do not take account of enough of the significant properties of the domain. It is also possible that inconsistency will be detected as a result of different usage of the same terms within different viewpoints, when in fact there is no real underlying conflict. Tony Hunter also illustrated how inconsistency can be a spur to action; in his examples, upon further investigation it transpires that there were unstated assumptions. He also described how we can reason in the presence of inconsistency by using quasi-classical logic. We do not want to eliminate inconsistency; we want to manage it. In particular, we may want to investigate the sources of inconsistency, we may want to ignore it or delay its resolution. If we do in the end resolve it, it is important to keep a trace of it, since this will remind us of what conflicts had to be addressed, and indeed who was involved in the decisions which led to its resolution.

Tom Maibaum and Jose Fiadeiro have worked together over many years on the Use of Multiple Notations. Tom Maibaum took up enthusiastically the role of the Requirements Engineer looking for help from the Formal Methodist. He motivated the use of multiple notations by different views (stakeholders) in order to reduce complexity and to allow for independent and natural expression of different aspects of a problem to be solved. His specific challenge to formal methods was:

> "How can we relate descriptions in different formalisms with possibly different structuring principles and almost certainly different structures?"

Jose Fiadeiro recast the problem:

> "How can we relate and integrate different notations so that we can understand the whole, detect interferences and support incremental development and evolution?"

Many authors have explored the idea of translating notations into a common style of predicate logic and then integrating on the basis of a common semantic domain. Maibaum and Fiadeiro have proposed as an alternative the use of category theory as a convenient mathematical framework for interaction: this allows what they call 'exoskeletal' representation of interaction (the explicit linkage of entities in different views, which must be treated as identical) and translation between structuring principles.

The Software Architecture theme was addressed by Jeff Kramer and Susan Eisenbach. In his abstract, Jeff Kramer says that software architecture is in the solution domain, while requirements are in the problem domain, though requirements themselves often do describe an architecture of sorts, perhaps best represented by Michael Jackson's problem frames. He also suggests that it can be very difficult to state or to understand requirements clearly until some feasible architecture has been contemplated. Thus, in many cases, the cycle of activities most likely to produce better understanding of how high-level system goals may be operationalised is to push forward towards design of a feasible implementation based on well-tried software architectures and then to step back again, having achieved greater insight into what is possible and the kind of behavioural characteristics to be expected from such an implementation. Of course, this assumes that the software architectures concerned are such that certain important behavioural characteristics—for example, performance measures—can be inferred without knowing the details of the plug-in, application-specific modules which will be needed. Thus Jeff was concerned to point out the utility of software architecture within the requirements process; Susan Eisenbach went on to show how such architectures can be formally described, using the pi-calculus and logic, illustrating with a simple example.

The final session of the workshop was devoted to the role of Domain Knowledge in requirements specification. Neil Maiden's talk described the work of the European NATURE project on determining the way in which domain knowledge could enable reuse of information about common user requirements or system contexts. However, he suggested there was still a need for firmer theoretical foundations for defining the notion of a domain. Jeremy Holland described his work on a project in which a formal representation of a particular domain of activity (that of clinical activity in a London hospital) was used in the development of new

information systems for the hospital. Based on his experience of that project, he felt that the existence of a formal domain model had assisted in reasoning about the domain, and hence about the system to be developed, exposing contradictions and inconsistencies in information given to him by system stakeholders. It had also led to greater control in designing the system through progressive refinement of the model. He concluded that the process had been time-consuming, but that the time had been well spent.

It is of course more difficult to summarise what emerged from the discussion groups since they were intended to be more open-ended. Here are some of the questions and points that arose during the discussions on the first day of the meeting:

Has the focus of formal methods been too narrow? It was suggested that a pragmatic view might be: informal, structured methods give some rapid initial results; formal models should then be built, but kept secret from the client (though the analyst should be able to read from the model into the client's world and terminology); formal proofs only done in mission or safety critical areas.

Is requirements capture/analysis/definition an engineering discipline? Engineering implies process with feedback, starting and stopping criteria.

Does RE ever stop?

Can we avoid thinking about solutions?

What are the skills needed by a Requirements Engineer?

What are the building blocks for RE?

The main benefit of FM for RE is in posing questions.

We need a better understanding of what RE problems are.

We need to be able to classify problems to see which techniques may help.

It is hard to talk about problem space as distinct from solution space.

The concerns of FM and RE are not keeping pace with changes in the real world; we need to look more at re-engineering and product families.

How can we achieve better synergy of academic expertise and the needs of industry?

Controlled case study experiments, improved metrics, cost/benefit analysis in different areas.

How do we know we are making progress?

How can we measure the quality of requirements?

How can we compare methods?

How do we select the right combination of techniques?

On the second day, discussion groups were asked to consider what formal methods could deliver to requirements engineering practitioners in the next 1, 5 or 10 years, and vice versa.

In answer to the first question (What can FM deliver for RE?), it was suggested that: FM might deliver a basis for problem and solution classifications, perhaps within 1 or 5 years, which would then enable a mapping between appropriate problem and solution patterns. Within 5 years, FM may be able to deliver better natural language paraphrasing of formal specifications to facilitate validation, and more usable formal methods! Within 10 years it was hoped that FM might be able to assist in problems of feature interaction, and that techniques for theorem proving might also be made more usable

It was also pointed out that different formal methods were likely to have different impacts in different types of system development. For example, it was thought that formal methods would have different impacts in the development of control, business information and highly interactive systems, with safety critical and control systems being likely to benefit most quickly, and business information systems being unlikely to benefit within the timeframe considered.

The second question (What can RE deliver for FM?) elicited the following suggestions:

A better understanding of approaches to negotiation which could be used in discussing trade-offs between different formal models.

A basis for formal modelling of impact, sensitivity and consequence analysis regarding changes to specifications.

A better way of maintaining links between a formal specification and the 'real world'.

A need for greater co-operation between the two communities of FM and RE in terms of transferring skills and techniques was also identified by three of the four groups.

In conclusion, our impression was that all the participants felt they had heard something useful and stimulating during the two days of the workshop. Of course it would be unrealistic to expect a meeting such as this to achieve major breakthroughs in an area as challenging as this one, but it must be of benefit to the members of the two overlapping communities present at the workshop if they understand better each other's concerns.

Third IEEE International Symposium on Requirements Engineering (RE'97)

Marriott Waterfront Hotel in Annapolis, Maryland, USA, January 5 to 10, 1997

Report by Zhong Zhang and Swarn Dhaliwal

About 190 computer scientists representing 19 countries attended this conference. Details about this conference can be found at http://www.itd.nrl.navy.mil/conf/ISRE97. This report summarizes various scientific activities conducted during the course of the symposium with a special reference to various kinds of commercial tools exhibited at the conference.

Several trends in research and application of formal requirements could be seen at this conference. It was exciting to see that the use of formal methods have received broad attention from industry since the last RE symposium.

One of the major highlights of RE'97 were the five tutorials and one mini-tutorial presented by a mix of academic and industrial speakers. All of the tutorials were very well

received and every session was packed to capacity with participants, most of whom were from industry. The full-day tutorial session "making Requirements Measurable" given by Bashar Nuseibeh and Suzanne Robertson from UK, examined requirements measurability by applying a requirements template to a familiar system. In Anthony Finkelstein's tutorial on "Requirements Traceability", he provided a comprehensive overview of the requirements traceability theories and the current tools used to support it. For those wanting to make their informal requirements more formal and measurable, tutorials on object-oriented and formal requirements methods were available. In the tutorial "Advanced Object-Oriented Requirements Specification Methods", Roel Wieringa discussed four methods for structured analysis: a) the Unified Modeling language of Rumbaugh, Booch and Jacobson; b) Fusion extended with use cases; c) OOA (Shlaer-Mellor); d) and the Yourdon Systems Method. Mats Heimdahl concentrated on safety critical systems, ina tutorial entitled "Software Requirements Specification and System Safety". The Software Cost Reduction method (SCR*) tool is industry-oriented, and has found wide-spread use in many industrial organizations. A tutorial on "The SCR Approach to Requirements specification and analysis" by Connie Heitmeyer was warmly received by the people from industry. In a one and half hour minitutorial on "Model Checking and requirements", Daniel Jackson, from CMU, presented a survey of different model checking techniques and tools and described the explained what model checking is. He remarked that "big companies are now willing to spend more money on formal specifications and testing", and underlined the importance of model checking by describing Intel's desire to hire the entire world's output of model checking Ph.D graduates.

Unfortunately, there was not much participation from "big" computing companies like Intel, Microsoft, and Sun. Their approaches to requirements engineering remain a mystery! However, the participant list of tutorial sessions indicated that smaller companies are willing to spend money on developing more precise (and formal!) specifications. This is now recognized as the first step in developing quality software.

Another highlight of the conference was the four keynote speeches delivered by eminent computer scientists. Dr. Anthony Hall, who pioneered the use of formal specifications in industrial projects and led the design of the CDIS air traffic information system, addressed the essential problem in requirement engineering in his speech entitled "What is the use of requirements engineering?". Professor Colin Potts, from the Georgia Institute of Technology, looked at the RE more from social scientist's perspective and his opinions led to an interesting discussion. Dr. John Rushby, the program director of SRI's computer science laboratory and lead developer of PVS, gave a talk entitled "Calculating with Requirements". He gave extensive examples from the NASA Space Shuttle software to explain the importance of measurable requirements. Professor David Harel from Weizmann Institute of science and the inventor of the statecharts notation, spoke on the topic "Will I be pretty, Will I be Rich?" which featured some thoughts on theory versus practice in systems engineering, presented in a very lively manner, with plenty of cartoons!

Several tools for requirements engineering were exhibited during the conference. Interested participants were given individual or group demonstrations by expert professionals. Some of the tools are already in commercial use in different places.

DOORS is a requirement traceability tool developed by Quality Systems & Software. By using a backbone database, it records all of the information noted by different people and provides a layered security mechanism to prevent illegal access and inadvertent modification. It had been implemented using C++ programming language uses an objected-oriented database as the backbone.

CORE has been developed by Vitech Corporation. It is a comprehensive stand-alone tool that supports the complete system engineering process.

GRAIL/KAOS intends to support the kaos methodology, which is a formal language developed at Universite catholique de Louvain of Belgium. GRAIL is an integrated tool that uses existing tools to support its functionality. For example, it uses "dot" and "Frame Maker" to support its graphic and text editors. Also, it communicates with an Orbix database to retrieve information. There seems to be growing trend towards integrating existing tools to build new tools.

*SCR**, which was developed at Naval Research Laboratory, supports a tabular approach to building state-based specifications for embedded control systems. They have recently integrated model checking capabilities into the tool, allowing for powerful analysis of safety requirements.

All of these tools support automated report generation and almost all of them can interact with other components such as Microsoft's OLE objects. The integration of these tools with the PC API seems to be a growing trend.

More and more companies are getting interested in requirements engineering. They are already using Object-Oriented requirements methods, and are increasingly interested in formal specification. Various tools already in practice or being developed promise to make requirements more measurable and more traceable.

In this report we have concentrated on the tutorials, keynotes and tools exhibition. There were, of course, many interesting papers in the technical programme on various topics related to requirements engineering. A large number of them emphasized the use of formal methods by demonstrating experiences (case studies) involving formal methods. We encourage everyone to read the proceedings to for these papers! Overall, the conference was very informative and educational. We had a great opportunity to meet and talk to the people working in requirements engineering.

RESG Seminar on Business Specification Patterns

RESG Seminar held at Imperial College, London, on 26 March 1997. The speaker was Haim Kilov.

Report by Ian Alexander (iany@easynet.co.uk)

Haim Kilov arrived to find a packed room 418 in the Computer Science Department at Imperial College. The eager ears belonged to a mixture of academics and consultants. Kilov at once challenged us to consider

> "Why are specifications more like fairy tales than like programs?"

And he continued to hammer away at the theme of precision of engineering expression throughout the afternoon. Typical Kilov sayings include "A warm and fuzzy feeling is not a definition" and "In real life, things don't exchange messages".

Readers not familiar with Kilov's style can at once glimpse his wonderful mixture of warmth, humour, and precision. As a speaker he occasionally risks assuming that the audience shares his breadth of technical knowledge or of literary allusion. For the benefit of this European audience, he assured us that the slides contained plenty of European quotations—from such luminaries as Edsger Dijkstra and Tony Hoare, among others.

How is precision to be achieved? According to Kilov, it is by following the same rules in our specifications that objects follow in programs. The concepts used in a good specification should be explicit, with exact semantics. Once we have discovered patterns that recur, we can reuse these without effort. But this is only possible if we have the ability to recognise such patterns, and a 'pattern-book' to keep them in.

Some of the themes must therefore be familiar to all engineers: we are to separate business and system concerns, seek reusable objects, aim to be simple and elegant.

Kilov reserves his most acerbic humour for those who imagine that one can discover a system's specification by inspecting its behaviour: 'Oh, it does what it does'. He quotes Parnas' unfavourable comparison of computer people with 'Professional Engineers' who 'use mathematics to specify, describe, and analyze their products. No professional engineer would claim that intuition alone is sufficient, or that mathematics is too difficult for an engineer.'

The Kilov method relies on a remarkably small and simple set of diagrammable constructs. Objects (simple boxes) are connected by a range of one-to-many relationships, each drawn with a triangular waypoint containing a type label. For example, composition is shown with the letter 'C'. Those who do not favour diagrams can write composes(A, B) if they prefer: Kilov does not want to join the wars of the diagrams.

The seminar concluded with a presentation of several patterns, such as of a business decision, presented as diagrams. It was noticeable that the audience had arrived with many different concerns and applications for business specifications, and there was a lively and interesting floor discussion after the talk.

RE-Papers

Problems When Acquiring Requirements for Commercial Off-The-Shelf Package Selection.

C. Ncube & N.A.M. Maiden,

Centre for Human-Computer Interface Design, City University,, Northampton Square, London EC1V OHB, Tel: +44-171-477-8412, Fax: +44-171-477-8859 E-mail:[C.Ncube, N.A.M.Maiden]@city.ac.uk

Introduction

We report 10 problems experienced during the selection of a complex commercial off-the-shelf (COTS) software system. Particular focus is put on the typical problems which arose during acquisition of requirements, evaluation and selection of the products and the decision making process for recommendation. We learned some valuable lessons and experiences about procurement oriented requirements engineering and the selection of COTS software packages. We are therefore willing to discuss our own experiences with other people who are involved in similar studies or have experienced similar problems. We have also compiled a list of our experiences, problems and lessons learned from the evaluation process which, we hope, can improve future similar evaluations.

The problem

Our customer was a large defence agency that wanted new methods and software tools to manage requirements and procurement activities for a new system, which would take 20 years to develop. We were asked to acquire requirements about their tool needs and recommend commercial available requirements engineering methods and tools (known as products) for trial by the client.

We had 11 weeks to make a recommendation to the client and this led to considerable time pressures. Requirements acquisition took place from both documents and stakeholders. Five meetings with between 6 and 10 stakeholders took place over a 3-week period. Each was divided into a review of the current requirements document and acquisition of new requirements. The final requirements document contained 133 atomic requirement statements in 22 basic hierarchies.

To save time, market research to identify candidate products was undertaken in parallel with requirements acquisition. An

initial version of the requirements document was drafted as a questionnaire and sent to over 30 candidate suppliers to determine the coverage of their products. After an initial paper evaluation, a shortlist of 6 candidate products was produced using supplier responses to the questionnaires. Next, a set of 35 complex test cases for product evaluation was developed from the final requirements document. The 35 test cases were further decomposed into 135 atomic compliance checks. Five of the 6 shortlisted suppliers demonstrated their product against the 135 compliance checks. During each evaluation, product compliance to requirements was recorded by all team members using both quantitative scores and qualitative comments. After each evaluation, the final product-requirement compliance scores were agreed by the members. These scores were then investigated using a range of analytic techniques. As a result, a trial use of two requirements engineering software tools was recommended.

The Problems Encountered

The requirements acquisition and product selection processes were not without problems. Here we outline 10 problems that were encountered during the process. In retrospect, some of these problems might have been avoided by asking more useful questions at the right time throughout acquisition.

Problem 1: Too much time was spent acquiring and modelling requirements met by all the 5 evaluated products. Most of these requirements did not enable us to discriminate between products during selection. We did not model other requirements in sufficient detail to enable effective product selection. As a result, product selection was more difficult.

Problem 2: It was difficult to measure product-requirement compliance because most of the requirements acquired from stakeholders were not "measurable". Due to time pressures, we were not able to ask questions such as "how can you measure this requirement?".

Problem 3: Generation of measurable test cases was difficult without prior knowledge about candidate products or prior experience of test case generation. It was also difficult without a prototype to determine the correct responses to complex data base queries during product evaluation.

Problem 4: The hierarchical structure of the requirements, when mapped to the sequential structure of the test cases, made test case generation difficult. For example the requirement "the product must be configurable to client needs" was included in all 35 test cases in order to evaluate how configurable were different product functions. This weakened the link between requirements and test cases, and made test case management more difficult.

Problem 5: Requirements management tools are complex and depend on other products such as data base management systems and word processors. We had to evaluate these other products to undertake a complete evaluation, which in turn made the evaluation more complex and time-consuming.

Problem 6: We sometimes needed to ask detailed questions during product demonstrations using information about the application domain. Such application domain information was not always acquired from stakeholders prior to evaluation or was not recallable by the team members during the evaluation.

Problem 7: The large number of requirements made product-requirement compliance a complex task. We made over 1500 compliance decisions during 18 hours of product evaluation. We lacked the techniques to record the rationale for these decisions. This, in turn, made the agreement of product-requirement compliance scores within the team after each evaluation more difficult than anticipated because all the reasons for the scores had not been recorded.

Problem 8: Biases are always possible when scoring product-requirement compliance. We did a post-evaluation analysis of individual and agreed product-requirement compliance scores and this revealed a trend towards agreement with one team member more than the others. This was due in part to the different experiences of the team members. Another reason for bias was fatigue. Each product evaluation lasted over 3 hours, during which over 300 product-requirement compliance decisions were made. Occasional lapses in attention are almost inevitable during such evaluations.

Problem 9: A client representative, (who was the study owner), weighted all higher-level requirement statements using simple percentage scores. He then categorised each lower-level requirements as either essential, desirable and optional. However, the weightings were sometimes inconsistent with both the identified essential requirements and the opinions of other stakeholders.

Problem 10: A requirements management tool is a complex COTS product. A proper evaluation needs effective demonstration by supplier representatives. However, in our experience, the quality of this demonstration varied considerably across products.

Learning the Lessons

Despite the reported problems, product selection was successful and our client was content with our recommendations. However the product selection process could have been improved. We have used the reported problems as well as those not reported here to propose simple techniques to improve requirements acquisition for COTS selection.

As part of our ongoing research, we are developing techniques to overcome these problems and we are aiming to validate and report these techniques in the near future. However, we are open to expressions of interest from people with similar problems and/or experiences.

CORE - Blimey!

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

Making Project Support Viable

In previous contributions I have implied that there are many viewpoints of a proposed system; that they are not all technical and that the quantity and complexity of information is often far greater than current tools can support effectively. In this contribution I indicate that a tool set is possible which can process information whose quantity, diversity and distribution is great and yet still support the specialised processing evident in many current support tools.

The first point to make is that technical and non-technical information do not exist in isolation from one another. Also, different shades of technical information or of non-technical information are rarely wholly mutually exclusive. Different types of view often share concepts, but use different terminology. One view may make use of concepts distinct from those in other views, but related to shared concepts. Failure to recognise overlaps and relationships between views is one way to ensure that what is done fails to meet project objectives.

For example project management and software developers both use component break-downs. Both use control of processes and both recognise iteration, selection and (in general) concurrency. Each has developed tools which support these concepts. However the management view refers to, for example, PERT and/or GANTT charts and work break-down structure, whereas software developers refer to scheduling and process/control hierarchies.

There are dependencies between the two views. A management work break-down identifies tasks which develop technical components. Management needs to know when a task is complete. That can only realistically be determined by examination of technical components for analytically detectable evidence of incompleteness/residual error. Component developers need to know about who, where and when they need to consult about other parts of the system (other components or other related projects). That can only realistically be determined by examining management information for details about project responsibilities and time-scales.

The second point to make is that within any one major view it is rarely true that there is only one way to proceed or to represent information and there is frequently conflict to be resolved in deciding how to proceed or what to include in a representation. Failure to recognise conflicts can lead to failure to resolve them or later ignorance about key decisions. It can also lead to failure to make clear to the entire audience for the product what it means and how it addresses their needs.

For example it is fashionable to preclude use of procedural specification. Yet observing actual non-procedural specifications it is quite common to find that they are highly procedural in nature or that they resulted from an informal procedural description which was never written down or was discarded.

A fashionable management approach is Just In Time management, which demands that stockpiling of components is minimised and components are made available just as the need for them becomes necessary. This works, until the component supply process fails in an unforeseen manner - whereupon disruption occurs, costing far more than the cost of storing extra spares.

In neither example is it my proposition that one approach is right and others wrong. I am saying that none of the approaches is universally best and that it is damaging to assume otherwise. Instead it is desirable that transformations between approaches should be sought. The larger the project, the more likely it is that different parts will find different approaches valuable. There is a need to support several possible approaches, deciding at points of conflict which influence should prevail.

The third point is that the quantity, variety and distribution of information relevant to a large project is far too great for most support tools I know of. There are three “flavours” of tool set: database tools (built on relational or OO principles), AI tools (built on highly flexible languages like LISP or Prolog) and bespoke tools (built as CASE tools to support a specific language/method).

Conventional relational database tools are too inflexible without recourse to very opaque data structures and with frequent need to descend into low level languages like COBOL, FORTRAN or C to perform important operations. Since I have not made direct use of any object oriented database I cannot say to what extent they might address the problems on which relational databases fall down. However, since they (presumably) rely on the use of classification schemes, and since much of the original source information for project requirements does not fit neatly into such a scheme until it has been massaged into another form, I am dubious of the idea that OO database technology alone is enough.

AI-based tools I know of are inadequate for handling large volumes of distributed data and again require the source data to be massaged into another form, leaving the mapping between forms largely undocumented. Attempts I have seen to give mass data handling capability to AI languages have always appeared to me to founder on the mismatch between the basic principle of flexible expression so important to AI languages and the rigidity of expression inherent in relational table record/field definition of database schemas.

CASE tools I know of are too specialised towards a specific method/language (e.g. support of a structured method such as Yourdon or JSD). The ones I have seen, even where they are built with some possibilities for handling large distributed data sets and interfacing to other methods/languages do not in fact seem to be practically usable for such interfacing.

The fourth and final point I would like to make is that it is unreasonable to expect any one contributor to solve the problem of building a tool set which overcomes the problems inherent in addressing the three earlier points. The nearest I have seen to an attempt to do that was the fashion in the 1980s for APSEs and IPSEs. I have not heard of and certainly not seen a successful general project support environment, in spite of having been involved in an attempt to produce one.

The one I was involved with was IST's ISTAR project which was very ambitious and had some excellent technicians involved in its development (no, I don't mean me!). It could not succeed as a general support environment for a number of reasons, but primary among these was the fact that the scope of the support tools and the distributed nature of the user environment was too great for the expertise available to the company at the time.

In support of this proposition I would point out that in the microcomputer world there are a number of tool builders with potentially enormous resources (finance, equipment and people) who have spent over ten years gradually improving the state of the art in software development tools to the point where there now exist several (arguably) excellent code support environments. However, the nearest they have to a general project support environment are a series of fragmented tools completely separated from or crudely bolted on to their IDE - such as version control, build control, project planning, code browser and database system.

The interaction between these is slightly above the level of pathetic. The reason for that is that there is no one around in these companies with an understanding of both how to build the individual tool types and how and why the information in the various types of tool really needs to be inter-related.

One thing that has become available which makes possible a significant improvement in the current situation is support for data exchange between tools and between sites. This means that the builders of an individual tool can make provision for the import and export of data between their tool and tools built by others. That is what Open Doc, OLE and the Internet has given us.

Arising from that is the pressure for movement away from software bloat, which was the feature of the 1980s and the first half of the 1990s, towards small dedicated tools acting as agents to perform tasks in a highly flexible manner, potentially across widely distributed sites and teams. That is what can follow from the availability of scripting languages and Java, with its ability to interact with programs on a particular host site, written in other languages.

These facilities mean that the builders of an individual tool or even of a small tool set do not have to solve the whole development support problem. Rather, they need only use the facilities I have mentioned, together with possibly one other, to make provision for the exchange of data between their tool/tool set and those of other support tool builders.

The builders of development or management methods/tool support need only specify the data they need to receive from other tools and the data (and its structure) they can make available to other tools/tool sets. For full support they would define the software interface via which such exchanges could be automated by direct calls between the tools. In the regime I am describing a particular development team's tool set would consist of a number of small special purpose tools with well defined data exchange interfaces between them.

For support tool builders the net effect would be twofold. First, any individual tool would be easier to build, since it would not be trying to serve several purposes simultaneously. Second, via scripting and/or Java facilities, various combinations of components could be tried to discover a viable combination of special purpose tools exchanging well defined data for a specific environment.

This does not require a vast widening of programming skills for those specifying methods and building support tools. It requires that they acquire a small set of additional skills which have relatively recently become available - like use of OLE or Open Doc, scripting languages and/or Java. These are techniques software developers are urging others to learn to use - it seems to me to be only right that we should ourselves be using them to their best purpose.

The one feature that may still be missing is something that provides a compatible mapping between AI languages and database support technology. This must support highly flexible means of expression without making too obscure the means by which data may be sought. It must permit flexible expression and inter-relation of large quantities or information and its distribution across a number of cultures (management, hardware, software, and various user domains) and sites (multiple developer, multiple user, multiple management, multiple customer).

This is something which has absorbed me for over a decade and for which I have no complete solution. However there is progress I believe can be made on this front, with technology whose feasibility I can demonstrate and for which I suspect there are individual research projects with solutions already waiting, but not yet turned to this purpose. That is something I propose to discuss in a later contribution.

Geoff Mullery
gmul@sml.win-uk.net

RE-Publications

Book Review: Haim Kilov and William Harvey eds; Object-Oriented Behavioral Specifications. Kluwer, 1996; ISBN 0-7923-9778-9.

Reviewed by Michael Jackson

This book is a collection of papers from four workshops on behavioural semantics, held at OOPSLA in the years 1992 to 1995. Like most workshops, they attracted contributions of widely varying nature and quality. The editors despaired of classifying the papers by any "particular semantic-oriented criterion". Instead, they chose "the least semantic way to classify papers: alphabetically, by the first author's last name."

Some of the papers are firmly of the formal methods school. For example, Gary Leavens contributes an overview of the Larch approach, showing how to use the Larch Shared Language (LSL) and then Larch/C++ to specify various C++ classes such as Quadrilateral and Parallelogram. Several papers use Z to specify abstract data types. Reino Kurki-Suonio explores fundamental aspects of collective behavioural specification of objects, and goes on to outline a modular design approach based on refinement and superposition. One paper is based on category theory.

Some of the other papers fall clearly at the other end of the spectrum. For example, Joseph Marabito and Anilkumar Bhate offer some thoughts about the culture of organisations. They claim to specify their 'culture molecule' quite precisely using Kilov and Ross's information modelling concepts. Culture is defined as "the composite specification of behaviour resulting from the inter-actions of its components"; it is "subtyped as a composition-assembly. That is to say, the specification of culture derives from all of its components and does not exist independently." Their specification, in textual form, is:

fit: composition (culture, {human, structure, information, tool, memory, world-view, strategy})

This is certainly not precision in any useful sense. It is not clear how such an application of a composition operator can be precise when the terms to which it is applied are so imprecise. As von Neumann and Morgenstern pointed out in "The Theory of Games and Economic Behaviour": There is no point in using exact methods where there is no clarity in the concepts and issues to which they are to be applied."

William Harvey, Karen Bilotta and Cynthia Grese put forward a tantalisingly interesting notion of 'information refraction'. Information, like light, is refracted where it crosses a boundary between distinct environments. The authors draw on a wide range of examples. They claim that the millenium problem is due to information refraction. They cite the Physician's Alert system, designed to alert a doctor to a patient's history of medical litigation. They mention latency in bank statements due to batch updates, the Challenger O-ring problem, and the F-18 fighter. But disappointingly they do not really develop their notion of information refraction into anything more specific than a vague danger that can be averted by their three familiar recommendations: to understand user needs, specify invariants, and design a model to simulate the 'real world'.

As one might expect from a collection of workshop papers, the book contains a lot of interesting material, along with some that is less interesting. The editors claim that the papers show that "abstract and precise specifications of behavioral semantics are being successfully used in industry, both for requirement specification and for program development." But in fact there is very little in the book to support this claim. Regrettably, it would be closer to the truth to say that what is abstract and precise is not being successfully used in industry, and what is being successfully used in industry is not abstract and precise. This may well be more an indictment of our field than a criticism of this book, but the book—with the possible exception of one paper—certainly brings it vividly to mind.

Straddling the practical-formal divide, and perhaps giving the lie to the comment above that precision and abstraction are not much used in practice in industry, is the paper by Leona Morgenstern: "Specifying and Reasoning about Business Rules in a Semantic Network." Morgenstern discusses the use of augmented semantic networks to capture sets of business rules in the form of logical formulae attached to the nodes of the network. She gives solid and convincing examples drawn from an expert system for enquiring about benefits in a medical insurance context. She also discusses rule inheritance along is-a arcs of the network, and a treatment of the inconsistencies that may arise.

Of the practitioner papers in the book, I found Jean Stanford's the most attractive. The paper is entitled "Enterprise Modeling with Use Cases: Extending Jacobson's Approach". Stanford took the Use Case methodology seriously, and tried to apply it just as described.

For example, she quotes Jacobson (The Object Advantage, 1994): "... each actor in the superordinate use-case model corresponds exactly to an actor in the subordinate use-case model. ...". But Stanford found that this prescription fails for her application: the actor in the subordinate use-case does not correspond to any actor in the higher-level use case "unless an extremely flexible use of the term 'corresponds exactly' is adopted."

Nor did the Use-Case notion of Value Added activities work well within the context of a single use-case. Stanford found that she had to add activities whose purpose is to support the enterprise infrastructure or to support implementation of other use-cases, not, as stipulated by the methodology, to provide direct value to the external actors of the use-case being described.

The great merits of this paper are its honest practicality and the modest solidity of its contribution. The chosen method was applied conscientiously. Where it failed, the failures were analysed and practical solutions were found. The paper distils the resulting experience into six recommendations that all practitioners of the use-case approach would do well to consider. In almost every way, this is a model of what such a paper should be.

Several of the more formal papers seem particularly useful and important. Kurki-Suonio's paper is insightful and instructive. A paper by Carlos Paredes, Jose Luiz Fiadeiro and Jose Felix Costa is entitled "Architectural Specifications: Modeling and Structuring Behaviour through Rules". Many potential readers will be daunted by this paper, which relies on category theory. But it addresses many issues of fundamental importance in object behaviour in a way that should repay patient study. For example, it discusses a systematic treatment of non-monotonic extension to a class definition. The definition of a Savings account may permit withdrawals and deposits while stipulating that the balance is always non-negative. A Credit account is a non-monotonic extension, re-using the definitions of the withdrawal and deposit operations but allowing the balance to be negative.

Another interesting paper with a formal flavour is Bernhard Rumpe and Cornel Klein's paper "Automata Describing Object Behaviour". Like Kurki-Suonio they offer a fundamental theory of behaviour and a refinement calculus intended to be used in development. They illustrate their ideas with a small example: a class 2D-Figure of two-dimensional graphic objects that can be selected and deselected on the screen, and shown as filled or empty. As often happens, a well-chosen small example is very illuminating. Industrial-scale application, of course, is quite another matter.

All in all, this collection is a very mixed bag. There are eighteen papers representing almost every possible level of precision and formality, of practicality and insight. Most readers will find something here of value; I certainly did.

Book Review: Linda A. Macaulay, Requirements Engineering. Springer, 1996, 202pp, ISBN 3-540-76006-7

Reviewed by Ismail Ismail, Department of Computer Science, University College London

This is an interesting and well crafted book, addressing a series of issues which are rapidly becoming realised at all levels of management. Macaulay's book, usefully, describes the human, organisational and technical issues involved in eliciting and deriving requirements: the main thrust of the book seems to follow a "good practice" paradigm, and subsequently, describes various "established" methods for understanding organisational and lower level concerns of developing software systems.

The book is clearly aimed at undergraduates and masters students, however I am not entirely convinced of it's utility to practicing Requirements Engineers since the level of description is a little too superficial for any real exploration.

The book is relatively short (202 pages including appendices) and sensibly organised in to six chapters (and three appendices) describing the rationale, rhetoric and routine of engineering requirements. The structure is as follows:

The first chapter motivates the plight of RE; a plethora of definitions are used to instantiate the root of the problem, followed by a description of the Requirements Engineer's task. The crux of Macaulay's book seems to centre around the notion that, ideally, a practicing Requirements Engineer would have extensive knowledge of multiple perspectives, including: Marketing, Psychology/Sociology, Object-Oriented Analysis, Structured Analysis, Participative Design, Human Factors (and HCI), Soft Systems, Quality and Formal Computer Science. She concentrates on a few of these approaches to RE and ignores the rest. I find this problematic on two counts: the claim that the book is meant to embrace the many areas not covered in conventional computer science literature is, inherently, unmet, and moreover, the reason for discarding the bulk of the methods is almost entirely on blamed on space.

The second chapter presents a discussion of the role of RE techniques and attempts to explore which techniques are needed. From the discussion a wish-list of seventy techniques are presented. The book focuses on Organisational, Group Session and Interactive Approaches.

The third chapter uses a case study to instantiate the problem, specifically the London Ambulance Service's Computer-Aided Dispatch System. Having discussed the case study, from the various stakeholders' points-of-view (except the customer), Macaulay presents the Soft Systems Methodology, ETHICS and Eason's solution to the problem of expectation failure: the author does not, however, discuss these methods with respect to the case study presented.

Chapter four addresses the Group Session approaches identified in chapter 2. Another case study is employed (A Study of a Multi-disciplinary Research and Development

Project) which has problems that be be attributed to lack of human communication. Joint Application Development, Quality Function Deployment and Cooperative Requirements Capture are presented as solutions to process breakdown. This chapter does not discuss these methods with respect to the case study, and moreover, it does not attempt any comparative analysis between the various methods.

Chapter five contains the last of the three approaches: Interactive. Macaulay describes six tools that foster interaction between designer and user: Designer-As-Apprentice, Focus Groups, Future Workshops, Prototyping, Cooperative Prototyping and Cooperative Evaluation. These methods are simply presented in the manner in which the original author wrote them, there is no real analysis or framework for rationalising usage of any one of these techniques.

Finally, chapter six presents a discussion of how one might construct a portfolio of RE techniques derived from various scenarios. This chapter is useful since it encapsulates the knowledge introduced in earlier chapters within the context of the customer-supplier relationship (and these scenarios mentioned above).

Overall, this book is exceptionally optimistic. It tends to focus on the utility of each method, without any real analysis or comparison with others, and therefore does not explore the true complexity of RE. While the case studies are interesting to read, the fact that the methods are not discussed in relation to the case is highly disappointing.

In conclusion, I would recommend this book as a source of reference for undergraduates studying requirements however would feel reluctant to suggest it to postgraduates requiring a little more critical analysis.

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 US DoD Data Analysis Centre for Software (DACS) list of resources on software engineering:

http://www.dacs.com/

Books

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

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/
(Files are called renl1, renl2, etc)

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

RequisitePro integrates requirements management, requirements traceability and change management capabilities to help software development teams manage the impact of fast-changing requirements. Requisite, Inc. has a Website at:
http://www.requirement.com:80/Requisite/Req20.htm

RE-Actions

Call for Committee Members

We are looking for two energetic volunteers to serve on the RESG committee as Events Officer and Associate Editor of this newsletter. In addition to regular committee duties (such attending committee meetings and attending and organising RESG events), the "job descriptions" of the two posts are as follows:

Events Officer

Maintain up-to-date information on all events organised by the group, including contact details of those responsible

for: (1) local arrangements, and/or (2) technical programme.
Gather information about forthcoming RESG events from committee members.
Act as point of contact for all queries (as a result of publicity) regarding events organised by the group.
Provide Publicity Officer with up-to-date information about forthcoming events.
Contribute to planning, organisation and running of SG events/meetings.
Attend meetings orgnised by the RESG (including events, committee meetings and AGM).

Associate Editor of Newsletter
Act as the "Roving Reporter" of the RESG, responsible for soliciting material for Requirenautics Quarterely (RQ).
Report on RESG events, by providing summary of proceedings of each meeting organised by the group.
Liaise with Editor-in-Chief, including the planning and implementation of new RQ departments/sections in RQ.
Contribute to planning, organisation and running of SG events/meetings.
Attend meetings orgnised by the RESG (including events, committee meetings and AGM)

Annual General Meeting

The annual general meeting of the RESG will be held at 13:30 on 11th June 1997, in room 418, Huxley Building, Imperial College, London. This is immediately prior to the afternoon meeting at on "Requirements for Off-the-Shelf Systems and Software"

The agenda is:
1. Apologies for absence
2. Minutes of previous AGM (1996)
3. Chairman's report
4. Treasurer's report
5. Election of Executive Committee Members
6. A.O.B.

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.

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

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 11 (July)	27th June 1997
Issue 12 (October)	26th September 1997
Issue 13 (January)	19th December 1997