BCS™

Requirements Quarterly

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

 http://www.resg.org.uk RQ39 (March 2006)

Contents

RE-Soundings

From the Editor

In this issue, we are pleased to be able to help to spread the word about a new course at the Open University related to RE. This important step should help create a new generation of software engineers well up to speed with the latest RE thinking.

We have a collaborative report on our Scenarios day (with thanks to Kathy Maitland, our very busy reporter), some thoughts on tacit requirements from Andrew Stone, and a story about an elephant.

There's a classic book review: Toulmin's The Uses of Argument (1958). If you ever need a rebuttal to the absurd notion prevalent in some software circles that anything written more than two years ago is obsolete, this is a perfect exemplar.

And of course we have news about the events we hope to see you at this year.

Ian Alexander,
Scenario Plus

Chairman´s Message

The RESG year got off to a great start with February's Scenarios Day. You can read the reports later in this issue. Scenarios events consistently act as a big draw, and no wonder given their power in discovering requirements and drawing out insights into stakeholders' concerns; an issue more troublesome than Michael Vaughan's knee. Starting the year with a bang of course risks anticlimax later in the year. Fortunately, I think we'll avoid this pitfall since the next event on Problem Frames (10th May) should be a cracker too. Held at the Open University, which has become a focus for research in problem frames, it should be a landmark event. Early booking advised.

May will be a busy month for many people but especially for our membership secretary Lucia Rapanotti who, in addition to organising our PF event, is taking the show on the road to Shanghai as co-organiser of the 2nd International Workshop on Applications and Advances in Problem Frames (IWAAPF'06) on 23rd May at ICSE. Two PF events in

one month; which to go to? Milton Keynes or Shanghai? Mid 20th or early 21st century landmark of urban planning? It's a tough call, but I hope to see you at the OU, and not just because it'll be cheaper to get to.

The next milestone will be REFSQ'06 to be held in the Grand Duchy of Luxembourg this year on the 5th and 6th of June. This is the year that REFSQ starts to evolve from a workshop into a full conference. One implication of this is that, for the first time, attendance, though space-limited (no Luxembourg gags, please), will not be confined to paper authors.

All the above only covers the first half of 2006. You can be sure there'll be plenty happening in the second half but, with the snow melting and the daffs about to appear, I can't bring myself to think that far ahead just yet. If like me, you enjoy anticipating the goodies, RE and otherwise, that Summer will bring, it looks like being a bumper year.

Pete Sawyer,
Computing Department, Lancaster University

RE-Treats

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

Forthcoming events organised by the RESG:

Problem Frames

10th May 2006, Berrill Theatre, Open University, Milton Keynes

This combined practitioner and researcher symposium introduces and investigates the Problem Frames approach - an approach to early life-cycle software engineering, which moves the engineer back to the problem to be solved rather than forward to the software and premature solution of a poorly defined problem.

This one-day event combines a morning tutorial on Problem Frames, delivered by Michael Jackson, and an afternoon panel of distinguished speakers.

The morning tutorial will cover Problem Frames foundation and techniques, including intellectual tools for representing and analysing software intensive problems. The afternoon session will provide a snapshot of current Problems Frames research and practice.

See the article "*RE at the Open University*" below for more details.

Registration is required: download the form from the RESG website at www.resg.org.uk.

Organiser: Lucia Rapanotti: L.Rapanotti @ open.ac.uk

AGM and Distinguished Speaker Event

July 2006, London

IEE/RESG Requirements Days

2nd October 2006, IEE, Savoy Place, London: 1-day Introduction to Requirements Course (see below). http://www.iee.org/Events/intro-req.cfm

3rd October 2006, IEE, Savoy Place, London: 8 talks covering all the essentials of requirements work, followed by a Banquet.

A Free Trip to RE'06 for a Student AND for Someone in Industry

The RESG would like to encourage its UK members to join in the major Requirements event of the year (and this newsletter has always urged you all to attend - Ed.)

RE'06 will be held in Minneapolis, USA next September. We know that seems far away, so we are offering substantial help for one research student and one person from industry to get to Minneapolis.

For students: please submit a poster of your current research to the RESG student representative.

For people in industry: please submit a letter explaining why you would like to attend RE'06.

The winning entry in each category will receive £500 (this may be supplemented if necessary for the student) in return for proof of travel expenses.

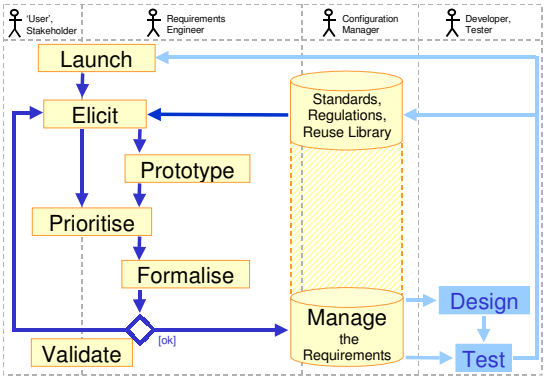
Student Contact: Zachos Konstantinos, City University, kzachos @ soi.city.ac.uk

Industry Contact: Dr Pete Sawyer, Computing Department, Lancaster University, sawyer @ comp.lancs.ac.uk (who will select the most suitable member of the committee to help you).

Introduction to Requirements Course

2nd October 2006, IEE, Savoy Place, London

presented by Ian Alexander



This seminar introduces requirements as a process of seven main steps, each supported by practical techniques, which are taught through group exercises.

The message of the course is that your organisation can take simple, cost-effective steps to do its requirements better.

http://www.iee.org/Events/intro-req.cfm

Early Aspects

October 2006, Lancaster

RE-Calls

Recent Calls for Papers and Participation

Open University - 2 New Courses

The Open University is offering two new postgraduate courses, that may be of interest to current and aspiring requirements practitioners.

Software Requirements for Business Systems (course code M883)

Managing the Software Enterprise (course code M882)

The courses run in part-time, distance learning format from May 2006 until October 2006 and start again in November 2006 and every six months thereafter. Either course can be studied individually for 15 academic credit points, or as part of a postgraduate diploma or MSc.

For further details see the Open University website at http://www.open.ac.uk.

See also the Open University article in *RE-Papers* below.

REFSQ

The 12th International Working Conference on Requirements Engineering: Foundation for Software Quality.

Now in its twelfth year, REFSQ is a highly interactive forum for researchers and practitioners to address the problem of ensuring software quality through requirements engineering (RE). The last decade has seen improvements in our understanding of RE with better practices supported by better techniques, methods and tools. Despite these successes, many quality-related problems remain, while new challenges

for RE constantly emerge. REFSQ seeks reports of innovative work in RE that contributes to the achievement of higher software quality.

In particular, we encourage people from the requirements engineering, software engineering, information systems, and embedded systems fields to present their approaches to RE. Contributions from related areas such as formal methods, systems engineering, economics and management and social sciences are especially welcome.

http://www.refsq.org

System Safety

The 1st IEE International Conference on System Safety

6-8 June 2006, The IEE, Savoy Place, London

System safety engineering (SSE) is the discipline concerned with achieving and assuring safety of systems, including their hardware, software and human elements. It encompasses, but is broader than, Functional Safety as it is concerned with hazards arising from physical causes, e.g. toxic materials and uncontrolled energy sources, as well as functional failures.

There are many challenges for SSE caused by changes in technology, increasing complexity and inter-working of systems, and reductions in the acceptability of risk to the public. SSE has to deal with these challenges – both societal and engineering – to ensure that deployed systems are acceptably safe, and remain so throughout their life.

RE' 06 Conference

11th - 15th September 2006, Minneapolis/St Pauls, Minnesota, USA. It's the major international RE event of the year.

We intend this year's conference to have an especially strong industrial theme, both from the programme and from being in an industrial heartland. Come along!

The website says:

Understanding the stakeholders' desires and needs

Requirements engineering has increasingly become a dominant activity in systems development—the more we can generate or outsource design and construction, the more we need requirements that adequately reflect the stakeholders' desires and needs.

The IEEE International Requirements Engineering conference is the premier requirements engineering conference, providing a forum for researchers, practitioners, educators, and students to present and discuss the most recent innovations, trends, experiences, and concerns in the field of requirements engineering.

There will be papers for everyone in the following categories:

- Technical Solution
- Scientific Evaluation

- Industrial Practice and Experience
- Vision Papers

The first 2 days of the conference are given over to

- **practical tutorials** which teach the essentials of requirements engineering, and
- **research workshops** in which all present join in collegiate discussion of their latest ideas and work.

http://www.ifi.unizh.ch/req/events/RE06/index.html

A Free Trip to RE'06 for a Student AND for Someone in Industry

The RESG would like to encourage its UK members to join in the major Requirements event of the year (and this newsletter has always urged you to attend - Ed.)

RE'06 will be held in Minneapolis, USA next September. We know that seems far away, so we are offering substantial help for one research student and one person from industry to get to Minneapolis.

For students: please submit a poster of your current research to the RESG student representative.

For people in industry: please write to the committee explaining why you would like to attend RE'06.

The winning entry in each category will receive £500 (this may be supplemented if necessary for the student) in return for proof of travel expenses.

We would also like to encourage you to submit papers and tutorial proposals to the conference. We would be happy to 'mentor' you through the writing process, though of course we cannot guarantee the outcome.

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

Industry Contact: Dr Pete Sawyer, Computing Department, Lancaster University, sawyer @ comp.lancs.ac.uk (who will select the most suitable member of the committee to help you).

RE-Readings

Reviews of recent Requirements Engineering events.

Scenarios, Stories and Use Cases Day

The RESG Scenarios, Stories and Use Cases day (to give it its full monicker) was held at City University on the 9th February 2006.

In the morning, Neil Maiden and Ian Alexander gave a tutorial on Scenarios, Stories, and Use Cases (with a copy of the book of that name for all participants).

In the afternoon, after a delicious buffet lunch for all, four speakers gave talks on their experiences with scenarios, ending with a Question Time-style panel.

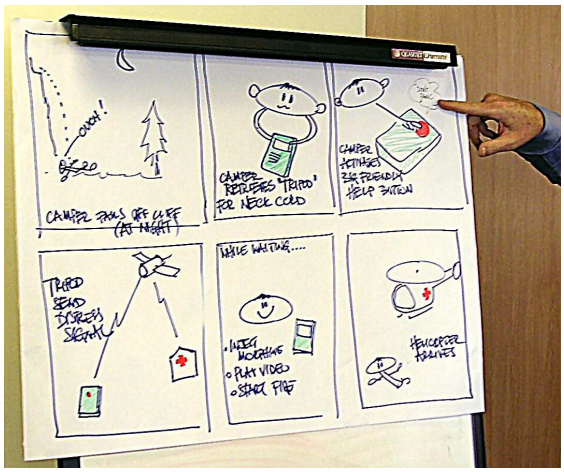
The meeting was very well attended: we closed the tutorial registrations at 40 to keep it interactive (though a few more than that turned up). There were well over 50 people present for the afternoon session. We are very grateful to **Monica Ferraro** for organising the registrations and local administration so smoothly.

The Morning Tutorial (a personal report by Kathleen Maitland)

Shall I tell you a story? The first part of the day was a two part tutorial. A very nice red-bearded man (Ian Alexander) started by explaining how stories could be used in the elicitation of requirements: not of computer systems, but instead of a torch-cum-emergency-aid (or as expressed in the slides a multi-function device).

In groups, we all explored the requirements of such a device – what could it be used for (well, my group designed it such that it incorporated a map to all the local pubs and restaurants in the area using GPS!), other groups stuck more to the given brief of trekkers or hill walkers needing an emergency device. Whilst, it was a fun exercise and James Robertson demonstrated his eminent artistic talents by drawing a helicopter and a person at the bottom of cliff.

The exercise made people think around a problem and explore ideas using story telling as a means of eliciting requirements and analysing them. It demonstrated the strengths and weakness of group dynamics and leadership, especially when dealing with blue sky ideas, which requires consensus if progress is to be made.



The 'Cliff Rescue' Scenario Team's Storyboard

The second part of the tutorial, led by an equally able story teller (Neil Maiden), took the group through the scenarios, goals and sequences of a bus passenger information system. Those of us who do not live in London and have not been on a bus in years required to have the concept explained to us. Again this

reflected a real life situation, where developers may not have experience of the system under development. Neil presented the ART-Scene Scenario Presenter, and more importantly its requirements checklist.

Again, in groups we gained practical experience of using the checklist in the requirements of the system through the use of events and alternative (ie Exception) events. The workshop especially aided developers who were new to requirements engineering by demonstrating tools and techniques to elicit and analyse system requirements in a friendly forum.

The Afternoon Seminar

Sebastian Uchitel (Imperial College) spoke on Scenarios, Goals, Architectures: using Models to explore requirements. Behaviour models allow you to identify wanted behaviours, and you can then use model-checking to validate the models. Unfortunately, such models are hard to build.

State machines are easier to make. You can use simple sequence charts (ITU or UML) to show intended behaviour, with a partial ordering of messages. You can create these from scenarios. You can turn those into a "label transition system" – a sequence of nodes joined by arcs – which can be represented graphically or in text. You can define things as deterministic or not as you need: all very simple. Then you can animate directly in the form of state transition diagrams (the active node is coloured red, etc); or you can add a simulated interface in the style of Harel and Marelly and play the scenarios through that.

But how do you synthesise architecture from scenarios when things get complicated? It seems not to scale: you can easily synthesise toy systems but real ones are another matter. That's because practitioners combine scenarios at different levels of abstraction, in different contexts, from different instances of architectures, smoothly and with no apparent joins. That isn't easy.

The team at Imperial conjectures that you can systematically relate scenarios to architecture. The Darwin architecture description language lets you describe components and bolt them together. UML 2 is only just coming round to recognising this as an important thing to do. Philips are using Koala, based on Darwin, to describe the architectural design of a television set (as boxes eg Tuner) linked by interfaces.

For instance, when you change channel on your TV you don't want to see a snow pattern while the TV changes the frequency to the new channel. So the tuner needs to tell the video to blank the screen until the new frequency has been tuned. Then the tuner needs to tell the video to unblank the image. The scenario thus explains (as a sequence diagram) how the architecture works.

More complex architectures can then be built up by combining components and drawing bigger message sequence charts. You can make models for types of component by generalising component behaviour, and representing communication through (generalisable) plug-in ports rather than directly from one component to another.

Quoting Michael Jackson misquoting Julius Caesar's '*Divide et Impera*' ("Divide and Conquer") principle:

"Divide to Conquer, Reunite to Rule",

you can then relate architecture to scenarios. Uchitel demonstrated how specification code, scenarios, statecharts and then automatic checking all fitted together for the TV example, bravely running the compiler in front of us and showing that the safety check promptly discovered a trace to deadlock as the combination of a TV switch and a Fork (to 2 video displays) turned up some unexpected behaviour. The tools thus quickly start to pay for themselves.

Goals and scenarios are complementary: state vs events, declarative vs operational, general vs example, what and why vs how. You can deal with all this formally using Fluents, assertions in a time-based logic. They provide a basis for checking assertions in state-based models: you can check operational (scenario) descriptions against goals. These can be animated either via statecharts, or in terms of interactions of components shown as boxes. You can see directly whether assertions – requirements – are met.

For instance, an assertion such as "you can't be sent a message if you are logged out" can be seen to be met, while if you are using a web server - browser architecture, the requirement "you can't view a message if you are logged out" is seen to be violated: the server can't black out the browser screen. Thus the weaker requirement is more appropriate for the environment, no matter how intuitive the stronger requirement seemed to be.

Thus, architecture, goals, models, scenarios and analysis are all starting to come together in a formal way.

Helen Sharp (The Open University) spoke on User Stories for Agile Software Development. This is almost the opposite end of the scenario spectrum from the Imperial College work. Sharp often works with practitioners, studying mature Agile teams who are developing Java programs.

Extreme Programming (XP) uses iterations of 1-3 weeks, like the other Agile methods that have followed it (see http://www.agilealliance.org). XP in particular focuses on the program code. Other things like models can be used, but code is king. XP has 12 practices such as pair programming, a 40-hour week, small releases, simple design, and minimal documentation. XP has an explicit value system, honouring communication, feedback, simplicity, courage and respect. Sharp denies being an Agile advocate, but she put its case eloquently nonetheless.

XP works with User Stories: descriptions of things people want to do; units of customer-visible functionality; promises of conversations; fundamental

units of development in XP; and usually written on one side of a 3x5" index card.

The three C's of stories are Cards, Conversation and Confirmation. Cards only contain some of the information representing a requirement. Conversations exchange thoughts and feelings verbally but supplemented with documents. Confirmation is via the acceptance test seen by the customer at the end of each iteration.

A card has a title, with a short story in the form:

> "as a <role>, I want <behaviour> so that <benefit> is provided."

Cards are often handwritten, signed and dated to show ownership. Clearly this is much like a scenario, if highly compressed, and like a use case indicates a requirement (which is unlikely ever to be documented as a shall-statement). Cards may be coloured to show status; other colours can be used to task specific pairs of programmers; bugs are written on red cards, etc. Cards are the focal point of development. The book says that stories come from "users", but in practice developers often write them too.

A story is a small chunk: half a day up to 10 days of development work. When done, they are formally archived.

Cards are kept on "The Wall", or in the hideous neo-Blairite jargon "Informative Workspaces". The wall can be a real wall, a flipchart easel, or a glass office-divider panel. Progress is directly visible to programmers, managers and customers – if all the cards are at bottom left, the iteration is complete.

There are challenges: the process relies heavily on communication and places a heavy load on the customer – stating requirements, prioritising, assessing, commenting (repeatedly). It isn't easy to get started. Writing good "useful" stories is difficult; discipline is essential. There has to be common ground – people are carefully chosen to fit in with the team: social ability counts for more than intellectual. The customer-developer relationship can be very tense, especially if there's a difference in culture between customer and developer, as is likely. So, perhaps the "extreme" method isn't as natural and intuitive and informal as it was talked up to be. But it can work and be highly productive.

Anthony Kesterton (IBM Rational) started his talk on Use Case-based Software Development with the joke:

> "'Squirrels are just rats with good PR' – and Use Cases are just requirements with good PR"

(laughter). Most people in the audience had heard of the Rational Unified Process (RUP) and several had actually used it, too – at least, they put their hands up when Kesterton asked them.

The UML specifies the diagrams, while the RUP describes the life-cycle, but neither say how the use cases are actually to be written – though this is highly important for development.

Kesterton described a 4-layered pyramid from a few needs at the apex to several features, more requirements and use cases, and then a large base to the pyramid for design, test cases, and documentation.

Use Cases paint the big picture, and a use case diagram paints a big picture of them: who (which actor) gets value from which use cases.

"These are very useful in business, as long as you don't mention Use Case, UML, or Actor", Kesterton quipped. Then we move to the written word, he said.

Once the use cases have been written, the RUP says they must be analysed. You study each use case, discover objects/classes to implement it, figure out what each such class must do, and determine the object interactions to achieve that behaviour. Finally, the design must be documented as a set of UML diagrams. All that stuff is called the "Use Case Realisation" – it doesn't belong in the use case itself. In long-lived systems, the use cases remain quite stable although their realisations may evolve. The realisation is a dashed-line oval, a UML "collaboration" in fact.

Another aspect of use cases is "user experience design". You decide what goes into the user interface (UI) before you decide how it looks. Of course you do that from the use cases, ie there is a progress from scenarios to abstract UI to concrete UI.

Use cases also jump-start testing. Testers are delighted to receive clearly-written scenarios, claimed Kesterton. Each path through a use case is a scenario, the basis of a test case.

Non-functional requirements are also vital (as is architecture), said Kesterton. But use cases are a well-established and effective technique for managing functional requirements. The use case realisation follows on naturally, with excellent traceability back to the use cases which drive the development and document its rationale. This becomes even more important when the coding is outsourced.

In answer to a question, Kesterton said that the intent of use cases and user stories is similar, but the implementation is different.

Similarly, Kesterton asserted that you could systematically replace "System" with "Business" and you'd end up with a business-level use case model which would also make excellent sense.

Stuart Burdett (Defence Science & Technology Laboratory, DSTL) spoke on Scenario-based requirement capture for the British Army's FRES programme. This is a £6.5 Bn development to replace a variety of old tanks, personnel carriers, reconnaisance and engineer vehicles, and so on. The Future Rapid Effects System (FRES) is based on an analysis done about 8 years ago, which showed we were good at putting in light forces (as in Sierra Leone) and were designed to cope with heavy forces

(from the Cold War), but were missing anything in between. Such lighter forces would (in the scenario) go into an area quickly, guard flanks, and so on, leaving the heavier work to heavier forces. FRES has to be mobile, powerful enough to intervene and stabilise, and set conditions for heavy forces. Its effects have to include infantry transport, reconnaissance, control of indirect fire, casualty treatment and evacuation, supply of materiel, and so on.

Now, if we are not going to specify vehicles directly (ie to design the system) then we need to define the required capability as "user requirements" as well as a concept of employment (ConUse) which is also a set of scenarios, telling the story of how FRES would be used.

What are these documents for? To get industry engaged early; to support the business case by addressing key uncertainties in budget, solution feasibility, scope and operational interfaces, and finally to give initial guidance on the optimum set of vehicle types and numbers, ie on the eventual design. Clearly, you can't get a definite price until industry has looked at what it has to build and worked out what it would cost.

All this replaces a large and cumbersome process of developing requirements by replacement / upgrade engineering: you assume the needs are much the same but you do things a little better with the new equipment. Such an approach did not look too closely at interfaces, and did much of the work implicitly.

The army has always used scenarios. The FRES team similarly had a background of doing analysis with fault trees, FMECA, etc. The URD was structured around scenarios, with inputs from the army's standard Task List (the METL(L)), and the high-level need expressed in military policy. Scenarios = theatre + missions + forces (on both sides) + behaviours (soldiers are constrained by strict rules of engagement nowadays).

For each scenario there was a Scheme of Manoeuvre, from preparation through to reorganisation afterwards. A small team of about 20 stakeholders played through the scenarios in sessions. The stakeholders were of 3 types: playing FRES roles directly; playing interfaces (such as artillery, not part of FRES itself); and liaison with people in other 'lines of development' such as training, people, information, doctrine, organisation, infrastructure and logistics – as well as equipment, FRES' own province. Responsibilities included thinking about what support was needed across interfaces, and what FRES would not have to do. Each scenario workshop went on for 2 days. The players (officers) were good at talking about tasks: "they do that all the time when they give orders" quipped Burdett.

The outputs were documented by capturing Requirements (what), Measures of Effectiveness (how well), and Justification (why) for each step of each scenario. Each workshop collected a pile of documents: 2700 requirements came out (with some duplication). Triage on those split them into 'good to go'; 'treatable' by categorising , clarifying, completing and making testable; and some that were returned to sender. Ian Alexander helped organise the data in a customised DOORS database. 630 good requirements came out of the process.

Ongoing developments include a joint MoD-development house process, and a government-led requirement development. The 16 weeks of activity led to a broad, high-quality stakeholder engagement; all involved felt they'd had a pivotal role in FRES. Interfaces were explicitly identified and quantified.

Pinning requirements to scenarios created a robust justification of the requirements. Using military language and structures meant a short learning time for stakeholders. And there was much easier differentiation of user and system requirements – talking about scenarios not equipment, and avoiding all that language like "the user requires the ability to …" which always used to confuse people. The interactive scenario-based elicitation approach was extremely effective. [Requirements] Tools are beneficial as long as they don't take over, remarked Burdett.

The Afternoon Seminar (a personal report by Kathleen Maitland)

In contrast to the morning tutorial, the afternoon consisted of four presentations.

The most controversial presentation was given by Dr. Helen Sharp on User Stories in Agile Software Development. Agile systems development revolves around the use of index cards and pair programmers. Index cards contained brief descriptions of the work that was required and they were prioritised and stuck on a surface for the developers to basically get on and do. The only documentation that exists is within the code, and this also helps to reduce the time taken to implement changes. Whilst the method obviously led to rapid systems development, it was not clear how the process started, or the development and impact on the systems architecture. The Agile software development approach is gaining in popularity: a delegate from BT said that they were using the approach to evolve current software systems.

Dr. Sebastian Uchitel from Imperial College gave an interesting presentation on his graphical animated requirements verification tool. Based on the premise that the use of scenarios is a popular method used in the definition of requirements, the tool enables developers to analyse system behaviour through a given specified software architecture. Uchitel demonstrated his tool through a scenario based on a TV tuner and switches to demonstrate instances of behaviour related to a given software architecture. Message sequence charts were used to express and analyse the systems behaviour for systems architecture.

What was interesting about this presentation was that we were seeing a graphical representation of the relationship between software architecture and scenarios, and in particular the impact that architecture can have on a systems ability to meet system goals and requirements. This was in complete contrast to presentation on Agile systems development where the question on the impact of the approach on systems architecture could not be answered.

The other two talks were more general in content. Anthony Kesterton from IBM Rational gave an interesting presentation on Use Case-Based Software Development. The popular technique of Use Case driven development was described and again was received well by those new to requirements engineering. Finally, Stuart Burdett from DSTL gave an interesting presentation of the use of scenarios in the development of military systems.

The Concluding Panel Discussion

The day concluded with a panel and questions from the floor. Again the main topic of discussion was the use of the Agile approach to systems development. A full RESG Event is clearly needed on this subject in the context of requirements engineering and system architecture.

Burdett said he was absolutely fascinated by the XP approach, which focussed directly on what people had to do. Perhaps this is an indication of the value of RESG meetings – people from different backgrounds meet, hear ideas they have never come across, and reflect on what they could learn from each other. I've often felt that the discussions over lunch and coffee are the best bits of many conferences. Similarly, simply having several expert speakers together, focussing on a single topic, is extremely valuable.

The Panel: Uchitel, Kesterton, Sharp and Burdett

Burdett's remarks on the choice between Replacement Engineering and what the RESG thinks of as (good, proper, canonical) Requirements Engineering are pertinent. We, like most people working within a 'paradigm", norma lly don't give a thought to why we're doing things as we do, nor indeed whether there might be an alternative. There is one, of course: not Agile – which is totally within the find-out-the-user's-needs-and-test-to-check-you-satisfied-them tradition – but simple Replacement Engineering. How d'you make the 1953 model saloon car? You lengthen the fins and add a bit more chrome to the 1952 model, of course. Silly question: within a paradigm of a new model every year, there's no doubt in anyone's mind how you go about developing next year's offering.

When you are sure that the new product has to be the same as before, but to look a little newer or to have a few new features, then there is no sense in starting from scratch.

When you aren't sure… aye, there's the rub: Replacement Engineering is risky because it implicitly assumes that next year will be like this one, and it never is, quite. Small increments of change USUALLY don't demand large jumps in design. But the next increment might, as Catastrophe Theory and Chaos Theory from mathematics show only too clearly.

Requirements Engineering is time-consuming and costly. Often it may not add much to the result: but when conditions are unknown and unstable, requirements work dramatically reduces risk, and its RoI (Return on Investment) becomes immeasurably large. Replacement Engineering is far cheaper… until your business falls over the cusp of the curve, down to the bottom of the Catastrophe Theory cliff.

The Future of Systems Integration

INCOSE Midlands Group, Loughborough University, 28th February 2006

Report by Kathleen Maitland

As a member of RESG, I am interested in the synergy and the difference between INCOSE events and RESG events. Primarily the event at Loughborough University was a showcase of Tool Vendor Products. My expectation was that this event would be similar to the one held by the RESG, but it was not. As a requirements engineer, I think of systems in terms of higher level concept of software packages in conjunction with organisational structures. I look at integrated systems from a business perspective. This seemed to be in contrast to the systems view that I have heard articulated at INCOSE meeting, which seemed to be closer to the code level. Taking this difference into consideration, and taking into account that I was once a software engineering who wrote command and control software I found the day very interesting.

In the morning there were six presentations by tool vendors. From the requirements perspective every one of them linked with Telelogic's DOORS requirements tool. It was therefore interesting that the Telelogic speaker – Jon Chard – presented their TAU design case tool. Again, the selection of a tool that is used at design/implementation stage of systems development demonstrated the difference between INCOSE and RESG. The presentation was vastly different to the talk presented at RESG's tool vendors' event. The focus of

the INCOSE presentation revolved around design and standards, especially MODAF and DODAF, which are Ministry of Defence/Department of Defense architecture framework standards, while the compliance with UML 2.0 and SysML.

Other talks were given by i-Logix who presented their Rhapsody Tool, which models behaviour and validates designs through simulations that employ back-animation with sequence diagrams. National Instruments gave a talk on LabVIEW. This a tool that developers can use to analyze real-world signals. Again, this tool is used at the design level and uses block diagrams to model processes in line with SysML. Artisan Software Tools presented their tool which models systems architecture. Based on the Zachman Enterprise Architecture, model items can be defined once in the model and can be referenced on any diagram as either a symbol or rich text reference. This can ensuring architectural consistency and completeness. IBM presented their Product Lifecycle Management (PLM) tool. Finally from the heavyweights of the CASE Tool industry to a relatively small company, BIW Techologies who presented Planweaver, a tool based on an optimised matrix used for planning, managing and integrating the different phases involved in complex processes. The tool delivers a programme or delivery schedule by linking the PlanWeaver to a project programming software application such as MS Project, Primavera or PowerProject. It appeared quite a simple but useful tool, in comparison to other presented.

I learnt a lot from attending this well organised and interesting presentation of Case Tools by the INCOSE Midlands Group. It was very different to RESG events. I would recommend that members who are interested in particular at the design level of systems development should look at what INCOSE are doing.

RE-Papers

Stuck Altimeters, or What Does your Mother Look Like?

by Andrew Stone, University of Lancaster

Unqualified theories of language acquisition abound; if humans have an innate knowledge and understanding of language as part of our internal meta-knowledge, as Chomsky insists we do, why are we unable to explain some of the fundamental things about ourselves? More importantly, why is my altimeter stuck?

For a species that asserted its dominance by its ability to form complex social relationships which depend strongly on vocal communication, there appears to be a problem with the languages that we have developed. Some concepts appear to be more difficult to explicate than others, and it isn't hard to find native speakers of any language lost for words when attempting to communicate important concepts. Often the best effort to communicate complex topics and qualities results in the formation of domain-restricted language, although this alone does not guarantee clarity of expression. The necessity for languages to be writable further compounds problems of ambiguity. Such symbolic representations of natural phenomena leave a lot to be desired; "There is neither pine nor apple in a pineapple", a Bahrainian friend once commented to me. English has become the de facto language of technology, and therefore to some extent, the *lingua franca* of requirements and software engineering, and it is with this burden that we must attempt to express problem domains in an unambiguous manner.

Does the limit to our ability to extract information from stakeholders depend solely on the restraints placed upon us by language? Polanyi thinks not (see Ian Alexander's review of Polanyi's book in RQ 25, Jan 2002). Michael Polanyi has emerged as the king of tacit knowledge; the clarity of his explanations are without doubt the result of his training in the physical sciences. Tacit is a word that did make it into English from Latin, and means "to be quiet". When applied to knowledge Polanyi offers us this compact definition, "Tacit Knowledge is knowing more than you can tell". The classic example of such tacit knowledge is the question "Can you recognise your mother's face?" While most people can, they are always unable to explain how they know the face in question is their mother. Fundamental questions such as "which algorithm did you use to recognise your mother's face?" or "what internal representation of your mother's face do you keep in your mind?" are unanswerable, and therefore prevent any further insight into the processes involved.

Similarly, stakeholders are likely to possess at least part of the knowledge about the operation of their problem domain in a tacit fashion. This in turn prevents stakeholders from explicating clearly the necessity of required functionality, or more critically, explaining the precise operation of their role within the organisation.

There are other effects the presence of tacit may have on requirements engineering in general; Anthony Finkelstein, at the 2004 RESG AGM, listed as one of the remaining problems in RE the inability of requirements engineers to define a system boundary. He further commented that ad hoc decisions are then made as to the required scope of the requirements elicitation process. If many of the system processes are held as tacit knowledge in the domain then defining a reasonable system boundary can never take place, unless this tacit knowledge is somehow exposed.

Identification of what might be tacit in a problem domain is clearly an important focus for research, but it appears to involve quantifying the unquantifiable.

Are there other kinds of knowledge that behave in a tacit-like manner, yet are able to be quantified? David Parnas relayed an amusing example of such knowledge in his RESG meeting in May last year. For some time he had been wondering about the origin of a requirement in the specification of a military aircraft. This aircraft, the A7, had two altimeters which the pilot could choose to view.

A traditional altimeter

If one failed, then the other altimeter would be used and the pilot would be alerted that one of them had failed. The next question is of course, what happens if they both fail? The requirement specified that a fixed altitude should be displayed on the altimeter of 3750ft. Parnas questioned this bizarre requirement, but was informed by several stakeholders that it was indeed correct. No explanations about the origin of the requirement were forthcoming until Parnas encountered a pilot, who explained that this was the mean cruising altitude of the aircraft, and therefore would most probably be the correct altitude at the point of failure! This non-intuitive requirement was behaving as if it were tacit knowledge embedded within the problem domain: difficult to identity, difficult to justify. This also serves as a demonstration that instances of tacit or almost-tacit knowledge may be exposed at any point in the system's development life cycle. By identifying knowledge that appears to be tacit-like and actively seeking its source some degree of ambiguity can be mitigated for requirements based on such knowledge.

To all those working on requirements specifications that feature "stuck altimeters" and the face of your mother, my condolences. Take some heart though, it's not your fault – neither is it the fault of our innately developed language. We appear to just be made that way.

© Andrew Stone 2006

RE at The Open University

Requirements Engineering lies at the heart of research in the Centre for Research in Computing at the Open University. Among members of its Requirements Engineering and Design (READ) group are prominent members of the RE academic community and well-known contributors to RESG, including its patron, Michael Jackson and its immediate past Chair, Bashar Nuseibeh. The lively research programme under READ spans a wide range of RE activities, from social, business, and organisational processes of eliciting requirements to the technical and mathematically grounded techniques for the specification and analysis of those requirements.

The group has a growing international reputation as the home of Problem Frames research, under the guidance of Michael Jackson. Problem Frames are a promising approach to early life-cycle software engineering. Their focus moves the engineer back to the problem to be solved rather than forward to the software and premature solution of a poorly defined problem. The influence of the Problem Frames approach is spreading in the fields of domain modelling, business process modelling, requirements engineering, and software architecture, as well as in software engineering in general. The 2nd International Workshop on Advances and Applications of Problem Frames (IWAAPF'06) will take place on May 23rd, co-located in Shanghai with ICSE'06, to continue the success of IWAAPF'04, held at ICSE'04. READ members Jon G. Hall and Lucia Rapanotti are among the instigators and organisers of both workshops.

May 10th sees the RESG Problem Frames Day, held at the Open University and with contributions from READ. The day will include a tutorial on the approach and snapshots of related research and practice. This will be an excellent opportunity for RESG members to find out more about Problem Frames from Michael Jackson himself.

Last but not least, May will also see the launch of the Open University flagship post-graduate course 'Software Requirements for Business Systems". The course aims to provide the student with both a theoretical and practical knowledge of Requirements Engineering (RE). It examines a disciplined approach to the process of eliciting, analysing, communicating and agreeing requirements as the essential first step in the development of software - or any other artefact for that matter! The course draws extensively from best research and practice in RE and adopts as its primary source the best-seller "Mastering the Requirements Process" by Suzanne and James Robertson. Students will be able to get their hands dirty on real-world case studies and make use of a software tool. They will also have access to the wisdom of experts in the RE field including Ian Alexander, Michael Jackson, Peri Loucopoulos, John Mylopoulos, and the Robertsons themselves, delivered to their doorstep in DVD format.

Further info:

You can find out more about READ and its members at http://computing-research.open.ac.uk/read/

A special issue on Problem Frames - a collection of the best contributions from IWAAPF'04 - was published in Elsevier's International Journal of Information and Software Technology (volume 47, number 14, 2005). The issue includes a comprehensive survey of Problem Frames research and practice, as well as an article by Michael Jackson on Problem Frames in their role as an engineering tool.

Information on the course 'Software Requirements for Business Systems" (course code M883) can be found from the Open University website at http://www.open.ac.uk.

Is Requirements Engineering a Systems or a Software Discipline?

In RQ38, Michael Jackson said

> "As members of the BCS RESG we regard our métier as a part of software engineering, and we call it requirements engineering."

This is both logical – the BCS is after all a *Computer* society – and natural, as to many people requirements are essentially about software.

Yet to a systems engineer, someone who is interested in building anything from a satellite system (ground stations, communications, a satellite, and a network of people to control and operate it) to a handheld electronic device (electronics, software, user interface, and so on), software is just a component of a system.

But software has remarkable and unusual properties, and it is unique in many ways. It is extraordinarily pervasive; it is already essentially ubiquitous in new systems, and is becoming so in every part of our lives, both at work and in our leisure. It has transformed the nature of work in an incredibly short time. Whether software has made working life any more efficient, given the average office worker's endless round of meetings, is another matter.

Already it is hard for the computer generation to imagine a time when people did not have telephones and cameras and diaries and alarm clocks and transistor radios and gramophones and typewriters and rolodex card indexes all rolled into one and somehow compressed to fit into people's pockets. (The things will be making tea and hot buttered toast next.)

But only a moment ago, people never knew that they wanted or needed or even conceivably could carry all those things around with them. It seemed perfectly fine to leave all those machines at home or in the office when we wanted a bit of peace and quiet; but as Aldous Huxley presciently wrote back in the 1930s, we are soon to be surrounded by a stream of impulses, and quiet will become impossible.

A couple of years ago, one of my nephews asked me how I had managed to get to my tutorials at university (we went to the same one) without a mobile phone or email. I could see that he was making a serious effort of imagination to conjure up such a primitive situation.

> "Simple",

I said, starting to feel like some sort of dinosaur.

> "I arranged with the tutor at the start of term which day of the week we'd meet, and each week I delivered my essay by bicycle to his or her office the day before, and then I cycled to the tutorial."

> "Yes, I can see that that *would* work…"

he muttered to himself, obviously wrestling with what to him was a seriously Jurassic Weltanschauung, and slowly coming round to the crux of the problem:

> "But what did you do if anything went wrong?"

I began to see where he was coming from: a world of instantly-created software constructs, all pointers and dynamically linked lists with nothing substantial underneath the whole edifice.

> "Usually nothing went wrong",

said the Dimetrodon,

> "but if the tutor was ill he'd let me know."

This explanation threw my nephew into something like confusion.

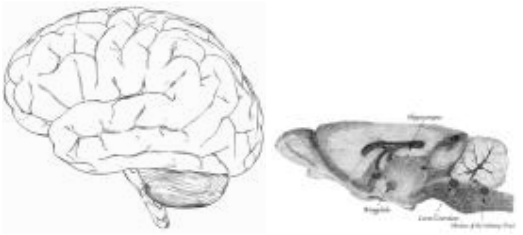
> "But how could he do that if you didn't have a mobile phone?"

asked the Modern Man, clearly convinced that this was the decisive argument. After all, if everything was handled dynamically, then you could dynamically handle every challenge, probably.

> "He just left a message with the college Porter."

said the Pterodactyl, wondering to himself how he had managed to live through two such extremely different Epochs, and leaving his nephew with the realisation that, indeed, carrier pigeons, stage coaches, or cuneiform tablets strapped to crossbow bolts might be ways of conveying information in the absence of microwave telephony.

And perhaps Evolution is the answer to the Systems or Software controversy too. The brain is just an organ like any other … except that it isn't. It organises and controls everything else. The human brain is just like a monkey brain … except that the cerebral cortex has expanded, ridiculously, enormously, ballooning out and dwarfing all the older parts, covering them with its bizarrely wreathed and contorted bulk, folded and refolded into great wrinkles to fit its huge surface into a small compass.



Human and Rat Brains in side view (not to scale), showing the enormously expanded human cortex. The rat's cortex is the small bump on its left (ie front)

So, from one perspective, the cortex is "just a region of the mammalian brain", which is indeed what it evolved from. But from another, it is THE Cerebral Cortex, the highest organ, the crucial and central and controlling element.

Software, too, is from one perspective "just a system component", and indeed it is: you have to bash the metal and etch the silicon and code the software. But even in a "hardware" component like a car, the software now represents up to half the cost of development. On a system project a dozen years ago I recall the software manager saying

> "You know, they just don't understand software."

and he didn't mean that the system people didn't know how to write code (that much was obvious). It was that they didn't know how to handle it. You had streams of work for the different disciplines: User Interface, Quality, Safety, Configuration Management and so on. You had subsystems: propulsion, communication, payload, … and a little box on the diagram called "software". Only, it didn't fit in there at all. It sneaked into every other subsystem too. And it joined all the other bits together. That was what the software manager meant. It was all very inconvenient.

As for requirements, how do they fit? The old hardware systems clearly had requirements, and they still do. Except, everything has changed. The obvious complexity of software leads the young to imagine that RE was invented just for it. Worse, quite often with business software, the rest of the system (computers, cables, um, that's about it) is quite trivial: it isn't going to be developed afresh or even changed because of the software, so the software IS the system to all intents and purposes.

Like the Cerebral Cortex that almost completely conceals and supercedes the rest of the brain and millions of years of evolution on which it still depends, software has become THE part of the system that many people need to concern themselves with. In fact, it's only in quite lucky and special circumstances, like robotics (or "autonomous systems", see RQ38) that people really spend much time on all the other exciting and strange parts of the System, like vision and walking: and even then, software takes up much of their energy.

Is software a system component? Of course it is. Do all systems contain other things as well as software? Ye-e-es, they do-o-o, BUT… somehow the system perspective isn't always all that helpful.

So, is requirements work part of systems engineering, or part of software engineering? Admit it, it's a silly question.

RE-flections

Not Near So Much Like a Requirement

Much more rational, I dare say

> [Miss Bingley] "I should like balls infinitely better," she replied, "if they were carried on in a different manner; but there is something insufferably tedious in the usual process of such a meeting. It would surely be much more rational if conversation instead of dancing were made the order of the day."
>
> [Her Brother] "Much more rational, my dear Caroline, I dare say, but it would not be near so much like a ball."
>
> Jane Austen, *Pride and Prejudice (1813)*, Chapter 11.

It would indeed be infinitely better if users could write down their requirements formally, so that there would be no possibility of error in their interpretation and translation into hardware and software. Alas, users show no signs of dropping their usual habits of vagueness, ambiguity, talking in natural language, and for that matter of changing their minds.

Formal specification would be much more rational, I dare say (my dear professor), but it would not be near so much like a communication with users.

Five Blind Men Describing an Elephant

Once upon a time, a circus with an Elephant came into a small town that had never seen a real live pachyderm before.

In the town was a house where five blind men lived. Their housekeeper came back from market in high excitement, full of news of the strange beast. At once the five men went to the circus enclosure, and asked to be allowed to feel for themselves what an Elephant was like. The circus owner was a kindly fellow, and he agreed to their request, as long as only one of them at a time went up to the animal, so as not to upset it and cause an accident.

Adam spent a few minutes with the beast, and returned, announcing that it was a long, thin, leathery, rather stiff rope-like animal with a tuft of bristles on its nose. He seemed a little disappointed at having found the Elephant to be such an insignificant little creature, but made the most of reporting his discoveries.

Bert went into the paddock next, and soon came back, saying that Adam was quite wrong. It was a huge, passive, cylindrical thing rather like a tree; a colossal column of rough, rather scaly-skinned flesh that from time to time twitched a little.

A naughty, lively, rubbery thing

Charlie's turn came, and he returned after only a minute or two, beaming with pleasure at his experience. It was nothing like the nonsense the other two had reported! The Elephant was a naughty, lively, rubbery thing that could grasp your hand like a monkey, or rub your head, or nuzzle your chest. What was more, it grunted or even roared a little. Charlie did not admit that he had been somewhat frightened by this active and surprising beast, which was why he had returned so soon.

David stalked into the paddock, determined to prove the others - who were plainly totally confused, or simply making up stories to impress - completely wrong. He was not disappointed. At once he felt a broad, flattened, warm papery thing, that shuddered and flapped like a flounder: the most extraordinary land animal he had ever heard tell of.

Eddie could tell that all his friends had been supping ale instead of going to feel the real Elephant. Perhaps they were all too terrified to find out for themselves! So he boldly went up to the unknown animal, and, holding out his arms stiffly, braced himself for whatever he might experience. It was not at all like a rope, or a tree, or a rubber monkey, or a papery flounder. It was cold, hard, smooth, polished, and almost cylindrical; though Eddie, who prided himself on accurate observation, noted carefully that in fact it was slightly curved, and slightly conical as he slid his hands bravely along its considerable length. This he reported to the assembled company, who were universally scornful of his account.

So, next time you meet someone who insists that requirements work is *all about* scenarios, or *really only* about listing shall-statements, or *essentially* just a matter of traceability, or *in truth just* a problem of defining acceptance criteria to facilitate testing, or when push comes to shove *best dealt with every time* by analysing the goals..... just remember that the beast may need to be described from more than one point of view.

And that goes for the requirements you're trying to define as well: never believe stakeholders' accounts – even when they're all true!

Proverb

After all that about ropy and cylindrical elephants (and rational ways of defining requirements), perhaps we need a Shakespearean reminder:

There are more things in heaven and earth, Horatio,
Than are dreamt of in your philosophy.
Hamlet (I, v, 166-167)

Or to put it another way:

To the man who only has a hammer,
every problem looks much like a nail.

RE-Publications

The Uses of Argument, by Stephen Toulmin

Cambridge University Press, 1958

This short and readable philosophy book has been reprinted at least 15 times since its first publication nearly half a century ago. It has influenced people far outside the quiet realms of philosophy and logic -- linguists, lawyers, and computer scientists among

them. The book has indeed founded almost single-handed the field of visual argumentation (see the review of *Visualizing Argumentation* by Kirschner et al in RQ 29).

Numerous commercial methods and tools attempt to model arguments broadly in the manner that Toulmin suggests, the most important from the point of view of systems engineering being safety case argumentation methods such as Adelard's ASCAD and the University of York's GSN.

The crux of Toulmin's argument is that Aristotelian logic, with its neat mathematical syllogisms, simply doesn't apply to the real world. When arguments are 'substantial', i.e. a substantial leap is required to get from facts to a conclusion, rather than when the conclusion is logically entailed by the premisses, then formal logic just doesn't help.

> "All men are mortal;
> Socrates is a man;
> so, Socrates is mortal"

is all very well in logic school and the maths class -- but outside, you actually never know enough to state the rule, and you usually don't have perfect data either. (Even the word 'fact' means 'a made thing', from the Latin *facere/factum*: a fact is something that you have established by observation, experiment, or argument. It is not a given in nature.)

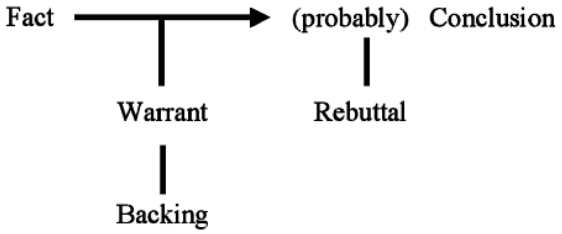
So if we can't have logic, how can we ever know anything? You can feel the discomfort; thousands of unbearably clever theses have argued that everything is relative, uncertain, and even worse, that the feeling that we know anything is just that, a feeling that we delude ourselves with. You know the sort of stuff -- everything is just a social construct, reality is not observable, and claims to knowledge are merely assertions of power -- the sort of thing that got philosophy a bad name.

Toulmin comes to the rescue, showing that indeed logic can't save us, but that instead, there are perfectly serviceable ways of arguing and justifying our conclusions that stop short of attaining certainty. We use them every day, and so do lawyers in court. In fact, every profession has its own forms of argument, and Toulmin ends the book by suggesting that there may be many schemata or patterns of possible argument:

> they should .. scrutinise the logical history, structure and *modus operandi* of the sciences using the eye of a naturalist, without preconceptions or prejudices imported from outside. This will mean seeing and describing the arguments in each field as they are, recognising how they work...

This may be a terrible shock to logicians who entertain claims of universal validity, but it ought to be a relief to the rest of us. Engineering arguments need to be put together in a way that makes clear how confident we can be in our conclusions. Do we know that signalling system is safe, or are we just hoping for the best?

Toulmin proposes a simple and really quite general scheme for representing arguments, and it has turned out to be very suitable for machine representation and for diagramming. A Conclusion is reached, more or less definitely, from some data, by way of a Warrant for the jump from the one to the other.



The warrant is necessary precisely because formal logic can't be applied. The fact that the person I'm looking at has fair skin, red hair and freckles absolutely does not entail the conclusion that they'll easily get sunburnt -- it isn't a tautology. What it is, is a practically-certain conclusion, given some knowledge of physiology.

Our warrant is something like 'People with red hair, fair skin and freckles are adapted to high northern latitudes where sunlight is scarce. They usually burn rapidly in the sun.' We could be deceived -- maybe the person has been beautifully made-up for a film shot -- but otherwise we are pretty sure we are right.

All these 'practically-certain's and 'pretty sure's are nonsense to the logician, but meat and drink in everyday life, and in requirements and systems engineering. Warrants in their turn may need to be supported by Backings, which could just be single statements of arguments, but which could equally be Conclusions to be established in their turn by way of further Warrants, and so on.

Finally, the plausibility of a Conclusion can be challenged by a Rebuttal, which is yet another piece of argumentation. The scheme is simple, robust, and like all good things obvious with hindsight.

This little book has stimulated and encouraged many practical lines of enquiry. It seems to me the best sort of academic work: clear, cogent, sweeping away the cobwebs of a lot of muddled thinking, and opening up a whole new field with immediate benefits to practical people in many walks of life.

Engineers may well find it directly useful, at least in its conclusions; students may well find it helps them to think in a practical way; and researchers may discover yet more interesting avenues to pursue through its carefully-reasoned pages (not to mention all the literature that it has spawned).

RE-Sources

Books, Papers

The RQ archive on the RESG website:
http://www.resg.org.uk

Al Davis' bibliography of requirements papers:
http://www.uccs.edu/~adavis/reqbib.htm

Ian Alexander's archive of requirements book reviews:
http://easyweb.easynet.co.uk/~iany/reviews/reviews.htm

Scenario Plus – free tools and templates:
http://www.scenarioplus.org.uk

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

IFIP Working Group 2.9 (Software RE):
http://wrobinson.cis.gsu.edu/IFIP2.9/

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

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

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

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

Mailing lists

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

To subscribe, send e-mail to majordomo@it.uts.edu.au with the following as the only line in the body of the message: subscribe RE-online <your email address>

RE-Actors: the committee of the RESG

Patron:
Prof. Michael Jackson, Independent Consultant, jacksonma @ acm.org
Chair:
Dr Pete Sawyer, Computing Department,
Lancaster University, sawyer @ comp.lancs.ac.uk
Vice-Chair:
Dr Kathy Maitland, University of Central England,
Kathleen.Maitland @ uce.ac.uk
Treasurer:
Prof. Neil Maiden, Centre for HCI Design, City University,
N.A.M.Maiden @ city.ac.uk
Secretary:
David Bush, National Air Traffic Services,
David.Bush @ nats.co.uk
Membership secretary:
Dr Lucia Rapanotti, The Open University, l.rapanotti @ open.ac.uk
Publicity officer:
William Heaven, Imperial College, wjh00 @ doc.ic.ac .uk
Newsletter editor:
Ian Alexander, Scenario Plus Ltd., ian @ scenarioplus.org .uk
Newsletter reporter:
Ljerka Beus-Dukic, University of Westminster,
L.Beus-Dukic @ westminster.ac.uk
Regional officer:
Steve Armstrong, The Open University,
S.Armstrong @ open.ac.uk
Student Liaison Officers:
Zachos Konstantinos, City University,
kzachos @ soi.city.ac.uk
Andrew Stone, Lancaster University,
a.stone1 @ lancaster.ac.uk
Immediate Past Chair:
Prof. Bashar Nuseibeh, The Open University, B.Nuseibeh @ open.ac.uk
Industrial liaison:
Prof Wolfgang Emmerich, University College London,
W.Emmerich @ cs.ucl.ac.uk
Suzanne Robertson, Atlantic Systems Guild, suzanne @ systemsguild.com
Gordon Woods, Independent Consultant,
gordon.woods @ bcs.org.uk
Alistair Mavin, Rolls-Royce, alistair.mavin @ rolls-royce.com

RQ39 (March 2006)

The Requirements Engineering Specialist Group of The British Computer Society

www.resg.org.uk

Individual Membership Form for 2006

1. Membership Type

Please indicate type of membership:

BCS/IEE members, please

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

Non-BCS / IEE member (£20) [ ]

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

If you need a receipt, please tick here [ ]

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

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

---

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

First Name:_____Surname:_____

Address for correspondence:_____

_____Postcode:_____

Phone:_____Fax:_____

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

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

Optionally, indicate:

Your organisation's name: _____

Its type of business/domain:_____

3. Your Specific Interests

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

---

4. Your Participation Preferences

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

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

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

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

5. Your Mail Preferences

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

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

6. Your signature:_____Date:_____

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