BCS™

Requirements Quarterly

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

 http://www.resg.org.uk RQ46 (December 2007)

Contents

RE-soundings

From the Editor

This issue is inevitably centred on RE'07, which was held for the first time in India. That ancient civilisation is now emerging as a modern world power in software, with an enormous supply of intelligent and well-educated engineering graduates. One Indian delegate remarked philosophically that it was only a wave, it would pass on to other countries soon.

It was a pleasure to meet so many bright people, and to teach a tutorial on requirements in that exciting environment.

India was still happily recognisable under the new high-tech gloss: we enjoyed shopping in the colourful markets, visiting the sights such as the monumental Moghul tombs and the beautiful Friday Mosque, and not least having a memorable conference banquet complete with sword dance (including three brave/foolhardy requirements engineers as willing victims).

Being told that a single company is recruiting 35,000 software engineers this year is startling, but also wonderful. Requirements work in India will surely go from strength to strength.

In the wide virtual world, I entered the 3rd Millennium as a fully-paid-up webcaster. It was an eerie experience with an invisible and inaudible audience, but they reportedly enjoyed it. You can listen to it if you're curious – see *RE-readings*.

Ian Alexander,
Scenario Plus

Chairman´s Message

I also had the great good fortune to go to Delhi. I'd never been to India before. The closest I'd come was a visit to neighbouring Pakistan in May of this year. The contrast between the ways in which India and Pakistan are reported through the media is startling. India is rightly presented as an emerging economic and technological giant, with a domestic technology sector that already dwarfs that of the UK. In Pakistan, by

contrast, it is political turmoil that dominates the headlines. Both countries share an enormous amount of history and culture, but have diverged economically. Perhaps in response to the success of its neighbouring giant, Pakistan has recently committed itself to making a huge investment in higher education. Much of this new money is directed at science and technology, with whole new technology-focused universities being created, complementing some excellent existing engineering universities. It will be fascinating to watch how Pakistan's technology base develops from this. As it does so, we in the UK are likely to benefit from a flow of talented software engineers into industry and the masters and doctoral programmes of our universities. Indeed – lucky us - we're already seeing the effects at Lancaster and at other institutions.

Back in the UK, the dark nights are upon us, making the colours of India and Pakistan seem like they were ages ago. If that's a depressing thought, remember that by the time RQ 47 appears, spring will be on its way, the daffodils will be appearing and the birds will be in song (Blackberry owners please note – see *RE-verberations*). It's not every BCS specialist group that helps its members cope with the cycle of the year! In the meantime, I hope you have a very happy Christmas and a rewarding start to 2008.

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:

What is a Competent Requirements Engineer?

5 March 2008, London

Are there gaps between what universities, industry, and training companies consider to be the skills of a competent requirements engineer or business analyst? This invited workshop will consider what the current situation is, and will decide what action if any should be taken on a competency framework.

Contact Kathleen.Maitland@uce.ac.uk

Scenarios, Stories, Use Cases

11 June 2008, City University, London

Scenarios Day is always one of the RESG's most popular events. The morning will be a Tutorial on how to write effective Use Cases. The afternoon will be a free seminar on the use of scenarios with speakers from industry and academia.

Contact Neil Maiden and Ian Alexander.

AGM and RE-Fresh

10 July 2008, London

RE-calls

Recent Calls for Papers and Participation

Mastering the Requirements Process

3 days, 25-27 February 2008, London, presented by Suzanne Robertson, Atlantic Systems Guild

http://www.irmuk.co.uk/1/ for full seminar details.

Introduction to Requirements

2 days, Spring 2007, The IET, London,
presented by Ian Alexander, Scenario Plus

http://www.theiet.org/courses for full course details.

CAISE'08

The International Conference on Advanced Information Systems Engineering celebrates its 20th birthday in 2008, so it must be doing something right.

June 16– 20, 2008 Montpellier, France

http://www.lirmm.fr/caise08/ for full details

RE-readings

Reviews of recent Requirements Engineering events

RE'07

15th IEEE International Requirements Engineering Conference, 15-19 October, 2007, Delhi, India

Alistair Sutcliffe, the program chair, announced that RE'07 had 200 submissions, of which 62 papers were reviewed, and 23 research papers and 6 short papers were accepted. 12 industry papers were accepted, along with 14 poster papers-with-demos. There were unusually 2 best papers – Jane Cleland-Huang's on

automated triage, and Jorg Aranda's survey of RE in small businesses, a much-neglected area.

Side view of Humayun's Tomb showing the elegantly restored gardens and water features

Bringing a Guru to India

Ivar Jacobson of Jaczone AB, Sweden gave the first keynote on "Enough of (Requirements) Process – Let's do Practices". This sounded incomprehensible, a secret code: surely processes always consisted of practices? There were 26 OOAD published methods, not to mention all the unpublished ones using structured analysis. The only one you hear about today, he explained, is the Rational Unified Process. He asked who was in the CMMI camp – 2 or 3 hands went up. Again only 2 or 3 hands went up for Agile methods. Jacobson seemed surprised that we were not that kind of audience: presumably in his world {RUP | CMMI | Agile} equals the entire universe of discourse. RUP was hated by the agile camp for ignoring people, he said. Agile's strength was in "social engineering". CMMI's strength is in metrics and process improvement, but it involves a heavy cost in management overhead, and by no means guarantees good software. Focussing on improving a bad way of making software is like paving sheep-tracks (he said "cow-paths" actually).

Jacobson's new paradigm is "practice", by which he means "a unit of adoption, teaching, planning, or execution of a process". Practices are to be first-class citizens, with processes second nature (if not second-class: the p-word is becoming a swear-word). A process becomes just a composition of practices, of ideas (isn't there some confusion here between theory and practice?).

The RUP began with Ericsson's approach, became Objectory, then the Unified Process and then IBM RUP. "Essential UP" followed, and now perhaps it will merge with CMMI and everyone's Agile into Practices. Perhaps.

Jacobson has written 20 process books which he admits people don't read. But people can't really learn skills just from books or websites, he agreed. There is no benefit in creating sects – he admitted he had tried that, but vowed not to repeat that mistake.

Every process has to try to be complete, so it gets too big. Every process borrows a mishmash of ideas from other processes, so it's always a lumpy soup. It's not easy to love processes. People don't follow them, can't follow them.

The history of use cases shows what can happen. It starts as a paper of a few pages; it grows to 300 pages in a book. It gets adopted into UML. It goes off into strange new directions. It is adopted by various processes. It becomes a bandwagon, a tyranny. It gets fossilised. The world moves on.

"We don't talk about best practices" – practices compete. Use Case-driven development is one practice. We can't prove one is better than another: there are many – Pair programming, Systems Engineering, Scrum, Test-driven development, Aspect orientation, Business modelling, SOA, Prince, Business Process Re-engineering, team practices like workshops and so forth. (To RQ's ear, many of these sound like approaches containing several practices; and some such as SE are at a higher, meta-level, containing tools to generate many different system development processes for different situations.)

A practice is separately describable even if not fully independent of each other (eg use case- and feature-driven development overlap and compete).

A practice has a beginning, a middle, and an end. It has a purpose. It has a product. It may conflict with some other practices (its peers).

A process is then seen as a composition of perhaps 8 or 9 practices. Composing practices is hard because of possible overlaps and conflicts – there is no magic.

The advantage is that you don't have to learn whole new processes to work in this way: you just add one or two new practices to your team's skill set.

The iconic RUP image (disciplines like Requirements down the left hand side, time along the x-axis, and whale-like humps and bumps showing amount of work for each discipline over time) was divided into disciplines not practices. Practices can cut across software engineering disciplines. You may need 3 or 4 practices to deliver value to your client: a practice will include a bit of requirements, a bit of design, a bit of coding, a bit of testing. Eg use-case driven development involves specifying the system as use cases, developing them and finally testing them to show that the specified behaviour is in fact delivered.

"Use cases are excellent test cases, by the way." (Hmm, perhaps a use case leads to many test cases – Ed.)

Practices vary in weight, application (general/specific), depth, breadth (one/many areas), prescriptiveness, ceremony, and flexibility. "Adding more and more detailed guidance doesn't help anyone." Instead, argues Jacobson, we should use intelligent (software) agents to provide active guidance, review, checking and automated help. 80% of people's jobs is "no-brain work". He was proud to have worked with Tata (one of India's largest manufacturing companies), he said. The outcome was a 20% cut in production costs.

The future holds composable, lightweight, social practices. We should start light, and add weight and ceremony when needed. We need to stop re-inventing the wheel, learn from past mistakes, and focus on collaboration and improvement. Stop throwing the baby out with the bathwater, he said. RQ observes that reading Shakespeare seems like reading a book of quotations. Listening to Jacobson is like hearing a book of clichés – whether that's because he's been the fount of process wisdom for the last 20 years, or for some other reason, is for you to judge.

Some Interesting Papers

PN Otto, working with Annie Anton, spoke on RE and the Law. It matters because projects in areas like healthcare are highly regulated. Legal texts are interesting but highly difficult. They come from various authorities – building codes are at state level, local rules can conflict, and both can be overridden by federal law. Regulations are heavy with cross-references, internally and to other laws. Also, laws are full of acronyms and technical terms. Case law muddies the problem further: statute law is interpreted by courts, and the history of such decisions strongly affects the meaning of the statute for future judgements. Hence, you need much guidance on what the law actually means. HIPAA (on healthcare) is 90 pages, but the summary of the privacy rule is 25 pages more, and security gets another 7 pages. Finally, ambiguity is everywhere, with words like "reasonably" anticipated threats or hazards – reasonably is not defined anywhere. That is intentional ambiguity. Unintended language ambiguity is also common: "the covered entity must promptly document and implement the revised policy or procedure…" – is implementing to be prompt, or is it only the documenting? We know that the little conjunctions "and" and "or" are risky in requirements: they are no kinder in the law.

Nine approaches (in the paper) are possible to handle legal rules. Deontic (Greek *deon* = obligation, duty) logic can capture rights and obligations in legal text, but this fails to cover case law and other extensions and conflicts, and has rarely been applied in practice. Access control rules can be derived directly from relevant legal text, abstracting away from many of the difficulties. But ignoring cross-references and low-level requirements could be problematic as these affect meaning. Goal modelling can cover the relationships between actors, trust, and delegation, but it's manual and error-prone and does not support queries.

Any successful legal text system will have to identify, classify, and manage texts with metadata. It must also prioritize texts and exceptions to them, and disambiguate legal statements. It must handle traces between laws, and provide a dictionary to ensure consistency. It must support semi-automated navigation and searching. Compliance (and indeed natural language processing) as a whole is a bigger topic than handling regulations. Future work will involve discussing with healthcare organizations how compliance can be handled.

Axel van Lamsweerde said it was a good list of challenges, but one had been missed. He had to sign on p3 for confidentiality, but on p15 he had to sign again to agree that all sponsors would be disclosed: conflicts arose from many sources. Otto replied that at least a system could uncover such conflicts for humans to try to resolve.

Anthony Finkelstein said that licences and patents implied a large body of law to be complied with, presenting a massive problem of search: you need to search every other patent to issue a new one. Otto said that was interesting but out of his scope; patents too had complicated conflict rules.

Kevin Ryan said he wished there were 2 hours for this. Many years ago a UK tax law was expressed in Algol, leading to much interest and the suggestion that all laws should be so written in future – until it emerged that the ambiguities and conflicts remained in the code!

RQ suspects that the effort is worthy, but that

a) theses complexities are inherent in the law with all its vagaries of precedent and case law;

b) clever lawyers are paid a lot to make the s imple seem complicated, so efforts at rationalisation are not going to make things easy any time soon.

Pioneering Fieldwork

Jorge Alanda gave a "best paper" on Requirements in the Wild – how small companies (less than 50 people) do it. Small companies form 95% of all US software companies, but no papers ever appeared in an RE conference before.

The study was exploratory, to generate testable hypotheses. It looked at companies that primarily do software, were small, and had been running at least 1 year. It used interviews (mostly with partners, mostly 1-3 hours) and site visits. The interviews were open, allowing interviewees to raise topics. Listening was non-judgemental.

The 7 cases in the paper included a medical imaging company inside a hospital. It uses Sand open-source software, specified in a Wiki. Another called Agilista has just 4 employees working on industrial automation

such as problem logging. A third sells office software to publishers and news agencies: it has no methodology, no documentation and is pride of its creative attitude; its employees are PhDs and graduates. Bespoka employs 45 people writing software under RUP with big specifications for banks, using many business rules. The next has 25 people coding software for mobile devices. Roaming Web has 5 people doing web content management software with Wikis and so on. Projects range from 4 hours (!) to 3 months. Requirements are listed in Excel. Rentco has 25 people doing rental management software.

All the companies had practices that worked for them: they earned enough to stay in business. The people were intelligent and good at what they were doing.

He showed pictures of giraffes, penguins, anteaters, mosquitoes and eagles all feeding in their very different ways. Lesson 1: everyone does RE differently. The diversity is striking, but everyone feels their own choices are natural. Types of customer affect your RE practices. Backgrounds and preferences of the founders, and skills of employees all affect practices. Practices evolve: perhaps (hypothesis 1) they adapt to particular niches, and natural selection removes poorly adapted companies. If so, software is an eco-system, and no one technique will ever be suitable for all small companies.

Secondly, like a line of birds flying in a V to profit from the turbulent updraft from the leading bird's wings, people in small companies have a strong cultural cohesion. They can communicate with each other in shorthand. They share backgrounds and personalities. They collaborate for many years. They reject radical changes. So (hypothesis 2) choice of RE practices is IRRELEVANT to small companies, since the efficiency of team dynamics overrides any possible process benefits. (Oh, if Jacobson had only heard that.) Large companies need process because teams are so large.

Thirdly, who does the RE in small companies? Like Batman who works out what to do for himself, the CEO (in 4 of 7 companies) is the requirements engineer. The other 3 had a senior person who'd been there a long time. Maybe the skillset for RE in small companies is a subset of the customer liaison by which the founder builds up the business. Strategy, sales, and specification are inseparably rolled into one. If the boss chooses wrongly the business will founder. We often try to abstract out RE from business considerations: that can't work in a small business.

Fourthly, there is a Catastrophist tone to much RE literature. But in small companies, RE errors are not catastrophes. They prefer to take the punches they know rather than to try anything too radical. Perhaps small companies that survive their birth pangs can sort out requirements problems by getting together in a room and talking them over. Maybe the small firms with RE problems went out of business!

A questioner asked if their methods could be improved. Aranda said he was talking with a couple of the companies, and they were improving, but at this stage the focus was investigation.

Time to ditch the term "NFR"?

Martin Glinz spoke on NFRs. What is an NFR? a property, an attribute, a constraint, a non-behavioural aspect, used to judge the quality of a system, a quality… it's really a mess.

Attribute has broad and narrow meanings. All requirements are qualities (a la Crosby), all are constraints. Is performance a quality or a separate concept of its own? Are NFRs global? Are process and project requirements also NFRs?

There's no consensus on how to classify NFRs, and several traditional classifications don't work. Eg at CERN in nuclear physics: "what shall the system do?" "Deal with the data volume in real time". Function or performance? To the physicist, that's what the system has to do, a function.

Secondly, concepts of kind, representation and satisfaction are muddled up. NFRs are "soft". Wrong. "The power converter control software shall work at 50 Hz A/C." This is a hard-edged, satisfy-it-or-not requirement. 49.9 Hz would be a failure. (Surely that's not the point, frequencies can wobble? -Ed.) Or NFR=qualitative or quantitative, FR=yes or no. False: "The system shall prevent any unauthorised access to customer data." This is a qualitative NFR. But what about "P(unauthorised access) <= 10^{-5}"? Clearly it's Quantitative, and an NFR. Or "Database shall only admit users authorised by username and password." Operational and Functional, or is that Security? The classification is broken, argued Glinz.

Then, should we write NFRs in a separate chapter at top level (a la Volere)? Only if they are really global. Or attach them to Use Cases (a la RUP)? Not appropriate either, if NFRs are global. NFRs are cross-cutting in nature, and we fail to handle this today.

Firstly, said Glinz, we need a better classification: not a tree but with four facets:

- Representation (operational, quantitative, qualitative, declarative)
- Kind (function, data, performance, specific quality, constraint)
- Satisfaction (hard, soft), and
- Role (prescriptive, normative, assumptive).

Assumptive works like this: "every alarm shall be attended to by the nurse" is not enforceable but happens to be important, ie we want to say it matters in our system and has to be recorded. (Perhaps it leads to training requirements – Ed.)

A new taxonomy splits project, system, and process requirements as non-overlapping categories. System requirements split into functional, attribute, and

constraints. Attributes split into performances and specific qualities. This allows each kind of requirement to have any representation, any satisfaction and any role. Functionality is the core concern (to use aspect-oriented terminology) and can be decomposed hierarchically, and NFRs are cross-cutting aspects. Then aspect weaving would in theory allow you to create the specification for each function with relevant bits of the NFR set. This faceted classification is more complex than a tree but it sets a vision. It has yet to pass the test of being adopted.

This stirred quite a reaction from the audience:

Axel van Lamsweerde wondered if adding yet another classification was what was needed. But satisfaction was not just hard vs soft: satisficing, satisfying as far as possible, might be important. Perhaps working on the semantics would help. Glinz replied that perhaps the Satisfaction facet needed elaboration, but that the idea of facets was not invalidated.

Patrick Heymans asked if the Role facet wasn't like Michael Jackson's approach, and how assumptions could be requirements? Glinz said yes to the first, and said that roles were disconnected from the other aspects.

Connie Heitmeyer said she was confused by the 'declarative' and 'qualitative' keywords. All requirements could be declarative, whether qualitative or not. Glinz said these were ways requirements could be written.

Invited Review: Lessons from the Trenches

Connie Heitmeyer of the Centre for High Assurance Computer Systems at the Naval Research Laboratory, Washington, gave an Invited Review talk on Lessons from the Software Development Trenches (where RE theory meets software practice. Her systems are so critical that they have to have compelling evidence of safety, survivability and so on. She has worked for instance for NASA on specifying requirements for a mission-critical software module responsible for detecting faults onboard the Space Station. Such software has to be verified formally to show that the code enforces rules for safety; similarly with military security applications, rules for data separation are shown to be correctly enforced.

A goal-based approach such as van Lamsweerde's KAOS is valuable for both eliciting and validating requirements. No matter how formal the subsequent analysis, an informal stage is needed to discover what people actually want in the first place. But developers find goals difficult, and none of 3 NASA applications had goals in their specifications.

Secondly, making assumptions explicit is invaluable. Researchers like David Parnas and Michael Jackson argue this is vital to separate concerns in the world from those in the machine. Heitmeyer's SCR toolkit provides an Assumptions Dictionary. It was important in helping to validate requirements.

Thirdly, it really helps to follow the propaganda and avoid implementation bias. This makes the requirements concise, separates out design decisions from requirements, and most importantly avoids excluding acceptable implementations that hadn't been thought of at requirements time. NASA's fault detection system did show implementation bias because the development contractors knew about an internal indicator for faults in the software they had already developed.

Fourthly, a formal specification language helped by removing ambiguous and vague requirements, and supported the use of tools to check consistency, completeness, and critical properties (as described above) like security and safety. In addition it enabled the automatic generation of test cases. But few developers know formal methods, and the methods are hard to teach.

Goals alone are insufficient, despite claims from eg Zave & Jackson. The problem is that just saying what is wanted isn't enough: you also need to know what is not wanted. You can do that with logical axioms to specify the required properties of a system. So, Heitmeyer uses both goals and axioms. That at once raises the question: can you derive formal, model-based specifications from goals? Small-scale research suggests it can be done, but it doesn't seem to scale up to serious problems. More likely, you develop both kinds of specification and check that they are consistent as best you can.

Priorities Calculated Semi-Automatically

Jane Cleland-Huang and Paula Laurent gave the "Best Paper", entitled Towards Automated Requirements Triage. They use Al Davis' definition of triage, which is determining which requirements to include in a product release. (That actually includes prioritisation and release planning as well as initial triage.)

The process is named for a Russian surgeon, Pirogov, who introduced a system of triage of casualties into 5 categories during the Crimean war.

Nikolay Ivanovich Pirogov
by Ilya Yefimovich Repin, 1881

The process mimics the way stakeholders prioritize requirements by taking many factors into account:

1. Incoming requirements are passed to multiple competing clustering algorithms to be clustered.
2. A human analyst prioritizes the clusters.
3. A human analyst then weights the importance of category types.
4. The system generates a prioritized list of requirements.

Step 1 is the crucial bit. One algorithm, unsupervised clustering, groups requirements by similarity of terms used, as in Jane's earlier work. It just makes a hierarchy of clusters, bottom up, by computing similarity of pairs of clusters and grouping the most similar ones together. It stops when each cluster is as internally cohesive as possible, and as distinct from other clusters as possible. These are objectively computable metrics.

The Step 1 algorithm was tested on the Icebreaker requirements described in the Robertson's *Mastering the Requirements Process* book. It is used as the baseline prioritisation. The algorithm worked quite well already on its own, but overall results have been significantly improved in research this year by adding further steps as described below.

Secondly, users are asked to supply business goals. The same formula is used to compute similarity of goals and requirements.

Thirdly, requirements are clustered with use case scenarios.

Fourthly, NFRs are detected as in Jane's RE'06 paper. Weighted indicator terms are discovered in a training stage; these terms are then used to guess if a requirement is about reliability, legal, maintainability etc. About 70-90% of the requirements in each NFR type were found (high recall), but the precision was not very good (only 30% or so). Then priority scores were computed as before by multiplying the probability that a requirement was of a certain NFR type by the weight of that NFR type.

The four clustering types were only weakly correlated (0.27 to 0.46) with each other.

The users were then asked to prioritize the clusters manually (step 2) and to choose the weights of the different category types (step 3).

The final 'global' result is that some requirements are promoted from their baseline priorities, ie the additional 3 clustering algorithms made some difference. But did those differences make the final prioritisation better or worse? Did it make sense? This was checked by having 2 researchers check for both inclusion and exclusion errors, ie were minor requirements wrongly given high rank, or were major requirements wrongly given low rank. To make this easier, the requirements were assigned to one of just 5 ranks (20% to Highest, 20% to High, etc).

The results were good: 58% of the Highest requirements were genuinely important, 25% were possibly so, and only 1 requirement was certainly misplaced in that category. Similar results were found for the next rank.

Pirogov is only as good as the algorithms used. Triage and prioritisation are extremely human-centric and complex: the aim is just to support human decision-making, not to replace it. Nevertheless, Pirogov offers a fresh approach for triage and prioritisation in large and possibly distributed projects.

Anthony Finkelstein was curious to know the process context in which prioritisation was applied. Was there no architectural information for trade-offs? Jane agreed, she had thought about it. She would like one algorithm to concern architectural components, so you'd cluster on components (as currently on goals and use cases, etc). (This is interesting, as it says that goal prioritisation and design/requirements trade-offs are basically similar activities. It also says that you can't fully prioritise requirements until you know the design, ie which came first – chicken or egg?)

Kevin Ryan asked if this could work on a real system: the set of requirements is impossible as it's too large to build, and contains contradictions. How do you prioritise those? Jane said she loved that as she had a student working on that right now.

Ian Alexander asked if with trade-offs or requirements prioritisation it could be fair to take a number of orthogonal criteria (at right angles to each other) and weight them so as to boil them all down to a brown goo on a single dimension? His paper had grappled with the problem. Jane replied that it was a great question with no trivial answer: we needed to explore approaches to it, but human involvement would always remain important.

Another Way to Discover Traces

A. Dekhtyar spoke on the related topic of discovering traceability using statistical techniques. The paper had the engaging title "2 Heads are Better than One, or Too Many Cooks in the Kitchen?". The team used a "committee of methods" including sets of keywords, semantic indexing using topics rather than keywords. Then the number of dimensions is reduced statistically by finding eigenvectors. Another method was "latent dirichlet allocation", basically a grouping of synonyms, replacing each use of a term in a synonym group with one term from that group. Then the groups were clustered. Did the committee of cooks do a good job? In one study, there were 361 true links and 232 x 221 requirements in the 2 documents to trace. The other had 52 x 22 requirements, with 35 true links. A team of 30 students worked out the true links manually. The result in both cases was that consensus improved precision a lot, but made recall somewhat worse, whereas majority voting only improved precision a little but did not worsen recall. Consensus also helped to cut down a large percentage of false positives (ie

wrongly created links). More heads are better than one, so Dekhtyar intends to bring more cooks into the kitchen. However, better results are needed to make the method practical for industry.

Globalisation and RE

Alistair Sutcliffe chaired the **panel on the effect of globalisation on RE**. He identified four aspects: Communication, Culture, Co-operation, and Contractual. All of these become more difficult with distance – indeed, a whole series of ugly neologisms like offshoring and outsourcing imply problems of distance.

Communication problems include validating requirements, negotiation, and RM at-a-distance, not to mention working across four time zones. Distance and time differences can force people to rely on inadequate means such as email, which imposes further delays and frustration: the small bandwidth, difficulty of sharing images, lack of face-to-face contact, scope for misinterpretation and to-ing and fro-ing as people ask questions and have to repeat them, referring to old emails way down the heap in the process: not exactly an ideal knowledge management medium.

Contractual problems flow from the other aspects, but include the willingness of junior partners such as Indian software houses to accept rather than "push back", and the challenge of creating workable Service Level Agreements instead of traditional development contracts. SLAs imply metrics for service levels, but perceived quality of service is affected by many factors outside a supplier's control.

Cultural problems include many failures of understanding and imagination. Developers in a firm in India had trouble a few years ago understanding requirements for a supermarket system: they had no experience of such shops (this is quickly changing now). Terms like "aisle" were foreign to them. The client had unthinkingly assumed that the domain was familiar, and relied on such terms in the specification.

Karim's Restaurant in Old Delhi cooks a wicked curry

Other cultural issues include utopian hopes: "we'll establish a little bit of the UK in India to get our software written". Indian developers can't necessarily respond to pressure in the same way: they may have to walk to different government offices to pay each of their utility bills in person, during office hours only. They may have to help sick relatives: cosy western assumptions that if you're sick or out of work, someone will give you money or a roof over your head, or whisk you swiftly to a free hospital, simply don't hold in much of the world. Just getting to work in the monsoon may take many hours, or simply be impossible. Overtime may be really difficult, so time pressure may directly lead to delay, whereas it would lead to late nights over hot computers (and often a met deadline) in the west.

Co-operation problems include the widening gap between clients and developers, as well as those imposed by communication delays and cultural factors. These cause a serious knowledge problem for developers in India. As the junior partner, they need to obtain specific knowledge of several different types, all not provided in specifications. Such knowledge is embedded in other things. It may be cognitive, tacit; it may be organisational (how does an English insurance company work?); it may be societal – schooling, religion, the labour market, etc. This knowledge may be found:

- In developers' heads: "embrained" in Brian Nicolson's rather Dawkins-like word
- In tools
- In development processes (methodologies, standards)
- In practices (informal ways of working)
- In notations.

One other major challenge for projects in India is the alarming rate of staff turnover: often 10% to 20% per year, sometimes more. Rapid growth in the economy (currently an amazing 8% per year, ie doubling in under a decade) puts a high premium on skilled and experienced developers, so poaching and career moves are frequent. This enormous rate of attrition is damaging to any project, especially in a small company where there may be only 1 or 2 people with a specific skill. Large companies like Infosys are less vulnerable, both because they have more skilled people and because they have more in the way of written procedures and standards.

All of this makes it quite a challenge to get a project to work. To overcome the problems, developers can:

- Share knowledge of breakdowns (projects can overrun in the monsoon season, very visible in India but invisible to clients in England, causing frustration)
- Provide training specifically in bridging the divide
- Use "straddlers", people with knowledge of both sides (cultural, and working with both companies); they can explain hidden assumptions, etc

- Frequent travel, providing face-to-face contact as well as cultural and organisational knowledge.

Quality Requirements

Jane Cleland-Huang chaired the **panel on quality requirements**. Someone talked a lot about quality standards. **John Mylopoulos'** premise is that we don't understand quality requirements well, so we can't manage them properly. He wants to sharpen definitions of squashy words like goal, quality, NFR (becoming a bugbear of the conference). He thinks there are 2 kinds of QR. Type 1 can be represented by global hard constraints on the system-to-be. Type 2 QRs qualify a set of functional requirements, and constitute criteria for selecting one design over another, among many ways to meet them.

Mylopoulos starts from Zave & Jackson's 10-year-old paper 4 Dark Corners of RE. It proposes a Venn diagram with a World/Application Domain bubble on the left, a Machine/System Domain bubble on the right, with the specification being the overlapping zone in the middle. Given the Domain and the Requirements, you have to find a Specification such that S satisfies D and R. But it says nothing about Qualities! It's all about functionality. A richer model (OK, he said "ontology", but we forgive him) needs to show how to bridge the gap between domain and system.

Qualities are observable, measurable properties of things (such as products). Quality constraints (Type 1) are hard, measurable qualities that we require, eg power consumption shall be less than w Watts. But some qualities are neither functional nor constraints – he suggests we call these criteria. He thinks these criteria like usability and security cannot be defined (at all sharply). They can however be decomposed into a mix of functions and hard constraints, eg usability can be broken down into subgoals like ease of learning, user tailorability, programmability, modularity, support for specific programming languages, and the like. These are often easier to define (and hence to verify – Ed.). The criteria are satisfied as far as possible: Mylopoulos is comfortable with the system engineer Herbert Simon's ugly term 'satisficing' for this trade-off process. Both types – constraints and criteria – are useful in different contexts.

Don Firesmith of the SEI argued "for the purpose of stimulating debate" that it wasn't true that there were pragmatists on one site and theorists on the other. Every project had to be both practical and well-founded.

Alistair Sutcliffe was with Don on this one – we have always known that softgoals, NFRs, fuzzy qualities, call them what you like, measure usability and so on. We've known for years that it's hard work to meet such things. Designers always trade-off designs against requirements.

Martin Glinz said that Agile people don't qualify their functions. Do all their projects fail? Clearly, there are different situations – not everyone needs measurable, verifiable QRs. Focus on value and risk, don't worry about means.

Delhi

All around us, of course, were the delights of India. Delhi itself has changed a lot in the last ten years – more traffic, more business, new skyscrapers, new cars, no bicycle rickshaws or cows in the new town – and hardly at all: the colour and confusion of the markets, and the negotiation skills of the traders are as they always were.

The Rajasthan Pot Dance

Many delegates took the opportunity to visit Agra and the Taj Mahal; most took advantage of the pre-banquet visit to Humayun's Tomb, now set in beautifully-restored gardens, thanks to the generosity of the Aga Khan, since the area has been declared a World Heritage Site.

The banquet was a delight, with proper Indian food served on the lawn on a warm evening. The fruit bats flapped overhead out to their feeding grounds as we enjoyed an ice-cold beer and dances from different parts of India, performed by a skilled troupe.

RQ specially loves the Rajasthan pot dance, in which a strong dancer ends up balancing 7 pots on her head – they are loaded one at a time, and she steps on the sharp edges of swords in between whiles.

As if that wasn't enough, we were also treated to the famous blindfolded swordswoman dance. Two volunteers are called for. They are blindfolded and made to kneel, with a cucumber in their mouths and another on their raised knee. A chapati is clapped over the dancer's eyes, followed by a blindfold. She then steps gingerly about the stage, making a show of blundering into the volunteers, and then the swords. She picks up both swords, and starts swinging them

about as if they were toys. She gets closer and closer to the two victims, I mean volunteers. Then in a moment she slices chunks off the cucumbers held in the mouths, and halves the vegetables resting on the hands. The volunteers are unsliced. The chopped vegetables are piled on the corner of the stage for all to see.

Whirling Swords, Volunteers with Vegetables

Another volunteer is called – Donald Firesmith bravely (foolishly?) steps up. He is blindfolded, made to undo his shirt, and lie down. A cucumber is rested on his belly. The still-blindfolded dancer takes a mattock – a digging implement with a heavy wooden handle, half way between a spade and an axe – and blunders into the supine form. She whirls the mattock the way a drum-majorette whirls a wand, only it's 100 times heavier and a lot more dangerous. She stops just short of beheading the volunteer. Finally, she swings the mattock and the cucumber falls into two.

RQ expects it will cost Don a lot to buy back all the videos the rest of the world's RE community shot of the event! It will cost you a beer to find out what RQ knows about how the trick is done.

The dancers then fanned out into the crowd, encouraging everyone to dance. It was unbeatable, except that next year's conference will be in Barcelona, so perhaps Xavier will organize dances from all over Spain? Let's hope so.

HeREford and Worcester

RESG / IET Joint Local Group Event

Monday 12th November 2007

"Thanks for your visit to us in Hereford & Worcester and for presenting your very informative and well prepared talk on Better Requirements. I found your material stimulating and practical and you brought to life a topic which too many engineers consider to be quite boring! It was also good to see some new faces in the audience."

I had a delightful time in the beautiful town of Great Malvern. Like Tolkien, I found it "oddly home", ie a place where I would have liked to have been born and raised.

At lunchtime I gave a talk to the engineers and managers at QinetiQ (the older generation may remember that place as DERA, DRA, or even RSRE Malvern). I spoke on *Use Cases*, without slides for a change. Having no comfort-blanket does force a speaker to focus on what really matters. The many questions revealed wide interest in the topic, but also that Use Cases had often been misapplied or found confusing.

After the sandwiches, I went for an open airy walk from the British Camp along the ridge of the Malvern Hills. The low winter sunshine gave beautiful views east over England to the scarp of the Cotswold hills, and west over Wales to many lines of hills dotted about behind the little fields of the shire. Below was the little church where Edward Elgar is buried. His Old English name means Oath-Guard Elf-Spear, so you can see where Tolkien got some of his ideas from.

Looking Westwards from the Malvern Hills

At 6, I had an early but very good pub dinner with the IET local representative – this was a co-hosted event, and I was pleased to discover that publicity in RQ had indeed brought some "new faces" along to the IET's local group. We had a jar of exactly the right kind of ale for a winter's evening:

> *"And there they brew a beer so brown*
> *That the Man in the Moon himself came down*
> *Each night to drink his fill."*

as Tolkien had his hobbits sing in just such a hostelry.

Well refreshed, I gave a version of my talk on "*10 Small Steps to Better Requirements*" to an audience that certainly contained a good number of people from other parts of QinetiQ, but also a mixture of practising and retired engineers from many other places. I was specially pleased by the Hon. Sec.'s thanks and feedback (quoted above), as it meant that for once I was not simply "preaching to the choir" but had a genuine chance to make some converts, as it were.

I stayed the night in the small but elegant Foley Arms hotel. There was a fine view across Worcestershire to

the Cotswolds from the breakfast room. I hope I get a chance to go back.

© Ian Alexander 2007

If you are out of town and know of a local group of the BCS or IET (or maybe of another organisation) that would like to run an RE event, get in touch with the Committee and we will find you a suitable speaker.

Roles and Goals Webcast

Putting a Stake in the Ground: Techniques and Tools for Capturing and Defining Requirements

RQNG, Thursday 15th November 2007, 11:00 AM Eastern Standard Time, 4:00 PM Greenwich Mean Time

Telelogic generously sponsored me to give a webcast, a talk-with-slides delivered live to a logged-in audience. I was professionally trained and introduced, then I was on my own in front of an invisible and inaudible group. Like the conjectured Dark Matter of astrophysics, however, they were not entirely undetectable. Since I was speaking about Roles & Goals, in other words about identifying stakeholder and getting goals from them, I began by asking them if they made goal diagrams, textual lists, or neither. After an anxious wait, I pushed the one-shot "Push Latest Results to Audience" and prayed. A tiny histogram popped up on screen, perfectly formed. People were listening!

The rest of the talk went uneventfully, the emergency printout of all the slides remaining mercifully untouched. To conclude the talk, I gave a simulated demonstration of my goal modelling tool (for the DOORS environment). No-one was sure if it was possible to have DOORS running live as part of the webcast, and in any case I'd have had to hammer away with both hands on keyboard and mouse while clutching a handset with a third hand. So – even discounting the risk of a system crash, typical for demos everywhere – we wisely agreed to present a sequence of screenshots of the tool doing its thing.

The questions at the end showed that people had indeed been listening, and were interested in the described techniques.

The webcast is available on demand at http://event.on24.com/eventRegistration/EventLobbyServlet?target=previewLobby.jsp&eventid=96059&sessionid=1&key=3542FF4E52ED286FC99244F9AFB69A80 (you will need to paste all of this in to your browser, ensuring there are no gaps or line-breaks).

© Ian Alexander 2007

RE-writings

Who is a Requirements Engineer?

Alistair Mavin, Rolls-Royce

Ian Alexander has been trying to persuade me to contribute to RQ for many years. All he needed to do was misquote me in these pages, and I'd surely have written an article long ago. Luckily for him, the review of the recent RESG Soapbox event (reported in RQ45) did not accurately report my contribution to the debate. I have therefore taken this opportunity to clarify my position and further consider the merits of the term 'requirements engineer'.

1) No-one is

The RESG Soapbox event was specifically set up to stimulate debate among the ranks. They offered free beer and invited disagreement. I was happy to oblige on both fronts. Ian provocatively suggested that requirements engineering is not an engineering discipline. Despite the fact that I work for a multinational engineering corporation and have recently gained CEng status, I responded that I completely agreed with him. However, I suggested that we should go further. In my view, the word 'requirements' is as limiting as the word 'engineering' is wrong. Most people that I consider to be good requirements engineers engage in a scope of work far wider than just requirements. At the very least they look across the top of the V model to the design aspects of development.

Most development projects involve some degree of alternating between requirements and design (or problem and solution). For example, if you start with some vague stakeholder requirements, you will naturally start to think of possible solutions (even though this is *very naughty* in a purely requirements-centric worldview). These solutions will be evaluated against the requirements, and will yield some changes to the requirements. The amended requirements will in turn impact the initial design and so on. So requirements and design will both change during development (which one drives the other is not necessarily important, and will vary across projects). The key point is that in the end the requirements and design must balance.

I don't believe that requirements engineering is or should be a distinct discipline at all. In my view, it is a part of the much broader discipline of systems engineering. To call yourself a requirements engineer is to imply that you consider requirements without considering design, which would be of little value to a project. I would contend that people involved in problem-solution-evaluation work should *not* call themselves requirements engineers, least of all the good ones. So clearly, nobody in their right minds would call themselves a requirements engineer.

2) Everyone is

All engineers (I use the term in its loosest sense) deal with requirements. The requirements may or may not

be stated "formally". They may or may not be recorded at all. It is possible that they may be traded, managed, stored in a database, traced, polished, verified, validated, reviewed, maintained, printed, bound, put on a shelf and much else besides. Or maybe none of these apply. However rigorous or lightweight the requirements process, requirements drive all projects (I also use the term 'project' very loosely).

At a general level, *anyone* doing *anything* does it for a reason. Somewhere, explicitly or otherwise, there is a want, a need, a desire that the person's behaviour aims to address. There is a problem, one or more possible solutions and some form of analysis to establish the most appropriate solution. For most real problems, the problem-solution-evaluation cycle is likely to be iterative. Once the solution has been implemented, there will be some kind of test of the design to check that it has satisfied the need.

In the case of intuitive leaps into novel design (classic examples include the Post-It note, SMS message, iPod etc) there may be no obvious requirement that has driven the development. In such cases, the solution comes first, and is seen as a business opportunity. In such cases it could be argued that there was an unstated requirement.

A three-year-old child can be an excellent negotiator (most of them are), without consciously understanding the term, concept or principles of negotiation. Similarly, many people will never have heard the term requirements engineer; yet will be highly effective requirements practitioners. This is because there is a requirements engineering aspect to most interactions between human beings.

For example, a painter calling at a house to give a quote for some decorating work will have to establish what needs decorating, the preparation needs, alternatives such as paint versus wallpaper, types of paint, finishes, the paint colours, who is providing the paint, brushes and ladders, when the job can be started, how long it will take and so on. Like any 'real project', these issues tend to be expressed (if they are expressed at all) in a rather incoherent manner. A typical householder is unlikely to state the problem context, identify the stakeholders, the scope of work, the requirements, and the acceptance criteria as distinct, atomic, measurable statements. Yet somehow most painters manage to do sufficient requirements engineering to provide a solution, get acceptance, and receive payment. They do this, miraculously, without a DOORS database of shall statements. This is because they ask questions, listen to the answers, and aim to satisfy the customer's needs. In order to stay in business, painters need to be good requirements engineers.

What about a man in a pub? When ordering a drink, a customer has many considerations, such as the problem context, requirements, conflicts, options etc. These may include the desire to quench their thirst, look cool, yield to peer group pressure, get drunk, get someone else drunk, appear to be drunk, appear to be sober, show off by spending money, save money, the size of the round, the time of day (last orders?), how they plan to get home (or do they plan to go somewhere else?), who will be home when they get there?, etc. etc. The analysis of such problems can be especially difficult after a few pints.

So if everyone 'does requirements', then it follows that everyone is a requirements engineer (whether they know it or not).

Keeping An Eye on Either Side of the Software / Model / World Divide

Will Heaven, Imperial College

What makes :) a face? The two eyes and the mouth, you might rightly say. But what makes : a pair of eyes and) a mouth? That's a question it's hard to get to grips with. Perhaps we could say that) "looks like" a mouth. But this surely begs the question---put an actual mouth next to a right parenthesis (even turning it 90°) and you're not about to confuse the two by any stretch of the imagination. Yet :) is instantly recognisable *as a face*.

Take another iconic image and ask the same uneasy question. What makes the map of the London Underground a representation of the London Underground? This map, plastered on the walls of every tube carriage and every platform, is possibly even less "like" the actual Underground than a juxtaposed colon and bracket are "like" a face. Yet one still represents the other. Somehow.

Both :) and the London Underground map are models of what they represent. They are abstractions of an actual, real-world thing. Software specifications are also models in a similar sense. At a comparable level, they are representative abstractions of something in the world. A task of the requirements practitioner is to construct such abstractions. But there is no real science of how this should be done. We're on shaky ground when we try to explain how a model does what it does, let alone trying to explain how to go about constructing one. There are tools and rules of thumb aplenty, but these ultimately support a requirements practitioner's instincts rather than dictate a process.

In short, we know surprisingly little about building good models---how to recognise one when we see one and how to go about getting one of our own. Yet since the late 60s and the development of a logic for programs by Floyd and Hoare, a large field of

Computer Science research has been concerned with the application of formal methods to software development in order to provide some guarantee that the software will behave as intended. From a research perspective, there have been many advances, and solutions to the difficulties of developing quality software are still fruitfully being pursued in this field today: one of the proposed "Grand Challenges for Computing Research" is the realisation of a verifying compiler within 15 years.

However, what this work doesn't dwell on is that verification is only ever as good as the model against which the software is verified. Indeed, a guarantee that a piece of software will behave in accordance with a flawed model is worse than no guarantee at all if it instils a spurious confidence in the fitness of the software for its *actual* purpose. This point was being emphasised at least as long ago as 1985, when B Cantwell Smith drew a clear distinction between the software-model relationship on the one hand and the model-world relationship on the other (as also noted by Michael Jackson in his recent Formal-Lite talk: see the RESG website for slides).

The overwhelming focus of the last 50 years' work on formal methods has been on the relationship between software and model. Relatively speaking, this is the easy part. The mathematics that describes this relationship is well understood. But there is no way to hook a model onto the world mathematically in the way software can be hooked onto the model. Any mathematical description of the world is a model, an abstraction. There is no getting at the world "directly": the divide cannot be crossed with mathematics alone. Arguably, most software failures are due to a flaw of some kind in the relationship between model and world. But getting this relationship right is hard. It is also what doing requirements is largely about.

This is of course not to say that research into and development of verification techniques aren't worthwhile, but that it should not be done without one eye on the other side of the model divide and the recognition of the limitations in formal methods that this brings. There is no doubt that formal methods are beginning to crawl down from the towers of academia and find new homes in mainstream languages. Java and C# include syntax for assertions so that a programmer can state (and a runtime environment check) that properties hold at given points during the execution of a program. There have been calls for the features of Spec# (a formal specification language for C#) to be included in C# itself. The popular Java application framework Spring has recently been augmented with an open source facility for supplementing code with formal specifications. Microsoft litters its code with boolean assertions and HP Labs provide a "Java Programming Toolkit"---a set of verification tools---on its website. All this embodies the vision of Bertrand Meyer, creator of the programming-cum-specification language Eiffel: "software is built alongside the arguments that guarantee its correctness".

But how can formal methods help on the model-world side of the picture? Recent work in formal methods has rightly placed a new importance on their use not simply for the construction of software from specifications (along the lines of refinement-based techniques such as Z and B) but on the analysis of the specifications themselves. Tools such as model checkers and the Alloy Analyzer, a so-called "model finder", allow a specifier to explore and develop a model in order to gain better confidence that it in fact specifies what the specifier thinks it does.

However, despite the power and usefulness of tools such as these, full automation generally comes at a price. For practical reasons, in order to be useful, many tools are deliberately designed to be *unsound* or *incomplete*, technical terms in this context meaning respectively (and roughly) that "a tool might tell you that a property holds when it doesn't" and "a tool might not tell you that a porperty holds when it does". One danger when using automated formal methods is that these limitations may not be borne in mind when acting on feedback from a tool.

To give a specific example, misleading feedback can be given by a tool in cases of "antecedent failure". It is a feature peculiar to the classical semantics of the logical implication

$$P \rightarrow Q$$

that the formula is true either when P (the antecedent) and Q (the consequent) are both true *or* when P is false (no matter what Q is). This always initially feels a little odd to students of logic. But it is simply what the formal symbol "->" means. It might be that the natural language gloss of "if P then Q" is the at the root of the discomfort, since it suggests that P is in some kind of causal relationship to Q, but that is not what $P \rightarrow Q$ actually says. Rather, it is semantically equivalent to $\neg(P \wedge \neg Q)$ which might be glossed as "it is not the case that P is true while Q is false". Thinking of it in this way, it feels less odd for the formula to be true simply because P is false. $P \rightarrow Q$ is thus simply shorthand for $\neg(P \wedge \neg Q)$ and indeed many formal systems do without the -> operator without loss of expressivity.

Implication is quite common in specifications---for example, operations are often specified with a "precondition", a property intended to hold *before* the operation occurs, and a "postcondition", a property intended to hold *after* the operation occurs (the goal-driven requirements framework KAOS, for instance, includes such formal structures). If an automated tool informs a user that an operation *op* satisfes its specification, this often means that the implication

$$precondition(op) \rightarrow postcondition(op)$$

is found to be true. But if there is a flaw in the model such that the precondition can never be true---it is "unsatisfiable"---and the implication holds only because of the falsehood of the antecedent, then this

feedback is very likely to mislead and potential errors in the operation itself (e.g. that it doesn't satisfy its postcondition) go unnoticed. Automated analysis of specifications can help identify problems such as unsatisfiable preconditions.

If formal methods are to integrate themselves usefully into mainstream software developments practices, then they must be automated. The general idea of automation is that it separates the user from the devilish mathematical detail. However, this separation can make a user lose sight of potential pitfalls in the application of these methods. Therefore, for automated tools to be genuinely useful, a user must be made aware at all times of the limitations involved in a given analysis.

The kinds of thing I'm thinking of aren't intrusive pop-ups whenever the user needs to know something, but rather qualifications on tools' results. For example, feedback of the form "this is true but it might only be for vacuous reason X" rather than the blunt "this is true". The user can then either use of the feedback with this qualification in mind, or investigate further in an effort to obtain a result without such a qualification. Part of my own work has involved developing an analysis tool along these lines. The Java Modelling Language (JML) community is also now taking steps in this direction with one of its verification tools.

Supporting the development of specifications in the first instance ought to be treated with the same reverence in formal circles as supporting the development of software against a specification traditionally has been. The model does not come for free and software built correctly to a flawed specification is often worse than worthless.

(a card-carrying advocate of formal methods)

Using ASCE in the Requirements Phase

Luke Emmet, Adelard, loe@adelard.com

The Requirements phase of the system engineering lifecycle covers a wide range of analysis and modelling activities prior to system development, including:

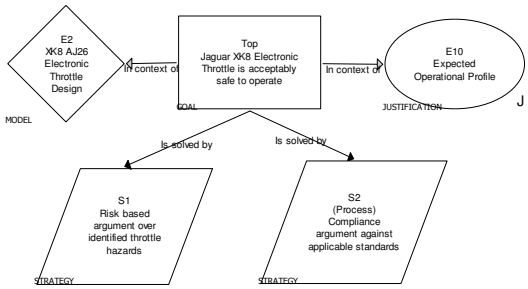
- requirements identification, involving the modelling of stakeholders, their interests and concerns and possible technical implementation options
- working with and analysing existing narrative content in a range of diverse sources, showing traceability of requirements to their underlying sources
- developing requirements models, often starting from informal models, later developed into more formal structures
- communicating with stakeholders and managing their involvement, input and review of emerging requirements sets

Using ASCE in the Requirements phase

ASCE [1] is a flexible and powerful software application designed to address a range of information management tasks.

It features a flexible graphical editor, with rich structured narrative behind each node.

ASCE was initially designed for developing, reviewing and publishing assurance and safety cases, using argumentation notations such as "Claim, Arguments, Evidence" [2] and the Goal Structuring Notation [3]. Such notations are already in wide use, particularly in the safety sector where there is a requirement to explicitly model the safety argument that will be delivered alongside the system. Good practice for safety case development emphasises the benefits of developing assurance arguments early in the lifecycle, in order to reduce operational and acceptance risks later on. As users develop the safety argument, "evidence requirements" are often identified that place requirements on the development cycle. For example if a key claim is made that system testing will demonstrate that the system will achieve a certain reliability level, this will raise requirements on the testing process to collect sufficient evidence to support such a claim.



ASCE fragment: Goal Structuring Notation argument fragment

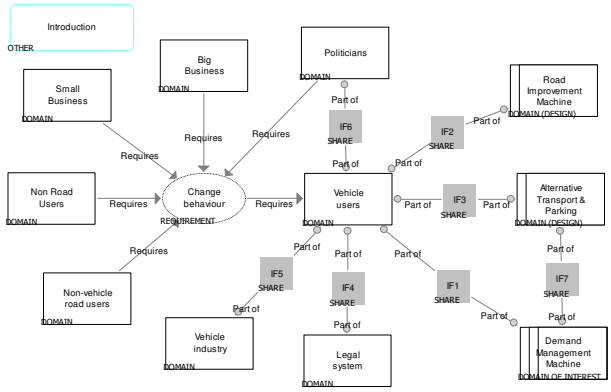
Beyond use of ASCE for developing and expressing the assurance argument for a system, it can also be used to support a range of other requirements activities by means of other notations and ways of working with the tool.

One key aspect of its design is the loose coupling between the graphical notations (schemas) and the core application. This means that arbitrary new notations can be developed, with their own distinct properties and visual characteristics. Based on user input, a number of additional notations have already been implemented to support RE activities, including:

- Hierarchical task analysis
- Problem frames
- Task planning

When the user chooses a particular notation, the user interface constrains the choice of modelling elements to those defined for that notation. As well as determining the kinds of elements that can be created,

the notation also defines the permitted relationships between them. Each notation contains a number of "check rules" which permit the user to check the structure of the current model (for example in GSN – "every goal should be solved by one or more sub goals or solutions"). In this way ASCE provides more structure and focus than a general diagramming tool such as Microsoft Visio which conceives of graphical elements in a very generic way. Although custom applications can be developed using Visio, this usually requires a lot of work to constrain the user interface in a usable way, so it is generally used only as a drawing package.



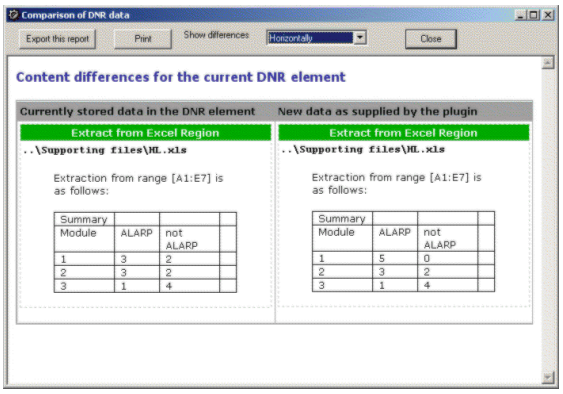
ASCE fragment: Problem Frames (see also RQ44 article [4])

As models grow and become more elaborate, users need effective ways to manage this increasing complexity. ASCE has a "user view" feature, which allows the user to focus on collections of nodes and links that serve a particular purpose. For example, different user views might be developed that contain elements (a) for a particular user scenario, (b) for a particular stakeholder, or (c) that were currently under development by the author. Content can be added and deleted from a user view in the same way as it can be from the main layout. User views can therefore be used to group and arrange related modelled elements.

As new applications for ASCE, and their corresponding notations, are developed, there is often a need to extend the tool to implement required behaviours associated with using that notation. For example, it may be necessary to analyse the graph structure and propagate information over it. ASCE has a plugin engine that allows additional functionality to be bolted-on, such as the ability to integrate with external services.

ASCE has a number of tools for managing the associated structured narrative associated with each node. Users can create narrative using an HTML based word processor, and copy content from other applications via the clipboard.

As well as handling standard hyperlinks to other parts of the current model and to external resources, ASCE has a facility to embed blocks of narrative content generated by plugins. These blocks of content – called DNRs (Dynamic Narrative Regions) – seamlessly map information from external sources into the current model.



Content comparison for supporting evidence held within an external spreadsheet

This facility is widely used in linking to external content that is being cited as evidence within assurance cases, but can be used within any ASCE file to provide supporting context for part of the current model. An example application of this is to pull in content from a node in another model, and thus enabling models to be linked together.

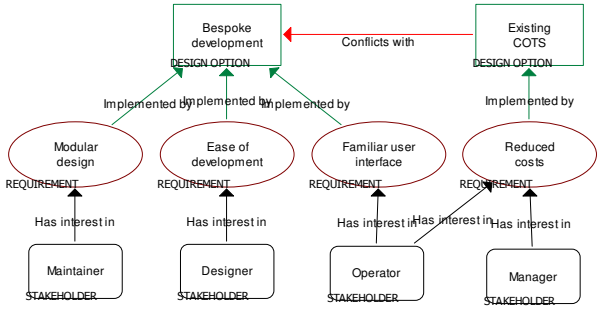
ASCE provides a validation facility for all such DNR elements within the narrative. This allows users to set up dependencies between their current model and external data that supports the modelling activity. As the external content changes, the user is notified. This allows a team to work together where each member may be responsible for different aspects of the information.

ASCE can be used as a tool to model and manage information ranging from the very informal (pure narrative with a heading structure) to the more formal, where concepts are explicitly represented as notational items on the canvas.

For example, a simple requirements modelling notation might contain the following elements:

- **node types**: requirement, stakeholder, design-option
- **link types**: has-interest, has-aversion-to, conflicts-with, implemented-by

Using such a notation ASCE would allow the analyst to model stakeholders and their interests, indicating (a) possible conflict between stakeholders and an emerging requirements set, (b) possible conflicts between requirements, and (c) possible options to implement these requirements. Within each node the analyst can record extra details, and provide links to the supporting material such as interview transcripts, and other emerging documentation.



ASCE fragment: A simple requirements notation

It took about an hour to develop this simple notation using the ASCE Schema Editor which is a tool for developing notations for ASCE. Although one could use a full blown notation such as UML to do this, some advantages of using ASCE include:

- simplicity of use
- improved stakeholder involvement – given the simplicity of this model, getting stakeholder input and review of the emerging model could be easier than trying to explain a more complex UML model
- fluid integration with supporting evidence or background material
- exporting to Microsoft Word and HTML export, using ASCE's standard reporting tools

Conclusions

ASCE is a flexible tool that can be used for a range of fine-grained content management tasks, including those needed to capture and model information within the requirements phase. Aiming to provide sufficient structure for the analysis process, together with narrative content facilities, the analyst can cover a range of modelling and information management tasks. Particularly in the early phases of the lifecycle when structure and requirements are being discovered, ASCE can compliment more formal approaches and provide a way to manage the diverse array of content, from which a more formal requirements structure may emerge.

References

[1] The Assurance and Safety Case Environment (ASCE) home page: http://www.adelard.com/software/asce

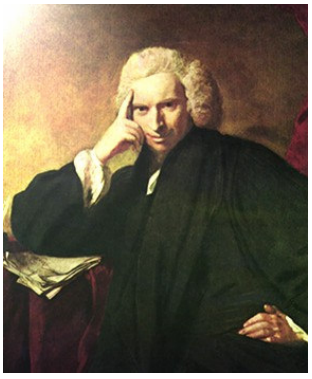
[2] The Adelard Safety Case development manual (ASCAD): http://www.adelard.com/web/hnav/resources/ascad

[3] What is GSN? - GSN user club: http://www.origin-consulting.com/gsnclub/gsn.php

[4] RE-writings - Requirements Quarterly RQ44 (June 2007)

RE-flections

Travel Broadens The Mind



Laurence Sterne, 1713-1768

One of the pleasures of travel – perhaps its chief delight, once you've got over the sun-sea-surf-sand-sex tweeny thing, or the how-nice-to-just-flop-out-while-the-children-play bit – is the realisation that there are other cultures out there.

"They order", said I, "this matter better in France"

wrote Laurence Sterne, author of *Tristram Shandy,* in the opening lines of *A Sentimental Journey*. Plenty of Englishmen visiting France in the intervening centuries have agreed, though the admiring glance is surprisingly returned by many on that side of the Channel.

The traveller's first reaction in a strange place is astonishment – you mean, you people don't eat marmite and Branston pickle and chips and roast beef and Yorkshire pudding?

Any disappointment is quickly supplanted by curiosity: what do they have for breakfast if there's only rice? Why do they drink retsina? Do they actually like manioc?

Inquiry into such matters leads to at least rudimentary communication, and the pleasures of language:

- Orang-Hutan = (Ape) Man of the Forest;
- Hutan Hujan = Rain Forest.
- Rintik = Dot, Spot;
- Hujan Rintik-Rintik = Drizzle.

Understanding starts to glimmer.

A crunchy prawn cracker is Krupuk (it's the noise it makes in your mouth: try saying it); the clove-scented cigarettes that crackle as they burn are Kretek. Not words but music. The language is Indonesian (a variant of Malay), by the way.

Perhaps you buy a dictionary to inquire a little further. Lengai = slow; Lengah = idle, careless; Lengit = sly and lazy (ah yes, I've met that type). These people enjoy verbal association and the echoes of words; the whole language is playfully observant. Duka = sorrow; Suka = to like; Suka-Duka = Ups and Downs. Mmm, it's compact, and elegant, too.

Pleasurable curiosity is extended into respect at the realization of the skill and resourcefulness of people, especially the oppressed: how do those women in the foothills of the Himalayas climb trees wearing saris and flipflops, with a machete in one hand?

The response deepens, though, into a rich belly-laugh of delight, at the understanding that there are whole peoples like the Nenets, Evenk or Khanty who can perfectly comfortably walk for months through the Siberian tundra with their reindeer, who have the skills to survive, no, to flourish in the seemingly desolate waste.

Nomadic Reindeer Herders

It is simply splendid to realize that not everybody wants to have electronic impulses continually relayed to them with an ever-increasing complexity of pocketable gadgetry while they rush to the next meeting. It is delightful to know that apples and blackberries are still excellent in jam, pie, or crumble, whatever barbarous meanings may have been put on these ancient words by marketing people eager to give their plastic-coated products an aura of wholesomeness. See this issue's *RE-verberations* for more on blackberries.

One of the delights of consultancy is this sort of cognitive travel too. After a month or two, one can forget the physical discomfort of airport lounges, dodgy hotel rooms, and hire-cars that reek of volatile organic compounds. What remains is the fascination of seeing with altogether fresh eyes, of experiencing another way of looking at the world.

Seeing a media company, a bank, a telco, a government laboratory from the inside – or rather, looking out of those goldfish bowls at the oddly-curved world visible through the thick glass – is, well, an experience. Even while one's mind is focused on the task at hand – tweaking a requirements database, or whatever – and another part of the brain is squeaking in disbelief, a small corner of consciousness is silently soaking it all up. Perhaps it's odd that all three things should be going on at once, but that's just how it is.

The differences between organisations are just as strong as the differences between peoples: they are cultural too. How does a banker think? Not like a product manager, nor an engineer, that's for certain. How does a multi-national organisation funded by all the governments of Europe achieve consensus on technical goals? Not without coping with political pressure, to put it mildly. How do requirements differ in a media company and a telco? More than somewhat.

RE-verberations

Rapid Progress … in the Wrong Direction

A recent Orange Blackberry advertisement has taken a bold step towards fulfilling Aldous Huxley's nightmarish prophecy, that we would all be subjected to a constant stream or barrage of 'impulses' throughout our days.

The organisation itself sounds, fittingly, like the monstrous chimaera of a genetically-modified fruit, twisted out of natural recognition by perverted scientists. It was probably hatched on the Island of Dr Moreau.

The hideously deformed all-in-lower-case advice offered by the advertisement, tastefully decked out in pumpkin-orange on a Hallowe'en-black background, was

> "take……..all the…unproductive gaps…….out…….of your…day" [sic]

Lest its (or should that be "their" – how does one speak about a chimaera: one or two?) meaning not be clear, the ogre growled on as seductively as it could manage:

> *"Whether you're in a cab, on the train or out and about, you can carry on working […], leaving more time for the important stuff in life."*

In other words, the acolytes of Dr Moreau believe that spending a few minutes of your day enjoying the birdsong and flowers of springtime in the park on your walk to the station is a waste of time.

In partnership with Dr Frankenstein's zombies, the semi-automated processes that pass for 'thought' in their decorticated skulls recommend that you abandon the precious moments you spend walking through the delicate yellows, oranges, and reds of autumn, admiring the grace and beauty of the trees and the impudent wildness of the mushrooms peeping up through the leaf-litter.

They want you to stay stuck inside your head, gawping into your cyborg-implant, worrying about when to reschedule the next quality review.

Get this: they want you to twitch and fret under the control of electronic impulses in your every waking

moment. Read the advertisement again, you'll see. Surely that's a Misuse Case from any reasonable perspective… hmm, except the Negative Stakeholder's.

How to Use Blackberries

RQ has a recipe for baking blackberries into a delicious cake (see photograph). He hasn't experimented with the cyber-variety, but will be happy to try baking some in the hot ashes of his barbecue.

Oddly enough, the device itself seems entirely sensible. A mini-keyboard for text entry is obviously a better idea than trying to use a numeric keypad for full text.

Exactly as Specified, Not as Expected

BBC Three has been running a splendid series this autumn in its program *The Real Hustle*.

BBC Image

The presenter/hustlers demonstrate that with a little practice, some smooth talking and a few cheap props, it is quite straightforward to separate ordinary people, even suspicious ones, from their hard-earned. They are so plausible that they can steal the entire contents of your living room, while you look on. In the case of Jessica-Jane Clement, RQ suspects that she could say anything at all to male punters and they would believe her, as long as she made eye contact. But there must be a different explanation for the success of the other two hustlers.

One particular scam had the team doing business as market traders. The goods were accurately described as 'Model Airplane', 'Box of 100 Coat Hangers', and '40 Cigarette Lighters'. The team answered each question truthfully. What were the coat hangers made of? Good quality steel. How many in the box? 100, guaranteed. You could hang up your coat or trousers on them. What was the Model Airplane like? It was simple to assemble, you could make it in under an hour. It was a popular design, many thousands had been sold. And the lighters? They were the disposable kind. The punters queued up to get their boxes.

The contents? A sheet of paper with instructions to fold it into an airplane; a bag of 100 nails, with instructions to hammer them into the wall to hang your coat on; a box of matches. No lies had been told. The customers got *exactly what had been specified*. They didn't get what they had expected, of course. Any resemblance to requirements? You don't say. The BBC humanely gave the disappointed punters their money back afterwards, by the way.

The Right Productivity Tool: Paper

In the Nov-Dec 2007 issue of *Software*, in his *Tools of the Trade* column, Diomidis Spinellis approvingly quotes Clifford Stoll:

> "A box of crayons and a big sheet of paper provide a more expressive medium for kids than computerized paint programs."

Spinellis realized that when he had to "struggle with a tough problem", he had to

> "turn away from [his] sophisticated software tools and grab a plain sheet of paper".

What was the secret behind this "millennia-old Chinese technology"? Paper, he mused, had many significant advantages. (Human Factors people call them 'affordances'.)

> "Paper is completely noiseless, it doesn't consume any standby power, it won't crash, it doesn't glare, it doesn't catch viruses,it won't heat the room, and it won't give you repetitive-stress injury."

He can use an arbitrary number of fonts and symbols. He can highlight, cross out, and underlline. He can effortlessly mix text and graphics, annotate his pseudocode with lines, arrows and code excerpts.

> "Try that on your code or diagram editor."

Also, it's a lot faster than writing or drawing on the screen. It doesn't slow you down by trying to check your syntax, so you can work undistracted: lack of features is a feature!

> "This means more time for thinking about the problem rather than the particulars of writing the code or satisfying a specific API's requirements."

And he doesn't waste time looking at Google or Wikipedia. He isn't troubled by mail pop-ups, chat client interruptions, incoming RSS items, or friends joining a voice-over-IP client. He mentions the finding of Tom deMarco and Tim Lister in *Peopleware*

(Dorset House, 1987): to be productive, we have to enter the valuable mental state called "flow". It can take 15 minutes or more to enter this state, but only a trivial interruption to leave it.

The moral of the story? – find ways to safeguard your creative and productive time. In particular, cherish the "flow" state, setting aside times and places to be away from the destructive interruptions that go under the name of "time-saving" (see "*Rapid Progress … in the Wrong Direction*" above). Shut the door. Turn off the mobile. Log off the computer. Have a nice cup of tea. Go for a walk. Take a bit of paper and a pencil. Let your mind focus on what you really want to say.

It's the exact opposite of taking all the "unproductive" gaps out of your day. And of course, it's the only way to be productive.

RE-aligned to BABoK

The International Institute of Business Analysts has awarded the Atlantic Systems Guild the status of Endorsed Education Provider.

As part of this, three Volere requirements courses (*Mastering the Requirements Process* Parts 1 & 2, and *Business Requirements Modelling*) have been approved as being aligned to the Business Analysis Body of Knowledge (BABoK) and hence are recommended for business analysts who wish to become Certified Business Analysis Professionals.

Details of the courses at http://www.volere.co.uk. For how to register for the CBAP examination, see under certification at http://www.theiiba.org.

Congratulations to Suzanne and James.

RE-partee

Certified Safe Process

One of the RE'07 tutorials was Donald Firesmith's "Engineering Safety- and Security-Related Requirements for Software-Intensive Systems". At the end of Ian's review of RE'07 (*RE-readings*) you can read about what Don volunteered himself for during the banquet.

I was curious to know whether Don was worried that he'd fatally compromised his reputation as a safety guru. "No" was his reply. "I knew these guys were ISO 9000 certified".

© Pete Sawyer 2007

Jargon Buster: Quality

- Fit For Purpose – where the requirements are undefined, we want the product to do just what we haven't said we wanted anyway
- Quality Requirement – a requirement statement that may be of either good or poor quality
- Wishful Thinking – hoping that the carpet will fly if you write enough requirements (cf Fit For Purpose)
- Oxymoron – piece of nonsense, a self-contradictory statement
- Requirement – statement of a problem
- Design – statement of a solution
- Design Requirement – *see* Oxymoron
- Noise – word or phrase that adds nothing to a specification
- The What Not the How – *see* Noise
- Frozen Baseline – decision to pretend that the requirements have now stopped changing (cf Wishful Thinking)

Sufficient Water Shall Be Provided

RE-sources

Books, Papers

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

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

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

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

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

Requirements Engineering, Student Newsletter:
www.cc.gatech.edu/computing/SW_Eng/resnews.html

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

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

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

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

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

Software Requirements Engineering Articles (India):
http://www.requirements.in

Media Electronica

RESG Mailing List
http://www.resg.org.uk/mailing_list.html

RE-online
http://discuss.it.uts.edu.au/mailman/listinfo/re-online

ReQuirements Networking Group
www.requirementsnetwork.com

RE Yahoo Group
http://groups.yahoo.com/group/Requirements-Engineering/

RE-actors: the committee of the RESG

(Left to Right) Kathy, Pete, Yijun, Steve, Lucia, Will, James, and Neil in Committee at City University

Patron:

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

Chair:

Dr Pete Sawyer,
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:

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

Membership secretary:

Yijun Yu, Y.Yu@open.ac.uk

Publicity officer:

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

Newsletter editor:

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

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

Stephen Nolan,
Stephen.Nolan @ Charteris.com

Regional officer:

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

Student liaison officers:

Dalal Alrajeh, Imperial College,
dalal.alrajeh@imperial.ac.uk

Immediate past chair:

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

Industrial liaison:

Suzanne Robertson,
Atlantic Systems Guild Ltd,
suzanne @ systemsguild.com

Alistair Mavin, Rolls-Royce,
alistair.mavin @ rolls-royce.com

Dr David Bush, NATS,
David.Bush @ nats.co.uk

Members without portfolio:

Emanuel Letier, University College London, e.letier@cs.ucl.ac.uk

Sara Jones, City University,
saraj@soi.city.ac.uk

James Lockerbie, City University,
ac769@soi.city.ac.uk

Contributing to RQ

To contribute to RQ please send contributions to Ian Alexander (iany @ scenarioplus.org .uk). Submissions must be in electronic form, preferably as plain ASCII text or rtf. Deadline for next issue: 7th December 2007

Joining the RESG

Visit http://www.resg.org.uk/membership.html for membership forms, or email membership-RESG@open.ac.uk