Requirenautics Quarterly

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

http://www.resg.org.uk

©2000, BCS RESG Issue 22 (October 2000)

RE-Locations

RE-Soundings

Editorial

Welcome to RQ 22. It's another bumper issue with two reviews of recent events and four articles. Two of the articles are from people who have made distinguished contributions to RE over many years - Michael Jackson (the group's new patron) and Geoff Mullery (long-standing regular contributor to RQ). Of course, as your humble newsletter editor, it's great to have star contributors like Michael and Geoff. What really makes me feel good, though, is knowing there's a huge pool of actual and potential contributors to RQ out there in the general readership. Recent signs are that people are starting to realise that RQ's purpose is for two-way traffic. You can us it to keep up to date with what's on and what issues currently excite the contributors. Equally, you can use it to disseminate your ideas and your experience. Hence, this issue I'm delighted to have two articles from practitioners who may be new to the idea of contributing to RQ, but not to the problems of handling requirements. Aidan Ward and Richard Veryard have written excellent, stimulating articles that draw on the power of analogy to pose important questions for where we think RE is going (and, incidentally, nicely complement Michael and Geoffs' articles). I've even been able to hold over a further article to the next issue. Now that really does make me happy!

Pete Sawyer,
Computing Department, Lancaster University

Chairman s Message

Greetings, after what I hope was an enjoyable summer. Before the summer break, the RESG held its Annual General Meeting at which I am happy to announce three new members have joined the Executive Committee: Martina Doolan, Orlena (Olly) Gotel and Alessandra Russo. Martina is Associate Membership Officer, and will be working with Membership Secretary David Shearer on the ever increasing membership of the RESG. Olly and Alessandra are ordinary committee members, working with the rest of the committee on developing the RESG programme of events and activities. Welcome aboard!

I am also happy to report that Suzanne Robertson, Principal Member of the Atlantic System Guild, has also joined the RESG Executive Committee. Suzanne has more than 30 years experience in systems specification and building, and her courses on requirements, systems analysis, design and problem solving are well known for their innovative workshops and business games. She will be guiding and advising the RESG, particularly to help focus some of its activities on industrially-relevant issues.

Finally, I am proud and happy to announce that consultant, researcher, author and scholar, Michael Jackson, has accepted the RESG Committee's invitation to be Patron of the group. Michael Jackson, winner of the BCS Lovelace Medal, has made seminal contributions to Software and Requirements Engineering research and practice, and we are delighted that he has agreed to share his experience and expertise to guide the RESG.

As always, the input and feedback of the RESG's key stakeholders - the RESG membership - is paramount in developing a quality programme of events. Please get in touch to let us know your requirements.

Bashar Nuseibeh,
Imperial College, London

RE-Treats

Next event organised by the group.

Large Scale Requirements Analysis as Heterogeneous Engineeng

Date: 3rd November 2000
Location: University of York
Contact: Laurence Brooks, University of York (lsb@cs.york.ac.uk)

DEV-TEST (Winter) 2000 — Writing Testable Requirements: A Practical Workshop

Date: 22nd November 2000
Location: Inmarsat Head Quarters Buyilding, 99 City Road, London
Contact: Ian Alexander, Independent Consultant (iany@easynet.co.uk) and David Shearer, University of Hertfordshire (d.w.shearer@herts.ac.uk)

Presented by UNICOM in association with the RESG (RESG members are entitled to a 20% discount).

Requirements for 1 Billion Users - RE for websites and product lines

Date: 29th November 2000
Location: Department of Computer Science, Pearson Building, University College London. (Room to be confirmed, please watch the RESG web pages)
Contact: David Shearer, University of Hertfordshire (d.w.shearer@herts.ac.uk) and Barbara Farbey, UCL (b.farbey@cs.ucl.ac.uk)

Speakers:

Nicola Hall of the Hansard Society for Parliamentary Government. The Hansard Society has developed a novel user interface for parliamentarians to access the net and manage their parliamentary business. More generally the Society "is interested in e-democracy as a means of making representative democracy stronger and more effective. We are interested in exploring inclusive ways of involving citizens in the parliamentary process".

Jeremy Speller, Deputy Director of Academic Services (Web & Research Services), Registrar's Division, University College London (UCL). The Registrar's Division manages the College Web site.

Guy Saward, Senior Lecturer in the Department of Computer Science at the University of Hertfordshire who suggests that "E-Commerce will never take off". He will argue that "for less than £1,000 and a day's work any business can reach a customer base of millions of users. This represents a huge jump in the user / developer ratio. The .com bubble has brought many sites and services on-line but unless they can deliver real value these businesses will not be sustainable. By looking at best practice in E-commerce and Requirements engineering we can identify where the majority of sites are going to crash and burn."

Dr Simon King, Chief Executive Officer of CricInfo, a leading sports site with millions, if not billions of users across the world.

Scenarios through the system life-cycle

Date: 7th December 2000
Location: IEE, Savoy Place, London
Contact: the IEE Events Office, Savoy Place, London WC2R 0BL (events@iee.org.uk) or
http://www.iee.org.uk/Events/a07dec00.htm
Organised by the IEE Professional Group A6 (Systems integration) and co-sponsored by the RESG (RESG members are entitled to IEE member rates).

RE-Calls

Recent Calls for Papers and Participation

Fifth Australian Workshop on Requirements Engineering (AWRE 2000), Queensland University of Technology (QUT), Brisbane, Australia 8-9 December 2000

Scope:
AWRE has become the leading Australian event that brings together researchers and practitioners in Requirements Engineering primarily from Australia but increasingly from other parts of the world. This year for the first time AWRE is co-located with two other conferences: the 21st International Conference on Information Systems (ICIS 2000) held on 10-13 December and The 11th Australasian Conference on Information Systems held on 6-8 December. This will provide a unique opportunity for participants of AWRE to attend two other conferences with the aim of investigating some common strands and/or opportunities for future collaboration with main stream IS researchers.

AWRE 2000 is intended to cover a wide range of topics in Requirements Engineering and hence solicits papers from all aspects of RE. Participation by active researchers and practitioners both in industry and academia and by research students is strongly recommended.

Enquiries:

Dr Didar Zowghi
Faculty of Information Technology
University of Technology, Sydney
Email: didar@it.uts.edu.au

Fifth IEEE International Symposium on Requirements Engineering (RE'01), Toronto, Canada, August 27-31m 2001

Scope:
RE'01 will provide an opportunity for researchers, practitioners, and students to exchange problems, solutions, and experiences in RE. It will emphasise the crucial role that RE plays in the development and delivery of systems, products, and services that permeate all aspects of life and increasingly serve users across national, cultural and professional boundaries. In addition to wanting systems to deliver required functions, users increasingly demand systems that are usable, reliable, secure and responsive. In a rapidly changing world, users and product managers expect today's products to be adaptable to their future technical and social environments.

Submissions:
Details of the submission procedure are available from the conference website:

http://www.re01.org/

Important Dates:
Paper abstract submissions (mandatory) February 15, 2001
Full paper submissions February 22, 2001
Tutorial proposal submissions April 6, 2001
Doctoral workshop submissions April 6, 2001
Posters and Research Demonstrations May 14, 2001
Camera-ready submissions June 1, 2001

RE-Key

A keynote article to mark the appointment of Michael Jackson as patron of the RESG

Variety in Requirements Engineering

By Michael Jackson

Many widely used approaches to requirements engineering are apparently intended for universal application. Look, for example, at the books about UML and the Rational Unified process. They don't mention any restriction on applicability: apparently you can use UML and RUP for any software development whatsoever. It seems reasonable. After all, computers are general-purpose machines. You can use the same computer hardware for running an e-commerce website or as a private telephone switch for an office; to forecast the weather or manage a lending library; to run a video game or to control the braking of a motor car; to schedule meetings or to administer bank accounts. The computer is general-purpose, so you need a general-purpose method for developing the requirements for these systems and their software — don't you?

No, you don't. You need a particular specialised method that is closely tuned to the particular kind of problem you are dealing with. Actually, you probably need several specialised methods, even if you are only dealing with one problem. Realistic problems are heterogeneous: they have to be broken down into a number of subproblems, and the subproblems won't be all of the same kind.

The problems will be different in various ways. They will be about different parts of the world: about the earth's atmosphere, about the Electronic Data Interchange message streams that arrive nightly at the bank, about motor car wheels and brakes, about books and library members. They will raise different concerns: a video game needs high-speed graphics, but weather forecasting needs high-speed calculation; an ABS system needs precise control engineering and maximum reliability, but a telephone switch needs robustness in the presence of errors arising from complex call-processing feature interactions; bank accounts need high security and cast-iron audit trails, but a meeting scheduler needs convenience above all. Their requirements must be developed in different social contexts: the meeting scheduler and lending library need negotiation and compromise among conflicting interests, but the braking system and weather forecasting surely don’t; the e-commerce website involves other organisations such as fulfilment and credit card companies, but the telephone switch doesn’t. The criterion for a successful system will be judged at different distances from the computer: the video game’s success can be judged sitting in front of the screen, and the telephone switch by using the attached telephones; but the meeting scheduler and the lending library must be judged more remotely — by their effects on the users’ and members’ activities, both when they are and when they are not interacting with the computer.

In the face of this diversity it’s absurd to expect one requirements engineering approach to work well for all applications. We need to develop and use many specialised methods. We must recognise several distinct branches of requirements engineering, just as there are several established branches of engineering. Car designers don’t undertake to design computers; chemical engineers don’t design tunnels. Why should a telephone switch requirements engineer expect to develop requirements for a lending library?

In RESG we can help to move towards the specialisation that is the absolute prerequisite of a mature discipline consistently producing high-quality products. We can focus our attention more on the problems and less on their solutions. We can look more critically at our projects, asking ourselves where they succeeded and where they failed, and diagnosing the causes in the suitability or unsuitability of the development methods we used. This way, we can create and maintain intellectual structures for learning from our experience. That is the indispensable foundation for all engineering disciplines worthy of the name.

RE-Readings

Reviews of recent Requirements Engineering events. Reports by Ian F. Alexander

REP 2000

A report on the second Requirements Engineering Process workshop (REP 2000) held at the Old Royal Naval College, Greenwich, London on 7th September 2000.

For the second year running, the Requirements Engineering Process workshop has been held under the auspices of DEXA. Greenwich, despite its dazzling classical architecture and modern exhibits, perhaps lacked the glamour of Florence, and the meeting was smaller though still lively. Camille Ben Achour and Anne Persson chaired the sessions, as last year, with skill and humour.

Nikos Prekas (UMIST, England) described how to combine strategic, method-based reasoning with a deliberative, descriptive approach to RE developed with **Peri Loucopoulos**. The former is prescriptive; you tell the engineer what to do in specific situations. The latter describes the concepts that the engineer might use when working out requirements. The two approaches can be combined, he argued, by describing process fragments (similar to Colette Rolland's Chunks) in Petri-Net style (state-transition-goal state) and by drawing a loop – goal, hypotheses, justifications, design action, with transitions labelled problem analysis, evaluation, resolution, and problem setting – to model the RE process as a reasoning loop. This seems strikingly similar both to Colin Potts' Inquiry Cycle and indeed to the Heron/Reason Cooperative Inquiry process, specialised for RE. The fragments and the loop are combined by saying that the process fragments are generic, while the loop represents a cycle for the specific case. Clearly, to apply the framework in practice would mean assembling a library of process fragments with the types of situations where they would be useful: i.e. patterns for RE. Despite the work in Paris and London on collecting such patterns, it seems clear that such a library remains a long way off.

Paul Gruenbacher (U. Southern California, USA) spoke about collaborative requirements negotiation with 'EasyWinWin', a method developed by Barry Boehm, Gruenbacher and others. The idea is simple: to coax stakeholders with possibly conflicting views through a negotiation process structured to encourage them at every stage to look for win-win solutions.

The 7-stage process consisted of reviewing the negotiation topics to reveal conflicts and constraints; brainstorming stakeholder objectives and interests (divergence); converging on win conditions; capturing a glossary of terms used (the technical abbreviations were easy; the problem was in the use of domain terms, applied in many senses); prioritizing the win conditions by usefulness and ease of use; making a win-win negotiation tree with nodes representing win conditions, supported by issues and options as well as agreements; and categorizing negotiation results by topic. Ideally, a concluded negotiation should document signed agreements on every issue.

It became clear that Gruenbacher was an expert facilitator; we discussed to what extent facilitation skills could be codified and taught, and whether people were born to the job. His own view was that some people could learn it, and could benefit from specific training.

There was an extensive (counselling, etc) repertoire of skills that could now be made available and selected from.

Pär Carlshamre (Ericsson Radio Systems, Linköping, Sweden) compared two 'fairly interesting' RE processes, namely those of Ericsson itself – RDEM, and Telelogic – REPEAT.

Ericsson Radio Systems developed software to operate and maintain mobile telephone networks, i.e. it had a huge network system in which it invested 250,000 hours of work for each release, which had to be synchronized with hardware releases. Leadtime, productivity, and above all quality (reliability of telephone switching and services) were key issues. Releases had to be exactly on time, and requirements had to be carefully prioritized. RDEM provided a state model of requirements (captured, specified, planned, realized) and explicitly grouped atomic requirements into 'Deltas'. These were not just the top 100 requirements, as dependencies between requirements mattered. A Delta had to be a coherent package in which the changes worked together to add value.

Telelogic had a CASE tool, essentially a COTS product, and a large deadline and delivery problem. It had applied the CMM to discover that it had a long road to reach CMM level 2, and was now making good progress. Its lifecycle also modelled states (new, assigned, classified, selected, rejected, applied) though it did not explicitly model deltas. What it did do was to put together a must-have list which was firmly limited to 70% of the available effort; a secondary wish-list was limited to 60% of the effort, i.e. the requirements spanned the range of 100 +/- 30% of what was expected to be achieved. By this 'psychological' device (we all agreed) Telelogic effectively prioritized the requirements and also gave itself room to do more than anticipated.

Didar Zowghi (U. Technology, Sydney, Australia) contributed an interesting paper on Defaults and Revisions. Requirement modelling can be considered as making theories in a non-monotonic logic. With the AI techniques of belief revision and default reasoning, you can manage changing requirements formally. The RE process can be described the same way.

Alfs Berztiss (U. Pittsburgh, PA, USA) spoke about a flexible requirements process, by which he meant an 11-step generic RE process designed to do for requirements what level 3 of the CMM does for software engineering (as the CMM does not cover RE in much detail: level 2 calls simply for a requirements review, use of requirements in software planning, and review and incorporation of changes into the software project).

The process consisted of purpose and environment definition; feasibility study; requirements gathering format; stakeholder identification; task identification; task refinement and review; specialization of tasks (for individual projects); review of the specializations; formalization; specialized requirements; and change control.

While not necessarily agreeing that RE is a branch of software engineering, this seemed overall a sensible and general approach.

Anne Persson (U. Skövde, Sweden) talked about the utility of participative enterprise modelling. She described a 'wicked problem' (aka ill-behaved) where it is hard but necessary to develop practical guidelines for the choice and application of methods. She felt that it was time for researchers to switch from developing more methods to trying to identify when different methods should be applied – ideally in a form that inexperienced modellers could use. She believed that modelling techniques could be applied equally in both business development and information systems (i.e. process modelling and requirements engineering). Situational factors, however, powerfully constrain the use of modelling techniques in different times and places. There may be an infinite range of these, but she wanted to capture patterns of knowledge – trigger questions that facilitators could ask, other conceptual tools they could use in different situations.

Pär Carlshamre said that it was hard to transfer the sort of knowledge described by Anne Persson; the end result was skill or art. **Alfs Berztiss** said that normally transfer was by apprenticeship, but in RE we did not yet have the Masters to teach the Prentices. **Tom van Engers** said that we had to capture the professionalism and skills of group and enterprise modelling facilitation, rather than saying that it was just a black art or all in the genes. **Anne Persson** replied that professionalism was made up of a combination of some inborn ability, learning, training, practice, and reflection on that practice.

F Houdek (DaimlerChrysler, Germany) spoke about his work with **Klaus Pohl** (U. Essen, Germany) on analyzing RE processes in a case study at DaimlerChrysler. Most process improvement approaches such as TQM and AMI were based on the Deming/Steward (1939!) cycle of Plan-Do-Check-Act (also like the cycles mentioned above).

All such cycles contain four main steps: assess the current situation; select areas of improvement and define improvement activities; implement them (in pilot projects); determine whether the desired effects were achieved.

As this had proven successful, he proposed to adapt the approach to improve RE processes themselves, starting with assessing and analyzing current RE processes. He and Pohl conducted classical semi-structured interviews (domain expert, scribe, analyst), each lasting about 2 hours. A questionnaire, with nine sections of a dozen questions each, formed the basic interview plan: admin details, environment, start-up, requirements documentation, requirements elicitation, change management, project finalization (gates and wrap-up meeting), tools used, and subjective judgements. The interviewees were responsible for car instrument clusters, which nowadays were among numerous control units attached to a communication bus in different parts of the car.

They tried and failed to decompose the RE process into neat pieces – activities and their interrelations. The process appeared amorphous, the micro-processes usually unanalysable. Some micro-processes could be identified: getting a decision, evaluating a prototype, performing an inspection – but these did not add up to a process model. (Everyone in the room who had tried a similar analysis nodded in agreement.) The domain experts were remarkably consistent in their subjective estimates, so it did not appear worthwhile to interview more of them. Looking at old documents ('archaeology') was helpful in that it indicated which section headings were entirely missing. He concluded that there was little in the RE research literature about how to assess RE processes; he had tried and failed to obtain a comprehensive picture of the current RE process in his company. Insights that might be useful were that a context model was a good starting point; that triggering events helped in the search for process details and micro-steps; and that looking at documents could be worthwhile before interviewing people. Now he aimed to construct processes and illustrate the process chunks with concrete scenarios.

Ian Alexander (Independent Consultant, England) presented the results of a controlled experiment aimed to show that scenario-driven search found more exceptions. Conventional elicitation tended to focus on stakeholders' primary goals, but exception cases often outnumbered 'normal' cases and could drive project cost. He described an experiment to see whether a scenario structure, namely a typed tree of discrete tasks (which could for example be sets of alternative or parallel subtasks) helped to find more exceptions than using a simple list of requirements. The experiment used an example project – a domestic burglar alarm – with groups of systems engineers on tools training courses. The main procedure consisted of preparing a set of scenarios interactively in the group, and then individually stepping through the scenarios looking for possible exceptions at each stage. The control procedure consisted of preparing a list of functional requirements interactively, and then individually looking for exceptions related to the list.

The results showed that individuals found more than three times as many exceptions when directed by scenarios as by requirements. A control group which did not first carry out the control procedure found as many exceptions as the main group. A model answer was prepared as a check on plausibility; filtered in that way, individuals still found more than twice as many exceptions using scenarios. No individual found more than 11 exceptions, however, whereas the groups together found 28 (one more was found by later analysis). It seems that the individuals effectively found random subsets of the model answer; they did not all tend to find some particular 'easy' subset. Possibly more worryingly, 12 of the exceptions involved human error or overload, while 10 related to intentional deception or system-defeating behaviour; the majority (22/29) would therefore not have been identified by approaches focussing purely on 'system' issues. It is therefore predicted that using scenarios, especially when 'negative' (opposite side, such as the burglar's point of view) scenarios are considered, will more effectively find problems that ought to be handled safely. **Anne Persson** suggested that a similar experiment could show whether groups working together were more effective at such tasks than individuals working separately.

Karl Cox (U. of Bournemouth, England) spoke about fitting scenarios to a generalized requirements process. He searched the literature for described uses of scenarios in RE, and presented a diagram showing what could be used where. He found that scenarios were often applied to a context rather than in a specific process step, and that they often crossed phase boundaries rather than being neatly categorisable. At least 4 uses are described for project inception; 7 for elicitation; 6 for analysis; 3 for discovery ('inventing new requirements from existing ones') – which the workshop argued was not a phase but part of every RE phase; 9 for specification; 7 for validation. He did not analyse the use of scenarios in design. The fact that scenario approaches often cross phase boundaries suggests that the generalized 'typical' process is not the ideal context for using scenarios; none of the approaches applied to all phases, so a combination of approaches was presumably needed.

Finally, **Camille Ben Achour** summed up the workshop papers and debates in a few words. It had struck him that all of us had been dealing with chaos – the unmodellable world of RE; the difficulty of bringing people together to negotiate; the need for flexible, intention-oriented, state-oriented descriptions; exceptions – what a system should do when chaos shows her hand; and we all, naturally, wanted to avoid it.

Scenarios in Requirements Elicitation & Specification

An RESG event held at Imperial College, London on 12th July 2000.

After the AGM (see RE-Funds - Ed), **Ian Alexander** welcomed everyone in a packed room 418 to the RESG event. Scenarios were an important tool with many possible and useful representations. The choice of these could be bound by ideology but in essence they were something entirely practical: a means of communicating between 'users' and engineers. RE involved not only Requirements but Actors (carrying out Actions or Tasks) with Goals (implying Intentions and Conflicts). RE had to bridge the gap between things in the real World and systems in Jackson's 'Machine'; and the entire process had to be conducted in a realistic business-like way with Constraints like performance and cost, following metricated business Processes. Scenarios spanned the whole range of these concepts.

The scene set, he welcomed the speakers.

Dr Eamonn O'Neill (University of Bath) gave a talk on "Scenarios in Participatory Task Model Generation & Validation". Task modelling went on throughout development, in a process that was iterative rather than waterfall-like.

The basis for his work was the need for user participation, Participatory Task Analysis (PTA), external (visible) and internal (in your head) models, the use of scenarios as task model 'wrappers', and tool support.

Usability depended on getting real requirements, and hence on looking into the users' domain and their tasks. Users were the primary source of knowledge. Traditionally, Task Analysis was something researchers did: the users were not actively involved – perhaps a fatal flaw. Participatory Design had active users, but they were involved in prototyping and designing and evaluating, not in sitting back and understanding. Alone, therefore, each technique would give you a partial model. Together they might give a better result, hence the PTA technique.

Task models ought to be refined in a rapid cycle, from users stating scenarios for their own roles, to a dialogue creating users' tasks, to analysis of roles, tasks, and work (object) flows, to validation by walkthrough with the users. He kept actual transcripts - at full length - of the task dialogue.

Internal models in users' and developers' heads contained, he asserted, shared understanding built up by sharing external models of the users' current work situation, which the developer hardly ever fully knows. (Evidently the basic philosophical questions about knowability raised by Wittgenstein and Polanyi are lurking in here.) The problem was to establish common ground; as they say in psycholinguistics, "even if you think you have agreed, you may differ". Theoretically, the participants are "externalising the internal" – and scenarios are the best way. They give a good external model of the much richer and deeper internal model.

Orally presented scenarios often go unrecorded. Rich communication includes natural language and gestures; all other types of model are less rich than scenarios, creating a risk of under-specification.

One problem is that common ground gets progressively less well externalised the more a small group ends up working with a nod and a wink and private conventions as a team; it becomes impossible to hand over to a newcomer.

Another problem is that traceability is impossible without external models, such as tasks, scenarios, and then requirements. You need tool support for the common ground. Scenarios and tasks can be supported by video, but video analysis is nightmarishly huge.

Steve Armstrong said that Quest Map allowed you to form a hyperlinked document structure of video. Eamonn O'Neill said maybe, you could index for later usefulness. But you never video (or index) everything.

Ronald Stamper said you could involve observers in understanding, e.g. in a game situation about an industry such as a steelworks. They learnt a lot and could change what others only observed. Eamonn O'Neill said that participant observation and similar techniques were still very little used in industry; it was a story of lip-service and academic research.

Dr. Neil Maiden (City University) spoke on "Scenarios for acquiring and validating systems requirements". He had worked on the CREWS (Co-operative Requirements Engineering With Scenarios) project from 1996-9, in particular on the SAVRE prototype. Scenario generation and use was problematic, being craft-based. Jacobson in 1992 had left it vague whether scenarios were large or small, and few tools were available back then.

A five-step scenario cycle could run: acquire requirements (ecritoire tool), specify use cases (use case modeller, domain modeller), automatically generate the scenarios (scenario generator), walk through or 'test' the scenarios at user level (Excel presenter) and finally revise the requirements document accordingly. The modular tools had all been validated in industry. But it was easy to generate too many scenarios.

Christopher Alexander had suggested a structure for patterns. These could be used to check requirements automatically. The generator tool could create normal course and alternative scenarios, could prompt you for generic system requirements, and was a basis for systematic walkthroughs. It predicted useful alternative courses and effectively synthesised existing taxonomies. For instance, an exception could be generic, a permutation, or a problem. A problem exception could be raised by a human or machine agent, by a human-machine interface, by communications between humans, by a device failure, etc. This was a basic but powerful approach when applied to practical cases through generic questions (could there be a human error here?).

Checking was facilitated by rules based on a simple entity relationship model: a scenario has events, which related to actions of agents. Events apply to exception classes to which exceptions belong. Actions are taken in response to classes of exceptions.

Patterns help to control complexity as they capture the essence of good sociotechnical systems design (Alexander again). The environment constrained design of artefacts, which in turn afforded desirable behaviours and inhibited undesirable ones in the environment. For instance, pub tables encourage chatting. Patterns can be used directly as (problem) frames to relate scenarios, which describe the environment, to requirements, which describe the system to be designed. As soon as you have a scenario, you can pattern-match it to a frame and the requirements are predicted, at least in outline.

Validation can similarly proceed using patterns; for instance, rule 13 says that you should collect first, fulfil objective last: e.g. you take your ticket out before the metro gate opens; you take your debit card out before cash appears. This is a good design pattern.

The same idea has been applied to naval warfare, with scenarios of ships. The project discovered more than 100 new classes of exception (doubling the previously known total) and created rules to apply in scenario walkthroughs. Another application has been for Mercedes cars for DaimlerChrysler.

Anthony Hall (Praxis) said you needed to be intelligent in generating scenarios, and Neil Maiden agreed.

Ian Graham (TriReme International) spoke about "Use Cases are from Mars, Generative Languages are from

Venus". He asked whether there could be a standard language such as UML or state charts to speak to all users. He answered No, except, he quipped, in some places like Telecomms (presumably referring to Jacobson's original work in Stockholm for Ericsson).

Developers, he asserted, were arrrogant, telling users to learn their language and use it for O'Neill's 'common ground'. In Michael Jackson's seminal paper, A Discipline of Description (REJ, 1998), Specification was defined to be about the Machine, and the customer was stated to want "effects felt some distance from the machine". Similarly the Business Model was about the World, linked to it by Requirements Analysis. Incidentally in the same issue of REJ, Jim Arlow had written pointedly about losing vital information in the case of airlines bookings.

Stories such as these generate use cases. Where did the stories come from? How could one know if a story was relevant? One had to build a natural language model of the business objectives first. Then you could do a task analysis of the business processes, which people still described, said Ian Graham with discomfort, with dataflow diagrams. UML was no help here; the books don't mention anything as far back towards the World as that.

For instance, in the sequence Customer (who isn't a User) places order (not a Use Case, therefore) with a Clerk, who Enters & Validates Order (now here's a Use Case) with the System, it is evident that Use Case analysis is only allowed to address half the situation.

Winograd and Flores, in a classical paper, stated that conversations have structure, and a recursive structure at that. For example, if the top-level event (I'd call it a Goal actually) was a wish to buy a house, there is a sequence of actions or tasks: set goal, offer/request, negotiate, accept contract, completion/handover. But in this sequence, each action can have formal pre- and post-conditions; and each one can be broken down into a more detailed conversation (think of negotiation, for instance). There are also alternative paths – you can withdraw at any stage of the negotiations, for example. This led to a contract-based approach to business transactions in which every step was governed by rules.

But in interviews, the means most often used to obtain requirements, "everyone tells a different lie", so Graham believed in workshops instead. These required skilled handling, and he thought it insane to try to teach facilitation: some people could be facilitators and some simply could not.

Once you had a set of Objectives, you had to obtain agreed Priorities for them, to enable you to drop the right items if time and money ran short. Business models had to be pragmatic and realistic; you couldn't make a standard object model here.

A Use Case was a socratic Universal, comparable to "all men are mortal". A (concrete) Scenario was an Individual case, comparable to "Socrates is a man". These could be combined to produce the Particular result, such as "Socrates is mortal". It was absolutely essential to test out the use cases to ensure that they were realistic. "UML Diagrams can include fakery" said money25

Shailey Minocha inquired how, if the rich picture was not shared, you could ensure completeness. Nicola Millard replied that the only thing to rely on was your gut feeling that the main areas had been covered.

The speakers then formed a Panel to take the meeting's questions.

Geoff Mullery said that Disagreement was a major issue (in the face of a remarkable convergence among the speakers). The only resolution method that had been mentioned was bullying! The key was, he felt, knowing that it happened, so one had to record disagreements. Neil Maiden replied that you could identify the scenarios in which the problem occurred, fragment the problem, and validate each part. Then negotiate with facilitation. And "get the real bastard involved". Ian Graham said it was a research project to take all current knowledge on how to run workshops and turn it into patterns: bullying, recording open issues, identifying respected people at the start, etc. Nicola Millard said that negotiation was a major item: you aimed for win/win. A "what-if ?" attitude allowed you to play out contrasting scenarios. This sometimes resolved problems. Ian Graham said that Eamonn O'Neill's idea of going to the root of the disagreement was central; the pattern was that it resolved both the (contrasting) scenarios. Neil Maiden said that his colleague at City University, Helen Sharp, would like to receive any such patterns for the City website.

Someone asked about notations and mental models. Nicola Millard replied that you had to aim for enough richness of data to cover it all. Neil Maiden said that his customers just wanted summat pragmatic, without much theory. Ian Graham said that $\forall x \ni A \ \exists \ y \mid y^n$ »[x] … etc – he had given a guy the party line on formal methods, and had got the reply that it didn't work as he had a complex case which demanded rules or an expert system. How could one do that in something like OCL? "You had to live for a very long time". Bashar Nuseibeh interjected "Would you fly in a plane defined by rich pictures then?" and Eamonn O'Neill replied no, but that was the only area. Hugh de Vastaire said that video was very dull to work with. What could one use instead? Ian Graham replied that you could use UML by rolling the sequence, activity, and use case diagrams all rolled into one neo-notation. He thought we needed to sort out the overlapping semantics of rich pictures, use cases, sequences, and so on. But Grady Booch had been right, we had to avoid reverting to dataflows.

Deborah Hopkins asked how one could capture non-functional requirements. Ian Graham argued that "the system has to go faster" was a function anyway, so talk of separate constraint requirements was old-fashioned. You just wrote a use case and put it all in. Neil Maiden said that was all very well but what about usability and training requirements, which did not easily appear as functions. Nicola Millard said a non-functional requirement for a system was a function for a trainer. Neil Maiden said that we did not know how to describe sequence tasks in models; text worked best. Geoff Mullery said that the representation technique had to be chosen according to the problem.

The meeting adjourned with all the speakers to the pub (with a large table) for several hours of erudite discussions.

RE-Papers

Requirements, time and trust

Aidan Ward
Antelope Projects Ltd

Introduction and metaphors

se casd

I still love an old quotation from the delightfully named Ron Stamper that the sure way to get the meaning of a piece of information to change is to attach a budget to it.

When you go back to an old system, its functionality is the sum of its design functionality and the workarounds of its users. Neither the designer nor the user can, as a rule, understand the system without the input of the other. This is not requirements failure, it is normal change.

Drivers for change in the meaning of information

Human beings are primarily meaning makers, and the meaning we attach to information and events changes constantly under its own pressure. Today's success selling Rover to BMW is tomorrow's exposure to retrenchment and job losses. The meaning of the facts and figures in the business cases for both is transformed by events. Imagine yourself as a senior executive wondering into whose hands the inside information might fall and what mischief they might do with it at various stages.

The first driver for change is sub-optimisation. All organisations and wider systems such as supply chains are subject to local optimisation attempts that drive global performance the wrong way. When we try to reduce hospital waiting lists we find clinical standards under pressure. When we apply external pressure to schools to improve SATS scores we get demotivated kids and cheating by teachers. Every initiative that creates a new use for information will be followed by an understanding of where sub-optimisation has occurred and a new set of meanings. Notice that any requirements exercise is de facto driven by a sub-optimised initiative and therefore will be overtaken by a reaction.

The second driver for change is lack of trust. It takes trust to maintain a shared set of meanings because in doing so the parties equate their interest in an initiative. The common case is for a superficial agreement on direction and meaning followed covertly by an exercise in how to turn the agreement to local advantage. Another common case is to distance from the agreement and to see no need to resolve inconsistencies with other meanings generated with other parties at a later date. This is often referred to as buy-in or its absence, but the roots of change are in the lack of trust and the behaviour it generates.

None of this is new or surprising. It is important to recognise, however, that the seeds of requirements change is embedded in the exercise of requirements gathering and modelling. The more you try to force consistency the greater the rate of change will be. My second favourite quote is Diego Gambetta talking about "constant semiotic warfare": there will always be a battle for meaning because it is an aspect of power and control.

Making progress against the flux of change

Anything that is pinned down, defined, is fair game for someone to move and redefine. This happens in our musical metaphor as well of course: fashions change in the performance of the classics and the language of jazz evolves rapidly. Classical musicians and critics may well make a principled fuss about the former, in general the latter is welcomed, innovation being of the essence.

Looking at development teams, the direct users of requirements techniques and the requirements themselves, we can see both musical tendencies. We see disciplined, proven requirements processes that systematically cover the ground and capture the meanings of the moment. We see skunkworks teams or Extreme programmers whose definition of a system requirement is sufficient unto the day. If we dig a bit deeper we can see that the orchestra and score of the requirements process protects the less adventurous requirements engineer from the need to nail his colours to the mast. The skunkworks substitutes high skill and complete trust amongst the team for the need to be systematic and to capture definitions. The other significant aspect of a skunkworks is its decoupling from the power structure, which we are suggesting is the root of change.

While someone is telling me to produce the goods, I will go through the motions. If I find a problem I will weigh the odds as to whether to raise it or let someone else find it. Only if I am intrinsically motivated and on reasonable terms with my colleagues will I take the risk of drawing attention to potential failure. Extrinsically motivated management of business risk is a non-starter. Risk here is an aspect of change. Problems left unaddressed will clearly become project risks.

You can no more command intrinsic motivation than you can command love. The stable resolution of a set of requirements is the same thing as the resolution of the interests, perspectives, agendas, conflicts or the stakeholders. The CDM we produce has to have room for manoeuvre, for difference and for change. We have to understand what to capture and what not, how to have flexible definitions and to leave decisions to the latest possible time. We have to be able to distinguish a flaw in the socio-technical system from wildly different perspectives that can still be accommodated.

Dialogue and the roots of action

One way of understanding when a requirements process is broken is to look at the set of conversations that are taking place and see what is undiscussable where. The worst requirements processes are obviously those where the client overspecifies out of weakness. Equally if the requirements stem from two tribes at loggerheads then the necessary communications will not take place.

A recent trend in looking for more effective business communication is to use dialogue techniques. In dialogue the urge to advocate your own position is suspended to allow the thinking of a group to emerge. Group thought is surprising, often transcends individual positions, and is hugely energising. It takes months or years to develop and is therefore ruled out in strongly sub-optimising situations. It is hardly surprising that the antidote to a disease lies outside the parameters of the disease itself.

We see something akin to dialogue in skunkworks teams, and dialogue engenders intrinsic motivation. So we have a process that is the opposite of the requirements process that is capable of truly resolving or transcending the differences of interest and agenda, but at the price of suspending attempts to specify in favour of letting solutions emerge without precondition. This is the Heisenberg condition for requirements: you can have an agreed course of action or a definition of what will be produced but not both at the same time.

Dialogue is the root of action because it aligns action with the energy of the team and generates energy in a joint agenda. Dialogue is the opposite of compromise and the sapping of energy and attrition of will that compromise represents. So if we want to align action with a set of shared requirements we need not to surface and make explicit but to dip down into a less conscious, less rational domain and look for real solutions.

We can't fit dialogue into a formal process because the dynamics of the resolutions that dialogue can provide can't be subject to the conditions imposed by the process. But we can certainly learn to observe the way that change gets driven and to recognise when it is not useful to try for explicit resolution of issues. We know when not to push for the "right" system when our local peace process is too fragile for even interim solutions. Less can often be more.

Questions about the requirements

The arguments in this article have been cast as dilemma to make space for creative resolutions. The main point is to avoid the assumption that requirements engineering is necessarily a convergent process. In any situation where the system to be built is strategic, large and expensive, the least bit controversial, or will take a long time to deliver, the process will tend towards divergence. There will be less and less agreement about the desirable properties of the system.

Far from helping to resolve these questions, prototyping and evolutionary delivery put requirements questions back into the lions den. This is where they belong, but by making the issues more concrete, the tendency to divergence will be greater, not less. What is really being prototyped is the new set of working relationships and the effect on the power structure. One response to this is to cast the whole exercise as a piece of managed socio-technical change, the requirements implications of which are mind-boggling.

If the roots of change are in the power structure, in lack of trust and in the simple passage of time then we can ask what tools we have to stabilise requirements and whether we even want to stabilise them given a close association between the creation of new meanings and business innovation. We then have to distinguish change that is mere noise from change that is useful, a notoriously difficult thing to do. The tools we know about are:

- the building of trust to allow exploration of possibilities, and
- the use of dialogue to affect the roots of action

With luck and a following wind we can build a process where requirements are never stable but are only moved against a sense of business change and not as part of the battle for power and influence in the organisation.

It is always good to ask what things are dealt with by a process and what things are not. By abstracting system properties from their roots in the meaning jungle, requirements engineering tries to establish consistency and completeness without understanding the roots of change. In doing so it becomes blind to its own actions in driving change. This is just another form of sub-optimisation, like drug law enforcement making it more profitable to traffic. Consistency and completeness are not good things in and of themselves.

I guess that a more aware requirements process would start with a picture of the necessary stakeholders, the patterns of trust that could stabilise the field of play and a use of dialogue to deal with irreconcilable differences. Requirements would be treated as emergent properties of a system we don't really know how to control. We would recognise good jazz and be grateful and we would allow some pieces to solidify into classics.

Getting a Sporting Chance

Richard Veryard
Veryard Projects

The only requirement is excellence

In this article, I want to expand the horizons of requirements engineering. I'm going to start with a domain that is not normally regarded as an engineering one. I believe that this domain illustrates some features that are already marginally relevant to requirements engineering, and could become more centrally relevant.

Imagine yourself as a sports coach, faced with four talented and highly motivated children. Each child presents a demand, which has to be translated into a ten-year training programme, building the physical and mental attributes necessary to compete in international sporting competitions. Not all the children will reach this level of excellence, but the training programme is (rightly or wrongly) going to be driven by this objective.

(I'm assuming that it takes around ten years to develop a world-class athlete. Many engineering projects have similar lead-times.)

The first child has great potential as a track athlete. She wants to be an Olympic sprinter. The task of the coach is to create a vision of a supremely fast, confident and competitive young woman, and to communicate this vision to the girl herself (and any other stakeholders, such as her parents). He then needs to decompose the vision into its components – speed, stamina, determination, recovery – and then to formulate a training plan that will help the girl herself to build and assemble these components. There may be some conflicts, trade-offs or sequence dependencies between these components, not to mention conflicting priorities between different stakeholders, and the coach will need to manage all of this. In some ways we can see this as a classic requirements engineering task: the requirement is pretty clear; much of the difficulty resides in the detail; the coach relies largely on past experience to find the best solution for the girl.

The second child has great potential as a footballer. He wants to represent his country in the World Cup. Now this presents a more complicated problem than the sprinter. The boy needs to develop into a useful member of an international team, but in order to do this he will need to be a useful player in a series of increasingly demanding junior teams. What counts as useful varies at different levels, and changes with different fashions over time. A boy with the wrong set of specialist footballing skills may be left out of the team, and thus fail to get the necessary match practice to develop to the next level. And there's no point in being the best Centre Forward in the world, if nobody wants to field Centre Forwards any more.

This means that the training plan needs to have a much higher degree of flexibility. The coach must understand changing team structures, and anticipate how these changing teams affect the requirements for individual players.

Thus the requirement is much less straightforward. The requirements analysis cannot be based purely on past experience, but must be influenced by some reasoning about the future. Coping with – or better, anticipating – requirements change has become an important theme within requirements engineering, and in this light we can still recognize the coach's task as a requirements engineering one.

The third child has great agility and enthusiasm for a wide range of individual and team sports. One option is to select a sport now, and to concentrate the training plan on the requirements for that sport. But better if possible to develop a training plan that is generic across the requirements for many sports. At some stage, it may be necessary to focus on a specific sport, but this choice can be left until later. This choice may depend on such factors as the popularity and television coverage of the sport, as well as the developing skills of the girl as compared to her peers.

Can we still regard this as an exercise in requirements engineering? Some rather generic skills requirements remain, plus a second-order requirement for opportunist responses to a wide range of sporting challenges. But these requirements are much more difficult to substantiate than for the first two children. They cannot be derived from observations of current sporting champions, except in a rather roundabout way, and may be based largely on hunch.

The fourth child isn't even committed to sport. He has something we can call an existential requirement: to be somebody. Becoming a sportsman is one way to fulfil this requirement – but only one way. A sports training programme may still give him something of value, even if he subsequently uses the components – stamina, determination, bonding and team spirit - in some other domain entirely. (But he will only get these components if he takes sports seriously to start with.)

What is the nature of the child's desire? Perhaps the child can only find out by exploring multiple options. This child's demand pulls the coach away from requirements engineering as most readers would recognize it, and towards something more akin to counselling. But of course the wise coach will acknowledge that this is not unique to the fourth child, but has been an integral part of the job in all four cases

The only requirement is global domination

Now imagine yourself as a requirements engineer, faced with four talented and highly motivated entrepreneurs …

One wants to build an efficient internet procurement process. One wants to construct a portal to which masses of other people will connect. One wants to build a website with a massive customer base, although she's open-minded as to the possible future uses of these assets. And the fourth simply wants to make money somehow.

A traditional requirements engineering exercise would elicit some verifiable requirements from some community of "users", and might build upon known solutions to similar requirements. But there aren't any users yet, and nothing to copy. To the extent that there are any other known solutions, there is a primary requirement to be significantly different (in some undefinable way) from any previous solution. But this is a negative requirement, not a positive one.

Do all these requests represent appropriate challenges for requirements engineering? (My answer is Yes.) Do requirements engineering techniques help you address these challenges? (My answer is: Yes to some extent.) And are there other complementary techniques that requirements engineers might usefully adopt? Undoubtedly.

CORE-Blimey!

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

Wheels Within Wheels

Ever since I have been involved with computing there has been some concept of a project life cycle model. The latest is the so-called Spiral Model, which assumes a series of re-applications of fairly simple life cycle models, gradually improving the "fit" of the developed system to its environment.

It will fail. Not in all cases, but regularly enough to show that it can't be treated on its own as the answer to the problems of developing systems over their entire life in the environment they serve. The questions I am addressing here are: Why do life cycle models fail? Is there anything which addresses the problem?

The reason life cycle models fail is that that they don't sufficiently reflect reality. Specifically, they do not recognise that a project - even on a single "spiral" pass does not in fact involve a single, simple phase model. They do recognise the need for re-work and, to some extent, evolution - but this is always presented as a simple maintenance activity.

A project produces and uses a series of products, each potentially with its own life cycle. Often the products themselves are produced via a series of intermediate products –each with *its* own life cycle. It is the failure to recognise these component life cycles and incorporate them into the wider life cycles in which they appear which is at the root of the project life cycle reality failure.

A related point about these families of life cycles, which must also be taken into account is that, like all families they are likely to be integrated with external (other) families, so the life cycles can not all be assumed to be under the control of any one "top level" project manager.

Many people will have met one of the independent life cycle problems. I referred to it in my contribution "Keeping Down Your Standards" (in issue 21 – Ed). That is the facility for the tools or the standards used by a long-running project to be updated – not at times convenient to a specific using project, but at times driven by a tool vendor's marketing decision or a standards body's committee work.

I mentioned I was caught by the decision to update the C++ standard in a non-backward compatible way and I know of projects which have been caught out similarly with Ada. Also, I and others have regularly had to adjust to new releases of the tools supporting languages like Ada and C++.

None of these things can be controlled by a specific application project's plans. Instead they must be accounted for in those plans – and it is not always a simple decision. The supplier's support or maintenance of the old release may be terminated in favour of supporting the new release. The cost of carrying on with permanent problems in the old release has to be weighed against the cost of moving to the incompatible new one.

You are probably tempted to say that the problem is simply one of configuration management, but it goes way beyond that and even if it did not there are still multiple other life cycles to take into account when considering what to do. Starting first with the non-development tools problem, there are problems of external life cycles and internal life cycles.

Many systems, particularly on large projects, have their environment's life cycle to take into account. The environment changes and it changes partly because of the introduction of the system under development and partly because of numerous other factors, like politics, technology changes, fashions and so on.

Only one of these is in any way under the control of a specific project manager, but the others are susceptible to varying degrees of forecasting, each in terms of that project's life cycle interests. Beer's VSM, which I have mentioned in several previous contributions mentions the need for both long and short term forecasting, which requires more than just an assertion that some future event might occur.

It requires consideration of things like causal relationships, synchronisation constraints and compatibility between various influences in the current and perceived future environment of the system under development. One specific type of environmental concern is the relationship of the system under development with other systems already in the environment or under development to be included in the same environment.

These systems may occasionally be developed in perfect synchronisation, but in my experience, the larger the project the more autonomous but related and barely synchronised projects there are in its environment, which will have a significant impact on that system's life cycle.

Typically these other systems may interrelate and also be involved with systems in an even wider environment. This is quite likely to result in the fact that one or more of the systems affecting a specific project are being developed for several different environments. This in turn may mean that one very complex system is being built to serve all relevant environments or there are several versions of the system – one for each relevant environment. In such cases a change in the time scales of one of the other projects, which may have nothing directly to do with a given project may nevertheless cause a significant change in the time scale of that project.

For a given project there is also an internal environment – of the development organisation – which may have an impact on the project. It is not unusual for a development organisation either to develop its own library facilities or to use commercial library facilities under licence for things like GUI, database, communications and so on. Even when the library is wholly owned by the developing organisation it is rarely wholly controlled by a particular project which uses it.

The key problem with this situation is that the library suppliers (internal or external) are subject to a range of different influences outside the control of the specific using project – typically there are other projects and the marketing arm of the parent organisation and perhaps even external user organisations.

In my experience the project user belonging to the same company usually comes well behind the marketing/external user side and the priority between several internal user projects is often determined by internal politics rather than technical issues like supplying what was promised on time. The individual project thus has to make and regularly update estimates of what will be available when and allow contingency for (perhaps deliberately) hidden delays in the estimated schedule.

Add to all this the possibility that *this* project's products may themselves have to be developed to meet several other projects' needs, so this project may find itself pressured to change in conflicting ways by other projects with conflicting needs or priorities. That means that this project's products may become more complex to design and build or may have to exist simultaneously in several closely related but different forms.

In either event this project’s products will each have a life cycle driven partly by the life cycles of the other projects with which they interact, so it is not possible to be wholly confident that this project has such a thing as a single life cycle or even (as the spiral model suggests) a succession of single life cycles.

Now, come away from all the external interaction with life cycles and let’s look at a nice simple project which nominally does have a succession of single life cycles as suggested by the spiral model. My assertion is that these life cycles are often not successive except in the sense that there is a defined order in which they *start*.

The order in which they proceed and perhaps even the order in which they complete is not necessarily the order in which they started. That can depend on a number of things, but the most likely thing is that each of the project’s products has its own little life cycle which proceeds only partly constrained by the a specific pass through the parent project’s spiral life cycle.

For example, some development approaches emphasise early development of the user interface and then development of the supporting functionality. In such a case it may well be that the moment the initial user interface is completed, and before any of the underpinning functionality is under development the next generation of user interface may enter its design stage.

As that second design stage proceeds it may well throw up ideas which can and should influence the way the functionality supporting the first user interface is specified or designed or implemented. Now the second pass through the spiral life cycle is directly influencing the first pass – even though the interface on the first pass has been fixed and will not be allowed to change until after completion of the first pass.

In some cases this type of apparent order-reversal may come up with something so dramatic that the first pass through the spiral is abandoned because it would be visibly wasteful to continue when the second pass has shown that an alternative user interface is better and requires a significantly different functionality.

You may at this point be inclined to suggest that what I am advocating is chaotic development and that it would be extremely dangerous to allow things to proceed that way. My answer is that what I am advocating is recognition of reality and that it is extremely dangerous to pretend that reality is “on the blink”.

The spiral model should be considered as a recursive model allowing also for loose coupling with other spirals. At each point, at each level of each pass through the spiral it should be remembered that there might be another internally-contained or loosely linked spiral life cycle required. The point of the current pass through a given spiral “arm” is to organise which things are simple product life cycles, which are spirals and which arm of each spiral contributes to the current “arm” of *this* spiral.

When we talk of complex relationships we often use the phrase in the title of this contribution – “wheels within wheels” – my assertion is that in dealing realistically with project life cycles we have about as complex a situation as most of us will meet in our technical work. Though perhaps, to make the current life cycle advocates feel wanted we should change the phrase in this case – to “spirals within spirals”.

*RE-*Sources

For a full listing of books, mailing lists, web pages and tools that have appeared in this section in previous newsletters, see the RQ archive at the RESG website: http://www.resg.org.uk

The requirement management place
http://www.rmplace.org

A good general resource for RE issues. Includes Alan Davis' Requirements Bibliography.

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

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

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

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

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

Reduced rates are available to all RESG members when subscribing to the REJ. Special 2001 Personal Rate £45.00 (A saving of £19.00).

Mailing lists

The SRE list

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

LINKAlert:
http://link.springer.de/alert

A free mailing service for the table of contents of the *International Journal on Software Tools for Technology Transfer*.

RE-Actors

The committee of RESG

Patron: Prof. Michael Jackson,

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

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

Secretary: Dr. Wolfgang Emmerich, University College London, Department of Computer Science, Gower Street, London WC1E 6BT, UK. E-Mail: W.Emmerich@cs.ucl.ac.uk, Tel: 0171 504 4413, Fax: +44 171 387 1397

Membership secretary: David Shearer, Department of Computer Science, University of Hertfordshire, Hatfield, AL10 9AB, UK. E-Mail: d.w.shearer@herts.ac.uk.

Associate membership secretary: Martina Doolan, School of Information Sciences, University of Hertfordshire, Hatfield, AL10 9AB, UK, E-Mail: m.a.doolan@herts.ac.uk.

Newsletter editor: Dr Peter Sawyer, Lancaster University, Computing Department, Lancaster, LA1 4YR, UK. E-Mail: sawyer@comp.lancs.ac.uk, Tel: 01524 593780, Fax: +44 524 593608.

Associate newsletter editor: Ian Alexander, 17A Rothschild Road, Chiswick, London W4 5HS. E-Mail: iany@easynet.co.uk, Tel: 0181-995 3057

Publicity officer: Dr. Vito Veneziano, Department of Computer Science, University of Hertfordshire, College Lane, Hatfield, AL10 9AB, UK. E-Mail: v.veneziano@herts.ac.uk, Tel: 01707 286196.

Web-master: Dr. Laurence Brooks, Department of Computer Science, University of York, York, YO10 5DD. E-Mail: Laurence.Brooks@cs.york.ac.uk, Tel: 01904 433242.

Industrial liaison officer: Dr. Barbara Farbey, Department of Computer Science, University College London, Gower Street, London WC1E 6BT, UK. E-Mail: B.Farbey@cs.ucl.ac.uk, Tel: o171 419 3672, Fax: 0171 387 1397.

Members-at-large:

Dr Olly Gotel, University College London, Department of Computer Science, Gower Street, London WC1E 6BT, UK. E-Mail: O.Gotel@cs.ucl.ac.uk.

Suzanne Robertson, Atlantic Systems Guild Ltd. 11 St. Mary's Terrace, London W2 1SU, E-Mail: suzanne@systemsguild.com.

RE-Creations

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

Deadline for next issue: 31st December 2000.

Dr Alessandra Russo, Department of Computing, Imperial College, 180 Queen's Gate, London SW7 2BZ, UK. E-Mail: ar3@doc.ic.ac.uk.

RE-Funds

Minutes of the RESG AGM

Held in room 418, Department of Computing
Imperial College
Wednesday 12th July 2000.

Present: Bashar Nuseibeh (BN), Neil Maiden(NM), David Shearer (DS), Carol Britton(CB), Vito Veneziano (VV), Barbara Farbey (BF), Ian Alexander (IA), Wolfgang Emmerich (WE)

Apologies: Pete Sawyer (PS), Laurence Brooks (LB)

1. **Welcome**: BN opened the meeting and welcomed those present.

2. **Minutes of previous meeting**: The minutes of last year's AGM, which appeared in RQ18, were agreed.

3. **Chairman's Report**: BN gave a short report on the activities of RESG during the past year.

3.1 Events: There had been 7 meetings at a variety of venues.

i. RE for E-commerce, London, 14th July 2000

ii. Managing Changing Requirements, Oxford, 31st Aug 1999

iii. Requirements and COTS, York, 23rd Nov 1999

iv. Simplifying Requirements Testing by Using Scenarios: A Practical Workshop, London, 24th Nov 1999

v. Dependable Distributed System Requirements, London, 16th Feb 2000

vi. A Moderate Debate on Extreme Programming, London, 9th March 2000

vii. Using UML in Anger for RE: a mini-tutorial, London, 17th May 2000

3.2 Newsletter: There had been 3 issues of RQ and a new editor, Pete Sawyer, had been appointed.

3.3 Industrial Liaison: Barbara Farbey had been appointed as industrial liaison officer. An invitation was issued to all industrial members of the group to contact BF or BN for more information and to discuss possible future meetings.

4. **Membership**: DS reported that the RESG now has more than 300 members.

5. **Treasurer's Report**: NM presented the Treasurer's report. The group's finances are in a healthy state (see below – Ed). BN mentioned that the group is keen to use funds to sponsor students and would welcome suggestions as to how this can be done. RESG is already sponsoring a doctoral workshop at the International Symposium for Requirements Engineering (RE'01) in Canada.

6. **Publicity**: VV has taken over from CB as Publicity Officer and is working with LB to integrate RESG publicity and website. Integration of RESG membership and membership is also being planned.

7. **Election of officers**: BN thanked the retiring members of the committee, Steve Easterbrook and Sara Jones. The rest of the previous year's committee were standing for re-election, along with three new members: Orlena Gotel, Alessandra Russo and Martina Doolan. The new committee was proposed by Steve Armstrong, seconded by Kathy Maitland and elected unanimously.

Accounts

Year-start	**£**
Bank Balance 05/99	5762.60
Gold Account 05/99	13181.77
Receipts	
Membership subscriptions	690.00
Event income	1314.25
Other receipts	894.27
Subtotal	2918.52
Payments	
Meeting payments	105.58
Printing and stationary	240.31
Events expenditure	903.23
Subtotal	1249.12
Net movement 1999/00	1959.40
Balance at 30/04/2000	20903.77