BCS™

Requirenautics Quarterly

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

http://www.resg.org.uk

©2002, BCS RESG Issue 25 (January 2002)

RE-Locations

RE-Soundings

Special announcement

RESG Membership is **FREE** for 2002.

To join, contact membership secretary Steve Armstrong (s.armstrong@open.ac.uk) or visit www.resg.org.uk.

Editorial

Belated best wishes for 2002. Since I'm writing this just before Christmas I'm going to take the liberty of presenting RQ25 as a bumper Christmas edition. Although there's no quiz on RE events of the year, or an oversize crossword (although a crossword might be a nice idea for a later issue ...), we do have a couple of thought-provoking and slightly unusual things for you to read.

We have a mini-theme in the *Re-Papers* section, stimulated by an article from Julie Dobson illustrating beautifully how requirements engineers need to be prepared to discover that real working practices sometimes deviate from prescribed practices and procedures. Julie's article rang a bell with me and I asked Mark Rouncefield to provide a companion article outlining some of the background to why this happens - a theme of research rooted in the CSCW community. Mark and colleagues have duly supplied an article about "cunning plans" and why plans and procedures are needed but not always followed in the workplace.

Much of the material on which the article draws comes from social scientists examining the impact of technology. As requirements engineers we need to be willing to learn from other disciplines. After all it's essential that we develop an understanding of the diverse domains that pose the problem contexts for our work. Part of the problem in understanding this is, of course, getting at users' tacit knowledge in order to understand their roles, their tasks and their requirements.

The second Christmas treat is Ian Alexander's review of the classic work on tacit knowledge by Michael

Polanyi. This was first published in 1966 (*RE-Publications* is right for once) and shows once again that the many of problems with which we grapple have been addressed before in different contexts.

Naturally, all this is supplemented with the usual healthy fare of articles, event reports and advertisements for forthcoming events. Happy reading.

Pete Sawyer
Computing Department, Lancaster University.

Chairman's message

At the RESG's 7th Annual General Meeting in November, the executive committee welcomed four new members. David Bush joins Efi Raili as Associate Industrial Liaison Officer, Steve Armstrong takes over from David Shearer as Membership Secretary, Juan Ramil takes on Publicity from Vito Veneziano, and Kathy Maitland takes over responsibility for representing the RESG in the north of the country. Alessandra Russo also takes up the newly created role of Vice Chair of the group.

This injection of new blood into the running of the RESG promises another eventful year for the group. A number of events are already being planned for 2002, and we intend to follow up the RESG's sponsorship of and close cooperation with RE'01, with a similar arrangement for RE'02 to be held in Essen, Germany, in September 2002.

With the changeover of Membership Secretary in progress, the RESG committee decided that a review of membership procedures would be timely, including a re-organisation of the group's membership database. To make this as convenient and as effective as possible for all concerned, membership of the RESG for 2002 will be *free*. All that is required to renew membership for 2002 is for members to check and, if necessary amend, their contact details. Steve Armstrong will be writing to all RESG members in the next few weeks, providing the contact information currently held on them in the membership database. Please help Steve by responding to his letter as quickly as possible.

Finally, a few words of thanks to those outgoing members of the RESG committee. Vito Venziano, Carol Britton, Martina Doolan, David Shearer, and Laurence Brooks, have contributed their time and energy to the group's activities over a number of years. On behalf of the RESG membership, I would like to extend to them our thanks and best wishes for the future.

I would also like to extend my best wishes to all RESG members for a happy and successful New Year.

Bashar Nuseibeh
The Open University

RE-Treats

Next event organised by the group

Key Challenges in Safety Requirements Engineering

Date: 2.00pm 27th February 2002
Location: Bath **-** venue to be confirmed
Contact: Efi Raili, Praxis Critical Systems.
(efi@praxis-cs.co.uk)

Safety Requirements is quite a wide topic. The event is aiming at providing some clarity into the definition of safety requirements and how various industries go about deriving and defining them. There is a lot of variety in various industries (and much confusion) as to what safety requirements are. What do we mean by safety requirements? What are they? Are they for example Safety Integrity Levels (SILs)? Are they a set of standards a system needs to comply to? Is there a process (formal or not) for deriving them? What are the major issues a safety engineer is faced with while engineering safety requirements?

Recycling Requirements for Systems and Product Families

Date: 9.30am 24th April 2002
Location: Imperial College, London
Contact: Raquel Monja, Centre for HCI Design, City University. (r.monja@city.ac.uk)

Important industrial and research trends are converging, leading to the opportunity to recycle stakeholder and systems requirements effectively for the first time. Many organisations are developing families of software and software-intensive products that share numerous requirements. Others are re-engineering their current systems to satisfy requirements more effectively. More effective knowledge management and organisational memories mean that requirements and other artefacts are now available to be recycled. All provide exciting new opportunities to improve the quality and productivity of requirements engineering processes. However, recycling requirements is not without its difficulties.

This one-day symposium composed of a tutorial, presentation and an expert panel, will investigate what, when and how to recycle requirements for software and software-intensive systems.

RE-Calls

Recent Calls for Papers and Participation

Requirements Engineering Challenges and Issues for Developing e-Government Electronic Public Service Delivery (EPSD) Software Systems, To be held in conjunction with the 6th World Multi Conference on Systemics, Cybernetics and Informatics, July 14 – 18, 2002, Orlando, Florida, USA.

http://homepages.feis.herts.ac.uk/~resg/html/upd._08_11_01_a.html

Most governments are currently engaged in ambitious, tight-scheduled e-government Electronic Public Service Delivery (EPSD) initiatives. The goal of these e-government initiatives is to provide the business community and citizens with an efficient and effective public service. However, the e-service delivery challenges for e-government are profound. Citizens and business organizations that need to deal with the government should have a choice of channels for accessing government services. Private and voluntary sector organizations should be able to access both central and local government information in order to deliver their services to the citizens. The mobility of citizens over a wide geographical range raises new essential system and user requirements for the e-government EPSD system. To meet the challenges posed by these systems, new and efficient requirements engineering methods, processes, and techniques that are suitable for developing government-to-citizen (G2C), government-to-business (G2B) and government-to-government (G2G) systems are required. This session is seeking thought-provoking presentations and revealing case studies that break new ground in understanding the requirements engineering challenges and issues that will lead to practical steps in developing 'joined-up' e-government EPSD systems.

International Council on Systems Engineering 12th Annual Symposium (INCOSE 2002), July 18 – August 1, 2002, Las Vegas, Nevada, USA.

http://www.incose.org/symp2002/

The Symposium theme "Engineering 21st Century Systems: Problem Solving through Structured Thinking" calls on delegates to discuss and debate the application of defined systems engineering processes to understand and solve the challenges that lie ahead in the definition and the design of new products and systems, the deployment of new technology, and the effective use and support of legacy systems that are still with us in this new century. The Symposium will provide a forum to address all aspects of 21 st Century problems and opportunities, the systems engineering principles, processes, methodologies and tools that can address them, and the resulting systems, products, and services.

Papers are invited that can make a contribution to the theme of the Symposium, within the framework of the traditional technical tracks and areas of special interest. All types of papers will be considered, including case studies, developmental work and technical analysis. Papers will be judged on a range of parameters including clarity of expression, effective communication of ideas, and technical content. Papers must be submitted in English, the official language of the INCOSE 2002 Symposium.

28th Euromicro Conference, Software Process and Product Improvement, September 4 - 6, 2002, Dortmund, Germany.

http://www.sea.uni-linz.ac.at/SPPI2002

In today's competitive world the quality of software systems is a key to economic success and stability. The Software Process and Product Improvement track concentrates on processes, methods, and tools improving the quality of software products.

Suggested topics of interest include, but are not restricted to:

- Software process assessment and improvement
- Organisational and business views to process improvement
- Quantitative models for development processes and products
- Distributed software development and virtual organisations
- Process and product improvement for e-business application engineering
- Use and usefulness of quality standards for software products and processes
- Approaches for modelling and enacting software processes
- Lightweight and flexible approaches
- Processes for component-based software development
- Verification and validation of software products
- Approaches improving dependability of software systems
- Industry best practice experiences and case studies in above areas

The framework of the track will contain sessions with a special focus. Proposals for special sessions or panels are welcome. Please send suggestions to the program chair. Special session titles will be posted on this web page.

IEEE Joint International Requirements Engineering Conference (ICRE'02 and RE'02), September 9 – 13, 2002, University of Essen, Germany.

http://www.re02.org

RE '02 celebrates two major milestones in RE

a) the 10th Anniversary of international IEEE RE conferences and symposium

b) the foundation of a new conference series, the IEEE Intl. Requirements Engineering Conference (RE) which results from the "unification" of the former IEEE Intl. Conference on RE (ICRE) and the IEEE Intl. Symposium on RE (RE)

The conference will include a technical and an industrial paper track, with refereed papers describing novel research, industrial problem statements, experience reports and surveys. The program also includes keynote speakers, state-of-the-art and practice tutorials, and an exhibition, which includes companies with RE tools and services, book publishers, and other related exhibitors. There will also be several associated workshops, including a doctoral workshop for PhD students, as well as research demos and tools. The RE 02 conference will provide an opportunity for practitioners and researchers to share ideas and experiences, while enjoying the hospitality of the University of Essen.

RE-Readings

Reviews of recent Requirements Engineering events.

RE'01, Toronto, Canada, 19-31 August 2001.

Report by Ian F. Alexander

The 5th IEE International Symposium on Requirements Engineering was, as you'd expect, a big event, held in the splendid surroundings of the monumental Royal York Hotel. Over 200 delegates, four keynote talks, dozens of technical papers, a separate state-of-the-practice track, minitutorials and panel sessions crammed into three busy days (not to mention two days of tutorials before the main conference) all made for an unforgettably enjoyable occasion. The only problem is that it is impossible to be in three places at once, so any report is inevitably a personal selection, and speakers are doubly disadvantaged by only getting to hear what they knew already rather than being able to listen to something new. Of course, much of the best talk happened in the spacious corridors as we gossiped over coffee in the breaks.

After a chaotic night in which the conference organizers (**Bashar Nuseibeh** and **Steve Easterbrook**) desperately rushed to help get a reserve keynote talk together, the scheduled speaker, **Pamela Zave** (AT&T Labs) appeared in the nick of time. She gave a fascinating talk on Requirements for Evolving Systems, a Telecomms Perspective, with beautiful handwritten slides. Telephone systems used to have just 3 nice simple interfaces: to enterprises, to other networks (e.g. abroad), and to ordinary users via the ubiquitous phone user interface. But since the 1960's features have been added one by one. This is nice – you can add anything – but it creates a terrible problem of feature interaction through conflicting goals, special cases and multiple possible behaviours. Zave led us step by step through the reasoning that led her to construct a formal method for 'feature engineering'. This means that requirements have to be understood via an abstract system architecture – it is easy to see why this is controversial. Her method is based on the old IBM pipes-and-filters approach to software composition, except that hers is channels-and-filters, and no-one knows the system requirements for telecomms.

Colin Potts (Georgia Institute of Technology), in one of the few technical papers I managed to get to, spoke about metaphors of intent. Humans habitually talk about machines as if they were people (see Reeves & Nass, *The Media Equation*) and use language of teleology (purposiveness) and anthropomorphism. Potts argued that we should choose metaphors suited to specific problems, and model our requirements accordingly. Concepts are based on resemblance (as Wittgenstein showed for the idea of 'a game' in *Philosophical Investigations*) rather than on dictionary definitions. Language doesn't convey meaning but affords it, as system functions afford capabilities to users. There are lots of grounded metaphors once you look (Potts referred us to Lakoff and Johnson's *Metaphors We Live By*) such as Important=Big, More=Up, Type of=Inside (causing all kinds of trouble with Venn diagrams!), Similar=Near, State=Place, and so on. People just assume these things so there isn't much real choice. Teleology is of course Human Agency – we always use human vocabulary such as 'demons', 'agents'. A commanded-behaviour machine is an operator or actor; a workpiece-editing machine is a scribe or secretary. Metaphor is everywhere. Terrific stuff.

Michael Jackson spoke as entertainingly as ever on formalism and informalism in RE. The rules of chess couldn't all be formalized: some decisions were explicitly left to the umpire. He logically examined the lyric "Everybody loves my baby, But my baby loves only me", concluding that "I am my baby" to laughter and applause. How do we know that's wrong? In an informal domain, denotations of terms are not exact;

no universally-quantified statement is ever perfectly true; and as the last statement was itself universally quantified, "the exception proves the rule" where 'proves' of course means 'tests' rather than 'confirms', i.e. the rule is tested by examining its boundary.

RE lies at the formal/informal boundary, so we have to make formalisations of informal domains. We can do this by having an explicit description subject, designations, unknown states, graceful degradation of descriptive truth (approximations break down gently), and appropriate problem/solution structures. A common error is to suppose that descriptions of the database are also true of the world; for instance, 'forall books there exists exactly 1 author' may be true in a library database but is nonsense outside it. The trouble with informality is unbounded relevance – you have no idea what facts may concern you: the fact that the moon is rising may sound irrelevant to the designers of a weapon, but if the moonrise sets off the early warning system, it becomes suddenly and horribly important. Classifying cases in the world that apply to the system is always going to be difficult; human judgement, like that of the stationmaster in a 19th century Punch cartoon deciding that the passenger's tortoise was an insect, and therefore could travel free, can resolve many system problems that elude formalisation. Formal reasoning based on formalisation can 'show the presence of bugs, not their absence' (like testing), but that doesn't mean formalisation isn't extremely important.

We had the great privilege of hearing **David Parnas** (McMaster), famous for his concept of 'information hiding', speaking on systematic documentation of requirements. We need complete, consistent and correct documents, as engineers of all types must certify their products safe to use and fit for purpose. Therefore, he said, "there are no requirements engineers". There were no audible gasps from the audience as he reduced our profession to a subtask.

He set out to do some upgrade work on an aircraft and found to his great surprise that there was no clear and readable specification. He put together one that 'passed the coffee stain test' – people really used it. 'One of the things that was good about our specification was that pilots found 500 errors' – it was accessible to subject-matter experts.

Parnas evidently enjoys teasing an audience; he gave us quips like 'Surprising how many thinkers use the Undefined Modeling Language', and 'Requirements in mathematical language are no use unless they're easier to read than the code'. But his real suggestion is that the traditional 2-variable model, namely that the outputs of a system are a function of its inputs, is false. It is complex to apply, as people like pilots know about the World, not System Inputs. A 4-variable model is better and simpler. In the World, you have a list of Monitored Variables and Controlled Variables; in the System, you have inputs and outputs. It sounds as if it goes well with Michael Jackson's approach. There are 2 specifications, so the pilots only have to read about things in the World.

Writing a specification comes down to two steps: list the variables of each type, and describe the relationships that held before you built your system, and those that should hold (what you want to achieve) with the system. Then you can check that all values are in the range allowed, and that they are consistent. Most formulae can be avoided by writing tables showing the different cases – you always find cases you overlooked when you do this, and the formulae often come down to simple values in each table cell: maths can be readable. You can then say clearly 'I don't know' (with partial functions and relations), 'I don't care' (any value in the range is OK) and 'I must find out' (you just say TBD).

Axel van Lamsweerde (Louvain) spoke about goal-oriented Requirements Engineering. The big questions were why – goals; what – requirements and assumptions; and who – people, devices, and software. Goals were used for everything – elicitation, elaboration, structuring, specification, analysis, negotiation, documentation, and evolution. One could reason about goals, once modelled. Goals made sense at high and low levels, and could be functional or not. Goals might require agents to co-operate; a workable definition of a requirement is a goal that can be assigned to a single agent within a system. At process level, goals could be owned by stakeholders, and their viewpoints could conflict. In the KAOS method, goals are decomposed into and/or trees, permitting top-down refinement. van Lamsweerde challenged Michael Jackson who was sitting in the front row: 'You don't seem to object'. 'You're standing up and I'm sitting down', replied Jackson, to laughter. Goals provided the right level of abstraction for the RE process; as well as goal models one needed object models, operation models, and so on.

Ian Alexander presented Richard Stevens' talk on systems engineering at the enterprise level. Many of the concepts of project organization and requirements applied above the level of individual projects. Crucial activities such as outsourcing, decision making, program management, and reuse needed to be organized across the enterprise. Existing tools and methods ought to be applied to these tasks. Forward-looking corporations were already starting to do so.

Don Gause (Savile Row LLC) spoke about the alligators in the swamp that he encountered in his consultancy work. All of them were, he swore, literally true. Here are a few of them.

> 'We've always done it that way'.
>
> 'I think that our functional specifications are mostly defined by the programmers at implementation time'.
>
> 'Don't bother me with the requirements, just give me the data elements'.

'We have no more time to spend on requirements. We must start development today'.

'We are speaking for the client'.

Gause was, of course, dryly humorous about these snappy beasts.

Brian Lawrence (Coyote Valley Software) spoke about rethinking requirements, and swore that he had not colluded with Gause in preparing his talk. He listed some popular misconceptions, such as that requirements were always required – no, you prioritise them; that the contents of the SRD were the requirements – no, they were in people's heads; that gathering and tracing were the key activities – no, it was negotiating them that mattered; that you can know all the requirements – no, they varied from fuzzily tacit to rock solid; that requirements documents served as communication tools – no, as with making toy aeroplanes, most of the value comes during modelling, not through later use. He concluded sadly that three quarters of the most serious defects in systems were caused by poor requirements: the defects couldn't be fixed except by redesign. There was a huge gap between what was known and what was popularly done.

I won't report on the **tools panel** here (see Re-Sources section - *Ed*); instead, I've made a separate report that collects up some short personal statements from all the tool vendors that I was able to interview.

The **Symposium Banquet** was just as splendid as you might imagine, and the food delicious. The young danced; the rest of us chatted far into the night, trying to solve Michael Jackson's ridiculously difficult puzzles in statistics and logic. (We did more or less solve most of them.)

Lucy Suchman (Lancaster) spoke about practice-based design. She engagingly admitted she had never been a systematic person, but in 20 years of ethnographic or anthropological work she had found that a lot of things that people took for granted were untrue.

She began by telling the wonderful tale of the Xerox photocopier, vintage 1983. It was marketed under the slogan 'Just push the green button' as it was so intelligent, but everyone complained it was complicated to use. She set up a video camera above the labs machine, and was rewarded with a famous film of computer scientists Alan Newell (a founder of AI) and Ron Kaplan (a computational linguist) trying and failing for an hour and a half to make the photocopy they wanted… Walking up to an unfamiliar artefact was a disorienting experience. No amount of system ingenuity eliminates the need for learning by users.

She described the problem of filing local authority planning documents so that they could be retrieved quickly. Why was filing so hard? Choosing a library category wasn't fun – there was lots of overlap. Using a uniform class catalogue forced conflicting constraints on filing. You needed multiple logics for coding; filing and retrieval had different concerns. A digital system could overcome these problems by permitting multiple methods of accessing a document and by allowing rework to improve the classification. The prototype proved popular but would take many years to replace the old system – no-one is going to re-file the entire archive. There were many outstanding research issues, such as distributed access versus 'call Dave and let him find it for you'; migration to online search; and what about the need for real signatures on paper? Standardisation and customisation of categories also pulled in opposite directions.

Innovation meant imagining what could be, based in knowledge of what is: it was much better to think by being in the world than in a white room.

Ian Alexander spoke about engineering use cases with DOORS. Systems engineering could benefit from scenarios at all levels, and for different activities, notably elicitation and specification. If you were using Use Cases, models therefore needed to be made to suit different purposes – more sketchy or more analytic; and going into different amounts of detail. A system specification model needed to go inside the system to show which subsystems were responsible for which steps: in other words, such models (like Jacobson's original telecomms use cases) had to be white-box as far as the system itself was concerned. Much of the confusion around use cases stemmed from the diversity of types of model that were needed. A template based on Cockburn's approach (for software) with minor modifications for describing systems could readily be handled in a requirements tool. A toolkit permitted automatic linking of included cases, simple metrics and export to a readable and navigable hypertext.

Tim Kelly (York, at the Rolls-Royce University Technology Centre) spoke about deriving safety requirements using scenarios. He presented a picture from the Civil Aviation Authority's admirable procedure for preparing a safety case. The safety life-cycle forms a parallel world alongside the usual development life-cycle, doing a complex dance with it as the user requirements feed into the functional hazard assessment, which in turn influences the allocation of functions to systems, and so on. Hazards could be classified tidily by looking at a function such as chaff release or reverse thrust, and asking simple questions like what would happen if the function went on when not required, or stayed off when it was required. You made a table of the resulting failure, the phase, effect, class (disaster, etc) and verification for each function. It was essential that the safety case functions were identical to those in the development life-cycle, so traceability (supported by a tool) was essential. If you modelled the system with use cases at different levels (dovetailing neatly with Ian Alexander's approach) then you could put hazard-handling cases in too at high level, and show at the level of subsystems what mitigating use cases would be needed. In the case of reverse thrust, for instance, which is an engine level function, a use case at aircraft level could be 'protect against wrong thrust reverser deployment'. This could be implemented at thrust reverser control – subsystem – level by automatically restowing the reverser if it deployed wrongly; that case needed to include another, to detect wrong deployment.

Mike Lowry of NASA, allocated the graveyard slot at the end of the conference, bravely spoke about RE and program synthesis, arguing that the rapid growth in complexity ('onboard software doubles every 3 years'). The number of errors was related to program size, and already numerous space missions had been lost through software errors, even though 'someone in NASA knew there was a problem'. Space software was in the worst quadrant – both tightly coupled and complexly interacting. His calculations predicted 0.4 mission-critical errors per 100k lines of code. Therefore drastic steps were needed to prevent disaster: managing complexity, better systems engineering, tolerating errors, and detecting errors were all likely. NASA's own motto 'better, faster, cheaper' implied higher reliability, in less time, for less money, which was impossible. Automatically generating programs using domain-specific models (of planetary orbits, etc) greatly reduced errors, but if you asked to see the date of the next new moon by specifying the time when the earth, moon, and sun next lined up, you'd get the date of the next eclipse!

RE'01 - another view.

Report by Elena Perez-Minana

From the aura of well being that seemed to pervade the attendees at the end of the closing down session of this year's RE-01, I personally feel that the fifth IEEE's Symposium on Requirements Engineering was a rousing success! The main aim of the event, which is, to bring together researchers and developers from both academia and industry, to present and discuss their current research results and experiences in the area, was definitely met.

The conference is mainly sponsored by the IEEE Computer Society. This year, it was planned and brought to life through the joint effort of the ACM, SIGSOFT, IFIP WG 2.9 and INCOSE. It brought together people from around the world working on Requirements Engineering (RE). The Royal York hotel, in Toronto, made a very nice setting for all the buzzing that was produced during the 3-day event.

The conference was preceded by two days packed with a series of half-day and one-day tutorials, each of which dealt with some key problem in the area. The quality of both the tutorials I attended, i.e. "Problem Frames" run by Michael Jackson, "Requirements-based Product Line Engineering" by Mike Mannion and Hermann Kaindl, was very good. In both cases, the points covered, the speakers, and the insightful questions raised by the participants, made perfect breeding ground for interesting, lively discussions.

Browsing through this year's technical program, NASA seems to be holding a very strong position in RE, if the number of presentations conducted by its researchers is anything to go by. An additional point in their favour, was the quality of the work and the high level of the presentations.

Complementing the technical program were four outstanding featured speakers. Pamela Zave, Technology Advisor from AT & T Laboratories, opened the conference by examining the need for an architecture that facilitates the definition of rapidly evolving systems in the context of telecommunications systems. Gene Spafford, professor at Perdue University, enlightened us all as he spoke of the number of non-obvious aspects of security - including privacy, auditability, and assurance - that are usually overlooked in the requirements capture process and which are increasingly important in today's way of working. Lucy Suchman, professor at Lancaster University, provided an interesting perspective on the benefits of taking an ethnographic approach to the task of identifying the components that make an effective information system. Michael Lowry, head researcher at NASA Ames Research Centre, gave a very interesting talk on the potential benefits of an integrated life-cycle framework, how it provides better support for three crucial aspect of software development: failure detection and diagnosis, change management, and software reuse.

RE-01 also attracted a number of RM tools from throughout the world who assembled in an exhibit hall adjacent to the conference venue. It was extremely useful to be able to switch at any time between two so different and at the same time complementary environments, i.e. one providing potential practical solutions to certain RM problems, the other a hive of exciting new ideas with good potential for providing the means to solve even harder problems.

The program also included a series of Panel Discussions, which allowed plenty of room for energetic exchange of opinions. The one I attended was the "Exhibitors Panel Discussion"; it was comprised of some of the tool vendors that were in the exhibition. Representatives from Telelogic, Rational, Future Tech Systems, Qualisoft, SDRC/TD Technologies, Structured Technology Group and TCP made up the panel. There was agreement among the panelists and the audience that the speed with which software evolves has also affected the availability of RM tools and the features they incorporate. Another issue in which there seem to be general agreement was the fact that it is not difficult to encounter a RM tool that is not much more than a colourful UI. Such tools do not provide much in terms of effective support for the management of requirements. Nevertheless, the competition is fierce, and as we acquire more experience in the evaluation of this type of tools, it will be easier to detect the phony ones. At the same time, it is necessary to remember that the success of deploying a tool in an organisation is very dependent on the organisation itself, its processes and its resources. This means it is a hard job to strike a final fine balance.

Finally, many organisations must admit that part of the problem in succeeding at installing a RM tool in their organisation lies in the poor quality of the requirements because it is extremely difficult, or rather impossible, to manage poorly written requirements with a RM tool. As the tools workshop was advertised at such short notice, and given the very full, initial program of activities, it was very poorly attended which under-valued a good opportunity to discuss first-hand practical experiences with suppliers and consumers on the issue of use and deployment of software tools.

Nevertheless, the learning opportunities were even more plentiful. In addition to the technical sessions, the panels, and the keynote talks, a series of well-known experts gave excellent state-of-the-practice talks.

The major areas of concern in the RE arena at this moment are:

- the deployment of requirements for Product family development. The correct handling of the naturally high level of variability and uniformity.
- the close relation there is between testing and requirements. It was very clear, that they go hand in hand, and are both very important if we are at all interested in producing high quality results.
- The growing interest in the application of measuring techniques to obtain information on how the product development process is managed and how it can be improved.

The evening events organised by the Committee constituted an excellent final touch to each day and also to a very stimulating and rewarding week.

Requirements for Mobile Systems, University College London, 21 November 2001.

Report by Ian F. Alexander

Wolfgang Emmerich (UCL) introduced the event saying it was timely after the sale of 3G licences by the government, and now that research into the area was starting in the UK.

Andrea Savigni (UCL) spoke about a Requirements Engineering Framework for Ubiquitous Web Applications or 'UWA'. The long-term research goal is to provide a framework for this kind of work in context-aware services, meaning that a mobile device can adapt its performance according to its environment, whether that means being in the mountains, or being close to a printer or computer or other devices. This is ambitious.

Problems for RE in context awareness are numerous: changing location, bandwidth, display characteristics, usage paradigms, platforms, and short time to market. So RE must be quick and effective as requirements change.

The work is influenced by van Lamsweerde's work on goals, Michael Jackson's World/Machine distinction, Maes' computational reflection (where a program reasons about itself), and Fickas & Feather's requirements monitoring (where a system looks to see if any requirements are being broken). The short-term motivation of UWA is an EU project between Politecnico di Milano, Linz, Crete, and UCL with industrial partners.

The approach aims to make multi-channel applications easier to maintain, without having to build new applications for every new device, and releasing versions of each release for each device – obviously a combinatorial problem. Therefore you need one flexible design that can adapt itself via so-called customisation mechanisms. Industrial progress is more limited, e.g. Oracle 9 Wireless Edition and Cocoon.

Applications are determined by Requirements with operationally immutable Goals. The Environment is monitored by a Context, which influences the Requirements. Applications provide and monitor Services (am I meeting the requirements?), which are constrained by the Environment. Therefore the requirements are part of the run-time system, as well as being fed into the design in the usual way. Changes at run-time actually change the requirements with the Context! This in turn yields a customisation, hence the architecture must itself be represented at run-time. Where should such knowledge reside? – clearly not on every device on its own, except for device-dependent knowledge. Middleware is needed to describe services.

UWA itself will focus on user goals with no formal use of KAOS. It will map requirements to both design elements and to customisation rules. Web designers are 'allergic to architecture' – there is just a client and a server! They are 'artists' so they want to recycle everything and have many conflicting viewpoints. UWA therefore used web-related requirements categories such as Content, Structure of Content, Navigation, Access and so on – very unlike conventional software engineering. But the combinatorial explosion cannot be prevented just by dealing with requirements – there are needs for many applications on many devices. To make things portable across platforms, you need customisation rules to say cut down on the colours, use no graphics on this device, or whatever. You can then have a small set of requirements which you combine as needed with device rules.

Elena Peres-Minana (Philips) asked if requirements really needed to change at run-time. Savigni said that if you didn't you lost track of them. Michael Jackson wondered whether the Feature Box model of telecom systems would solve the problem. David Bush asked if the work meant the transparency assumptions of the ISO 7-layer model were broken. Michael Jackson also asked whether the meta-requirements were (therefore) the real requirements? These were all good questions.

Chiara Cardamone (Unipower and UCL) spoke about Requirements for M-Shopping on Wireless Devices.

Unipower produces the TescoDirect shopping solution, part of which is based on a Palm portable device.

Consider the scenario of a busy manager on a train, wanting to do some shopping. He uses his Personal Digital Assistant (PDA) to browse, select and order the goods he wants. The user profile clearly means a shopper who has a Wireless Information Device of some kind, and specific needs that preclude normal shopping.

In E-shopping, only 35% of transactions end in a sale – obstacles arise at each stage, causing some users to drop out. Why do people want E-shopping? Because it’s fashionable; quick; easy; enjoyable; and fun (like web shopping). But fears corresponding to all these reasons also exist.

Constraints include limited processors, memory, battery power, graphic definition, display size and poor reliability of communications – lines are noisy and can go down. These are quite formidable problems.

The requirements are for something intuitive, short, enjoyable, and with good feedback and clean layout with no distractions. User Interface design is crucial.

Devices include PDA’s with Tablets, Keyboard Devices (pocket computers), and Smart Mobile Phones. The project therefore has made 3 prototypes corresponding to these classes of device.

On a Pen-type device (Palm), a prototype showed what the UI might look like, with no database interaction. The user clicks through a list of options such as Bakery, Beverages and Frozen Foods. Easy icons at the top indicate food, payment, favourites, shopping trolley, checkout. It is quick to select Bakery, Bread, specific bread type and then size and quantity, with products in alphabetical order. Shoppers can enter a shopping list with short descriptions of what they want; if matches are found, lists of matching products are displayed. It really does look very easy. But handwriting on a Palm tablet is slow and error-prone, so pop-up lists are preferred.

Another prototype illustrated Collaborative Shopping to prepare a shared shopping list. The prototype appears on a mobile phone; the user scrolls through lists on the tiny screen. Clarity is of the essence.

Why not use voice recognition, someone asked. Because the environment is very noisy, as on a train, and menus are economical on limited devices. A Palm PDA with 2 Mbytes of RAM would be unable to handle voice recognition, while doing the job on a server would be costly and vulnerable to noise.

Marco Corsaro (IMIWeb) spoke about Requirements for Mobile Stock Trading. IMIWeb allows direct access to all the major world stock markets.

Access is provided through 3 different types of device: WAP phones, laptop computers and PDAs. As with the previous talks, this means a multi-channel system architecture, which is a common server with an Oracle database and an Apache web server, and both wireless and fixed wire access. Applications run on an ATG Dynamo application server to generate HTML, with a Tarot 2000 back office behind the database. Applications change frequently.

Customers already buy shares over the slow mobile channels that exist today; broadband will bring marked improvements. The speed is the same whether you use a laptop with a wireless card or an iPAQ sitting next to a mobile phone.

Corsaro bravely gave a live demonstration using his iPAQ and mobile on an epidiascope. The screen shows today’s most traded stocks and a graph of the market. The search tool lets him search for a company by name or symbol. Brokers’ news feeds are available via hyperlinks. Trading is accessed via user ID, password and your place of birth. It takes some seconds to get authorisation. The interface is created with JHTML, very simple but effective. There are hyperlinks, labels, and boxes to fill in as well as tables with colour-coded backgrounds to the cells. He sees he will make a £2 profit on his Orange shares so he sells. It takes a few seconds for the order to be confirmed by the iPAQ, complete with commission (£15 – a flat rate). The connection duly goes down, so we have to check that all is well: he goes back to the screen where he made the trade, which was still on the iPAQ – an essential feature. It wasn’t in fact sent, so he sends the order again and this time a server confirmation is received. Sessions time out after 15 minutes in any case. The mobile phone screen also confirms with a Short Message that the order is OK.

IMIWeb offers the only site that provides such versatile trading: it is all server-based so it is highly independent of the mobile device type actually used.

Elena Perez-Minana asked about the requirements for such systems. Corsaro said the requirements were very simple – just buying and selling shares – and all they did was to resize the user interface to handle small screens. There were very few wireless trading customers today ‘and we probably know all of them’. When the connection is fast and reliable there will be many more. People use laptops and modems at the moment.

Cecilia Mascolo (UCL, CS) spoke about Non-Functional Requirements for Mobile Systems. Mobility itself just means ability to change something’s location, reconfiguring it; dynamic mobility means in addition moving it while it is running. Mobility makes things dynamic – location and context (other entities and services within reach); and makes some resources scarce, notably battery, memory, and display. In addition, bandwidth is low, reachability is limited, and connection is slow and expensive, with frequent disconnections that are so common they can’t even be thought of as exceptions. Short-time connection is the pattern. All of these make mobile services very different from fixed ones.

Fixed applications do not need to know where hosts are, so middleware can conceal such details; but in mobile systems, applications can work better if they know the context, so (as discussed in earlier talks) transparency needs to be given up.

Similarly, connectivity cannot be assumed to be stable and continuous. Synchronous communication isn't satisfactory. Users need to be able to disconnect (voluntarily to save money, or involuntarily if the line goes down – e.g. the train goes into a tunnel) and work off-line in most cases – unless (as with share trading) absolutely current data are vital.

In 'Nomadic' computing, the situation is between traditional and fully mobile ad-hoc: hosts are rather static, but clients migrate.

A fully ad-hoc system is totally wireless with everything moving – even the basestations can come and go. This can happen with battlefield systems, emergency services such as dealing with major disasters, and few other currently conceivable situations. With no fixed infrastructure, clusters can form dynamically, and messages must be routed by any available path. But if there is open wireless coverage then security is an issue. Existing work is focusing on making middleware lighter; on asynchronous communication using tuple space – you just put your message in a bag, and anyone who can handle it does so; and on data replication, caching, and reconciliation which allow for weak consistency (e.g. using possibly old data in the railway tunnel and correcting it later).

The future must be in allowing for adaptation to the Context, e.g. for varying bandwidth; for disconnected operation; and sacrificing transparency to improve performance by making system behaviour depend on the current application. Much more user interface work is also needed. What are requirements in such an environment? Do we need them at run-time with meta-level representations to keep track of context? Mobility introduces a basic kernel of new roles and non-functional requirements that we will have to deal with.

The event ended with a speakers' panel session. Wolfgang Emmerich said that anyone who doesn't recognize that low bandwidth (by orders of magnitude worse than fixed networks) and disconnection are part of the mobile world will fail. Bashar Nuseibeh said that security is much more application-specific. Wolfgang Emmerich replied that you can't rely on normal things like firewalls: the environment is different. M. Jackson (not our patron) said most of the requirements for mobility were not special – bandwidth will improve; but it is necessary for hosts to know when clients are mobile, e.g. for marketing reasons. Applications don't need to care what they are running on. But Andrea Savigni argued against this: it may help to tell an application that power is scarce, and it can then omit graphics or whatever to save on power. Michael Jackson said that of course bandwidth and reliability will improve, but – as with PCs – demands on these resources will also rise. Mobile systems will always be quantitatively and qualitatively different. Wolfgang Emmerich said that task modelling and user interface prototyping will become more important, altering the balance of techniques used in RE. Bashar Nuseibeh said that perhaps the process was therefore rather different.

Wolfgang said that you needed to have requirements at run-time as you didn't know what you would need until then. Michael Jackson said yes, but there was a paradox in that. You don't want to defer what you ask for until run-time! Andrea Savigni said that you did want to change the behaviour of the application depending on its context. Michael Jackson said the context was surely a variable, and you had to compute the desired behaviour from that: Andrea was using 'requirement' to mean 'the purpose of that computation' as well as 'the result of that computation'.

The audience was a good mix of academics, from students to professors, with industrial consultants and research workers. Everyone thanked Wolfgang Emmerich for an excellent meeting.

RE-Papers

Being Clear: Requirements are EITHER Needs OR Specifications

Ian Alexander

For many years now, every systems engineer has learnt that it is essential to write requirements for every system. These, we were told, had to be separated into User and System requirements. Or perhaps we learnt to write Functional and Technical requirements, which seemed to amount to much the same thing. And when it came to verification (on the right-hand or rising limb of the 'V-model'), there were similarly two sets of acceptance tests, which seemed at best rather confusing.

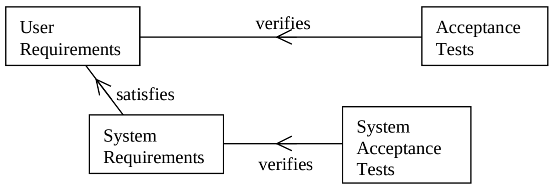

Figure 1. A Confusing Information Model

Either way, we had in each case to produce two documents, both of which seemed to be overheads on the plain tasks of designing the system, coding the software, and testing it. I well remember the shamefaced look of a respected colleague many years ago as I accidentally caught him sticking back the write-protect tab on an old 5½" floppy: it was the requirements archive disk, and he was performing the necessary task of keeping the requirements consistent with the code, while also complying with the (impossible) demand to freeze the requirements before starting the design. A write-secretly tab was the only solution.

My aim here is not to add to the criticisms heaped on the naïve waterfall model (basically the left-hand, descending limb of the 'V') – it was a necessary stage in the evolution of a workable development life-cycle. Instead, I want to look at the first two steps of the life-cycle, and see what they are meant to achieve.

Specifying Systems

The system requirements are probably clearer in intention, so let us begin with them. They specify what the system is to do, and as everyone knows, they are not supposed to say how the system is to achieve those results – that is the job of the design. Incidentally, they do not consist only of system functions. It is equally important that the system complies with safety standards, or provides a given quality of service. No-one has ever succeeded in getting a universally agreed name for these extra things – Constraints is about the nearest – so we have to make do with rough-and-ready terms like Non-functional requirements or the informal '-ilities' (such as reliability and maintainability), or the loose 'Quality requirements'.

Assuming, then, that systems are specified by a combination of functions – the system enables someone to get such-and-such a result, and constraints – the system is unavailable for at most such-and-such an amount of time per year, then you might think that the job of specification was complete. But no; we are also supposed to write a user requirements document. As Ben Kovitz has so wittily said, this has all too often led tired engineers to write as follows:

> User requirement: The user shall be able to do this-and-that.
>
> System requirement: The system shall enable the user to do this-and-that.

Well, you don't need to be told (do you?) that a lot of time and paper is being wasted if your organisation is duplicating the requirements in that way.

Defining Business Needs

What, then, are user requirements for? They can't be meant to specify systems! Instead, they should be saying what business users need, leaving the question of how those needs might be met. This should sound familiar: it is similar to the job of specifying a system, leaving aside for a moment the issue of how the system should be structured. In other words, the needs of the business should be defined as a problem that a system can solve.

Why do business users need something? They are evidently trying to follow some set of business processes in their work, and discovering that some of those processes are difficult to implement. Therefore the logical starting point for any statement of business needs is a business process model, comprehensive enough to cover the whole of the problem area – you can never model everything, and there is generally no reason to try. If you don't know what the processes in the problem area are, the users's needs will appear as a disconnected bunch of isolated complaints, or a confused wish-list.

When you have a simple (or even a simplistic) business process model, the users' needs will click into place: first he needs to do this, then they need to do that, and so on. The same goes for the business needs: when you have a simple and clear set of stories – first I want to do this, then that, then the next thing – then the system specification will be easy to understand, because each feature makes sense in terms of what the people need.

Different Kinds of Requirements

So, requirements mean very different things in different places:

A Business Process Model says what a business does. Such models are not requirements, but they provide a context for understanding requirements of all kinds.

Business Needs say what a business wants to be able to do, but currently can't, or at least, can't do easily, quickly, conveniently, and economically (this is where the constraints come in: ultimately they come from the business, just like system functions).

System Specifications say what a system must do, both in terms of its functions and in terms of the qualities or constraints that govern the delivery of those functions.

It should now be clear what was wrong with making the User and System requirements virtual copies of each other: the documents should be talking about business problem and system solution. All too often they both talk about the solution. This isn't surprising: engineers who work for a company that designs and makes systems, and who are only allowed to look at solutions, do not know the business problems and are not in a position to write user documents.

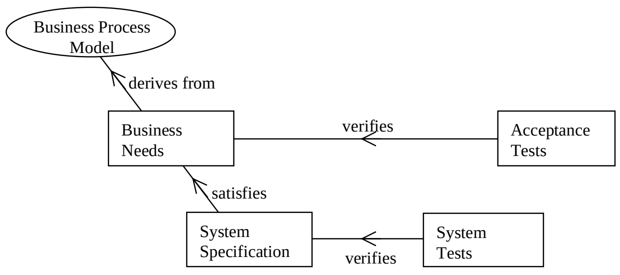

Figure 2. An Unambiguous Information Model

User or Business documents belong to businesses, not to engineers. If the problem domain is insurance, then the business needs must talk the language of insurance, and describe the insurance problems that the insurance workers are experiencing (they don't become 'users' until they get a system to use; the name 'users' betrays a system point of view). To system people, insurance people are ’[problem] domain experts’ as opposed to themselves, people expert in the solution domain.

Requirements engineers form a bridge between the worlds of people who use and people who make systems. We are in the curious position of scarcely ever owning the documents we handle – they belong either to one camp or to the other. The key task is to enable the needs to jump the divide. After that, all the tools and techniques of requirements management can be applied to try to ensure that the needs are fully satisfied: but all the effort is wasted unless the problem is correctly defined and communicated in the first place.

Requirements are EITHER Needs OR Specifications

It is therefore crucial to separate out the job of defining the business need from the job of specifying the system. Both jobs are loosely called 'writing requirements' but I suggest that this dangerously glosses over two radically different activities: defining what people need, and finding a solution. If you want an unambiguous way of saying what should be going on in a development project, choose terms that make clear that problem and solution are different: Business Needs and System Specification, for example. Here is a redrawn and hopefully clearer version of the top levels of the life-cycle.

Finally, consider the purposes of the high-level verification documents. The acceptance tests are to demonstrate that the business needs have been met. The system tests are to show that the system works as specified. If the system specification exhaustively satisfies the business needs, then you could assume that showing the system worked would automatically show that the business needs would be met by the system. This would be a brave assumption. But that – and the whole problem of requirement traceability – is another story.

References

Ben Kovitz, Practical Software Requirements, Manning, 1999

Diagrams Toolkit (for editing and navigating Information Models)

http://www.scenarioplus.org.uk/download_diagrams.html

Requirements by Apprenticing

Julie Dobson

We collect data about our assets as a result of regular inspections. To give an overview of this a pole which supports an overhead line has approximately 18 condition points- which include items such as the pole itself, the stay, warning signs etc. Each condition point has a number of checks against it, which the engineer has to report.

As part of requirement analysis for hand held technology to replace existing paper data collection sheets, I decided to have a field visit with one of our engineers to experience what actually was the process for collection of data of our assets. I was instantly shocked by the amount of data recording sheets the engineer had to carry to cover a line of 20 poles - 40 sheets of A4 paper.

It was a cold and windy day and the route we were taking took us across mainly agricultural land. Therefore we needed the wellington boots and waterproofs. With work in hand we started on our route. As the first pole was found the engineer retrieved the recording sheet and started his job of reporting against all the respective condition points. These were mainly tick box type responses but any exceptions required him to give an explanation in shorthand. This continued throughout the day and when we had completed the work, we took the rather wet and soggy reporting sheets back to the depot.

I thanked the engineer for the experience and then went to join him for a well-earned cup of hot tea in the rest area.

I had gained enough information that day to identify the key requirements for collecting data via an electronic solution - or so I thought I had.

As I was leaving the site the engineer called me over and pulled a little green book out of his top pocket. He told me that on a normal working day he would not take out the 40 sheets of paper. He carried a small notebook which he used to record exceptions only. When he come into the depot in the evening he would go through the sheets and fill them in.

What I had experienced that day was the *formal* process for doing the work. In reality the engineers had their own process.

"Cunning Plans": Some Notes on Plans, Procedures and CSCW.

Mark Hartswood, Rob Procter, Roger Slack, Alexander Voß, John A Hughes, Karen Clarke and Mark Rouncefield

The interesting thing about ’cunning plans’ is often that any cunning that is associated with a plan is entirely the product of shrewd and devious human activity rather than attaching to the plan itself. It is only in the working out of the plan that it becomes, as Blackadder might say "as foxy as a fox that has just won first prize in the foxiness competition". The relevance of plans to human action has been famously outlined by Suchman (1987) in her book ’Plans and Situated Actions: the problem of human-machine communication’ which presents a powerful critique of the user modeling and planning-based approaches to design. Suchman suggests that: *"plans are resources for situated action but do not in any strong sense determine its course"*. Her central argument is that plans do not thoroughly determine *in advance* and *causally direct* in every detail courses of action. A plan is an abstract construction that needs to be to be applied in specific circumstances. Plans do not simply 'execute themselves' nor is the relationship between the plan and the action it directs a mechanical one. Plans are *accomplished* activities. This argument has been a feature of CSCW research almost from its beginning as numerous empirical studies have highlighted the gulf between abstract idealised plans and their situated accomplishment - perhaps most noticeably in Orr's (1996) accounts of the 'war stories' of photocopier technicians. As Selznick (1948) argues: *"The formal administrative design can never adequately or fully reflect the concrete organization to which it refers, for the obvious reasons that no abstract plan or pattern can … exhaustively describe an empirical totality. At the same time, that which is not included in the abstract design … is vitally relevant to the maintenance and development of the formal system itself."* (p25) At its heart lies both an observed empirical reality about organizational life and a desire to avoid underrating the skills and competencies that are required in even the most routine of tasks. From this viewpoint 'routineness' is observed to be an accomplishment produced through the everyday, practised exercise of complex skills. Such an approach involves an acknowledgement of the 'practical rationality' - what Boden (1994) calls 'local logics' and 'local knowledge' - exhibited in day-to-day conduct which, in turn, problematises 'idealised' notions of rationality.

This perspective upon plans and action had particular significance for the office automation movement of the 1970s that viewed clerical work as simply 'routine', involving the repeated execution of planned procedures and thus a prime candidate for automation. Typical of this research was a characterisation of offices according to the levels of routineness they were said to display and characterisations of clerical work as predefined 'reactions' to inputs: *" Once a clerk is told about a situation, s/he can consult a predefined procedure (formally or informally) to determine what action should be taken by the organisation. The organisation does not rely on the clerk to decide what to do; instead the organisation provides a procedure which instructs the clerk how to react to the situation."* (Zisman, 1977). The initial focus of this research was the analysis and modelling of work processes as a means to inform the construction of information systems. The Office Procedure Specification Language (OPSL) (Zisman, 1977), for example, attempted to provide a means to describe office procedures as a collection of activities, documents and messages and to model features such as the pre-conditions for any task initiation, the temporal orderings of events and the flow of information objects. The problem that most effected office automation research was the problem of exceptions. Various researchers demonstrated the rich and complex nature of supposedly 'routine' activities and the skilled and cooperative decision-making and negotiation necessary to 'get the work done'. There appeared to be important discrepancies between the formal office procedures that supposedly governed office work and the practical action as actually carried out by office staff. This was demonstrated by focusing upon how workers creatively solved 'exceptions' and dealt with contingencies. However, these field studies also showed how it was often necessary — in order to get the work done at all — to deviate from plans and improvise or 'ad hoc' procedures in the light of the exigencies of some unfolding situation. Consequently, as a result of her ethnographic studies Suchman (1983) is able to suggest a radically different sense to 'routine' and illustrate the importance of an ethnographic orientation to the status of procedural plans — an orientation which sees them as accomplished products. As she notes: *"the procedural structure of organizational activities is the product of the orderly work of the office, rather than a reflection of some enduring structure that stands behind the work"* Suchman (1983).

The problem was that office automation developers had assumed that they could take formal models of an office (such as standard procedures) to model the flow of information and embed these models within computer systems. *"The Office Automation research ran into problems because it embedded models of work in systems as if they were computer executable versions of what actually happened, of how work was actually done. The status of these models was transformed from being a resource —a resource which may provide a reference point, a grounding, a basis for discussions, a coordination mechanism and so on— into being a constraint upon how things could be done. Office automation systems have not had the impact or acceptance that was initially expected because such systems implemented an information flow that was idealised and neglected the work needed to make the 'flow' possible in the first place"*. (Pycock 1999) The work of Suchman and others illustrated how important it was to consider the 'fit' of these models with the ways in which work was actually done. The problem they revealed was that of 'automating a fiction'. It is people that do the work in organisations, not idealised models. It is the everyday judgement of workers, in interpreting and improvising standard procedures, that gets the work done and *makes* it routine. This experience has led to a greater awareness of the issues facing those hoping to model cooperative work: *"Models cannot capture all of the exceptions, and furthermore should not be based upon the premise that this is possible. Keeping in mind that models are limited abstractions of reality, it must be expected that procedures are open-ended with inherent escape hatches to handle unanticipated exceptions and emergencies. "* (Ellis, 1983). A focus upon supporting work with resources rather than automating it has become a distinctive characteristic of CSCW influenced by this understanding of the status of model and plan representations. As Schmidt (1997) writes: *"In a way CSCW can be said to have been born with these concerns. The office automation movement had already given way to disillusionment, and artificial intelligence was increasingly being confronted with unfulfilled promises. At the same time, a number of critical studies had demonstrated that the problems were deep rooted: office procedures were of a different nature than presumed by the protagonists of office automation. The general conclusion of these studies were that such constructs, instead of dctermining action causally, serve as 'maps' which responsible and competent actors may consult to accomplish their work. Thus, Lucy Suchman's radical critique of cognitive science 1431 and the 'situated action' perspective she proposed has played a significant role in defining the CSCW agenda and has become a shared frame of reference to many, perhaps most, of us. For good reasons, then, dcsigners of CSCW systems have been advised to treat them with great caution."* However, within this general characterisation there are some important issues as well as some misunderstandings that need consideration. Much of the critique of the planning model in CSCW, namely, that plans are ineffective, arises from misinterpretations of Suchman's work, in particular the idea that people do not follow plans because, in actuality, they are *post-hoc* rationalisations of courses of actions. Suchman does accord an important status to plans as resources for the conduct of work, arguing that plans are resources that guide the sequential organisation of activity, rather than laying out a sequence of work which is then blindly followed. Suchman's critique is directed at the notion that there are mental plans that operate as *causal determinants* of subsequent courses of action. The essence of Suchman's critique is that the plan is an abstract construction that will, at the very least, require articulation with, and application to, the specifics of the circumstances in which it is to be followed.

As the application of IT becomes more entwined with the complexities of organisational working, so the challenges facing IT systems designers increase and recent empirical research continues to emphasize the potency of Suchman's original insights. A study by Voß et al (2001) of production planning and management a manufacturing plant producing mass-customised diesel engines illustrates how these activities are subject to various 'worldly contingencies' and how the production process emerges from situated and resourceful activity. The findings support a contingent view of production planning and scheduling. The contingent view emphasises the incompleteness of knowledge and the arbitrary, uncontrolled or unanticipated circumstances that affect action. The implementation of a production plan is a production worker's formulation, produced in response to issues concerning the 'local logics' of day-to-day production management. This is done in the knowledge that workers may be required to account for a decision, or make a case in ways that can be seen and understood as manifestly complying with production objectives and rules. In this sense, as Suchman originally argued, production plans are less a device for directing production than a template for accounting for it.

Underlying much of the current work on production planning and management systems is the notion that to achieve the prescription of a task everything must somehow be rendered uniform and predictable. This pursuit of uniformity manifests itself in numerous ways. Yet any attempt to see this as simply following the script is wholly unwarranted. If, as is generally implied, the aim of production planning technology is to embed knowledge properties in systems, then production knowledge needs to be captured and managed in a way that will make it accurate, available, accessible and effective. Such a task is hardly trivial, and documenting the divergence of plans and actual production is not being critical of the *principle* of planning. Instead it points to the investigation of the subtle but essential competencies involved in making sense of (and thereby being able to make it available to others) the practical, here-and-now implications of a production plan. Supporting production work in all its contingent aspects, we believe, re-quires that planning systems pay attention to the occasioned character of the logic of production. This is not constituted as mastery of the organisations processes and procedures, but in whether, when and how to deploy these more standardised forms in the routine accomplishment of the work in hand.

Some of the misunderstandings and misrepresentations of Suchman's work are detailed (and perhaps instantiated) in Schmidt's (1997) paper 'Of maps and scripts: the status of formal constructs in cooperative work'. In what is perhaps best seen as an extension and exploration of Suchman's work, Schmidt suggests that the role of formal constructs in cooperative work remains misunderstood and that few CSCW researchers have attempted to adequately address this issue: *"the prevalent understanding in CSCW of the status of formal constructs in cooperative work is problematic. The empirical evidence for the received understanding is not as robust as we may have believed and there is evidence from other studies that indicates that formal constructs are not always as feeble and ephemeral as we may have taken for granted*". He points to the success of various workflow technologies to suggest that *"the role of formal constructs is more differentiated than generally taken for granted. They not only serve as 'maps' but also as 'scripts'"* and that: *"Instead of merely observing in case study after case study that procedures are impoverished abstractions when confronted with the multifarious and contingent nature of practical action; it is necessary to investigate precisely how they stipulate the articulation of cooperative work, how they are interpreted and used, designed and adapted by competent actors 'who have to live with them from day to day'"*.

Schmidt also points to some of the methodological issues that arise from ascribing 'ceremonial status' to constructs such as plans. He suggests that much of the debate implies that members of the organizational settings take formal constructs literally - as if constructs such as plans are supposed to be exhaustive specifications for doing work. When set up this way it hardly comes as a surprise when empirical findings suggest otherwise since, as Bittner points out: '*..rational schemes appear as unrealistic normative idealizations only when one considers them literally, i.e., without considering some tacit background assumptions that bureaucrats take for granted.*' Schmidt further argues that "*the putative contradiction between the formal constructs and the actual practice of the engineers may be the investigator's own construction and that the design of these constructs presumes the observed practice*". But it is questionable whether this, the contrast between formal and informal action, is the main aspect to which Suchman draws our attention but the problems of embedding such features in systems and expecting them to implement themselves. In his paper Schmidt proposes continued detailed empirical investigation, suggesting that there may be differences in the use of formal constructs both between small and large-scale settings and between routine and non-routine work activities and settings. This, Schmidt suggests, limits the extent to which we can generalize about plans and situated action.

Schmidt goes on to consider Suchman's study of the accounting office and her argument that procedures rather than shaping action provide criteria for judging how work should turn out at the end of the day. He argues that; *"This interpretation of the case is not supported by the published data. The study presents an analysis of a recover from breakdown. It does not attempt to demonstrate that prescribed procedures do not* - *in some form and to some extent* - *determine the handling of routine cases; it does not even attempt to give an analysis of how prescribed procedures are used in routine cases…. the authors do not take into account the fact that the situations studied are beyond the 'jurisdiction' of these constructs, that is, beyond the operational conditions for which they had presumably been designed"*. Using the examples of the checklist and the kanban system, Schmidt suggests that other studies lead to rather different conclusions as to how formal constructs are used by actors in everyday work activities. Rather than being Suchman's 'maps' such formal constructs act as 'scripts' - a precomputation of interdependencies among activities that, at critical points, provides instructions as to required next steps. *"The kanban system thus determines action in a far stronger sense than the map of a traveler determines the traveler's movements "*.

Despite these disagreements on method, the possibilities of generalization and level of detail, both Suchman and Schmidt point out that 'following the plan' often involves more than can be specified within it. The construction and use of plans in 'real world, real time' activities do not typically involve the supposition that literally everything must be spelled out in minute detail. 'Practically' and 'characteristically' plans are 'recipient designed', that is, spelled out to an extent to which those who are to follow them are, for example, familiar with the circumstances in which they are to follow them, sufficiently trained in the tasks involved, and a host of other possible considerations. Nor does the making of plans indicate any expectation that the course of actions which they specify will, of necessity, follow through. Indeed, the point of plans is often precisely to direct courses of action to maximise the chances that these courses of action will ensue despite the contingencies that can arise. Thus plans often include 'fail safe' devices to cope with situations where things are 'not going to plan' by specifying arrangements for adaptation of the plan to exceptions, unforeseen circumstances, even extensive revision, as well as mechanisms to oversee the implementation of the plan and enforce its requirements

Plans and procedures develop and are modified, unfold, in real time. What the plan agreed, what interdependencies there are, may only become clear as the courses of action specified in the plan unfold, creating additional workloads in terms of coordination and the awareness of work. The successful accomplishment of a 'plan' is consequently dependent on the practical understandings about what the plan specifies in *these* circumstances, using *these* resources, *these* people, and so on. Although plans may be presented as abstractions, as manuals, as statements of procedures, and so forth, the 'just what' it takes to realise them is a practical matter of 'making the plan work' through all the various and inevitable contingencies that can arise. It is such activities which maintain the plan by dealing with 'those things which arise', 'the things not planned for', the 'things which suddenly come up' so that even 'deviations' from the plan can be accommodated to sustain its 'spirit'. Suchman's work, by placing plans, procedures and decisionmaking within its social and organisational context, presents plans as *elements* which enable workers to make sense of both their own and others work and to come to a decision about future courses of action. What this emphasises is the importance of seeing how and in what ways plans and procedures are interwoven into a highly variegated set of phenomena that make up the social organisation of work.

Another argument that derives from Suchman's work on the problematic status of plans, hinges on the 'plan in the machine' and the 'essential incompleteness of instructions'. As Suchman describes it, system development work consists of the 'mapping' of a use purpose that is transformed into a system specification and then a programme executable plan. The 'plan in the machine' is conveyed to users through instructions providing for the 'step-wise' accomplishment of procedure. But no account of the embodied practical actions required to realise instructions is provided in the course of instruction. As Garfinkel (1967) indicates with the term ‘irredeemable incompleteness’ of instructions, a considerable amount of work is required to carry out instructions. Suchman relates how this can result in a fateful interactional impasse. *"Interaction between people and machines requires essentially the same interpretive work that characterises interaction between people, but with fundamentally different resources available to the participants. In particular, people make use of a rich array of linguistic, nonverbal, and inferential resources in finding the intelligibility of actions and events, in making their own actions sensible, and in managing the troubles in understanding that inevitably arise. Today’s machines in contrast, rely on a fixed array of sensory inputs, mapped to a predetermined set of internal states and responses. The result is an asymmetry that substantially limits the scope of interaction between people and machines. Taken seriously, this asymmetry poses .. outstanding problems for the design of interactive machines.[Particularly] the problem of how to lessen the asymmetry by extending the access of the machine to the actions and circumstances of the user".*

Thus, in using a computer, one aspect of making sense of things in the production of information is the ability to discriminate quickly between relevant information and ‘noise’. Such expertise becomes important given that users’ errors and failures are constituted with reference to their interactions with the machine. Consequently when the user does succeed in producing an action, they must then interpret the machine response and utilise this interpretation as the basis for subsequent action. Typically users make a series of selections from menus. The design of the system projects the course of the users’ actions as the enactment of various procedures for doing the job. The total sequence of procedures constitutes the ‘plan in the machine’ which has been implemented as programme. However, user and system have a different relationship to the plan. The plan *determines* the system’s behaviour, but the user is required to *find the plan* as the product of a set of procedural instructions. Even when presented with instructions that ‘anyone’ should be able to understand and follow, practical troubles still arise and users characteristically rush to premature and often mistaken conclusions about what has happened, what is happening, what the machine ‘meant’, what the machine ‘is thinking’, and so on. This is because the descriptions contained in instructions are always subject to further definition and elaboration. Instructions rely on the ability of the recipient to do the implicit work of anchoring descriptions to concrete objects and actions. Successful instruction following consists in constructing a particular course of action that is accountable to the general description that the instruction provides - that is, people are rule users rather than rule followers. It is clear, however, from the observations of use that many users fail to find the implicit plan

In some ways the skills involved in 'following a plan' become visible only 'in the breach'. A simple example of just such a 'breach' provides a demonstration of some of the complexities involved in using computer systems even when dealing with apparently everyday or ‘routine’ activities; in this case a bank clerk transferring an account from one branch to another.

> *“1. Transferring an account...looking at follow up screen..'Transfer Associate Products'..*
> *2. Looks at PIF (manual) for Enquiry code...*
> *3. Enters Customer Product History Screen...*
> *4. Looks at PIF again..*
> *5. Writes details on form..*
> *..*
> *7. Types into Update Account 'Transfer' Screen..screen shows error ..quits it..*
> *8. Types into Transfer Accounts Between Branches Screen..gets same error message as before..*”get lost”..
> *9. Looks at PIF again..*
> *10. Types into Update Associate Product Screen...Transfer Customer..*
> *11. Crosses fingers.*
> *12.* “It works”

This is not a particularly profound example. Here a worker is simply trying to transfer an account from one branch to another for a customer who has changed address. The problem is that she has never done this procedure before and so is attempting to use the manual (the PIF) and the 'Action Sheets' to find out what to do. However, using the manual does not resolve her problem since she is not sure exactly how to transfer the account, which particular screen to use - does she use the 'update account' screen, the 'transfer accounts between branches' screen or the 'update associate product' screen? The answer is not intuitive and nor, apparently, is the manual especially helpful in this case but eventually, through trial and error, the task is accomplished. What this extract identifies are some *specifi*c aspects of technology in use. It describes some of the real world practices whereby technology use is enmeshed into a ‘system’ of work, and how, in highly particular but nevertheless generally applicable ways, the technology is used, misused, and rejected in the ‘flow’ of activities-being-done. Of interest here is the notion of ‘intentionality’; that users actively construct a concrete sense of the objects they are working upon in order to get their activities done. The intelligibility of the technology in the course of accomplishing the transfer of an account is a continuous *achievement*. Members make sense of the screen responses and prompts and the directions of the manual to work out ‘what is going on’, based upon a mosaic of ‘recipe knowledge’(Schutz 1964). In this instance the 'recipe knowledge' is initially insufficient for the purpose at hand but such false or premature actions are not simply a product of impatience and inexperience. Nor is this a simple instance of the fact that goals do not always match those implied by the manual designers. (Suchman 1986). Those who are relatively new to the technology can be especially disadvantaged because they have not yet had sufficient experience to build a stock of 'recipe' type knowledge. They have only limited experience and resources with which identify possible actions and interpretations relevant to accomplishing goals and addressing queries, and, as a result, the strategies they adopt are often local and fragmentary. In this instance the clerk is unsure as to her next action because she does not know how to characterise her problem in a way that is appropriate for the program. She cannot 'follow the plan' because she is unsure what plan she should be following - is she transferring a product, updating a product or updating an account? - the machine gives her little indication, other than error messages, what it is doing and she is unable to interrogate the machine to ask what is happening.

This aspect of interaction with 'the plan in the machine' is highlighted in Dourish and Button's (1996) account of 'technomethodology'. As they note: *"What computational abstractions share with the abstractions of natural, everyday interaction is that they are organised to reveal certain things (and hide others) for certain purposes. What they do not share with the abstractions of everyday activities is the observable-reportable nature of everyday action* .." The key property of human action is the way in which it is made 'accountable' - observable and reportable - as it happens. But abstract computational behaviour is not accountable in this fashion. 'Technomethodology', as Dourish and Button describe it, involves more than a call to design computer systems so that people can understand them but is intended to build upon the indexical or situated character of action to make system behaviour 'accountable'. *"In other words, a user will encounter a system in myriad settings and circumstances, and will attempt to find the system's behaviour rational and sensible with respect to whatever those infinitely variable circumstances might be, ... What this implies, then, is that the creation of an account for a system's behaviour is not a "one-off" business. It cannot be handled once-and-for-all during a design phase conducted in the isolation of a software development organisation in Silicon Valley. The creation of the account happens, instead, in every circumstance in which the system is used, because the account and the circumstance of the use are intimitely co-related".* They suggest that to manage the relationship between the user's work and the system's action there is a need to provide users with more information about how the system goes about performing the activities that have been requested. They point to the requirement for systems to - *" provide cues as to not only what the system was doing, but why it was being done, and what was likely to be done next, uniquely for the immediate circumstances"* - offering the possibility of a 'cunning plan' that explains itself as it is implemented. Now that would be as foxy as a fox that won first prize in a foxiness competition.

Boden, D. (1994) *The business of talk : organizations in action*. Cambridge : Polity Press, 1994

Bittner, E. (1965). The concept of organization. *Social research*, 32, 239-255.

Button, G. and Dourish, P. (1996), Technomethodology: Paradoxes and Possibilities. In *Proceedings of CHI'96 Human Factors in Computing Systems*. New York: ACM Press.

Dourish P, Button G. On "Technomethodology": Foundational relationships between Ethnomethodology and System Design. Human-Computer Interaction 13(4), 395-432, 1998.

Ellis, C. A. (1983), Formal and informal models of office activity. In *Proceedings of Information Processing 83. The 9th IFIP World Computer Congress*. Paris: North-Holland.

Garfinkel, H. Studies in Ethnomethodology. Englewood Cliffs, New Jersey. Prentice Hall, 1967.

Grudin, J. The Computer Reaches Out. In *Proceedings of CHI'90* (Seattle WA, 1990), ACM Press.

Orr, Julian E. (1996) Talking about machines : an ethnography of a modern job. Ithaca, N.Y. Cornell U.P., 1996

Pycock, J. (1999) Designing Systems: Studies of Design Practice Unpublished PhD. Manchester University.

Schmidt, K. (1997) 'Of maps and scripts: the status of formal constructs in cooperative work' In *Proceedings of GROUP 1997*. ACM Press.

Selznick, P. (1948) 'Foundations of the theory of organization' American Sociological Review, 13, pp 25-35.

Suchman, L. A. (1983) : "Office procedures as practical action: Models of work and system design," ACM Trans. On Ofice Information Systems, vol. 1, no. 4, Oct. 1983, pp. 320-328.

Suchman, L. A. (1987) : Plans and Situated Actions: The Problem of Human-Machine Communication, Cambridge Univ. Press, 1987.

Suchman, L. (1995a). Representations of Work. Editorial, special issue of CACM, Vol. 38(9), pp. 33-34.

Suchman, L. (1995b). Making Work Visible. Special issue of CACM, Vol. 38(9), pp. 56-64.

Suchman, L. A., and E. Wynn (1984) : 'Procedures and problems in the office," Office: Technology and People, vol. 2,1984, pp. 133-154.

Voß, A., Procter, R., Slack, R., Hartswood, M., Williams R. and Rouncefield, M. (2001) 'Production Management and Ordinary Action an investigation of situated, resourceful action in production planning and control'. In Proceedings of the 20th UK Planning and Scheduling SIG Workshop (PLANSIG 2001), University of Edinburgh Dec 13th-14th 2001

Voss, A., Procter, R. and Williams, R. Innovation in Use: Interleaving day-to-day operation and systems development. In Cherkasky, T., Greenbaum, J. and Mambery, P. (Eds.) Proceedings of the CPSR/IFIP WG 9.1 Participatory Design Conference, New York, (New York, December), 2000, p. 192-201.

Zisman, M.D. (1977), *Representation, Specification and Automation of Office Procedures*. Report from the Dept. of Decision Science, The Wharton School, University of Pennsylvania.

RE-Creations

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

Deadline for next issue: 1st March 2002.

RE-Publications

Book Reviews

All books reviewed by Ian Alexander

Use Cases – Requirements In Context
Daryl Kulak and Eamonn Guiney
Addison-Wesley, 2000.
ISBN 0-201-65767-8 (paper)

Kulak and Guiney have written an obviously important book, contributing to the hotly-debated literature on Use Cases. Use Cases are the Object-Oriented world's answer to the problem of requirements. The Object-Oriented paradigm is steadily rising in importance in both software and systems contexts. This makes the concept of the Use Case crucial in requirements engineering. This is one of the few book-sized works on use cases (another is Alistair Cockburn's *Writing Effective Use Cases,* Addison-Wesley 2001) as opposed to UML. It is interesting to compare the two books.

Kulak and Guiney focus quite strongly on software and on the Unified Modeling Language (UML), but at the same time propose their own definition of a use case and what it should contain, supported by clearly documented templates and examples. They explain in a fresh way the difference between the user's and the developer's points of view, and use this to critique traditional and older approaches to requirements -- would you really want users to look through DFDs, ERDs, and long lists of *shall*s, or try to define requirements by embodying them in prototypes? All those techniques have a place but they do not substitute for presenting things in a shared language.

Cockburn concentrates on goals of use cases at different levels, and then on documenting them as stories that define requirements with a more-or-less standardized template. His emphasis is strongly on use cases as a written narrative form, and he is critical of totemic belief in diagrams – you need to think out what you want, and to 'make the story shine through'. He is not much in favour of tricky semantics, and cautions readers against diving into the subtleties of UML use case notation such as the *extends* and *generalizes* relationships. His book is strong on advice on how to write good use cases, with plenty of real examples and clear guidelines. There are (quite difficult) self-test exercises with answers. It is a reflective but solidly practical book with a tight focus on requirements – it does not look at process in any detail and Cockburn insists (wrongly in my view) that use cases can only be black-box behavioural requirements. Therefore he does not look at the way use cases can peek inside the box during design, though he does mention the transition to test cases.

Kulak and Guiney stay with the show-and-tell industrial style, without exercises, questions, or projects for the reader. They set use cases much more firmly in the context of the UML, starting with 'a look at the nine UML diagrams in some detail'. The use case is presented initially as a diagram, though the authors correct themselves a paragraph later with 'Use cases are text descriptions…'. They rather spoil their case by continuing '…of the interaction between some outside actors and the computer system.' This is unfortunately a gross oversimplification: there may be no significant outside actors, or the actors may be other parts of the system, and the system certainly need not be a computer.

I think it is fair to say that Kulak and Guiney are concerned mainly with software – the cover blurb says Kulak is CEO of Water-Logic Software, an Internet and technology consultancy, with a background in managing software development, while Guiney is a consultant to the money management industry. Their approach may therefore be expected to be well-suited to business software, but may not necessarily carry so well to systems of other kinds.

Much of the book is devoted to looking at how to build up detailed use cases in successive iterations. Like the Robertsons (*Mastering the Requirements Process*, Addison-Wesley 1999), the authors rightly emphasize that you can't expect to get perfect requirements instantly. Their 'methodology' is 'to iterate through levels of fidelity with your use cases until you have everything you need to build a great system'. Their proposed levels are

- Façade - Outline and high-level descriptions
- Filled - Broadening and deepening
- Focused - Narrowing and pruning
- Finished - Touching up and fine-tuning.

Leaving aside the possibly somewhat forced alliteration of these names, this insistence on iteration is welcome, though one may question whether a fixed structure of four fidelity cycles is suited to every project. The authors state that their 'iterative and incremental lifecycle is not a set of lifecycle phases for requirements, something that would be cumbersome; rather, it is a way to categorize the activities needed to develop use cases'. This means the cycles can be taken informally, but that brings a risk that the value of iteration is lost through haste. Incidentally the description of this approach as 'iterative and incremental' is confusing, given that iteration typically means reworking requirements and design for a new product version, whereas incremental development means fixing the requirements and design, and releasing a new increment of functionality without disturbing the existing design. Process is not a strong point in this book.

Where the book scores is in its rather detailed case studies – 'Sell Property' and 'Track Costume Sales' which are each taken rather formally through the four iterations. These case studies make up nearly half the book's 300 pages, and they illustrate in detail the use of Kulak and Guiney's use case templates.

The templates themselves are not unlike Cockburn's, and therefore also have much in common with the rather heavy Rational Unified Process (RUP) version, which Cockburn discusses. Academics may get excited about the differences in template structure between authors, but in truth the three templates mentioned (there are others) are all quite similar and probably all work well in practice. In any case, projects are free to modify templates to suit their own needs, and should do so.

In the initial *façade* iteration, you just fill in the name and the summary. In *filled*, you fill in at least the basic course of events to define the main story. Then you think out what else can happen and polish up the details.

The book does briefly peek beyond requirements, with an appendix on how 'Use cases can drive the entire application development lifecycle' from business modeling through requirements gathering, analysis, design, construction, testing and even deployment.

By 'deployment' is meant planning successive releases of (software) functionality to 'make sense from the users' perspectives'. This is welcome, as it allows requirements thinking to extend to working out how the transition to using the new system will work, step by step. Obviously, requirements grouped in a use case need to be released together as an integrated set. In this context, 'users' plainly means 'anyone from the client organization', hardly the most sophisticated definition: the authors are a little uncritical in their use of developer-oriented terminology.

Design gets a little more detail in the appendix, with good ideas like using use case Actors (roles) as templates for user security profiles (the rights and restrictions to be applied to specific user roles). The authors mention, tantalisingly briefly, that 'it is possible to create use cases for *internal subsystems*' (their italics). The subsystem remains black-box, but of course – though they don't say so – that means opening the lid of the system itself, making it 'white-box'. This means that use cases actually have a much wider role in systems engineering than is discussed either by Cockburn or by Kulak and Guiney.

The lack of emphasis on systems verges on hostility in one or two places. I rather take issue with the assertion that

'By turning requirements *specification* into requirements *engineering* (a term we've avoided), teams often produce huge amounts of requirements documentation.' (p54)

This seems an altogether unwarranted assault on engineering. Given that the rest of the book concentrates almost exclusively on (business) software, one may suspect that the authors do not wholly appreciate the need to keep track of systems, subsystems, and interfaces in complex projects such as in transportation, telecommunications, defence, and aerospace. It really will not do simply to assert that

'Strategies for reducing volume are to leave the rote requirements (table maintenance and the like) until the end and to abstract use cases and business rules as much as possible.'

These two pieces of advice -- do custodial requirements last, as the Robertsons (p334 of their book) have long advised, and abstract away from individual details to create general rules, as everyone has tried to do for a generation -- are fine as far as they go, and no doubt helpful for people specifying business software, but they in no way address the desperate need to maintain order in large systems projects by defining and tracing requirements from one level down to the next, not to mention to design elements and test cases. Sometimes I feel that there are no real controversies, just people in different environments talking past each other.

The book is well-indexed, has a short bibliography (suggesting that many of the ideas presented are novel), and is attractively illustrated with text illustrations by Erin Lavkulich. It concentrates more on life-cycle structure and less on writing skills than Cockburn, and perhaps has less 'guru' flair about it, but it is an important reference for anyone wanting to put use cases to the test on a project. Many of the issues are plainly contentious enough for the book to have value to researchers also. Unlike Cockburn, there is no explicit support for self-test or for student exercises, so the book is less immediately useful for teaching.

The Tacit Dimension
Michael Polanyi (1891-1976)
Doubleday & Co., 1966
Reprinted Peter Smith, Gloucester, Massachusetts, 1983
ISBN 0-8446-5999-1 (boards)

This quiet little philosophy book, by a religiously-inclined research fellow of Merton College, Oxford who went to America to study further, back in the early 1960's, is a surprising thing to treat as a classic of modern requirements engineering. But such it undoubtedly is.

Referenced by many learned papers written by people who never read a page of his writings, Polanyi is now unquestionably the father of Tacit Knowledge - at least, if you discount the massive contribution of Wittgenstein, who is perhaps even less read and understood despite the clarity of his work.

The Tacit Dimension is a terse and clear exposition of Polanyi's thinking, developed over more than 20 years in lectures, books, and essays. The care pays off: the book stands alone admirably as an account of what tacit knowing actually means. Why it matters in our domain is, of course, another matter.

The nub of Polanyi's argument, and a much-quoted aphorism, is that, quite simply,

"*we can know more than we can tell*." (his italics)

He goes on:

"This fact seems obvious enough; but it is not easy to say exactly what it means. Take an example. We know a person's face, and can recognize it among a thousand, indeed among a million. Yet we usually cannot tell how we recognize a face we know. So most of this knowledge cannot be put into words."

There in a nutshell is the key problem in requirements elicitation: people know, and even know that they know, but cannot readily tell us in detail what their knowledge is.

Fortunately Polanyi at once gives us some better news. In the case of facial recognition, the police have

"a large collection of pictures showing a variety of noses, mouths and other features. From these the witness selects the particulars of the face he knows... This may suggest that we can communicate, after all... provided we are given adequate means for expressing ourselves."

The situation, then, is that we possess knowledge we cannot tell, tacit knowledge, but that skilled elicitation enables us to talk about at least some of it and effectively share it when necessary. The good news is that requirements engineers shouldn't be out of work any time soon, as the task is difficult and requires skill. The tacitness is not destroyed by tricks such as the police's Photofit, as we cannot tell how we 'match the features we remember with those in the collection' - it is the same mystery expressed in a different form.

Polanyi reflects on the tacit dimension in various ways, and thereby derives an ontology of this strange way of knowing. In particular he shows that the knowledge is *embodied*. For example, when we use a tool of any kind to explore something, we very soon don't feel the tool, we feel the thing through the tool. For instance, if you take a pencil and tap various surfaces with it, you can quickly tell whether you are tapping something hard and hollow like an empty tin, or firm, like a desk, or yielding, like a pile of paper. The tool becomes an extension of your hand, and you interiorize its behaviour as a probe. To put it even more strongly, you start to 'indwell' or inhabit the tool, just as you inhabit your clothes and your body. How does the brain know anything about the world? Through the fingers, the skin, the ears, the other senses ... it suddenly becomes obvious that most of our knowledge is tacit, giving rise to most of the classical problems in philosophy. We no more know how we stand up or walk than we know how we strike a tennis ball with a racquet, ride a bicycle, play the violin, or design a program. We have a convincing illusion of being aware of all around us, while all the time we have practically no idea of how we actually operate, or what skill is. Perhaps the thing that most needs explanation is not tacit knowing at all, but the belief that knowledge is explicit.

After this introduction, you will not be surprised to hear that Plato comes into the story. In the *Meno*, Plato points out that it is contradictory to say that you can see a problem but that you don't know the solution. How do you know there's a problem if you don't know what it is?

"To see a problem that will lead to a great discovery is not just to see something hidden, but to see something of which the rest of humanity cannot have even an inkling."

Let me just say in a few words what Plato's argument is, in case you aren't familiar with it. Plato argues that you never discover anything, you just remember what you knew already, as when he questions a slave until the poor fellow 'remembers' the theorem of Pythagoras. This shows, says Plato, that the perfect form of all things is eternal, and we are born with complete knowledge of it.

"The solution which Plato offered for this paradox was that all discovery is a remembering of past lives. This explanation has hardly ever been accepted, but neither has any other solution been offered for avoiding the contradiction."

There is something delicious in realizing that Polanyi is cheerfully saying that all the philosophers have been wrong for 2000 years, simply because they didn't see that knowledge was often tacit. A scientist who has a tacit grasp of a problem may indeed be well on the way to a solution, but can't state it, yet, and certainly can't measure it. 'We can have a tacit foreknowledge of yet undiscovered things'.

The practical effect on handling requirements is profound. System users cannot tell us what they need in the new system, because they don't explicitly know: our job is to tease out that knowledge and have different people agree it explicitly. Interviewing is good at eliciting current problems (as Lauesen shows) but poor at identifying the shape of a desired solution. Scenario workshops, prototypes and demonstrations are better for that. Elicitation must address different types of tacit knowing, from deeply ingrained skill to simple facts that were too obvious to be worth stating. If you think requirements engineering is just a matter of documenting and tracing known requirements, you urgently need to study Polanyi.

This is a splendid book, short enough to read in a few sessions, but strong enough to change perceptions for a lifetime. Every requirements engineer should read and interiorize it.

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.

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.

Mailing lists

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

The RE-online mailing list aims to act as a forum for exchange of ideas among the requirements engineering researchers and practitioners. To subscribe to RE-online mailing list, send e-mail to majordomo@it.uts.edu.au with the following as the first and only line in the body of the message:

subscribe RE-online <your email address>

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*.

Tool digest - a special report on the tool vendors at RE'01 by Ian Alexander

RE'01 was attended by a selection of tool vendors. I thought it would be interesting, given that you all know the sales hype already, to ask each representative for a personal one-sentence opinion of the unique selling points, key advantages, central sales propositions or what have you of their products and services. Not all the vendors offer comparable goods: *caveat emptor*. Here is what they said. They are certainly diverse in tone.

Jim Heumann of Rational: *Rational is about tools but also about services, lots of teams locally that serve people, best practices and thought leadership, and of course our goal is to help people write better software – in a nutshell.* http://www.rational.com

Leon Stucki of Future Tech Systems: *We allow you to ENVISION the power of information leveraging throughout your organization: one definition of each piece of information – used in an infinite number of ways, e.g. organization charts, standards and procedures .. it's a meta-modeling tool.* http://www.future-tech.com

Joseph Craig of Qualisoft (based in Michigan, USA): *QFD designer is a way of taking the verbatims from the customer and making them actionable – keeping the customer as the focus of attention, using QFD (house of quality) to emphasize the human element.* http://www.qualisoft.com

Gordon Brimble of IgaTech: *RDT provides highly capable document handling for parsing input documents and creating output documents, capture of derivations that link derived requirements to record the logic behind requirement flowdown and integration with requirements modeling tools.* http://www.igatech.com

Nancy Rundlet of Telelogic: *With DOORS, we provide worldwide support, Word-like ease of use, scalability from 1 user to several hundred, and ease of establishing traceability and displaying it to multiple levels.* http://www.telelogic.com

Antonio Monzón of TCP Sistemas e Ingeniería: *with IrqA, we cover the full requirements specification cycle, not only RM and capture but also analysis, specification – features related to the construction of a specification; we have graphical, visual features like State Machines, Use Cases, graphical structuring of specifications – functional, non-functional, test cases, diagrams of review processes, information models, link matrices.* http://www.tcpsi.es

Harold Knight of SDRC: *Slate is fundamentally different in Systems Engineering because we manage all components of the design in true Object-Oriented fashion – not documents or paper but information, so we are a system design tool – system engineers can design and view systems from any perspective.* http://www.SDRC.com

Serge Chicoine of Objective SST Corp: *We provide specialized high-end training (not Word!) customized to meet the needs of the customer, specialized in Systems Engineering processes and applications, with quality assured by experience of our instructors before they joined us – in Canada, the USA and wider afield.* http://www.objectivesst.com

Chip Carey of Starbase Corp: *The exciting thing about RM and Caliber RM in particular is that it brings all departments together within the software development lifecycle and puts them all on the same page – it provides a mechanism for communication and collaboration and effectively provides a synergy where before they were perhaps separate efforts and maybe counter-productive.* http://www.starbase.com

RE-Actors

The committee of RESG

Patron: Prof. Michael Jackson, Independent Consultant. E-Mail: jackson@acm.org.

Chair: Prof. Bashar Nuseibeh, Computing Department, Faculty of Maths and Computing, The Open University, Walton Hall, Milton Keynes, MK7 6AA, UK. E-Mail: B.A.Nuseibeh@open.ac.uk.

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

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.

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.

Membership secretary: Steve Armstrong, Computing Department, Faculty of Maths and Computing, The Open University, Walton Hall, Milton Keynes, MK7 6AA, UK. E-Mail: S.Armstrong@open.ac.uk.

Newsletter editor: Dr Peter Sawyer, Lancaster University, Computing Department, Lancaster, LA1 4YR, UK. E-Mail: sawyer@comp.lancs.ac.uk.

Newsletter reporter: Ian Alexander, 17A Rothschild Road, Chiswick, London W4 5HS. E-Mail: iany@easynet.co.uk.

Publicity & WWW officer: Juan Ramil, Computing Department, Faculty of Maths and Computing, The Open University, Walton Hall, Milton Keynes, MK7 6AA, UK. E-Mail: J.F.Ramil@open.ac.uk.

Regional officer & Chair for the North of England: Dr Kathy Maitland, University of Central England, Perry Bar Campus, Birmingham, B42 2SU. E-Mail: Kathleen.Maitland@uce.ac.uk.

Industrial liaison officer: Dr Efi Raili, Praxis Critical Systems, 20 Manvers Street, Bath BA1 1PX. E-Mail: efi@praxis-cs.co.uk.

Associate Industrial liaison officer: David Bush, National Air Traffic Services, UK. E-Mail: David.Bush@nats.co.uk.

Members-at-large:

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

RE-Funds

Minutes of the RESG AGM

Held at University College London
Wednesday 21st November 2001.

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. Minutes of the Previous Meeting: The Chairman reminded members that the minutes had been circulated as part of RQ 22. He reported that there had been no adverse responses on the content. No further changes or comments were raised by the meeting, and there were no matters arising.

The Chairman advised members that the minutes for this meeting would be available on the website.

2. Chairman's Report: The Chairman reviewed the events in which RESG had been involved over the year, commenting that it was a busy and successful year:

2.1 RESG Events:

Practical uses of Scenarios, IC London, 12th July 2000.

Large Scale Requirements Analysis as Heterogeneous Engineering, UMIST Manchester, 3rd November 2000.

Requirements for 1 Billion Users! RE for Websites and Product Lines, UCL London, 29th November 2000.

Risk Management and Mitigation in large systems Engineering, Bath, 7th February 2001.

Mini-tutorial: Requirements Negotiation based on Win-Win, IC London, 5th April 2001.

Requirements for Systems Accessible by All, Hatfield, 19th September 2001.

Requirements for Mobile Systems, UCL London, 21st November 2001.

2.2 Co-Sponsored Events

Mastering the Requirements Process, London, 18-20th September 2000.

Writing Testable Requirements: A practical workshop, London, 22nd November 2000.

Scenarios through the Systems Life Cycle, IEE London, 7th December 2000.

Integrating People, Processes and Technology, London and Reading, 6 & 7 June 2001.

5th IEEE International Symposium on Requirements Engineering (RE'01), Toronto, Canada, 27th-31st August 2001.

The Chairman reported the success of the Newsletter, which had 4 issues in the year (Issues 21-24). He recorded his thanks to Pete Sawyer for editing these. He encouraged the members to be as active as possible in contributing to the newsletter, which might include short paragraphs of opinion, or comments or full articles. The Chairman also reported that Michael Jackson had agreed to become the Patron for the RESG. The Chairman expressed his desire for a strong industrial liaison theme in the RESG to ensure that there remained an emphasis on the practical.

3. Membership Report: In the necessary absence of the Membership Secretary, the Chairman reported that at present membership stood around the 300 number, but that the membership list was in some need of updating. He advised members that membership for 2002 would be free, provided that the membership form was returned with correct contact details completed.

4. Treasurer's Report: In the necessary absence of the Treasurer, the Chairman presented a summary of the accounts for the year. These are included at the end of these minutes. He expressed the view that the RESG was doing well financially with a slight profit reported for the year. He reported that the committee had agreed that for 2002 ordinary meetings would be free although Workshops and Tutorial meetings would still be charged for.

5. Publicity: The Chairman highlighted the website and the RESG mailing list as the chief mechanisms for publicity. He hoped to see the website more widely used as a means for publicising job opportunities, and the mailing list system extended. In response to questions the Chairman agreed:

That members and associated organisations were welcome to link to the RESG website.

That the webmaster would be happy to include links from the website to external pages (although his discretion would apply in order to prevent the RESG site becoming too much of an advertising site).

That Data Protection issues in the management of mailing lists was critical, and that where possible members should relay RESG mailings to their own (appropriate) mailing lists rather than provide those lists to RESG.

6. Election of Executive Committee Members: The Chairman provided this list of Executive Committee members standing for election/re-election, and added that David Bush, standing for election as a member-at-large had agreed, if elected, to shadow Efi Raili in the Industrial Liaison role.

Chair:	Bashar Nuseibeh
Vice-Chair:	Alessandra Russo
Secretary:	Wolfgang Emmerich
Treasurer:	Neil Maiden
Membership Secretary:	Stephen Armstrong
Newsletter Editor:	Peter Sawyer
Associate Editor/Reporter:	Ian Alexander
Industrial Liaison Officer:	Efi Raili
Publicity Officer:	Juan Ramil
Northern Representative:	Kathy Maitland
Members-at-large:	David Bush Suzanne Robertson
Patron:	Michael Jackson

The election was proposed by Michael Jackson, and seconded by Michael Jackson (a different one!) and Gary Birch. The motion was passed unanimously.

7. Any Other Business: The Chairman reminded the meeting that the next meeting was in Bath on 27th Feb 2002, entitled 'Key Challenges in Safety Requirements Engineering'.

Michael Jackson noted that the BCS Newcastle Branch and CIM (North East) were holding an event bringing together IT and marketing people, the Chairman agreed to circulate information provided in the next mailing to members.

There being no further business the meeting ended at 13.55pm.

Accounts

Year-start	**£**
Bank Balance 05/00	481.43
Gold Account 05/00	21961.57
Receipts	
Membership subscriptions	540.00
Event income	2150.00
Interest receipts	970.42
	3660.42
Payments	
Committee Expenses	270.00
RE'01 Sponsorship	1500.00
Newsletter	581.30
Interest/Other Payments	219.87
Events Expenditure	838.70
	3409.87
Net movement 2000/01	+250.55
Balance at 30/04/2001	**22693.55**