Requirenautics Quarterly

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

http://www.resg.org.uk

RE-Locations

RE-Soundings

Editorial

This issue includes some excellent articles from people in a good position to comment on our business. Karl Wiegers describes his experience as (surely) one of the busiest people in RE training and the spread of good RE practice. He casts light on the state-of-the-practice - and what's revealed isn't always pretty. Karl's article is a great reality check for those of us who sometimes forget how hard it is to change industry practice for the better - often because we don't take the trouble to properly understand what "better" means. Karl's article is by no means a horror story, however. It's clear that there are quality organisations out there that know what good practices are when they see them, and know how to use them for the benefits of themselves and their customers.

In "Under the Hood", Rob Tillaart removes the mystery of how we can get from user-focused use cases to requirements that developers can make use of. Rob's article dovetails nicely with Karl Wiegers' experiences with use case training and the problems getting real, useful information from the naive or misinformed application of use cases.

Finally, Simon Hutton explains how systems engineering has informed the UK Ministry of Defence's policy on "Smart Acquisition". Since the MoD is a major source of income for software and systems engineering companies, Simon's article should be of considerable interest. One of the most interesting aspects is the light it sheds on how a very large organisation can change the way it does systems engineering, and, of course, on the stimuli for why changes take place.

Expect all these issues and more to pop up at R-Day!

Pete Sawyer
Computing Department, Lancaster University.

Chairman's message

I am hoping that the first time you pick up and read this issue of RQ will be at R-Day on 31st March 2004.

The RESG does not organise an annual conference, but every few years the executive committee gets a surge of energy and enthusiasm and puts on a special day packed with goodies for the requirements engineering community. This time round looks to be no exception. The day is coincidentally, and happily, co-located with the Programme Committee meeting of the International Requirements Engineering Conference (RE'04), which will be held in Kyoto, Japan, in September 2004 (and of which the RESG is a sponsor). This means that about 40 of the world's top RE researchers and practitioners will be gathered in London to discuss the RE'04 programme on the previous two days. We thought that this would be too good an opportunity to miss getting some of them to give talks, workshops, and tutorials to the UK RE community.

The R-Day programme is as packed with as many RE-related activities as is feasible in a single day. I would like to take this opportunity to thank all the RESG committee members for their phenomenal work in putting this event together, without giving up the day job. It would be wrong of me to single out particular committee members for their efforts - this has been a genuinely collaborative group effort, and everyone has pulled together to pull this off.

If you are reading the newsletter and you are not a member, why not join today?! If you have not renewed your membership - shock, horror, how could you?!!

To everybody: welcome.

Bashar Nuseibeh
The Open University

RE-Treats

For further details of all events, see www.resg.org.uk
Next event organised by the group.

R-Day'04

Date: 31st March 2004, 9.00 am – 5.00pm
Location: Thistle City Barbican, Clerkenwell, London
Contact: Alessandra Russo (ar3@doc.ic.ac.uk)

R-Day'04 is a special full-day RESG event aimed at RE practitioners and researchers. There will be a full programme of parallel tracks presenting work dealing with many of the problems associated with the generation and management of requirements. There will be presentations by internationally known experts, mini-tutorials, poster sessions, a tools exhibition and a book display. Best of all, R-Day'04 will provide an opportunity for practitioners and researchers to share ideas and experiences while enjoying the exciting cultural diversity of the city of London.

The range of topics for R-Day'04 include:

2. Ontology Development
3. Requirements Evolution
4. Non-functional Requirements
5. Integrating Requirements, Design, and Testing
6. Research Methodology
7. Scenario-based RE
8. Metrics and Measurement
9. Agile Methodologies
10. Building Bridges between Business and Development
11. Problem Frames

Engineering Organisational Solutions from Human Information Requirements

Date: 12th May 2004, 2.00 – 5.00 pm
Location: University College London, London
Contact: Pete Sawyer (sawyer@comp.lancs.ac.uk)

This one-day interactive workshop presents the MEASUR methods. Practical applications have yielded major benefits in system quality, cost reduction, and adaptability to changing business needs. They are based on analysing knowledge, responsibilities and meanings to produce rigorous specifications of human information requirements. From MEASUR specifications, supporting prototype computer systems can be generated directly.

MEASUR advances the theory by replacing the usual "information flow" model by one of "information fields" defined by the communities that share sets of norms (social, organisational or legal) that determine their members' information needs.

The workshop will be presented by Ronald Stamper and Yasser Ades.

RE-Calls

Recent Calls for Papers and Participation

Twenty-Sixth International Conference on Software Engineering (ICSE 2004)

23rd - 28th May, Edinburgh International Conference Centre, Edinburgh, Scotland

http://conferences.iee.org.uk/icse2004

ICSE is the premier software engineering conference, providing a forum for researchers, practitioners and educators to present and discuss the most recent innovations, trends, experiences and concerns in the field of software engineering.

ICSE 2004 will offer an exciting program of events, including keynote talks by leaders in the field, tutorials, workshops and presentations of technical papers on innovative research and the cutting edge of practice. The main program will be complemented by an array of social events, providing further opportunities for informal networking.

Tenth Anniversary International Workshop on Requirements Engineering: Foundation For Software Quality (REFSQ'04)

7th - 8th June 2004, Riga, Latvia (In conjunction with CaiSE'04)
www.refsq.org

At nine previous REFSQ workshops, researchers and practitioners from various disciplines have contributed to improve the definition and implementation of quality requirements. REFSQ'04 will continue this tradition as a highly interactive stage for discussion of quality related problems in requirements engineering as they have developed over the last years. In particular, we encourage people from the requirements engineering, software engineering and information systems fields to present their approaches to higher software quality and to discuss how requirements engineering can contribute to it.

Key dates:

- Submission deadline: 1st March 2004.
- Notifications to authors: 15th April 2004

Twelfth IEEE International Requirements Engineering Conference (RE'04)

6th - 10th September 2004, Kyoto, Japan
http://www.re04.org

The importance of requirements engineering has been recognised for many years. In the 1990s this recognition led to an IEEE Conference and Symposium series. Ten years on, the RE Conference has become the international platform for presenting new research, transferring research results to industrial practice, and presenting industrial experiences and best-practice to the widest possible audience.

In 2004 RE will take place in Kyoto, Japan for the first time. To reflect this RE'04 will continue to be interested in all aspects of RE, but is particularly interested in requirements for embedded systems in automotive and consumer products, and requirements engineering for innovative product design.

Remaining key dates:

- Doctoral symposium submissions: 16th April 2004
- Poster and research demonstration submissions: 23rd April 2004

RE-Readings

Reviews of recent Requirements Engineering events.

Requirements Engineering (RE) Training: the Who, the How, the What

3rd December 2003, Imperial College, London

Yet another interesting and packed RESG event in which very good ideas were put forward by both groups that attended the meeting, i.e. the speakers and the audience.

During his presentation, Ian Bray described the results of a study that he carried out on a random group of UK companies in order to determine the importance and rigour with which each one applied a requirements process, if one was used at all. From his conclusions we gathered that there is a lot of diversity in the manner in which this aspect of system development is treated in an organisation. It was quite disappointing to learn that still quite a large percentage does not take seriously the requirements activity even though various practitioners in the area keep hammering almost to the point of exhaustion the importance of the activity. There was common agreement amongst the presenters and the audience that a considerable mix of technical, social, and managerial skills are needed to extract, specify, and manipulate, in an effective way,

the system requirements of any system that is specified and developed to satisfy the needs of any community in any context.

Both Pete Sawyer and Ian Bray gave a good account of the approach taken in their respective universities to educate the students that take a requirements course. They both agreed on how important it is to teach the basic knowledge like requirements analysis, specifications, elicitation, modelling, requirements methods, and requirements management. They also stressed the importance of teaching the students the art of teamwork, of being good listeners, in addition to learning how to apply the various technical aspects of the requirements area that are described in the lecture theatre. The above, together with what Ian Bray labeled, as "deep knowledge" constitute the three bodies of knowledge that any requirements engineer needs to carry out a good job. Pete Sawyer gave a more detailed account of the manner in which students at Lancaster university are educated, listing the skills deemed as necessary by the Software Engineering Education Knowledge (SEEK) body, and referring to "Problem understanding" as the difficult knowledge that must be imparted to the students. Overall, there was general agreement on the content of a reasonable RE course in academia. Pete pursued a more in-depth discourse of the syllabus that should be covered to comply with the body of standards that govern this area. Ian complemented the story very well by delving more into the requirements of industry in the current times. Personally, I concluded that the universities are trying to do a good job although there are knowledge gaps that are difficult to fill. In the meantime, the state of the Requirements area in industry has a lot of ground to cover before it is in a position to demand the kind of requirements training for experts that other areas provide for their experts.

In the second half of the session, the audience had the opportunity to listen to two excellent presentations which described alternative ways of training RE practitioners that are already working in industry. Stefanie Lindstaedt, from the Know Center in Austria, described a new way of learning that is gaining popularity in certain sectors of industry, i.e. task-oriented collaborative learning. The training and application of RE knowledge is achieved through a communal effort supported by a software tool that enables the community to hold all the knowledge in a structured manner, so that it can be easily accessed and digested by the various members of the community that are linked through the tool. The approach is very reminiscent of the way of life advocated by a commune, and one from which an organisation could certainly benefit, if it is properly managed, i.e. if the information that is held in the repository corresponds to a reliable source, if it is well maintained, and last but not least, if any newcomer that arrives in the organisation is able to quickly learn how to use it to their own benefit. The presentation concluded with an example of the ad-hoc tool used to train and apply RESCUE, a scenario based requirements engineering process.

Ken Jackson, from Telelogic, closed the session with a presentation, which showed an alternative way of training RE practitioners in industry and much more. In his talk, Ken provided the audience with an answer to the key questions listed in the workshop's abstract, the What, the Who, and the How of RE. He gave a very entertaining presentation answering these questions and in so doing, providing a glimpse of the reasons behind the form and content of Telelogic's training courses for requirements engineers. Like the previous speakers, he described the need for an initial grounding on the basics of Requirements Engineering in which it is important to stress that the requirements provide *"the map and compass of the product lifecycle"*. He also stressed the importance of modelling, and showed how establishing traceability between the different artifacts produced at the various levels of the product lifecycle can help understand how the stakeholders' needs are *satisfied and qualified*. In the final part of the presentation, he gave an outline of the type of courses organised by Telelogic and the audience they were targeting. The convenience of supporting an organisation's requirements process with a tool was also raised, although it was indicated that it is possible to be trained and to apply RE principles without tool support, although the task is very difficult when the systems under development are of a complex nature, something quite common in this time and age.

At the end of the session, both attendees and presenters agreed that in order to implement a RE process in any organisation, it is necessary to deal with a number of challenges in which the cultural change the organisation must undergo can be viewed as the most important one. Once all its members are agreed in this change, the RE specialists, and the other members of the organisation, can concentrate in dealing with the other many challenges such as requirements elicitation, requirements specification, and requirements management.

Elena Pérez-Miñana 2004

RE-Papers

The Real World of Requirements Engineering

Karl Wiegers
Process Impact

Over the past several years, I've presented hundreds of seminars on software requirements engineering to thousands of people through in-house seminars, conferences, public seminars, and professional society meetings[1]. Attendees come from organizations that build corporate and government information systems, Web sites, commercial software packages, embedded systems, and other products.

After working with so many different organizations and project participants, I've made several observations about the current practice of RE in North American industry and government projects. This article summarizes these observations, based on a completely unscientific analysis of the organizations who invited me to talk to them about requirements. I have no way of knowing how accurately these experiences represent the software industry as a whole. The companies who aren't concerned about getting better requirements don't call me. Neither do the ones who have already mastered their requirements problems. The people I talk to are the ones who realize they need to get better requirements processes and better requirements specifications.

Why an Interest in Requirements?

"The pain has become too great." I've heard this refrain from multiple clients. They decided they need to do a better job on RE because of the consequences they suffer from their current requirements approaches. "Pain" comes in the form of late deliveries, extensive rework, massive unpaid overtime, quality problems, unsatisfied customers, and unfulfilled expectations. Other groups want to learn how to elicit and record requirements more consistently and effectively across their projects. Some students have come to refresh and validate their previous experience and knowledge about RE. The good news is that many organizations now realize the likelihood of project success goes up dramatically if the technical work is based on well-understood and agreed-upon requirements.

Who Attends Seminars?

Most of my seminar attendees perform the role of requirements analyst (aka business analyst, system analyst, or requirements engineer) on their projects. Other attendees include developers, testers, project managers, product managers, and documentation writers. Only rarely are actual end users or marketing people in the room. Their presence always leads to interesting and fruitful discussions. A marketing manager once stimulated a class discussion when I got to the part of my seminar where I discuss customer rights and responsibilities with respect to requirements.

But I've also heard technical people say, "It looks like the marketing people blew off yet another class" when the empty chairs represented absent marketing staff. One class of 24 engineers and testers identified marketing as the origin of every requirements-related problem they encountered, a conclusion I rather doubt was true. I've sometimes given overview presentations on requirements, ranging from 90 minutes to a full day, for audiences that were composed exclusively of users or managers.

When I do have a mixed audience, I like to randomize the participants in small group discussions to get people from different functional areas discussing their mutual problems and issues. This helps to build a collaborative mindset. Students recognize that requirements are not just a technical issue or a business issue, but rather the point at which the interests of all project stakeholders intersect. A student from a mixed-audience seminar once wrote a telling comment on an evaluation form: "I feel more sympathy for the developers now." This isn't a bad conclusion to reach.

Very rarely do senior managers or executives attend my requirements seminars. However, students often say, "Where's my manager? My manager needs to hear this. We can't do any of these new things unless our manager buys in." Having managers participate in learning experiences sends many important signals to the other class members. It says, "This is important enough for me to spend a day or two sitting here, so it should be important to all of us." The CEO of a major car-rental company once sat through an entire one day seminar. He made some insightful observations. For example, when I discussed requirements traceability, the CEO asked, "Why *wouldn't* you do this for all of your critical business applications?" Good question.

Methods, Tools, and Practices

My perception is that there is a substantial gap between the body of requirements engineering knowledge and the routine activities of most practicing requirements analysts. The types of organizations that typically have many dedicated business analysts are corporations such as insurance companies that build information systems for internal use or Web sites. These analysts could have come out of the information technology world or from the user side of the business. I have no firm data, but there appears to be a slight preponderance of former users now serving as

[1] eLearning versions of some of my requirements engineering courseware are available from http://www.processimpact.com/elearning.shtml.

business analysts. In other organizations, the role of requirements analyst is performed by a product manager, the project manager, developers, or key users. Sometimes the role is highly diffuse and there is no clear agreement on who does what regarding requirements development and management on a project.

The role of the analyst is primarily that of a communicator, bridging the gulf between the business and technical domains. It's not clear that most analysts recognize the need to beef up their skills and knowledge to be able to communicate effectively with both users and development staff. I have encountered a few organizations that split the analyst function into a business analyst from the customer side and a system analyst from the IT side. This helps compensate for shortcomings in skills and knowledge that a single individual might have, but it introduces an additional layer of communication, translation, and verification.

Few people serving in the analyst role appear to have been trained in how to perform this difficult function. Most seem to be just doing the best they can based on their previous experience and their personal desire to do good work. I see little evidence that any established RE methodologies are being practiced, except that adherents of the Rational Unified Process—and others—are attempting to apply the use case technique, with mixed success (more about use cases later).

I do a lot of audience sampling, asking how many people are familiar with one technique or another that I discuss. There's a fair amount of awareness of techniques such as the classical structured analysis models (data flow diagram, entity-relationship diagram, and so forth). However, teams in North America seem to perform little modeling of their requirements in practice, other than data modeling. There's only modest awareness and use of the Unified Modeling Language. Almost no one uses data dictionaries, context diagrams, state-transition diagrams (except for engineers building real-time systems), quality function deployment, descriptions of user classes, and other "good practices" I've found helpful over the past 20 years. I'm always puzzled to see such a large gap between the methods and techniques that people know about and what they routinely do on their projects.

Below I present some observations on several specific shortcomings in how I see requirements engineering being practiced in corporate and government organizations. Please keep in mind that these are simply my personal observations, not the result of a scientific study or a balanced survey. My purpose is neither to criticize the well-meaning and hard-working requirements analysts, nor to criticize the RE researchers who are continually seeking better ways to perform this difficult task. I'm simply sharing observations that make me concerned about the effective transfer of technology, skills, and knowledge from the software literature to those who routinely practice requirements engineering.

Vision and Scope. I think it's valuable to create a concise and focused product vision statement that helps align project stakeholders towards common business objectives. It's also essential to clearly define the scope of a project that's building a specific release of that product. The scope defines the boundary between what's in and what's out for a given project. However, few projects I encounter have such a vision statement. I do a practice activity to let the students write a vision statement for their project using a simple keyword template [Wiegers 2003]. I'm always impressed with how much they accomplish in just five minutes.

Many projects also lack a well-defined scope boundary that makes it easy to evaluate whether a proposed new feature or function is in scope or not. This makes it hard to manage scope, leading to the inevitable scope creep that makes so many projects bigger and later than anyone expected.

Customer Involvement. There is broad agreement in the software industry that customer involvement is a critical factor in developing high-quality software that meets customer needs. The software literature contains ample recommendations for engaging suitable customer representatives. Examples include: contextual inquiry [Beyer and Holtzblatt 1998], usage-centered design [Constantine and Lockwood 1999], product champions [Wiegers 2003], Joint Application Design [Wood and Silver 1995], and collaborative requirements workshops [Gottesdiener 2002]. The agile methodology movement, typified by Extreme Programming, stresses the importance of having an on-site customer to determine requirements, set priorities, and answer questions [Beck 2000]. (As an aside, few attendees at my corporate and government seminars are aware of the agile methodology movement.)

But many analysts I meet lament the difficulty they have getting adequate and accurate customer input. Often they do not work with appropriate representatives of the real users. Sometimes they can't contact users at all and must rely on surrogates to make their best guess as to what users need. Analysts might have to interact with managers of the users to get input on requirements. However, those managers have limited time to spend working with the analyst and sometimes don't accurately understand the users' requirements.

A few people in my audiences indicate that they've tried the product champion approach I advocate, in which key user representatives serve as the voice of the customer for each user class. In almost every case, people who have tried the product champion approach found it to be very effective. But it's not widely used. It is gratifying for me to return to a client some months after a training class to find that they now have product champions in place for their applications and that

they're enjoying the benefits of improved customer collaboration.

Use Cases. The use case technique is a well-established best practice for requirements elicitation. I've successfully employed use cases on multiple projects. But out there in practitioner land, the method is fraught with problems. Most of the people in my audiences have heard of use cases and some have tried them, but often without sustained success. There's considerable confusion about just what use cases are. I have seen use cases that corresponded one-to-one with specific user interface screens. Some use cases include excessive technical and implementation detail.

Some use case enthusiasts and authors maintain that the use cases *are* the functional requirements. You simply give the use case descriptions to the developers and they build the system. However, I've never spoken to a single individual who has found this approach to succeed. Over and over people tell me that they tried giving use cases to developers but there was just too much information missing for the developers to do a good job.

In my experience, this is because use cases are best employed to represent the *user* requirements; it's up to the requirements analyst to derive the corresponding *functional* requirements, which are what developers really implement. But analysts haven't been taught how to perform this translation from user requirements, written in a form that users can understand, to software functional requirements that guide the designer's work. There's also a lot of confusion about how formal and elaborate to make the use cases.

Adopting the use-case approach is often problematic. One client of mine had previously brought in a 5-day course on use cases presented by another company. The binder of class materials was huge! The method presented was so overwhelming that no one in the organization ever did anything with the class material. Sometimes teams try to convert their entire RE approach to use cases at once. The use cases become an end in themselves, rather than being a vehicle for learning about user needs so the analyst can define the necessary system functionality and attributes. When use cases are the only container being used for requirements, analysts feel they have to force every fragment of functionality to fit into a use case. Projects sometimes collapse under the weight of the massive use case document, which doesn't predispose teams to try them again.

In my training seminars I often hold practice requirements elicitation workshops based on use cases. About 25% of the teams grasp the approach immediately and make excellent progress. Another 50% or so need some coaching but eventually catch on. The rest have difficulty converting to the user- and usage-centric requirements dialog from their traditional elicitation discussions, which collect an array of random — but important — information. "What do you want?" and "What are your requirements?" are the two worst questions an analyst can ask during requirements elicitation. We need to find better ways to help organizations adopt the highly effective use case method.

Quality Attributes. Elicitation discussions at the companies I've encountered focus almost exclusively on functionality. Other than performance and perhaps security, quality attributes are not typically explored in discussions about requirements for information systems. Teams building embedded systems are more aware of the importance of various nonfunctional requirements. Those few quality attribute requirements that I have seen typically are written in a vague and platitudinous fashion. Adopting Tom Gilb's Planguage notation would go a long way toward improving how quality and performance requirements are specified.

Business Rules. Nearly all teams I've worked with on requirements engineering acknowledge that they handle the business rules that affect their products too casually. They're concerned that vital knowledge will be lost when key individuals depart, but they make little effort to capture this knowledge in the form of a business rules catalog. Missing in action are documented traceability links between business rules, functional requirements derived from those rules, and code that enforces the rules. One student said his project had so many business rules that they couldn't possibly write them all down. I was perplexed. How could they ensure that their system complied with all the pertinent rules if they didn't even have the rules recorded anywhere?

Requirements Management. Quite a few organizations have acquired commercial requirements management tools, but few are using them effectively. There's a lot of expensive shelfware out there. When people ask me for advice about requirements management tools, I recommend that they not use one until they can already write good requirements. These are requirements *management* tools, not requirements *development* tools. It doesn't matter how well you manage bad requirements.

Virtually no organizations I encounter are doing requirements traceability. One government organization put considerable effort into recording traceability data in a requirements management tool and generating massive traceability reports. As it turned out, though, no one actually looked at the reports, so the effort was largely wasted.

What to Do?

As I stated earlier, I have no way of knowing how representative these observations are of the software industry as a whole. After delivering training on requirements engineering in many venues for about eight years, though, I'm concerned about the state of the practice among both full- and part-time requirements analysts. We have many valuable,

pragmatic tools and techniques available that can help any organization improve how it develops and manages its software requirements. Except for those organizations that have already pushed current RE to the limits and need something better, the challenge is not to develop new RE methods [Wiegers 1998]. The real challenge is to give the bulk of practicing requirements analysts the practical knowledge, skills, and judgment they need to be effective on a difficult job.

Busy analysts need simple and practical techniques. They need clearly defined job responsibilities (see "Requirements Analyst Job Description," http://www.processimpact.com/process_assets/RA_job_description.doc). They need training in the soft skills of facilitation and interviewing. They need training in the business domain, in modeling, and in technical writing. They need ready access to a body of knowledge and practical, readable texts; many have been published in recent years. And perhaps they need a professional society or a certification program to elevate the requirements analyst role to the professional status it deserves.

References

[Beck 2000] Beck, Kent (2000) *Extreme Programming Explained: Embrace Change*, Addison-Wesley.

[Beyer and Holtzblatt 1998] Beyer, Hugh, and Karen Holtzblatt (1998) *Contextual Design: Defining Customer-Centered Systems*, Morgan Kaufmann Publishers, Inc.

[Constantine and Lockwood 1999] Constantine, Larry L., and Lucy A.D. Lockwood (1999) *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*, Addison-Wesley.

[Gottesdiener 2002] Gottesdiener, Ellen (2002) *Requirements by Collaboration: Workshops for Defining Needs*, Addison-Wesley.

[Wiegers 1998] Wiegers, Karl. E. (1998) "Read My Lips: No New Models!" *IEEE Software* 15(5) 1998.

[Wiegers 2003] Wiegers, Karl E. (2003) *Software Requirements, 2nd Ed.*, Microsoft Press.

[Wood and Silver 1995] Wood, Jane, and Denise Silver (1995) *Joint Application Development, 2d Ed.*, John Wiley & Sons.

Under the Hood

Rob Tillaart
Océ Technologies BV

Eliciting requirements for a new product or enhancing an existing one is a difficult task. However the software industry is getting more mature in this area as some methods and frameworks exist. If we look to any other major industry, e.g. trains or automobile industry we see it always take 80 to 100 years for an industry to become mature in most aspects. So the software industry is only halfway and we have at least 40 years to go. However the way to maturity goes in small steps, some say "baby steps" [1] and in this article we propose just such a baby step to improve the requirement elicitation.

Use cases [2, 3] are often used to describe in an informal way the steps needed to realize a business event. From these use cases the detailed requirements are distilled and refined as the details of how a business event must work are understood better.

A possible use case description could be like:

> Business event: Taxi driver starts a new ride.
>
> The taxi driver asks the customer where to go. The customer tells his destination to the driver and the driver makes some settings on the meter and starts the meter.

This is an informal way to present the business event, and we can see some requirements pop up already. However we are going to formalize this use case to get more information out of it. We do this by making a difference between what is visible and what happens under the hood of the system. The rationale behind this separation is as follows:

The visible part of the story focuses on the

- what of the functional requirements
- and the look and feel requirements (interaction with humans)

The "under the hood" part of the story focuses on the

- how of the functional requirements,
- and gives insight in a number of non-functional requirements.

When we look at use cases we often see only the visible part of the use cases described. Now we present a more formal way to describe the use case of the taxi driver.

The visible part:

V1: The taxi driver asks the destination
V2: The customer gives the destination
V3: The taxi driver makes some settings on the meter
V4: The taxi driver starts the meter

Now we add the "under the hood" part.

V1: The taxi driver asks the destination
V2: The customer gives the destination

V3: The taxi driver makes some settings on the meter
U1: The meter accepts the settings and starts to initialize itself
U2: The meter looks up the entered zip code
U3: The meter calculates the price per mile
U4: The meter resets the money counter

V4: The taxi driver starts the meter
U5: The meter starts counting

By detailing the (not visible) under the hood part we automatically see more and more requirements (often initially as questions) pop up. In the following table we present some questions that come up per under the hood step of the taxi driver scenario.

U1: The meter accepts the settings:

When are which settings accepted?

U2: The meter does a lookup of the zip code:

That takes time. How long does it take? How long may it take?

Must we provide feedback about progress to the driver?

U3: The meter calculates the price per mile:

Which parameters are needed to calculate this? What does the formula looks like? What are valid input/output values?

U4: The meter resets the money counter:

Resets to what? 0 or some minimum starting price?

U5: The meter starts counting.

In what steps does it count?

When is such a step made? This sure relates to the price/mile.

Are there other parameters in this formula?

This table shows that describing an "under the hood" part gives rise to many questions relating to both functional and non-functional requirements. It even includes hints for the fit-criteria for some of those requirements. Answering these questions or asking them to the experts or stakeholders will reveal details of the behavior of the system. A question like: *"In what steps does a taxi meter count?"* can have an answer like *"Both distance and time."* You know it displays money that's for sure, but: *"How does the meter translate distance and time to money?"* can reveal a quite complex formula. This formula could be: *"The price the customer has to pay is the starting price for the zip code of the destination plus the maximum of the distance times the distance price and the time times the time price."* Understanding answers like this gives much insight in the very details of the use case.

Although the example above is quite simple it shows that a little bit more storytelling raises a lot of good questions which brings us closer to the essence of the requirements of the system. For me the "under the hood" part is an informal way to think over and describe the essence of the use case realization and it helps me to go over the bridge between requirements and architecture.

Summary

In this story we presented the "under the hood" refinement of use case steps. By thinking about and defining the details of what happens under the hood for every step of a use case, we can raise questions about the essence of a business event. Answering these questions helps us to write better requirements and refine their fit criteria.

1. "What about Bob?", Frank Oz (director), Touchstone Pictures, 1991.
2. [Roberston 99] Roberston, S, Robertson, J.: *Mastering the Requirements Process*, Addison-Wesley, 1999.
3. [Rumbaugh 99] Rumbaugh, J., Jacobson, J., Booch, G.: *The Unified Modeling Language Reference Manual*, Addison-Wesley, 1999.

Smart Acquisition, Smart Requirements

Simon Hutton
3SL

The Smart Acquisition approach adopted by the Ministry of Defence (MOD) aims to "acquire defence capability faster, cheaper, better and more effectively integrated" [1]. This article finally fulfils a promise made outside a pub after the last RESG AGM, and introduces some aspects of Smart Acquisition. Readers will be familiar with the concepts, but may find the context of interest.

Background

The 1997 Strategic Defence Review (SDR) considered the capabilities of the Armed Forces in the context of the less-predictable realities of a post-cold war environment. An integral aspect was the Acquisition Organisation Review (AOR), a fundamental examination of how the MOD procured equipment within an annual budget of £9.43 billion (1997). The AOR concluded that MOD was not cost-effective in equipment acquisition, and recommended several changes to procurement policies, to the processes for procurement and support, and to the organisation structure. SDR described this approach as "Smart Procurement – faster, cheaper and better", although the term is now "Smart Acquisition" and integration has been added to emphasise the importance of a system-of-systems approach.

Acquisition Organisations

Smart Acquisition is built on organisational relationships, with the emphasis on clearly identified customers and the formation of Integrated Project Teams (IPT) to deliver and maintain the customer needs.

April 1999 was an implementation milestone with the transformation of the Procurement Executive to Agency status as the Defence Procurement Agency (DPA), and the formation of the Defence Logistics Organisation (DLO) from three service-oriented organisations. October 1999 was another significant milestone with the formation of a centralised

Equipment Capability Customer (ECC) organisation, providing a focus for capability requirements that meet the needs of the Armed Forces. By April 2000, 141 Integrated Project Teams (IPT) were operational, delivering capabilities identified by the ECC.

The ECC is responsible identifying the capability required to meet UK defence objectives, for translating these ideas into an approved programme and for ensuring cost-effective capability delivery. The ECC defines, refines and maintains a valid and affordable user requirement throughout the life of the project, and as the acceptance authority is responsible for accepting the system into service.

The IPT is formed as the user requirements are developing, and is responsible for creating, refining and maintaining a valid system requirement. Each IPT has a clearly defined team leader, empowered to make cost, performance and time scale trade-offs within clearly defined boundaries. The Requirements Manager (RM) within each team is a serving member of the Armed Forces, providing a link between the IPT and the ECC as well as in-house military advice. The RM ensures the IPT provides sufficient system definition to enable whole life costing, analysis of candidate options and to inform iterative debate as the user requirements evolve. The IPT is accountable to the DPA until the In-Service Date (ISD) when it transfers to the DLO.

The Second Customer role is fulfilled by the front-line commands and is responsible for realising military capability through all lines of development. Customer 2 also provides coherent operational input to the user and system requirements, and becomes the lead customer when the system and all lines of development are in place at ISD.

Military Capability

It is worth noting that equipment is one element of effective military capability, as illustrated in Figure 1.

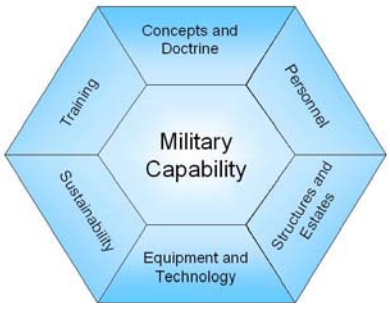

Figure 1 - Elements of Military Capability

These elements or Lines of Development (LOD) ensure all aspects are considered in addition to equipment. This is important for military equipment when personnel, in-theatre support and tactical procedures can be as important as equipment design and performance. This holistic approach to delivering capability also encourages integration with other systems, in particular logistics systems that may be procured as separate projects.

System Life Cycle

Smart Acquisition takes a whole life approach, ensuring the system procured to meet a defined capability is the most effective option based on the total cost of ownership. This encourages the IPT and industry to explore options that may be more expensive to procure, but that offer considerable savings during operation. This is particularly relevant when the in-service life of military equipment often exceeds 30 years!

Each project develops a realistic, costed Through Life Management Plan (TLMP) to plan and control the project across all LOD within the context of a system life cycle. The Smart Acquisition life cycle comprises six stages (CADMID) and two approvals gates. The concept phase defines the required capability and explores concept options. The Initial Gate business case presents the case for closing the capability gap, requesting formal approval to conduct the assessment phase. The system requirements are progressively refined in conjunction with the system design during the assessment phase. The Main Gate business case confirms that there is a cost effective system that will deliver the required capability, and that risks have been progressively reduced to acceptable levels. Approval at main gate is to procure the system, with demonstration and manufacture stages continuing to reduce the development risks and implement, integrate and accept the system. Once the system achieves In-Service Date (ISD), the capability is available for operational use with effective support, trained operators and tactical procedures in place. The final disposal stage is included to ensure all through life costs are included. An important issue when the equipment portfolio includes the cost of nuclear ownership and munitions.

Smart Requirements Model

Prior to SDR, operational requirements were expressed in equipment terms, based on the perceived solution. Detail was added as the project progressed, but development was constrained in terms of time, affordability and technology by the solution defined by the requirement. There was a tendency to express requirements in terms of equipment that was understood and in service, with little incentive to understand the real need in terms of capability. As a result, considerable expenditure was required improving in-service systems to meet operational needs that had not been recognised early in the life cycle.

These shortcomings were recognised in the AOR Report [2], with a recommendation that “a revised front-end process should be introduced which delivers robust requirements”. The result was the Smart Requirements Model, a “method for capturing,

engineering and managing requirements based on the principles of systems engineering" [3].

The Smart Requirement Model is based on a generic process defining and baselining the User Requirement Document (URD) and the System Requirement Document (SRD). Both are continually evolving, coherent and structured sets of unique requirement statements, ensuring a top-down analysis based on needs rather than solutions.

Smart Requirement Process

The Smart Requirements Model includes the generic requirement process [4] illustrated in Figure 2.

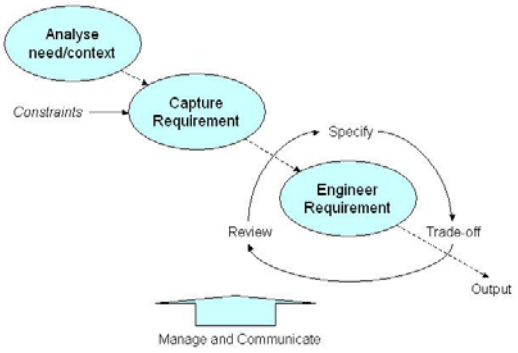

Figure 2 – Smart Requirement Process

The process recognises that requirement definition is invariably iterative, based on the following main activities:

- *Analyse* the need to establish the scope of the requirements and the external and operational development factors that must be considered;
- *Capture* requirements;
- *Specify* requirements in terms that are unique, concise, unambiguous and measurable;
- *Review* to provide quality assurance, and obtain stakeholders agreement;
- *Communicate* the requirements to promote a common understanding and to support reviews and approvals;
- *Manage* the requirements to maintain change and configuration controls, traceability and to assess the impact of change and risk.

Process Outputs

User requirements define the needs for a bounded operational capability as outputs or results required by users, independent of the type of system that may provide the answer. The URD is developed from a bounded, single statement of need and updated as necessary throughout the life of the system to reflect evolving user needs, affordability and changing assumptions. The URD is baselined to support project reviews, approvals and system validation, following the five-part format outlined by the URD Policy Paper [5]:

- Part 1 – General description
- Part 2 – Key User Requirements
- Part 3 – User Requirements
- Part 4 – Context Documents
- Part 5 – Index, glossary

The context documents are included to aid understanding by industry and to support the approvals process. Details will include mission profiles, support factors, the concept of employment (CONEMP) and the concept of operations (CONOPS).

The Model also identifies a series of attributes for each requirement included in Part 3:

12. Unique Identifier based on the requirement hierarchy;
13. Requirement Descriptor;
14. Effectiveness Envelope;
15. Priority (Key, 1, 2, 3 to capture the willingness to trade-off within each requirement);
16. Justification;
17. Verification Criteria, describing how the achievement of the requirement will be demonstrated;
18. Remarks;
19. Status (candidate, traded, cancelled).

The IPT-owned SRD describes in functional terms what the system must do in order to meet the user needs. An outline SRD at Initial Gate provides sufficient detail to support the analysis of options, whole-life costing and assessment phase planning. This definition is refined during assessment phase to provide a complete SRD at Main Gate that has been verified against the user requirements and accepted by the customer. This approved SRD then supports cost-capability trade-off activities, contracts and tender assessment during the demonstration phase. It also forms the basis for system acceptance prior to ISD, and is the basis for in-service upgrades necessary to maintain or enhance performance.

Key Requirements

The introduction of Smart Acquisition also required effective project monitoring, to ensure difficulties were recognised in sufficient time. Key requirements are used as an indicator of IPT and DPA performance, with the predicted achievement of 98% of key requirements providing a corporate target [6]. Key requirements are defined as individual requirements assessed as essential to the achievement of the mission need, or which are of particular interest to management.

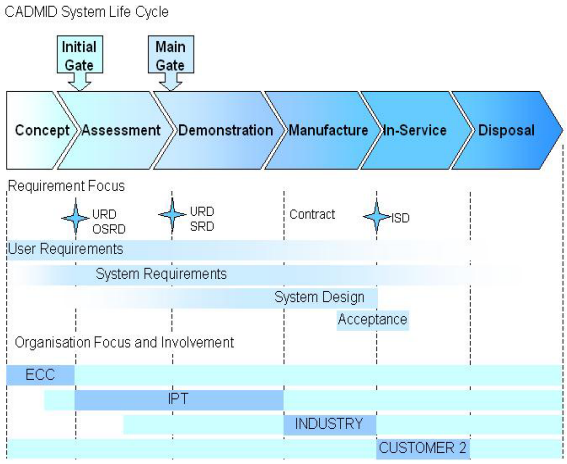

Figure 3 – Requirements and the Life Cycle

Key User Requirements (KUR) define the capability boundary, facilitating the assessment of trade-offs, feasibility and successful delivery. They are not expected to characterise the whole requirement, but to capture the critical aspects of the capability required for the investment to be effective. They also enable the measurement of IPT performance in meeting customer needs, but will be limited in numbers to ease management.

Summary

Smart Acquisition was introduced to enable the procurement of cost-effective equipment more suited to the less predictable post cold war operational environment. A Smart Requirements Model based on the principles of systems engineering encourages a capability rather than a solution led approach to progressive system definition within an Integrated Project Team. The recognition of wider lines of development and the need for integration within the wider operational context ensures systems being procured under Smart Acquisition have the potential to deliver military capability faster, cheaper and better than before.

References

1. MOD: *Smart Acquisition Handbook* (Version 5, January 2004)
2. McKinsey and Company: *Transforming the UK Defence Procurement System* (20 February 1998)
3. MOD: *The Smart Requirements Model* (18 May 1999)
4. ibid
5. MOD: *The User Requirements Document Policy Paper* (January 2000)
6. Defence Procurement Agency Business Plan (2003)

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: 31st May 2004.

RE-Publications

Perspectives on Software Requirements

Julio Cesar Sampaio do Prado Leite, Jorge Horacio Doorn (Editors)
Kluwer, 2004
ISBN 1-4020-7625-8

This is a book that deals with many of the issues facing software requirements engineering. The book is a snapshot of different types of research results that enables the reader to gain a better understanding of the discipline. The first thing that makes this book stand out from the many other publications on requirements is that many of the contributions involve a south American author. This means that, although part of the work has been conducted either in Europe or the U.S.A., with the publication we have a good opportunity to get a glimpse of the thinking that pervades requirements engineering in the southern half of America.

In the introductory chapter, the reader is provided with a statement of the editors' understanding of the term requirements engineering (RE), stressing the fact that the collection of contributions in the book deal with problems of software requirements engineering, many of which might need to be tackled when specifying systems of other natures. The domain of software requirements is partitioned into three broad areas: elicitation, modelling, and analysis, and each of the chapters in the book deals with some aspect of one of these three areas.

In the second chapter, Daniel Berry and Erik Kamsties present a good overview of the state of the art in the techniques that have been designed to manage ambiguity, together with a good description of the problems that the techniques are meant to solve. It is a collaboration of the University of Waterloo, and the University of Essen.

The third chapter is a contribution by Barbara Paech and Kirstin Kohler from the Fraunhofer Insitute of Experimental Software Engineering. It provides a structured description of the different aspects that have to be handled when using object-oriented (OO) development, and how to manage requirements in this context. It includes the decision types that should be

supported by any method integrating RE and OO, arranging them in four abstraction levels, and providing a reasonable framework for thinking of all the aspects that must be solved to build an OO system, one which comprises a good user interface and satisfies the user requirements. It also introduces a method for integrating RE and OO for user interface specification, TORE. Finally, it compares TORE with two other development methods, Armour/Miller and the Rational Unified Process (RUP). The comparison is made in terms of decision types, describing for each of them whether it is covered by the method, and by what means. It does not present any method as being worse or better than any other, but it does provide you with the means to evaluate the method and decide on whether it would satisfy the needs of a particular development activity.

William Robinson, from Georgia State University, is the author of chapter four. The research presented deals with the sources of conflict in requirements. It includes a strategy to identify conflicting requirements interaction, together with a strategy to eliminate these conflicts. The technique, called root requirements analysis, provides the means to determine the degree of conflict by calculating a set of conflict measures. The main difficulty with the technique resides in its scalability.

Francisco Pinheiro deals with requirements traceability in chapter five. It provides a good outline of traceability across all the artefacts generated in a software system development process. It stresses the advantages gained by the organisation that invests in traceability. It also refers to the difficulties any organisation must overcome to succeed. It is a good summary but more from a theoretical perspective than a practical one.

Luiz Marcio Cysneiros and Eric Yu discuss non-functional requirements in Chapter six. The material is the result of a collaboration between York University (Toronto) and the University of Toronto. It shows an approach to elicit non-functional requirements and points out future trends in their treatment. The approach is based on the use of a lexicon, the Language Extended Lexicon (LEL), to model both functional and non-functional requirements. Although it has been used in various real life case studies, it has probably only been applied to subsets of the application areas because it is not clear how it scales to large complex problems.

Rubén Prieto-Díaz provides further details on non-functional requirements in Chapter seven, narrowing the discussion to the topic of security. This is handled through the application of the Common Criteria (CC), an international standard for evaluating product security. Through the analysis of the CC the author presents an important resource of non-functional requirements.

Julio Cesar Sampaio do Prado Leite et al are the authors of the work that is described in chapter eight, which is the result of a joint effort of Brazilian and Argentinean researchers. It describes how scenarios can help in the definition and delimitation of a system's context. It includes a good introduction to the subject together with a detailed outline of a scenario-based process to specify a system's context.

Requirements specification for product line development, and agent driven requirements constitute the subjects of chapters ten and eleven respectively. The discussion on requirements for product lines describes a method for dealing both with RE and with reusability. The contents of the chapter were written by Klaus Schmid.

The final chapter in the book was authored by Jaelson Castro, John Mylopoulos and Carla Silva. It contains a discussion of agent-driven requirements that is mainly an introduction to the subject, together with a description of how a business organisation metaphor supports the task of modelling agent-based systems using object models.

Overall, this book provides a good snapshot of the contributions made by many of our South-American counterparts to the subject. Although it is possible to extract interesting bits of information throughout, I feel a common thread running through all the research efforts that were described could have improved it.

Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine

David Harel and Rami Marelly
Springer Verlag, 2003
ISBN 3-540-00787-3

Prof. Harel is one of those rare people in our times who deserves to be called a polymath (a master in many fields of learning, if your Greek is a bit rusty). Early in his career, he did pioneering research on basic computer science, such as on what can be computed, on database query languages, and on the theory of automata. More recently he invented statecharts (in 1983, aka state transition diagrams) and helped to design the commercial tools Statemate (1984, strikingly automating the use of statecharts, and showing that formal properties like reachability had a practical use in industry; the tool can generate code directly from the diagrams) and Rhapsody (1997, a model-driven software development environment). He contributed centrally to the design of UML, which incorporates statecharts and sequence charts to define system behaviour. He's well-known for his wonderful book *Algorithmics: The Spirit of Computing* (1987, 3rd edition 2003).

With this background, it isn't surprising that Harel, ably assisted by his graduate student Rami Marelly, has turned his attention to scenarios, and in particular to making them both precise and executable. Scenarios can be represented in a limited way (without nuances such as what is mandatory, and hence, which scenarios are actually possible!) as UML sequence diagrams, a notation which shows a vertical timeline for each role -- human or machine -- of interest, with boxes on the timelines to indicate when each role is active, and connecting lines to show communication between roles (including software objects). The ITU Message Sequence Chart (MSC) is similarly widely adopted, but limited in its expressive power.

Harel has therefore extended the idea of defining behaviour by inventing (1998, with Werner Damm) the Live Sequence Chart (LSC), essentially an MSC with a richer visual language to express constraints on what is allowed to happen. As Harel is a practical engineer who likes to show that his ideas work, the book includes a CD with working software, the "Play Engine" -- fun, but described as "not a commercial product" -- and there's a website (http://www.wisdom.weizmann.ac.il/~playbook) where you can find out more.

In Harel's view, conventional development rather weakly connects use cases to requirements, and similarly weakly connects requirements to tests and other forms of verification. Code generation or synthesis is mainly by hand. He suggests that a better approach is for system developers to "play in" scenarios into a system's GUI (or object model diagram if there is no GUI), "clicking buttons, rotating knobs and sending messages (calling functions) to hidden objects, in an intuitive drag & drop manner." The developer using the Play Engine plays in both the incoming events (e.g. clicks) and "the desired reactions of the system and the conditions that may or must hold." The Play Engine performs the magic of simultaneously showing the current status on the GUI and constructs the corresponding LSCs to record the played-in scenario.

The developer can then at once "play out" the recorded scenario to test that the behaviour is as intended. The Play Engine lets the developer play at using the system as if the product under development already existed (Harel does use the horrible words user and end-user to mean developer and product operator). The GUI does what it should, and the system under development behaves as an executable model.

So far so simple. The rest of the book illustrates these concepts with worked examples (in colour), gradually introducing all the complexities that are needed to give LSCs precise semantics rich enough to capture scenarios, variations, exceptions, and indeed "anti-scenarios", anything that is required NOT to happen. (These aren't the same thing as threats or misuse cases, though those can generate specific anti-scenarios, namely ones with hostile intent. Many anti-scenarios are simply errors, like lift doors opening when they shouldn't.)

The LSC notation is inevitably more difficult than familiar MSCs or sequence diagrams, as it has more work to do. It is quite code-like, as indeed it has to be as precise as code. Harel is plainly sensitive to the complaint that LSCs would therefore be just as tricky to debug as code, and tools such as the Play Engine are clearly part of his answer: the notation is quite formal with an exact meaning, but that meaning can be translated into terms that everyone can understand and validate, namely a system's behaviour as seen via its GUI. Thus there are in fact three levels of formality: the pages of logical definition and proof at the ends of the chapters in the body of the book; the graphical LSCs whose meaning is thus formally defined; and the scenarios that correspond to those diagrams. This is a satisfying answer, but for the fact that the Play Engine is still largely a toy. Does it only apply to software? Not necessarily; any system that can be modelled as communicating objects (The Accounts Department, Purchasing, ...) could be so treated, so in principle it applies to systems of all types, whether software, hardware or peopleware.

Will LSCs be as successful as statecharts? Will Play-in and Play-out be part of the development methods of the future? Maybe they will; but in any case, making development less hit-and-miss, and more closely tied to requirements and scenarios, is certain to remain a challenge to researchers and industry alike.

Come, Let's Play is a book that can be read at different levels by different audiences: few other than researchers will check the formal proofs that are written out in full; students may learn the semantics of the LSC, and do projects with GUIs and code generation; while curious practitioners will probably skim the book and perhaps pass an evening playing with the demonstrations.

Practical Foundations of Business System Specifications

Haim Kilov and Kenneth Baclawski (Editors)
Kluwer, 2003
ISBN 1-4020-1480-5

This is the third in the series of OOPSLA Workshops on Behavioral Semantics led by Haim Kilov (the others were in 1996 and 1999). It is a simple collection of workshop papers, linked only by the general theme, the preface, and the index.

Kilov's position is clearly restated in the preface: the need to be 'abstract, precise, and explicit'. The preface is headed by quotations from Roger Bacon and others. In it the editors write:

> One of the unfortunate characteristics of computing science and software engineering is a noteworthy lack of interest in work done "long ago". It is taken for granted by many that a two-year old book could not possibly still be relevant. Yet the computing classics published in the 1960s show that many concepts considered new to be a recent invention have existed for a long time, perhaps under different names. This includes, among many others, such concepts as pair programming, component factories, the gross inadequacies of box-and-line diagrams, and the confusion generated by a set of tacit assumptions. Then, as now, software engineering - especially including business and system specifications - ought to be based on the types of theoretical foundations and practical disciplines that are traditional in the established branches of engineering.

In other words, things would be better if people read books, and then used them to write better specifications with diagrams that meant something. Some hope. However the editors wisely caution:

> Different users of specifications, especially business experts, may have serious difficulties in *reading* and understanding them due to the complexity and size of the specifications.

But they at once optimistically conclude:

> However if the basic concepts are explained (and the irrelevant details ignored via abstraction) then these difficulties become resolved.

This then is the challenge: to move away from human vagueness to improve systems development, while staying close enough to humans to improve requirements capture and validation.

The papers are arranged purely alphabetically. The editors somewhat despairingly confess:

> While it is always tempting to provide a classification for a collection of papers, once again we admit defeat.

So readers have no option than to play lucky dip and hope for a paper relevant to their work; of course they can search for keywords or the names of authors they know. Given that there are strong themes in these volumes, it would be possible to create a collaborative work which dealt with each theme in turn, arguing the pros and cons of alternative approaches. That would certainly be more approachable.

As for the papers themselves, UML looms ever larger, and along with RM-ODP more and more of the papers address aspects of its formalisation. For example, Kilov himself, with Michael Guttman, has contributed a paper introspecting about modeling the process of business modeling; and one with Othmar Bernet arguing the case for diagrams with precise meanings. 3 authors from Microsoft Research in Redmond write about .NET Contracts: Attaching Specifications to Components.

The volume will mainly be of interest to researchers working in the field of precise specification.

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

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

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

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

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/

For 2004, *Springer-Verlag* are continuing to offer RESG members a substantial discount on subscriptions to the REJ. Members can subscribe for only £38 (print + online) or £27 (online only). See www.springeronline.com.

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

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

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*.

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: Prof. Neil Maiden, Centre for HCI Design, City University, Northampton Square, London EC1V OHB, UK. E-Mail: N.A.M.Maiden@city.ac.uk.

Secretary: David Bush, National Air Traffic Services, UK. E-Mail: David.Bush@nats.co.uk.

Membership secretaries:

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.

Dr 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.

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 officer: Dr Sebastian Uchitel, Department of Computing, Imperial College, 180 Queen's Gate, London SW7 2BZ, UK. E-Mail: su2@doc.ic.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.

Student Liaison Officer: Carina Alves, University College London, Department of Computer Science, Gower Street, London WC1E 6BT, UK. E-Mail: c.alves@cs.ucl.ac.uk

Industrial liaison:

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

Elena Pérez Miñana, Philips Digital Systems Lab. E-Mail: elena.perez-minana@philips.com.

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

Gordon Woods, Independent Consultant, E-Mail: Gordon@cigitech.demon.co.uk

Requirements Engineering Journal

For 2004, *Springer-Verlag* are continuing to offer RESG members a substantial discount on subscriptions to the REJ. Members can subscribe for only £38 (print + online) or £27 (online only). See www.springeronline.com.