BCS™

# Requirements Quarterly

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

© 2008 RESG http://www.resg.org.uk RQ48 (June 2008)

## *Contents*



## RE-soundings

### From the Editor

This RQ is a Special Issue, on Adaptive RE, and my last – at least, for now; and I expect I will continue to review books and write a RE-flection from time to time.

My new book, Discovering Requirements, is scheduled to appear in March 2009. It has recently gone down the hatch for formal review by the publishers, John Wiley, so when I come back from my summer holidays I will probably have a lot of editing to do. I hope to have a book launch party sometime in the spring – RQ will keep you informed.

Pete has done a wonderful job as chairman and an equally good job of this Special Issue. I hope RQ will continue to run a "Special" every year to ring the changes from reporting on events.

As well as writing, I have been developing new course materials for teaching RE. It seems that the market is at last changing rapidly in a direction long foretold: towards shorter, modular courses and online instruction. Curiously, this seems to demand more, not less from printed books and face-to-face seminars, courses, and workshops. The RESG, too, is looking at an increased web presence, with more opportunities to participate for people near and far. I hope you'll continue to take part in the RESG's journey of discovery.

*Ian Alexander, Scenario Plus*

### Chairman's Message

This issue has reminded me why Ian has been such a great RQ editor. While my job has been confined to commissioning two pieces on adaptive systems (from Nelly Bencomo and Jon Whittle, and from Mohammed Salifu), Ian has been busy with overall editorship, writing articles and comment and (co-)running a big event (see RE-flections). Oh. And sending me gentle, but essential reminders to get on with it. Ian's done all this while finishing his latest book. Readers need not

fear the loss of Ian from the editorship too much. He will still be a big cog in the RESG and doubtless we will prevail upon him for the odd article.

Well, here is Ian's last RQ and I hope you will agree that it's well up to standard. If you don't, come along to the RESG party, soapbox and AGM at Imperial College from 4.00pm on the 10 July and give us some stick. I hope to see you there.

*Pete Sawyer, RESG Chair*

## RE-treats

For further details of all events, see www.resg.org.uk Forthcoming events organised by the RESG:

### RESG Party, Soapbox and AGM

10 July 2008, Imperial College, London

As last year, we'll be having a mid-summer networking event. This is an opportunity to meet other members of the specialist group, harangue the committee, hear what we have in store for the coming year and, if you have an opinion you want us all to hear, have a go on the soap box. There will be drinks, nibbles and music. It's timed to help at least some people drop by after work so starts at 4.00pm and goes on into the early evening. We hope to see you all at Imperial College on the 10th July. See the website for more details.

Contact Pete Sawyer

### REET'08

at RE'08, September 8/9, 2008, Barcelona

The RESG is happy to be co-sponsoring this year's 3rd International Workshop on Requirements Engineering Education and Training (REET), to be held on the 9th September alongside RE'08 in sunny Barcelona. The one-day workshop will address issues related to RE education, both as part of a formal university degree and as ongoing skills training within the workplace.

RE education and training is increasingly recognized as a critical component in the success of a software development project. This has led to a growing identification of the importance of incorporating significant RE components into the curriculum of university degrees in Software Engineering, Computer Science, Information Technology and other related areas. Furthermore many industrial organizations are recognizing the need to develop RE related training programs as part of their ongoing process improvement initiatives.

If you have something to add to a discussion in this area, please submit a paper or simply sign up to attend. Full details are available on the both the RESG website (www.resg.org.uk/sources.html) and the REET'08 website (re.cti.depaul.edu/REET08/). In addition to topics related to curriculum development, creative contributions related to pedagogical techniques for teaching RE skills are strongly encouraged. These skills include requirements elicitation, modeling, analysis, conflict negotiation, consensus building, and requirements specification writing and reviewing skills. Submissions could take the form of experience reports or demonstrations of specific teaching techniques and training materials.

Paper submission deadline is the 4th July. Please consider sending something!

### Ubiquitous Requirements

15 October 2008, London

Computers are becoming ubiquitous and so are the requirements for the applications that they run. To date, related areas of ubiquitous, pervasive and ambient, computing have been technology-led. The technology is maturing very fast, however, and we are starting to see real applications. But is RE ready? This event will explore what is special about ubicom for which RE needs to find the answers.

Contact Pete Sawyer

### So You Want to be a Requirements Analyst

2pm, 5 November 2008, University of Westminster, London

Maybe you are thinking about becoming a requirements analyst. Do you know what kind of life can you expect? What your biggest joys and deepest sorrows will be? How you will spend your days? What skills you will need? Are the things that are taught in universities and described in books actually useful? During this event, several practising requirements analysts will tell you what their job is really like.

Contact Emanuel Letier, University College London

### PhD Student Event

December 2008, London

Contact Dalal Alrajeh, Imperial College

### Creativity Tutorial

March 2009, London

Contact Neil Maiden

# RE-calls

Recent Calls for Papers and Participation

## RE'08

The theme of this year's conference is RE for a sustainable world.

September 8-12, 2008, Barcelona, Catalonia, Spain
http://www.re08.org/

## Mastering the Requirements Process

3 Days, 15-17 September 2008, London
http://www.irmuk.co.uk/1/

## Introduction to Requirements

2 days, 21-22 October 2008, The IET, London, presented by Ian Alexander, Scenario Plus
http://www.theiet.org/courses

# RE-writings

## To Adapt or Not to Adapt: that is the Requirements Question

Nelly Bencomo and Jon Whittle, Lancaster University

On a wintry January evening somewhere in southwest Germany, a group of forty researchers and practitioners scrambled off a train and bundled, windswept and bedraggled, into a series of waiting taxicabs for a forty-five minute journey into nowhere. We left Frankfurt railway station at four o'clock. It was already dark. So when we arrived two hours later at Türkismühle, it could just as easily as have been any small town in central Europe. The taxi ride did not make things any clearer. As we trusted ourselves wholeheartedly to the cab driver, a few of us turned to each other and wondered if we were indeed at the start of a week of academic meetings or had unwittingly found ourselves as protagonists in a film noir thriller. Fortunately for us, the former was indeed the case.

**Participants of a Schloss Dagstuhl Seminar**

So what *were* we doing here? Well, as it turns out, we were the lucky participants of a Schloss Dagstuhl seminar entitled *Software Engineering for Self-Adaptive Systems*. The aim of this seminar was to bring together smart people from all walks of the software life, including requirements engineers, software architects, middleware and programming language experts. Our remit was simple. Within the space of five short days, we were to come up with solutions for the very pressing problems of how to systematically design and build "self-adaptive systems".[1] A self-adaptive system is a system that is able to autonomously modify its behaviour in response to environmental changes. Despite the science-fiction sounding nature of the term, such systems are now regularly deployed in enterprise computing, embedded and pervasive systems. As yet, however, we lack a clear understanding of how to engineer the software for these systems. Traditional methodologies break down because of the inherent flexibility required for self-adaptation.

As it turned out, perhaps the most exciting part of the seminar (at least to RESG members!) were the very extensive discussion sessions on requirements engineering. A small group of renegades joined forces to debate the topic and, over beef roulade and apple cake during the day, and, cambozola and weissbier in the evenings, eight of us managed to come to some sort of conclusions to the following question: what does requirements engineering for self-adaptive systems mean? As well as ourselves, the starring actors were: Betty Cheng, Anthony Finkelstein, Jeff Kramer, Jeff Magee, Sooyong Park, and Schahram Dustdar.

It turns out that only preliminary work has been done on this topic. Immediately, a number of important but as yet unanswered research questions came to mind. Self-adaptive systems must continuously monitor changes in their context and react accordingly. But a system cannot monitor everything all the time because of the vast resources that would be needed. Therefore, from a RE point of view, one must ask which aspects of the environment should the self-adaptive system monitor? And exactly what should the system do if it detects a less than optimal pattern in the environment? The system still needs to maintain a set of high-level requirements that should be enforced regardless of the environmental conditions. But non-critical

[1] A final report, "*Software Engineering for Self-Adaptive Systems: A Research Road Map*" was the main output of this seminar.

requirements could well be relaxed, thus allowing the system a degree of flexibility during or after adaptation. Clearly, self-adaptation entails new perceptions of the way requirements should be conceived in comparison with the traditional practices of focusing on static goals.

One of the main challenges that self-adaptation poses is that when designing a self-adaptive system, we cannot assume that all adaptations are known in advance – that is, we cannot anticipate requirements for the entire set of possible environmental conditions and their respective adaptation specifications. RE for self-adaptive systems, therefore, must deal with *uncertainty* because the expectations on the environment frequently vary over time. For example, if a system is to respond to cyber-attacks, one cannot possibly know all attacks in advance since malicious actors develop new attack types all the time. As a result, requirements for self-adaptive systems may involve degrees of uncertainty or may necessarily be specified as "incomplete". The requirements specification therefore should cope with:

- incomplete information about the environment;
- incomplete information about the respective behaviour that the system might expose;
- evolution of the requirements at runtime.

Our endeavours led us to formulate a number of short- and long-term research challenges in RE for self-adaptive systems. We think that these challenges can be progressively tackled. We outline these below, starting with shorter-term challenges and progressing to more visionary ideas.

**Challenge one: a new requirements language for self-adaptation**

Current languages for requirements engineering are not well suited to dealing with uncertainty. We suggest that richer requirements languages are needed. For example, in goal-modelling notations such as KAOS and i*, there is no explicit support for uncertainty or adaptivity, and scenario-based notations generally do not provide any support either. Certainly, the most frequent notation for specifying requirements in industry is still using natural language prose. Traditionally, requirements documents make statements such as "the system shall do this". For self-adaptive systems, the prescriptive notion of "shall" needs to be relaxed and could, for example, be replaced with "the system may do this" or "if the system cannot do this, then it should eventually do that." This leads us to believe that a new requirements vocabulary for self-adaptive systems is needed, that gives stakeholders the flexibility to explicitly account for uncertainty in their requirements documents. In its simplest form, this new language might be just a set of keywords describing common forms of flexibility:

Traditional RE:

- "*the system SHALL do this ...* "

Adaptive RE:

- "*the system MIGHT do this ...*"
- "*…it MAY do this... AS LONG AS it EVENTUALLY does this ...* "
- "*the system OUGHT to do this... but if not, it SHOULD EVENTUALLY do this ...*"

A more complex version would allow for a hierarchy of requirements that are less or more critical and, therefore, less or more adaptable. An even more complex version might be a formal notation with concepts such as those given above defined precisely. In any case, such a vocabulary would change the level of discourse in requirements from prescriptive to flexible. There would need to be a clear definition of terms, of course, as well as a composition calculus for defining how the terms relate to each other and compose.

**Challenge two: mapping requirements for adaptation to adaptive architectures**

Given a new requirements language that explicitly handles uncertainty, it will be necessary to provide systematic methods for refining models in this language down to specific architectures that support runtime adaptation. There are a variety of technical options for implementing reconfigurability at the architecture level, including component-based, aspect-oriented and product-line based approaches, or hybrid solutions. However, there could be a big gap in expressiveness between a requirements language that incorporates uncertainty and these existing architecture structuring methods.

**Challenge three: managing uncertainty**

Introducing uncertainty into software engineering processes implies the need to manage this uncertainty. In this sense, some requirements will be considered as invariants (unchangeable), while others will permit some degree of flexibility. For example, a system cannot start out as a transport robot and self-adapt into a robot chef! What we want to emphasize here is that the original intent cannot change. Allowing uncertainty levels when developing self-adaptive systems requires a trade-off between flexibility and assurance such that the critical high-level goals of the application are always met.

**Challenge four: requirements reflection**

As noted above, self-adaptation deals with requirements that vary at runtime. Therefore, it is important that requirements lend themselves to be dynamically observed, i.e. during execution. Reflection enables a system to observe its own structure and behaviour to potentially reason about the observations. Leveraging and extending beyond complementary approaches, Finkelstein coined the term "requirements reflection" that would enable systems to be aware of their own requirements at runtime. This would require an appropriate model of the requirements to be

available online. Such an idea inspires interesting research questions, such as: could a system dynamically observe its requirements? In other words, can we make requirements runtime objects? Further research is needed to examine how current and new technologies may provide the infrastructure to do this.

**Challenge five: online requirements refinement**

By definition, self-adaptive systems react to changes in their environment. But in the future, they should also be able to react to changes in their requirements. In particular, new requirements might be added at run time, and the system may even add new requirements itself. For example, the isolation and long duration of deep space exploration missions could easily lead to a circumstance where a spacecraft should add new requirements to deal with situations that simply could not be conceived of by its designers. The natural consequence of such thinking, as pointed out by Kramer and Magee [3], is that RE processes should be performed *at run time*, so that new requirements can be dynamically added and autonomously refined into a design that satisfies the requirements by making use of existing capabilities.

**A final note …**

**It's Not the Arriving but the Journey that Matters**

As with all such meetings, our Dagstuhl seminar succeeded more in articulating problems than solutions. As the train pulled out of Türkismühle station, however, we realized with enthusiasm what great challenges lay ahead for the RE community. It was now day time. We had been "locked up" with each other for an intense week of brainstorming and contemplation. In striving to get as much out of the sessions as possible, we hardly had time to notice our beautiful surroundings. Still, there was a two hour journey to the airport remaining. Perhaps there was yet time for the German countryside to inspire some great ideas…

*After disseminating and discussing the results from the seminar with our colleagues and during the SEAMS'08 workshop at ICSE'08, we have found that the terms* uncertainty *and* incompleteness *in requirements, and* evolution *of requirements at* run time *can provoke engaging and controversial discussions. Therefore, we look forward to your comments and feedback!!!*

**References**

Dagstuhl Seminar Proceedings 08031, *Software Engineering for Self-Adaptive Systems*, http://drops.dagstuhl.de/portals/index.php?semnr=08031

Software Engineering for Adaptive and Self-Managing Systems Workshop (SEAMS 2008) at ICSE'08 http://www.hpi.uni-potsdam.de/giese/events/2008/seams2008/

Jeff Kramer, Jeff Magee, *Self-Managed Systems: an Architectural Challenge*, 2007 Future of Software Engineering (FOSE'07), p.259-268, May 23-25

## RE in Context-Aware Software

Mohammed Salifu, The Open University

**Introduction**

The taking up of 'smartphones' has increased the provision of mobile services and applications in areas such as mobile banking, social networking and general entertainment. To manage these multifaceted services, there is a need for quality of services such as usability and efficiency when smartphones are used in different environments. In meeting this challenge, smartphones should be made context-sensitive.

Context sensitive devices require context-aware software applications, which monitor changes in their environment and switch their behaviour in order to continually satisfy requirements. Therefore, context-aware applications belong to a general class of software systems also known as self-managing or autonomic systems.

This article outlines the challenges of context-awareness problems and scopes our approach via the use of problem-oriented analysis.

**Context-Awareness Scope**

Specifying monitoring and switching in context-aware applications can be difficult due to their dependence on varying environmental properties. Also, given the need for self reliance (i.e., minimal human intervention), self-managed systems behaviours must be rigorously analysed and their specifications formally verified so as to assure confidence. Such analysis requires reasoning about self-managing systems at different levels of abstraction, from the problem space to the design space. To this end, Kramer and Magee [1] have proposed an architecture-based approach, which consists of three layers in abstraction: Component Control at the bottom, Change Management in the middle, and Goal Management at the top. This approach aims to provide a link between the design space (component control level) and problem space (goal management level). Underpinning this approach is the application of the standard principle of separation of concerns. Kramer and Magee's approach represents a refinement of a long list of architectural approaches in dealing with self-managing systems [1].

We share in the view that self-managing systems must be analysed at different levels of abstraction. Therefore, in addition to the 'vertical' separation of concerns across different layers, we advocate 'horizontal' separation of concerns within layers. Also, we observe a need for a 'problem analysis' step between the 'goal management' and 'change management' layers. This is because while goals capture the intentions of stakeholders, they do not necessarily bring to light the underlying contextual constraints that must be addressed in determining whether the solution will satisfy the goal. Adequate analysis of the problem context, beyond the intentions of stakeholders, is imperative in context-aware applications due to the need for self reliance and the impact of the context on continual requirements satisfaction.

Within our problem analysis, we have identified three categories of context-awareness problems that require systematic and detailed analysis: (a) the classification of different application behaviours for different contexts; (b) the monitoring of environmental properties to assess their impact on continual requirements satisfaction; and (c) the selection of appropriate matching of different behaviours that ensure requirements satisfaction in all contexts. Our problem-oriented approach is aiming at analysing these categories of problems in deriving specifications for context-awareness.

**RE in Context-Awareness**

We discuss the analysis of the three categories of problems as follows.

*Problem Description:* The role of context in analysing nearly all software applications is widely recognised. Given the pivotal role of context in context-aware applications, problem descriptions for this class of applications require a clear separation of concerns among context, requirements, and specifications. Such a separation provides a means for the identification of the underlying causes that motivate variations in application behaviour. For example, we are able to identify situations in which the requirements remain stable but changes in contexts necessitate different specifications in satisfying the same requirements. Therefore, we found use of three descriptions for a problem proposed by Jackson: (1) a description of the context in which the problem resides in terms of known domain properties of the world, denoted as the *world W*; (b) a description of the required properties of the world, denoted as the *requirement R*; and (c) a description specifying what the computer system running the software-to-be must do to meet the requirements, denoted as the *specification S*. These three descriptions are related by the formula [5]:

$$W, S \vdash R$$

The symbol "$\vdash$" denotes entailment, that is, the satisfaction of S in W entails that of R. The "," is the separator for context and specifications.

*Context Analysis*: Having a sound conceptual basis for separating the concerns of context, requirements and specifications only takes us so far. To fully ascertain the impact of contextual changes, further refinement of the context of applications is required. This is because we both need to identify the relevant contextual variables whose changes may violate requirements satisfaction, and the dependency among the variables that may constrain the activities of context-awareness (i.e., monitoring and switching).

Given that context-aware applications must depend on themselves for continual requirements satisfaction, a formal analysis of the context, possibly with tool support, is required to assure confidence. Therefore, in addition to the incorporation of a systematic separation of concerns, we have provided a practical characterisation of concepts for refining context. This enables us to analyse the relationships between contextual properties and the satisfaction of requirements, as activities of context-awareness [2].

*Individual Problem Analysis:* Individual problem analysis focuses on concerns that must be addressed in preparation for context-awareness. Therefore, the focus is on the analysis of the changeable context and its impact on deriving different behaviours; the monitoring of an individual environmental property; and the switching from one application behaviour to another. The problem analysis of monitoring each (individual) contextual variable addresses the concerns of monitoring an informal environment and the fidelity of the information obtained, which creates a need to verify and validate the adequacy of the output of monitoring. Furthermore, where a contextual variable is not directly observable, a transformation may be required in identifying a more observable equivalent. Similarly, the problem analysis of switching between a pair of behaviours addresses the concern of identifying appropriate contextual situations at which switching can be carried out. Finally, we need to analyse and resolve the conflicts between the need for continual requirement satisfaction and the constraints of the context that inhibits switching.

Our approach [2, 3] provides concepts for an informal but systematic analysis of application contexts, which can then be used to derive proper behaviour for changing context through monitoring and switching. The analysis of individual problems is informal while the relationships between them are formalised. This provides flexibility in analysing individual problems while imposing necessary constraints on the derived context-awareness specifications.

*Problem Analysis of Context-Awareness:* The problem analysis of context-awareness aims at deriving appropriate conditions for the composed monitoring and switching activities and for assessing their impact on satisfaction levels of context-sensitive requirements. Different parts of the requirements for applications such as security and performance exhibit different sensitivity to changes in the operating environment.

The relationship between context-sensitive requirements and contextual changes represents a form of contextual dependency. Knowledge about this form of contextual dependency and those among contextual variables provide a means for enhancing efficiency in monitoring and switching. Using contextual dependencies, one can avoid monitoring all variables all of the time and avoid 'non-essential' switching, as lowering the satisfaction level of requirements following a contextual change, may not necessarily violate the requirements.

Using our characterised concepts for refining context, we formulate two theorems for monitoring and switching, which define the necessity criterion for monitoring a contextual variable, and necessary condition for switching application behaviour [4]. These theorems guide a formal analysis of the overall context-awareness problem. Given different specifications for different context situations (derived by individual problem analysis) and knowledge about contextual dependencies, confined by these theorems, we are able to encode a constraint satisfiability formula, the solution of which produces a specification for context-aware behaviour. This forms the basis for automated analysis of the impact of varying context on monitoring and switching, and for verifying the resulting context-aware behaviour through simulation.

*Automated Analysis Support:* In order to manage the context-awareness problem space and analysis process, some form of tool support is necessary. This is because the identification of relevant contextual variables and derivation of different behaviours is iterative and time-consuming. Therefore, there is a need for on-the-fly verification and validation of newly added behaviours to ensure that the existing context-awareness control 'engine' is not broken.

Our approach enables us to transform concerns in problem frame based descriptions into statecharts and process models; benefiting from using the logical relation $W, S \vdash R$. Using built-in simulations in process models and statechart with our own tool, one can validate context-awareness specifications and demonstrate to clients without requiring full implementation. Showing the validation specification explicitly as a state machine or process model, one can continually verify the model even when new behaviours are added by executing a recompiled simulation.

**Conclusion**

We have outlined the scope and challenges of the context-awareness problem and our contribution through the use of our problem-oriented approach in addressing them. This has highlighted the role of context-aware software in developing smart devices, capable of adapting their behaviour in response to environmental changes. Further challenges by way of identifying patterns of monitoring and switching behaviours that facilitate 'normal' problem analysis are ongoing.

**References**

1. Kramer, J. and J. Magee, *Self-Managed Systems: an Architectural Challenge*, in Proc 29th Int. Conference on Software Engineering (*ICSE'07*) Minneapolis, 2007.

2. Salifu, M., Nuseibeh, B., Rapanotti, L., Tun, T., *Using Problem Descriptions to Represent Variability for Context-Aware Application*. in *First International Workshop on Variability Modelling of Software-Intensive Systems*. 2007. Limerick, Ireland: Lero.

3. Salifu, M., Nuseibeh, B., and Yu, Y., *Specifying Monitoring and Switching Problems in Context*, in Proc. *15th IEEE International Requirements Engineering Conference*. 2007: Delhi, India. p. 211-220.

4. Salifu, M., Yu, Y., and Nuseibeh, B. *Analysing Monitoring and Switching Requirements using Constraint Satisfiability*. in *Technical Report -ISSN 1744-1986*. 2008: The Open University.

5. Zave, P. and Jackson, M., *Four Dark Corners of Requirements Engineering*. ACM Transactions on Software Engineering and Methodology, 1997. **6**(1): p. 1-30.

# RE-flections

## Getting the Best from Scenarios in Your Project

(Tutorial and Seminar) 1-day Event, Tuesday 10th June 2008, Northampton Suite, City University, London

**Morning Tutorial: Introduction to Scenarios**

In the morning, Ian Alexander gave a tutorial which looked at the nature of scenarios in all their many forms, from stories and storyboards to fully-dressed use cases. The pros and cons of the different styles were explained.

It then examined ways of discovering scenarios, with practical workshop techniques and common scenario patterns. Use cases, being extremely popular but poorly understood, were explained in some detail.

Exceptions are key components of use cases; the tutorial explored how to use scenario structure to improve the search for exceptions, and how to identify the ones that really matter.

The tutorial then looked at negative scenarios, including intentional threats and forbidden combinations, and considered how to discover them.

Finally, the valuable relationship of scenarios and test cases was examined. Ways that you can use scenarios to start working on tests early in a project were discussed.

**Debriefing a Group Exercise**

The tutorial was accompanied by group exercises in scenario discovery.

**Afternoon Seminar: Techniques**

**Paul Grünbacher,** Johannes Kepler University Linz, Austria, spoke on Scenarios in the Wild: Experiences with Mobile Tools for Scenario-Based Requirements Discovery.

In workshops, "people have only limited ability to describe what they do and how they do it without immediate access to the social and material aspects of their lives", said Grünbacher, quoting Blomberg et al (2003). Workshops are costly, and it is hard to get the busiest but most needed stakeholders to attend them. Further, participants may not understand the dynamics of scenarios; and unexpected events are difficult to simulate.

An alternative is to observe users, as for instance with Beyer and Holtzblatt's Contextual Inquiry. Their book, however, is quite abstract; people like to be guided by a definite method with steps to follow one at a time. So, said Grünbacher, we put the Art-Scene scenario presenter on a mobile device (a PDA rather than a laptop, to avoid getting in the way of operations) to enable the capture of requirements during periods of actual observation. The mobile approach (MSP) is not limited to capturing text; multimedia elements such as audio clips can be used to describe requirements. For example, he played a clip of pilots talking to air traffic control, recorded on the PDA (from within the scenario presenter tool).

The result was to gather requirements at about 8x the rate of a workshop, and at less cost because only one stakeholder was interviewed / observed at a time (for just 15 minutes each).

Grünbacher did in fact try doing this "in the field" – in fact on a mountain – to capture navigation requirements for cross-country skiing (much like Ian Alexander's tutorial example). It didn't work very well, as the skiers were much faster than the analysts; and the low temperature exhausted the batteries in 45 minutes, causing the analysts to resort to an unplanned use of pencil and paper.

Currently the tool does not allow scenarios to be edited in the field; you can just add requirements to them. The scenarios were generated from use cases, ie the analysts were presented with pre-defined scenarios. Neil Maiden agreed that this was over-prescriptive; new scenarios and variations are continually observed in the field. The tool does however allow you to annotate a scenario so you can ask questions later.

Scenarios can thus be used in workshops; with visualizations; or on a mobile device, in the field. All have their uses.

**Paul Grünbacher with Mobile Scenario Presenter on his PDA**

Prof. **Jon Whittle**, Lancaster University, spoke on Brainstorming Potential Attacks with Executable Misuse Cases.

It's no longer enough just to consider security at the coding stage: you need to start with model-based security engineering far earlier in development. Relevant models include scenarios of various kinds, eg using attack trees. Making such models executable allows people to see attacks actually succeeding or failing, which brings security concerns to life. "Red teams" can then actively devise security attacks; results can be fed directly into design models at an early stage. Better, ways to handle attacks, once understood, can be used again and again.

Misuse cases are use cases from the point of view of an actor hostile to the system under design. Hence, misuse cases are undesired behaviours that threaten existing use cases, but can be mitigated by new mitigation cases. By recording such cases and their relationships

(threatens, mitigates), you can execute various possible security scenarios.

For example, voters should not be able to vote many times; should be able to remain anonymous; and should have their votes recorded correctly. But quite simple attacks are possible on all three of these requirements. Whittle models the wanted scenarios using modified UML sequence diagrams; but structured text or tables could probably do a similar job.

Attacks are modelled in four parts: a) model the whole normal story as an overview; b) model the modification scenario the attacker needs to prepare to break in (eg, reprogramming a smart card to ignore a disable message); c) model the attack scenario itself; and finally d) model a mitigation scenario to counter the attack.

Of course, there might be other attacks; so you need to repeat the later steps for each further attack you can identify. Whittle has a tool (built on top of the IBM Rational Software Modeler, including a use case simulator and finite state machine generator, and a generic aspect-oriented modelling tool) which automatically composes the scenarios into the original overview diagram(s). This seems to be useful for quite a wide range of attacks – not all: for instance, attacks that depend on knowledge of a specific data structure can clearly only be modelled and countered at the coding stage.

**Case Studies**

**Alistair Mavin**, Rolls-Royce, spoke on Using Scenarios to Discover Requirements for Aircraft Engine Control Systems.

**Alistair Mavin with Airbus A380**

Engine control systems operate in a harsh environment – from -80 to +50 degrees Celsius, for instance; they are safety-critical; stringently regulated; and must provide ever increasing functionality on decreasing development timescales. Rolls-Royce itself demands more diagnostic and prognostic information.

A typical modern control system has built-in redundancy with many sensors and dual channels to ensure reliability, resulting in over 100,000 lines of code developed with up to 20 suppliers.

It is easier for people to recognise stated requirements than to recall unstated ones, so a tool (Art-Scene) presents concepts and asks what-if questions about possibly-missed requirements. Many of these turned out to be typical HAZOP-type threats, so the tool's repository was extended with known hazards. For instance "missing", "lower", "partial loss", "reversed", eg "What if the temperature signal between temperature sensor and control system is missing?" is proposed as a possible exception.

As a case study, variable stator vanes (VSVs) can be rotated to improve the efficiency and operating range of an engine's compressor. The tool guided a non-specialist systems engineer to identify exceptions and handle them with new (derived) requirements. It took 3 hours to go through the VSV scenario, and discovered an average of 36 requirements per event (and record a justification for each one). As a control for the study, an experienced safety engineer used a traditional HAZOP approach. That took 4 hours and found an average of 20 requirements per event. The requirements were more detailed, and were also found earlier in development. It thus looks as if Art-Scene is both quicker and more effective, using less experienced people; but the set-up time was not included. It does seem however that a multidisciplinary review of requirements, design, and derived requirements is justified; safety engineers are thus allowed to concentrate on key system properties.

Future work will look at how to handle multiple failures, and exceptions in all life-cycle phases (not only operational use, for instance). On the tooling side, the use of graphical representations, and possible integration with a requirements management tool will also be investigated.

**Neil Maiden**, City University London, spoke on Using Storyboards in Requirements Processes.


**Neil Maiden on Storyboards**

Storyboarding was developed back in the 1930s film industry with Walt Disney especially. Now the technique is in wide use in film, theatre, "animatics",

business and interactive media. Storyboards are popular in interaction design, and the same techniques are equally valid in requirements discovery – indeed, there is no reason for any barrier between these disciplines, said Maiden. The interesting thing is not what visualization techniques are used, but what they can achieve, he suggested. They can support use cases and scenarios; help in walking through interaction designs as early prototypes; enable exploration of business situations and contexts; and in expressing complex concepts of operation.

Storyboards (one from Star Trek was shown) are powerful, expressive (you could do a whole use case like that), and immediate. The work involved would usefully cause excessively fine grained detail to be suppressed, leaving the main points at issue clearly displayed.

Brainstorming is often proposed as a requirements workshop technique, but in practice it is horribly difficult to get much from such a session, as the outputs are so unstructured. But with a storyboard to show episodes, it is easy to group suggested requirements under the relevant storyboard frames. This gives them a context and rationale with little effort, making them immediately comprehensible.

In a creativity workshop, giving people a template with 16 empty boxes to draw in and annotate below, it turns out that people can readily combine drawings and written ideas into a structured storyboard – creating ready-structured scenarios much earlier than usual on projects.

This has worked well in air traffic control, a very conservative domain, where people demand something more rigorous after the workshops. There is thus some resistance, but people in that domain were willing to accept rich storyboards – creating a model of their entire system. The semantics for the storyboards were invented by workshop participants: that way, they felt they owned the technique and could accept it.

Workshops have already been critiqued as not cost-effective. The CRIS creativity support tool encourages people to think creatively through the images that people have created and build stories around the requirements from Art-Scene and the images from the CombinFormation tool. This helps to capture the benefit from workshops into individual work afterwards.

Storyboarding is a simple, powerful requirements technique, fitting well with scenarios and use cases, but is rarely supported by development processes. It has great promise and should be used more. New technologies make it increasingly attractive both for researchers and for industry.

The day ended with a Question Time panel session of all the speakers.

**The Question Time Panel**

*© Ian Alexander 2008*

# RE-verberations

## Are PINs Less Secure than Signatures?

The February 2008 issue of *Computer* carries an alarming article from Vaclav Matyas, Jan Krhovjak and Marek Kumpost on an experiment concerning "*Authorizing Card Payments with PINs*". Chip-and-PIN technology has been trumpeted by the banks as a great leap forward in the fight against fraud.


Chip-and-PIN

The immediate consequence has been a drop in card fraud in those countries such as the UK and the Czech republic that use Chip-and-PIN. Card fraudsters seem to have switched their attention to countries which still rely on signatures, eg when UK cards are used outside Europe.

But will this (relatively) happy situation last? The researchers at the Brno University of Technology suspected that PIN numbers used in shops might be vulnerable to a simple attack: eavesdropping.

Worse, since banks put more faith in PINs than they ever did in signatures – "the PIN was correct so you must have authorized the transaction" – it has become harder for customers to defend themselves against illegal use.

Rather than simply asserting that this could be a risk, the researchers set up a two-part experiment.

In part 1, they borrowed the university bookstore and some student volunteers to see how easy it was to discover a PIN in a shop environment. The store contained 4 eavesdroppers, 3 experiment supervisors, some hired bystanders, and two store assistants as well

as the shoppers. The shoppers were told they were subjects in a slightly different experiment – that it was a survey of the relative comfort and convenience of the two means of payment.

The results were startling.

Given a PIN pad with robust visual shielding around three sides, the thieves guessed the PIN in 6 instances out of 17 payments: a 35.5% success rate.

For the PIN pad without any shielding (still a very common device in shops), the eavesdroppers correctly guessed the PIN in 12 of 15 purchases (80%). 10 of the PINs were guessed directly by individual observers; the other 2 were assembled from the shared knowledge (one observer was confident of the first 2 digits, etc).

For the signature-based transactions, the shoppers were given half-an-hour to practice forging a signature before going shopping for books. The merchants discovered 12 of the 17 cheats: 8 at the first attempt, and 4 more after challenging the shoppers for a second signature. None of the control group of honest shoppers was challenged to sign a second time. Both the participants and the experiment supervisors felt that the signature verifier did a good thorough job – it was a person who had worked in a jeweller's where checks were more thorough than would be usual in a university bookshop.

In part 2, the experiment was run using real payment cards in a large supermarket, with the assistance of the banks and suitable legal protection. Of the three groups of eavesdroppers, PINs were guessed correctly in 25%, 27%, and 68% of cases. The third group included one boldly assertive eavesdropper who managed to observe shoppers from close range.

RQ suggests that your motto should be "caveat emptor". The new technology is helpful to the banks, but apparently rather less so to shoppers. Shops are much less secure places than banks, and shoppers are far more tolerant of strangers coming up behind them in shops. Perhaps our perception of the risk of losing a PIN needs adjusting.

### Systems Engineering in the News

The 28 March 2008 Newsnight programme with Gavin Esler discussed the embarrassing launch of Terminal 5 at London's Heathrow Airport with experts Heinz Wolff and Allen Fairbairn. It certainly isn't every day that you see a systems engineer on the news.

**Discussing How NOT to Open an Airport Terminal**

The journalist noted that T5 is hardly the first big system that has gone wrong, nor even the first airport terminal to have terrible baggage-handling problems. Denver Airport back in 1994 was the classic example.

Allen observed that large systems risk becoming unmanageable through the complexity of interactions. These can be of any type; the Airbus A380 "superjumbo" was delayed for two years by the sheer difficulty of fitting 330 miles of cabling into the airframe – it just took up more room than anticipated. Perhaps the unglamorous nature of this problem is typical – it isn't enough just to focus on the exciting technical challenges: we have somehow to achieve the dilbertesque goal of "focussing on everything".


**A Systems Engineer on TV – you saw it here first**

"Do we delude ourselves?", wondered Esler. Surely yes: Hubris is followed by Nemesis. "One minute there is pride, backslapping and praise for the new project; the next there is chaos".

The programme very sensibly avoided speculating about the causes of the T5 debacle. Wolff wondered whether gradual commissioning would not have worked well, but he guessed it would have been very expensive (though perhaps less so than what actually happened).

## RE-readings

### Adrenaline Junkies and Template Zombies

Tom DeMarco, Peter Hruschka, Suzanne Robertson, Tim Lister, Steve McMenamin, James Robertson

Dorset House, 2008

The members of the Atlantic Systems Guild have published numerous books over the years, but this is the first time they have written one all together. It is hard to write a multi-author book, and most attempts at it don't really work. *Adrenaline Junkies* does work, and it speaks with a single voice. Whether that was

achieved by one of the group's editing like crazy, through the group's spending a weekend together, or simply through having worked together for years, is not stated. However it was done, the result is an attractive, funny and easy-to-read small book of essays. In fact, it is a compelling read; you sit and read one essay, muse on it, and promptly read another.

This is a book of patterns, that in the way of such things, you recognise but have never heard spoken. The individual essays are quirkily titled, and each illustrated by a photograph, drawing or graphic. These work well, despite their diversity of graphic style and relationship to their themes. The essays, too, vary widely in length: the authors are bold enough to say little where little is needed: another rare gift in today's prolix world of blogs and citizen-publishers.

The only book that is remotely comparable to *Adrenaline Junkies* is Michael Jackson's *Software Requirements and Specifications: a Lexicon of Practice, Principles, and Prejudices*. That, obviously, had a single famous and highly experienced author, who had himself created at least 3 software development methods over the years. Jackson's book - also a delight - is similarly a collection of essays, but it is arranged as a Lexicon (A to Z, minus a few letters here and there). Being a set of essays by one author, it is delightfully or maddeningly one-sided, according to taste; and it reveals Jackson's skill as an essayist - a rare thing in the world of engineering. Sensing, perhaps, that the alphabetic organisation feels quite weak, Jackson suggests no fewer than 14 short tours - paths through the book - on various overlapping themes. He later wrote up one of these themes - Problem Frames. Perhaps another 13 unwritten books lurk just under the surface of the Lexicon.

*Adrenaline Junkies*, like Jackson's *Lexicon*, provides much insight, with flashes of barbed humour. Its authors, too, have created various software methods and templates over the years. It too is in no particular order, save that the authors have striven for "*the most enjoyable reading experience*". There aren't any suggested tours, because none stand out: you have to find your own groupings or favourites. The "interlude" of *Project-Speak* is simply a delight, but the deeper pleasure comes from recognising the wicked portraits of some especially clueless roles on projects. Who hasn't met the project manager in *Management by Mood Ring* who always talks in optimistic, eternal-present tones with nary a mention of progress towards targets or deadlines? And what about the *Film Critics* who perpetually lob tomatoes into a project, with no feeling of responsibility to help make it work any better? Or *Children of Lake Wobegon*, where everybody's performance rating is above average?! Alistair Cockburn, quoted inside the front cover, is right on the mark when he says "*I suspect you will start using these phrases in your work - I already have*." But I suspect that the patterns in this book, as in Jackson's, in fact DO fit together: that Mood Ring management + Film Critic staff + Lake Wobegon mis-mentoring = Timid Organisation Planning For Failure, or something. In other words, while each essay tells a small truth, there may be larger syndromes at work here - a few more books waiting to hatch.

It's a brave book, too, that dares to speak about negative patterns that consultants see in organisations. "*The beatings will continue until morale improves*" is the self-confessedly "sour note" in *Happy Clappy Meetings*: be happy, or else. Publishing that particular pattern is a declaration that no corner of management doublespeak is safe; this book speaks truth to power (as the Quakers long ago said they would).

Other patterns that you will recognise with a certain grim fascination include *Short Pencil*: "*I hate working for a company that makes you turn in a short pencil before you can get a long one.*" I actually recall the flip side of this awful pattern: our project team from a software house found it could raise morale in our client organisation by giving out handfuls of biros to people who in all other respects looked like workers in a first-world country; and this was England.

So strong is the impact of the negative, that it is the essays describing positive patterns which stand out as different. My first impression was that there were very few of these, so it is with surprise that I see on looking back through the book that there are plenty of them. *Poker Night* praises doing something other than work (anything, not just poker) together as a team. Food++ tells the same story, really, though food certainly carries a special message of togetherness. *I Don't Know* explicitly praises honesty (and *Telling The Truth Slowly*, the anti-pattern, makes the reader wince: lying is no good, but telling the truth won't make you popular).

With so many patterns (there are 86 altogether), it's inevitable that not everyone will agree with everything, and this is certainly not a book afraid of controversy. Is the *Phillips Head* screw really so much better than the slot-head screw? Perhaps it's a matter of opinion: the Phillips head makes power driving a practical option; but it's really easy to grind the screw-head smooth with a power driver; and, looking at the bigger picture, the shockingly badly crafted furniture that passes for fitted kitchens, bedrooms and studies nowadays owes a lot to the replacement of craftsman-made dovetail joints with cheap angle brackets and Phillips screws. Perhaps the innovations favoured some stakeholders but not others: more profit for the big retailers and their shareholders; lower prices for consumers, perhaps; deskilling and unemployment for the old craftsmen.

The book as a whole is confident and convincing. Few other groups in the world today could have assembled such a wealth of expertise in project management, and none, perhaps, could have written about it so engagingly. I think you'll enjoy reading it. If you're in a position of power, I hope you'll take it to heart.

## Guide to Requirements SL-07 Template with Examples

Soren Lauesen

Lauesen Publishing, 2008

Lauesen has already shown himself to be a capable teacher of both requirements and user interface design. With this new booklet, he reveals his intensely practical side. The *Guide, Template and Examples* is a slim volume that follows the ancient teacher's maxim: Show, Don't Tell. The Guide is essentially just a short description of what each part of the Template is about. The Examples are from a real project and cover functional as well as non-functional requirements.

The odd title, by the way, is simply "Soren Lauesen, version 2007" - new versions will be issued and published as necessary, incorporating reader feedback and experience. That desirable process could only be accomplished by Lauesen's publishing the book himself.

There is no flowery rhetoric here; little theory; and no academic bibliography at all: pretty remarkable for a university professor. Instead, you get a compact, practical guide to what to put in your requirements specification, at least, if your project is for software - most probably of the data-handling variety - and you share Lauesen's concern to write down clear requirements to communicate a large customer's wishes directly to a supplier (a software house). In other words, this is a book specially for work of the copy, study, and edit kind, for what used to be called "user requirements".

Actually, it is rather more than that. Each chunk of the Template has a column for the (customer's) requirements on the left, and a column for the "Proposed solutions" to its right. The customer is, in other words, invited to write down what he expects or suggests in the way of a possibly-workable solution.

This is rather a wise move on Lauesen's part, because customers frequently come up with solutions rather than requirements.

The template coaxes them into moving such things into the "proposed" category, leaving the requirements field invitingly blank. Of course, that invites reflection: what must our requirement be, if that is what we think our solution will be like? With any luck, it will be a clear statement of stakeholders' goals, which will help the supplier mightily in understanding what is in fact wanted.

Lauesen's examples are nearly all from a healthcare project. Together they add up to a complete specification, except that only some of the use cases, some of the data classes, and some of the integration requirements are described. They quietly convey the message "this is how I did the requirements on this project". People have been pleading for such a book for years.

Some cautions are in order. The template does not attempt to cover "system" and "subsystem" requirements that might be written by the supplier and perhaps his subcontractors, once the architecture and scope are known. No attempt is made, either, to cover domains other than software, so there is scope for more templates and matching guidance if you know a good way of writing requirements for other kinds of product.

Finally, this is not a book of processes and techniques: it does not teach you how to identify your stakeholders, model people's goals, narrate their scenarios, or much else. But what it does do, it does supremely well, and it has its niche to itself.

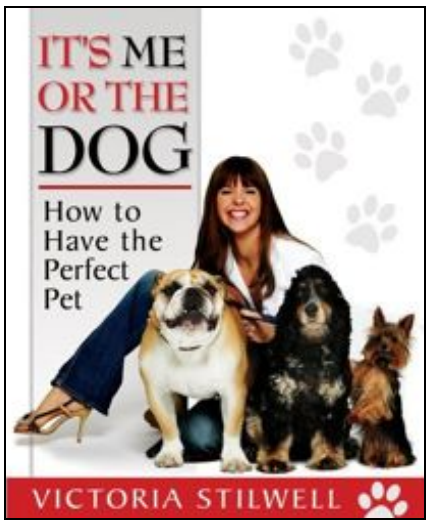
This is a really good and valuable contribution to the industrial literature on requirements. In a sense, it is a coming-of-age: requirements work is growing up, and Lauesen has given the world a serious, quiet, practical guide to help people get better results from their projects. The template itself can be freely downloaded from Lauesen's website, http://www.itu.dk/~slauesen.

| Availability requirements: | Example solutions: | Code: |
|---|---|---|
| 1. The availability must be calculated periodically. The calculation should compensate for the number of users experiencing breakdowns. | *The availability is stated monthly and calculated as described above.* | |
| 2. *In the period from 8:00 to 18:00 on weekdays, the system must have high availability.* | *In these periods the total availability is at least ___%. (The customer expects 99%)* | |
| 3. *In other periods the availability may be lower.* | *In these periods the total availability is at least ___%. (The customer expects 95%)* | |

**Lauesen's Template provides an inviting space to capture both requirements and solutions**

# RE-partee

## The Dog shall be Compatible with Owner




**It's Me or the Dog, by Victoria Stilwell**

Which dog does she look like, then?

## Updated Definitions File

**Stockholder** – person who has an interest in a company

**User Case** – user who always writes in with a long list of software bugs

**Object** – to complain at a software product review

**Class** – elegance and stylishness in a software product

**Goal** – a score! A spectacular sales success

**Quality** – you can't measure it, but you know it when you see it

**Metric** – measuring things in kilos rather than pounds

**Performance** – remarkably good speech by the CEO at the AGM

**Usability** – condition of an electronic product before the boys have played with it

**Evolutionary Development** – improvement in dress sense of male team member now he has a steady girlfriend

**Misuse Case** - Сценарии некорректного использования (Russian) (not kidding, honest)

## Updated Lamp-post Joke

When RQ was young, there was a joke that ran:

> *"What do you say when you walk into a lamp-post?"*
>
> *"Excuse me!"*

This was presumably funny because

a) the British used to apologise about everything

b) you have to be quite short-sighted not to notice a 20-foot tall post, and short-sightedness was popularly supposed to have been caused by something that it wasn't British to mention.

But times change. With so many walkers, cyclists and motorists now focusing on tiny screens instead of looking where they are going, the problem seems to be lack of attention to the peripheral visual field: in other words, trying to text and walk at the same time.

It has become an epidemic. It is alleged that 6.6 million accidents of the invisible lamp-post variety happened in Britain last year, or about 1 for every ten people.

It used to be said of US President Gerald Ford that he couldn't chew gum and walk at the same time. This was grossly unfair because he could do it perfectly well. It was just that he couldn't do the two things out of time, like trying to brush your teeth and pat your head simultaneously but at different speeds.

It was said, conversely, of the great drummer Ginger Baker (he of the incredible 13-minute drum solo in *Toad* on the Cream album *Wheels of Fire*) that he could do 19 beats with one drumstick to 20 beats with the other, thus achieving a secondary "beats" effect when the two rhythms added on every 19/20 beats (or interfered constructively, if you're a physicist).

**Padded Lamp-post, for bumping into**

It seems that the Great British Public have abilities that are more Fordesque than Bakeroid.

Various technical solutions have been proposed, including:

- padded lamp-posts (see photograph),
- ultrasonic or radar lamp-post detectors on mobile phones,
- talking lamp-posts,
- head-up displays to let people look up while texting, and best of all
- a vibration detector that disables texting when the phone is jiggled about as if the owner is walking.

Switching the thing to "off" (see Glossary for explanation) seems not to be an option.

# RE-sources

## Books, Papers

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

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

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

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

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

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

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

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

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

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

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

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

## Media Electronica

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

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

**ReQuirements Networking Group**
www.requirementsnetwork.com

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

## RE-actors: the committee of the RESG

**Patron**:

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

**Chair**:

Dr Pete Sawyer,
Lancaster University,
sawyer @ comp.lancs.ac.uk

**Vice-chair**:

Dr Kathy Maitland,
University of Central England,
Kathleen.Maitland @ uce.ac.uk

**Treasurer**:

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

**Secretary:**

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

**Membership secretary**:

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

**Publicity officer:**

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

**Newsletter editor**:

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

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

**Student liaison:**

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

**Immediate past chair:**

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

**Industrial liaison:**

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

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

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

**Members without portfolio:**

Prof. Neil Maiden, Centre for HCI Design, City University,
N.A.M.Maiden @ city.ac.uk

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

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

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

### Contributing to RQ

To contribute to RQ please send contributions to Ian Alexander (iany @ scenarioplus.org .uk). Submissions must be in electronic form, preferably as plain ASCII text or rtf. Deadline for next issue: 7th September 2008

### Joining the RESG

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