Requirenautics Quarterly

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

http://www.resg.org.uk

 Issue 21 (July 2000)

RE-Locations

RE-Soundings

Editorial

Welcome to RQ21. This issue neatly encapsulates the hottest current issues in RE.

Compared to the software and systems engineering communities, the RE research community's continuing fascination with notations sometimes looks a bit anachronistic. However, recent years have seen recognition that effective RE is about more than modeling, and is more than a bolt-on activity to the front of the software process. A consensus has emerged that RE is a continuous process - but a process that is distinct from, and operates in parallel with, the software process. This view of the RE process broadly conforms to the systems and software engineering orthodoxy. Of course, orthodoxy is always open to challenge and this particular orthodoxy has been challenged by attempts to respond to demands for ever-shorter development schedules.

All three of these viewpoints on RE - the notion-centric, the orthodox process-oriented, and the iconoclastic challenger - have been represented by recent RESG events, reviews of which appear in this issue.

Jim Arlow's compressed tutorial on *Using UML in Anger* represents the notation-centric. Actually, this is unfair to Jim. His tutorial is really about the effective deployment of the UML notations in an RE process, rather than promoting UML as a panacea.

Suzanne Robertson treated us to her seminar on *Mastering the Requirements Process*. This seminar, her (and husband James') Volere process and their excellent book (reviewed in RQ17) are among the best starting points for organisations who want to bring their handling of requirements under control. This work is imaginative, pragmatic and non-doctrinal. It represents the state-of-the-art of orthodox RE.

Finally, the iconoclast is represented by extreme programming (XP) which was the subject of a debate hosted jointly by the RESG and the advanced programming SG.

There's a spread of philosophies in this issue, then. But read Ian Alexander's reviews of these three events and you might be surprised as much by their commonalities as by their differences.

In addition to the event reports, Geoff Mullery's *CORE-Blimey* column looks at the use and abuse of standards in RE - again, challenging orthodox views on this topic. Finally, the tireless Ian Alexander contributes a paper that explores parallels between venture capital investment and system procurement.

Enjoy what's left of the Summer.

Pete Sawyer,
Computing Department, Lancaster University

Chairman's Message

The RESG organised two fun meetings this quarter. Our "moderate" debate on Extreme Programming in March generated lots of interesting discussion, both at the meeting itself and in the pub afterwards! Jim Arlow's UML mini-tutorial in May also proved to be very popular with a predominantly practitioner audience, some of whom resisted leaving the meeting room long after Imperial College closed its doors for the evening!

We didn't organise anything this June, but guess what, somebody else did! The 5th IEEE International Conference on Requirements Engineering (ICRE-2000) was held near Chicago, USA, between 19-23rd June 2000. The conference is the premier technology transfer conference in the field, and I was happy to see many RESG members attending.

The next RESG meeting will be on the subject of Scenarios on 12th July 2000. It will be preceded by the RESG's (short!) Annual General Meeting (AGM). I really would like to encourage as many RESG members as possible to attend this meeting. It is a chance to meet the RESG committee, to hear the reports of the different officers on the various activities of the group, and perhaps consider joining yourself? We are always looking for new, motivated, volunteers to serve on the RESG committee. We are an informal lot, who work hard and play hard ... Please do drop me a line if you would if you would like to chat about ways in which you could help the group.

All for now. Back to work. Would anyone like to help me develop an automated exam marking system? I have some very precise goals, measurable requirements, and a real need! Help!

Bashar Nuseibeh,
Imperial College, London

Farewell Sara

After six years of sterling service to the RESG, as Membership Secretary and active committee member, Sara Jones is stepping down from the RESG Committee. On behalf of the RESG membership, the RESG Executive Committee would like to thank Sara for her outstanding contribution to the group, and to wish her all the best for the future.

RE-Treats

Next event organised by the group.

Scenarios in Requirements Elicitation and Specification

Date: 12th July 2000
Location: London
Contact: Ian Alexander, Independent Consultant (iany@easynet.co.uk) and David Shearer, University of Hertfordshire (d.w.shearer@herts.ac.uk)

Requirements for 1 Billion Users - RE for websites and product lines (to be confirmed)

Date: 29th November 2000
Location: London
Contact: David Shearer, University of Hertfordshire (d.w.shearer@herts.ac.uk) and Barbara Farbey, UCL (b.farbey@cs.ucl.ac.uk)

RE-Calls

Recent Calls for Papers and Participation

REP 2000 - Second International Workshop on the Requirements Engineering Process, Greenwich, United Kingdom, September 6-8, 2000

Scope:
Whereas conventional RE approaches focus on models and languages to express system specifications, there has been a recent shift towards Requirements Engineering Processes (REPs). The prevalent view is that requirements emerge from a process of learning in which they are elicited, prioritised, negotiated, evaluated and documented. Requirements evolve with time. This necessitates managing requirement evolution and aligning requirements to organisational changes.

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

http://panoramix.univ-paris1.fr/CRINFO/REP/

Important Dates:
Paper Submission March 15, 2000
Notification of acceptance April 10, 2000
Camera-Ready copy due May 1, 2000

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

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

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

Submissions:
Paper submissions should not exceed 5000 words. The first page of the submission should include the title, all authors' names, complete information (address, phone/fax, Email). Electronic copy of the paper (in PDF, Postscript or MS Word format only) should be sent to:

Dr Didar Zowghi
Faculty of Information Technology
University of Technology, Sydney
P O Box 123
Broadway, NSW 2007, Australia
Phone: +61 2 9514 1860
Fax: +61 2 9514 1807
Email: didar@it.uts.edu.au

Important Dates:
Paper submissions August 7, 2000
Notification of acceptance October 10, 2000
Final version for publication October 30, 2000

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

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

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

http://www.re01.org/

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

RE-Readings

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

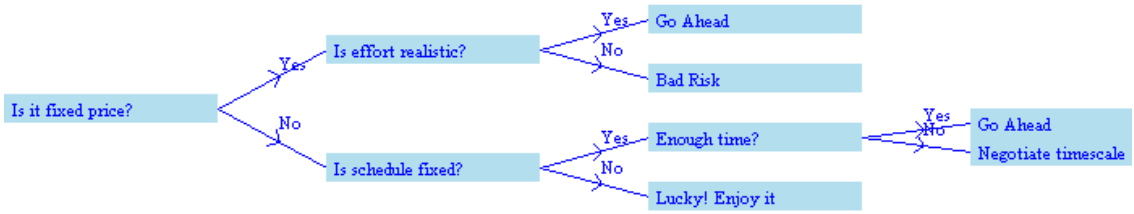
Extreme Programming

A moderated debate organised as a joint event by the RESG and the Advanced Programming SG at Imperial College, London on 9th March 2000.

The chairman, **Bashar Nuseibeh**, welcomed an audience of some 50 people to Imperial College, explaining that the format for the evening would be a neutral talk on the principles of Extreme Programming (XP) by Sue Eisenbach, followed by short speeches moderately for and against XP by Paul Dyson and Ian Alexander. The speakers would then answer questions as a panel, before adjourning to a nearby hostelry for the APSG's very civilized approach to lubricating the wheels of intellectual debate.

Sue Eisenbach (Imperial College) listed some websites that explain what XP is and argue its case:

c2.com/cgi/wiki/ExtremePlanning (and related subjects)
www.Xprogramming.com



www.armaties.com/extreme.htm
www.kinetica.com/ootips/xp.html

Wiki, incidentally, is a wonderful site, both for its mechanism which provides an editable hypertext, and for the quality of its debate on many topics.

Sue Eisenbach then explained what XP offers in the context of what the customer can be presumed to want, along with what developers want.

The customer wants to set objectives and to have them followed; to know how long a job will take and how long it will cost; to set the scope, stating which features are in or out of the software; to know the project's status; to be allowed to make reasonable changes to the requirements, and to know what those will cost; to be told of risks to cost, schedule, and quality; and to see progress via a steady flow of deliverables.

On the other hand, developers want to know the objectives that are set for them, with priorities; what to build; to be able to meet the customers and users easily; to have a say in setting realistic schedules; to let the customer know what is really happening; and not least, not to be constantly interrupted.

Could XP meet this challenge? It offers "soft" software, a way of handling requirements change, the promise of quickly producing something useful and then continually enhancing it, and the claim that average programmers can produce great software.

To achieve this, XP demands dialogue in groups, a shared programming language, and contact with users. Its approach is based on staged delivery – not necessarily extreme, perhaps; separation of business 'what' from technical 'how' – very classical; plenty of testing, and plenty of reviews – to which every one pays lip-service.

So what is so extreme about XP? The life-cycle is certainly a surprise. Analysis is reduced to collecting "user stories" interactively; these are worked up into scenarios (acceptance test scripts) that can be tested directly. The programmers are required to prepare end-to-end tests before any "classes" (generic code objects) are written. The programmers can then construct unit tests for each class they intend to write, and only then do they start programming – in pairs, continuously reviewing each other's code. The resulting design is examined for awkwardnesses and inflexibilities, and these are removed by a redesign process called refactoring which is intended to produce clean well-structured code. The project then takes a breath to obtain user feedback and reschedule itself for the next iteration. XP thus involves a range of extreme practices such as extreme scheduling (user stories – cost estimates – customer decides next step – iterate…) and extreme testing.

Extreme and Unusual practices include pair programming and refactoring, both of which strike conventional managers as expensive and risky.

Extreme but Usual practices include relying on user stories, schedule negotiation, staged delivery, early testing, and stress on simplicity.

Paul Dyson (Steel Tiger) presented the case for XP in the most disarmingly moderate way possible – by describing his own experience with the technique.

The patterns community including Ward Cunningham and Kent Beck had over the past 5 years or so discussed XP, especially on the Wiki Wiki Web (which apparently means 'Express' in Hawaiian). Paul said he had been 'very excited' by the ideas, but had never used it until he decided to try it on a project.

He began by holding a Planning Game; then gathered User Stories from the customer to identify Engineering Tasks and enable his development team to estimate the project. He then launched into Pair Programming, Unit Testing and Continuous Integration – a build every three hours.

It should be explained that Unit and Acceptance Testing are not really different in XP, because each User Story results in one or a very few test scripts (it depends on whether you allow use cases to branch because of exceptions and suchlike things – if so there will be several scripts per story, but this risks a theological debate…*Ed.*).

"My job was on the line" said Paul, with a big review meeting with the customer coming up. But the programmers spoke up in his defence: "I like this continuous interchange" and "We've got to have testing". That was 18 months ago. Paul then started up in E-commerce with a few friends (the audience hummed to itself, speculating about Paul's imminent millionairedom). The project had been estimated at 11 man-years, but was completed by 6 people in 14 months. The customer was delighted and the job was finished ahead of schedule.

Paul's second story was from his new company, Steel Tiger; some people down at the pub learnt how it got its name. The first project went from idea to working website in 3 months, even though the client didn't know his own requirements. Steel Tiger built "the roughest dirtiest search engine" in 2 days, and took it straight back to the customer, who revealed that he had only wanted a few simple answers anyway. Just as all seemed to be going well there was a corporate takeover and all the requirements changed. Fortunately, the effect was a new set of views on the same simple core solution, and the project succeeded in just 3 months.

"All I know is that it worked for me on these two projects" said Paul. It might not be for everyone or all projects: it was a people thing. Some people preferred to fail safely, their tails securely covered against blame, than to risk taking responsibility and possibly succeeding.

Ian Alexander (Independent Consultant) argued the case against XP by saying that 'XP may not ALWAYS be a good idea'. He quoted a Guinness advertisement 'I don't like it because I haven't tried it' and therefore decided to have a go at "doing the simplest thing that could possibly work" – an XP slogan. The goal was to make a tool to draw Decision Trees – the test, a correct diagram as output – so he designed a tool that simply halved the Y-axis for each new level of decision.

The resulting diagram shows that the code passes its test – sort of. After a few levels the decision boxes will overlap as the Y-axis space is subdivided – even though there may be plentiful space above and below the boxes. Ian presented two further solutions, one relying on spacing the boxes as widely as possible in each column, and one using a fixed spacing: both 'correct' in some sense but both requiring redesign of the code. He then showed a subtle solution by customising a COTS tool, presenting an unfamiliar style of diagram but needing no coding. A tree drawn using that algorithm would be complex but visually harmonious. Was that the requirement?

Rework could certainly lead to better algorithms. 'Experience is a hard teacher, but fools will learn from no other'. Any of the algorithms could have been arrived at by thinking through and prototyping the requirements without coding, in discussion with the users. Yet full dialogue was and remained rare, like good requirements.

XP had obvious merits, he said. It recognised the place of the stakeholders, insisted on dialogue, thought carefully about the code, demanded thorough testing, avoided gold-plating, and provided rapid iteration to let users get what they wanted.

Against this, it tended to skimp on documentation, risked coding first and specifying afterwards, and was perhaps inspirational rather than systematic – like some political parties.

The real problem with XP, though, lay with larger systems, and with safety. Could XP possibly work on projects with hundreds of engineers at different sites, with complex contractual boundaries, managers, lawyers, procurement officers, over periods of many years?

On mission-critical, embedded, and safety-related systems, safety and dependability were more important than anything else. Precise specification in advance of design and subcontracting was essential.

That did not mean that businesses could not be inspirational at an appropriate scale. The courier service UPS was reputed to insist on total compliance with its system on national and international work – but to give large freedom to achieve the fastest delivery locally. If bicycles were quickest in traffic-clogged streets, then that was fine with the company.

XP, he argued, was ideal for jobs involving perhaps 2-10 programmers, 1-5 stakeholders, a few months' work, and not least a situation where 'if it works, it's ok' and any tweaks would just improve the code. Few people would wish to fly in airliners with XP engine control software, or would trust their savings to a bank with XP automatic payments systems.

At the other end of the scale, the occasional day's consultancy involving customising an export utility from a database would cost twice as much to provide with XP, unacceptable even if the result was of better quality.

XP certainly had a place, and ought to be cost-effective and satisfactory for many small-to-medium jobs. XP did not reduce the basic need to agree and document system requirements. User Stories were examples, from which people could induce any number of requirements. Some of those would be right.

The meeting adjourned for a well-earned drink.

Mastering the Requirements Process

A 3-day course given by Suzanne Robertson and co-sponsored by the RESG, held at the Harrington Hall Hotel, Kensington during 27-29 March 2000

The course began with Suzanne Robertson's trademark, a classical oil painting. In this case it was by the Dutch old master Pieter Brueghel, full of colour and detail, rich in vivid constrasts, and based of course on real life sketches ... just as successful systems are based on observation, knowledge of real stakeholders and sketched requirements, later sharpened up.

The course participants included people from oil companies, software, van hire, defence, banking, the British Library, a charity, payroll handling and security. Or, like requirements classified another way entirely, we were from the UK, Norway, the USA, Kuwait, Holland, Belgium, Austria and Eire.

Issues raised by the participants were time-to-market, lack of requirements, acceptance criteria, changing processes, the cultural divide, death by survey – a surfeit of requirements analysis, reorganisation, ownership, repeatability, integration, speaking the users' language, and excessive focus on solutions. Suzanne Robertson was naturally delighted by this diversity, and it quickly became apparent that far from being VOLERE-process bound, she was both splendid at dealing with people and quite pragmatic about applying methods to problems.

The key themes of the course are well described in the Robertsons' book (reviewed in RQ 17) and include a focus on stakeholders, trawling for requirements, prototyping, the quality gateway and not least the fit (or acceptance) criterion.

If there was a real surprise it was the emphasis on precision, not only through careful detail as via the CRC-like 'snow cards' but also with use of quite formal process and data descriptions. These could be through traditional DFDs and ERDs, or their UML equivalents, which as Suzanne said are becoming the industry standard way of describing problems graphically.

Precision is hard to attain with natural language – the Cambridge philosopher Mary Hesse famously came up with over 60 uses of the word 'natural', for instance. The best we can hope for is to create sharp Michael Jackson-style designations of the terms we intend to use, and then apply these strictly in our specifications. This approach gives rise to a general approach to metrics: for every set of rules, we can apply a set of checks. For instance, we can check our use of terms against our naming conventions, or our specification contents against our requirements process model.

Suzanne was rightly cautious about recommending specific tools but gave us a useful list of tools and websites with concise details of each offering.

The participants all seemed happy with the value they received and there was certainly plenty of meat in the presentation, richly garnished with notes, exercises and references.

We had fun picking holes in the London Underground's ticketing system. Why did the designers insist strictly on type of ticket, then destination, then cash, and why aren't the station buttons grouped, without even visible capital letters? After working out some scenarios your reporter role-played the part of a ticket machine along with another participant playing Manuelo (seemingly of Fawlty Towers fame) trying to buy a ticket via a Spanish speech interface – direct to KHarrington KHall Khotel, which is not the same as Harrow and Wealdstone station – and then attempting to pay in Pesetas. Suzanne took it in good humour.

The seminar was very efficiently organized by IRM. The hotel staff were helpful, the food was excellent, and everything worked and was on time. As in all the best-specified projects.

Using UML in Anger for Requirements Engineering

The RESG was very pleased to welcome Dr. Jim Arlow of Zühlke GmBH on the 17th May 2000 to Imperial College, London for this mini-tutorial, in which he presented the key parts of his admirable 4-day course in an afternoon.
The tutorial started conventionally enough with a clear description of the reason for Requirements Engineering in the first place, with reference to the Standish reports and traditional 'shall' requirements.

Arlow then introduced use cases as the UML approach to requirements capture. The principles of use cases were explained, with plenty of discussion and questions from a practical and experienced audience.

He explained wonderfully clearly what the symbols meant – often they had a very limited meaning, not at all what one might have expected. UML uses the stick-person icon to mean what it calls 'Actor', but confusingly this does not mean a human at all, but a role that can be played by a human or a machine agent. The terms Role and Agent are in wide use in Requirements Engineering, so their use might have been preferable – but *fait accompli* is everything in 'industry standards'. Like Ian Graham, I think all the arrows in UML point the wrong way, resulting in oddities like (subclass) specialization being described bottom-up as 'generalization'. This is indeed the logical inverse, but one can't help wondering whether this was not mainly a deliberate break with the past, rather than anything particularly coherent.

Some of the inevitable tensions in the use case approach came out when Arlow defined the meanings of the <<include>> and <<extend>> relationships. Given the propaganda that a use case is a simple sequence of events, the pass is sold by the admission that extensions – unlike inclusions – can be conditional. (Pre-) Conditions can be written in – using semi-formal expressions such as [! first offence], making it easy to separate out the cases without writing complex requirement text. Arlow also allows steps in a use case to begin with 'If', and groups of steps to be prefixed with 'While' and 'For'. Indeed, steps can be parallels – for instance in a textual 'While' loop, there may be two activities that both run continuously, such as a display and some background music. However this is not supported by any formal syntax or semantics. Similarly, extensions often represent exceptions or error conditions, but there is no standard way of indicating when this is in fact what they are being used for. When use cases get too complicated, Arlow recommends describing them as a set of simple (unbranched) scenarios – like Hollywood scripts. A single Primary Scenario defines the 'happy time' case; the Secondary Scenarios – that I would call exception cases – branch off from it.

Use cases, in fact, can be simple or complex, and can be used to express almost anything – though other UML constructs such as activity diagrams, state charts, and swimlane (sequence) diagrams may often be more expressive. In particular, it is tempting but wrong to imagine that a visual sequence of use case bubbles on a diagram represents a temporal sequence, a high-level use case. Alistair Cockburn and I have argued that there can be good reason to nest use cases, but Arlow rightly points out that the danger here is of a backsliding into top-down functional decomposition, with all the problems that that brings.

Unlike some UML partisans, Arlow recognizes quite clearly that use cases do not do everything. Firstly, they do not help in describing constraints or non-functional requirements; even the constraint language merely goes into 'notes' which clutter the diagrams, and can be replaced by free natural-language text – not much improvement there. Secondly, they do not offer much support for embedded systems or complex algorithmic models – such as weather prediction or chemical reaction simulation – where there are at best very few use cases, and most of the complexity lies elsewhere.

There was some discussion from the floor of the difference between business and system use cases; Arlow seemed to feel that a system focus was appropriate, while some of the participants had found it useful to create a high-level set of business use cases to describe the problem, and then a lower-level set of system cases to start to define the solution. This seems an entirely sensible and desirable approach, reflecting Jackson's World/Machine distinction.

Participants were able to have a go at creating a use case model of the e-commerce system using Rational Rose. The elegant seminar room with built-in networked computers and flat screens had been set up with Rose available at every desk. The tool started in a welter of error messages, but – ignoring these – we were able to draw use case diagrams. It seemed strange that the tool still does not support the drawing of a simple box to indicate the system boundary, but allowed a range of more complex notations such as stereotypes for relationships. The participants must have been enjoying themselves, for long after the chairman broke up the meeting and thanked Dr. Arlow for his excellent tutorial, people were still huddled round their screens developing their models!

RE-Papers

Engineering E-business: where business innovation meets systems engineering

Ian Alexander
Independent consultant

E-business has made technology highly fashionable. London's billboards and underground stations are covered wall-to-wall with advertisements for companies with attractively natural names like Breathe and Egg. The products on offer range from estate agency to banking, chic clothes to last minute holidays. Conventional companies are caught up in the excitement and are hurriedly developing their Internet strategies

Risk and Reward

But being fashionable opens you to the risk of becoming a fashion victim. The new dotcom companies are clearly high-risk ventures, as the recent demise of Boo.com shows. Operating only on the Web, they must identify a new market niche, establish a market image from scratch, attract clients and keep them, create effective business processes, and not least acquire whatever physical resources they need to fulfil the orders they receive. Getting a warehouse and its staff up to speed may be more difficult than constructing a website; persuading suppliers to ramp up deliveries quickly may be harder than making a set of Java applets reliable.

Adding E- to an existing business can definitely be easier. If like Simply you are selling personal computers and peripherals by phone and credit card, adding a Web 'channel' for the same goods is a low-risk step. The customers already exist, and are in Simply's case presumably rather familiar with the Web already. The order fulfilment mechanism – the warehouse and distribution system – is already in place. The additional requirements are mainly technical: for a reliable website, with a secure connection for payment details. Existing customers can be offered a modest discount to encourage them to log on rather than phoning a salesman. For example, Simply offers free delivery for online orders. Even if in the first year or two the website does not do enough business to cover its costs, the company can continue to flourish through its traditional channels.

The situation is less safe where an existing company branches out in a new direction with an E-business venture. Walt Disney corporation has recently shut down its ToySmart.com subsidiary; despite Disney's deep pockets, there was not enough commonality between its core business and the sale of toys to get the new company into profitability quickly enough.

The risk is even greater for startup dotcoms. Many, it seems, are not just not yet making a profit; they have no revenue stream to speak of, and no detailed business plan or capability offering a realistic hope of moving into profit. The coming year will certainly be painful for these companies as venture capital is withdrawn.

Early this year, a bunch of aggressive technology companies displaced several large and successful firms from the FTSE 100 index of top companies. This seemed premature to many, given their low or absent dividends which did not reflect their high capitalisation. The rules had been changed, we were told. Just a few months later, the bubble is deflating – if it has not actually burst – and the rules appear much the same as ever.

The public rightly suspects that there may be vested interests in talking up new offerings like the dotcoms, whether these are well-founded or not. The directors of a new company can enjoy the publicity, excitement, and trappings of success, while the financial institutions backing the launch get a commission as long as the Initial Public Offering (IPO) is successful – though possibly at some risk to their reputations. Similarly, the venture capitalist backing a dotcom up to IPO stands to make a handsome profit. If the new company fails to survive in the marketplace, the losers are the share-buying public. Arguably they are at risk of being exploited.

The Esouk Model

Fortunately, there is a better and more responsible model of dotcom launch. Newly-hatched companies are at their most vulnerable while they are small and inexperienced, especially if they lack well-defined business processes and a reliable infrastructure.

An alternative approach is for an organisation that combines technical and commercial experience to 'incubate' or 'accelerate' dotcoms, bringing them to market with better preparation and proper support through the critical early stages. Esouk, itself a dotcom, specialises in this remarkable activity. (Its name is Arabic for 'to bring to market'.)

The required technical experience consists of practical knowledge of hardware and software, of architecture, tools, suppliers, and not least of the system lifecycle, including both specification and verification.

Thomas Reitstetter, Technical Director of eSouk.com, and previously head of technology of AOL UK, says that you have to have the technical expertise to identify the criteria necessary and achieve successful and long-term fulfilment. According to Reitstetter,

"It is critical to have a qualified understanding of vendor capabilities to make an informed decision on technical requirements for the survival of your portfolio companies. Once we have accurately assessed the needs of each Internet startup, [the risk in] selecting a vendor who meets the criteria is significantly reduced, and laying the foundations can commence."

The needed commercial experience consists of a realistic understanding of the markets, competitor activity, legal council, financial expertise and involves close co-operation with venture capitalists.

The basic approach is to correctly identify the requirements necessary for success. In other words, the idea is to use capital responsibly in a rational framework of business management and systems engineering to manage risk from day one.

This is a model where systems engineering meets business management, to create highly effective technology solutions in a ruthlessly Darwinian world: Natural Selection is represented directly by consumer choice. If consumers buy it, it succeeds; if they don't, it goes the way of New Smoking Mixture and the Sinclair C-5 car. Every software project takes risks, especially if the customers and stakeholders who are expected to use or benefit from the software have not been thoroughly consulted or their needs identified. Risks are rather sharper in E-business, and lead to sudden wealth or business failure.

The Esouk process begins with an idea – or rather, with a surfeit of ideas. Any firm offering the slightest glimmer of hope to wannabe millionaires is bombarded with getrichquick ideas, many of them impractical. Esouk gets its fair share of these, but would rather hatch its own ideas where possible. Ideas are cheap; bringing them to fruition is another matter.

So the next step is the first of several filterings. A proposal generated either internally or externally is presented and discussed at regular investment meetings. At this stage many are weeded out or sent back for further preparation.

A business proposal that meets eSouk.com's initial investment criteria is then considered in detail to determine whether the business proposal is viable and meets the criteria for a sound investment opportunity. Further research is then necessary to prepare for presentation of the business plan to the investment panel.

This consists of two basic parts – a commercial case and a technical case and includes detailed requirements and a matching high-level architecture. If this seems surprising at this stage – shouldn't requirements precede design? – then think about the relationship between the tasks of defining what is wanted and defining how to meet that need. In the old waterfall model of the system lifecycle, each stage was supposed to be completed before the next began. Apart from taking a long time – a costly luxury in a fastpaced world – this created the strange idea that one could or should be able to define accurately what a system should do without any idea of its design. In the newer spiral or evolutionary models of the lifecycle, each iteration encompasses the whole system, from requirements through design to testing – first at a high or conceptual level, then in more detail. The waterfall model's separation of problem from solution is retained, but each time round the spiral, the concepts are sharpened and tested. The result is faster and more accurate system development.

What this means in Esouk's case is that the technical director sketches out a high-level system design and discusses this with the launch director to ensure it meets the needs of the new company. The system is 'tested' by walking through the principal scenarios (use cases) that it must support. The design is then improved or modified as necessary, along with the requirements.

An Example Scenario

For example, if an Amazon-like online bookshop was being specified, one scenario would be that the customer could come and browse through the catalogue, or search to see what their favourite author had written. The customer could then decide to buy one of the books, give credit card details and confirm the order. The system would then fulfil the order by instructing the warehouse to pack and despatch the book, and would then take payment. There is quite a close relationship between customer requirements, such as being able to browse and then buy online, system requirements such as the system's being able to instruct the warehouse what to despatch, and system design, such as the system having a communication mechanism with the warehouse. The scenario actually presupposes all of these things, so there is a chicken-and-egg situation between the different lifecycle documents.

Once the initial proposal is ready, it is presented to a review board which formally considers all aspects of the business case. Is the basic idea sound? Is the business model realistic? Would the system work as intended? If the board is convinced, the fledgling company is awarded substantial funding. Still under the control of the launch director and with the full technical support of the parent company, its systems are developed, tested and prepared for launch. This takes several months and is tightly timetabled. The company takes on staff, prepares its marketing campaign, gets its website built, acquires all the other resources it needs, and rehearses its business processes under its mother's watchful eye.

At the launch the new dotcom aims to make an impression on the market, get good press reviews, and deliver a reliable and good-quality service. It then hopes to move rapidly through acquiring a market share to making a profit, and at a suitable moment to move into premises of its own and run its own affairs.

Systems Engineering in Action

It is hard not to believe that all of this greatly increases the chances of success for a new dotcom, and could provide valuable support for corporations embarking on E-projects. The unique fusion of venture capital, financial and legal knowledge with human resource management and highly skilled systems engineering is a world away from the dodgy.com startup operating on a wing, a prayer, and someone else's money. E-businesses have been on the

receiving end of some rough market sentiment in recent months. eSouk.com's sensible and pragmatic approach ought not to be affected much by the current alarm in the world's stock markets, as people realise the risk they have been taking with some of the less prepared startups. Hopefully both wise investors and eSouk will continue to reap the benefits.

CORE-Blimey!

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

Keeping Down Your Standards

In my contribution on an ENS (Enabling Network Systems, newsletter issue 20) in connection with the key tools I mentioned the question of to what extent they should be standardised – and ventured the unsupported opinion that it should be as little as possible. Here I aim to explain why.

My problem with adoption of standards is not the principle, but the practice. Problems arise at all levels of standardisation – *international standards* (e.g. administered by ISO committees), *institutional standards* (e.g. administered by the US DOD or UK MOD), *organisational standards* (administered by individual companies) and the maverick of the standards approaches, *de facto standards* (adopted by common usage). In practice these all tend to lead to a standardisation situation which sometimes over-controls and/or under-controls things.

Some standards are based on theories rather than practice – one or more people develop a theory how things should be done and the theory is expressed as a standard – often without regard to the extent to which it can or should be applied to all projects. The thinking is that the originators are clever people, so they must be right – but anyone or any group attempting to define a standard does so from a position of limited experience. Only a god can know everything about all systems. For the rest of us there is guaranteed to be something missed or which does not work in *all* circumstances.

If you look at either the ISO or CMM approach to definition of quality systems you see that they recognise this problem. They both recognise the need for an evolving quality system based on informed (measured) judgement about what works and what does not work *in practice as well as in theory*.

In practice, taking the processes of international, institutional and organisational standards definition and use, once a standardisation decision is made it is very difficult and expensive to get it changed. At the level of international and institutional standards, the onwards expense of a change can sometimes be very large.

The history of the ISO C++ standards shows two examples. In the decision on how to test for success after using the 'new' operator they first adopted the old C approach of testing for a null pointer to see if the memory allocation failed. Later they replaced this with use of an exception mechanism. Also they introduced a pointer as a mechanism for handling the need to identify which virtual function to call at run time – but the pointer was an integral part of the data structure which defines the class.

Both of these changes were not backward compatible with previous implementations of C++, so they rendered many existing C++ programs non-standard – and more to the point, they meant that it was dangerous for such projects to upgrade to new versions of the C++ compiler because of the side-effects of these changes.

I have had legacy applications written in C++ which required significant and complex rework due to both these enhancements. I am not suggesting that enhancements were not needed for the language overall. My point is that an early decision on a standard approach to something which is not properly understood can be very damaging to legacy projects and can catch out unwary legacy programmers developing new applications.

There are ways to reduce this type of problem (it can't be wholly removed because we can't always tell we don't understand a problem). The CORBA approach seems to me to be better in such cases. Where a problem area is not fully understood, but it is known that standardisation will eventually be needed they explicitly highlight the area but leave its standards fulfilment open until a solution can be agreed. I would go further and suggest it should be agreed and tested in practice.

The de facto approach to standardisation is recommended by some people. It is often assumed to occur via a process of general acclaim. The idea is that a large majority of people choose the standard, so that must one which works. Unfortunately, though this may sometimes be true, but there are numerous cases where it is not.

The dominant position of some computer industry tools (for example some operating system and office product lines) is a case in point. Such products do appeal to many users but they do not entirely achieve dominance simply by technical excellence.

Many organisations decide that the benefits of bulk purchase, plus a "one size fits all" assumption make it a good idea to standardise on one source of each key product. These may be good decisions, but they do not always occur because the selected products are the best – sometimes it is because the product manufacturer is perceived as dominant. It is an adaptation of the phrase "Nobody ever got fired for

buying IBM". *That* is arguably what has led to de facto standards in operating systems and office products.

So de facto standards are not necessarily best – and even where they are they may still not efficiently achieve the level of quality needed in a specific project's case. Nor will they necessarily evolve to efficiently achieve that level. The supplier of the standard is guided by a complex mix of influences, only one of which is improved efficiency and quality across a range of user applications.

So if nothing works well enough, what can be done? Return to the ISO/CMM approach. What we need is an evolving process of gradual and measured improvement – taking into account the fact that the market does have an influence in determining what works and what doesn't, but selecting a de facto standard only if it can be shown to be sufficient and is placed in the public domain or at least opened to priorities other than those of the owner.

That is exactly the job currently being attempted by international and institutional standards bodies and the open software movement, but it is only partially succeeding. A key point to observe is that, wherever there is a standard agreed there are always developers who find a case for extending or bending the standard. This is not a reprehensible practice (though there have been occasions where it has been done for reprehensible purposes – such as to highjack the standard for a proprietary purpose).

It is a natural process which has occurred repeatedly in other industries and in the hardware side of the computer industry. There is often a period of competition between rival standards claimants or rival extensions to actual standards, followed by victory for one or a generally acceptable compromise (recall the video recording format wars or the 56K modem standards wars).

For the unwary this can lead to cost implications – but the important point here is that it is usually clear from an early stage that these are only rivals to become the new standard. That means that (in the world of software) either can be used with foreknowledge of the possible need for future change – and we are not that bad at designing or programming defensively when the need is clear.

If you add this idea to the CORBA approach of explicit identification of areas requiring standards, but with no current solution you reach the approach which I believe should be adopted. Standards bodies should, instead of rushing into a premature standard definition, explicitly identify the need for a standard and warn that its implementation has been left open.

Developers and researchers should then be free for a period to propose *and experiment with implementations of* standards in such areas. Practical assessment of such experiments together with negotiations between the developers should form part of the work of the standards forum in developing new formal releases of the enhanced actual standards.

This should occur, not at the earliest possible moment, but at the earliest moment that it can be demonstrated that there is a practically workable implementation of a standard for an area previously left open. On top of that it must still be possible for people to add in experimental releases features which were not previously foreseen as necessary.

I don't know the history of the C++ Standard Template Library, but from what I have read the STL was introduced and applied first via non-standard (proprietary) implementations which were shown in practice to have value and later included in the ISO C++ standards definition. That is an example of the above approach to standards evolution. Another, perhaps more contentious example would be the language extensions used in Borland/Inprise's C++ Builder to handle what they call components.

Turning back to an ENS let me put the above comments on standardisation into that context. There are numerous researchers experimenting with techniques relevant to supporting the key tool requirements of an ENS, but the techniques overlap and none individually satisfies all the needs of even one of the key tools, let alone the combination. This may seem to imply that it is premature to consider any form of ENS standardisation since, after all, no actual ENS currently exists!

And yet, in the very crudest form an ENS *does* already exist – the Internet. What I have written here and in some previous newsletter contributions merely points out the ways the Internet needs to evolve to become, or to support the existence of a full ENS. Some aspects of the Internet which have become de facto standards are now subject to committees on standardisation and there are sure to be further enhancements to the Internet which require the attention of standardisation.

What I am talking about in asking for standards minimisation is ensuring that nothing is done in standardisation of the Internet, or later in standardisation of the key tools of an ENS or any other suggested Internet extension, which prevents this process of proposal, experimental implementation, evaluation and where appropriate adoption of proven new standards extensions.

The key mistake in addressing Internet or any other area's standardisation is to rush to the marketplace with a standard which is in reality premature and will either inhibit future important standards extensions or cause significant problems with unforeseen requirements to change legacy systems after implementation of non-backward compatible changes to the previously agreed standard.

In Unix they used to say, if a program has nothing to say it should say nothing. In standardisation that requires just a little modification. If a standard perceives a need for something to be said, but doesn't know what to say, it

should say nothing other than that it doesn't know what to say.

In other words, standardise only when you can prove that standardisation is necessary and a practically workable standard has been found.

RE-Publications

Book Reviews

Object-Oriented Systems Development, a gentle introduction
Carol Britton and Jill Doake
McGraw Hill
ISBN 007 709224 4

Reviewed by Ian Alexander

This is that rare thing, a textbook that is readable, unpretentious, and genuinely helpful. Britton and Doake have collaborated very carefully to present the basic concepts (and several more advanced ones) of object orientation. They have also managed to achieve one of the key goals of requirements engineering, namely to see things from the point of view of the users, in this case their students.

How many authors can truly put their hands on their hearts and claim, as this book does, that no technical knowledge is assumed? The authors make a serious effort to start from the beginning and build up the reader's knowledge piece by piece. They modestly claim only to be introducing the student to the subject, but one cannot help wondering whether their understanding and explanatory powers are not considerably greater than the guru authors of many more ambitious texts.

The book starts out by showing why there is a problem with traditional structured methods – the software crisis of ballooning complexity and cost overruns. It then looks at development approaches including the basic variety of system life-cycles and the methodologies that accompany them.

A chapter is devoted to requirements engineering, which in plain practical terms gives a helpful introduction to our subject, templates for interviewing and sensible advice on getting started:

"Although direct questions are needed to control the interview, a lot of information can also be discovered by smiling, nodding encouragingly and making the interviewee feel that what they are saying is important."

The book then starts on object modelling, with clear illustrations of different types of model including architect's drawings and maps, before diving into class diagrams. The reader is drawn into the concepts including cardinality, reuse, and association, without any of the usual fuss.

After that come use cases and scenarios, also good Requirements Engineering topics but here leading up to sequence, collaboration, and state diagrams to define the behaviour of a system. UML gets a brief mention.

The book then boldly moves into implementation, with increasingly detailed class diagrams for the worked example, a car park, and precise explanations of how to implement relationships such as association and aggregation. The authors sympathetically admit that

"We have found that for students new to the topic, one of the hardest things about implementing an O-O system is coping with distribution of functionality amongst classes… To achieve one use case we will find that we are jumping about between classes. This is especially disconcerting to students used to top-down functional decomposition…"

The book closes with chapters dealing briefly with persistent data and testing.

The extensive appendices cover the worked example, providing background material, models, Java code, and a summary of UML notation. Students get a crib of answers to selected exercises, a short but properly annotated bibliography of "accessible and reasonably easy books" including especially some "that we feel would be useful in the next stage of learning". The glossary is terse and helpful, as is the index.

RE-Bites...

The Landing Pilot is the Non-Handling Pilot until the decision altitude call, when the Handling Non-Landing Pilot hands the handling to the Non-Handling Landing Pilot, unless the latter "calls go around," in which case the Handling Non-Landing Pilot continues handling and the Non-Handling Landing Pilot continues non-handling until the next call of "land" or "go around" as appropriate. In view of recent confusions over these rules, it was deemed necessary to restate them clearly.

From a British Airways Memorandum (via the SRE mailing list), quoted in Pilot Magazine, December 1996

Students wanting to get started on systems development could do well to work through this book; teachers wanting a really practical first book on O-O should check it out. Old hands used to decomposing functions may also find it worth a quiet look while on holiday.

RE-Sources

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

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

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

RE-Creations

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

Deadline for next issue: 30th September 2000.

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

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

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

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

Mailing lists

The SRE list

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

subscribe SRE *your-first-name your-second-name*.

RE-Actors

The committee of RESG

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

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

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

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

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

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

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

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

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