Requirements Quarterly

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

© 2005 BCS RESG http://www.resg.org.uk RQ38 (December 2005)

Contents

RE-Soundings

From the Editor

In this issue, we are delighted to include a reflective article from our Patron, Professor Michael Jackson, on Requirements 'Engineering'. Some of us have always been a little doubtful about the E in RESG – more propaganda than substance perhaps. Read the article and see what you think (and then send us a letter about it!).

We have an exciting series of events planned for 2006: we hope to see you there. Coming along very soon is Scenarios Day on the 9th of February. This event combines a morning tutorial with an afternoon of talks from experts in an intentionally wide range of uses of scenarios.

Meanwhile, this issue reviews two interesting and very different books, reports on not one but two events related to dependability requirements (safety, security, survivability, and the like), reflects on the news this year, and looks at the evidence for why requirements are needed.

Have a very happy Christmas.

Ian Alexander,
Scenario Plus

Chairman´s Message

The final RESG event of 2005 seems like a good way to bring the year to a close. In part that's due to the Dependability event at Newcastle (see Ljerka Beus-Dukic's review later in this issue) being the first event successfully organised by me, so it's nice to have broken my duck.

More significantly, perhaps, is the fact that the event reminded us of the importance of understanding the wider environment in which systems operate. The discovery and expression of safety and security requirements through hazard analysis and threat identification requires us to think hard about the context and environment of the proposed system.

Safety and security are just properties of systems that happen to have stimulated much creative work on how to analyse problems and their context. The same concern, to understand all we can about the problem context, should drive all system development that involve significant risk, whether financial, to reputation or whatever.

The trouble is we can't know everything about problem spaces and some things have to be taken as articles of faith. We owe a debt to Donald Rumsfeld, for the clarity of his exposition of this phenomenon. He noted that the environment might contain:

- known knowns: things we know we know – like the stopping distance of a train of mass A, velocity B, track gradient C and the presence of leaves or ice.
- known unknowns: things we know we don't know. The key here is to recognise our ignorance and derive requirements that provide mitigation. It isn't possible to be sure what form the next security attack on our system will take, so this missing knowledge might motivate, for example, intrusion detection and recovery requirements;
- most pernicious of all are the unknown unknowns: things we don't know we don't know.

Perhaps what's needed is an analogue of safety engineering's ALARP (As Low As Reasonably Practicable) principle that, instead of classifying risk, classifies what we know (and thereby, indirectly, risk also) - AKARP.

Have a happy Christmas and a New Year free of unknown unknowns.

Pete Sawyer,
Computing Department, Lancaster University

RE-Treats

For further details of all events, see www.resg.org.uk

Forthcoming events organised by the RESG:

Scenarios, Stories & Use Cases Day

Thursday 9th February 2006

Northampton Suite, City University, London

This one-day event combines a morning tutorial on developing Scenarios (as Use Cases) given by Ian Alexander and Neil Maiden, and an afternoon panel of 4 distinguished speakers who will give talks on applications of Scenarios and Use Cases in industry.

The morning tutorial will cover the theory and practice of scenario use, including storytelling, capturing goals and sequences, discovering exceptions, and validating scenarios with role-play and structured walkthroughs.

The afternoon talks and speakers are:

- Sebastian Uchitel (Imperial College) on **Executable Scenarios**
- Helen Sharp (Open University) on **User Stories for Agile Software Development**
- Peter Haumer (IBM Rational) on **Use Case-based Software Development**
- Stuart Burdett (Defence Science & Technology Laboratory, DSTL) on **Using Scenarios to Develop Future Military Systems**.

Refreshments and lunch are included in the tutorial price. Tutorial participants receive a copy of the book *Scenarios, Stories, Use Cases*. The afternoon session is free to RESG members.

Contact and Registration N.A.M.Maiden @ city.ac.uk

Problem Frames

May 2006, Open University, Milton Keynes

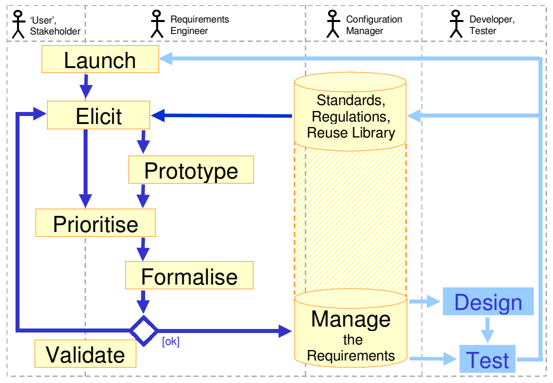
AGM and Distinguished Speaker Event

July 2006, London

IEE/RESG Requirements Days

2nd October 2006, IEE, Savoy Place, London: 8 talks covering all the essentials of requirements work, followed by a Banquet.

3rd October 2006, IEE, Savoy Place, London: 1-day Introduction to Requirements seminar. presented by Ian Alexander.



This seminar introduces requirements as a process of seven main steps, each supported by practical techniques, which are taught through group exercises.

The message of the course is that your organisation can take simple, cost-effective steps to do its requirements better.

http://www.iee.org/Events/intro-req.cfm

Early Aspects

October 2006, Lancaster

RE-Calls

Recent Calls for Papers and Participation

Electro-Technology Research Prize

The Royal Academy of Engineering is asking young academics and postgraduate researchers to put forward research proposals that can be exploited in the UK in the field of electro-technologies, including instrumentation IT, software and hardware. The winning individual or team will receive £10,000 to spend on themselves and £30,000 to develop the proposal.

http://www.raeng.org.uk/prizes/era

john.stephens@hq.bcs.org.uk

Mastering the Requirements Process

20-22 February 2006, London

presented by Suzanne Robertson, Atlantic Systems Guild

This 3 day seminar & workshop presents a complete process for eliciting the users' requirements, testing for correctness and recording them clearly, comprehensibly and unambiguously. Delegates will learn to:

- Determine their client' s needs, exactly
- Write complete, traceable and testable requirements
- Precisely define the scope of the project
- Discover the stakeholders and keep them involved
- Get the requirements quickly and incrementally

http://www.irmuk.co.uk/1

System Safety

The 1st IEE International Conference on System Safety

6-8 June 2006, The IEE, Savoy Place, London

System safety engineering (SSE) is the discipline concerned with achieving and assuring safety of systems, including their hardware, software and human elements. It encompasses, but is broader than, Functional Safety as it is concerned with hazards arising from physical causes, e.g. toxic materials and uncontrolled energy sources, as well as functional failures.

There are many challenges for SSE caused by changes in technology, increasing complexity and inter-working of systems, and reductions in the acceptability of risk to the public. SSE has to deal with these challenges – both societal and engineering – to ensure that deployed systems are acceptably safe, and remain so throughout their life.

A Free Trip to RE'06 for a Student AND for Someone in Industry

The RESG would like to encourage its UK members to join in the major Requirements event of the year (and this newsletter has always urged you all to attend - Ed.)

RE'06 will be held in Minneapolis, USA next September. We know that seems far away, so we are offering substantial help for one research student and one person from industry to get to Minneapolis.

For students: please submit a poster of your current research to the RESG student representative.

For people in industry: please write to the committee explaining why you would like to attend RE'06.

The winning entry in each category will receive £500 (this may be supplemented if necessary for the student) in return for proof of travel expenses.

We would also like to encourage you to submit papers and tutorial proposals to the conference. We would be happy to 'mentor' you through the writing process, though of course we cannot guarantee the outcome.

Student Contact: Zachos Konstantinos, City University, kzachos @ soi.city.ac.uk

Industry Contact: Dr Pete Sawyer, Computing Department, Lancaster University, sawyer @ comp.lancs.ac.uk (who will select the most suitable member of the committee to help you).

The RE'06 website is at http://www.ifi.unizh.ch/req/events/RE06/index.html

RE-Readings

Reviews of recent Requirements Engineering events.

RE for Dependability

University of Newcastle upon Tyne, 7 December 2005

The meeting, hosted by the School of Computing Science, started with a lavish lunch (thumbs up, hosts!) which got us all well prepared for the bleak prospect of being stranded at a cardiovascular department in an NHS hospital waiting to be rescued by firefighters. [See account of Chris Johnson's talk below for explanation. - Ed.]

For those of you who are not at home with the term dependability here is a brief definition:

> "Dependability is the system property that integrates such attributes as reliability, availability, safety, security, survivability, and maintainability."

The first speaker, **Charles Haley** from the Open University, gave a talk on "Arguing Security: Validating Security Requirements using Structured Argumentation". His talk was about security RE:

- determining intentional behaviour in problem space that can lead to harm
- adding requirements to mitigate the possibility of harm
- validating that these requirements serve the intended purpose

How do you determine an intention? Haley's answer is:

> "Understand the context!
> Know and test your assumptions!"

In order to prove that system satisfies security requirements, the OU's Security Requirements Group added satisfaction arguments and trust assumptions. Claims about system behaviour or environment are either argued (satisfaction arguments) or not argued (trust assumptions). Satisfaction argument structure (inner part) is based on Toulmin argumentation using grounds, warrants and rebuttals as claims.

[We'll have a review of Stephen Toulmin's classic book *The Uses of Argument* (1958) in the next issue of RQ. We reviewed *Visualising Argumentation* by Kirschner, Buckingham Shum, and Carr, making use of Toulmin argumentation, in RQ29. - Ed.]

The role of arguments is to expose assumptions which are security killers.

David Bush (UK National Air Traffic Services) talked about "Early lifecycle hazard identification using i*". His interest in determining protective safety requirements as early as possible, led him to a pioneering use of the i* approach. He combines i* with hazard and operability analysis (HAZOP). Identifying hazards early in system development offers scope for hazard removal, reduction and mitigation before requirements and design are committed.

Bush reported that use of i* greatly increased the number of hazards he was able to identify. Further, because it allowed problems to be expressed in terms of the problem domain, it was well received by key stakeholders

Chris Johnson, from the University of Glasgow, started his talk (Learning the Lessons of Hurricane Katrina: Developing Large-Scale Simulations for Hospital Evacuations) by showing interactive simulations of building evacuations. It was soon clear why he and his research group are interested in analysing and learning from past accidents: evacuation plans and live drills often provide a limited range of disaster scenarios. In his highly animated talk, Johnson pointed out some of the real scenarios which occurred during the evacuation of New Orleans hospitals. Events in New Orleans exposed the difficulty in extending human behavioural models from other public buildings (e.g. office blocks, entertainment complexes) to hospital evacuations.

While Haley and Bush are concerned with top-down analysis of and reasoning about the environment to try to identify the security threats/safety hazards, Johnson's approach is to do bottom-up analysis of actual incidents to establish causality. An interesting thing might be, while in principle the two approaches are complementary, in practice how well can we integrate top-down and bottom-up approaches like these?

The last talk was given by **David Greathead** from the host university. His talk on "DIRC and the Psychology of Programming" clearly illustrated that DIRC is a multi-disciplinary project, bringing together researchers with different backgrounds (in this case, computer science and psychology). As part of the Diversity research theme, David Greathead's and Cliff Jones's work looks for a link between personality type and code review ability.

Different personality types were identified using the Myers-Briggs Type Indicator which profiles personality along Jungian lines, based on four bipolar preferences (e.g. Thinking/Feeling, Sensing/Intuition).

Comparison of NT (iNtuition/Thinking) and non-NT students has shown that NT students performed significantly better at code review tasks.

These encouraging initial results will perhaps strengthen their intention to look for other areas of software development which may be linked to personality. One might ask a question (and I did) what is this to do with dependability? The answer was in fact given at the beginning of the talk: DIRC is concerned with the dependability of computer-based systems including hardware, software and <u>people</u>.

Despite scary stories linked to dependability (or better, lack of it), all participants left quite cheery. Perhaps, those strawberries perversely dipped in white chocolate, did a trick.

© Ljerka Beus-Dukic 2005

IEE Forum on Autonomous Systems

Savoy Place, London, 25 November 2005

Stuart Arnold (of QinetiQ, and chairman of the IEE's Systems Engineering PN) welcomed everybody to the meeting. The theme of the meeting was that by removing humans from the control loop, we could focus on new attributes of systems.

Bill Bardo of the Defence Technology College (DTC) gave a brief Keynote talk, entitled Directing Autonomy.

Autonomy is not only about vehicles, said Bardo, but also about sensors that can operate autonomously. Levels of autonomy range from 1 (human operated), 2 (human assisted like antiskid brakes on a car), 3 (human delegated, like engine controls) right through to 6 (fully autonomous). The only fully autonomous systems are the space probes with which we've lost contact — they're truly on their own unless they've picked up some alien friends, Bardo quipped.

Autonomy levels can equally be considered as a matter of levels of modelling from abstract and close to the user (world models, planning) via behavioural autonomy to concrete, "high fidelity" models enabling vehicles to control themselves.

The SE process might have to change for such systems. The DTC aims to do this both via specialised research into algorithms, planning, sensors, communications and control, power, and SE itself — bottom up or "technology push" and via a well-organised team to work on SE projects. The MoD provides the "project pull", demanding capability and providing "vignettes" — scenarios where systems would be needed.

The DTC aims to demonstrate that its research is useful rather than hoping that people will read its reports. Demonstrations could range from large-scale integrations and those using linking "threads" (ie scenarios again), through theme-level demos of different algorithms, sensors, etc, down to small demos by individual projects.

Other theories and systems — fractals, chaos, biology for instance — have important lessons for SE. Energy-using systems like animals and vehicles behave differently from systems at energy equilibrium (as Prigogine showed, said Bardo). Similarly, Shannon's law sets limits on information handling. Even quantum theory applies as it contradicts Shannon to an extent; all these things are now converging. Clearly SE is being considered at a deep level.

Joanne Thoms (DTC) spoke on Autonomy: Beyond OODA? She quoted the OED's definition of autonomy. This implies self-governing, imposing rules on itself because it understands what is going on. The OODA loop of Colonel Boyd is Observe, Orient, Decide, Act, and "it works". This provides closed-loop responses to observed events; it can't cope with multiple platforms, or future plans of the opponent, or long-term effects or indirect implications of its own actions.

OODA does not address human attributes like cultural traditions, genetic heritage, previous experience. Nor do we know if providing these would be useful for our goals. Wisdom is, she said, dealing with "known unknowns".

To orient well, our systems must be able to

- Perceive sensory stimuli;
- Predict and understand consequences of their own actions.

They'll then be able to

- Deliberate on current problems to make plans and achieve their goals
- Effect results.

But if we can't predict the actions of autonomous systems, how will we trust them? And if we can't trust them, will we use them? Going beyond human control is a big step, she observed. "I don't know how we will understand self-organising systems".

The basic capabilities do match the OODA loop quite well, but Deliberation covers the entire process, and goes beyond it.

Ian Alexander asked if we needed to ask for facilities to look "inside the box", to see what decisions the systems were making; back in the 1980s AI systems tried to be reflective, to report on themselves. Thoms replied that the challenge was to find a way that wasn't swamped in detail to articulate autonomous decisions, that related to understandable (abstract, human-level) goals.

If we give machines the ability to reason, how will they feel about being turned off? (Shades of Asimov). How much trust will they have in us, she asked. Evidently Thoms is not a narrowly siloed thinker and researcher.

Research now is different from AI; it needs to bring together different disciplines to succeed, but we don't yet know which ones we need, said Thoms. A rule-based system demands an exact answer; but we need a good-enough answer, not an optimal one. When multiple decisions all interact with each other (say 28 tactical processes and 12 longer-term ones), rules alone are not sufficient.

Duncan Priestley (Loughborough University) spoke on an Architectural Approach to the Challenges Arising from the Exploitation of Autonomous Systems. Warfare is changing: it's more often asymmetric (like guerilla warfare); it has a rapid tempo, demanding agile platforms that interact effectively with each other. This means that the environment changes too, making it non-deterministic. Decision-making is thus both necessary and difficult. Situational awareness must improve. How can we use autonomous systems in such a space? We can't just replace manned vehicles without trust, as Thoms said.

We need to evaluate designs in terms of core "–ility" requirements like survivability, interoperability, etc, said Priestley. This is a separate research goal. Architectures claim perceived benefits such as enhanced decision support, but these are hard to demonstrate. Vignettes including search-and-rescue scenarios (helicopters from a carrier respond to a call from a downed pilot in hostile territory…) provide a way of exploring the issues objectively. This is somewhat like the OODA loop but closer to old-fashioned wargaming and simulation. An event occurs, you observe and monitor leading to your recognising that the event has occurred, you evaluate the situation, determine your options, evaluate them, and finally choose one and implement it (before a deadline). You can try to save time on all of these steps except the last.

To do that, you need to evaluate competing architectures. The evaluation criteria depend on the scenario. Sometimes, interoperability, timeliness and flexibility may be key, eg flood rescue; in other scenarios such as anti-guerilla operations, security, speed and survivability may be the most important. Then the benefits can be scored (red for making scenario steps worse, green for better) by analysing scenarios as trees (a sequence of large goals decomposed into smaller steps) and colouring each node. A largely green tree means that architecture worked better than its rivals.

MoDAF and DoDAF are very deficient in decision support, said Priestley. We want to add something on to the side of those.

Phil Sutton (Director-General of Research & Technology at the MoD) spoke on the MoD's research agenda. He showed no PowerPoint slides (a rare thing nowadays). There are powerful destabilising forces in the world today. We mustn't "fight the last war" but need to focus on the value of human life and the issues today. This places a high responsibility on all of us who support the front line. MoD wants to work with the very best people, you (in this room), he said. We need to focus on what really matters: helping our forces to do their job in difficult circumstances, as well as possible and with minimum loss of human life. We want to develop new ways of collaborating with industry.

Joanna Thoms asked his view on autonomy and defence Lines of Development (LoD). Sutton replied that autonomy in its extreme form must change the way we do everything.

Eric Nettleton (BAE Systems) spoke on ANSER Past and Present: Multi-UAV Experimental Research. (ANSER is Autonomous Navigation Sensing & Experimental Research.) He made everybody laugh by showing a film of a UAV (Unmanned Aerial Vehicle, in this case the Brumby aircraft) that first chased a cow off its grass runway and then took off and did aerobatics.

Decentralised Data Fusion (DDF) looks at the problem of sensor fusion in a network. Each sensor is connected to a processing node that communicates with its neighbours. Each node ends up with the same, fused picture of what all the sensors have discovered about the situation. Trials have gone up to 8 nodes; simulations work up to thousands. To achieve such scalability, you have to manage bandwidth intelligently — there is never enough for everything. DDF does not communicate raw sensor information (that would scale with the number of sensors) but processes it first. Each node also only has to know about its neighbours, again saving much bandwidth.

Brumby flies at 100 knots and weighs 45 kg at takeoff; its wingspan is nearly 3 m and its endurance is up to 115 minutes. It has basic sensors for rudder, aileron etc, and two mission sensors: a black-and-white camera, and a millimetre-wave radar or a laser/vision system. Each mission sensor has its own processor and communications hardware, ie it is a completely packaged node. ANSER 1 uses its sensors and processing to build up a composite picture or map of the environment. Information is pressed in a fusion loop; new information is additive, an attractive and important property. This makes it fast as well. Each node subtracts out what is being reported from what it knows: the difference is what is new.

In trials the experimental targets were white plastic squares 3' on a side; wombat holes also proved good natural targets for recognition. Four aircraft together built up a tactical picture. This is quite hard to represent as a process. It was illustrated with a simulated terrain map over which the planes could be seen flying; each target was displayed as a white square inside an ellipse of uncertainty in position: the ellipses visibly shrank as more information was collected. The processing was all done on board on modest 266 MHz CPUs. The targets were identified with simple Calman filters.

Newer work is more general, allowing for recognition of more complex objects. The challenge remains to minimise communication load, eg not sending a million points to represent a probability distribution. Work is also progressing on classifying targets rather than just recording their position and velocity.

Hani Hagras and **Martin Colley** (University of Essex) spoke on Collaborating Multi-Robotic Agents for Operations in Inaccessible Environments. That includes underwater, in drainpipes etc as well as rugged places and battlefields. In such places you need small robots that can navigate autonomously, using little power and communicating over small bandwidths.

Identical miniaturised robots with limited capabilities collaborate to form a team. Each agent uses a biologically-inspired Spiking Neural Network (SNN). SNNs resemble neurons that deal with sensory-motor patterns (as streams of spike voltages called action potentials by biologists) with very little power and bandwidth; hence, neural networks can be simple and small. Because these work well in animals, they should do the same for robots.

Colley looked at making robots to work in small tubes such as oil refinery cooling pipes. These are subject to buildup of lime scale, which can block the pipes, so robots go inside to look for it. The smallest robots are about 25 mm on a side. They can detect scale by measuring local pH (acidity/alkalinity); this is just one example of the sensors that could be carried. When a robot detects no faults it wanders randomly; when it finds a fault (an alkaline region) it stays close to it (perhaps spiralling inwards) and signals to other robots to approach. Processing power is strictly limited as at this small scale you can't dissipate much heat.

We can reflect that this approach is strikingly similar to the behaviour of pond invertebrates. Perhaps robot evolution is recapitulating billions of years of biological evolution.

Moira Smith (Waterfall Solutions, which runs some research projects at the DTC) spoke on Fundamental Design Issues in Co-operative Autonomous Vehicle Systems, especially concerning sensors. These include underwater and other platforms, not just UAVs. There are many technology challenges as other speakers including Bill Bardo have indicated.

Imaging could be by Synthetic Aperture Radar (SAR) but affordability is a concern for all AVs. "Cheap" may be a matter of opinion in defence systems (laughter) but it is a key issue for research. For instance, motion causes linear optical flow (and blur); forward motion and roll causes diagonal flow. The image blur from such effects can be reduced by applying image processing (convolution filters) to recover sharp images.

Other challenges include maximising coverage of search space, and covering a whole area with the minimum of AVs and time. These require algorithms for guiding search and for collaboration among agents. Contrary to common belief, egistering images still remains a hard problem, said Smith. Registration gets hard as sensors move further apart; vibration, aberrations and different sensor modalities add complexity. New algorithms can do better than traditional ones. Results need to be presented readably. This might be by 3D imaging, cartoon diagrams or other means: an interesting research topic.

Dewi Jones (University of Wales) spoke about using UAVs to inspect power lines. There are 1.5 million wooden poles supporting 150,000 km of 11kV power lines. All have to be inspected regularly for large scale effects — sagging spans, leaning poles, and tree encroachment. Medium scale effects include pole-mounted equipment; small scale problems include shattered insulation … especially when wet. Birds can nest atop poles. Foot patrols, pole climbing (rare) and manned helicopter patrols are slow or expensive (or both). Helicopters have to fly 5 to 10m above live lines. Their surveillance could be enhanced with video cameras but that isn't usual today. Video gives better post-mission analysis than eyeball inspection; images can go into the company database; and job planning improves.

UAVs represent a next step from manned helicopters, offering more frequent missions and hence faster turnaround, lower costs (1/3 of manned helicopter's), and better data too. Unfortunately the CAA regulations do not allow UAVs to be operated out of sight of a human operator (1.5 km); up to 15 km is needed to make sense. Robotic "see and avoid" using machine vision is essential for safe and reliable operation.

An alternative idea would be a rotor craft (perhaps a ducted fan device like a tiny hovercraft) that picks up its power from the lines and hover along slowly over the lines. It is essentially tethered. As it can't fly without power it is impossible for it to fly off and cause danger elsewhere; it's not flammable; the blades are shielded; the electric motor is very quiet (so it won't frighten farm animals). It has a long range. And since it's constrained it can't be used to infringe civil liberties eg by filming people.

How could this be realized? It has to maintain its position relative to the lines. It has to keep its power pick-up in contact with the lines. It has to negotiate obstacles eg to turn on to a spur line, so it must fly freely for brief periods.

The requirements for visual power-line inspection are thus well-defined. They are quite unlike those for a battlefield UAV, and the vehicle proposed is quite different. Perhaps this means that autonomous systems research needs to proceed on many separate fronts rather than hoping for one universal solution.

Paul Newman (Oxford University) spoke on Automating Answers to "Where am I?". Mobile robotics will become ubiquitous, he said, as research advances among groups all over the world (he thanked his many international collaborators). Learning, moving and perceiving are all still "fiendishly hard" problems. But navigation remains an issue at the heart of robotics. "All higher level operations are dependent on location information." Newman thinks that planting markers all over the world isn't the answer. We navigate using the world itself to localise ourselves. SLAM, Simultaneous Location and Mapping, aims to find where the robot is in real time. Researchers now understand how to do this. A robot walks the corridors in MIT, scanning with its laser, and developing a 3-D map in real time, locating itself (with an uncertainty ellipse around its image of itself) as it goes (he showed a video).

How can this work outside in the real world? Do you make a Big Ben or Westminster Bridge detector, ie do you design your robot using prior knowledge of the world? Clearly it would be nicer to look at the actual context and see what is startling in that context: a speaker on an otherwise blank wall; a car in an otherwise empty carpark; a lake in an expanse of forest. That means identifying a local "normal" background, and then "exceptions" to that norm. That in turn can be handled mathematically, and in theory it could produce a SLAM map. But it doesn't work on a large scale as it doesn't handle uncertainty (which leads to drift, error in position). You need to be able to close the loop, to recognise when you have succeeded in coming back to where you started from. That enables you to correct for such errors. If you can match a sequence of images to another sequence, then you start to believe that yes, you've been here before, and the loop is closed.

The robot has to avoid being confused by repetitive structures (like the arches of New College Cloister, or similarly parked cars on a street). Newman's robot can now do this, driving around the outside of the Engineering building and correctly discovering when it has returned to its start point. The same software works sub-sea as well, with a yellow torpedo-like craft.

SLAM is starting to work; it's going to take years more research to add enough introspection to make it more robust. More semantics are needed to label places with their names and improve user interfaces. At the moment it can't be let loose for fear of falling over the roadside kerb, colliding with people, and insurance (laughter). SLAM is hard, quite well-understood, and above all central to mobile autonomy. A system-wide approach is needed.

Professor **Peter Johnson** (University of Bath) spoke on Interactive to Autonomous Systems. He is interested in changing from the old paradigm of humans-operating-things, through HCI (the current paradigm): humans-interacting-with-things; to the emerging paradigm of collaborating with things. Research issues include interaction, awareness, collaboration, and trust.

Collaboration means having shared goals (so it's not just co-ordination). This always comes with a resource overhead. Collaboration breakdowns are illuminating. Johnson has investigated the Qantas 747 flight that overshot the runway at Bangkok in 1999. It took an appalling 20 minutes to get everybody off (by a miracle nobody was seriously hurt). Collaboration failed in many ways. Knowledge of bad weather was not shared. The captain was ordered to "go round" but changed his mind. Automatic braking did not work because the captain reduced power on only 3 of the engines, missing the 4th, so the system inferred he was taking off and released the brakes.

Awareness is a poorly-defined concept. We barely distinguish levels of system awareness; understanding of goals, intentions, and implications is almost absent even in research work.

For instance, the artificial horizon display in aircraft cockpits shows the altitude on the right; to tell if the plane is climbing or descending the pilot has to do subtraction sums! A simple down-pointing red arrow below the altitude display and labelled 1000 could announce "we are trying to descend at 1000 feet/minute". But currently avionics designers are not trying to communicate intentions.

Trust too is important. The 747 crew trusted the braking system to stop the plane; the captain failed to trust the "go around" command. Both were misplaced.

Trust is a belief in an attribute of a thing upon which you have a degree of reliance. (Uncertainly and error are rightly implied.) Belief can be treated as a continuous quantity; reliance (the other component of trust) is boolean — you do or you don't. Thus there is a decision whether to trust something or not; so there is a decision-making process. I, the truster, trust something or someone, the trustee. My trust is mediated by something, eg an ID card.

Two problems arise with trust: misuse, ie inappropriate reliance on something (such as automation); and disuse, ie inappropriate rejection of something which is in fact quite trustworthy (such as automation, perhaps).

Research is looking at how Awareness, Trust, and Collaboration are connected. Then, ultimately we can design for collaborativity (a new "ility").

Clive Pygott (QinetiQ) spoke about the Outline Safety Case for the Use of Autonomous UAVs in Unsegregated Airspace. He too is interested in trust (see Peter Johnson's talk, above).

For UAVs to operate in civilian airspace, the CAA has to certify the aircraft type as safe. That means developing an outline safety case. There are obvious challenges. Agent software verification (using SOAR and CSP languages) might help. Currently as already mentioned the CAA demands a pilot within sight of all UAVs. The rules of the air say "avoid clouds" even if eg a radar is being used not visual sensing, as other aircraft don't expect planes to pop out of clouds.

A safety case was constructed using GSN, the Goal Structuring Notation. This essentially just describes an argument graphically, decomposing goals to show underlying strategies, concepts, subgoals, assumptions, and solutions.

The safety case itself essentially argues that the UAV is equivalent to a manned aircraft, as far as the Air Traffic Controller (ATC) is concerned. It has to behave predictably, and what is more it has to obey instructions from the ATC. What to do if the radio fails "is obviously something that has to be addressed" said Pygott.

The UAV also has to appear safe to other air users — it has to know the rules in the CAA's CAP 293 standard, even if those rules are a bit vague, with phrases like "near head-on" — perhaps that means a 3° cone, for instance. You also have to believe that the hardware is sufficiently reliable. Current full-authority avionics run on 4-redundant processors, for example. Having just 2 processors isn't enough: what do you do if they disagree? Either could be wrong.

Thus, some parts of the safety case are much like those for a manned aircraft; others are unique (can a 3rd party take control?), and several engineering challenges remain. Pygott's team can demonstrate there is no deadlock and no livelock: but that doesn't mean they know the software is correct.

GSN has helped to present the arguments exceptionally clearly, said Pygott. The approach would work for any safety-critical software. Progress has been made towards using the software agent paradigm and languages for such systems.

Pygott answered questions from the floor about the relationship of predictability and safety, and of transparency and safety: did a UAV have to behave exactly like a manned aircraft, and if so was achieving flight certification the same as passing the Turing Test (laughter). He said he was concerned with these questions only as far as the CAA insisted upon them.

It was striking how throughout the presentations the theme of requirements engineering recurred. Goals, scenarios, required qualities (the 'ilities'), evaluation criteria and even argumentation (using GSN) were at the heart of the problems facing autonomous systems: like all SE problems, perhaps. The quality of the talks and the discussions (both formal and those over coffee) was excellent: stimulating, reflective, intelligent, open.

© Ian Alexander 2005

***RE*-Papers**

But Is It Engineering?

Michael Jackson {jacksonma @ acm.org}

Professor Jackson is the Patron of the RESG. He is well known for his pioneering Jackson Structured Design and Jackson Structured Programming (JSD, JSP) methods.

Software developers have long aspired to join the ranks of recognised engineers. The famous NATO Software Engineering conferences of 1968 and 1969 were explicitly motivated by a belief in

> "the need for software manufacture to be based on the types of theoretical foundations and practical disciplines that are traditional in the established branches of engineering."

As members of the BCS RESG we regard our métier as a part of software engineering, and we call it requirements engineering.

The reports of the NATO conferences make good reading, but they reveal a curious blindness. Searching the report texts for the string 'engineering' produces significant references only to software engineering itself: there are none to the branches on which the participants hoped to model their own. The conferences were motivated by a desire to emulate the apparent successes of established engineering, but they didn't describe or discuss or analyse what established engineers actually do, or how they benefit from their theoretical foundations and practical disciplines.

One notable feature of established engineering is a firm division into specialised branches. A civil engineer does not design aeroplanes, and an electrical engineer does not design bridges.

The different branches share underlying principles and laws because they are all largely concerned with the physical properties of the world; the mechanics of structures applies both to the aeroplane wing and to the piers and roadway of the bridge. But these shared foundations lie well below the level of the design activity of a practising engineer, which is always highly specialised within one particular branch.

The established branches are characterised by their specialised products, and we should expect branches of software engineering, and of requirements engineering with it, to be similarly characterised.

Any colour you like as long as it's black…
Progress since the 1914 Model T Ford (via the 1936 Mercedes-Benz 260D, and a modern Jaguar) exemplifies *normal design*

This specialisation by product is recognisable in the software itself: an operating system kernel is very different from a compiler and from a database management system. Each is developed by specialist software engineers, and the corresponding products have improved in quality and reliability over the past decades. But the main weight of specialisation that should concern us as requirements engineers lies outside the computer, in the problem worlds of software-intensive systems. A system for administering undergraduate courses and a system for managing bank accounts may both rely on the same DBMS and the same OS, but they are very different in more important ways. They have different problem worlds, different interactions with those worlds, and different requirements.

The importance of specialisation is that it provides the indispensable context in which lessons learned about the engineering of systems of one particular kind can be developed, stored and disseminated among the specialists, and so made available to improve the products. This is the foundation of *normal design*, in which, according to Walter Vincenti (*What Engineers Know and How They Know It, Johns Hopkins 1993*),

> "the engineer knows at the outset how the device in question works, what are its customary features, and that, if properly designed along such lines, it has a good likelihood of accomplishing the desired task."

That normal design is a prerequisite of dependable systems is notably illustrated by the motor car industry.

Motor cars have been produced since around 1885, and in the 120 years of their development have reached a remarkable level of refinement and reliability. This success is due primarily to the fact that the design of each successive new model is only a small perturbation of the previous model's design.

Where there is no standard design precedent to rely on, the expectation of success drops dramatically. As Vincenti says of such *radical design*,

> "the designer has never seen such a device before and has no presumption of success. The problem is to design something that will function well enough to warrant further development."

If, as requirement engineers, we want to emulate the established engineering branches, we should pay a lot of attention to establishing and strengthening specialisations within our discipline. Without specialisation we are engaged in radical design, and we can have no presumption of success.

Another characteristic of the established branches that deserves emulation is their determined focus on failure.

Physical structures such as bridges must inevitably fail sooner or later, because the materials of which their parts are made are continually attacked by the elements and inevitably degrade over time. Other causes of failure are defective parts or a faulty implementation in which the product is not fabricated according to the design.

A major concern for the designer is to anticipate failures and to ensure—so far as possible—that they will not be catastrophic. Precedent and the canons of normal design play a large role here. They can indicate, on the basis of recorded experience, which failures are most probable and which are most damaging, and can help the designer to choose the most effective avoidance or mitigation. There are many lessons here for requirements engineers that deserve our attention. Do we routinely consider which requirements are more critical than others, and take care to specify a structure in which a failure of the less critical cannot cause failure of the more critical?

Engineered physical structures like the Golden Gate Bridge must inevitably fail sooner or later

Emphasis on failure may seem unduly pessimistic for requirements engineers. After all, software is not degraded by atmospheric conditions or the passage of time. Why should we not simply get it right, and avoid failure altogether?

There is a fundamental reason why failure is inevitable even in the absence of programming errors. The reason lies in the informal nature of the physical and human world, which limits our power to analyse the problem world of the system. Requirements engineering, from elicitation to software specification, rests on analytical models of the problem world.

In some developments, these models may be only implicit; in some they may be very informally stated; in some they may be explicit and formal. But they are always present, even if only as assumptions in the developers' minds. However conscientiously they have been made they will inevitably be imperfect: the problem world of a realistic system will always be capable of furnishing hard cases and unexpected conditions and behaviours that the developers' models do not accommodate.

The established branches of engineering deal with this problem—the imperfection of their analytical models—by over-engineering critical components. If the model shows that a 5W resistor will dissipate just enough heat, then a 10W resistor will dissipate more heat and will be less likely to fail. If the model shows that a 6" beam is strong enough to bear the load, then a 9" beam will certainly bear a heavier load.

Building codes mandate such safety factors, but they are not a panacea. They increase cost, make the produce more cumbersome, and may even be self-frustrating: the 9" beam may make the whole structure too heavy to support itself safely. They are, nonetheless, a vital tool in improving dependability of physical products.

In software engineering generally, and in requirements engineering in particular, this tool is rarely available. It is seldom clear how to increase system reliability by introducing a safety factor, especially where there is economic pressure to increase automation and reduce the human role in determining system function. The lessons of the established branches can not be carried over in any simple-minded way into the engineering of software-intensive systems. But we must not ignore them, as they were—so surprisingly—ignored in the NATO conferences so long ago.

© Michael Jackson 2005

A review of Walter Vincenti's book, What Engineers Know and How They Know It*, mentioned in this article, can be found below in this issue of RQ.*

RE-verberations

This section is for items of news that have a bearing on requirements work. RQ would like to hear of such things from its readers.

Why IT Projects are Different

The September 2005 issue of the BCS' *IT NOW* magazine reported a debate on whether "IT projects" (that's software projects to you or me) are really different from any other sort of engineering development. The debate is recorded at http://www.bcs.org/thoughtleadership/complex . RQ is interested to note that:

> "Programmes fail for management reasons, not technical reasons."

That seems a slightly sweeping statement, and indeed it is contradicted a short while later in the same article:

> "Many suppliers bid low, knowing they can get back the costs on scope changes and overruns.
>
> Requirements are often lacking – which can give suppliers another way to make money on top of their lowest-cost bids."

In other words, technical and management reasons are intimately interlocked – in fact so closely related that it is futile to try to think about them separately (they're all requirement concerns).

To complete the retraction of the initial claim, the article concludes (boldface emphasis added):

> "**Clarity of vision** of **what is to be achieved** is needed at the outset. Such vision also clarifies **what the programme is *not* going to do**, which is also an important issue.
>
> The **business outcomes** must be agreed. The key factor here is money as successful programmes need **a real business reason** for progressing. This means [that] the stakeholders who are going to benefit financially must be committed, which requires contracts."

The central place of requirements in "IT projects" is thus emphasized no fewer than five times. Perhaps "management reasons" aren't the whole story after all.

Safe Social Intercourse

The July 2005 issue of *IEEE Computer* contained a Technology News column '*Instant Messaging: A New Target for Hackers*' by Neal Leavitt.

The drift of the article is that Instant Messaging or IM (if you don't know what that is, ask your children) is the next fertile seedbed after e-mail for the propagation of viruses, worms, spyware, phishing, malware and other poisonous and nasty weeds in the computing garden.

However, as you might guess from its name, IM is much quicker than email:

> "The most dangerous part about the attacks is their speed of propagation, caused by IM's real-time capabilities… a simulation showed [that] IM viruses could spread to 500,000 computers in less than 30 seconds."

In 30 seconds. Hmm, that makes the 'rapid' 24-hour turnaround promised by anti-virus software vendors when they pick up a new virus (ok, I'll rephrase that) look rather flat-footed. Obviously this stuff is going to be highly infectious. And that's not the worst of it.

> "And adolescents, who comprise the fastest-growing segment of IM users, don't generally practice safe computing as much as adults…"

The adolescents, of course, use peer-to-peer transfer of files ('mom, I'm just swapping some music tracks') and messages with attachments,

> "so they bypass most of e-mail's server- and security-gateway-based virus scanning"

as the article cheerfully puts it. So IM viruses are far more infective, are spread especially by the young, and are far harder to detect than their conventional e-mail counterparts. Of course, a built-in IM virus scanner could find them, so you won't be surprised to hear that the *Gabby.a* worm works

> "by sending recipients a hyperlink and tricking them into clicking on it. Victims then get to a Web page that uploads spyware, as well as a worm that opens a backdoor to the machine and eliminates Windows services such as those used with antivirus and firewall software."

Charming: the worm behaves like an immune-suppressing retrovirus such as HIV that kills off your T-cells so it can then infect you at its leisure.

The article suggests some technological fixes: well, it is an engineering magazine, so what else should it do? One idea being tested is to queue requests to send to additional contacts. The idea is that if you have already communicated with partners a, b, c, and d repeatedly in the recent past, it was probably safe to do so, but new partners e, f, g, and h could be risky (honestly, I'm talking about computer viruses here). So the 'virus throttling' mechanism deliberately gets slower and slower the more partners the virus or worm tries to have. This should delay law-abiding IM work very little, but malicious programs should pretty much grind to a halt.

Other fixes include banning IM altogether ('Now then men, stay out of the red-light district during your shore leave') or at least having a policy on what it can be used for; and general virus-checkers that look for tricks like trying to overflow buffers or sending executable files as attachments.

Trouble is, no amount of contraceptive, I mean computing, research or advice is going to do much about adolescent behaviour. If you're talking about a socio-technical system of people and machines, it's only as secure as the people will let it be. Requirements that ignore this awkward fact about adolescents (and company workers) are known technically as 'wishful thinking'.

Yesterday's Weather

The 22 November 2005 issue of *The Cutter Edge* contained an article '*Yesterday's Weather*' by Jim Highsmith. He referred to a 1979 article in the Journal of the Royal Statistical Society about the use of time series analysis for weather forecasting. The researchers (Spyros Makridakis and Michèle Hibon) showed that a naïve "the weather will be the same as yesterday" (corrected for the season) prediction did better than any more elaborate model. I seem to recall that this caused some mirth on talk radio at the time. But even then, a weatherman who stood up and confidently announced a forecast was widely trusted.

Leaving aside the undoubted improvements in meteorology since then, Highsmith points out that Extreme Programming advocates planning based on "yesterday's weather", ie the next iteration is based on the deliverables of the just-completed iteration.

Planning is inherently uncertain, and many (if not most) attempts to look into the future border on the magical and superstitious. Highsmith hints that astrology and Tarot card-reading are roughly as effective as the finest management techniques today, with a major ingredient being confidence masquerading as knowledge.

> "High levels of uncertainty have a tendency to paralyze activity. People want better information before acting, but in fact, sometimes only acting will produce the very information they need."

This effect is beautifully illustrated with a story:

> "a group lost in the Pyrenees mountains .. used an old map they found to find their way to safety, only to discover that the map was of the Alps. Action, not the map itself, saved them. However, the map gave them courage..."

A quite different advantage of superstition is illustrated with another story (storytelling and scenarios are evidently powerful things). The Labrador Indians had a hunting ritual in which a caribou shoulder blade was scorched in a fire. When the bone cracked, the pattern of cracks was interpreted as a map of the hunting grounds.

> "Since the cracks were random, it actually benefited their hunting because they didn't overhunt [any] particular area. The superstition .. was justification for random action."

Superstition, therefore, can be good for you if it leads to decisiveness. (An old saying runs: "a good manager is decisive, and wrong no more than half the time".)

We may observe that consultants and trainers, too, have to be clear in their advice, just like management gurus. While we wouldn't want to admit to being, ahem, less than certain in our knowledge, the fact is that it's better to give a clear half-true reply that people can understand and act upon, than a complex muddle of hedged bets. GCSE questions are invariably hard to answer if you consider all the exceptions and ifs and buts. The right approach (if you are consulted by your children) is to forget all the complexities, and consider what answer must be expected at that level.

So, the Use Case rule:

> Describe the Normal Course (or "Happy Day" scenario) first.

is a good one. First, tell the simple, clear story, and in many situations that will be enough. When more is needed, and when people are ready for it, then go into the special cases and the A-level answers.

The Cutter Edge *is available from* `http://www.cutter.com/research/sample.html`.

RE-flections

One Counter-Example Is Enough

There is quite a debate going on about the evidence for why requirements are needed. The Standish 'Chaos' Report has come under fire and its owners have so far kept an eerie silence. What the Standish Corp. and others have tried to do is to gather statistical evidence on how many projects fail, and for what reasons. This is pretty hard to do, as failing projects may well want to keep quiet, and companies do not want to give their competitors ammunition in the struggle to win new work.

However, the Chaos report did not assert that, for instance, bad requirements work caused x% of all projects to fail. Instead, it asked large corporations how many of their projects failed, and then which factors it considered important in causing failure.

There are plenty of things you could say about this as an inquiry method – for instance, suppose that 3 out of 4 corporations just threw the survey form in the bin, and that of the rest, most of the forms were filled in by the process improvement department. The 3 out of 4 might have been those who didn't acknowledge that requirements were even an issue, and who consequently had the worst projects: we don't know, and nor presumably did Standish.

Process improvement departments might deliberately colour their responses to show how vital their work was, and therefore how problem-ridden the unimproved projects were. Shock! Horror! There are heaps of problems! Quick, spend money on process improvement!

Furthermore, a list of factors that someone thinks might be causes of failure is not the same as an analysis of what actually went wrong on any given project, nor as it might seem to be, an analysis of the relative impact of the different causes of failure of a set of projects: it's a generalisation at arm's length, a piece of 'tender-minded' thinking (see the book review below).

Quite different statistical evidence could be collected by experiments on students, but the applicability of such results is also a moot point.

So, if statistics cannot deliver, is there any way to gather evidence that bad requirements work is a problem?

Yes there is.

Remember that all you need to disprove someone's claim that X does not exist is one instance of X, or two or three if you are cautious. You don't have to do statistics to try to show that lack of requirements is always bad; you just have to find one case where it was a serious problem, and the claim that requirements work doesn't matter is disproved. In other words, showing that

$$\exists X: \text{problematic}(X)$$

is sufficient, and a great deal easier than showing that

$$\forall X: \text{problematic}(X).$$

Let's be concrete about this, then.

- Do I know of an actual project where lack of requirements work caused disaster?
- Yes.

A scientific analysis-and-modelling package project was knocked together based on a chat in a pub, as a 'good idea' that was 'obviously useful' and which 'could easily be developed by the graduates'. No-one was given responsibility for the overall specifications, and the system lacked a coherent architecture and interface specifications. In fact there were scarcely any written requirements. Nobody looked at the user interface either. The new programmers did their best, and the individual components, some of them doing very complicated three-dimensional geometry and interpolation, certainly worked. But the interfaces were a continuing and very costly nightmare, and when the product was finally shown to clients, it remained very buggy.

At least as importantly, it was very difficult to use: nobody had thought how it would be applied to problems, or indeed what problems it would actually be used to solve. The system was 'inside-out', making sense to the designers of its component parts, but almost no sense to its users. A whole lot of the complicated work had to be done in the user's head. Not surprisingly, it didn't exactly sell like hot cakes.

Let's go further.

- Do I know of an actual project where a single bad requirement was enough to cause serious trouble?
- Yes.

An image processing and distribution service in the days of slow modems and plain-old-telephone-system phone lines allowed users to view a catalogue with thumbnail images, and to place orders for high-resolution images. This meant that users had to be allowed to ask to see what they were about to buy before committing themselves.

The requirement should have said something like

> "Users shall be able to view thumbnail images."

Unfortunately it said

> "…to view images."

Yes it did. At the acceptance tests, the customer asked to be shown how to download an image. The head programmer showed him how to get a thumbnail on screen in a few moments.

> "Yes, that's fine, but I want to see a full image please."

There was some umming and erring.

> "Here it is in the requirements specification, 'view images'."

> "Yes, but that will take 24 hours or so over the slow modem link; the phone line will go down every few hours, so you'll never be able to transfer a whole image."

> "Never mind the 'yes, buts', it says I can have it and I want it."

Really. So the project went over time and over budget to do its best to make it possible for the customer to do something that was never going to work properly given the available technology.

The developers of the image distribution service got paid eventually, which is more than happened in another company which foolishly included the requirement

> "Each search shall be completed within 2.5 seconds"

in their **telephone system** specifications. At the acceptance tests, the developers showed off the pre-arranged searches which all easily passed. Unluckily,

> "each search"

could include the surname

> "Chang"

which is very common in countries with a large Chinese population. The customer asked for a demonstration, which, as he presumably anticipated, failed by a wide margin.

The search was painstakingly optimised, but the time never did come down below 3 seconds, and the customer got a free system. He'd found an existence proof of a broken requirement, and didn't need statistical evidence to see him through a lawsuit: he had a watertight case, and he knew it.

Could you give similar examples? Yes, I thought so. Let's not be shy: ***we know*** bad requirements can be disastrous.

© Ian Alexander 2005

RQ readers are invited to send in their experiences of requirements disasters they have personally *witnessed.*

Are Contracts the Root of All Evil, as exemplified by the cases mentioned here, or is something else going on? Let us know what you think.

Special Guest Proverb

After all that about existence proofs of very small but critical requirements problems (with a Chinese flavour), we need a proverb about…, well, you work it out for yourself:

It's not the mountain under your feet:
it's the stone in your shoe that wears you down.

Attributed to Confucius.

RE-Creations

To contribute to RQ please send contributions to Ian Alexander (ian @ scenarioplus.org .uk).
Submissions must be in electronic form, preferably as plain ASCII text or rtf. Deadline for next issue: 15th March 2006

RE-Publications

Prof. Michael Jackson recommended the following book (see his paper in this issue).

What Engineers Know And How They Know It

Walter G. Vincenti, Johns Hopkins, 1990

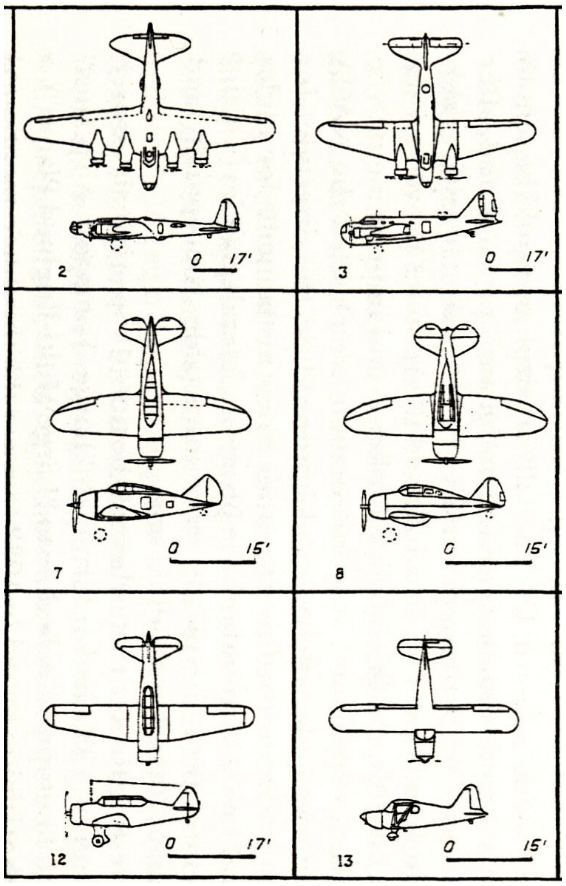
What is engineering knowledge anyway? Is Engineering just Science's poor cousin, Progress wrapped in an oily rag, or is there something distinctively different about it?

One answer, of course, is that engineering must be practical. Theories can be fine and elegant, but if the plane flies and the bridge stands up, the engineer was right regardless of how the result was arrived at.

But Vincenti goes about his task in a more satisfying way. In William James' terms, he is 'tough-minded' [1]: he collects evidence instead of creating vague theories. An aeronautical engineer, he's taken a careful retrospective look at five aspects of the growth of knowledge in aeronautical engineering between 1908 and 1953 - the time when things were uncertain, and people were feeling their way sometimes almost entirely in the dark towards safe, reliable, predictable ways of making aircraft.

This knowledge is not necessarily spectacular, but it is the kind that matters. How do you rivet a thin metal skin on to a support without leaving the head of the rivet sticking out, and without cracking the skin? Should you counter-sink the hole? Should you hammer the metal into shape first? Can the rivet itself act as the shaping tool? Does the metal have to be hot? How do you make sure the rivet goes in straight, and is hammered tight enough? There are plenty of wrong answers, and they are safety-critical. Could scientific inquiry answer such questions? Possibly, but it would take an immense amount of time and work. Instead, engineers try things out, measure, analyse, and inspect, in a careful but more limited, and in a sense also a more focussed way.

Vincenti warns that the stories will not be easy going, and this may be true for historians coming to his material. But engineers will probably not find the technicalities at all difficult; the story depends on them, of course, and the tale is fascinating, gripping as a good novel.



Vincenti's accounts are all the better for being both true and widely applicable to the questions that arise in development work such as requirements definition today. In fact, one of the 'stories' is precisely about 'The Establishment of Design Requirements'. Not surprisingly, it's the longest and most complex of the chapters, but it was almost impossible to put down. It is rich in details, often astonishing, sometimes funny:

> In an exchange of penciled intraoffice memoranda, Captain Hatcher let down his hair with a candor that rarely appears in more official documents:
>
> > At present we simply specify that the airplane shall be perfect in all respects and leave it up to the contractor to guess what we really want in terms of degree of stability, controllability, maneuverability, control forces, etc. He does the best he can and then starts building new tails, ailerons, etc. until we say we are satisfied.
>
> Unsatisfactory airplane characteristics were doubtless due in part to the inability of designers to design for what was wanted. As indicated by Hatcher's memo, however, what was wanted was far from clear. (p77)

Couldn't happen today? Well, what is a 'fitness for purpose' clause, then?

Vincenti is masterly in showing how engineers fumbled towards knowledge; few writers have attempted anything like this. Sometimes, Vincenti shows, the community was led astray by prejudices held by special groups of stakeholders (as we' d now say), especially pilots; sometimes by common sense; sometimes indeed by the apparent implications of (correct) mathematical modelling!

Most startlingly, perhaps, it turns out that stable, reusable requirements are the OUTPUT of years of effort trying to understand the properties of different designs in a domain. The requirements epitomise the laws governing a wide range of possible designs (not all imaginable aircraft: helicopters for instance certainly behave quite differently).

What price 'solution-independent' user requirements, then? Today's aeronautical engineers see it as obvious that manoeuvrability depends on stick-force-per-g, ie that an agile fighter plane must respond to a light pull on the stick. But that wasn't at all obvious for 25 years of research and development!

Perhaps there are things we know but can't explain -- along the lines of Polanyi [2] (in fact, Vincenti does mention that skilled tasks such as riveting 'can't be learned from the book' but are in the 'neuromuscular skill' of the practitioners) and important practical things that were known that are being forgotten. Reflection on practice, as Donald Schön advocates [3], is valuable, but easier said than done.

This is a wonderful book, written with an engineer's precision, a historian's eye for detail, and a storyteller's clarity. It is impossible to read it without being struck by one reflection and insight after another. Learning is difficult, when you're doing it for the first time! And of course, every project is a new set of discoveries. Students and experienced engineers will find Vincenti invaluable. It is a book to muse over, to take practical ideas from, and to come back to.

[1] William James, *Pragmatism*, Lecture 1: The Present Dilemma in Philosophy, 1907 http://www.authorama.com/pragmatism-2.html

[2] Michael Polanyi, *The Tacit Dimension*, Doubleday, 1966 (reviewed in RQ25, January 2002)

[3] Donald A. Schön, *The Reflective Practitioner*, Basic Books, 1983

Competitive Engineering

A Handbook for Systems Engineering, Requirements Engineering and Software Engineering Management Using Planguage

by Tom Gilb

Elsevier Butterworth-Heinemann 2005

Tom Gilb is well known as a speaker and consultant, as the author of *Principles of Software Engineering*

Management (1988) and of *Software Metrics* (1977), and as the inventor of "Planguage", a way of structuring requirements.

This is an unusual book in several ways, and as such not easy to review. It might well have been entitled "The Planguage Reference Manual" as a large part of its structure is a systematic walk through the main components of the approach: the language's Concepts and its 'Parameters and Grammar'; and the processes that use the language. The introduction makes clear that the book will not be very readable: it is somewhat expanded from the dry style of a reference manual, but not by much, and the sharp edges of the hierarchical structure are everywhere visible. That in itself makes the book something of a challenge.

Despite the fact that the Planguage glossary section takes up nearly a third of the book, it's still only about a quarter of the current glossary, which is still growing. There is however a "complete" glossary of Planguage on the Elsevier website (http://books.elsevier.com/companions/0750665076/ along with a review by Jerry Huller of Raytheon, and a sample chapter, both useful if you want to know what the book is like). The book is thus an incomplete introduction to the reference material. This is an odd compromise: perhaps a more 'chatty' (to use Gilb's own word) overview in the style of a tutorial would have been a better choice.

The book, however, is far from consisting only of reference material. Gilb' s thinking is based on several famous names from an earlier age, notably Shewhart and Deming' s Plan-Do-Study-Act, or Plan-Do-Check-Act if you prefer. (By the way, Gilb draws it as a rectangle rather than the usual circle with 4 arrows - an idiosyncratic choice as the symbol for a cycle.) That is of course a management take on process control: continuously iterate between measuring and putting a hand on the tiller to keep the boat on course. Iterative life-cycles (Gilb calls this Evo, which stands for both 'evolutionary' and for 'evolutionary project management') are much in fashion today, from the CMM's concept of continuous improvement to the rapid cycles of Agile software development.

The book, too, is full of simple plain wisdom, often expressed in aphorisms and proverbs. For instance, "The basic ideas of S[pecification] Q[uality] C[ontrol] are simple: A stitch in time saves nine... An ounce of prevention is worth a pound of cure". Or: "It is important to distinguish 'ends' from 'means'"; design ideas are labelled ' false requirements' . Good sensible stuff. Similarly the idea of looking for the 'few key requirements' is advised: "I often use the concept 'Top Ten'". The boxed reminder immediately below runs:

> Remember, for many projects, even delivering a single top objective on time and to financial budget, would be an advance on their current experiences!

This is the voice of experience (and of an expert communicator and trainer): it's obviously true, and it suggests a good place to start if a project is in a mess. But it sits oddly with the formal structure of the body of the book.

Other attractively homespun elements include Tenniel's fine cartoon of Alice (in Wonderland) talking to the Cheshire Cat. Alice says she doesn't much care where she gets to; to which the Cat replies "Then it doesn' t matter which way you go". In the context of getting projects to create "clear measurable requirements" (Philip Crosby) this may strike a chord with some beleaguered readers.

The nub of the problem posed by *Competitive Engineering*, however, is whether this sound advice and practical wisdom means that one has to accept Planguage lock, stock, and smoking barrel. The author is clearly aware that it's a lot to swallow, and the reader is urged to look at a little at a time, to take whatever he needs, and so on. However, essentially the book presents a solution approach for all the fields mentioned in the book's subtitle, complete with its own private language and notation. This has many aspects.

For example, "performance requirements" are stated to include reliability, portability, and usability; the term "binary" is used to mean "Boolean"; "function requirement" is used instead of "functional", "meters" instead of "acceptance criteria" and so on. Other seemingly essential concepts seem to be missing altogether: you won't find scenario (or use case), decision, verification and validation, or even trade-off here. But perhaps the most striking absence is the ordinary literature on requirements engineering, software engineering, or systems engineering. Does it all count for nothing?

There are certainly some good things in this book, not least its strong emphasis on iteration, on communication, on clarity, on reducing risk, and perhaps especially on measurement (there are telling quotes from Lord Kelvin and Simon Ramo on that subject). But these things are the daily stuff of systems and software engineering: they are the purpose and bread-and-butter of requirements work, and Planguage has no monopoly on any of them.

This book is clearly aimed at industry, and it is associated with a successful consulting and training practice. It contains much that is helpful and practical. While the analytic definitions of principles, rules and concepts may prove too difficult for many readers, the plain advice (for instance) to define your requirements in terms of stakeholders, scope, conditions, rationale, dependencies, links, and testability can hardly be faulted.

© Ian Alexander 2005

RE-Sponses

RQ welcomes comments and reactions to articles and reports published in its pages.

RE-Membering

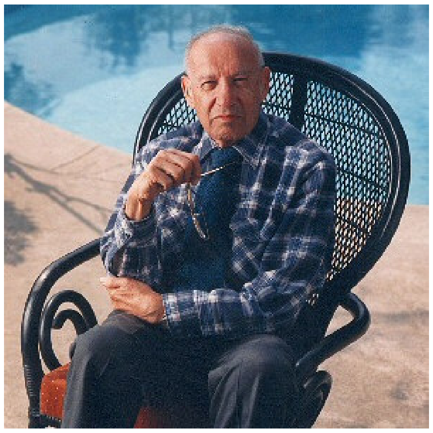
Peter Drucker, the First Management Guru

"Management is doing things right; leadership is doing the right things."

Peter Drucker died on the 19th November 2005, aged 95. The world has changed greatly since 1909, and some of the change for the better in management thinking is certainly down to him.

Drucker was that rare thing, an academic (for half a century) firmly rooted in practicality: interested in theory that works. He believed passionately that managing was about engaging with people (really, you can see some parallels with RE in all this?). And he is supremely quotable: a product of the clarity of his thinking.

"There is nothing so useless as doing efficiently that which should not be done at all."



Some nowadays think of Drucker as the father of targets and management by objectives. But (alas) Drucker is no exception to one lamentable rule, which is that devotees generally misunderstand the teachings of their gurus.

He felt that objectives had to be set by agreement with all concerned (what we'd call the stakeholders), not by imposing targets from above. He was wise and experienced enough to know which of the two approaches was likely to work, as many in high office today do not.

"Plans are only good intentions unless they immediately degenerate into hard work."

Drucker also believed that management was a practice to be learnt through experience, not a subject that could be taught to greenhorns.

"Teaching 23-year-olds in an MBA programme strikes me as largely a waste of time. They lack the background of experience. You can teach them skills - accounting and what have you - but you can't teach them management."

In the case of requirements, there are plenty of useful skills that can be taught to students, but the systems engineering aspects are, at the very least, much easier to teach to engineers who have some experience under their belts.

Drucker decided not to become a professional economist because his interest was in people not numbers. Management is far more than just making a hopeful GANTT chart in a PM tool, and Drucker would have had little time for email or telephone mandarins. He saw management as a combination of essential functions:

- planning,
- organising,
- motivating,
- measuring, and
- co-ordinating.

Academic fashion is as fickle as any other kind, and Drucker's practical, common-sense brand of management theory is far from current business training and theory.

For all that, he remains admired and respected both by fellow academics, and perhaps more importantly, by practising managers everywhere in the profession that he more than anyone helped to create. He's perhaps less well known by engineers, which is a pity, as he's a like mind.

Let's leave the last word to Drucker himself. It's just as applicable to engineering as to management, and of course the two must go together.

"The best way to predict the future is to create it."

Go to it.

© Ian Alexander 2005

RE-Sources

Books, Papers

See also the 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 "Manage Requirements":
http://www.goldpractices.com/practices/mr/index.php

Mailing lists

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

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

subscribe RE-online <your email address>

LINKAlert:

http://link.springer.de/alert

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

RE-Actors: the committee of the RESG

Patron:

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

Chair:

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

Vice-Chair:

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

Treasurer:

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

Secretary:

David Bush, National Air Traffic Services,
David.Bush @ nats.co.uk

Membership secretary:

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

Publicity officer:

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

Newsletter editor:

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

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

Regional officer:

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

Student Liaison Officers:

Zachos Konstantinos, City University,
kzachos @ soi.city.ac.uk

Andrew Stone, Lancaster University,
a.stone1 @ lancaster.ac.uk

Immediate Past Chair:

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

Industrial liaison:

Prof Wolfgang Emmerich, University College London,
W.Emmerich @ cs.ucl.ac.uk

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

Gordon Woods, Independent Consultant,
gordon.woods @ bcs.org.uk

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

The Requirements Engineering Specialist Group of The British Computer Society

RESG
www.resg.org.uk

Individual Membership Form for 2005

1. Membership Type

Please indicate type of membership:

		BCS/IEE members, please
BCS / IEE member (£10)	[ ]	*indicate membership number*__________________
Non-BCS / IEE member (£20)	[ ]	
Full-time student (free)	[ ] *Studying at:*____	

If you need a receipt, please tick here [ ]

Corporate Membership also available – details at www.resg.org.uk or ask the Membership Secretary

Payment by cheque only. Please make it payable to 'The BCS Requirements Engineering Specialist Group"

____

2. Your Details Title: Mr/Mrs/Ms/Dr/Professor/Other:_______(delete as appropriate)

First Name:__________________________Surname:____

Address for correspondence:____

____Postcode:______________________

Phone:____Fax:________________________________

E-mail address: *Please write your e-mail address clearly using BLOCK CAPITALS*

As an RESG member your e-mail address will be added to the RESG mailing list

Optionally, indicate:

Your organisation's name:____

Its type of business/domain:____

3. Your Specific Interests

Areas of interest for the RESG given at *http://www.resg.org.uk/about_us.html*. Is there another area would you like us to add? ____

4. Your Participation Preferences

Please indicate what timings/duration for RESG events would suit you:

Whole day [ ] Half day [ ] Evening [ ] Other (please specify) [ ]_____________

Please indicate whether you would be willing to help the RESG with:

Publicity [ ] Newsletter contributions [ ] Organisation of meetings [ ]

5. Your Mail Preferences

The RQ quarterly newsletter is regularly sent to all RESG members. It will be sent as a PDF e-mail attachment unless you prefer a hard-copy of RQ delivered by post. For hardcopy delivery, please tick here [ ]

I understand that the information supplied on this form will be held in a database and used for the purpose of distributing information relevant to the activities of the RESG.

6. Your signature:____Date:________________________________

Please send this form with payment (if applicable) to: Lucia Rapanotti
RESG Membership Secretary membership-RESG@open.ac.uk Fax +44 (0)1908-652140
Computing Dept., The Open University, Walton Hall, Milton Keynes, MK7 6AA, U.K.