bcs The Chartered Institute for IT

Requirements Engineering Specialist Group

Requirements Quarterly

© 2011 RESG http://www.resg.org.uk RQ56 (May 2011)

Contents

RE-soundings *1*
From the Editor **1**
Chairman's Message **2**
RE-treats *2*
i* Showcase'11 **2**
RESG – Agile Workshop (ECOOP'11) **3**
REET'11 **4**
LEAN Methodologies Workshop **4**
RESG AGM and Annual Party **4**
Careers in RE **4**
RE-member *4*
REET Workshop Summary **4**
Brownfield Evening, 2010 **5**
Careers in RE **7**
RE for Self-Adaptive Systems **8**
RE-flections *11*
Sky-High Goals, Lower Costs **11**
RE-verberations *12*
HFS: Human Flesh Search **12**
RE-writings *14*
Managing Complexity in Large Development Programmes **14**
RE-partee *21*
RE-sources *21*
Books, Papers **21**
Media Electronica **21**
RE-actor *22*
The committee of the RESG **22**

RE-soundings

From the Editor

This issue sees me take over from Simon Hutton as editor – thanks Simon for 2 years of RQ! The delay in getting the current issue out should be put down to this reshuffling behind the scenes and me relearning some basic word-processing skills. A couple of years ago I stepped down as Publicity Officer, after 5 years of managing the website and mailing lists, so it's good to be thrown back into the fray once more.

As Ian notes below, the RESG is well into its second decade and still going strong. Though perhaps no spring chicken, the reports of recent events in the following pages should convince you that the RESG is nonetheless still filled to the brim with youthful exuberance. Anyone who didn't realise RE was fun should turn immediately to the report of 2010's REET workshop. And we look to a future filled with robots and spaceships in the report of our recent Self-Adaptive Systems event and in a reflection on the commercial space program.

Also well-reflected in the current issue is the way the RESG successfully brings academics and practitioners together, with many events seeing the two camps rubbing up against each other in newly productive ways. The reports of the Brownfield Evening, Self-Adaptive Systems workshop and, of course, the Careers in RE events all bring this out in different ways. The academic / industrial crossover is also apparent in the third and last instalment of the Managing Complexity series, which concludes the issue.

We see it also in the events to come: the i* Industry event at City University in June, the Agile Workshop at ECOOP in July, and of course there's always a friendly mix at the Annual Party, where we hope to see you all. May the cross-pollination long continue!

William Heaven, Editor

Chairman's Message

Several of your committee members have this month received BCS long service lapel badges – mine is for (more than) ten years' working for the RE community that is the RESG. In that time it has changed considerably. RE is today much better accepted than it used to be; but it remains poorly understood. Some old truisms still work: people continue to think RE is simple, but it's still not easy.

One of the hidden challenges that makes RE less than easy is the question of domain-specificity.

It is, frankly, impossible to believe that, say, requirements discovery does not consist of a familiar, repeatable set of problems which need the application of a broad set of techniques, wherever it occurs. You need to model the context, identify stakeholders' goals, discover scenarios, capture the rationale for contentious decisions, prioritise, and so on. Everyone has their own patent approach to asking all these questions; and not every kind of question is needed in every situation, but the fact is, we broadly agree that learning how to model goals and context and the rest is a good idea.

And the same goes for the other half of RE, requirements management. Your project may need a big heavy RM tool, or may be small enough for a simple spreadsheet to do the job. But we pretty much agree that keeping some kind of shopping list, and ticking off each item when you've got it, is not so much a good idea as, frankly, absolutely necessary.

So RE is domain-general, then? We are all agreed?

But the people in every domain any of us have ever come across believe that their domain is special!

Clearly, something is missing here. I suspect the answer lies in one simple truth, which we'd all instantly accept, but for the fact that it contradicts a central dogma of RE. The dogma? That requirements can be written, developed, done, without knowledge of design. The truth? That requirements always reflect design.

Well, let me qualify that. The word requirement is unfortunate, for it has two entirely different meanings. The first is a genuine business or stakeholder need – for example, you must record every invoice you issue, for tax purposes. There is no mention here of design, of how the recording is to be done. The second is an item of specification at some level – whole system, subsystem, component, whatever. For example, the word-processing subsystem must save each invoice that the user creates. The specification item openly acknowledges a design decomposition into named subsystems.

Now, in a domain – like railway or tele-communications – much of the design decomposition is embodied in widely-shared conventions. These may be standards, or regulations, or simply the ordinary assumptions about how things always work in the domain. You can call it ontology if you are into high-sounding terms of Greek origin: for example, that trains run on two rails of standard gauge, or that billing stops when the caller puts the phone down.

So in such a domain – I suspect it's EVERY domain – people believe firmly that RE must be domain-specific. Everything – the format, the approach, the techniques used, the language – will be tailored intricately to the domain.

But we believe, don't we, that requirements, when pure, when separated from mere design questions, are the same everywhere? Ah, but type 2 requirements, that is, items of specification, by definition cannot be separated from design. They are design. And they make up the overwhelming majority of the things that are called requirements.

In that case the task of the RESG is, on the one hand, to clarify and refine the toolkit which is 'pure' RE; and on the other, to implant this toolkit in each domain we encounter: to specialise and tailor and adapt it to the domain, so that it works in that context.

The only thing we have to do is to drop the central dogma. Requirements, or rather specification items, can never be free of design. Perhaps that won't sound so heretical by the time the newest members of the RESG committee get their long-service medals.

Ian Alexander, RESG Chair

RE-treats

i* Showcase'11: Exploring the Goals of your Systems and Businesses

21st June 2011 - City University

This is a joint RESG event with the i* community.

As the i* modelling approach is becoming more widely known, more case studies of practical applications are becoming available. This half-day event targeted at practitioners aims to showcase a variety of practical applications of i*, and to provide a forum for exchanging ideas and experiences among users and researchers. We seek contributions for presentations and posters demonstrating the practical use of i* modelling in a wide variety of application domains and usage contexts. The emphasis at this event will be on practical applicability and utility, rather than on research novelty. Presentations from completed projects as well as work-in-progress are welcome.

Workshop format

- An overview of i* modelling

- A round of short presentation (5 minutes each) to present highlights of each poster, and to illustrate the diversity and breadth of usage scenarios and application domains.

- 2-3 detailed presentations to help clarify and reinforce i* concepts and the kinds of analyses presented in the introductory overview and tutorial.

- Poster session and socializing - for face-to-face interactions and in-depth discussions around each poster. Participants are encouraged to bring laptops for demos around each poster.

Important Dates

Submission of proposals: 27 May 2010

Send to Jennifer Horkoff: www.cs.utoronto.ca/~jenhork/

RESG – Agile Workshop (ECOOP'11)

26 July 2011 – Lancaster University

Ever since agile first emerged in the 1990s, there has been debate about what role, if any, requirements engineering (RE) can play for agile practitioners and their customers. That agile development needs requirements is not disputed, but the relevance of the assumptions, methodologies, techniques and tools that make up the discipline that has become known as RE, is. Thus, while agile emphasizes incremental discovery and satisfaction cycles with face-to-face interaction rather than documentation, RE has traditionally stressed full understanding of requirements before commitment to coding and rigorously maintained, version-managed and traced requirements documents. Yet both of these views are stereotypes, rendered even less valid by the evolution that has occurred in both the agile and RE worlds. Thus, for example, techniques have emerged from the RE community for dealing with volatile domains where the requirements can't be fully known before coding begins; sometimes not even before deployment. Similarly techniques have been developed in the agile community for modelling, structuring, and analyzing requirements knowledge.

The aim of the Agile RE workshop is to take stock of the two worldviews to discover whether agile needs RE, and whether novel RE practices can deliver what agile needs.

Workshop Format

We aim to make Agile RE a genuine working event, in which all attendees actively participate during what we hope will be a really stimulating day. We encourage participation from both the agile and the RE communities as well as from non-aligned but interested people, from seasoned practitioners and PhD students, from people with concrete experience to report and people primarily interested in learning. The event will begin with a keynote by Wolfgang Emmerich, Professor of Distributed Computing at University College London and Chairman of Zuhlke Engineering Ltd, an early adopter and one of the thought leaders of Agile Development in the UK.

We will use the paper presentations to draw out themes for discussion; these might be issues needing more research, factors inhibiting industry take-up or just points of mutual benefit to the RE and agile communities. The rest of the day will be spent in interactive, plenary and group discussions of these themes, with the aim of identifying an agenda for further research, technology transfer or for better communication and dissemination.

Papers

We invite submissions of papers describing experiences, challenges, vision, and ideas on the need for, or use of requirements engineering techniques in agile software development projects. Topics of interest include but are not limited to:

- New requirements techniques, methods or tools aimed at supporting agile development

- New agile development techniques, methods or tools incorporating novel requirements handling

- Evaluations of requirements techniques, methods or tools with implications for the support of agile development

- Evaluations of agile development with implications for requirements handling

- Experience reports of requirements handling in agile development

- Position statements on the relationship between RE and agile development

Important Dates

Submission of papers: 15 April 2011

Author notification: 20 May 2011

See website for further details:

www.comp.lancs.ac.uk/~gacitur1/AREW11/index.htm

REET'11

29 August 2011 – Trento, Italy

The 6th International Workshop on Requirements Engineering Education and Training (REET'11) held in conjunction with RE'11 will address issues related to RE education, both as part of a formal university degree and as ongoing skills training within the workplace. The workshop is intended to go much deeper than a surface discussion of curriculum issues and will examine specific ideas and techniques for teaching and assessing skills needed by an effective requirements engineer.

The format of REET'11 will be an interactive one focusing on practical ideas and approaches for teaching RE skills and for constructing effective RE curricular. Paper presentations will be used to foster discussion. Demonstrations of specific teaching techniques and materials will be encouraged. The intent is to involve workshop attendees in performing the activities as well as interactive discussions.

Important Dates

Submission of papers: 6 June 2011

Author notification: 6 July 2011

www.ecs.westminster.ac.uk/REET11

LEAN Methodologies Workshop

August 2011 – Telford

Further details will be published in the next issue. In the meantime check for updates on our website.

RESG AGM and Annual Party

6-9pm, 7 September 2011 - Imperial College London

Sanaz Yeganefard (University of Southampton) will give a short talk after the AGM on the Event-B formalism for structuring requirements.

The keynote talk will be given by RESG co-founder, Prof. Neil Maiden.

Careers in RE

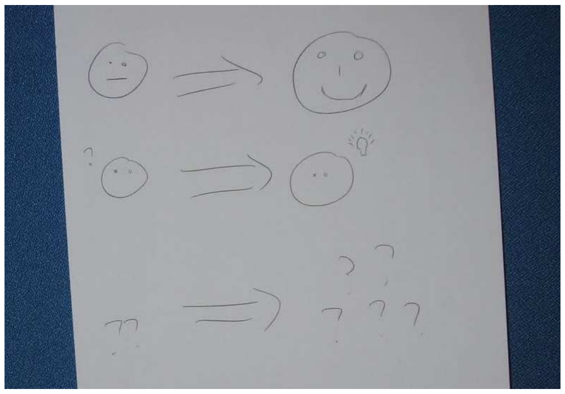
November 2011 - University of Westminster

Further details will be published in the next issue. In the meantime check for updates on our website, or read the report of 2010's event in the following pages.

RE-member

REET Workshop Summary

28 September 2010 – Sydney, Australia



REET'10: "workshop success" picture

Our 5th Requirements Engineering Education and Training (REET) workshop was a huge success! Honestly, this workshop is just fun – of course we can be serious and have smart discussions, but really I feel like we have found a nice blend of keeping it informal and comfortable while also productive. And I truly do mean "we" - all Ljerka and I did was decide which papers to accept and to put together an agenda around those. The essence of a successful workshop is really in the group of participants – their attitude, their desires for the workshop, and their willingness to engage. And once again, all 14 of the 2010 REET participants were just outstanding!

We had originally accepted 7 papers, but 2 had to drop out, so we had 5 that were presented in the workshop. That left us plenty of time to have discussions and interactive activities throughout the day.

At the beginning of the day, we started with a brainstorming activity where the participants worked in groups of 2 to draw pictures (no words!) of what "success" would look like for them at the end of the workshop. That was good fun – got our creative minds collaborating to create pictures to explain it.

After lunch we did a "chair" activity led by **Martin Mahaux**, where in essence we had 1 more chairs than we had people, and Martin played the role of walking around the room trying to sit in the empty chair – the rest of us had the challenge of trying to keep him from sitting in an empty chair by moving around. It was an interesting exercise in how a large group works together – starting with complete chaos and tripping over each other, moving to a bit more of an attempt at organization, but really just a bit of mis-communicating with one another, and lastly we landed in a space that wasn't great, but we were starting to work as a team to work against Martin's role.

Late afternoon I led an activity called "No Value", where you take any topic (we used "business objectives") and you ask students to talk about how you can use that topic to add no value on your project. For example, I could ignore stakeholders and make up the project's business objectives on my own – then they'd add no value. So we did this as a group and then took all the "no value" statements and turned them into positive statements of how to use the topic to add value.

Near the end of the day, **Birgit Penzenstadler** led an activity where we had to use rich pictures only to convey our business user's needs for an automatic refrigerator system. That was funny because we all really wanted to use words in our diagrams – and particularly our engineers in the groups wanted to use more formal models.

And then we had a neat activity where **Anja Wever** asked questions and we all wrote anonymous answers on paper, which she then read. Things like "What was your most embarrassing moment?" She was demonstrating how this is a nice ice break for the right type of group because it lets everyone know that everyone else in the group is just like them - human and personable.

We had very good discussions sprinkled throughout the day as well that were often off-topic from the papers. I had asked participants to send questions ahead of time, so that sparked some of the discussion. The questions that inspired much of our conversation include:

- What's the best way to get a new college grad with no software training up to speed as an RE?
- Are companies willing to spend the money on RE training?
- What do we do in our university programs to ensure that our graduates have the professional attitude of a requirements engineer – and what is that professional attitude?
- How do I change RE training if my company or customers decided to use cloud computing services?
- Does preparation for creativity and collaboration differ according to the region where you come from? How do they value intellectualism in comparison with pragmatism? How well are people prepared for this in your country? How do you prepare well for it?

REET'10 "chair activity"

- How do you excite students/developers about RE?
- How do you make small groups engage better when confronted with very quiet people.
- What are some techniques to prevent course participants from taking long break at the wrong time?
- My company can't afford RE training, what can we do?
- If our CHAOS report results keep showing us that requirements have been the root cause of software failure for so many years, why do we still have so many problems in requirements? Are we getting better?

If all goes well, I hope to see you at REET'11 in Trento, Italy!

© Joy Beatty 2010

Taken from the Seilevel Software Requirements Blog with permission of the author Joy Beatty:

requirements.seilevel.com/blog/2010/09/live-from-re10-reet10-workshop-summary.html

Brownfield Evening, 2010

12 October 2010 – BCS Southampton St., London

This was the second Brownfield event run by the RESG, and we hope, in the nature of all things old and brown, that many more events, blogs, articles, guidelines, templates, podcasts, presentations and, heaven help us, even tweets may join the throng.

But we are rushing ahead of ourselves. Brownfield projects are simply those that are not starting from scratch. In general, only rather small projects are at all likely to start with a clean sheet of paper. Most systems replace old, existing systems. Most products either have to fit into a product line or at least have to interface to existing products and systems.

Having to start from somewhere not of your choosing – very likely, with an obsolescent and creaking system that desperately needs replacing – is inevitably uncomfortable. Common features of brownfield developments include:

- service to continue during upgrades
- many stakeholder interests
- changes are hard and costly
- constrained by existing interfaces
- constrained to look and feel like existing system
- skill and experience are needed to keep the old system running
- existing design elements must be reused
- large investments have already been made

The list of sorrows could be extended, but you'll get the general idea.

I concluded my introduction with a slide about brownfield and greenfield development. There was a big brown circle showing that a rhetorical 95% of projects were brown – building on existing products, systems, services and infrastructure. And then there popped up a big green square, asserting not altogether untruthfully that 95% of requirements textbooks, research and standards were for greenfield development. The audience chuckled knowingly.

Brownfield projects can get unmanageably large, according to Hopkins & Jenkins

The speakers for the evening, Ian Gallagher and Phil Cantor, gamely faced the challenge of saying something coherent about how, in practice, they dealt with enormous brownfield projects.

It is a curiosity of engineering life that, when surveyed, academics felt they were well ahead of practitioners in RE knowledge and skill, while practitioners likewise felt they were well ahead of academics.

Perhaps brownfield work is the reason for this paradoxical pair of beliefs. Perhaps academics see mainly small (student-sized) projects in daily life, or modest (studiable) projects that begin and end within the frame of a PhD or research project. So they develop techniques suitable for such greenfield situations.

Conversely, practitioners see mainly a small corner of a large, confusing, slow, difficult, critical and very long-lived brownfield project. They have to grapple with what Richard Hopkins and Kevin Jenkins call "*Eating the IT Elephant*" (IBM Press, 2008) – the huge grey-brown beast that would certainly stick in the gullet of anyone foolish enough to try. The neat accounts in traditional RE textbooks of how to draw flowcharts and analyze use cases look pitifully small in comparison, and they say nothing about the sheer difficulty of getting swarms of stakeholders to agree, not to mention trying to surmount multiple barriers – legal, commercial, security, and not least financial.

Ian Gallagher of Altran Praxis described in deliberately vague no-names-no-pack-drill terms three large projects – two of them in defence – that he'd worked on. Each represented hundreds of millions of pounds. On the first, textbook RE techniques worked well: it was just a long slog. On the second, the problems of reuse were severe – antique hardware and software that needed extensive replacement, with a mass of interfaces and impossible costs: the project of which the software was a part was eventually ignominiously cancelled in favour of a clean break with the past. On the third, much of the effort went into the socio-technical realm, addressing issues with stakeholders, the law, standards, and other "soft" (but difficult) matters. You can see his slides (and mine) on the RESG website.

Phil Cantor of Smartstream spoke in equally guarded terms (perhaps this too is a pattern in brownfield, the systems that are being discussed are household names and therefore can't be mentioned!) about his work in banking.

On one frozen elephant of a billion-pound project, things had got so bad that basically all the IT budget was locked down into maintaining the status quo: there was scarcely any money or time for innovation or radical change beyond low-level bug-fixing.

Samuel Smiles, by Sir George Reid

What one had to do, argued Cantor, was to break out of the trap, bite the bullet (metaphors fail here) and in short spend enough money to make a real difference.

Samuel Smiles' pioneering book on *Self-Help* (1859) contains a famous description of a poor workman whose trousers were so stiff with grease and patches – he couldn't afford a new pair – that they stood up by themselves. How exactly Smiles managed to get the trousers off the workman for the demonstration, history does not relate.

When code has been hacked for so many years that it consists solely of grease and patches, it becomes "brittle", in programmer-speak. In other words, it gets impossible to alter; if you try sewing in yet another patch, it simply tears somewhere else and you are no better off than before.

When a financial software product has been hacked for over 30 years, things are even worse. It is as if, said Cantor, you had to maintain a tower block, and it was so flaky that you didn't trust any part of it. When you had to fix a toilet, you built 20 storeys' worth of scaffolding, reached in through the window and then down to the floor, and plumbed in a new toilet from there. And then everybody else does the same for every fix for years and years.

In fact, he went on, it's worse than that. Everyone is so terrified of breaking something that every change in the last 30 years has been made with a conditional statement, IF switch-456 is enabled THEN run-this-new-block-of-code. And the old code is basically just commented out but left in place in case the bug-fix patch made things worse. And then you get the typical financial product.

Younger members of the audience paled or turned a delicate shade of green as they tried to imagine such a horrible situation, compared to the crisp lines of Ruby on Rails and PHP that they wrote on their student projects.

The meeting ended with a lively set of group discussions, ably facilitated by Ljerka Beus-Dukic. But the debate on Brownfield RE continues, with contributions on our website

www.resg.org.uk/index.php/Brownfield_RE

and a synthesis of many opinions by Richard Veryard in his excellent blogs at

feeds.feedburner.com/RV4RESG and

rvsoapbox.blogspot.com/2010/09/zero-based-requirements.html

among other places.

How can one deal effectively with brownfield problems? How does it work in your industry? What would you like researchers to do for you? There are spaces for you to contribute your views and to report on your experiences:

- here in the pages of RQ
- on the RESG website
- by joining in with the Requirements Engineering discussion group on LinkedIn

  www.linkedin.com/groups?home=&gid=2662234

We look forward to hearing from you.

© Ian Alexander 2010

Careers in RE

17 November 2010 – University of Westminster

The aim of this RESG event is to bring closer requirements engineering education and industry practice. This year, we had a panel of 5 industrialists with work experience ranging from 4 to 25 years. As usual, the panellists took their turn to tell the audience about their careers and their encounters with requirements engineering.

Rune Stamselberg (Independent Consultant) began his career as a typical software engineer designing, coding, and testing software for medical devices, set top boxes and the like. He recently discovered his previous challenging technical roles were not as challenging as his latest role where he needs to use his 'people' skills more often than his technical knowledge.

Careers in RE panel

Ian Gallagher (Senior Consultant with Altran Praxis) took a positive view on having to deal with a lot of different people from different industry sectors with varied backgrounds. He said that it is the part of his job he enjoys the most as it provides him with an excitement of the unknown and a variety of experiences.

Mary Seddon talked about her current role as a Business Analyst for Taylor & Francis Group supporting IT-enabled business change and transformation which she needs to get through from initiation to going live. She pointed out that her extensive experience in the publishing world and non-technical background helps her to find a common 'application' language with her 'clients' who are in the same publishing business but speak different 'application' languages.

Marko Dukic (Management Consultant and co-founder of several crowd-sourced recommendation websites) started his presentation by showing the advert for his website YourNextRead.com. He contrasted his experience in requirements gathering as a consultant with working with a co-founder and designing for the general public. He emphasised the need to be pragmatic in design, use effective tools and the importance of creating the right narrative.

Jamal Ahmed, University of Westminster graduate, now Pre-Sales Engineer for 29West (provider of messaging middleware) spoke about volatility of requirements and their priorities. He reflected on his requirements engineering experience from a regulated world of safety-critical applications where he completed his industrial placement (BAE Systems) and the agile world of financial services (Lehman Brothers, Namura) where decisions made can be overturned over night.

The event was well attended (students of Requirements Engineering at the University of Westminster were in majority) and the questions to the panellists were about their usage of elicitation techniques (interviews, workshops etc.) and UML (not used as often as one would expect). Noticeably, soft skills (or lack of them) came up in many of the discussions, and a need to acquire soft skills before getting out in industry became apparent.

© Ljerka Beus-Dukic 2010

The Future of Requirements Engineering for Self-Adaptive Systems

2 March 2011 – University College London

The launch 10 years ago of an industry-led "autonomic computing" initiative, in which self-management was proposed as a means to tackle the problem of growing complexity in computing systems, helped spark the following decade's explosion of academic research concerned with the difficulties of engineering systems that were autonomous, dynamic, self-adaptive, self-healing, self-configuring, self-managed (now commonly dubbed self-*), etc. Research projects were proposed and funded, dedicated workshops and conferences became major events. But what broader impact has this clearly fruitful research area had outside academia? Indeed, is academia taking this research in the right direction?

The aim of this discussion-oriented half-day workshop was to consider – from the perspective of requirements – the state-of-the-art vs. the state-of-practice in engineering self-managed systems. In the end, we stirred up less controversy than we'd been expecting, gleaning instead a fascinating survey of cutting edge work in the field and directions for the future. and realising that industry and academia are equally excited about the prospect of this science-fiction future.

But let's step back a bit. What does the term "self-adaptive system" mean to you? A practical approach to many of the engineering problems of modern software systems or a flight of fancy? There is no denying that the term is a hot new buzzword, yet in some it engenders an excited sense of expanded possibility and in others an entrenched scepticism – isn't this just configurable interfaces under another name? Even among enthusiasts there's a sense that we haven't yet found sufficiently compelling cases studies – or the killer app – to really motivate work in this area. We are left with this exciting concept and the sense that this may indeed be the future of software engineering – the foundation to building robots and pilotless aircraft and a self-healing internet – but we're not 100 percent sure how to proceed.

It doesn't help that there isn't really a consensus on what a self-adaptive system is, even among those working in the field. I've participated in workshops where people have argued that washing machines are autonomous because you load them up, turn them on, and walk away. On the other hand, I've heard a similarly extreme argument that we should expect an autonomous system, once switched on, to turn its back on us and disappear about its own business – making the idea that we can give an autonomous system tasks an absurdity.

So it wasn't surprising that most of the speakers in this workshop took pains to define their idea of a self-adaptive system. **Emmanuel Letier** chaired the afternoon's discussion and set the tone in his introduction by asking whether research is addressing the right problems and what the implications of self-adaptive systems are for requirements engineering.

Our keynote speaker was **Darren Ansell**, technical manager at BAE Systems for the ASTRAEA programme, which is a UK industry-led consortium focusing on the technologies, systems, facilities, procedures and regulations that will allow autonomous vehicles to operate safely and routinely in civil airspace over the United Kingdom. It was fascinating to hear about the concerns of the aviation industry in its early investigation of this cutting edge technology. Perhaps unsurprisingly, Darren had quite a lot to say about the need to verify the safety of systems – especially civilian aircraft – that had been designed to adapt in potentially unforeseen ways, and the difficulties of doing this. But interestingly, Darren also encouraged us to bear in mind that self-adaptation could turn out to be a mechanism for making systems more, rather than less, safe. In situations characterised by long periods of inactivity punctuated by sudden bursts of activity demanding quick reflexes and immediate decision making, human operators may not be ideal agents. By giving over more agency to the software we can avoid human error brought on by drowsiness, for instance. The recent revelations reported by the media that a large percentage of airline pilots claim to have felt drowsy and even fallen asleep while flying underlines this observation.

However, the aviation industry will be approaching the adoption of such technologies in small steps. Picking up on the idea that autonomy comes in several strengths, Darren explained how his industry considers there to be multiple degrees of self-management. At the top end of the scale we have fully autonomous systems, acting independently of any operator whatsoever. Below that we find systems that act autonomously but inform human operators about what they're doing, with the further option of allowing operator override. Then come systems that make their own decision about how to behave but seek permission before doing so. Finally, at the lower end of the scale we have remote controlled systems, many examples of which are currently in use, such as the military drones flying over swathes of the Middle East.

With the scene set by Darren's overview of the state-of-practice in industry, the rest of the afternoon saw the invited academic speakers present their views of the challenges to requirements engineering brought by this technology. Perhaps typically for blue-sky thinkers, most of the academics were interested in the challenges of building systems that displayed near full autonomy. There isn't space for a full summary of each talk but I will pick some key points to give the flavour of the field.

Pete Sawyer of Lancaster University talked about the problems of *uncertainty* in designing self-managing systems. He distinguished two kinds of uncertainty that were relevant: firstly, since a self-managed system is designed to adapt to unforeseen circumstances its designers will be uncertain what form or behaviour the system may exhibit for all points in the future. Secondly, we must consider uncertainty from the system's point of view: since self-management presupposes that the system is able to work out how to proceed by itself, we must consider what happens when the system is uncertain about its situation. In short, when dealing with self-managed systems we must consider both the designer's uncertainty – of the situations the system will encounter – and the system's uncertainty – of the situation it has in fact encountered. Having noted the distinction, Pete discussed his work on extending requirements modelling techniques to more precisely capture the former notion, from the designer's point of view.

Mazeiar Salehie, from Lero: The Irish Software Engineering Research Centre, shared some early thoughts on a new research project looking at security and privacy issues for self-managed systems. Again, the question was thrown up whether such systems present a genuinely different way to approach system design and thus require the solving of new problems. We should watch Mazeiar's future work for his conclusions.

Yijun Yu talked about how requirements models can be used at runtime to provide the basis for system monitoring. Yijun pointed out the benefits gained when requirements are first-class entities at runtime, noting how they could (and should) be used by a self-managed system when adapting itself to maintain satisfaction of goals – relating, for example, to performance. Yijun also made the observation that this kind of requirements-driven self-tuning may be particularly useful in the web-based service-oriented computing architectures of the cloud, where the inter-dependencies of federated service providers make it especially difficult for human operators to diagnose and fix outages. Indeed, Yijun was one of two speakers to cite the enormous $50k lost by Amazon for each minute of the three-hour downtime of its cloud services in 2008. Investigation suggested that most of the three hours was spent tracking down the point of failure. Several people at the workshop suspected that a self-managed and self-monitoring architecture might have been able to fix the problem more quickly. At the very least, the potential losses that can be incurred make further research into these proposed solutions financially viable.

The last two talks of the afternoon focussed on practical attempts to implement a fully self-managed system. **Nicolas D'Ippolito**, jointly from the University of Buenos Aires and Imperial College London, spoke about one of the key technical challenges of implementing self-management. If a system is truly to decide itself how best to achieve a set of given goals it must be able to generate a controller – specifying how to behave – automatically. However, designing a system that can do such a thing is no simple task. What we want is to generate a controller that, if executed, guarantees satisfaction of the system's given goals. But such a controller must allow for the system's environment behaving unexpectedly – it's even helpful, when seeking guarantees, to give a face to the environment and think of it as behaving maliciously, keen to thwart the system's attempts. Part of the technical problem here is to accurately encode the often implicit assumptions that system designers make of the environment into a runtime model for the system to use in generating its controller. Nicolas suggested that it's helpful to think of the interplay between software and its environment in game-theoretical terms, where the software needs to be able to follow a strategy – set out in the controller – that can ensure success even as the environment is doing its best to win. Nicolas illustrated his more arcane examples with a winning recipe for cooking biscotti, and when questioned about some of the finer details of his example by Ljerka Beus-Dukic he allayed all doubts by assuring us he'd first consulted his mother.

Rounding off the workshop, and picking up where Nicolas left off, I demonstrated in my talk how his techniques for automatically generating behaviour specifications from system goals could be placed within a runtime feedback loop that integrates the monitoring of goal satisfaction with self-adaptation mechanisms. I showed videos of some of the experiments (carried out with colleagues at Imperial College London) using a six-wheeled robot the size of a piece of hand luggage and a disembodied robotic arm with pincer digits. These experiments typically involved setting the robots a task – such as collecting a bunch of coloured balls – and then changing the resources available to the robots – such as disabling a navigation sensor or camera – while they were in the middle of carrying the task out. What we were interested to see was whether we could design a software architecture that allowed the robots to reconfigure their resources – their available components – in such a way that allowed them to go on to carry out the task assigned – to succeed in satisfying a set of goals in the face of changing assumptions about the environment. For the relatively simple scenarios we explored we found that they

Visions of the future ...

could. In other words, given the means to achieve a task, the robots could work out how best to go about achieving it themselves, updating their plans on the fly as the environment changed: from high-level goals to task execution in a fully automated loop.

However, for this technology to be generally applicable we need systems that operate well outside of lab environments. It's one thing for a robot to self-adapt in a relatively closed environment and quite another for it to work out what it needs to do given all the possible things that might go wrong in a real-world setting.

One potential solution might lie in designing systems that are able to learn by making mistakes, proceeding by means of a trial and error approach. However, this would require us to fundamentally reassess a lot of our traditional design methodologies and engineering principles – what kinds of mistake would be allowable, for example?

Key to this is an observation made by a member of the audience, just as the day was ending: in our thinking about systems – and thus their modelling – there is usually a large gap between “value” perceived at the control level and “value” perceived at the human level. In other words, we need good, generally applicable ways to bridge the conceptual gap between a failure at the system control level – a violated low-level requirement – and the consequent failure at the “human” goal level. A self-managed system needs to be able to self-diagnose before it can self-heal. A clear relation between these different value levels seems necessary for automatic analysis to be possible.

We may not have resolved all the workshop's original questions, but there was little doubt among the audience that this is one of the most exciting and challenging areas in software engineering today. Before those planes start taking off without us, research needs to catch up with the current drive of industry.

RE-flections

Sky-High Goals, Lower Costs

In the December 2010 issue of *Scientific American* magazine (www.scientificamerican.com), David Freedman discusses "*Jump-starting the orbital economy*" – turning space travel from a costly government-financed research venture into a profitable business.

Unmanned geostationary satellites have already made that leap, of course: that strange orbit at 35,800 kilometres up, around the equator, is thickly populated with squat but profitable boxes of communications equipment, doing everything from monitoring the weather to providing television to far-flung islands.

Manned space flight is another matter. NASA launches its last space shuttle this summer: after that, the funding, like the enthusiasm for costly adventures with no sight of a financial return, dries up. President Obama has cancelled NASA's Constellation launcher, the costly agency-funded replacement for the shuttle. According to Freedman, most of the $9 billion of R&D effort so far spent will be written off.

But NASA is not giving up the ghost – or should I say the spirit of adventure – just yet. Instead, the space agency has decided to pass the mantle of invention to private companies. They are expected to come up with innovative ideas; in return, NASA will provide them with seed-corn funding to help them get started – and after that, if their ideas work, NASA will provide them with a guaranteed customer. Just the thing to help you get unit costs down as you ramp up production.

And costs will really have to come down if space travel is ever "to boldly go" anywhere.

According to the NASA website (www.nasa.gov), a shuttle mission costs about $450 million; Wikipedia estimates the true cost at more like $1.5 billion per flight. The number of crew varies, but for 7 crew and in round numbers that's an eye-watering $200 million per person.

Could "a small private operation" do better than "armies of engineers, technicians and managers backed by billions in funding and decades-long development cycles"?

From Cost-plus to cost efficiency

In other words, could a leaner, more agile approach possibly work for the costliest, riskiest and highest frontier of them all?

SpaceX's Falcon 9 launcher has reached orbit. It could one day carry a crew vehicle (Wikipedia)

NASA has always relied "on the commercial sector to build spacecraft". But "what will change under the plan is the *way* NASA will work with private firms." Like the Pentagon, "NASA hires contractors on a 'cost-plus' basis, which means NASA reimburses them for whatever they spend and then tosses in a guaranteed profit" – not exactly a formula for economy, especially as every change means more profit for the contractor, and a warm cosy feeling of increased safety for the agency.

Under Obama's plan, budgets are fixed. Over budget, you make a loss; under budget, you make a profit. A penny saved is a penny earned. Sounds familiar? Well, it's a big change for aerospace contractors.

Even so, more than 10 companies are hoping to launch people into space. The hopefuls include Orbital Science Corporation with its Taurus 2 launcher; SpaceX's Falcon 9; and a variant Delta rocket built by United Launch Alliance (Boeing and Lockheed Martin); not to mention "joyrides to suborbit" from Xcor, Virgin Galactic, Blue Origin or Armadillo Aerospace.

With older technology, United Launch Alliance already puts payloads into orbit for $100 million, so “the only question is whether [such giant companies] could build them under fixed-price rules and cost-cutting pressures”.

SpaceX hopes to charge a quarter of that, just $25 million per launch. That is comparable to what the very richest adventurers are willing to pay. “Since 2001 Russia has flown seven tourists to the space station – one of them flew twice – via the Soyuz [launcher] at prices ranging between $30 million and $50 million. At significantly lower prices, the number of takers would climb”, observes Freedman.

Lon Levin, president of SkySevenVentures, remarks that “If the price were to head down toward $1 million, might hundreds of people buy a ticket? It’s possible, and that could make it a real business.”

So far, SpaceX’s successful rocket tests are the only concrete “evidence so far that private industry might succeed.” It’s promising, but still more than a bit worrying.

From ‘Exactly How’ to ‘What’

But according to Freedman, “to encourage this type of innovation, NASA has to let go. The agency has always told its contractors ***exactly how*** it wants its space vehicles built, yet under [Obama’s] new plan NASA would ***simply state what*** it wants a finished system to be able to do, such as safely ferrying a certain amount of weight into orbit.

“We won’t be overly prescriptive in how we expect contractors to meet our requirements, we’ll ***just list high-level goals*** and give them maximum flexibility for how to meet those”, says Phil McAlister of NASA, unconscious of any possible pun on the word “high”.

“Then at specific milestones we’ll be verifying **that the requirements have been met**, and we’ll provide whatever oversight is necessary to make sure.”

Could this hands-off, goal-oriented approach to RE do what fifty years of manned space flight have failed to do? It’s a tall order (sorry). But given people willing to take risks with life and limb – as well as deep pockets – who is to say it isn’t possible. As for whether it’s fair on people and planet for the wealthy to behave like that, well, that’s another matter entirely.

© Ian Alexander 2010

RE-verberations

HFS: Human Flesh Search

For a system to produce purposive results, it has to have been specified, designed, and tested against its requirements; it has to be commissioned and deployed prior to use; and it inevitably stays the same until it is deliberately changed, whether by redesign, “maintenance” software coding, or the updating of business rules if table-driven.

Only – all these laws are false, in the case of systems built on social networks and making use of the combined power of the Internet and human capabilities.

The International Council on Systems Engineering, INCOSE, has of course long defined a system as “a construct or collection of different elements that together produce results not obtainable by the elements alone. The elements, or parts, can include people, hardware, software, facilities, policies and documents...” (Eberhardt Rechtin, The Art of Systems Architecting, 2nd Ed., 2000).

It should be no surprise that the “parts” of an aircraft, or a bank, include both people and policies as well as hardware and software: these systems couldn’t work without rules to govern them and people to maintain them.

But these sober, Systems Engineering thoughts pale into insignificance beside the lurid accounts of Human Flesh Search that have appeared recently in the Western press. Western, for HFS stems from China – perhaps the first of many significant innovations from that dynamic economy and its many scientists and engineers.

HFS is an almost wilfully over-literal “translation” of the Chinese root:

人肉搜索

which means something like People-Powered Search (you may recognise the first character as “Human” if like me you were ever introduced to Chinese script – and writing with a paintbrush – on a school Open Day).

Or, as Fei-Yue Wang at the Chinese Academy of Sciences and colleagues in the August 2010 issue of IEEE Computer explains, HFS refers to “searches conducted with help from human users”, “often targeted at finding the identity of a human being”.

This prosaic meaning didn’t stop the media from taking the shortest path to a juicy story. “Many erroneous notions have appeared on various blogs, on wiki sites and in media reporting”, comment Wang et al.

Among the dodgy definitions of Human Flesh Search published in 2008 when the story broke in the West, SearchEngineWatch came up with:

"finding and punishing people who publish material Web users consider inappropriate."

The Murdoch site Times Online offered:

"digital witch hunts."

And on a site which really should have known better, a guardian.co.uk blog entry in Comment Is Free described HFS as:

"an internet mob that hunts down real people online, then verbally abuses them and publishes the victim's private information."

Wang et al comment drily "Chinese-based sources offer broader definitions." ChinaSupertrends.com defines HFS as:

"online crowds gathering via China's bulletin board systems, chat rooms and instant messaging to collaborate on a common task."

That task can be anything from fighting corruption, as in the famous "Outrageously-priced haircut" case, to uncovering fraudulent claims, as in the equally famous "South China Tiger" HFS case. And, yes, there have been some tragic cases, such as the lovelorn youth who claimed he was dying and needed to get in touch with his lost girlfriend – and then killed her when the HFS tracked her down for him. But hard cases make bad law.

The haircut in question came from a barber's shop in Zhengzhou in Henan Province that charged up to 200 times the normal price, responding to complaints by asserting that it had "deep connections (presumably with the local government and the police)". The local people responded effectively by using the Internet to identify who these deep connections were and to organize demonstrations in front of the barber's shop: they also vandalized it, so there is some truth to some of the claims about HFS. However the case ended positively with official punishment of the barber's shop management.

The South China Tiger case involved a hunter from Shaanxi province who claimed to have photographed the animal in question, which was believed extinct in the wild. The claim triggered an enormous amount of web activity; the photo appeared in Science magazine ("Rare-Tiger Photo Flap Makes Fur Fly in China"); and successive groups from the general public to professional photographers investigated and analysed the claim in minute detail. This led to the discovery of "the original calendar cover painting" from which the photograph had been created and edited (i.e. definitely fraudulently photoshopped).

The South China Tiger case is analysed by Wang and colleagues. They show that attention moved spontaneously from a single thread on the general site 163.com (with many entries and very many casual participants) to the professional photographers' forum xitek.com, which cast serious doubt on the purported photograph. Attention then moved to tianya.cn, mop.com and sina.com.cn and half-a-dozen others which were less heavily involved. It took just over a month (from 12 October 2007 to 16 November 2007) from publicity being given to the photo to the discovery of the original calendar image.

The HFS systems in these and other cases were by no means all the same: they involved different groups of people, working with different types of search engine and different human "search" skills – humans are after all creative system "parts" and can fill in arbitrary gaps in machine capabilities, given sufficient time and motivation. HFS systems arise spontaneously, evolve dynamically in response to perceived problem needs – like finding someone or something, or applying photogrammetric skill, or searching official records, or applying pressure to local government.

The networks have unique properties, different for instance from blog sites or search engines. Both In- and Out-degree distributions follow power laws: "a few participants generated most of the citations, and a few participants got cited in these citations. We also found that out-degree dropped off much more rapidly than in-degree ... fewer nodes that cited many other nodes than the nodes that received citations from a large number of other nodes."

There were also (very) "few participants who appeared on more than one discussion forum. Although small in number, these nodes played a pivotal role in transferring and sharing information across discussion forums and often across HFS engines."

In other words, an effective HFS system, tailored precisely to track down the South China Tiger, was formed by a few existing forums, several large groups of people with diverse skills (publicity and blogging; photography; the energy to search offline for tiger images...); and a few individual people who had suitable combinations of interests and hence could act as bridges between separate online systems.

With these "parts", exactly the right system could form, evolve, set goals, acquire resources, and reach an accurate conclusion, rapidly and without either explicit financial investment or (ahem) requirements.

© Ian Alexander 2010

RE-writings

Managing Complexity in Large Develop-ment Programmes

Article 3: Conformance to Standards, Multiple Lifecycles, and Whole Lifecycle Management

This is the last of three articles on Managing Complexity in Large Development Programmes by Steve Rivkin. The first article in RQ54 described some of the problems that can occur when development programmes are undertaken with inadequate sets of requirements, and/or with no linkage to the designs that are produced during the lifecycle. The second article in RQ55 described a Requirements-driven Design approach, in which the DOORS Database was used to manage the requirements, providing full traceability and verification. This final article in the series will consider some of the additional issues related to large, complex development programmes, such as: conformance to standards, multiple concurrent lifecycles, the management of legal and non-technical requirements, and whole lifecycle management.will show how these problems can be avoided by using a rigorous 'Requirements-driven' approach to evolve the Requirements, and link them to the corresponding designs, providing verification and full traceability back to the Stakeholders' Requirements. The final article in the series will consider conformance to standards, complex lifecycles, and the management of legal and non-technical requirements.

Conformance to Standards

In a European, or global market, it is essential to conform to accepted international standards. A generic, conformant design can be created which can be easily adapted with a degree of localisation for each specific country.

Illustration of a Lifecycle for a Standard

There are many standards which are applicable to the whole lifecycle of a development project. Several key examples are described below.

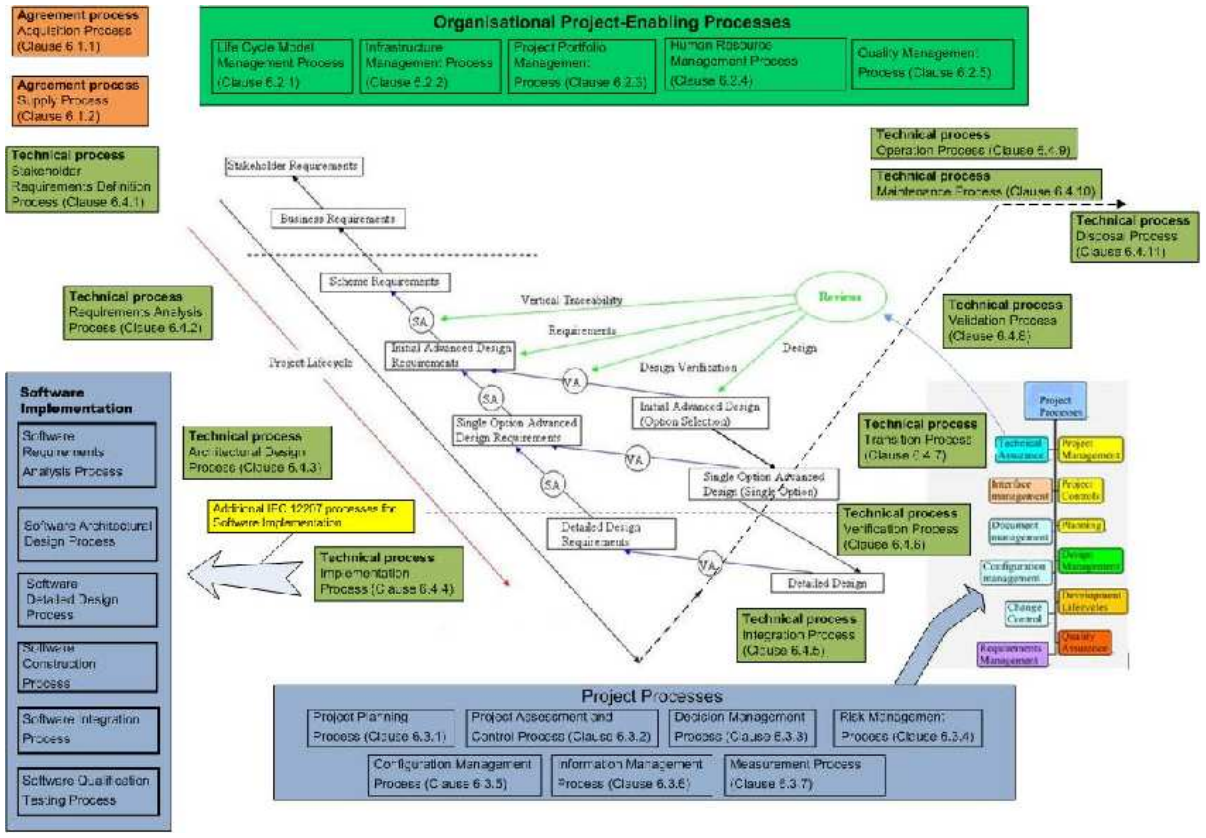

Agreement process
Acquisition Process
(Clause 6.1.1)
Agreement process
Supply Process
(Clause 6.1.2)
Organisational Project-Enabling Processes
Life Cycle Model Management Process (Clause 6.2.1)
Infrastructure Management Process (Clause 6.2.2)
Project Portfolio Management Process (Clause 6.2.3)
Human Resource Management Process (Clause 6.2.4)
Quality Management Process (Clause 6.2.5)
Technical process
Stakeholder Requirements Definition Process (Clause 6.4.1)
Technical process
Operation Process (Clause 6.4.9)
Technical process
Maintenance Process (Clause 6.4.10)
Technical process
Disposal Process
(Clause 6.4.11)
Stakeholder Requirements
Business Requirements
Scheme Requirements
Vertical Traceability
Requirements
Review
Design
Design Verification
Technical process
Requirements Analysis Process (Clause 6.4.2)
Technical process
Validation Process
(Clause 6.4.8)
Project Lifecycle
SA
Initial Advanced Design Requirements
VA
Initial Advanced Design (Option Selection)
Single Option Advanced Design Requirements
Single Option Advanced Design (Single Option)
Software Implementation
Software Requirements Analysis Process
Software Architectural Design Process
Software Detailed Design Process
Software Construction Process
Software Integration Process
Software Qualification Testing Process
Technical process
Architectural Design Process (Clause 6.4.3)
Additional IEC 12207 processes for Software Implementation
Technical process
Implementation Process (Clause 6.4.4)
Detailed Design Requirements
Detailed Design
Technical process
Transition Process
(Clause 6.4.7)
Technical process
Verification Process
(Clause 6.4.6)
Technical process
Integration Process
(Clause 6.4.5)
Project Processes
Technical Assurance
Project Management
Interface management
Project Controls
Document management
Planning
Configuration management
Design Management
Change Control
Development Lifecycles
Requirements Management
Quality Assurance
Project Processes
Project Planning Process (Clause 6.3.1)
Project Assessment and Control Process (Clause 6.3.2)
Decision Management Process (Clause 6.3.3)
Risk Management Process (Clause 6.3.4)
Configuration Management Process (Clause 6.3.5)
Information Management Process (Clause 6.3.6)
Measurement Process (Clause 6.3.7)

Figure 1. ISO/IEC 15288 and ISO/IEC 12207 processes mapped according to their occurrence in their lifecycle

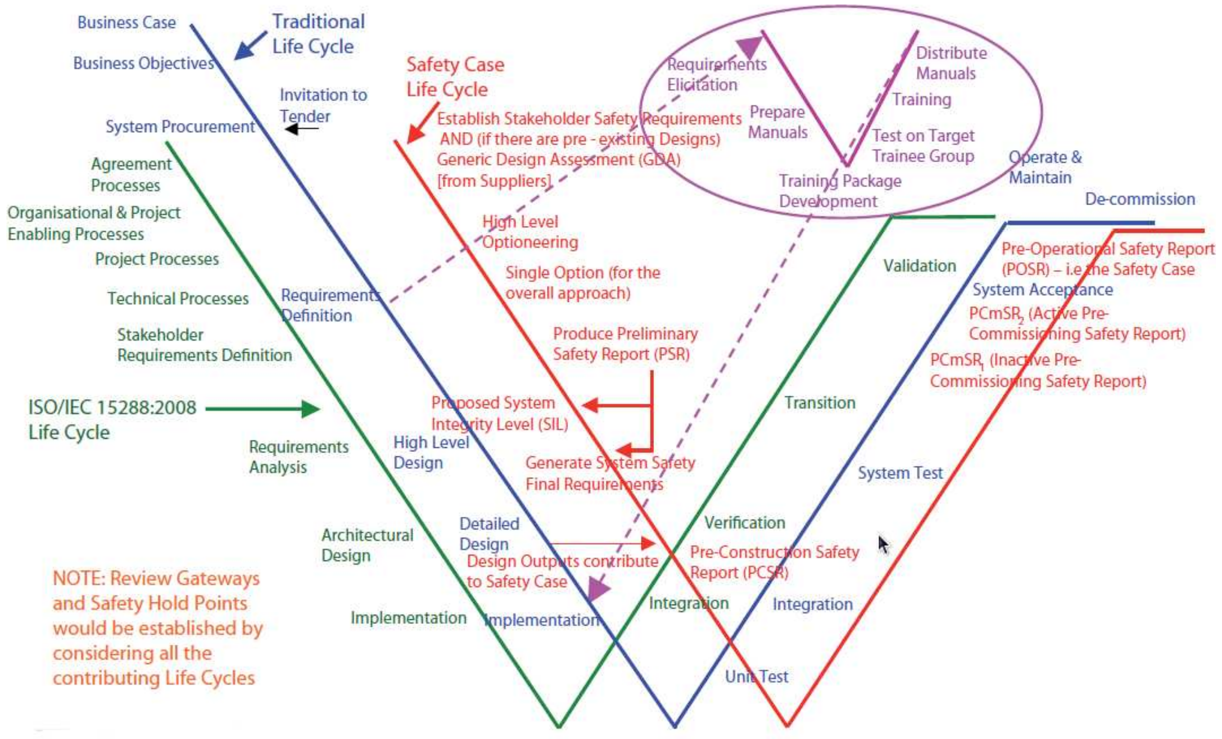

Business Case
Traditional Life Cycle
Business Objectives
Invitation to Tender
System Procurement
Safety Case Life Cycle
Establish Stakeholder Safety Requirements AND (if there are pre - existing Designs) Generic Design Assessment (GDA) [from Suppliers]
Requirements Elicitation
Prepare Manuals
Training Package Development
Test on Target Trainee Group
Training
Distribute Manuals
Operate & Maintain
De-commission
Agreement Processes
Organisational & Project Enabling Processes
Project Processes
Technical Processes
Stakeholder Requirements Definition
Requirements Definition
High Level Optioneering
Single Option (for the overall approach)
Produce Preliminary Safety Report (PSR)
Validation
Pre-Operational Safety Report (POSR) – i.e. the Safety Case
System Acceptance
PCmSR2 (Active Pre-Commissioning Safety Report)
PCmSR1 (Inactive Pre-Commissioning Safety Report)
ISO/IEC 15288:2008 Life Cycle
Requirements Analysis
High Level Design
Proposed System Integrity Level (SIL)
Generate System Safety Final Requirements
Transition
System Test
Architectural Design
Detailed Design
Design Outputs contribute to Safety Case
Verification
Pre-Construction Safety Report (PCSR)
NOTE: Review Gateways and Safety Hold Points would be established by considering all the contributing Life Cycles
Implementation
Implementation
Integration
Integration
Unit Test

Figure 2. Multiple V-Models

ISO/IEC15288 is a standard that contains processes to support a complete V-Model lifecycle, and is used here as an illustration of how such a standard operates over a lifecycle. The standard is concerned with those systems that are man-made and may be configured with one or more of the following: hardware, software, data, humans, processes (e.g. processes for providing service to users), procedures (e.g., operator instructions), facilities, materials and naturally occurring entities.

ISO/IEC 15288 establishes a common framework for describing the lifecycle of systems created by humans. It defines a set of processes and associated terminology. These processes can be applied at any level in the hierarchy of a system's structure. Selected sets of these processes can be applied throughout the lifecycle for managing and performing the stages of a system's lifecycle. This is accomplished through the involvement of all interested parties, with the ultimate goal of achieving customer satisfaction. This standard also provides processes that support the definition, control and improvement of the lifecycle processes used within an organization or a project. Organizations and projects can use these lifecycle processes when acquiring and supplying systems.

When a system element is software, the software lifecycle processes documented in ISO/IEC 12207 may be used to implement that system element. ISO/IEC 15288:2008 and ISO/IEC 12207:2008 are harmonized for concurrent use on a single project or in a single organization. Figure 1 shows a V-Model, and illustrates the Requirements-driven Design approach, with levels of requirements linked to corresponding levels of designs. The processes of Standards ISO/IEC 15288 and ISO/IEC 12207 are shown mapped onto the lifecycle according to how they would be used to support the lifecycle. The software processes are shown in blue, and the software lifecycle would normally start once the main subsystems have been identified and the appropriate level of functional definition and design has been reached in the main system lifecycle.

Mechanics of Conformance

The mechanics of achieving conformance in the Requirements-driven approach is by the use of Compliance Arguments[1] (CAs) to merge in the requirements of a standard at the appropriate stages of the development lifecycle. Requirements are created at the various stages of the development lifecycle, specifying what is required by the standard at each specific stage. Each CA references the clause in the standard that has necessitated the creation of the

[1] CAs are described in the second article of this series, in the April 2010 issue of Project Manager Today

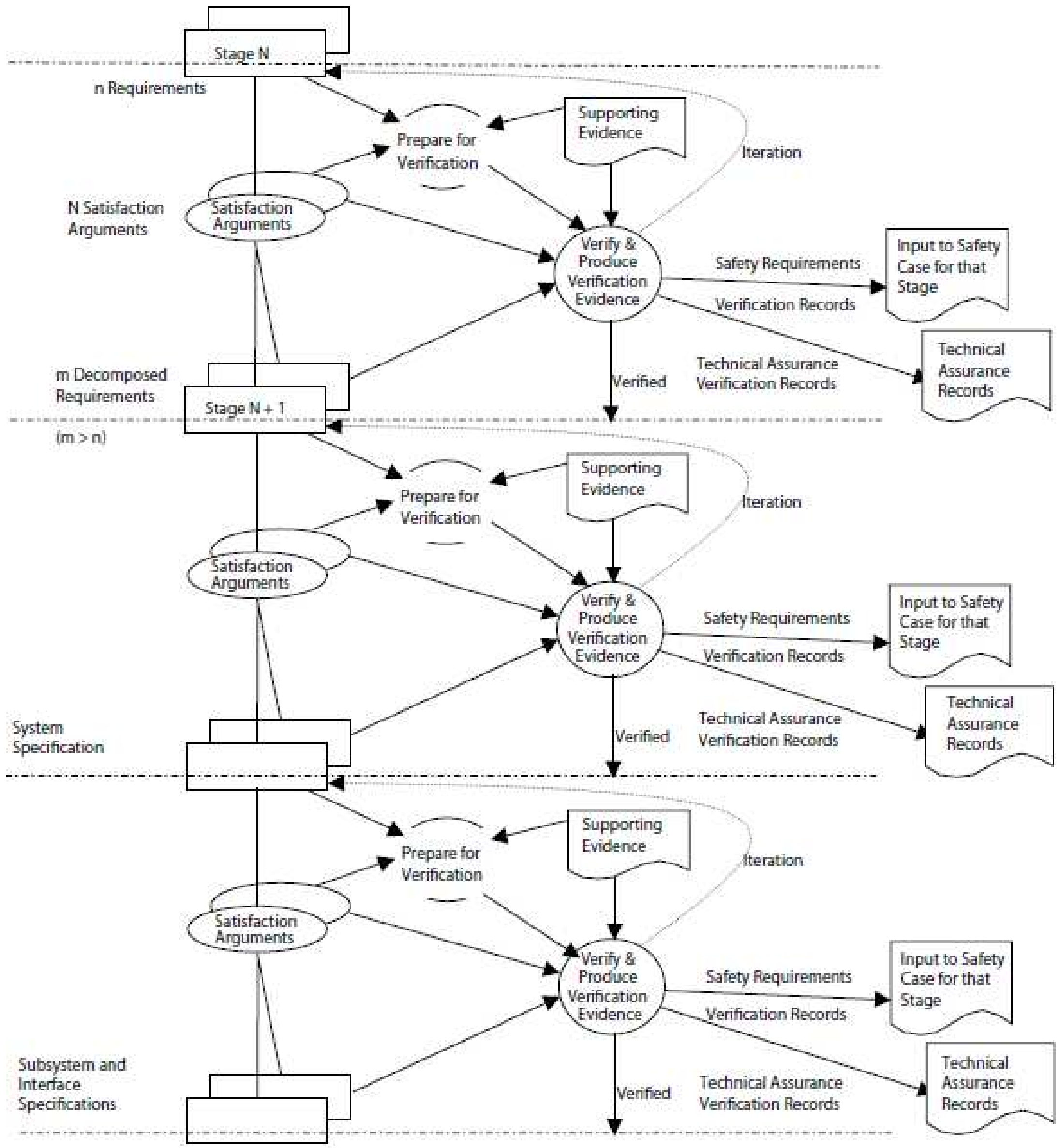


Figure 3. Incremental Contribution to the Safety Case

corresponding requirement at that stage in the lifecycle. The argument in the CA states why the requirement is necessary, and how it addresses the clause in the standard.

Multiple Lifecycles

Where a standard operates throughout a lifecycle, it runs in parallel with the traditional development lifecycle. In development programmes that need to comply with safety regulations, a further Safety Case lifecycle also operates in parallel. An example of this is shown in Figure 2 where the ISO/IEC 15288 lifecycle, a traditional V-Model development lifecycle, and the UK Safety Case lifecycle for the nuclear industry, operate in parallel. Managing three, or more, separate lifecycles that operate in parallel would be extremely complex and error prone. The situation can be made more manageable by mapping all the lifecycles onto the traditional lifecycle, as this is the most widely understood. The gateways for each lifecycle can be synchronised with those of the traditional lifecycle, and the contents/deliverables of the gateways for the additional lifecycles can be mapped onto the corresponding gateways of the traditional lifecycle.

Other additional lifecycles may occur during the main project lifecycle. A Training Package is shown in figure 2 as a ‘mini V-Model’. This may perhaps be subcontracted out, and it starts during the ‘Requirements Definition’ stage of the traditional lifecycle, and is delivered during the ‘Implementation’ stage. The Training Package may be developed using a V Model, or some other lifecycle which is appropriate to the specific nature of the development.

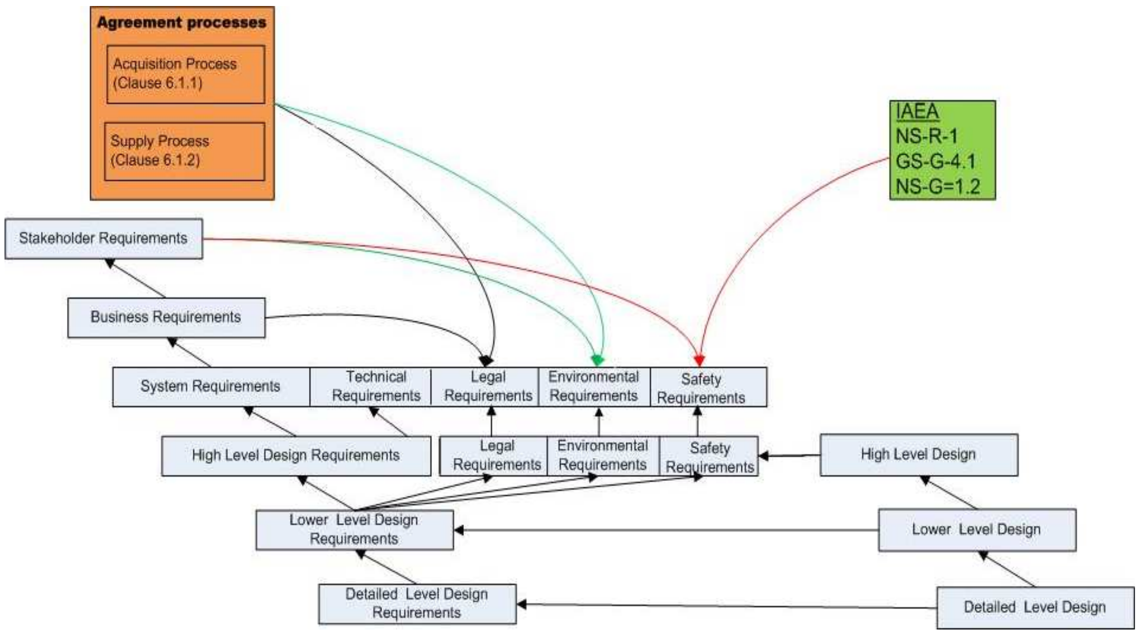


Figure 4. Merging non-technical requirements into the

A number of sectors in industry use specialist lifecycles that are specific to their sector. For example, in the rail industry in the UK, Network Rail uses the GRIP (Guide to Railway Investment Projects) lifecycle, which is an eight-stage process designed to minimise and mitigate risks. Where buildings are required, the RIBA (Royal Institute of British Architects) **Outline Plan of Work** may be used to help explain the different stages involved in the development of a building. It organises the process of managing, and designing building projects and administering building contracts into a number of key Work Stages. In sectors that are concerned with production facilities ISO 15926 is used to facilitate the integration of data from various sources to support the lifecycle activities and processes of production. In development programmes that require specialist approaches to meet safety requirements, and where a high SIL (Safety Integrity Level)[2] is required, a number of standards can be incorporated into the main lifecycle, such as EN 50128 for the rail industry, IEC 61508/61513 for the nuclear industry, and IEC 61508/61511 for plant control in the refineries, petrochemical, chemical, pharmaceutical, pulp and paper, and power industries.

These specialist lifecycles can be merged into the traditional lifecycle programme and gateways, as described above for ISO/IEC 15288. Using Requirements-driven development provides full verification and traceability. As each stage of the lifecycle is completed, the completed sets of requirements, and where appropriate, the designs, together with the corresponding verification evidence make an Incremental contribution to the Safety Case (Figure 3).

Managing Commercial Requirements

The Requirements-driven approach is capable of supporting both the technical and the non-technical requirements for a system. Continuing with the example of ISO/IEC 15228, figure 4 shows several sets of non-technical requirements (e.g. legal and environmental) that are relatively high up in the requirements hierarchy. The legal requirements are shown as having been generated from the Business Requirements and the Agreement Processes of ISO/IEC 15228, and are requirements that cover contractual arrangements and 3rd party agreements. Similarly, the Environmental Requirements have been derived from the Stakeholder Requirements and the Agreement processes of ISO/IEC 15228, and the Safety Requirements have been derived from the Stakeholder Requirements and IAEA (International Atomic Energy Agency) documentation. These are Requirements which may be binding contractually, and must move from being purely documented requirements to becoming part of the implemented system that meets these and all the other requirements that have been produced during the project lifecycle.

The project's legal advisors will be involved with formulating the various contracts and 3rd party agreements. Working with the Requirements Manager the legal team can provide the contractual and 3rd party agreement information that enables the legal requirements to be captured within the requirements hierarchy in the DOORS Database. Without this

[2] There are four SIL categories, SIL 1 to SIL 4, where SIL 4, where SIL 4 has the most stringent requirements. If a system is designated SIL 0, there are no special safety measures required.

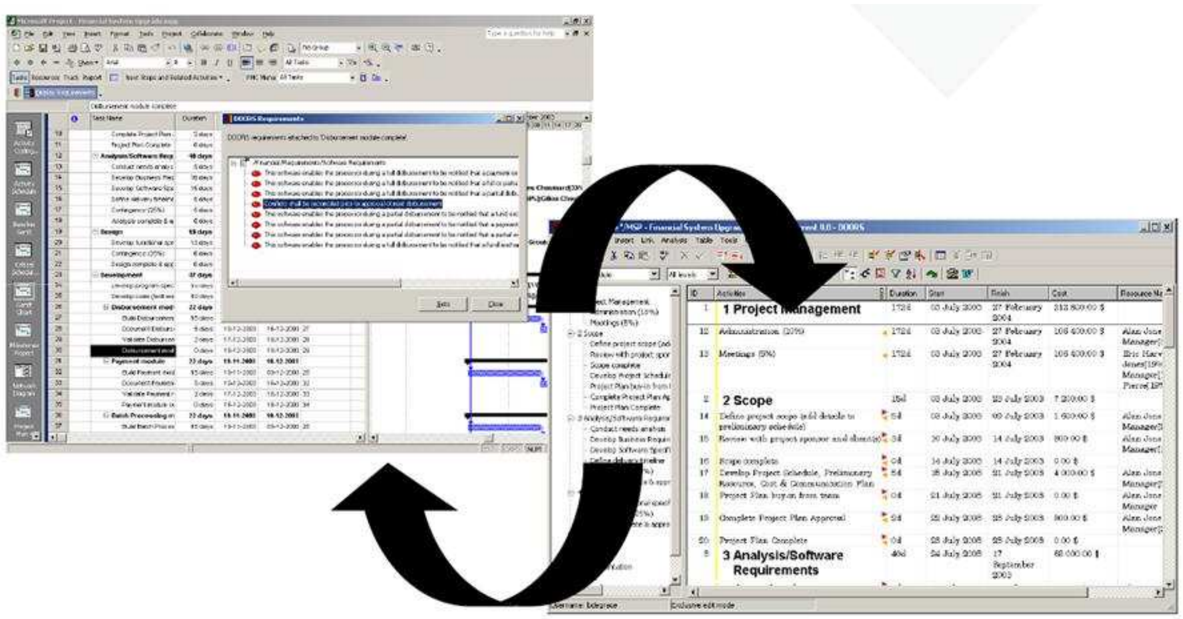


Figure 5. DOORS View and Project Planning View where Projects are Stored in Both Databases

capture, there is a risk that some of the legal obligations of the project may be overlooked during the design and implementation stages of the project, since there is normally little or no communication between the legal teams and the development teams. The contracts and 3rd party agreements solely constitute legal documents, but in system terms they require that some resulting activity must take place to satisfy the legal obligation.

Under normal circumstances, it is more efficient if generic designs are produced for the various elements of a system. Figure 4 shows how lower down in the requirements hierarchy, the Legal, Environmental and Safety Requirements merge into the Technical Requirements stream for implementation in the design. However some of these Requirements, which may have been derived from non-technical sources, will require non-generic implementation. This ensures that all the legal obligations are satisfied by the project. As all elements of the designs, and all requirements, are traceable in DOORS, the sources of requirements, and the reasons why they are necessary, can be determined easily.

Managing Multiple Development Contracts

In the second article of this series the importance was stressed of defining early in the project lifecycle the key project support processes that form part of the supporting project infrastructure. Typically such processes are documented as numerous sets of textual procedures, which are accessed as single documents only as and when they are required. In such a scenario it is difficult to see the context in which the procedure fits and its relationship to other procedures. Currently, there are numerous graphical applications that allow process models to be captured as sets of connected graphical nodes. One of the key elements for success in any project is communication. Using graphical applications such as these the behaviour and inter-relationships of the key project processes can be communicated graphically, and they can be easily assimilated by the project teams. Where necessary, the text version of a procedure can be appended to the process diagram using hyperlinks. Also, it is possible to link through to the DOORS database from a node in some graphical applications (such as MindManager[3]), so that once the operation of a procedure is understood, subsequent actions can be performed directly in DOORS, allowing managers of the infrastructure processes to update data in DOORS relating to their process. An example of this type of approach can be found by examining the APM's graphical MindManager documentation for the PRINCE methodology. Additional benefit can be gained in the future on similar projects by re-using some aspects of the infrastructure, subject to making a number of modifications to fit the new project context.

The creation of the key project processes that interact with DOORS helps achieve the integration of the project processes and infrastructure with DOORS. To assist further in the management of this integrated environment, PMConnex[4] is a product which can be used to create a Project Planning View of

[3]MindManager is a mind mapping application that is produced by Mindjet® (http://www.mindjet.com)

[4] PMConnex is produced by PMC (Project Management Centre Inc), 1 Antares Drive, Suite 550, Ottawa ON, K2E 8C4, Canada

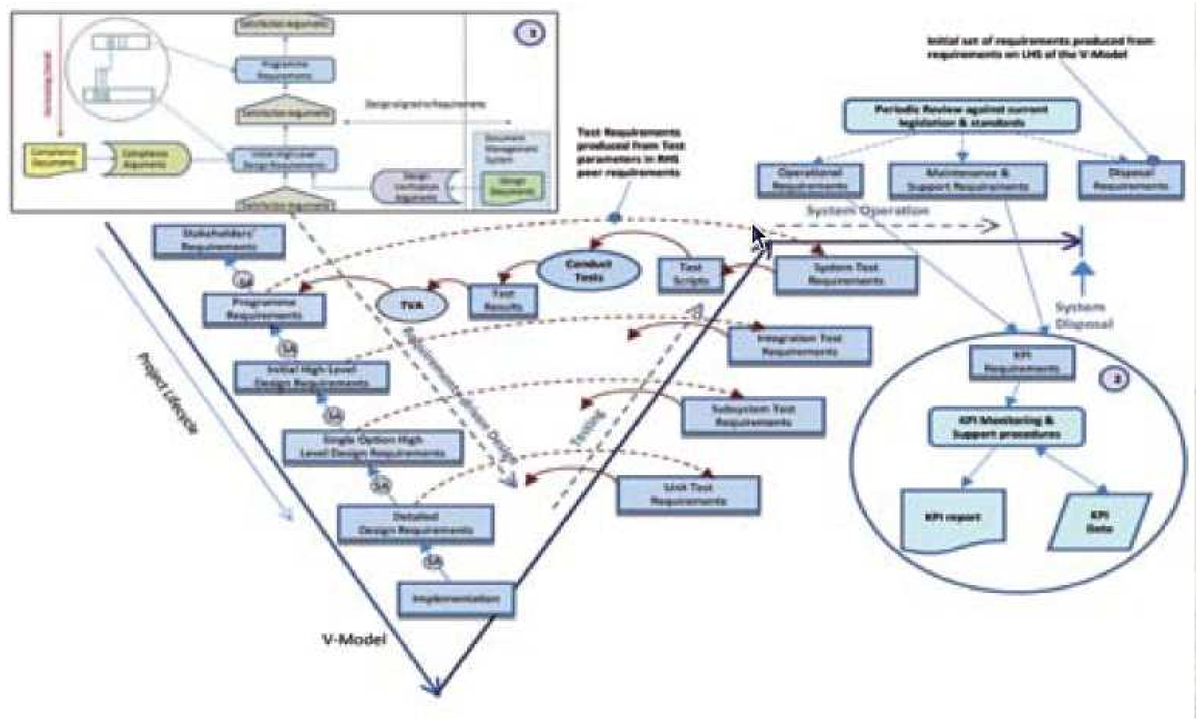

Stakeholders' Requirements
Programme Requirements
Initial High Level Design Requirements
Single Option High Level Design Requirements
Detailed Design Requirements
Implementation
SA
V-Model
Project Lifecycle
TVA
Test Results
Conduct Tests
Test Scripts
Test Requirements produced from Test parameters in RHS peer requirements
System Test Requirements
Integration Test Requirements
Subsystem Test Requirements
Unit Test Requirements
Testing
System Operation
System Disposal
Initial set of requirements produced from requirements on LHS of the V-Model
Periodic Review against current legislation & standards
Operational Requirements
Maintenance & Support Requirements
Disposal Requirements
KPI Requirements
KPI Monitoring & Support procedures
KPI report
KPI Data

Figure 6. Management of the Whole Lifecycle

requirements in DOORS that are linked to activities in the programme, as shown in figure 5. Projects are stored in both the DOORS Database and the PMConnex Database. Links can be established between the project's requirements and WBS elements and/or activities in the programme, and the projects in both databases can be synchronised to achieve alignment. PMConnex can produce Project Planning Views between DOORS and Microsoft Project or Primavera.

The access control and remote working capabilities of DOORS are powerful features that can be used to support the management of multiple contracts. Sets of requirements can be allocated to, or produced by the various subcontractors, and the requirements in these sets can be connected to the requirements in the Project Team's development environment via Satisfaction Arguments (SA). (Note: SAs were discussed in the previous article, and a portion of the diagrammatic structure showing how SAs fit into the requirements hierarchy is shown in inset 1 of figure 6).

The subcontractors can evolve more detailed requirements and related designs within their own lifecycle and development environment, and be free to edit the requirements and operate their own change control. Using the access control facilities in DOORS, they cannot change requirements in the Project Team's environment, as they will only be granted read access to the layer of requirements in the Project Team's environment that they connect to with their SAs. The subcontractors are able to access the Project Team environment using the remote access facilities of DOORS. The subsystems that are developed and implemented can be tested by the subcontractors against the initial set of requirements apportioned to them, and the subcontracted deliverables reviewed against contracted set of requirements. The completed subsystems can be subsequently integrated into the overall system during the subsystem and system integration testing phases of the Project Team's development lifecycle (see figure 7).

The right hand side (RHS) of the V-Model in figure 7 shows the progression through various levels of testing, from unit testing to system testing. When the requirements on the left hand side (LHS) of the V-Model are created, the method of testing also needs to be specified. This information is used to formulate the system test requirements, which are used to define what is needed in the Test Scripts. Peer-to-peer testing is conducted using the RHS Test Scripts to test the LHS requirements, producing a set of test results. A Test Verification Argument (TVA) is created linking each RHS test result to its LHS requirement, where the TVA shows how the test and test result verify that the LHS requirement has been met.

Operation, Maintenance, and Disposal of the System

On completion of testing and final acceptance, the system enters into its operational phase. The Operational Requirements need to be defined from the flow down from the Stakeholders' Requirements during the creation of the LHS requirements hierarchy. Similarly, the Maintenance and Support Requirements are evolved during the creation of the LHS requirements hierarchy, and typically address a number of key areas:

- Performance of the Maintenance Contractors (e.g. response time, problem reports, etc.)
- Condition Monitoring of System Assets
- Maintenance Regime

The need to give early consideration to the Key Performance Indicators (KPIs) that are necessary to measure the performance of various aspects of the system was discussed briefly in the previous article. The resulting KPI Requirements, produced as part of the LHS requirements hierarchy, enable the KPI Monitoring and Support Procedures, plus any measuring mechanisms and systems, to be incorporated into the system design. The KPI Monitoring and Support Procedures obtain metrics on the performance of the operational system, and also on the maintenance and support provided. This is stored as KPI Data (see figure 7, inset 2), which is used by the KPI Monitoring and Support Procedures to produce regular performance reports against KPI targets.

Although the disposal of the system is not normally prominent in most peoples' minds during the design of a system, it is important that it is given adequate consideration during the design phase. Various factors such as build materials, construction, location, can impact on the options available for disposal when the system has reached the end of its operational lifetime. The Disposal Requirements need to be established as part of the LHS requirements hierarchy, as a set of requirements that are equally as important as any other system requirements.

The Operational, Maintenance and Support, and Disposal Requirements may exist for many years whilst the system is operational. It is important that they are subjected to periodic review to ensure that they remain valid in the light of legislation which is current at a future date.

Summary

This series of three articles has highlighted a number of problems that can occur in large development programmes, and suggested ways in which they can be avoided. Particular emphasis has been given to a Requirements-driven Design approach. Although the structures may appear complex, there is a simple principle that operates at the heart of this approach: Specify **what** you require of the system, show **how** you will implement it, and provide the evidence to prove that you have produced the system that was required. This principle is valid for any development, so that the approach can be used across all business sectors.

The current world-wide recession makes it more important than ever to ensure development programmes are conducted efficiently and that they deliver products that meet the specified requirements. The Requirements-driven approach is one that attempts to ensure that 'the right thing is built in the right way'.

© Dr Steve Rivkin

After graduating in Applied Physics, Steve completed both his MSc and PhD in Computer Science at the University of Manchester. Subsequently, he gained considerable project management experience in high profile software development projects. As a Commercial and Operations Manager for a leading software house, he supervised projects for the Industrial, Utilities, and Space Divisions.

Consultancy assignments include project management and systems integration work in the transport sector, on both Highways Agency and major rail projects. Steve has had responsibility for defining and implementing Requirements Management strategies in the rail industry using the DOORS Requirements Database. He has also given workshops and presentations in the nuclear sector describing a Requirements-driven approach to system development.

Steve is a freelance consultant, specialising in systems integration, requirements management, and project management. He is also available for presentations and workshops. Steve can be contacted by email at steve.rivkin@ontrackprojects.com

RE-partee

RE-sources

Books, Papers

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

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

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

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

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

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

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

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

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

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

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

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

Media Electronica

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

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

ReQuirements Networking Group
www.requirementsnetwork.com

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

RE-actor

The committee of the RESG

Contributing to RQ

To contribute to RQ please send contributions to William Heaven (william.heaven@gmail.com). Submissions must be in electronic form, preferably as plain text.

The deadline for RQ 57 (September 2011) is 15th August 2011

Joining the RESG

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