Requirements and risk: singing from the same hymn-sheet

Dresner, D.G.[1] and Garfield, J.E.[2]

[1] School of Computer Science, The University of Manchester, Oxford Road, M13 9PL.
[2] Worcester Business School, University of Worcester, City Campus, Worcester, WR1 3AS.

Abstract

Failing to elicit requirements is as much of a risk in the traditional, negative sense as successfully defining requirements is a positive step towards successful systems development. The discipline of risk management has long since had to deal with the spectre of emergent risk and its inherent lack of predictability. Just as risk management considers how any number of vulnerabilities in a system may be exploited by accident or by malicious intent that preys upon exposure to otherwise independent factors, so successful requirements elicitation is beholden to the ability to recognise the need for, and define, derived requirements. In this paper we suggest that risk assessment and requirements elicitation are two manifestations of the same activity: creating trustworthy software[1]. We propose the research and development of a methodology where the two disciplines converge.

Keywords: Derived requirements, elicitation, design, risk management, trustworthy software.

Introduction

Requirements that are not fully explored during the early phases of development run the risk of an adverse impact on the development of systems (Fuxman et al., 2003) in much the same way that realised risk can have undesirable outcomes (Swann, 2000). Indeed there is much evidence to suggest the need for ascertaining good requirements early in development (Orr, 2004; Kauppinen, 2005) as a foundation for further development (Hofmann and Lehmer, 2001). Complexity reveals an ever growing catalogues of defects (like the Common Weakness Enumeration - CWE). Manifestations of these emerging defects cause us to ask if the formality and rigour we expect in the software engineering discipline is better described by the process of scientific discovery espoused by Koestler's sleepwalkers (Koestler, 1959) creating a growing case study of emergent risk and derived requirements.

However, in many approaches to development, functional and non-functional requirements are gathered hastily (e.g. McAllister, 2006), followed by the rapid development of design, system coding and implementation, e.g. Rapid Application Development (Martin, 1991). This almost certainly leads to the development of an information system which is not strategically viable and will not add value. A number of studies have shown that the cost of fixing requirements errors grows dramatically the later they are corrected (Soni, 2014; Axelrod, 2013) and this is in line with the study of quality defects in manufacturing (Crosby, 1979). Further consequences of which can at best affect the success of the system – such as the Libra court-management system - which automated defective processes rather than re-

[1] BSI, *PAS 754:2014, Software Trustworthiness. Governance and management. Specification*, 2014

engineering them - and at worst the entire organisation and even people's lives – 28 people were killed as a result of a Patriot missile failure based on a poorly derived requirement for accuracy (Dresner, 2011).

This paper focuses on the objective of successfully eliciting derived requirements during the design of systems at the same time as considering risks whose treatment will be effected and affected by the system as written. This contrasts with the expensive (and potentially catastrophic) emergence of requirement of a system in use where risks are realised with expensive consequences for recovery and rework. We contend that considering risk management as a separate discipline to requirements elicitation is inefficient and dangerous. It creates an air-gap between processes that have for too long been overshadowed by the enthusiasm to produce tangible code and the viament of satisfying new and emerging requirements as maintenance rather than admitting to the truth of earlier failures and a need to budget for new development.

Derived requirements

A derived requirement is defined by Young (2004) as one that is further refined from a higher-level requirement or a requirement that results from choosing a specific implementation or system element. This is compatible with the SWEBOK's label of *emergent properties* (IEEE, 2014) which depend on the interoperation of components of the software. Leffingwell and Wigreg (2000) state that important requirement characteristics for derived requirements are traceability, consistency and abstraction. As derived requirements can change as a result of changes in the design, usually without reference to the customer, it is important to keep track of what is derived and what is not (Brooksby, 2003) in much the same way that the balance of risk must be tracked for its sensitivity to changes in probability and impact. Figure 1 shows the relationship between derived requirements and customer requirements. The loops between design and derived requirements indicate the many possible levels of design. A successful design derives non-functional requirements with the same priority as those elicited directly from the customer, baking in the trustworthiness (BSI, 2014).

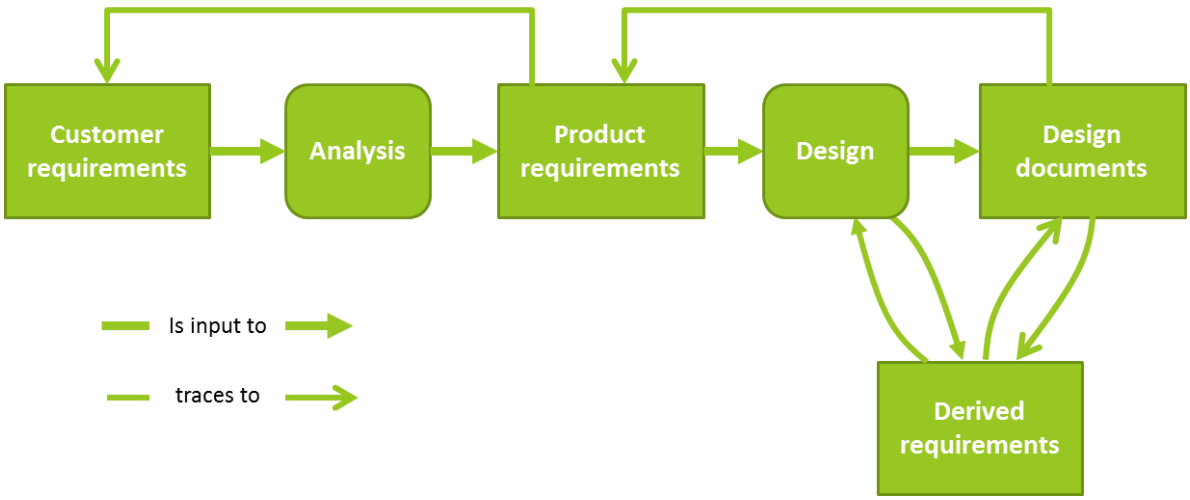

Figure 1: Requirement types (Brooksby, 2003)

Although derived requirements are not elicited directly from the customer/user they need to be treated just like other requirements in the system (Brooksby, 2003). Not satisfying them negatively impacts system performance, their absences constitutes risk, and consequently reduces customer satisfaction (Leffingwell and Wigreg, 2000). It does however tend to be more difficult to elicit derived requirements, and some of them may be overlooked in the process (Leffingwell and Wigreg, 2000) particularly when stakeholders have not been consulted (Checkland, 1981 and Carr, 2003). They should however be easier to change than user requirements and can be derived and deleted. But it is not always easy to know when the derivation of requirements should finish and when a derived requirement should have been part of the core requirements elicitation.

Known and unknown requirements and risks

Governance (BSI, 2014) sets out the involvement – or teamwork – that is needed to derive requirements and an understanding of risks to a system (Carr, Konda, et al., 2003). The categorisation of risk is based on CMU SEI's taxonomy, which describes risks as having one of three characteristics: **known risks** that are well understood and will surface time after time in a risk assessment; **unknown risks** that did not make it into the risk register because the assessment did not call on the right kind of expertise; and **unknowable risks** that could not have been reasonably predicted even with a wide enough representation from the contemporary knowledge base.

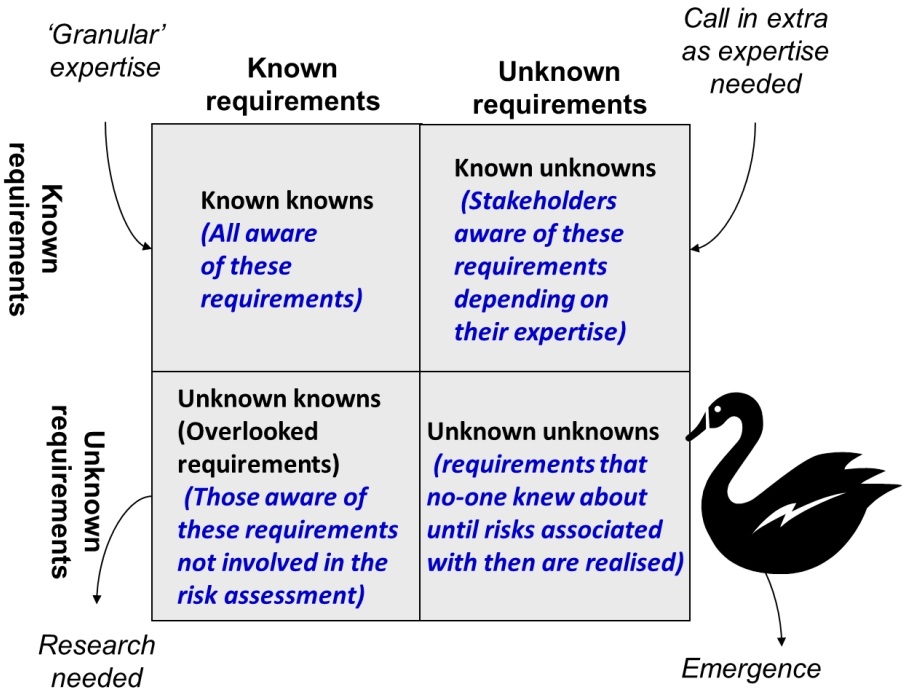

Figure 2 : Known and unknown requirements

Derived requirements may manifest in any part of the quadrant (Figure 2) to show the direction requirements elicitation should take to derive as many requirements as honest governance drives. This quadrant was originally drawn up to consider risks which are 'easily' identified (known), those which are identified by specialist stakeholders (Alexander, 2007) involvement (known unknowns), those which would have been assessed had the

appropriate stakeholder been consulted (unknown knowns) and the 'black swan' risks which were outside contemporary knowledge (Taleb, 2008). For teamwork in particular consideration of a 'worldview' is needed to create the conditions for derived requirements to emerge early enough to remove or reduce the cost of rework and increase the risk of system success. Checkland (1981) incorporates this aspect into the 'Weltanschauung' of the Soft Systems Methodology. Just as the management process of risk assessment (and then treatment) surrounds a system asset with the means of its protection, so the requirements elicitation process needs to consider treating the risks that would prevent a requirement being met, and the positive treatment of what needs to be done to develop the certainty that the requirement will be realised Figure 3. These are two manifestations of risk treatment which we suggest can be handled together.

For example, the methodology to achieve for software trustworthiness (BSI, 2014) establishes a Trustworthy Software Constraint and Dependency Model before implementation to include external dependencies, such as customers and how they form an supply chain. The 'Trustworthy Software Framework' establishes a demand to understand the use cases for software and elicit derived requirements from them. This leads to risk management considerations - such as privacy and cryptography – being built into the design at an early stage. This is an efficiency shown in Crosby's Quality Management Grid (Crosby, 1979) which shows the cost of quality based on removing defects. The cost of removing defects reduces the earlier in the life cycle it's done, and the maturity of the organisation required to achieve this increases. The consolidation of processes such as requirements elicitation and risk management is likely to be a metric of maturity. Security is a primary system/software quality characteristic (ISO, 2011) so security vulnerabilities and breaches are manifestations of quality defects and remedial controls are tantamount to derived requirements.

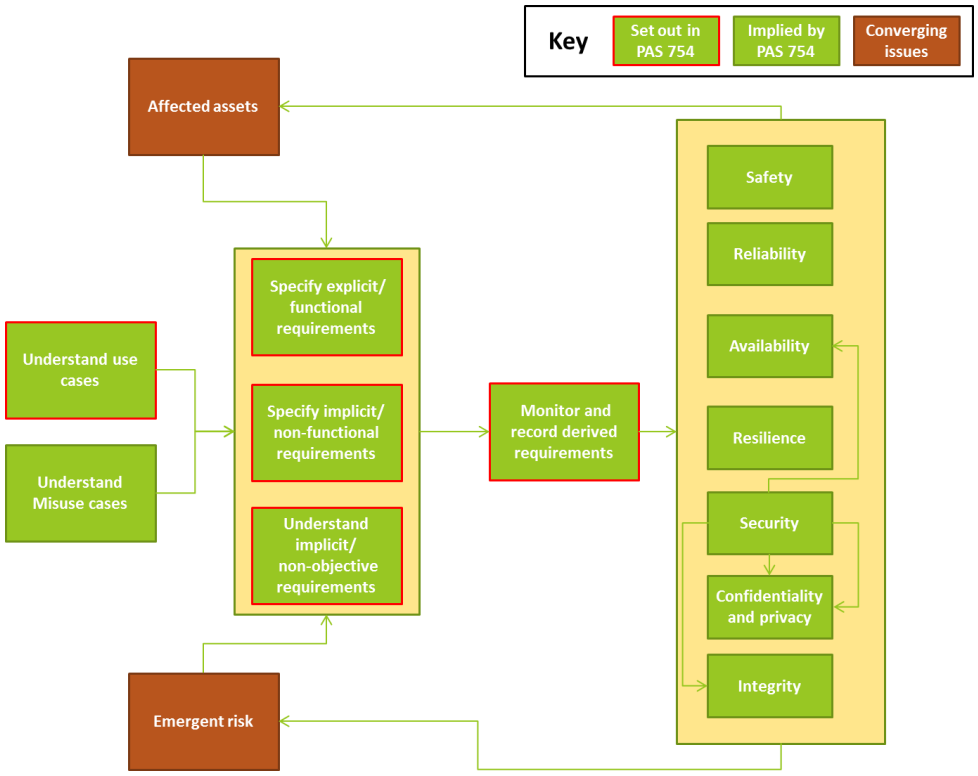

Figure 3: Known process elements for risk-based requirements elicitation

The risks of placing transient content onto websites may result in advertisements (for example) appearing with other material with which the advertisers do not want to be associated. Assessing the software requirements derives the need to manage such risk, but this calls for the knowledge of the content owners that such a technology is available and could be used and for them to define what is acceptable to them and to the website designers and programmers to recognise where constraints can or cannot be made. It may be an instance where a risk is accepted or the facility rejected on account of the risk. An extrapolation of this example is the lack of control afforded to an advertiser placing adverts on a social networking site and the freedom of users to define content outside the close control of the website[2].

Moreover the choice of solution may reveal or create another, even more complex problem (Rittel and Webber, 1984) which may be expressed as a measure of risk. Derived requirements could therefore be termed as 'wicked' problems. 'Wicked' problems are typically characterised by uncertainty, conflict and uniqueness (Fischer et al., 1991). Conflicts may result from trade-offs with other requirements. And as the size of systems project increases so too does the complexity and the risk of errors. Therefore, derived requirements must be consistent with respect to the other requirements (Leffingwell and Wigreg, 2000).

ISO/IEC 25010 (and its predecessor ISO.IEC 9126) has long since defined a requirements framework for systems and software quality based on characteristics which must be defined by a system's requirements. PAS 754 (BSI, 2014) isolates five non-functional requirements as being the defining characteristics of trustworthy software these are: safety, reliability, availability, resilience and security (Watson, 2014). We suggest that a method can be defined that benchmarks requirements decisions for their impact on these five characteristics to enable emergent properties of a system to be teased out. Throughout development – and subsequently managed change – the understanding of system's safety, reliability, availability, resilience and security must be whole. This would refine and integrate current practice to create a more sensitive method than expressing the impact of software risk in terms of probability and information asset value.

Systems frequently fail to adequately protect the assets in their charge (Price Waterhouse Cooper, 2014). The UK Trustworthy Software Initiative (www.uk-tsi.org) reports that 90% of the incidents reported to GovCERTUK (Computer Emergency Response Team (CERT) for UK Government) can be attributed to software bugs. The failure of the body of knowledge to encourage such protection (Dresner and Jones 2014), has led us to focus on the security vector in particular although as security comprises confidentiality, integrity, and availability (ISO, 2013), it is inseparable from common requirements that may be expected in every system depending on the risk decisions made during times of design or change.

We have already seen that the requirements that take into consideration human factors can be measured against the security controls adopted by the system. Figure 4 draws on research by Dresner and Garfield (2014) and shows the areas of adequate and inadequate

[2] http://news.bbc.co.uk/1/hi/uk_politics/6929161.stm, 08 August 2007

requirements. The better that the system has security controls built in – defined here as the manifestation of the system owner's risk attitude – then the less reliant that the system's assets are on its customers and actors for their security and by implication the other (non-functional) quality requirements as expressed by quality characteristics of the system.

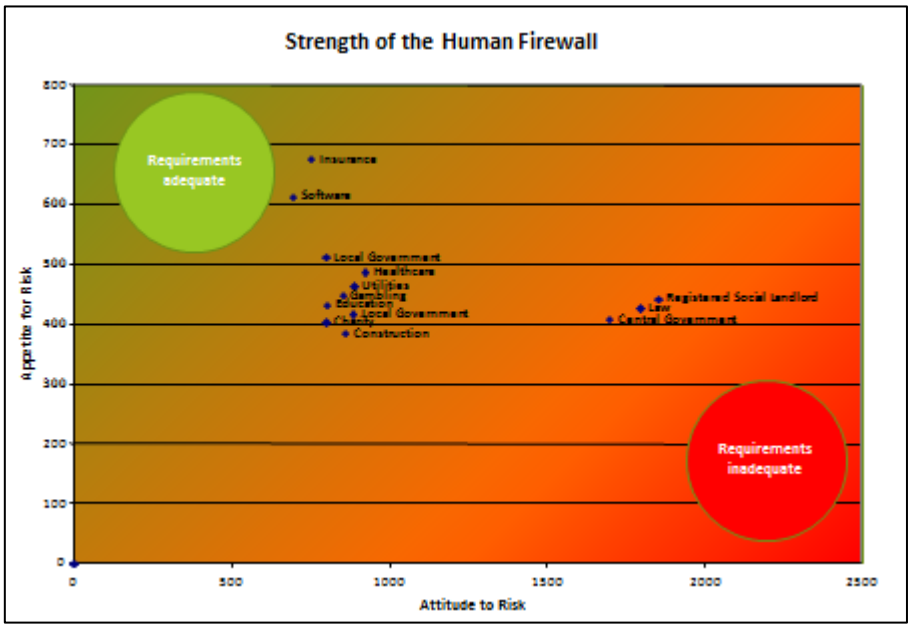

Figure 4: Adequate and inadequate requirements

Conclusion

It could be said that requirements always tend to be incomplete. And the complex or otherwise termed 'wicked' (Rittel and Webber, 1984) nature of derived requirements makes their determination difficult as does the nature of emergent risk to the elicitation of building in risk treatments to a system. The process of risk assessment and treatment is part of requirements elicitation and vice versa. Considering measures for monitoring and elicitation of derived requirements could go some way toward lessening the impact that lack of elicitation may cause in the same way that mitigating actions, or controls, are derived to reduce the impact of negative risk and similarly – albeit culturally less often – increase the risk of success. The sooner that risk management and requirements elicitation are brought together into a single discipline, the sooner we may approach the nirvana of trustworthy software.

References

Alexander, I.F. (2007). *A Taxonomy of Stakeholders: Human Roles in System Development*, Issues and Trends in Technology and Human Interaction, Editor(s): Bernd, Carsten, Stahl (De Montfort University, UK) IRM Press. 25-71 pp.

Axelrod, C.W. (2013) *Engineering safe and secure software systems*, Norwood, Massachusetts: Artech House.

Brooksby, R. (2003) *Requirements and change*, Cambridge: Ravenbrook Ltd.

BSI (2014) *PAS 754:2014 Software trustworthiness – Governance and management – Specification*, British Standards Institution

Carr, M.J., Konda, S.L., et al. (2003) Taxonomy-Based Risk Identification, Software Engineering Institute.

Checkland, P. (1981) *Systems thinking, systems practice*, London: John Wiley and Sons Ltd.

Crosby, P. (1979). *Quality is free: The Art of Making Quality Certain*, McGraw Hill.

Dresner, D.G., (2011) *A study of standards and the mitigation of risk in information systems*, A thesis submitted to The University of Manchester for the degree of Doctor of Philosophy in the Faculty of Humanities

Dresner, D. G., Jones, N. (2014) *The Three Laws of Information and Cyber Security*, Cybertalk Issue 6, Softbox

Dresner, D.G. and Garfield, J.E. (2014) Balancing risk appetite and risk attitude in requirements: a framework for user liberation, UK Academy for Information Systems (UKAIS 2014), St Catherine's College, University of Oxford, Oxford, 7 – 9 April 2014.

Faisandier, A. (2012) *Systems architecture and design*, Belberaud, France: Sinergy'Com.

Fischer, G., Lemke, A.C., McCall, R. and Morch, A. (1991) Making argumentation serve design, *Human Computer Interaction*, 6 (3-4), 393-419.

Fuxman, A., Liu, L., Pistore, M. and Roveri, M. (2003) Specifying and analysing early requirements: some experimental results, Proceedings of the 11th IEEE international conference on Requirements Engineering, Monterey Bay, California, USA, 8-12 September 2003.

Hillson D., and Murray-Webster, R., (2007). *Understanding and Managing Risk Attitude* (2nd edition), Gower Publishing.

Hofmann, H.F. and Lehmer, F. (2001) Requirements Engineering as a success factor in software projects, *IEEE Software*, 18(4), 58-66.

IEEE (2014). *Guide to the Software Engineering, Body of Knowledge, Version 3.0, SWEBOK®*,

ISO (2011*) ISO/IEC 25010 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*, International Standards Organisation

ISO (2013) *ISO/IEC 27001 Information technology — Securitytechniques — Information securitymanagement systems — Requirements*, International Standards Organisation

Kauppinen, M. (2005) *Introducing Requirements Engineering into product development: towards systematic user requirements definition*, PhD Department of Computer Science and Engineering, University of Technology, Espoo, Finland.

Koestler A., (1959) *The Sleepwalkers: A History of Man's Changing Vision of the Universe*, Hutchinson

Leffingwell, D. and and Wigrig, D. (2000) *Managing software requirements. A unified approach*, London: Addison-Wesley.

McAllister, C.A. (2006) *Requirements determination of information systems: user and developer perceptions of factors contributing to misunderstandings*, Ann Arbor, Michigan: ProQuest.

Martin, J. (1991) *Rapid Application Development*, New York: Macmillan, USA.

Orr, K. (2004) Agile requirements: opportunities or oxymoron? *IEEE Software*, 21(3), 71-73.

Price Waterhouse Cooper (2014) *Information Security Breaches Survey 2014 Technical Report*, Department for Business, Innovation and Skills (BIS)

Rittel, H.W.J. and Webber, M. (1984) Planning problems are wicked problems, In Cross, N. (ed) *Developments in design methodology*, Chichester: John Wiley and Sons, 135-144.

Soni, M. (2014) *Defect prevention: reducing costs and enhancing quality*, Six Sigma.

Swann, G. M. P., (2000) *The Economics of Standardization,* Department of Trade and Industry

Taleb, N. N., (2008) *The Black Swan: The Impact of the Highly Improbable*, Penguin

Watson, T., *Trustworthy software as a foundation for UK cyber security*, Cyber Security Review, Summer 2014

Young, R.R. (2004) *The Requirements Engineering Handbook*, London: Artech House Technology Management and Professional Development.