The state-of-the-art of Open Source Requirements Elicitation

Mark Elkins and Brian Dupée

Faculty of Technology Research Centre
Southampton Solent University, Southampton UK
mark_elkins@bcs.org

Abstract

Open source, free software, requirements elicitation, and requirements engineering are introduced prior to considering a concatenation of open source and requirements elicitation as a topic. This review of the literature suggests there is little previous research on this concatenated topic. From the limited literature unearthed there is evidence of non adoption of modern software engineering practice, informal requirements elicitation and engineering, developer led requirements, doubt over the use of formal standards, and open source as a socio-technical process that may reduce conflict. A previously published suggested process for open source requirements engineering and elicitation is also considered. Finally it is concluded that as open source ideology is producing quality software it might have something different to offer the current state-of-the-art in requirements elicitation.

1 Introduction

This need for a literature review of requirements elicitation that included consideration of open source software originally came about as part of the requirements for an internal MPhil to PhD transfer report written by the lead author in May 2008 for Southampton Solent University. In addition the lead author has a general interest in the concatenated topic of requirements elicitation and open source due to his involvement as a founder member and committee member of the British Computer Society (BCS) Open Source Specialist Group (OSSG).

1.1 Open source and free software

The original and main principle behind open source ideology is that software code should be free for anyone to access, modify, and distribute regardless of whether or not they wrote the code. "Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with..." certain "...criteria" set out in "The Open Source Definition" (Open Source Initiative 2006), which is managed by the Open Source Initiative (OSI). The foundation of this essentially comes from the earlier definition of free software defined by Richard Stallman in the GNU public licence (GPL) (GNU project 1991). "The Free Software Foundation (FSF), established in 1985, is dedicated to promoting computer users' rights to use, study, copy, modify, and redistribute computer programs" (The Free Software Foundation 2007). The FSF state in "The Free Software Definition" (The Free Software Foundation 2005) that "Free software' is a matter of liberty, not price. To understand the concept, you should think of 'free' as in 'free speech,' not as in 'free beer'".

It is not easy to understand the difference between "free software" and "open source". Sam Williams in chapter 11 of his book entitled "Richard Stallman's Crusade for Free Software" (Williams 2002) attempts to tackle this issue. A brief synopsis of that chapter might find that the terms 'open source' and 'free software' are to a great extent interchangeable, but the free software movement from the perspective of Richard Stallman prefer the term 'free software' to 'open source' and dislike the more market friendly attitude of open source. For example although the majority of open source software appears to be licensed under the GPL (originally created for the free software movement), there are other open source licences available that are considered more business friendly. Ardent supporters of the free software movement such as Stallman might reject these.

1.2 Requirements elicitation

From a top level perspective requirements elicitation might be defined simply as 'the methodologies used to find the requirements'. Fox (2007) states "requirements elicitation – the activity of determining stakeholders needs and desires for a product". The two definitions might be considered as similar if the word 'methodologies' is considered to be fairly interchangeable with the word 'activity' and it is assumed that the whole purpose of requirements elicitation is to find the stakeholders requirements. Gilb (2005) offers a definition not inconsistent with this purpose when stating that "Requirements give information to the system designers and to a wide range of stakeholders. They state what the stakeholders want the system to achieve". Aurum and Wohlin (2005) mention something similar whilst defining various stakeholders with

> "Clearly a requirement is a collection of needs arising from the user and various other stakeholders (general organization, community, government bodies and industry standards), all of which must be met".

The 2004 version of the IEEE SWEBOK Guide to the Software Engineering Body of Knowledge (Sawyer and Kotonya 2004) states

> "Requirements elicitation is concerned with where software requirements come from and how the software engineer can collect them. It is the first stage in building an understanding of the problem the software is required to solve".

Within the literature some alternative terms exist in place of requirements elicitation. For example there is no mention of the term elicitation or requirements elicitation within the index of Robertson and Robertson (2006). Instead the term "trawling for requirements" is used to cover the requirements elicitation process. Hull, Jackson, and Dick (2005) also contains no mention of the term requirements elicitation or elicitation within the index. Instead the term "requirements capture" is used within the index, which actually directs the reader to a section headed "Capture Requirements". However Nuseibeh and Easterbrook (2000) are critical of the term capture. Instead they prefer the term elicitation "...to avoid the suggestion that requirements are out there to be collected simply by asking the right questions".

Zowghi and Coulin (2005) suggest that the existence of different terms is endemic within the literature when stating that "...there is very little uniformity in the research literature and practice concerning the names given to the activities often performed during requirements elicitation". Therefore in order to set a comprehensive background for requirements elicitation research it could be argued that these differences have to be covered to appreciate the current 'state of the art'.

1.2.1 Elicitation techniques

Nuseibeh and Easterbrook (2000) offer a very comprehensive view of elicitation techniques, which includes grouping them into "...a number of classes...". They mention that "...there are a plethora of elicitation techniques available to the requirements engineer...". More recently Zowghi and Coulin (2005) reinforce this view when mentioning that "...there are literally hundreds of different techniques ...from a variety of sources that can an have been employed for requirements elicitation". They believe the following selection is representative of the range described in the literature and practised in industry at the time: interviews, questionnaires, task analysis, domain analysis, introspection, repertory grids, card sorting, laddering, group work, brainstorming, joint application development (JAD), requirements workshops, ethnography, observation, protocol analysis, apprenticing, prototyping, goal based approaches, scenarios, and viewpoints.

Another example of techniques included in recent literature are those in Chapter 5 of Robertson and Robertson (2006) entitled "Trawling for Requirements": business use case, understanding the current situation and modelling it, apprenticing, observing structures and patterns, interviewing stakeholders, getting to the essence of the work, solving the right problem, business use case workshops, outcomes, scenarios, business rules, creativity workshops, brainstorming, personas, mind maps, wallpaper, video and photographs, wikis, blogs, and discussion forums, document archaeology, family therapy, and soft systems and viewpoints.

1.2.2 Elicitation frameworks

The techniques mentioned above may be used within a variety of elicitation frameworks, which might alternatively be referred to in the literature as methods, approaches, or processes. In view of the confusion over terminology it might be useful to attempt to make a clear definition of what the term elicitation framework is intended to mean within this research. Therefore the following definition is suggested:

> 'The immediate guiding framework within which individual elicitation techniques are used and contained'.

Examples of different frameworks include: Structured Analysis and Design (SAD) (Zowghi and Coulin 2005), Object-orientated (Kendal and Kendall 2005), Service Orientated Architecture (SOA) (Newcomer and Lomow 2005), Business processes (Holt 2005), Agile Methodology (Sillitti and Succi 2005), and Open Source (Scacchi 2004).

1.2.3 Conceptual need

There appears to be some confusion within requirements elicitation literature over whether mention of the initial identification of the conceptual need/problem for a software project is included. Identification of initial need is the foundation that all software projects are based upon and hence the importance this has on the quality of software produced (Brooks 1995; Leffingwell and Widrig 2003). Indeed "One of the most important goals of elicitation is to find out what problem needs to be solved, and hence identify system boundaries" (Nuseibeh and Easterbrook 2000).

> "Only by understanding the essence of the problem do we become able to find the correct requirements, and eventually the right solution. In fact, once you have found the essence of a problem, its solution is usually self-evident" (Robertson and Robertson 2006).

It is common to find the literature making the assumption that the need for a software project has already been identified when discussing the elicitation process (Sommerville and Sawyer 1997; Zowghi and Coulin 2005). However such literature may nevertheless discuss the purpose of elicitation in the context of finding out exactly what a system is required to do (Zowghi and Coulin 2005). Indeed the elicitation process can legitimately identify that the need for a software project is unfounded and at this stage the project will be terminated (Boehm 2000).

Confusion in literature about the conceptual need being primary is a problem that may come directly from the academic world. For some reason, academics often assume that the need is already proven whereas experienced practitioners (Robertson and Robertson 2006) appear less likely to fall into this trap. The difference is perhaps the approach taken in the academic and practitioner views.

Ultimately it is possible that the initial need has been identified and that elsewhere in the software project life cycle that need is not fully reflected (Taylor 2001; Warren 2004) if proper steps are not followed to improve the chance that software quality is achieved (British Standards Institution 2001). Therefore other matters in the software project process such as poor project management could undermine successful requirements elicitation.

1.2.4 Mass market driven development

"We face a dilemma, where we need to deploy finished, embedded software for millions of users without knowing what the customer really wants" (Rossi and Tuunanen 2003). This highlights an important issue within the elicitation process over whether or not a 'one-size-fits-all' generic approach to requirements elicitation is correct or not. For example Regnell and Brinkkemper (2005) discuss

> "Market-driven development... where the development costs of a generic product are divided among many buyers on an open market and where the potential profit is rewarded to the producer... is different from customer-specific development (also called bespoke development), where one single customer pays all development costs and the resulting product is specific to the needs of that one customer".

1.2.5 Business versus technical requirements

Aurum and Wohlin (2005) attempt to classify different types of requirements. Within that classification is "Business requirements versus technical requirements". For example software for a business information system may require a different type of elicitation process to that intended for use in a motor car engine management system. The main reason might be because the elicitation process for the business information system takes place on an ongoing basis whereas that for the engine management system might traditionally be intended to stay the same once it has been created. However according to Berry and Lawrence (1998) "While the literature draws a distinction between different types of requirements, in practice it is not always easy to identify such differences”. For instance it may be possible to alter the engine management software after a car is several years old by sending new instructions to the firmware within that system. This might not be something the original car manufacturer intended. Nevertheless it could perhaps be considered desirable for the purpose of improving fuel consumption. Therefore the engine management system might blur the distinctions between business and technical software requirements.

1.2.6 Elicitation of legal requirements

Chapter eight of Robertson and Robertson (2006) highlights the growing importance of legal “...compliance requirements...” (Woollacott 2007) such as “ The Sarbanes-Oxley Act (SOX; officially entitled the Public Company Accounting Reform and Investor Protection Act of 2002)...”. Although SOX is legislation enacted in the United States (US) the multi-national nature of many large organizations means that non-US parts of an organization “...that do business with their US counterparts...” now seek to comply with SOX. Other types of legal requirements that may have to be considered include disabilities and data protection legislation. Therefore a “requirements analyst” must trawl/elicit these types of requirements for inclusion in a new or existing system and consider the suitability of which elicitation techniques and frameworks they might best use for this purpose.

1.3 Requirements engineering

It is important to mention that requirements elicitation is part of the requirements engineering process, which in the context of software engineering is essentially about a set of critical activities that seek to ensure the user gets the software they need (Aurum and Wohlin 2005; Hull et al 2005). Core activities include elicitation, modelling and analysis, verification, and validation (Hickey and Davis 2002; Nuseibeh and Easterbrook 2000; Aurum and Wohlin 2005).

Whereas as the context of this process in the past might have only dealt with the first phase of the software development life cycle it is now considered an ongoing process throughout the entire life cycle (Robertson and Robertson 2006; Hull et al 2005; Aurum and Wohlin 2005). A recent definition by Fox (2007), which highlights that view states “requirements engineering – the activity of creating, modifying, and managing requirements over the life of a product”.

2 Open source and requirements elicitation

The free transfer of knowledge, encouraged under open source ideology, it might be assumed could assist the requirements elicitation process by removing barriers, which might otherwise have hidden the true needs and wants of stakeholders. Therefore it could be argued that open source is as much a technique as it is a framework. There is also probably nothing to prohibit the use of open source software coupled with the current state-of-the-art of requirements elicitation as outlined above.

2.1 Suggested process for open source requirements engineering and elicitation

One recent paper by Paech and Reuschenbach (2006) entitled “Open Source Requirements Engineering” by implication appears from the title to closely fit the topic of requirements engineering and open source.

“The paper describes an experience in RE for the selection of a university-wide open source

E-Learning software at the University of Heidelberg" and claims that "We have presented a successful process for open source requirements engineering". The authors state that "We call the process open source requirements engineering for several reasons:

- The requirements concern open source software (OSS).
- The spirit of the RE effort was similar to OSS efforts: user driven and just-in-time
- The process also seems suitable for RE during OSS development".

However this process appears to have no clear connection with how Scacchi (2004) reports what happens in the open source community from a requirements engineering perspective. For example Paech and Reuschenbach (2006) indicate that ".. 36 high-level requirements were elicited..." through "...brainstorming...". In essence they are suggesting a process that they think would work for open source requirements engineering. This can be seen when they state that "The user-driven and just-in-time spirit of the process seems to fit well to OSS development". However even though they consider it will work well for OSS development it would appear that this is not an identified accepted process being used by the open source community. Therefore the title of their work "Open Source Requirements Engineering" is somewhat misleading. Also it would be difficult for the open source community to consider it because it is not published in an open source way perhaps using the GNU Free Documentation License (GNU project 2002), or a Creative Commons licence (Creative commons n.d). For example "With a Creative Commons licence, you keep your copyright but allow people to copy and distribute your work provided they give you credit...".

2.2 Lack of literature

Of the five references supplied by Paech and Reuschenbach (2006) there are three that suggest an open source content. All of these are journal articles as opposed to conference proceedings. Two of these are from 2004 and one is from 2001. This appears to vindicate a lack of abundance of information about open source and requirements engineering. Indeed Scacchi (2004) (which is a reference cited by Paech and Reuschenbach (2006)), states that "...little is known about how community participants coordinate software development across different settings, or about what software processes, work practices, and organizational contexts they need for success".

2.3 Non adoption of modern software engineering practice

Scacchi (2004) finds about Free and Open Source Software (FOSS) that "Unlike the software engineering world, FOSS development communities don't seem to readily adopt modern software engineering processes". Such a situation may appear a cause for concern for some as this may suggest the theory and practice of requirements engineering and elicitation (as mentioned above), which might be considered of high importance in modern software engineering, is simply being avoided. However Scacchi then states "FOSS communities develop software that's extremely valuable, generally reliable, globally distributed, made available for acquisition at little or no cost, and readily used in its associated community".

2.4 Informal requirements elicitation and engineering

Open Source requirements elicitation appears to take place in an informal way according to

Scacchi (2004) who reports that

> "FOSS requirements take the form of threaded messages or discussions on Web sites that are available for open review, elaboration, refutation, or refinement. Requirements analysis and specification are implied activities. They routinely emerge as a by-product of community discourse about what its software should or shouldn't do and who'll take responsibility for contributing new or modified system functionality. The requirements appear as after-the-fact assertions in private and public email discussion threads, ad hoc software artifacts (such as source code fragments included in a message), and site content updates that continually emerge.
>
> More conventionally, requirements analysis, specification, and validation aren't performed as a necessary task that produces a mandated requirements deliverable. FOSS systems seem to evolve through minor improvements or mutations that are expressed, recombined, and redistributed across many releases with short life cycles. As a result, these mutations articulate and adapt a FOSS system to what its user-developers want it to do while reinventing the system"

Interestingly from the literature mentioned on requirements elicitation techniques above Robertson and Robertson (2006) report the use of "...wikis, blogs, and discussion forums...".

2.5 Developer led requirements

It appears that open source requirements tend to be developer led. According to Thomas and Hunt (2004) "A developer has a need and starts developing software to fill that need. (That's why much OS software is tools for developers.)". Scacchi (2004) states

> "In short, requirements take these forms because FOSS developers implement their systems and then assert that certain features are necessary. They don't result from the explicitly stated needs of user representatives, focus groups, or product marketing strategists".

This connects with the idea put forward by Raymond (2002) that "Every good work of software starts by scratching a developers personal itch". Srijith (2002) cites Asklund and Bendix (2002) who also mention a developer led perspective when stating that "OSS projects are run by developers for developers and there is no time nor interest or resources for bureaucratic overhead".

Therefore as there is evidence that many software projects generally have not taken care to define the business case (British Computer Society 2005; Bulger 2000) there would appear to be a clear danger with developer led projects. For example a "...lack of knowledge among technologists about what business users want" was mentioned in a "...review of 1,000 projects by the UK (United Kingdom) Office of Government Commerce (OGC)...", which "...found that technology was one of the least likely reasons for a project to fail" (British Computer Society 2005).

2.6 Standards and legal requirements

According to Aurum and Wohlin (2005) a standard can be those of "... stakeholders (general organization, community, government bodies and industry standards)...". It could be argued that because FOSS development can involve a large amount people that unless all their input is recorded it may be difficult to ascertain whether or not the final product has considered certain standards of bodies such as International Organization for Standardization (ISO) or IEEE, which basically attempt to guide the development of improved software quality. Scacchi (2004) reports a lack of formal records when stating that "Studies to date have yet to find records of formal requirements elicitation, capture, analysis, and validation—the kind suggested by modern software engineering textbooks...". This would seem to indicate that it is not known whether or what standards are being considered in the development of open source software. Another facet of standards is the growing importance of legal requirements, outlined above. If there is non adoption of these then it is possible open source development might be subject to litigation.

However the more ubiquitous types of open source software such as the Firefox browser and Ubuntu operating system are clearly taking account of many technical standards (and legal requirements such as disability legislation). Otherwise they simply would not work as well as they do in for example viewing websites and working in a device independent way with different hardware. The difficulty comes when trying to assess the use of standards such as software quality standards considered by TickIT (British Standards Institution 2001), which cannot be seen working in software in the same way as a standard such as XML. Another factor to consider is that there is some evidence that the open source community can be vigorously defensive about adherence to the use of open standards (Allison 2005).

2.7 Open source as a socio-technical process

Scacchi (2004) states that

> "FOSS technology transfer isn't an engineering process—at least not yet. It's instead a sociotechnical process that entails the development of constructive social relationships; informally negotiated social agreements; and a routine willingness to search, browse, download, and try out FOSS assets. It's also a commitment to continually participate in public.... Community building and sustained participation are essential, recurring activities that let FOSS persist without centrally planned and managed corporate software development centers".

Shah (2005) offers some explanation behind the idea of using open source methodology beyond the principle of open software code, which offers further evidence of open source as a socio-technical process. He states

> "Community-based innovation has contributed to technological and industrial advances in many fields. Users are at the center of this model: they discover new needs and desires, cooperate with other users within innovation communities, and sometimes even commercialize their innovations. The community-based innovation model is pervasive across time and context, contributing to the development of physical and virtual products and shaping products, industries, and scientific disciplines. Yet, for a number of reasons, communities of users often go unnoticed by firms, policymakers, and society at large".

It might be argued that in the light of what Scacchi (2004) and Shah (2005) mention about open source in the context of a socio-technical process that instead of making use of modern requirements engineering and elicitation theory and practice (as outlined within section 1, above) it has something different to offer that perhaps could be incorporated into it. However whereas Shah (2005) states “Users are at the Center...” Scacchi (2004) appears to indicate that “...user-developers...” are.

2.7.1 Reducing conflict

“Do we really understand the nature of conflict among stakeholders in requirements negotiation?” (Ma and Hall 2006). The community based nature of open source mentioned by Shah (2005) and others (Scacchi 2004; Raymond 2002; Von Hippel 2007) may have the potential to reduce conflict. This is because it has been suggested that “...RE is a social and technical process involving extensive interactions among different stakeholders from different backgrounds with different individual and organizational goals” (Ma and Hall 2006).

3 Conclusion

The above review of the literature appears to indicate that, although open source ideology may have positive attributes, far more research is needed to gather evidence of what happens in open source software requirements elicitation due to a lack of available material.

From the limited literature unearthed there is consideration of a previously published suggested process for open source requirements engineering and elicitation. This appears not to have a clear connection with other available literature on the topic nor is it published in an open source way. Rather it is a suggested process of what the authors think is suitable for open source development.

Available literature indicates that it is not known whether or what standards are being considered in the development of open source software. However the more ubiquitous types of open source software such as the Firefox browser and Ubuntu operating system are clearly taking account of many technical standards otherwise they simply would not work as effectively as they do. For example in viewing websites and working in a device independent way with different types of hardware. This is in many ways the nub of open source and requirements elicitation in that it is clear quality software is being produced and yet there appears a lack of use of formal state-of-the-art of requirements engineering and elicitation theory and practice, which other evidence suggests is required for software quality.

Some evidence suggests that open source development is very much based on developer led requirements with a non adoption of modern software engineering practice. Informal requirements elicitation and engineering are suggested as the norm. Yet software like the Firefox browser is clearly a success with the general public as opposed to those just developing it.

Due essentially to the community based nature of open source it may have the potential to commend itself as a socio-technical process that entails the development of constructive social relationships and other related attributes. This may have the potential to reduce conflict in the requirements engineering process.

Finally because open source ideology is producing quality software then maybe it has something different to offer the current state-of-the-art in requirements elicitation, which perhaps could be incorporated into it.

4 References

Allison, J. (2005). A Tale of two standards. In C. Di Bona, D. Cooper, and M. Stone (Eds.), Open Sources 2.0 (pp.37-56). Sebastopol, CA: O'reilly.

Asklund, U., & Bendix, L. (2002, February). A study on configuration management in open source software projects. IEE Proceedings – Software, 149 (1), 40-46. Cited in Srijith, K.N. (2002, October). Study on management of open source software projects, Report No. CS 5212 Software Project management Project Report – Research Paper [online]. National University of Singapore. Available http://srijith.net/publications/cs5212_report.pdf [accessed 3rd December 2007]

Aurum, A. & Wohlin, C. (2005). Requirements Engineering: Setting the Context. In A. Aurum, and C. Wohlin (Eds.), Engineering and managing software requirements (pp.1-15). Berlin : Springer.

Berry, D.M., & Lawrence, B. (1998). Requirements Engineering. IEEE Software, 25(2), 26-29. cited in Aurum, A. & Wohlin, C. (2005, p.4). Requirements Engineering: Setting the Context. In A. Aurum, and C. Wohlin (Eds.), Engineering and managing software requirements (pp.1-15). Berlin : Springer.

Boehm, B. (2000, September). Project termination doesn't equal project failure. Computer, 33 (9), 94-96.

British Computer Society. (2005, March 16). Thought Leadership, Why are complex IT projects different? [online]. British Computer Society. Available: http://ww.bcs.org/thoughtleadership/complex [accessed 15 January 2006].

British Standards Institution. (2001). The TicketIT Guide (Issue 5.0). British Standards Institution.

Brooks, F. P. (1995). The Mythical Man-Month (2nd ed.). Addison Wesley Longman.

Bulger, P.M.J. (2000). Large–scale software projects - World-Class success or failure?. Cheadle, Cheshire: Bulger Associates Ltd.

Creative commons. (n.d.). Licence your work [online]. Creative commons. Available: http://creativecommons.org/license/ [accessed 8 May 2007].

Fox C. (2007). Introduction to Software Engineering Design, Processes, Principles, and Patterns with UML 2. Boston, Massachusetts: Pearson/Addison Wesley.

The Free Software Foundation. (2005, February 12). The Free Software Definition [online], The Free Software Foundation. Available: http://www.fsf.org/licensing/essays/free-sw.html [accessed 9 May 2007].

The Free Software Foundation. (2007, May 2). Home page [online], The Free Software Foundation. Available: http://www.fsf.org/ [accessed 9 May 2007].

Gilb, T. (2005). Competitive engineering, A handbook for systems engineering, requirements engineering, and software engineering using planguage. Elsevier Butterworth Heinemann.

GNU project. (1991, June). GNU General Public License (Version 2) [online], GNU project. Available: http://www.gnu.org/copyleft/gpl.html [accessed 9 May 2007].

GNU project. (2002). GNU Free Documentation License [online], GNU project. Available: http://www.gnu.org/licenses/fdl.html [accessed 9 May 2007].

Hickey, A.M., & Davis, A. M. (2002). Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Processes. In Proceedings of the 36th Hawaii International Conference on System Sciences, Los Alamitos, CA, USA, pp. 96-105.

Holt, J. (2005). A pragmatic guide to business process modelling. Swindon: The British Computer Society.

Hull E., Jackson K., & Dick J. (2005). Requirements Engineering (2nd Ed.). Springer.

Kendall K.E, & Kendall J.E.(2005). Systems Analysis and Design (6th Ed.). Upper Saddle River, New Jersey: Prentice Hall.

Leffingwell, D., & Widrig, D. (2003). Managing Software Requirements, A Use Case Approach (2nd ed.). Boston: Addison-Wesley.

Ma, N., & Hall, T. (September 2006). Conflict Research, Call for Companies to Participate in a Research Project, Do we really understand the nature of conflict among stakeholders in requirements negotiation?. Requirements Quarterly, pp3-4, RQ41.

Newcomer, E., & Lomow, G. (2005). Understanding SOA with Web services. Upper Saddle, New Jersey:Addison-Wesley.

Nuseibeh, R., & Easterbrook, S. (2000). Requirements engineering: a roadmap. ACM and IEEE Computer Society Press.

Open Source Initiative. (2006, July 7). The Open Source Definition [online]. Available: http://www.opensource.org/docs/osd [accessed 9 May 2007].

Paech, B. & Reuschenbach, B. (2006). Open Source Requirements Engineering. In Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06) (pp. 252 – 259). Washington, DC: IEEE Computer Society.

Raymond, E. (2002). The cathedral & the bazaar, Musings on Linux and Open Source by an accidental revolutionary (2nd ed.). Sebastopol: O'Reilly and Associates.

Regnell, B., & Brinkkemper, S. (2005). Market-Driven Requirements Engineering for Software Products. In A. Aurum, and C. Wohlin (Eds.), Engineering and managing software requirements (pp.285-308). Berlin : Springer.

Robertson S, & Robertson J. (2006). Mastering the Requirements Process (2nd ed.). Addison Wesley.

Rossi, M., & Tuunanen, T. (2003). Marketing Meets Requirements Engineering. In Proceedings of the 11th IEEE International Conference on Requirements Engineering, Monterey Bay, CA, USA, 2003, p.341.

Sawyer, P., & Kotonya, G, (Eds.). (2004). Software Requirements. In P. Borque, & R. Dupuis (Eds.), Guide to the Software Engineering Body of Knowledge (pp. 2.1-2.18) [online]. Los Alamitos, CA: IEEE Computer Society. Available: http://www.swebok.org/ironman/pdf/SWEBOK_Guide_2004.

Scacchi, W. (2004). Free and open source development practices in the game community. IEEE Software, January/February, 59-66.

Shah, S.K. (2005). Open beyond software. In C. Di Bona, D. Cooper, and M. Stone (Eds.), Open Sources 2.0 (pp.339-360). Sebastopol, CA: O'reilly.

Sillitti, A., & Succi, G. (2005). Requirements Engineering for Agile Methods. In A. Aurum, and C. Wohlin (Eds.), Engineering and managing software requirements (pp.19-46). Berlin : Springer.

Sommerville, I., & Sawyer, P. (1997). Requirements Engineering, A Good Practice Guide. Chichester, West Sussex: John Wiley & Sons.

Srijith, K.N. (2002, October). Study on management of open source software projects, Report No. CS 5212 Software Project management Project Report – Research Paper [online]. National University of Singapore. Available http://srijith.net/publications/cs5212_report.pdf [accessed 3rd December 2007].

Taylor, A. (2001). IT projects sink or swim, BCS Review 2001 [online]. British Computer Society. Available: http://,www.bcs.org/review/2001/articles/itservices/projects.htm [accessed 16 December 2004].

Thomas, D., & Hunt, A. (2004, July/August). Open Source Ecosystems. IEEE Software, 89-91.

Von Hippel, E. (2007). Democratizing Innovation. Cambridge, Massachusetts; London:The MIT Press.

Warren J.H. (2004). Requirements and Formality, Requirenautics Quarterly. 33, 12.

Woollacott, P. (2007, March). Cybercrime comes of age. ITNow, 06-07.

Williams, S. (2002, March). Richard Stallman's Crusade for Free Software [online]. O'Reilly. Available: http://www.oreilly.com/openbook/freedom/ [accessed 9 May 2007].

Zowghi, D. & Coulin, C. (2005). Requirements Elicitation: A survey of Techniques, Approaches, and Tools. In A. Aurum, and C. Wohlin (Eds.), Engineering and managing software requirements (pp.19-46). Berlin : Springer.