Bridging the Research-Practice Gap in Requirements Engineering

Sarita Pais
Auckland University of Technology, New Zealand
grk0058@aut.ac.nz

Alison Talbot
Auckland University of Technology, New Zealand
alital66@aut.ac.nz

Andy Connor
Auckland University of Technology, New Zealand
andrew.connor@aut.ac.nz

ABSTRACT

This paper examines the perceived research-practice gap in software requirements engineering, with a particular focus on requirement specification and modelling. Various contributions by researchers to write requirements specifications in the literature are reviewed and in addition practitioners view points are also taken into consideration. On comparing the research and practice in this field, possible causes for the gap are identified. The barriers to adopt research contributions in practice are also reviewed. Finally recommendations to overcome this gap are made, which are the basis for an on-going study that aims to produce empirical results related to assessing requirements engineering capability in New Zealand.

Keywords

Software requirements engineering, Requirements specification.

1. Introduction

According to Zave (1997), Requirements Engineering (RE) is a process of translating the real world's informal state into a more formal specification. In essence, it is a process of creating and maintaining a system-requirement document to support the development of a given product. This view is expanded on by Nuseibeh & Easterbrook (2000) who assert;

> "The primary measure of success of a software system is the degree to which it meets the purpose for which it was intended. Broadly speaking, *software systems requirements engineering* (RE) is the process of discovering that purpose, by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, and subsequent implementation."

Although there are different activities in this process, the requirement specification sub-process is one in which system requirements are documented in a structured way. It captures users and other stakeholders' requirements in building a system. However, capturing these requirements is not an easy task. According to Brooks (1987) the most difficult task in building a system is what to build, and if this is not done properly it results in a poor system which is difficult to rectify later and will cost more. Moreover, users often do not know what they want and their needs keep changing (Hsia *et al*., 1994). In order to produce good user requirement there has been tremendous efforts from researchers to produce frameworks, models or tools to automate this process. However, these contributions from researchers have not been adopted completely by practitioners (Berry & Lawrence, 1998). Could it be that researchers have no input from practitioners and the software industry for appropriate research framework? Even if researchers are aware of the practitioners need, why are they not able to provide efficient tools or appropriate processes? Are there other issues that prevent the adoption of new tools or processes? Thus there is a gap between research and practice in Requirements Engineering.

This paper attempts to understand the issues underlying this gap and what can be done best to minimise this gap. In order to undertake this task, firstly the literature is searched where researchers have highlighted and stressed the gap, and then the sub-topic area of requirement specification is looked into. Here, the research directions are seen from the researchers' and practitioners' point of view, whether there is any input from practitioners to researchers. What are the different issues? Can these issues be solved and if not, why? Are there any recommendations to bridge this gap?

2. The Research-Practice Gap

Stating user requirements has been a thorny issue since before the time of Brooks (1987). The difficulty of this was again stressed in 1990's by Hsia *et al.* (1994), and it still remains a challenge today. Although the techniques to write requirements have been a research subject for some time, the results of such research has not been followed by the practitioners. The state of practice is requirements are often still written in natural language, despite the drawbacks of such a representation. Requirements are also seldom read, resulting in systems which do not satisfy the user. Lamsweerde (2000) briefed the on-going problems of RE which contribute to large scale project failure for US and European companies. It is true to this day that RE is still a complex task (Cheng & Atlee, 2007). Unlike other parts in software engineering, it has to cover both technical and human aspects (Faulk, 1997). Developers do not have the domain knowledge of the system to be built. Although there are significant CASE tools, Commercial off the Shelf (COTS) products, conceptual models, and Requirement Management tools produced through the insights of researchers to aid in practice. Not all are used by the practitioners for various reasons. It has been observed that "without more technology transfer, RE practice is unlikely to improve and much RE research will remain irrelevant" (Kaindl *et al.,* 2002).

Even if tools or methods are used, the guidelines for their use are not efficiently nor completely followed. This leads to the Research-Practice gap in RE in general. On the other hand, it is also noticed that practice has led to form theory in IS (Fitzgerald, 2003) like prototyping and defining the stages of system development life cycle. Is this trend also visible in RE and in particular requirement specification? Or do researchers have enough inputs? Is this gap significant even in the requirement specification? This shall be covered in the next paragraph.

Wieringa & Heerkens (2006) question whether the research community is responding well to this challenge. They hypothesize that, of the papers presented at conferences, what are called research papers are often design papers. And while that in itself is not a problem, they see that there are very few other papers in RE conferences. Davis & Hickey (2002) argue that many requirements engineering researchers fail to understand current practices. They conclude, citing Redwine & Riddle (1985), that "When we as requirements researchers lament that technology transfer takes a whopping 15 years, perhaps we should look no farther than ourselves."

2.1 New Zealand Perspective

Whilst the research practice gap is of concern in the global arena, it is important to consider the New Zealand perspective. While there is potentially 20 years worth of data, it is apparent that there has been little exploration by researchers as to what current practice is in the local software development industry, let alone that which focuses on Requirements Engineering. Only a limited number of articles that have investigated New Zealand companies' requirements analysis processes have been found in the literature (Groves *et al.* 2000a; Groves *et al.* 2000b; Kemp *et al.* 2003). In part this may be a reflection of Fitzgerald's (Fitzgerald 2003) assertion that less academic research value is placed on research that is published in practitioner oriented journals hence this is not considered a "worthwhile" area of analysis.

The first of the New Zealand studies (Groves *et al.* 2000a, Groves *et al.* 2000b) was carried out in 1999 and analyses data gathered from 24 telephone interviews and four in-depth interviews. The information recorded included items such as size of company, kind of developments undertaken, formality of the process and/or notation, proportion of project spent in requirement specification, standards in place within the organisation that the process is subject to, types of testing undertaken and tools and languages used. In the analysis of the results, no attempt was made to measure the effectiveness of the processes, or estimate the overall capability (as in measuring against the capability maturity matrix) of each company. The survey did provide some useful basic information relevant to the local industry and as such provides an excellent snapshot of practice at that time.

The second study (Kemp *et al.* 2003; Phillips *et al.*, 2005) investigated the broader topic of software engineering and tool support. It took the form of structured interviews with five New Zealand software developers and in addition to questions about tools, covered project lifecycle, and management. As with the previous survey, no attempt was made to measure practice against any form of capability model. However the authors did note

that all of their respondents had some form of formal project management in place, but that the degree of structure varied markedly. Unfortunately the information gathered referred to analysis/design activities rather than separating out requirements analysis/engineering tasks. In this respect the article is confined in its usefulness to providing evidence about the overall project process.

There still exists a pressing need to review and assess current software development practices in New Zealand, particularly in the area of requirements engineering. Similarly, there is scope to investigate whether international practices are appropriate for New Zealand companies to adopt. The outcomes of research related to companies similar to those typical in New Zealand, such as the work of Nikula *et al.* (2000) which addressed RE practices in Small to Medium Sized Enterprises need to be analysed for applicability in the cultural, social and technological environment of New Zealand. Other work, such as that of Damian & Zowghi (2003), which details requirements engineering challenges in multi-site projects (which included Australasian projects) may also be off use in informing future work.

To address this need, the authors will be undertaking a requirements engineering capability assessment of New Zealand software development companies as a means to identify shortcomings in current industry practice and identify how the gap between research and practice may be closed as a means to improve software development in New Zealand.

3. Requirements Specification

In the Requirements Engineering process, the development of a requirement specification commences after the initial elicitation of requirements. User requirements need to be documented well to ensure that proper instructions are available to developers to enable them to design and code an appropriate solution. This documentation could be in natural language. However developers prefer more formal specifications which can be directly useful to build the system. Researchers have contributed towards using conceptual models as used in System Analysis and Design and Object Oriented Analysis and Design. These models define a scope for RE to detect any oversight and inconsistencies. However the downside of this is, it reduce creativity in the RE process (Pohl & Peters, 1996). Moreover such models, like the Unified Modelling Language (UML), are not well understood by users (Kaindl *et al.*, 2002). Hence practitioners really do not follow a structured method in capturing requirements from users which is often reflected in poor performance of the developed system. In a survey conducted in over 3800 organisations in 17 European countries, it claimed that 50% of problems in software projects lie in the requirement specification (Lamsweerde, 2000). This is a grave situation and it is important to learn more about the research and practice in requirement specification. It is essential to understand what contributions have come from researchers and practitioners in this area, what differences and similarities there are between them.

Gorschek & Svahnberg (2005) looked at the requirements practices of 6 companies and found the following defects to be common in the current processes or documentation:

- Lack of standard templates or minimum set of attributes for specifying requirements;
- Quality requirements expressed in a form that is not testable;
- Lack of requirements review;
- Poor or no recording requirement and decision history.

These problems have already been addressed by both research and practitioner authors so this may indicate poor training or a reluctance within organizations for structured methods. It may also be that these companies are not truly “requirements ready”, and the defects in the requirements process are in fact a reflection of some other lack of capability in the company, particularly the ability to accommodate and adapt to change. Sommerville (2005) points out that, with experience, “the initial assumptions that underpinned much RE research and practice were unrealistic”. It is now understood that change is inevitable; no one can understand the whole problem before starting system development and new insights as development progresses lead to changes in requirements.

3.1. Research Perspective

Researchers differentiate RE as a process of what the system should do (Berry & Lawrence, 1998) while others stress for more formal specification as they have the qualities of good RE (Faulk, 1997). However, Nuseibeh and Easterbrook (2000) said that it is important to know when to formalise it, considering users favour natural language. The concept of modelling and specification is sometimes misunderstood (Machado *et al.* 2005). A conceptual model has to be converted with standard language to represent the system model to be called specification.

3.1.1 Unified Modelling Language (UML)

When structured programming was used in the past, structured analysis and design methods were employed to document the formal specification. Now with object oriented programming more in vogue, object oriented analysis and design methods are used (Faulk, 1997). With object oriented analysis, data and related process are kept together which was not the case in structured analysis. Then there are models for various purposes. Activity oriented specification can be depicted as DFD or using UML notations as use case and activity diagram. Data oriented specification can be ERD and class diagram (UML). The other good point here is that same models are used in RE which is easy for developers to understand and proceed with the system development. There is no rewriting at each step in the software development process. However the models at RE level may not contain all the merits to represent the user’s needs. Moreover the user is not familiar with these models.

3.1.2 Prototyping

Prototypes help in capturing the elicitation requirement leading to proper specifications (Pohl & Peters, 1996). This was also supported by Faulk (1997), as it takes less time and there is no need to write requirement specification. Hence prototype can serve the purpose, saving time and cost for quick development process. Moreover, users can understand and review the proposed system.

3.1.3 Commercial off the Shelf (COTS)

The integration of COTS products as a means to achieve a given functionality is again a useful model which reduces the overhead of the RE process (Nuseibeh & Easterbrook, 2000). They can be used as a checklist to cover all possible constraints. However, they may not be able to locate the root cause of any investigation in RE. Hence a qualitative approach is not possible.

3.1.4 Goals

Lamsweerde (2000) gave his contributions on identifying goals in RE and then converting them to more formal methods. Goals could cover both the functional and non-functional requirements. A goal can start from the elicitation stage from the business perspective and end up in a more formal and structured specification as in object oriented (Lamsweerde, 2003).

3.1.7 Scenario Based Models

Scenario based models are gaining importance in recent times (Cheng & Atlee, 2007). It is easy for practitioners and users with non technical background to put together their ideas and brainstorm on it, to refine requirements of the proposed system. SCRAM (Scenario-based Requirement Analysis Method) is one such method which also incorporates prototyping (Sutcliffe, 2003) within it. Moreover, Scenarios can be expressed informally using natural language and more formally in modelling languages (Rolland & Prakash, 2000).

3.1.8 Capability Maturity

Hall, Beecham & Rainer (2002) conducted an empirical analysis of 12 software development companies, with a view to identifying patterns in occurrences of problems related to requirements engineering. One of the conclusions of their work is that the number of problems generally tended to decrease as the maturity of the company increased. The Capability Maturity Model (Sawyer *et al.* 1999) appears to have been adopted as the de facto industry standard to measure the effectiveness of an organisations requirements engineering practice. This is evidenced by its use in such studies as that undertaken by Gorschek and Svahnberg (2005). Recently a newer model, the Requirements Engineering Maturity Measurement Framework (Niazi *et al.* 2007), was proposed. However it is still based on the Capability Maturity Matrix.

Sommerville & Ransom (2005) conducted an empirical study in industry of requirements engineering process maturity assessment and improvement that concluded that the RE process maturity model was useful in supporting maturity assessment and identifying potential process improvements. They also indicated that there was evidence to suggest that requirements engineering process improvement would lead to business benefits, though these benefits could be a consequence of the changes to the RE process or from side-effects such as greater self-awareness of business processes.

3.2 Practice Perspective

Practitioners generally do not differentiate much between the “what and how”, or between the end product and the actual process of building the system (Berry & Lawrence, 1998). They also do not waste much time in writing formal requirement specification (Faulk, 1997) as they are not practical. They are considered to be difficult to use, as users do not understand them. It costs them more of their tightly budgeted project time. Most tools developed by researchers are in prototype stage and have scalability issues when applied to real life systems. Hence practitioners have no trust in them. This view point as not changed much in recent times and is evidence from a survey listed below.

3.2.1 Mixed Results

According to a survey (Neill & Laplante, 2003) 35% of companies still use waterfall methodologies in RE. Although prototyping is not part of waterfall, 60% did it. 50% used scenarios and use cases. 51% used informal methods like natural language. 33% did not use any methodologies. The percentage of agile methodologies was negligible. It is very clear from these statistics that there is a mix of practices in industry. UML is still not a prominent methodology. As the statistics are from 2003, there might be a slight increase in the adoption of UML in RE. Again, this is the viewpoint of researchers and there are not many journals to support the viewpoint of practitioners. Normally practitioners know the good and bad points of the method they use and are not aware of other methods. It is researchers who make comparative studies of different methods and publish their work, but perhaps do not disseminate the results to practitioners.

3.2.2 Formal Methods

It is well known that RE is not time effective. However, this approach is changing as RE play a significant role in the success of finished product (Nuseibeh & Easterbrook, 2000). However there is no good reputation among stakeholders of using formal methods (Kaindl et al., 2002). In that case, it is not clear if formal methods are approved in the current practice. A lack of understanding of formal methods themselves may be a contributing factor to the lack of adoption.

3.2.3 Unified Modelling Language

UML is fast becoming as an emerging leader in analysis and design of software systems. These same models are widely accepted in RE too. Business use case is initially used to draw up the requirements from the users (Leffingwell & Widrig, 2000). They are kept simple for non-technical stakeholders. Many of the user's requirements may not be feasible. With the help of these formal methods users are now able to get a better insight into the proposed system. These specifications are also approved by developers as it is easy to convert these formal models into other stages in software development (Machado *et al.* 2005).

3.2.4 Agile methodologes

Agile methodologies are becoming more prevalent today, and in many implementations of the agile principles there is no requirement specification documents maintained (Cao & Ramesh, 2008). A practical prototype is built after face-to-face communication with the user. The prototype is reviewed and evolved. Users are part of the development process and need their continuous feedback for further progress. This could cripple development work sometimes as it is needs the continuous involvement of users.

Thus practitioners have their own various view points of requirement engineering. The next paragraph is an attempt to analyse research activities and industry practises. The reasons for the gap between research-practice in requirement specifications are identified and some recommendations to overcome this gap are suggested at the end.

4. Discussion

Thus far, various viewpoints of researchers and practitioners were put forward. It is now important to collate them together. Putting researchers work on one side and practitioners needs on the other side, a working solution has to be put forward. Before any solution is planned by researchers, it is important to know what works in practical world for the practitioners and what method is efficient to put together requirement specifications.

4.1 Needs of the Practitioners

Practitioners need a well managed RE tool. It should be ease to use. The whole process of RE should be cost effective to balance overheads, time and training (Cheng & Atlee, 2007). In other words it is a practical tool. Empirical studies from software companies should be used in RE Research to build tools that conform to these needs.

As complexity in software development has increased, it is important to involve users in its development (Faulk, 1997), especially requirement specification. As formal methods are not easy for users to understand, a tool which will capture natural language requirements into formal methods is desirable.

Requirement specification is not standardized (Faulk, 1997). Today UML is getting recognition as the de facto standard (Kaindl *et al.*, 2002). However, it not clear if UML is able to completely cover all aspects of RE.

Most of the practitioners work is published as white papers which has little credit as compared to research papers. Most of their work is not accepted by research publication and is biased against them (Fitzgerald, 2003).

4.2 Current Research Contributions

Researchers are aware of practitioners' needs. Attempts are made to automate RE by converting natural language specifications into formal methods, considering both the functional and non-functional specifications, building repositories of requirements to reduce time and cost. More details of these contributions are given below.

4.2.1 Repository of Natural Language Requirements

Natural Language is found to be the most convenient way to extract user's requirements. Although requirement engineers would prefer formal methods, they have to deal with users in natural language. Moreover formal methods can check internal consistencies. However, they fail to check external factors as they are user dependent. Therefore natural language is an appropriate solution. Hence it is important to keep a repository of such natural language requirements (och Dag & Gervasi, 2005). A matching requirement technique is searched same as in information retrieval. Challenges of matching the appropriate requirement in the repository is still in its infancy. Although linguistic engineering technique is used, there is a need for lexical match too. Hence there is a growing concern among researchers to develop a tool to convert natural language specification into formal methods.

4.2.2 Functional and Non-functional Requirements

Hsia et al (1994) put the importance of a unified framework dealing with both functional and non-functional requirements. However, most conceptual models can capture functional requirements but not non-functional requirements (Nuseibeh & Easterbrook, 2000). The Research community is currently focusing on this. Machado *et al.* (2005) came up with a solution for non-functional requirement like performance considering time factor which can be measured in sequence diagrams of UML. As these constraints are equally important to build good system, it has to be covered early on in the RE. Other examples are web based interfaces with poor security. This has to be an important concern too. Then there are other issues like usability and reliability of systems. Hence non-functional requirements need to be given equal importance as functional ones.

Although researchers have contributed to the needs of the practitioners, there is something still missing. Hence, it is still not accepted completely. The next section will illustrate the reasons for it.

4.3 Barriers

There are several reasons cited for the research-practice gap. Although researchers are aware of needs of the practitioners, there are other issues, some of which are not easy to implement. These issues were brought to light in the last two decades and yet a complete workable solution is not evident in literature.

4.3.1 No Collaboration

There is little or no collaboration between researchers and practitioners (Cheng & Atlee, 2007; Pohl & Peters, 1996). This view point has not changed to this day. The inputs to research do not reflect the issues of RE in practice. Even if researchers come up with a solution, the scalability of such research outputs is not covered (Pohl & Peters, 1996) due to non availability of industrial data or collaboration. Hence data with industrial strength should be employed in research (Cheng & Atlee, 2007).

4.3.2 No training

RE is not taught in-depth in many universities. Students have only some vague knowledge through software engineering. Hence there is a lack of well trained requirement engineers (Pohl & Peters, 1996). Lack of formal education has also been identified by do Prado Leite (2000) as a major obstacle for closing the research-practice gap in RE. Further training or technical support is not provided (Hsia *et al.* 1994) from the researcher once their work is published, and indeed many researchers will move on to the next big problem once they have reached a proof of concept stage in their current work. Hence, most practical outputs from their work is still in prototype. No further support given is by researchers to practitioners in choosing the correct tool (Pohl & Peters, 1996). Developers find the methods unsuitable and do not trust them (Hsia *et al.* 1994). Hence cannot be easily adopted on a commercial scale.

5. Recommendations

After realising the issues for the gap between research and practice in requirement specifications, a few recommendations are made to bridge this gap.

5.1 Empirical studies

The survey study (Neill & Laplante, 2003) mentioned earlier only gave the different choice of capturing requirement specifications as it was a survey study. There were no details of any issues in the current practice of the practitioners or any suggestions. Hence research should be conducted to get the essence of a qualitative study, involving case studies (Jiang *et al.* 2005) on several organisations. This will provide evidence to the state of practice on RE (Zave, 1997). This gives direct evidence of what is required by the industry to capture requirement specifications. However, it is difficult to generalise and apply to all projects. Kaindl *et al*. (2002) make a strong recommendation for more

research on the economics of RE to gain concrete knowledge of what organizations can gain from applying state of the art requirements approaches.

5.2 Automation tool

There is a need to find a tool which can automatically translate natural language specifications into formal ones (Zave, 1997). Even a partial translation could help (Hsia *et al.*1994) as most of the practitioners still prefer natural language for RE. Somé (2006) came up with a specification model collating all use cases, such that the specifications could be generated on the fly.

5.3 Repository

A database collection of all frameworks used to capture RE knowledge, especially dealing with specifications should be made available (Cheng & Atlee, 2007). There should be recommendations made to suit different projects. This will reduce the time factor in writing requirement specifications and an easy adoption of formal methods.

5.4 Collaboration

Most researchers are from an academic background, some do have some past experience in industry. However, the trend in the software industry is very dynamic as the industry is still not mature. Past experience in industry may provide little help for researchers to envision the current industry problems in RE. The development of tool and practice should go hand in hand (Lamsweerde, 2004). Hence, more continuous collaboration between researchers and practitioners will help to bridge this gap.

6. Conclusions

This report was an effort to highlight the gap between researchers and practitioners in RE especially in requirement specification area. Researchers have given formal methods while practitioners still prefer natural language to write the requirement specifications. After studying the needs of the practitioners and issues using formal methods, a few recommendations are made to narrow this gap.

References

Berry, D. M., & Lawrence, B. (1998). Requirement engineering. IEEE Software, 15(2), 26-29.

Brooks, F. (1987). No silver bullet: Essence and accidents of software engineering. IEEE computer, 10-19.

Cao, L., & Ramesh, B. (2008). Agile requirements engineering practices: An Empirical Study. IEEE Software, 25(1), 60-67.

Cheng, B. H. C. & Atlee, J. M. (2007). Research directions in requirement engineering. International Conference on Software Engineering, Future of Software Engineering, IEEE Computer Society Washington, DC, USA, 285-303.

Damian, D. & Zowghi, D. (2003) Requirements engineering challenges in multi-site software development organizations. Requirements Engineering Journal 8, 149-160.

Davis, A. M., & Hickey, A. M. (2002). Requirements researchers: Do we practice what we preach? Requirements Engineering, 7(2), 107-111.

do Prado Leite, J. C. S. (2000). Is there a Gap between RE research and RE practice? Proceedings of the Fourth International Conference on Requirement Engineering (ICRE 2000). IEEE Computer Society, 73-74

Faulk, S. R. (1997). Software requirements: A tutorial. Software Requirements Engineering, IEEE Computer Society Press, 1-22.

Fitzgerald, B. (2003). Informing each other: Bridging the gap between researcher and practitioners. Informing Science, 6, 13-19.

Gorschek, T., & Svahnberg, M. (2005). Requirements experience in practice: Studies of six companies. In A. Aurum & C. Wohlin (Eds.), *Engineering and Managing Software Requirements*. Berlin: Springer.

Groves, L., R. Nickson, et al. (2000a). "A survey of software development practices in the New Zealand Software industry." Proceedings of the International Australian Software Engineering Conference 2000.

Groves, L., R. Nickson, et al. (2000b). "A survey of software requirements specification practices in the New Zealand software industry." Proceedings ASWEC: 189-201.

Hall, T., Beecham, S., & Rainer, A. (2002). Requirements problems in twelve software companies: an empirical analysis. *IEE Proceedings on Software, 149*(5), 153-160.

Hsia, P., Davis, A. M., & Kung, D. C. (1994). Status report: Requirement engineering. IEEE Software, 10(6), 75-79.

Jiang, L., Eberlein, A., & Far, B. H. (2005). Combining requirements engineering techniques - Theory and case study. Proceedings of the 12th IEEE International Conference and workshops on the Engineering of Computer-Based Systems, ISBN:0769523080.

Kaindl, H., Brinkkemper, S., Bubenko, J. A., Farbey, B., Greenspan, S. J., Heitmeyer, C. L., et al. (2002). Requirement engineering and technology transfer: Obstacles, incentives and improvement agenda. Springer-Verlag, London, 7(3), 113-123.

Kemp, C., Phillips, E. A. & Alam, J. (2003). Software engineering practices and tool support: An exploratory study in New Zealand, Australian Journal of Information Systems, 11(1), 37-54.

Lamsweerde, A. (2000). Requirement engineering in the Year 00: A research perspective. Proceedings of 22nd International Conference on Software Engineering,(ICSE'2000): Limerick, Ireland, Invited Paper, ACM Press, 5-19.

Lamsweerde, A. (2003). Goal oriented requirement engineering: From system objectives to UML models to precise software specifications. Proceedings of the 25th International Conference on Software Engineering, Portland, Oregon, 744-745.

Lamsweerde, A. (2004). Goal oriented requirement engineering - A Round trip from research to practice. Proceedings of the 12th IEEE International Requirement Engineering Conference, Kyoto, Japan.

Leffingwell, D., & Widrig, D. (2000). Managing software requirements: A unified approach: Addison-Wesley.

Machado, R. J., Ramos, I., & Fernandes, J. M. (2005). Specification of requirement models. A. Aurum & C. Wohlin (Eds). engineering and Managing Software Requirements. Springer: Berlin.

Neill, C. J., & Laplante, P. A. (2003). Requirement engineering: The state of the practice. IEEE Software, 20(6), 40-45.

Niazi, M., K. Cox, et al. (2007). "A measurement framework for assessing the maturity of requirements engineering process." Software Quality Journal Volume 16, (Number 2 / June, 2007).

Nikula, U., Sajaniemi, J., & Kälviäinen, H. (2000). A State-of-the-Practice Survey on Requirements Engineering in Small-and Medium-Sized Enterprises. *TBRC Research Report, 1*.

Nuseibeh, B., & Easterbrook, S. (2000). Requirement engineering: A roadmap. ACM Computing Surveys, 35-46.

och Dag, J. N., & Gervasi, V. (2005). Managing large repositories of natural Language requirements. A. Aurum & C. Wohlin (Eds). Engineering and Managing Software Requirements. Springer: Berlin.

Phillips, E. A., Kemp, C. & Hedderley, D. (2005). Software development methods and tools: A New Zealand study, Australian Journal of Information Systems, 12(2), 21-49.

Pohl, K., & Peters, P. (1996). Workshop summary second international workshop on requirement engineering: Foundation of software Quality. ACM SIGSOFT Software Engineering Notes, 21(1), 31-34.

Redwine S, Riddle W. Software technology maturity. In IEEE eighth international conference on software engineering, 1985, pp 189–200

Rolland, C., & Prakash, N. (2000). From conceptual modelling to requirement engineering. Annals of Software Engineering, 10(1/4), 151-176.

Sawyer, P., Sommerville, I. & Villier, S. (1999). Capturing the benefits of requirements engineering. IEEE Software, 16(2), 78-85.

Some, S. S. (2006). Supporting use case based requirement engineering. Information and Software Technology, 48(1), 43-58.

Sommerville, I. (2005). Integrated requirements engineering: A tutorial. *IEEE Software* (Jan/Feb 2005), 16-23.

Sommerville, I., & Ransom, J. (2005). An empirical study of industrial requirements engineering process assessment and improvement. ACM Transactions on Software Engineering and Methodology (TOSEM), 14(1), 85-117.

Sutcliffe, A. (2003). Scenario-based requirements engineering. Proceedings of the 11th IEEE International Conference on Requirements Engineering, IEEE Computer Society, 320-330.

Wieringa, R. J., & Heerkens, J. M. G. (2006). The methodological soundness of requirements engineering papers: a conceptual framework and two case studies. Requirements Engineering, 2006(11), 295-307.

Zave, P. (1997). Classification of research efforts in requirements engineering. ACM Computing Surveys, 29(4), 315-321.

Copyright

You must insert the following Copyright notice at the end of your paper: