

Requirements Engineering and Software Project Success: An Industrial Survey in Australia and the U.S.

June Verner

*Empirical Software
Engineering*

*National ICT Australia
Australian Technology
Park*

june.verner@nicta.com.au

Karl Cox

*Empirical Software
Engineering*

*National ICT Australia
Australian Technology
Park*

karl.cox@nicta.com.au

Steven Bleistein

*Computer Science and
Engineering,
University of New South
Wales &*

*National ICT Australia
steven.bleistein@nicta.com.au*

Narciso Cerpa

*Departamento de
Ingeniería de Sistemas,
Universidad de Talca,
Camino Los Niches Km.
1, Curicó, Chile
ncerpa@utalca.cl*

Abstract

Because requirements engineering is recognized as critical to successful software projects we surveyed a number of software practitioners regarding their software development practices during recent software projects. Relationships between requirements practices and software project outcomes enable us to better understand requirements issues and their relationship with project success. We asked three sets of questions directly related to requirements issues: 1) requirements practices, 2) the sponsor and customers/users, and 3) project management. Our respondents were from business organizations in the U.S. and Australia, and were almost exclusively involved in in-house software development. The most significant factors from each question set were: 1) the requirements were good, 2) there was a high level of Customer/User involvement, and 3) the requirements were managed effectively. Overall, the best predictor of project success was that the requirements were good together with the requirements were managed effectively (93% of projects were predicted correctly). Our survey shows that effective project management is fundamental to effective requirements engineering.

1. Introduction

Requirements engineering (RE) can be simply described as identifying a problem's context, locating the customer's requirements within that context and delivering a specification that meets customer needs within that context. There are many requirements methodologies that purport to do this, for example, soft systems methodology [7], scenario analysis [6], and UML [3]. Sometimes they work, sometimes they do not. The implication of such requirements methodologies, if we can label at least aspects of them as such, is that the application of 'x' method will produce the right requirements irrespective of the problem's characteristics.

This is conventional wisdom and unsurprisingly, the creators and vendors of requirements methodologies claim, with one exception [19] that their approach is a hammer and all problems are nails. While there are many factors other than just application of a requirements methodology that influence the success or failure of software projects in practice, in this paper we focus only on requirements engineering. As Davis and Hickey state, we need as researchers, to be aware of what is really going on in practice to be able to position our research within that context [12]. Without this, we will forever practice our art in a context-free bubble.

To document practitioners' views regarding software project success or failure and the practices they consider important to successful projects, we conducted wide ranging structured discussions with twenty-one senior software practitioners employed by a large U. S. financial organization. This discussion focused on software projects with which the practitioners had been recently involved. We later had similar discussions with another group of software practitioners from the U.S who worked in a variety of other commercial organizations and also developed in-house software. Based on these discussions we developed a questionnaire in order to investigate those software development practices that lead to successful project outcomes. We chose a survey because of its simplicity and our wish to find relationships amongst variables; a survey gave us coverage of a greater number of projects at lesser cost than would an equivalent number of interviews or a series of case studies.

The original practitioner group responded to our questionnaire by answering it twice, once for a project they considered successful and once for a project they considered a failure. Our questionnaire was later distributed to the second group of practitioners. Then the questionnaire was answered by a group of Australian practitioners based in Sydney. Our sample is not random

but rather a convenience sample of practitioners known to us.

The questionnaire was organized into a number of sections covering the entire software development process. We also asked respondents if they considered the project they referenced, when answering the questionnaire, to be a success. Only questions relating to requirements engineering and requirements management are considered in this paper. Sections of the questionnaire, not considered here are discussed elsewhere e.g., [33, 34].

Our perspective could be broadly considered a review of in-house requirements management practices. This perspective is important because most software engineering research has emphasized “technical matters above behavioral matters” [17]. Moreover, there has been a general lack of quantitative survey-based research regarding early non-technical aspects of software development. In addition, in-house software development failure is unlikely to receive the same attention as third-party software development failures with their attendant litigation and negative media coverage. Our motivation is to show which requirements engineering practices are directly related to project success. The Standish reports on the state of software engineering practice indicate that requirements engineering is critical to software success [32].

When requirements are poorly defined and RE processes are *ad hoc*, the end result is nearly always an unsatisfactory product or a cancelled project. A Standish Group Report revealed that three of the top ten reasons for challenged projects or outright project failure were lack of user involvement, unstable requirements and poor project management [32]. A more recent survey of twelve UK companies found that requirements problems accounted for 48% of all software problems [18]. Another recent report from the U.K. stated that only 16% of software projects could be considered truly successful. “Projects are often poorly defined, codes of practice are frequently ignored and there is a woeful inability to learn from past experience” [20]. Another survey of 150 companies in the U.S. showed that the majority requirements modelling technique of choice was “none” [27].

In Section 2, we discuss some details of the questionnaire responses, in Section 3, the results of the questionnaire, and in Section 4, the implications of our results. Section 5 presents our conclusions and suggestions for further research.

2. Questionnaire Responses

We received completed questionnaires from 143 respondents, reporting on 164 distinct projects. As noted earlier, the majority of our respondents were developers involved with software for use within their own organizations (financial institutions, banks, pharmaceutical companies, insurance companies, etc.). The responses to the first set of 42 questionnaires described 42 projects, 21 regarded as successful and 21 unsuccessful. The second set of responses included descriptions of 80 unique projects reported from various companies in the northeastern U.S. The third set of responses, completed by developers working in Sydney, Australia, included descriptions of 42 unique projects

A sample of 164 projects is a reasonable size for empirical software engineering research. Sixty-six percent of projects were regarded as successful and 34% unsuccessful, 88% were development projects (63% successful), and 12% were large (in terms of effort) maintenance/enhancement projects (75% successful). The percentage of projects by number of full-time IT employees is 1-4 = 39%; 5-9 = 24%; 10-19 = 19%; 20-29 = 5%; 30-39 = 4%; 40-99 = 6%; and 100-180 = 8% (range 1-180, median 6).

3. Results and Analysis

The percentage of “yes” responses to the survey questions is shown in Table 1. Table 2 shows significant correlations with project success (<0.05) as well as some associations between responses to selected questions. We have classified our questions in these tables as follows: “C” refers to questions that deal with the project sponsor, customers and users; “R” to questions directly related to requirements; and “M” to questions related to the management of the development process.

3.1. Requirements Questions

Although good project management necessitates that requirements are complete and consistent [28], gathering requirements with a specific methodology (R1) was not significantly correlated with project success (Table 2). However, in 49% of our projects respondents did not know what requirements methodology was used. For the ones that did know, four projects used prototyping and eleven used JAD sessions with prototyping; for the remainder of projects, interviews and focus groups were the main requirements gathering method.

Table 1: Percentage “Yes” Responses to Questions

ID	Question	Success ¹ % Yes	Failure ² % Yes	Total ³ % Yes
C0	Were the stakeholders committed and involved?	66	57	63
C1	There was a high level of customer/user involvement	80	47	69
C2	There was a high level of customer/user confidence in the development team	70	29	56
C3	There was a low level of customer/user turnover	73	55	65
C4	Senior level project sponsorship lasted right through the project	80	50	70
C5	You were affected by large numbers of customers/users	29	33	30
R1	Were requirements gathered using a specific method?	53	50	52
R2	Were requirements complete and accurate at project start?	47	25	40
R3	If not complete at start were requirements completed later?	80	23	56
R4	Overall, were the requirements good?	81	28	66
R5	Did the project have a well defined scope?	81	46	69
R6	Did the scope increase during the project?	61	74	66
R7	Customers/users made adequate time available for requirements gathering?	80	42	66
R8	Was there a central repository for requirements?	77	44	66
R9	Did requirements result in well defined deliverables?	79	37	64
R10	Did the size of the project have a negative impact on requirements?	31	52	38
M1	The requirements were managed effectively	86	35	64
M2	Was the project manager experienced in the application area	69	69	69
M3	Was a defined development methodology used?	73	50	66
M4	Was the methodology appropriate for the project?	81	46	65
M5	Was delivery decision made with appropriate requirements information?	67	20	51
M6	The project manager was able to choose the development methodology	41	25	34

Thirteen of the 15 projects using prototyping and/or JAD were successful. Eight U.S. projects used UML to document requirements; only four were successful; practitioners commented that there were “too many new things without a pilot” and “unfamiliar methods used”. This indicates that either UML imposes what might be considered non-requirements notations upon the requirements, or that some developers were unfamiliar with UML. Also, the failed UML projects all had other problems, e.g., poor estimates, and no risk management, so their failures were not necessarily only due to use of UML. No Australian projects used UML

What appears to be more important than a defined *requirements* gathering methodology (R1) is that the project has a defined software *development* methodology (M3) that is *appropriate* for the project (M4), as both of these variables were significantly correlated with project success. Surprisingly, one-third of projects did not have a defined development methodology.

Nearly half the projects began with incomplete requirements (R2). It is therefore not surprising that the scope was changed for many projects (R6); a χ^2 test of R2 with R6 was significant. The scope was also more likely to change for larger projects.

¹ This column represents the percentage of “yes” answers to questions for successful projects.

² This column represents the percentage of “yes” answers to questions for projects that were failures.

³ This column represents the percentage of “yes” answers to the questions for all projects.

Table 2: Correlations of Questions to Project Success and to Other Questions

ID	Question	Direction of Success Relation -ship	Significant Correlation with Project Success	χ^2 Correlation with Other Questions
C0	Were the stakeholders committed and involved?			C4
C1	There was a high level of customer/user involvement	+	0.000	C2, C4, R5
C2	There was a high level of customer/user confidence in the development team	+	0.000	C1, R5
C3	There was a low level of customer/user turnover	+	0.019	
C4	Senior level project sponsorship lasted right through the project	+	0.000	C0, C1, R5
C5	You were affected by large numbers of customers/users			
R1	Were requirements gathered using a specific method?			M4
R2	Were requirements complete and accurate at project start?	+	0.006	R6 (-), M4, R5
R3	If not complete at start were requirements completed later?	+	0.000	M4, R5
R4	Overall, were the requirements good?	+	0.000	M4
R5	Did the project have a well defined scope?	+	0.000	M1, M3, M4, M5, C1, C2, C4, R2, R3, R4, R6 (-), R7, R8, R9, R10 (-)
R6	Did the scope increase during the project?			R2(-), R5 (-)
R7	Customers/users made adequate time available for requirements gathering?	+	0.000	M4, R5
R8	Was there a central repository for requirements?	+	0.000	M1, M4, R5
R9	Did requirements result in well defined deliverables?	+	0.000	M4, R5
R10	Did the size of the project have an impact on requirements?	-	0.000	R5 (-)
M1	The requirements were managed effectively	+	0.000	R8, M4, R5
M2	Was the project manager experienced in the application area			
M3	Was a defined development methodology used?	+	0.007	M4, R5
M4	Was the methodology appropriate for the project?	+	0.000	R1,R2,R3,R4,R5,R7, R8,R9,C4,M1, M5
M5	Was delivery decision made with appropriate requirements information?	+	0.000	M4, R5
M6	The project manager was able to choose the development methodology			

Our results, shown in Tables 1 and 2, indicate that requirements continue to be a big problem for software development [26, 31] and one of the most common causes of runaway projects [16]. Given that control over requirements is necessary to move from the lowest CMMI level, it is clear that many of the organizations in our sample are still at the lowest level [9]. These results agree with [27], whose respondents thought that their companies did not do enough requirements engineering. While sixty percent of projects began with poor requirements, fewer than 10% of projects used a development methodology designed to deal with unclear requirements.

Not surprisingly, and consistent with observations made by Glass [15], we found that good requirements (R4), that were complete and accurate at the start of the project (R2), with a well-defined project scope (R5), resulting in well-defined software deliverables (R9), were all positively correlated with project success. The importance of user involvement in requirements gathering (R7) supports the observations of both Clavadetscher [8] and Glass [15]. We found that if requirements were initially incomplete, completing the requirements during the project (R3) was positively correlated with project success. Although Boehm [2] includes a “continuing stream of requirements changes” in his top ten risk items, we did not find that changing the scope during the project (R6) was correlated with project failure. Also, being able to effectively manage requirements and any changes to them (M1) through a central repository (R8) was positively correlated with project success. The fact that only 66% of our projects used a central repository supports the suggestion that “we fail to use requirements management to surface (early) errors or problems” [8].

When the size of a project impacted on requirements gathering (R10), project failure was more likely. This result agrees with [15], suggesting that project size hampers requirements gathering, and leads to unclear, incomplete, and potentially unstable requirements. Large numbers of customers and users had no significant impact on project failure.

Using logistic regression with the responses to the requirements questions, the best predictor of project success was R4 (*the requirements were good*) which predicted 89% successes, 58% failures, and 78% of projects correctly overall.

3.2. Sponsor, Customer and User Questions

A project that has customers/users who have a low turnover rate (C3), who have confidence in the development team (C2), and who have a high level of involvement in the project (C1), is likely to be a success. However, having a large number of customers and users (C5) was not correlated with project failure. Evidence shows that a high level of customer/user involvement right through the project from requirements elicitation to acceptance testing is necessary for project success [32]. The correlation between customer/user involvement (C1) with level of confidence they have in the development team (C2) is interesting and leads us to ask about causal effects. “Are customers/users involved because they are confident in the development team or if they are involved do they become more confident in the development team?” We suspect that the answer is the former. This leads us to suggest that development teams who do not present themselves well to users and manage customer/user expectations properly may be sowing the seeds for failure.

We were not surprised that a high degree of senior level sponsorship that lasted right through the project (C4) was significantly related to (C0) committed and involved stakeholders and (C1) a high level of customer/user involvement.

Using logistic regression with responses to the sponsor, customer and user questions, the best predictor of project success was C1 (*there was a high level of customer/user involvement*), with C2 (*there was a high level of customer/user confidence in the development team*) which predicted 90% successes, 51% failures and 78% correctly overall. On its own C2 (*there was a high level of customer/user confidence in the development team*) predicted 70% projects correctly overall.

3.3. Project Management Questions

A PM experienced in the application area (M2) was not correlated with project success. “Successful project managers are generalists, not technical specialists”; while a certain level of technical competence is helpful, managerial and interpersonal skills are more important [21].

A project that has a PM who manages requirements effectively (M1), and uses a well defined software development methodology (M3) that is appropriate for the project (M4) and that has estimates of effort and schedule made with appropriate requirements information

(M5) is likely to be successful. Good estimates of effort and schedule (C4) have a huge effect on project success [13]. As early as 1975 Brooks stated that more projects have gone awry for lack of calendar time than from all other causes combined [5]. Optimistic estimation is still one of the two most common causes for runaway projects [16] with cost and schedule failures exceeding any other kinds of software failures in practice [14]. Boehm [2] includes unrealistic schedules and budgets in his top 10 risk items.

Using logistic regression with the responses to the project management set of questions, M5 (*making delivery decisions with appropriate requirements information*), with M4 (*the development methodology was appropriate for the project*) and M1 (*the requirements were managed effectively*) was the best predictor of project outcome, predicting 86% successes, 77% failures, and 83% correctly overall. On its own M1 (*the requirements were managed effectively*) predicted project outcome correctly for 77% of projects. This result supports Davis, who claims that requirements triage is critical: determining which requirements a product must have given a time constraint and resources available within that time frame [11].

3.4. Important Correlations

The most important project success prediction factors are that the requirements were good (R4) and that the requirements were managed effectively (M1). These two factors alone correctly predicted 93% of successful projects. Having good requirements is highly correlated with a high level of customer/user involvement. It is difficult to get good requirements without customer/user involvement.

We investigate two factors more thoroughly since they are discussed little in the requirements research literature. These are (R5) did the project have a well-defined scope, and (M4) was the development methodology appropriate for the project? As shown in Table 2, both have significant correlations with many other factors. Note that there are also many other significant correlations that we have not discussed in this paper nor are shown in Table 2.

3.4.1. Scope

A well-defined scope is critical to project success. We found that scope was significantly, positively correlated with a number of factors:

- C1, a high level of customer/user involvement. Without this level of customer/user involvement, it is

not easy to identify the problem to be solved. Without this identification it is impossible to define a project's scope. Asking, "what functions do you want?" and not asking, "what is this system for, who's involved?" is not likely to help define scope accurately. You can only ask these questions throughout the project when you have a high level of customer/user involvement.

- C2, there is a high level of customer/user confidence in the development team. C2 is significantly correlated with C1. It is interesting that this is an important factor though not particularly well addressed in the research literature. Without a high level of confidence, one is less likely to elicit the right scope and from this weak starting place one is less likely to elicit the right requirements.
- C4, senior project sponsorship lasted right through the project. This is a critical success factor. High level support induces greater cooperation that could be missing without such sponsorship. A high level sponsor may also be more aware of the wider scope of the project's impact.
- R2, the requirements were complete and accurately defined at the start of the project. There is a natural correlation to scope. With inaccurate scope or unmanaged scope creep, it will be difficult to identify a complete requirements set.
- R3, the requirements were completed at some stage in the project. Similar to R2, a well-defined scope even if it creeps, can still allow a complete requirements set at some point during development. So long as the project chunk being worked on at one time has well-defined scope and the requirements are complete, then project success is more likely.
- R4, overall the requirements were good. Given a well-defined scope, it should be easier to identify all the necessary requirements, i.e. requirements were good.
- R6, did the scope increase during the project? This is negatively correlated; that is, the more scope increased, the less likely it was to be well-defined.
- R7, customers/users made adequate time available for requirements gathering. Exploration of the problem space with customers and users who have time to discuss this allows for better scoping of the project and of manageable chunks for development.
- R8, there was a central requirements repository. This is a critical success factor. It is entirely necessary to have one, and only one, repository to store the requirements. This, of course, aids in scoping the project. It is easy for the development team to see the scope of their project and know that it is the agreed scope project-wide.

- R9, the requirements resulted in well-defined deliverables. This is often difficult to do without a well-defined scope simply because the goal posts may keep shifting.
- R10, the size of the project had an impact on requirements. This is negatively correlated; that is, the larger the project, the more important it is to define scope. It is also much more difficult to achieve.
- M1, the requirements were managed effectively. A well-defined scope and decomposition of the project into related, manageable requirements chunks is difficult. Good project management and in particular, requirements management, is essential for a successful project outcome.
- M3, a defined development methodology was used. A development methodology appropriate to the problem enables a better scoping of the requirements in that there is more likelihood that the project is scoped according to the relevant aspects of the defined methodology. As an example, all requirements relating to an information system will be scoped together to fit an information systems method within the wider methodology.
- M4, the methodology was appropriate for the project. This is very similar to M3 above. A project's parts are significantly better scoped if the development methodology of choice is appropriate to the project's parts.
- M5, the delivery decision was made with appropriate requirements information. A well-defined scope will significantly improve the success of delivery decisions because without this knowledge it will be difficult to know what can be delivered as a complete piece of work within the project.
- R2, the requirements were complete and accurate at the start. Naturally, complete requirements allow for an identification of sub-problem types within the a project, and then a choice of the appropriate methodology becomes more apparent.
- R3, requirements were completed at some point in the development. As for R2, understanding aspects of the problem allows the right choice of method for that problem part. So though requirements might not be completed at the start, awareness of the types of problems being addressed allows for choice of the right methods.
- R4, overall the requirements were good. Requirements are often better achieved when they are developed using the appropriate method.
- R5, see M4 in section 3.4.1.
- R7, customers/users made adequate time for requirements gathering. When this occurs, it is easier to get the right requirements, to understand the problem and then to select the appropriate methodology.
- R8, there was a central repository for requirements. This helps the appropriate selection of methodology simply because there is one location to look for requirements and therefore one place to organize the requirements appropriately. It is easier to select the methodology based on this structure and single point of information.
- R9, did the requirements result in well-defined deliverables? An appropriate methodology and an appropriate, well-defined scope, allow for well-defined deliverables that are actually delivered according to their scope as defined.
- M1, the requirements were managed effectively. Requirements management is part of project management. Methodological selection is simplified through good requirements management as it is easier to understand the problem to be solved, and from there select ways to do that appropriately.
- M5, delivery decisions were made with the appropriate requirements information. This is correlated with an appropriate methodology. It is much easier to make these kinds of decisions when you can trust the approach you are using for development.

3.4.2. Appropriate Methodology

An appropriate lifecycle development methodology is shown to be significantly correlated with project success. There is some literature to support this notion, for instance [19], though it appears vendors are happy to assume their one-size-fits-all does indeed fit. An appropriate methodology, M4, is significantly correlated with:

- C4, senior level sponsorship lasted right through the project. A senior sponsor can enforce the right methodology and can equally defend a project manager or developer's choice of methodology. This support is important to successful uptake of the approach.

There are many factors other than those we have discussed above. You can't make accurate delivery schedules without scoping your project. You can't get this information except for a combination of factors, including budget, which we have not considered at all. Although this discussion appears simple, it is more complex than we portray. Politics, for instance, is

something almost entirely ignored in the requirements research literature. In the workplace, it is a highly significant factor to what requirements are actually delivered and what methodology is selected.

4. Discussion

The developers we surveyed mainly develop in-house software for their organization's use. Their organizations have a heavy reliance on software for many business functions. While we would not assume that our results are typical of all organizations, we believe that they are reasonably typical of organizations that develop in-house software. Surveys are of course based on self-reported data which reflects what people say happened, not what they actually did or experienced. Because we surveyed software developers our results are limited to their knowledge, attitudes, and beliefs regarding the projects and PMs with which they were involved. However, as the majority of projects are fairly small (63% employed fewer than 10 people and 84% fewer than 20), we believe that our respondents have a reasonable knowledge of most project events. The overall preponderance of small projects may however, bias our results.

Overall, the best logistic regression prediction equation using data from each of the three groups of questions, was R4 (*overall the requirements were good*) with M1 (*the requirements were managed effectively*) which predicted 93% successes, 59% failures and 83% correctly overall.

We were surprised that so many projects started (and continued), with unclear requirements. Why are PMs prepared to go ahead with projects without either appropriate requirements or a development methodology able to deal with unclear requirements? It is common knowledge that good requirements lead to software development success so why are PMs apparently so unaware that they are prepared to jeopardize project success in this fashion? Poor requirements have a negative effect on the estimation process; this then leads to schedule and cost underestimates, inadequate staffing and then staffing itself becomes a major risk factor.

Many project management problems are in fact requirements problems in disguise, particularly those related to scheduling and effort estimation. The results suggest that senior management needs better education regarding the importance of adequate requirements, and that good requirements are necessary to produce appropriate schedule and effort estimates.

While some consider that using UML for requirements modeling and management is helpful, in this research we find no supporting evidence. To the contrary, in at least one project the use of UML was forced upon the development team with no accompanying training; project failure was the outcome.

It might be important to distinguish between scope creep and requirements creep more clearly. Evolving requirements throughout a project tends to have no significant impact on success as long as requirements were considered complete at some point during the project. In contrast, scope increase was not correlated with project success. Perhaps scope ought to be defined as the boundaries of the problem domain within which to seek requirements. It is important that these boundaries be defined clearly early in the project, whereas the requirements within those boundaries may evolve continuously.

Finally, having a central repository for requirements is clearly correlated with project success. This is good news, because it is relatively easy to do. In fact, it is difficult to understand why a project would not have a central repository for requirements given the technology available today.

5. Conclusions and Further Research

We discovered that:

- 1) it is not the number of users involved that is important, but rather managing the size of the project in terms of functionality;
- 2) it is not the requirements methodology per se, but rather use of an appropriate software development methodology into which the requirements methodology fits;
- 3) it is not scope creep, but rather that scope is well defined when it creeps;
- 4) it is not a project manager experienced in the application area, but rather a project manager who manages requirements effectively;
- 5) it is not necessarily having complete requirements at the start of the project but rather completing the requirements at some stage during the project; and
- 6) projects that had a central repository for requirements were more likely to succeed.

The most important correlations for project success are to get good requirements and to manage those requirements effectively. Getting good requirements means a number

of things. Some that are important are a high level of customer/user involvement, high-level sponsorship throughout, to scope the project effectively and it is critical to have a good project manager who can manage, rather than one who just happens to know the application domain.

Table 1 shows that current practices are fair at best. There is much opportunity for improvement at the start of a project in the requirements area. This is very important if we wish to increase the quality and success of our software projects. Analysis of our survey suggests further research is required in order to investigate:

- What kinds of pressures lead project managers not only to start projects with poor requirements, but also to actually complete them without really knowing what the requirements are?
- How might the requirements analysis activity be better integrated with scheduling and cost estimation?
- It may be important to distinguish more clearly between project requirements versus project scope. Is it possible that a good definition of scope at the outset of a project enables project teams to better deal with loosely defined requirements that later evolve?
- Customer involvement and customer confidence in the project team indicate better likelihood of success. How are these interrelated? Do customers become more involved because they are confident in the team, or are they confident because they are involved? What motivates customer involvement and confidence?

This research serves as a starting point in motivating continuing research in requirements practice in industry and project success factors. We intend to continue with this research in the future.

References

- [1] Boehm B. W., *Software Engineering Economics*, Prentice Hall, Englewood Cliffs, NJ (1981).
- [2] Boehm B. W., "Software Risk Management Principles and Practice", *IEEE Software* Vol. 8 No 1, 1991, pp32-41
- [3] Booch, G., Rumbaugh, J., Jacobson, The UML User Guide, Addison Wesley, 1999.
- [4] Brooks F. P. Jr., *The Mythical Man Month. Essays on Software Engineering*, Addison Wesley, USA (1975).
- [5] Brossler P. "Knowledge Management at a Software House: A progress report" *Proc Workshop on Learning organizations* (1999), pp 77-83.
- [6] Carroll, J. (ed.), *Scenario-Based Design: Envisioning Work and Technology in System Development*, Wiley Publications, 1995.
- [7] Checkland, P., *Systems Thinking, Systems Practice*, Wiley Publications, 1981.
- [8] Clavadetscher, C. "User Involvement: Key To Success", *IEEE Software*. Vol. 15, Issue 2 (Mar/Apr 1998) p. 30, 32.
- [9] CMMI <http://www.sei.cmu.edu/cmmi/> accessed 5th July 2004
- [10] Collier B, T. DeMarco and P. Fearey, "A Defined Process for Project Postmortem Review", *IEEE Software*, Vol. 13, No. 4 (1996), pp. 65-72
- [11] Davis, A., 2003, "The Art of Requirements Triage", *IEEE Computer*, March, pp42-49
- [12] Davis, A., and A. Hickey, "Requirements Researchers: Do We Practice What We Preach?", *Requirements Engineering Journal*, 2002, 7, pp107-111.
- [13] DeMarco T. and T. Lister, *Waltzing With Bears*, Dorset House Publishing New York NY, 2003
- [14] Glass R "Error-Free Software Remains Extremely Elusive", *IEEE Software* January/February, 2003, pp103-104.
- [15] Glass, R. L. "How Not To Prepare For A Consulting Assignment And Other Ugly Consultancy Truths", *Communications of the ACM*. Volume 41, Issue 12 (December 1998) pp11-13.
- [16] Glass R. L., "Frequently Forgotten Fundamental Facts about Software Engineering" *IEEE Software*, May/June 2001, pp112, 110, 111.
- [17] Glass R. L., "Project Retrospectives, and Why They Never Happen", *IEEE Software* 2002, September/October, pp112, 111.
- [18] T. Hall, S. Beecham, and A. Rainer, "Requirements problems in twelve companies: an empirical analysis" *IEE Proceedings Software*, 149 (5), pp153-160, 2002
- [19] Jackson, M., *Problem Frames*, Addison Wesley, 2001.
- [20] Jobserve.com, "UK Wasting Billions on IT Projects", <http://www.jobserve.com/news/NewsStory.asp?e=e&SID=SID2598>, 21/4/2004.
- [21] Jurison, J., *Communications of AIS, Software Project Management, The Manager's view* (1999) Volume 2, Article 17.
- [22] Kerth, N. L., *Project Retrospectives: A Handbook for Team Reviews*, Dorset House Publishing, New York 2001.
- [23] Keuffel, W., "Planning for and mitigating risk" *Software Development* 7 9 (1999) pp81-85
- [24] Kwak, Y. H. and Stoddard J., "Project risk management: Lessons learned from software development environments", in press, to appear in *Technovation*, 2004
- [25] Kwak, Y. H. and Ibbs, C. W. "Calculating project management's return on investment" *Project Management Journal* 31, 2, 2000 pp38-47

- [26] Moynihan T., "How experienced Project Managers Assess Risk", IEEE Software, Vol 14, (MAY/JUNE 1997) pp35-41.
- [27] Neill C. J. and Laplante P. A., Requirements Engineering: State of the Practice" IEEE Software, November/December 2003 pp 40-45.
- [28] Osmundson J. S., Michael J. B., Machniak, M. J., and Grossman M. A., "Quality management metrics for software development" accepted for Information and Management, 2003.
- [29] Phan, Dien D., Douglas R. Vogel and Jay F. Nunamaker Jr., "Empirical studies in software development projects: Field survey and OS/400 study, Information & Management 28 (1995) pp271-280
- [30] Pressman, R. Software Engineering: A Practitioners Approach, McGraw Hill (1996).
- [31] Schenk, K.D and Vitalari, N. P., et al. "Differences Between Novice and Expert Systems Analysts: What Do We Know and What Do We Do?", Journal of Management Information Systems, Vol. 15, Issue 1 (Summer 1998) pp9-51.
- [32] Standish Group "Chaos: A Recipe for Success", Standish Group International, 1999
- [33] Verner J. M. and Evanco W. M., "A Investigation into Software Process Knowledge" in Managing Software Engineering Knowledge, (eds) A. Aurum, R. Jeffery, C. Wohlin and M. Handzic, Springer-Verlag, 2003, pp29-47.
- [34] Verner J. M. and Evanco W. M., "In-house Software Development: What Software Project Management Practices Lead to Success?" IEEE Software Jan/Feb vol. 22, issue 1, 2005, pp86-93