What factors lead to software project failure?

June Verner
NICTA
Alexandria
Sydney
Australia
june.verner@nicta.com.au

Jennifer Sampson
NICTA
Alexandria
Sydney
Australia
jennifer.sampson@nicta.com.au

Narciso Cerpa
University of Talca
Talca
Chile
n.cerpa@utalca.cl.

Abstract **- It has been suggested that there is more than one reason for a software development project to fail. However, most of the literature that discusses project failure tends to be rather general, supplying us with lists of risk and failure factors, and focusing on the negative business effects of the failure. Very little research has attempted an in-depth investigation of a number of failed projects to identify exactly what are the factors behind the failure. In this research we analyze data from 70 failed projects. This data provides us with practitioners' perspectives on 57 development and management factors for projects they considered were failures. Our results show that all projects we investigated suffered from numerous failure factors. For a single project the number of such factors ranges from 5 to 47. While there does not appear to be any overarching set of failure factors we discovered that all of the projects suffered from poor project management. Most projects additionally suffered from organizational factors outside the project manager's control. We conclude with suggestions for minimizing the four most common failure factors.**

Index terms
software project failure, software project management, failure factors, project risk

I. INTRODUCTION

Most of the literature that discusses project failure is based on a handful of failed project case studies, although Charette [4] lists 31 failed projects and discusses three of these projects in more detail. Most of the research on failed software projects identifies various factors that lead to failure, but this is usually done in a rather generalized fashion; the reason for the failure of a specific project is often glossed over; rather it is the effect of the failure that is described in detail, e.g., the failure of project X resulted in a loss of $350 million dollars and as a consequence the company went into receivership. Both Charette [5] and Yardley (cited in [6]) agree that there is generally more than one reason for the failure of a software project; it is usually a combination of technical, project management and business decisions, and each dimension interacts with the others in complicated ways. Project managers (PM) and development teams must deal with many pressures from project stakeholders (i.e., upper level management, marketing, accounting, customers, and users) during the software development process. These pressures impact the effort, cost and quality of the software produced.

Project failures affect software development staff because late projects usually cause developers to suffer long hours of unpaid overtime that impacts on their personal lives, leading to loss of motivation, and stress, resulting in high staff turnover and its associated costs [9]. High staff turnover is itself associated with project failure.

Many software project failure factors have been described in the literature [1]-[17], including:

- organizational structure,
- unrealistic or unarticulated goals,
- software that fails to meet the real business needs,
- badly defined system requirements, user requirements and requirements specification,
- the project management process, poor project management,
- software development methodologies, sloppy development practices,
- scheduling and project budget,
- inaccurate estimates of needed resources,
- poor reporting of the project status,
- inability to handle project complexity,

- unmanaged risks,
- poor communication among customers, developers and users,
- use of immature technology,
- stakeholder politics,
- commercial pressures,
- customer satisfaction,
- product quality,
- leadership, upper management support,
- personality conflicts,
- business processes and resources,
- poor, or no tracking tools.

We are interested in providing, from the perspective of software practitioners, quantitative evidence identifying those aspects of the development process that contribute to project failure. Do projects fail because of a single overwhelming factor or is it true that they fail only when there is a combination of factors as some researchers have suggested? How many factors must be in conjunction before a project becomes a failure, and are there any common patterns in those factors?

The rest of this paper is organized as follows. In the next section (Section 2) we discuss some of the background to our research; Section 3 discusses the data we collected, methodology used and our data analysis; Section 4 reviews our findings, while Section 5 covers limitations to this work. We conclude (Section 6) with lessons learned and further research.

II. BACKGROUND

Developing software systems is an expensive, and often a difficult process as software development projects are affected by a series of problems, such as poor project management, cost and schedule overruns, poor quality software and under-motivated developers [1], [3].

Boehm suggested in 1991 that realistic schedule and budgets together with a continuing steam of changes in requirements are high risk factors for software development projects [2]. The Standish Group indicated in 1994 that approximately 31% of corporate software development projects are cancelled before completion and 53% cost nearly 200% of their original estimate [18]. Glass called challenged projects "runaway projects" when he discussed 16 project disasters in 1998 [6]. He found that the failed projects he reviewed were mostly huge, and that the failure factors were not just management factors but also included technical factors. Linberg, in 1999, found that 20% of software projects failed, and that 46% experienced cost and schedule overruns or significantly reduced functionality [12]. Glass revisited failed projects in 2002 and found that poor estimation was high on his list of failure factors [7]. He suggested that we do estimation very badly and that most of our estimates are more like wishes than realistic targets [7]. To make matters worse, we seem to have no idea how to improve on these bad practices. The result is an impossible estimation target, with shortcuts taken, and good practices skipped. "The evil twin of optimistic estimation", and "unstable requirements" result from customers and users only having a vague idea of the required solution at the outset of a project when initial estimates need to be done [8]. Naturally this makes it difficult for estimators and developers. It is commonly accepted that requirements change throughout a project and the system's life cycle. Managing these changes becomes a challenge [7]. Other studies, suggest various failure rates for software development projects up to 85% [10]. In 2006 the Standish Group suggested that 67% of IT projects fail or are challenged (cited in [15]). In 2007 Sauer et. al. [17] suggest that the situation is not so bad and in fact 67% of the projects they investigated in the UK delivered close to budget, schedule and scope expectations.

Recently Charette noted that "billions of dollars are wasted each year on failed software projects" and that "we have a dismal history of projects that have gone awry" [4], Charette provides a long list of high profile failed projects from around the world in his "Hall of Shame" and suggests that from 5%-15% of projects will be abandoned before or shortly after delivery as hopelessly inadequate [4]. Most IT experts agree that such failures occur far more often than they should.

Although there are many guidelines for successful software development (e.g., [6], [12], few project post-mortems are conducted, and little understanding is gained from the results of past projects within organizations. Most project failures are predictable and avoidable though it is not always possible to identify what success and failure factors are important, early enough to take action.

If we examine the literature we find some recent research that investigates failed software projects and provides details of the factors believed to be behind the project failures [5], [12], [19]. However, if we carefully study these projects most authors don't provide enough detail about them, and leave us with a number of questions regarding who or what was really behind the failure. Were

these problems caused by poor project management, senior management interference/lack of support, or should both groups share the blame? In the following paragraphs we speculate about the reasons for the failures discussed.

Charette [5] describes Arianespace, a failed project that suffered from a number of specification and design errors. This immediately suggests two things: 1) a problem with the requirements, and 2) requirements changes leading to problems with the design; but we are not given any information about the time spent on requirements gathering, who was involved, what method was used, how much time was spent and what the state of the requirements was, when design was started, or requirements volatility. In addition we don't know what software development methodology was used. Design errors and specification errors are symptoms of an underlying problem which is not discussed. High level management may be behind this failure but it is more likely a project management problem.

A Sydney Water Board project [5] failed because 1) rework costs exceeded the value-added work; 2) inadequate planning and specifications led to numerous change requests; and 3) cost and schedule overruns. These problems all probably stem from inadequate requirements. Rework is usually caused by requirements changes, so excessive rework costs are likely due to problems with the initial requirements. Inadequate planning and specification leading to numerous change requests, is also probably caused by inadequate requirements at the start of the project. You can't plan adequately and specify a product, if you don't know what you are trying to build. So the blame may lie with the project manager.

A UK Inland Revenue project [5] failed because of software errors. Is this due to lack of testing? And is a lack of testing due to underestimates? And were the underestimates caused by inadequate requirements?

A Washington State project failed because of inadequate requirements [5], so who gets the blame for that? Was it management or the project manager? And a US Internal Revenue Service project [5] suffered from inadequate requirements, politics, underestimates (not really surprising if they didn't know what they were supposed to be building) and immature software development practices.

The failed London Stock Exchange project has it all, 1) delusional management; 2) excessively complex design; 3) no one wanted to know the true status of the project; 4) missed deadlines; 5) soaring costs; and 6) poor project management [5]. The organization was driven by fear and arrogance with senior management problems and poor project management [5].

Failed systems such as the FBI system [5] (failure to confront reality and an impossible definition), the FAA system (an unachievable schedule and people added late to a late project) and the Oxford Health System [5] (politics and lack of management support) all appear to have serious senior management problems.

Linberg [12] also describes some failed projects with all the classic underestimation problems described by Brooks [4] so long ago. He discusses in some detail, a failed project so we are able to make a more informed decision regarding where the blame lies for the failure. He notes that the project had problems with 1) underestimated project scope; 2) poor understanding of scope; 3) unclear requirements; 5) politics; 6) management style; 7) complexity; 8) conflict with team members and outside managers; 9) noisy environment; 10) poor tool support; 11) plans developed by managers with no software development experience, and no PM involvement; 12) lack of senior management support; 13) unrealistic schedule and budget; 14) lack of resources; and 15) low project priority. This project exhibits several classic senior management problems that probably would have condemned the project without some additional PM problems as well. Could the project have survived the problems caused by: plans made by management without software experience, politics, low project priority and lack of management support, a noisy environment, and conflicts between team members and outside management? Probably not.

We are interested in updating the results of these prior studies and testing the validity of previously reported anecdotal evidence; thus our work builds on that previously reported by the Standish Group [17], Boehm [2], Glass [6], [7], Linberg [11] Charette [4] Verner and Cerpa [19] and Sauer et al [17], etc. We wish to investigate everyday projects that are not high profile enough to be reported in the literature. To date, no one has taken a set of project data and teased it out to investigate failure factors for a whole group of such projects and identified how many failure factors are manifest in projects that fail.

In the next section we briefly discuss our data collection, methodology, data analysis and findings.

III. METHODOLOGY AND DATA ANALYSIS

Based on extensive discussions with over 90 software practitioners who develop software, and the software engineering literature, we developed a questionnaire in order to investigate software developers' perceptions of software development practices that affect project outcomes.

The survey, which contained 88 questions, and was organized into a number of sections covering software development: sponsor/senior management, customer/user, requirements, estimation and scheduling, development process, the project manager, project management, and the development team, asked respondents about a recent project on which they had worked. We also asked respondents if they considered the project they referenced when answering the survey was a success or a failure.

Our questionnaire was distributed to practitioners from a large US financial institution who each responded by answering it twice, once for a recent project they considered successful and once for a recent project they considered a failure. The questionnaire was later distributed to a second group of practitioners, from various companies in North Eastern U.S.A., a third group of practitioners from a number of different Australian companies; and a fourth group of practitioners from a number of different Chilean organizations; each of the latter groups answered the questionnaire once.

We received completed questionnaires describing 304 projects. After removing those projects with large numbers of missing values our sample consisted of 235 projects. We then selected only those projects that developers considered were failures (70 projects). Each of these 70 projects was described by a different respondent. The research reported here adds to that reported in [19] which investigated the frequencies of failure factors and combinations of failure factors.

Forty-nine of the failed projects were in-house developments and twenty-one were outsourced projects. Twenty-eight failed projects were from the U.S., nine from Australia, and thirty-three from Chile. Sixty-five were development projects and five were maintenance projects. Sixty percent of our projects had a team size of fewer than ten developers, and ten percent had more than fifty-five developers. The largest project had a team of 180 developers.

Of the 88 questions asked in the questionnaire, 57 related to a factor which can be considered as either a failure factor or a success factor. Many questions required answers on a 5 point Likert scale, for example, "very poor, poor, average, good, and very good". If the response was "poor or very poor" we consider that response identifies a failure factor and we counted that response as a failure factor for that project, and accordingly added one to the total of failure factors for the project. Other questions required a yes/no answer and for those questions it depended on the wording of the question whether "no" or "yes" identified a failure factor

We examined the failed projects, having identified the failure factors for each project, and calculated the total number of failure factors existing for each project.

IV. FINDINGS

Figure 1 provides a graph showing the number of failure factors for the projects. The median number of failure factors is 28; with a minimum of five and a maximum of 47. So given that we investigated 57 failure factors, the worst project had significantly more failure factors than success factors. We note that no project suffered from just one failure factor; all projects had multiple problems and many of the factors are related to each other; causing an escalation of problems. This finding agrees with Glass [7], [8], Charette [5] and Yardley [cited in 6].

Overall the most frequent factors for all 70 projects are:

1) The delivery date impacted the development process; 93% of the failed projects;
2) The project was underestimated; 81% of the failed projects;
3) Risks were not re-assessed, controlled, or managed through the project; 76% of the failed projects;
4) Staff were not rewarded for working long hours.; 73% of the failed projects;
5) Delivery decision was made without adequate requirements information; 73% of the failed projects;
6) Staff had an unpleasant experience working on the project; 73% of failed projects.

For more detail about these failure factors, and the most common combinations of factors see [19].

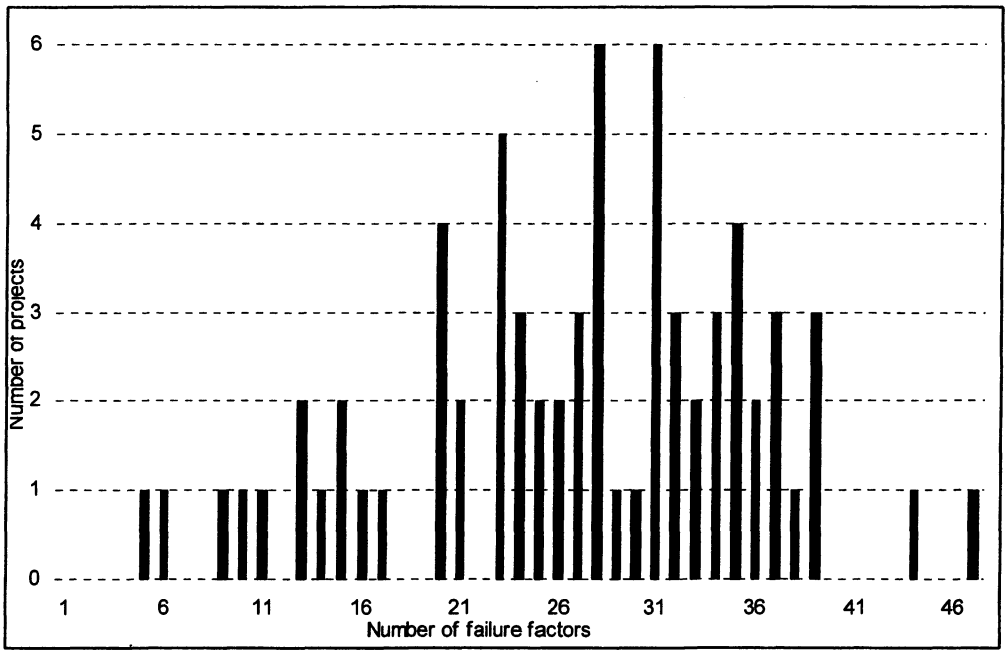

FIGURE 1. Number of projects vs number of failure factors

Table 1 lists the 8 projects, with the fewest failure factors and their actual failure factors. The only factors included in this table are those present in the projects with the fewest failure factors.

1) the delivery date affected the development process was a factor in 6 of the 8 projects and
2) the project was underestimated was a factor in 5 of the 8 projects.

We now consider in more detail the five projects with the fewest failure factors.

Project 63, with the fewest failure factors of all the projects (5) was an outsourced project that suffered from problems with an aggressive schedule that affected both team motivation and the development process, large numbers of customers/users whose expectations were not managed, negative interference from senior management, and ineffective change control. Where does the blame lie for this particular failure? We would suggest with both with the PM and senior management. The project manager, who had a background in software development and 4 years' PM experience, was involved with the estimates, so may be partially to blame for the aggressive schedule; the PM was certainly responsible for the lack of project reviews; one would expect the management of customer expectations was also a PM responsibility, as is ensuring that change control was effective. However, large numbers of customers and users and negative interference from senior management are both out of the PM's control.

We identify 6 failure factors for project 24, an in-house development, with 6 staff members, and a project manager with 14 years experience and a software development background. This project was underestimated, and neither the customers/users, nor developers were involved in the estimates, though the project manager did have input into the estimates. An aggressive schedule affected the development process and team members' lives, as well as their motivation. The blame here can be laid squarely on the poor schedule estimation and those involved in the estimates (a group consisting of senior management and the PM). Our respondent commented that the organization cut the project budget towards the end of the project

Project 13, which had a PM with a business background, 3 years' PM experience and 10 developers, was an in-house development project. This project's problems include the PM not being given full authority to manage the project; neither PM nor developers were involved in estimates that were made without appropriate requirements information, there was no method for requirements gathering, the project was not adequately staffed, the project manager did not control the project, and staff were not rewarded for working long hours.

Table 1 Failure factors by project

Project #	63	24	13	35	59	23	70	68
PM not given full authority to manage project			X					
Sponsor commitment did not last through project							X	
Senior management negatively impacted project	X					X		X
Low level of customers/user involvement							X	
Involved customers/users did not stay through project							X	
Customers/users not involved in schedule estimates		X		X			X	X
High customer/user turnover							X	
Customers/users did not have realistic expectations							X	
Problems due to large numbers of customer/uses involved	X							
No method for requirements gathering			X					X
Project did not have good requirements at any stage							X	
Scope of project not well defined					X			
Scope changes during project					X		X	
Customers/users did not make enough time for requirements gathering							X	
No single requirements repository for requirements						X		X
Size of project affected requirements elicitation					X		X	
PM had no input into estimates			X					
Delivery decision not made with appropriate requirements info			X		X			
Developers not involved in estimates		X	X	X		X		
Project underestimated		X		X	X	X		X
Delivery date affected development process	X	X		X	X	X	X	
Project not adequate staffed			X					X
Staff added late to meet aggressive schedule					X	X		X
Project manager did not control the project			X					
PM changed during project					X			
Staff not appreciated for working long hours				X				X
Staff not rewarded for working long hours			X		X			
Schedule affected team members' lives		X				X	X	
Inappropriate development methodology						X		
Risks not identified at start of the project				X				X
Risks not incorporated into project plan				X				X
Risks not reassessed, and controlled					X			X
No effective project plan								X
PM abandoned plan under pressure						X		
Customer/user expectations not managed	X							
No reviews at end of each phase			X	X		X		
Project not replanned after requirements changes					X			
Change control not monitored effectively	X					X	X	X
Team did not have adequate experience with tools and techniques				X		X		
Aggressive schedule affected team motivation		X		X		X		X

The blame for the failure of project 13 must be shared between management and the PM, but most of the blame should fall on management who interfered with the PM and underestimated the project ensuring it was doomed from the start.

Project 35 was an in-house development; there were 12 developers, and its PM had a background in software development but only 1 year's experience in project management. This project failed because it was underestimated; neither customers/users, nor developers were involved in the estimates, although the PM did have input into the schedule; the delivery date, and the aggressive schedule, affected the development process, and there were no project reviews at the end of the phases; the team did not have adequate experience with the tools and techniques used, staff were not appreciated for working long hours, and risks were not identified or incorporated into the project plan. According to our respondent the project failed because of unrealistic deadlines; in addition too much pressure was put on the team and quality assurance was minimal. This project had an inexperienced PM who agreed to an aggressive schedule, had problems managing the process, did not deal well with project risks and did not ensure that the staff was familiar with the tools and techniques to be used. The blame here can be laid on the PM although blame should be shared by the organization for not supporting an inexperienced PM

Project 59 was an outsourced project that shows all the classic symptoms that lead to project failure, mentioned so many times is the literature. The PM had a software development background and 10 years' experience. This was a large project with an ill-defined scope, and scope changes during the project; the PM was involved in the estimates which were made without adequate requirements information. Hence it is not surprising that the project was underestimated with an aggressive schedule, that the delivery date affected the development process, and staff were added late. Even though this project began with poor requirements it was not re-planned when the requirements changed. The PM was changed part way through the project. The blame here should be laid mainly at the first PM's door but management had a role in the failure as well. It was probably not possible for the replacement PM to save the project.

We have shown that there is more than one reason for the failure of a software project; it is a combination of technical, project management and business decisions and each dimension interacts with the others in complicated ways. In summary most of our failed projects have impossible estimation targets, with shortcuts taken, and good practices skipped. Blame in most cases should be equally shared between poor project management and unsupportive upper level management.

VI. LIMITATIONS

We recognize some limitations to our study. Surveys are based on self-reported data which reflects people's perceptions, not what might have actually happened. Because we surveyed software developers our results are limited to their knowledge, attitudes, and beliefs regarding the projects and PMs with whom they were involved. The questions in our survey, and hence the factors we identify, are based on both the literature and discussions with practitioners who raised issues that they perceived were important to their day-to-day activities on real projects.

VII. LESSONS LEARNED AND FURTHER RESEARCH

Our results show that projects do not fail for one single reason alone; they fail for multiple reasons. These findings are in full agreement with Glass [6] who noted there are generally more than one or two reasons for a software project to fail, and it usually is a combination of several factors. Unlike the projects Glass [7] discussed, however, our projects were not huge; they are normal everyday in-house and outsourced developments whose failures are unlikely to be reported in the press.

Most of our failed projects had problems with poor project management, though some certainly had problems that were out of the PM's control, and mainly caused by higher level management.

Eighty one percent of our 70 failed projects were underestimated; this leads to all sorts of problems in practice; this result agrees with Glass [7] who suggested that we do estimation very badly. When the customer does not assign enough time to do the requirements, the estimation process is affected - you can't estimate if you don't know what is to be developed; the result is an under-estimated project, which will likely trigger the classic problems associated with under-estimation, such as staff added late, inadequate staff, an aggressive schedule that affects the development process, team motivation and team member's lives. Staff added late to meet an aggressive schedule is still a major factor in project failure and, given that Brooks law, "Adding manpower to a late software project makes it later", was first promulgated over thirty years ago in 1975, our result is surprising; adding staff late was also highlighted by Glass in 2002 [8].

Because we were interested in the projects that were under-estimated, we investigated such projects further and found that of the 24 projects that did not have any project manager input into the estimates, approximately 88% had a delivery decision made with inappropriate requirements. This suggests that the project manager should be involved in the delivery decision. Our results are also in agreement with Glass in that unstable requirements are one of the most common causes of project failure, leading to changes in scope, which is also an important failure factor for our projects [8].

In these failed projects we identify two poorly done project management practices; when software development is impacted by its delivery date, there is not enough time to do reviews at the end of each phase, and when risks are not re-assessed and controlled during the project, there is no insight into problems within the project, such as having inadequate staff, which results in an unpleasant experience for developers.

The projects that we investigated in this research are not huge, high profile projects whose failures are reported in the press, they are examples of everyday projects that organizations deal with day to day. However, they failed for exactly the same reasons as the high profile projects reported in the literature [4], [5], [6].

Although Brooks' work [4] that discussed software development issues, particularly underestimated projects, adding staff late to projects, and change control problems, has been discussed and extended by many authors over the past 30 years, our projects still fail for the same reasons. It is

extremely disconcerting to discover that the reasons for project failure include many of same problems he noted so many years ago. Glass [8] agrees, he observed in 2002 "We seem to need to learn the same lessons over and over again".

Of course, for an organization to determine key findings related to the success or failure of a project, most practitioners and researchers agree that it is necessary to carry out an independent post project review. We note here that in our initial data set of 304 projects only a very small percentage of post project reviews occurred and those that did were almost entirely for successful projects [20]. However, project managers may reduce the problems associated with some of factors identified, by following a number of guidelines (see Table 2).

Table 2
Guidelines for failure prevention

Failure Factor	Guideline
Delivery date impacted the development process	Keep projects short and manageable without being obsessively minimalist. Change targets only for strategic reasons (for example, changes to increase business value of the project).
Project was underestimated	Develop and or retain or hire experienced project managers with estimation skills in the specific application of the project. Invest in detailed planning in order to reduce estimation errors.
Risks were not re-assessed, controlled, or managed through the project	For any project, the baseline risk of underperforming is 25% [17]. Account for project uncertainty and regularly re-assess the project plan identifying fundamental risks. Select a project manager based on the number of projects they have managed.
Staff not rewarded for working long hours.	In the event that overtime is necessary appropriate rewards either monetary or time-in-lieu should be given, and outlined at the start of the project. Provide recognition and support for the project team members.

In Table 2, we provide guidelines for the four most common failure factors. Our guidelines are adapted from Sauer, Gemino and Reich [17] who described how organizations can reduce IT project risk, and Morgenshtern, Raz and Dvir [14] who provide specific guidelines for improving estimation performance and reducing estimation errors. Project management tasks such as project estimation are important for reducing project risk. In a recent empirical study, Morgenshtern et al., [14] found the variable most highly correlated with estimation accuracy is the number of projects that the estimator has managed rather than the number of years of experience in managing projects. Sauer et al. [17] recommend that projects are kept short and manageable. Similarly, Hihn and Habib-aghi [10] found that project size affects time estimations; in small projects, communication is tighter and problems are easily dealt with and therefore, time estimation errors are smaller.

In further research we intend to investigate our successful projects. We are interested to discover how may failure factors a project can suffer from and still be perceived as successful.

ACKNOWLEDMENT

This research has been fully supported by the Chilean grant **FONDECYT 1030785**.

REFERENCES

[1] Bennatan, E.M., *On Time Within Budget*, John Wiley and Sons, 2000.

[2] Boehm B. W. Software Risk Management: Principles and Practices, *IEEE Software*, 8, 1, 1991, pp.32-41.

[3] Boehm B. W., Software Engineering Economics, Prentice Hall, New Jersey, 1981.

[4] Brooks, F. P. Jr., *The Mythical Man Month. Essays on Software Engineering*, Addison Wesley, USA, 1975.

[5] [Charettte, R., Why software fails, *IEEE Spectrum*, September, 2005, pp.42-49.

[6] Dalcher, D. and Brodie, L., *Successful IT projects.* Thomson Learning. Middesex University Press, 2007

[7] Glass, R. L,. *Software Runaways*, Prentice-Hall, New York, 1998.

[8] Glass R. L., *Facts and Fallacies of Software Engineering*, Addison Wesley, 2002.

[9] Hall, T., Beecham S, Verner J., and Wilson D., "The impact of staff turnover on software projects: the importance of understanding what makes software practitioners tick" *ACM SIGMIS Conference*, 2008, pp. 30-39

[10] Hinn, J., Habib-aghi, H., "Cost estimation of software intensive projects: a survey of current practice", in: *Proceedings of the 13th International Conference on Software Engineering*, Austin, TX, 1991, pp. 276–286.

[11] Jørgensen, M and Moløkken-Østvold, K. "How large are software cost overruns? A review of the 1994 CHAOS report", *Information and Software Technology*, 48, 2006, pp. 297-301.

[12] Linberg, K. R., Software Developer Perceptions About Software Project Failure: A Case Study, *Journal of Systems and Software*, 49, 1999, pp. 177-192.

[13] McConnell, S., *Rapid Development*, Microsoft Press. Redmond, Washington, 1996.

[14] Morgenshtern O., Raz T., and Dvir D.,. "Factors affecting duration and effort estimation errors in software development projects", *Information and Software Technology* Vol 49, 2007, pp. 827–837.

[15] Pinto, J. K. and Mandel, S. J., The Causes of Project Failure, *IEEE Transactions on Engineering Management*, 34, 7, November, 1990, pp. 67-72.

[16] Pressman, R., "Fear of Trying: The Plight of Rookie Project Managers", *IEEE Software*, January-February, 1998, pp. 50-51, 54.

[17] Sauer C., Gemino A., and Reich B. H., "The impact of size and volatility on IT project performance", *CACM*, Vol 50, No 11, 2007, pp.79-84.

[18] Standish Group International, *CHAOS: Project Failure and Success Report Report*, 1994, Dennis, Massachusetts, http://www.pm2go.com/sample_research/chaos_1994_2.asp

[19] Verner J and Cerpa N., "Software Project Failure: Do we ever learn?" *CACM* (in press).

[20] Verner J. M , Evanco W. M., and Cerpa N., "State of the Practice: Effort Estimation and Software Project Success" *Information and Software Technology*, February, 2007, pp. 181-193