DOI: 10.1145/1610252.1610286

BY NARCISO CERPA AND JUNE M. VERNER

Why Did Your Project Fail?

WE HAVE BEEN DEVELOPING SOFTWARE SINCE THE 1960S but still have not learned enough to ensure that our software development projects are successful. Boehm[2] suggested that realistic schedule and budgets together with a continuing steam of requirements changes are high risk factors. The Standish Group in 1994 noted that approximately 31% of corporate software development projects were cancelled before completion and 53% were challenged and cost 180% above their original estimate.[13] Glass discussed 16 project disasters.[5] 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.[8] Later, Glass revisited failed projects and found that poor estimation was high on his list of failure factors.[6]

In 2007 the Standish Group reported that 35% of software projects started in 2006 were successful compared with only 16% in the corresponding 1994 report; however, the 2007 CHAOS report still identifies 46% (53% in 1994) of software projects as challenged (having cost or time overruns or not fully meeting user's requirements) and 19% (31% in 1994) as outright failures.[12] The validity of the Standish Group findings has been questioned as not consistent with cost overrun results of other surveys.[7] Jørgensen and Moløkken-Østvold suggested that there are serious problems with the way the Standish Group conducted their research and that the findings were biased toward reports of failure because a random sample of top IT executives was asked to share failure stories when mailed confidential surveys.

Recently Charette[4] commented 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 suggests that from 5%-15% of projects will be abandoned before or shortly after delivery as hopelessly inadequate.[4] Other recent studies, suggest various failure rates for software development projects up to 85%.[7] Stories of software failure capture public attention and in general there is a perception that software quality is not improving but getting worse.

Developing software systems is an expensive, and often a difficult process. Although there are many guidelines for successful software development,[9,11] few project post-mortems are conducted, and little understanding is gained from the results of past projects. The project manager (PM) and the development team must deal with many pressures from project stakeholders (for example, upper level management, marketing, accounting, customers, and users) during the software development process. These pressures impact both the cost and the quality of the software produced. There are generally more than one or two reasons for a software project to fail, and it usually is a combination of technical, project management and business decisions. Many software project failure factors have been described in the literature.[1-13] However, most organizations do not see preventing failure as an urgent matter. It is not understood why this attitude persists.[4]

Because most of the literature is based on a handful of failed project case studies, in our research we analyze 70 failed software projects to determine those practices that affected project outcome. We are interested in providing, from the perspective of software practitioners, quantitative evidence targeting those aspects of the development process that contribute to project failure. Essentially, we are interested in updating the results of prior studies and testing the validity of previously reported anecdotal evidence. To date, no one has taken a set of project data and teased it out to identify, for a whole group of such projects, the most common failure factors. We are interested in everyday projects that are not high profile enough to be reported in the literature. Our work builds on that previously reported by Boehm,[2] Charette,[4] Glass,[5,6] Jørgensen and Moløkken,[7] Linberg,[8] and the Standish Group,[12,13] among others.

Methodology and Data Analysis

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

The survey, which contained 88 questions, was organized into a number of sections covering software development. The sections included: sponsor/senior management, customer/user, requirements, estimation and scheduling, development process, the project manager and project management, and the development team. We also asked respondents if they considered the project they referenced when answering the survey was a success or a failure. Our questionnaire was initially distributed to practitioners from a large U.S. 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 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 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). Forty-nine of the failed projects were in-house developments and 21 were outsourced projects. Twenty-eight failed projects were from the U.S., nine were from Australia, and 33 were from Chile. Sixty-five were development projects and five were maintenance projects. Sixty percent of our projects had a team size of fewer than 10 developers. Ten percent of projects had more than 55 developers. The largest project had a team of 180 developers. We examined the failed projects using frequency analyses and then investigated relationships between the most important factors.

Table 1. Percentage of failed projects per failure factor.

Software Project Failure Factors	Percentage of Projects (%)		
	In-house	Outsourced	Overall
Delivery date impacted the development process	93.9	90.5	92.9
Project under-estimated	83.7	76.2	81.4
Risks were not re-assessed, controlled, or managed through the project	73.4	80.9	75.7
Staff were not rewarded for working long hours	81.6	57.1	74.3
Delivery decision made without adequate requirements information	83.7	47.6	72.9
Staff had an unpleasant experience working on the project	83.7	47.6	72.9
Customers/Users not involved in making schedule estimates	69.4	76.2	71.4
Risk not incorporated into the project plan	65.3	80.9	70.0
Change control not monitored, nor dealt with effectively	63.3	85.7	70.0
Customer/Users had unrealistic expectations	69.4	66.7	68.6
Process did not have reviews at the end of each phase	75.5	47.6	67.1
Development Methodology was inappropriate for the project	71.4	52.4	65.7
Aggressive schedule affected team motivation	69.4	57.1	65.7
Scope changed during the project	67.3	57.1	64.3
Schedule had a negative effect on team member's life	71.4	42.9	62.9
Project had inadequate staff to meet the schedule	63.3	57.1	61.4
Staff added late to meet an aggressive schedule	61.2	61.9	61.4
Customers/Users did not make adequate time available for requirements gathering	61.2	57.1	60.0

Findings

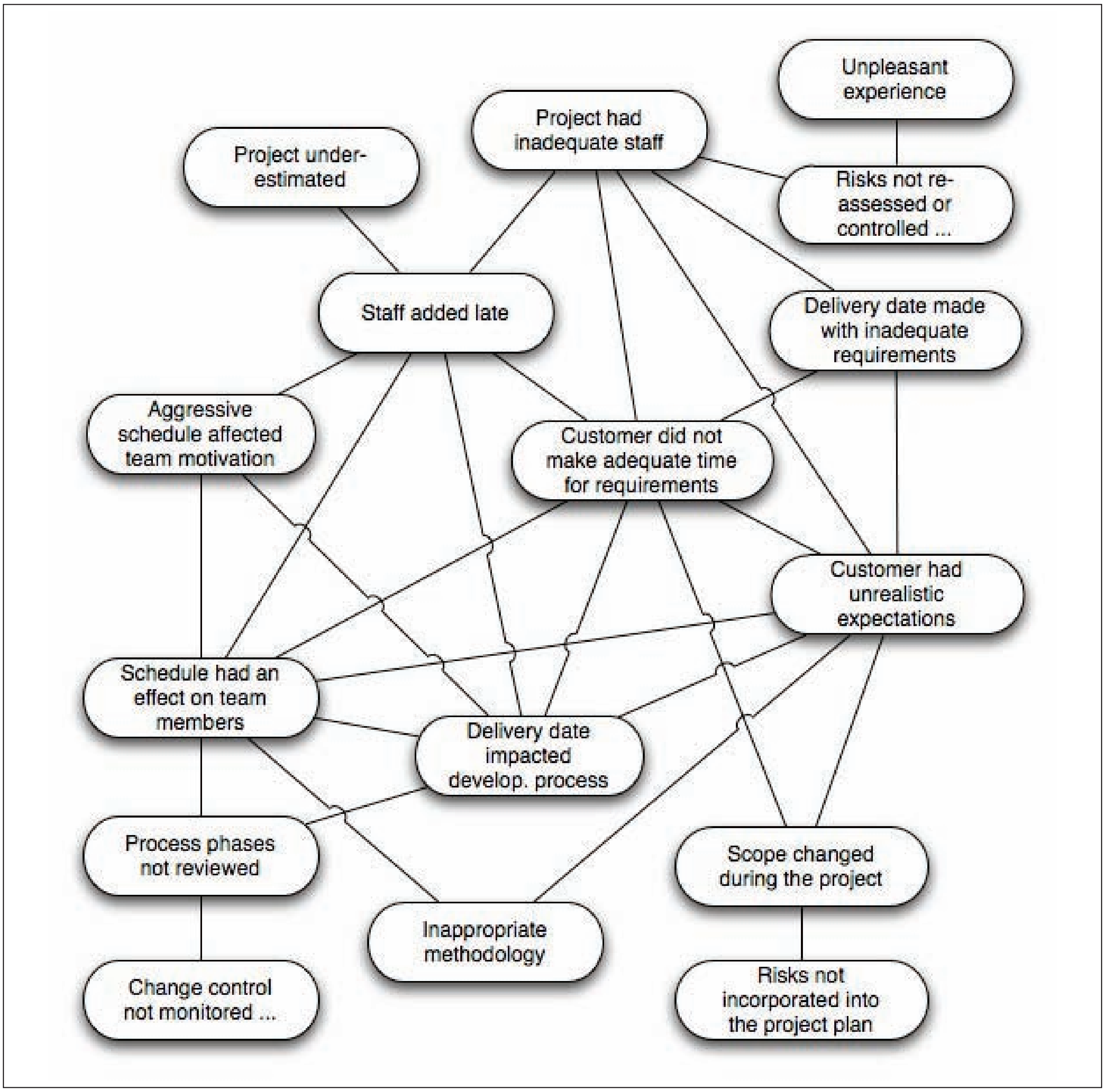
One problem that became very clear from our discussions with developers and project managers was that the political climate in some organizations was such that no project manager could admit to a failed project. Even if none of the project benefits were realized project managers time and again assured us that their project was a success. The developers who were involved with many of these projects were much more likely to believe the project was a failure.

We believe that political climate is one reason for a lack of post mortem reviews. We show (Table 1) the percentage of failed projects overall and the most frequent failure factors in descending order of frequency. We have included in Table 1 all failure factors found in at least sixty percent of the projects. We note that all projects did not suffer from just one failure factor; they all had multiple problems and many of the factors are related to each other; causing an escalation of problems (see Figure 1). As Table 1 shows, overall the most frequent factors are:

1. delivery date impacted the development process,
2. project was underestimated,
3. risks were not re-assessed, controlled, or managed through the project, and
4. staff were not rewarded for working long hours.

While these failure factors are all related to the management of the projects some of the factors may be partially or wholly out of the project manager's control (for example, rewards for staff, cus-

Figure 1. Significant relationships between the most important software project failure factors


Unpleasant experience
Project had inadequate staff
Project under-estimated
Risks not re-assessed or controlled ...
Staff added late
Delivery date made with inadequate requirements
Aggressive schedule affected team motivation
Customer did not make adequate time for requirements
Customer had unrealistic expectations
Schedule had an effect on team members
Delivery date impacted develop. process
Process phases not reviewed
Scope changed during the project
Inappropriate methodology
Change control not monitored ...
Risks not incorporated into the project plan

tomers/users did not make adequate time available for requirements gathering). As noted earlier, the projects did not have just one problem. Seventy six percent of projects suffered from the two top failure factors: both a delivery date that the developers felt was impacted by the development process, and a project that was underestimated. Fifty six percent of projects had all three of the top failure factors: a delivery date that the developers felt was impacted by the development process, a project that was underestimated, and risks that were not re-assessed, controlled, or managed through the project. Forty six percent had all top four failure factors; 33% had all the top five failure factors, and 23% had all the top six failure factors.

There are four factors in Table 1 which we could consider "people" factors and will de-motivate team members; they include:

1. lack of rewards for working long hours,
2. an unpleasant experience while working on the project,
3. an aggressive schedule that affected team motivation, and
4. a schedule that had a negative effect on the team members lives.

If we break the data into in-house and outsourced projects we get slightly different results. The in-house projects fail because:

1. delivery date impacted the development process,
2. project was underestimated,
3. delivery date decision was made without appropriate requirements information, and

4. staff had an unpleasant experience working on the project.
5. fifty-one percent of the in-house projects had all top 4 in-house failure factors.

On the other hand, the top 4 failure factors for outsourced projects are:

1. delivery date impacted the development process,
2. change control was neither monitored nor dealt with effectively
3. risks were not incorporated into project plan, and
4. risks were not re-assessed, controlled, or managed through the project.
5. fifty-two percent of the outsourced projects had all top four outsourced failure factors.

The in-house projects had more problems that are likely to de-motivate the development staff. Overall the outsourced projects have significantly better requirements (delivery decision is made with better requirements than in the in-house projects), and the development staff are treated significantly better in the outsourced projects (staff get a better personal experience and more rewards than in-house development staff). However, the outsourced projects had more problems with general project management practices such as risk management and change control.

Figure 1 shows the multiplying effect that a single failure factor can have on a project. Note how "project had inadequate staff" has relationships with many other undesirable project aspects. The effect of "customer did not make adequate time for requirements" also has relationships with many other undesirable factors.

Lessons Learned

Our results show that projects do not fail for one single reason alone; they fail for multiple reasons. These findings agree with Glass[5] 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[5] discussed, however, our projects were not huge. A common problem with our failed projects was inadequate requirements when the delivery decision was made (73%). The customer did not spend enough time with developers to define the requirements properly; this can lead to unrealistic expectations. Then, because the initial requirements are poor, it is not surprising that there are scope changes during the project. It is interesting to speculate on the causal effects between "delivery date made with inadequate requirements" and "project had inadequate staff."

Did we have inadequate requirements because we did not have enough and/or appropriate staff to elicit the requirements? Or do we have inadequate staff because the delivery date was made with inadequate requirements? Our research shows relationships between factors but not the causal effect, and we can only speculate on the actual direction of the relationships. To exacerbate the early problems with requirements, most of our projects did not have an appropriate methodology that could deal with the inadequate requirements.

We found that estimates were badly done in 81% of our 70 failed projects; this leads to all sorts of problems in practice; this result is in agreement with Glass[6] 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, aggressive schedule affecting team motivation and their lives. Staff added late to meet an aggressive schedule is still a major factor in project failure. Given that Brooks law, "adding manpower to a late software project makes it later," was first promulgated over 30 years ago in 1975, our result is surprising. Adding staff late was also highlighted by Glass in 2002.[6]

Because we were interested in the projects that were under-estimated, we investigated such projects further. We found that of the 24 projects that did not have any input into the estimates by the project manager, approximately 88% had a delivery decision made with inappropriate requirements. This suggests that the project manager should be involved in the delivery decision. This result agrees with Glass[6] who suggested that software estimation is usually done by the wrong people. 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 the scope, which is also an important failure factor in our projects.

We identified two poorly done project management practices:

1. When software development is impacted by its delivery date, there is not enough time to do reviews at the end of each phase, and
2. When risks are not re-assessed and controlled during the project, there is no insight into problems within the project, such as inadequate staff, which then results in an unpleasant experience for developers.

We note that some of the problems are different for outsourced projects. Here some of the factors, even though still a problem, are not ranked as highly as they are for the in-house projects, for example:

"Staff were not rewarded for working long hours,"

"Delivery date decision made with inadequate requirements information," and

"Staff had an unpleasant experience working on the project."

Other problems such as "Risks not incorporated into the project plan" and "change control not monitored nor dealt with effectively" are ranked much higher for the outsourced projects.

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

Although Brooks listed software development issues, particularly under-estimated projects, adding staff late to projects, and change control problems, that have been noted, discussed and extended by many authors over the past 30 years,[3] 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 noted by Brooks so many years ago. Glass[6] agrees when he noted in 2002, "We seem to need to learn the same lessons over and over again." While organizations do not conduct post mortem project reviews they are unlikely to understand why their projects fail.

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), 32-41.
3. Brooks, F. P. Jr., *The Mythical Man Month. Essays on Software Engineering*. Addison Wesley, 1975.
4. Charettte, R. Why software fails. *IEEE Spectrum*, (Sept. 2005), 42-49.
5. Glass, R. L. *Software Runaways*. Prentice-Hall, New York, 1998.
6. Glass R. L. *Facts and Fallacies of Software Engineering*. Addison Wesley, 2002.
7. 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), 297-301.
8. Linberg, K. R. Software developer perceptions about software project failure: A case study. *Journal of Systems and Software 49*, 1999.
9. McConnell, S. *Rapid Development*, Microsoft Press. Redmond, WA, 1996.
10. Pinto, J. K. and Mandel, S. J., The causes of project failure. *IEEE Transactions on Engineering Management 34*, 7, (Nov. 1990).
11. Pressman, R. Fear of trying: The plight of rookie project managers. *IEEE Software*, (Jan.-Feb. 1998).
12. Rubenstein, D. Standish group report: There's less development chaos today. SD Times, (Mar. 1, 2007); http://www.sdtimes.com/article/story-20070301-01.html
13. Standish Group International. CHAOS: Project failure and success report report. (1994), Dennis, MA; http://www.pm2go.com/sample_ research/chaos_1994_2.asp.

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

Narciso Cerpa (ncerpa@utalca.cl) is an associate professor in the computer science department of the Faculty of Engineering at the Universidad de Talca, Chile.

June M. Verner (June.Verner@gmail.com) is a visiting professor of software engineering at CSE, University of New South Wales, Sydney, Australia.