December 2006

The Impact of Software Process Maturity and Software Development Risk on the Performance of Software Development Projects

Dany Di Tullio
Queen's University

Bouchaib Bahli
Concordia University

THE IMPACT OF SOFTWARE PROCESS MATURITY AND SOFTWARE DEVELOPMENT RISK ON THE PERFORMANCE OF SOFTWARE DEVELOPMENT PROJECTS

Social, Behavioral and Organizational Aspects of Information Systems

Dany Di Tullio
Queen's School of Business
Queen's University
dditullio@business.queensu.ca

Bouchaib Bahli
John Molson School of Business
Concordia University
bbahli@jmsb.concordia.ca

Abstract

Despite the increasing efforts of organizations to improve the development processes of their software projects, there remain few empirical and generalizable findings when it comes to key questions regarding software process improvement initiatives. Rarely there was any empirical examination in an integrative model on how process improvement efforts affect key organizational concerns such as software project performance and the threat of risks in today's dynamic and complex business environment. In addressing this knowledge gap, we propose and test a research model that allows for an empirical examination of the relationships between software process maturity and the performance of software development projects while assessing the impact of risk on software project performance. Data were collected from officially CMM-appraised organizations to test developed hypotheses using proven metrics identified in the literature. Our findings support our theoretical framework in providing evidence of the positive impact of software development process maturity on the performance of software projects while underscoring the negative effect of risk on software project performance. Key results and a discussion of the findings are also provided.

Keywords: Software process maturity, software process improvement, software development risk, software project performance

Introduction

Given the pervasiveness of software in today's organizations, the importance of software reliability has become a major concern and stresses the critical nature of the software development process. However, while the advancement of technology continues to progress at a considerable pace, the development process in itself seems to be having trouble keeping up (Rai and Al-Hindi 2000). The managerial aspect of software development projects is often undertaken without adequate planning, with a poor grasp of the overall development process, and with a lack of a well-established management framework even as focus is shifting from a technology perspective to a more process-centric view of software development (Humphrey 1989; Rai and Al-Hindi 2000). Carefully conceived management practices are thus needed to improve the software development process and to gain better control over uncertain and risky environments and are now emerging as viable solutions to the software crisis (Barki et al. 1993; Barki et al. 2001; Canfora et al. 2005; Fitzgerald and O'Kane 1999; Humphrey 1989). In addressing these concerns, significant efforts have lately focused on the design and evolution of software development processes with the objective to improve their capability and maturity (Ravichandran and Rai 2000). Many firms now consider software process

improvement initiatives as a strategic issue (Iversen and Ngwenyama 2006). As an example, the Capability Maturity Model (CMM) consists of "a coherent, ordered set of incremental improvements, all having experienced success in the field, packaged into a roadmap that shows how effective practices can be built on one another in a logical progression" (Herbsleb et al. 1997, p. 30).

This particular software process maturity model is now used by major firms in every sector of the economy and around the world (Process Maturity Profile 2004). Accordingly, the substantial adoption of process improvement initiatives (Process Maturity Profile 2004), the significant implementation efforts they entail (Ibbs and Kwak 2000), and the growing concern for risk in system development (Barki et al. 2001; Iversen et al. 2004; Wallace et al. 2004), stress the need for providing answers to important practical questions. Essential to the advancement of knowledge pertaining to software process improvement are the following key research questions: What is the impact of software development process maturity on the performance of software development projects? What is the relationship between software development risk and the performance of software development projects? It is time to move beyond isolated case studies and anecdotes that have so often characterized this sort of debate in the past and adopt a scientific approach in researching software process improvement efforts (Herbsleb et al. 1997). In addressing this need, this research is a clear step in that direction.

To answer these questions, a conceptual model aimed at allowing for rigorous empirical investigation is proposed. Grounded in prior research in the areas of software process improvement, risk in information systems (IS), and software project performance, hypotheses are developed and empirically tested in order to provide explanations as to the effects of software process maturity on the performance of software development projects as well as the influence of software development risk on performance. An overview of the relevant concepts is offered followed by the suggestion of a research model. These concepts are then applied in developing a series of testable hypotheses. Data collection is described along with the metrics used to assess the constructs. The ensuing research results are presented and discussed. The last section concludes with a review of this paper's prime contributions for research and practice.

Theoretical Background

Software Process Improvement

Software process improvement (SPI) is largely concerned with improving a software project's capability to develop high-quality software based on the requirements of customers or end-users (McFeely 1996). The underlying principles of SPI are key elements of an effective software process and the description of an evolutionary, stepwise improvement path for software organizations from ad hoc, immature processes, to mature, disciplined ones. Thorough assessments of a firm's practices allow for the application of normative models for improving its software development processes. Many firms consider software process improvement as a strategic issue (Iversen and Ngwenyama 2006) due to the fact that a failed process would certainly lead to failed software. Among the different models available, the most popular ones include the Capability Maturity Model (CMM), Bootstrap, and the Software Process Improvement and Capability determination (SPICE) (Iversen et al. 2004).

The improvement of software processes and the improvement models is acquiring an increasing importance in the software process community (Canfora et al. 2005). While the Software Engineering Institute (SEI) has reported an increase in process maturity certification holders (SEI.com), and although there are two published studies that we know of on the management of risks in software process improvement (Iversen et al.; Statz et al. 1997), there isn't any research on the impact of process improvement and software development risk on software project performance. The motivation of the present study is to integrate and empirically validate these three constructs in one comprehensive nomological research model. This will allow both researchers and practitioners to understand whether software process improvement as well as software project risk have an effect, if any, on software project performance. In such a case, software development project managers would need to pay attention to software process improvement and not be limited to software development methodologies as-is. Hence, we believe that this key observation contributes to the body of knowledge on software process improvement literature.

Risk in Information Systems Research

As a large proportion of software project failures are management-related, the search for appropriate managerial action to solve this problem has attracted much attention (Schmidt et al. 2001). The concept of risk in IS can be organized into two distinct streams of research: the rational decision theory perspective of risk and the behavioural perspective of risk (Lyytinen et al. 1998; March and Shapira 1987).

Rational decision theory addresses the concept of risk in a quantitative manner or in other words as the variation in the distribution of possible outcomes, their odds of occurring, and their subjective values (Arrow 1965). Rational choice theory postulates that managers dealing with risk first calculate alternatives and then select the option yielding the highest outcome among the available risk-return combinations (Yates 1992). According to this view, the assessment of the probabilities of undesirable events and their associated losses is necessary in order to measure the degree of risk. The quantitative evaluation of risk is thus a key concern in this context (Boehm 1989; Boehm and Ross 1989). However, several difficulties arise when assessing risk using a quantitative evaluation of probabilities (Barki et al. 1993). In many cases, probability distributions of undesirable events are very difficult to assess and can be unreliable (Post and Diltz 1986).

The behavioural perspective of risk (March and Shapira 1987) more accurately defines the assumptions that underlie most risk management approaches. Risk research that falls into this perspective assumes that risk management approaches are concerned with ambiguous losses and depend on multidimensional and qualitative models. It is also assumed that these risk management methods try to steer clear from risks or master them through sequential pruning exercises (Lyytinen et al. 1998). These approaches of evaluating risk focus on the factors that influence the occurrence of undesirable events instead of assessing their probabilities of occurring. Barki, Rivard, and Talbot (1993, 2001) directly addressed the difficulty of coupling the concept of risk with outcome probabilities in devising an instrument comprised of uncertainty and risk variables derived from previous research in risk and uncertainty literature. They define software development risk as project uncertainty multiplied by the magnitude of potential loss due to project failure. This definition refers to *uncertainty* rather than *probability* and assumes a single unsatisfactory outcome: project failure (Barki et al. 1993, 2001). The authors grouped project characteristics that influence the occurrence of project failure along five dimensions: technological newness, application size, lack of expertise, application complexity, and organizational environment while also assessing the magnitude of potential loss due to project failure (Barki et al. 1993, 2001).

Based on the above discussion, we adopt the behavioral perspective of risk in assessing its influence on the performance of software development projects.

Software Project Performance

Performance measurement is a central issue when attempting to evaluate software development projects. As organizations continue to invest substantial amounts of resources in the development and maintenance of information systems, performance issues are increasingly considered by managers in order to assess the overall effectiveness and degree of success of IS projects. The assessment of software project performance as a two-dimensional construct comprised of both a process and a product dimension has garnered increasing support throughout the years (Agresti 1986; Barki et al. 2001; Cooprider and Henderson 1990-91; Jiang et al. 2004; Nidumolu 1995; Rai and Al-Hindi 2000; Ravichandran and Rai 2000; Riddle 1984). At the core of this theoretical perspective of performance is the idea that one of the key goals of performance measurement is not only to assess and to improve the final output (tangible or intangible) of the production process, but to also consider the processes used to obtain that output. The importance of adopting a two-dimensional view of performance is also supported by the fact that there is a potential conflict between the efficiency of the process and the quality of the product. For example, software development projects may deliver systems of high quality while significantly exceeding budget and schedule constraints. On the other hand, well-managed projects that consistently remain within the projected schedule and budget targets may very well deliver products of poor quality (Nidumolu 1995). Insightful results have been obtained in using the two-dimensional view of performance. Specifically, results suggest that, in order to increase both the process and product performance of software development projects, a project's risk management profile, which includes such activities as formal planning, internal integration, and user participation, needs to vary according to the project's exposure to risk (Barki et al. 2001). Projects exposed to low degrees of risk require a different risk management profile than do projects characterized by a high level of risk exposure. In the latter case, the risk management profile should include such things as high information processing capability approaches as well

as high levels of formal planning in order to increase the performance of software development projects (Barki et al. 2001). The two-dimensional view of performance was selected in the context of this study, specifically Nidumolu's (1995) conceptualization of performance, as it clearly addresses the importance of adopting both the process and product perspectives of performance.

Research Model and Hypotheses

The conceptual model, shown in Figure 1, was developed to answer the key research questions that motivate this paper: What is the impact of software development process maturity on the performance of software development projects? What is the relationship between software development risk and the performance of software development projects?

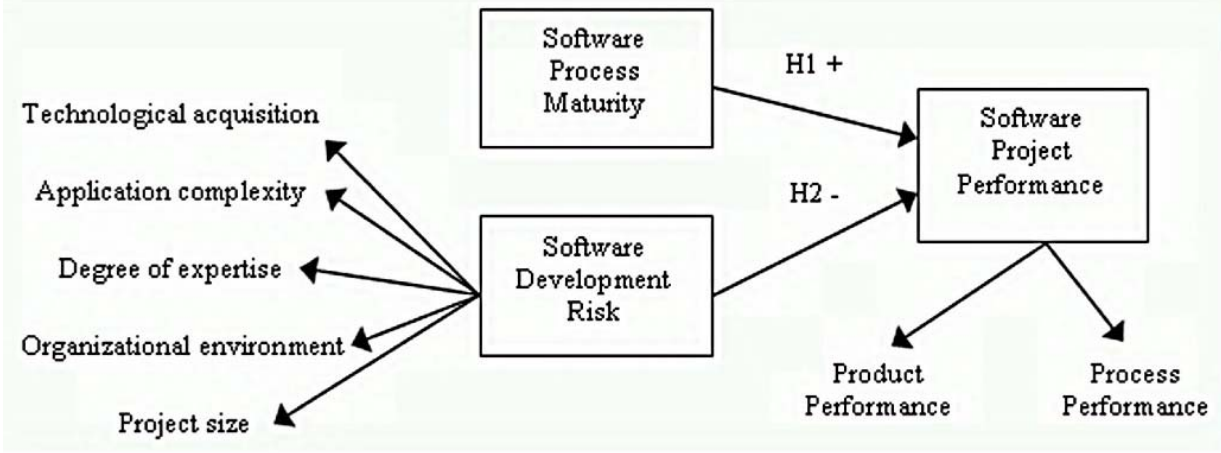

Figure 1. The Conceptual Framework

In order to answer these questions, metrics pertaining to software development process maturity, software project performance, and software development risk were identified and incorporated into the conceptual framework. Nidumolu's (1995) software project performance construct and Barki et al.'s (1993) risk exposure construct were identified as the most relevant conceptualizations of software project performance and software development risk in the context of this study. Furthermore, in order to capture the maturity of software development practices, we decided to survey organizations that had been recently officially appraised based on the CMM for software (SW-CMM). This decision was taken based on the fact that, although the CMMI (CMM Integration) is the latest version of the model, a large number of firms are still using the SW-CMM while many are in the process of migrating to CMMI. Therefore, in order to obtain an adequate sample size, SW-CMM-appraised organizations were surveyed. Grounded in prior research in the areas of software process improvement, risk in information systems, and software project performance, testable hypotheses are developed next.

Process Maturity and the Performance of Software Development Projects

Software process maturity has been related to performance improvements in a number of case studies that reveal substantial value to organizations that have implemented well conceived process improvement efforts. The quality standards incorporated into software process improvement models have been shown to increase project performance. For example, Bull HN Information Systems Inc., the US subsidiary of Groupe BULL, one of the largest European systems integrator, established the Capability Maturity Model as the source of goals for its software process improvement (Herbsleb et al. 1994). Process improvement efforts were beneficial on several levels including schedule (coding time, testing time) and quality (yearly amount of defects reported by customers) (Herbsleb et al. 1994). Similarly, a six-year study at Hewlett-Packard found that delivered defects were greatly

reduced and cost savings over 100 million dollars were achieved through software process improvements (Myers 1994) while a review of software process improvement efforts in 13 organizations showed improvements in cycle time, defect density, and productivity (Herbsleb et al. 1994).

Empirical research results have also provided support for the impact of software development process maturity on software project performance. Software project performance in terms of both process and product performance has been shown to be positively affected by the maturity of a firm's software development processes (Jiang et al. 2004). More specifically, as organizations progress in terms of the maturity of their software processes, performance indicators such as project costs and schedule improve (Jiang et al. 2004). High maturity organizations in terms of software development processes are more likely to have less difficulty adhering to cost and schedule targets (Lawlis et al. 1995). Higher levels of software process maturity also positively affect staff morale as well as the ability to meet budgets (Herbsleb and Goldenson 1996). Other performance measures are also addressed such as software product quality, system development cycle time, and development effort (Harter et al. 2000). Improvements in process maturity entail higher product quality, while higher quality in turn leads to reduced cycle time and development effort in software products (Harter et al. 2000). Furthermore, the net effect of increases in software process maturity on development cycle time and system development effort is negative (Harter et al. 2000). Therefore, these findings provide preliminary support for the following hypothesis.

H1: *Software development process maturity is positively related to software project performance.*

Software Development Risk and Software Project Performance

Software development risk has been found to negatively affect overall software project performance. Individual project risk variables such as the lack of a development team's general expertise, the intensity of conflicts among team group members, and the lack of clarity of role definitions among team members, are were found to be significantly related to project efficiency (Jiang and Klein 2000). Project efficiency incorporated such items as considerations on the amount and quality of work, adherence to schedules and budgets, speed and efficiency, and ability to meet goals (Jiang and Klein 2000). Other specific items such as top management involvement and user support in a software development project were also found to be significantly related to a development team's perception of its performance. Findings indicate that when software development team members consider their projects as not benefiting from user support and/or top management support, teams do not perform well (Jiang et al. 2000). Project management risks were also found to be significantly related to both the process performance and the product performance of software development projects (Wallace et al. 2004). Indeed, such factors as organizational environment risk, user risk, requirements risk, project complexity risk, planning and control risk, as well as team risk, have a significant negative impact on both the process and product performance of software development projects (Wallace et al. 2004). Taken together, these variables indicate the negative impact of project management risk on both the process and product dimensions of performance and thus stress the need to address software development risk when considering key organizational concern such as the performance of software development projects (Wallace et al. 2004).

Moreover, risk has also been studied as residual performance risk which is defined as the difficulty in estimating performance-related outcomes during later stages of software development projects (Nidumolu 1995; Nidumolu 1996). Project uncertainty increases residual performance risk while both project uncertainty and residual performance risk have a direct negative effect on project performance (Nidumolu 1995). Furthermore, the presence of residual performance risk reduces process control or the extent to which the development process is under control and was not shown to have an impact on product flexibility- the extent to which the software product is able to support new products or functions in response to changing business needs (Nidumolu 1996). These findings were later further supported by recent research undertaken in Korea (Na et al. 2004). Results in a Korean setting showed that increases in requirements uncertainty are directly associated with increases in residual performance risk and decreases in software project performance in terms of process and product performance in Korean software development projects (Na et al. 2004). These findings provide additional evidence as to the generalizability of the impact of risk on the performance of software development projects and stress the importance of addressing risk regardless of national and cultural boundaries. It can therefore be argued that risk plays an important role in the performance outcome of software development projects. Specifically, a project's level of software development risk may very well negatively influence software project performance as is hypothesized below.

H2: *Software development risk is negatively related to software project performance.*

Research Methodology

Research Variables and Measures

The variables used in this study fall into three constructs: software process maturity, software project performance, and software development risk, and were adopted from prior research. Each construct is described in turn below.

Software development process maturity was measured on the CMM maturity scale and reflects an organization's software process capability while allowing for a better understanding of the necessary steps that need to be taken in order to lay the foundations for continuous software process improvement (Paulk et al. 1995). The CMM framework is comprised of 18 key process areas such as software project planning, organization process focus, software quality management, and defect prevention (Paulk et al. 1995). A software process is assigned the highest maturity level if the goals in the 18 key process areas of the CMM are met.

The software project performance construct used in this study was adopted from Nidumolu (1995, 1996) who addresses the dichotomist view of performance. The importance of adopting a two-dimensional view of software project performance stems from the fact that there is a potential conflict between the efficiency of the processes involved and the quality of the end product. Software development projects may very well deliver systems of high quality while significantly exceeding budget and schedule constraints. Then again, well-managed projects that consistently remain within the projected schedule and budget targets may very well deliver products of poor quality (Nidumolu 1995). Nidumolu's (1995, 1996) conceptualization of performance not only clearly addresses the importance of adopting both the process and product perspectives of performance but is also highly significant in the context of this study as it directly refers to the process performance of software development projects, a key measurement concern in assessing the impact of software process improvement.

The instrument used to measure software development risk was adopted from Jiang and Klein (2000) who adapted and validated the original risk measure developed by Barki, Rivard, and Talbot (1993). The software development risk construct is comprised of software development risk factors grouped along five dimensions: technological acquisition, project size, degree of expertise, organizational environment, and application complexity. The construct is comprised of a total of 46 items organized into five dimensions of risk factors, all measured using a 7-point Likert-type scale with anchors that range from "Strongly disagree" and "No expertise" to "Strongly Agree" and "Outstanding Expertise".

Data Collection and Research Sample

In order to test the relationships depicted in the research model, a survey research methodology was adopted. A database of approximately 1000 officially SW-CMM-appraised organizations was used and respondents were invited to answer an online questionnaire containing the items used to assess the constructs. Dillman's (2000) recommendations for developing and administering Web surveys were followed. Respondents were IS managers who could provide answers with regard to their organization's software development projects. Upon completing data collection, 107 usable questionnaires were used in this study. This consists of a response rate of approximately 10%. Half the organizations (53%) in the sample are from the software development and IT services sectors while 40% are relatively large organizations with more than 1000 employees. Moreover, almost half of the organizations in the sample (46%) had software development teams that had less than 20 members for their last completed software project while 20% had more than 60 team members.

Data Analysis and Research Results

Assessment of the Measurement Model

Partial least squares (PLS) was used to test the research model and hypotheses (Chin 1998). A PLS analysis involves two stages: (1) the assessment of the measurement model which includes item reliability, convergent validity, and discriminant validity, and (2) the assessment of the structural model. In assessing the measurement model using

PLS, individual item loadings, Cronbach's alpha coefficients, and the average extracted variances by construct were examined as a test of the model's reliability.

Table 1. PLS Factor Loadings

Variables	CMM Level	Software Development Risk (α=0.83, ICR=0.70, AVE=0.84)	Software Project Performance (α=0.95, ICR=0.94, AVE=0.97)
LEV	**1**		
RF1		0.26	
RF2		**0.51**	
RF3		**0.92**	
RF4		0.11	
RF5		0.41	
PR1			**0.95**
PR2			**0.93**

Individual item loadings help determine item reliability which indicates whether given items measure a specific construct only. Item reliability was assessed by examining these loadings on their respective constructs. A rule of thumb employed by many researchers is to accept items which have a loading score of 0.70 or higher (Rivard and Huff 1988). In addition, a score of at least 0.5 is acceptable if other items measuring the same construct have a high reliability score (Chin 1998). Two software project risk variables met these criteria as project size (RF2) has a value of 0.51 while degree of expertise (RF3) demonstrates a loading value of 0.92 and are shown in bold in Table 1. In the case of the software project performance construct, both the process performance and product performance variables have high loading values above 0.9 as shown in bold in Table 1.

Evidence of the reliability of the constructs used in the study was also obtained by calculating Cronbach's alpha coefficients in order to determine whether the items comprising each construct are internally consistent. A score of 0.7 or higher indicates adequate construct reliability (Nunnally 1978). As shown in Table 1, based on this criterion, the software development risk ($\alpha = 0.83$) and software project performance ($\alpha = 0.95$) constructs both demonstrate sufficient reliability. Furthermore, the average variance extracted (AVE) indicates whether significant variance is shared between each variable and their respective construct. A score of 0.5 represents an acceptable level of variance extracted (Fornell and Larcker 1981). Based on this criterion, the variance extracted for both constructs is more than enough. Therefore, significant variance was shared between each item and their respective construct, indicating adequate variance extracted and construct reliability.

Assessment of the Structural Model

The primary objective of this study is to provide empirical evidence as to the relationships between software development process maturity and software project performance while assessing the impact of software development risk on project performance. The structural model and hypotheses were therefore assessed by taking into consideration the path coefficients along with their level of significance. Each hypothesis was tested using PLS Graph (Chin 1995) which provided both of these values. Hypothesis 1 tested the relationship between software process maturity and software project performance. A positive relationship was predicted. In other words, increased levels of software process maturity predicted to lead to higher levels of software project performance. Results indicate that software process maturity is indeed significantly and positively related to software project performance (path = 0.19; $p < 0.01$). Hypothesis 1 is therefore supported. Moreover, hypothesis 2 tested the relationship between software development risk and software project performance which predicted that higher levels of risk would entail lower levels of performance. The second hypothesis was also supported as a negative and significant relationship was found (path = -0.51, $p < 0.001$). Finally, the percentage of variance explained (R^2) of software project performance was 31%. A detailed figure of the structural model results appears below (see Figure 2).

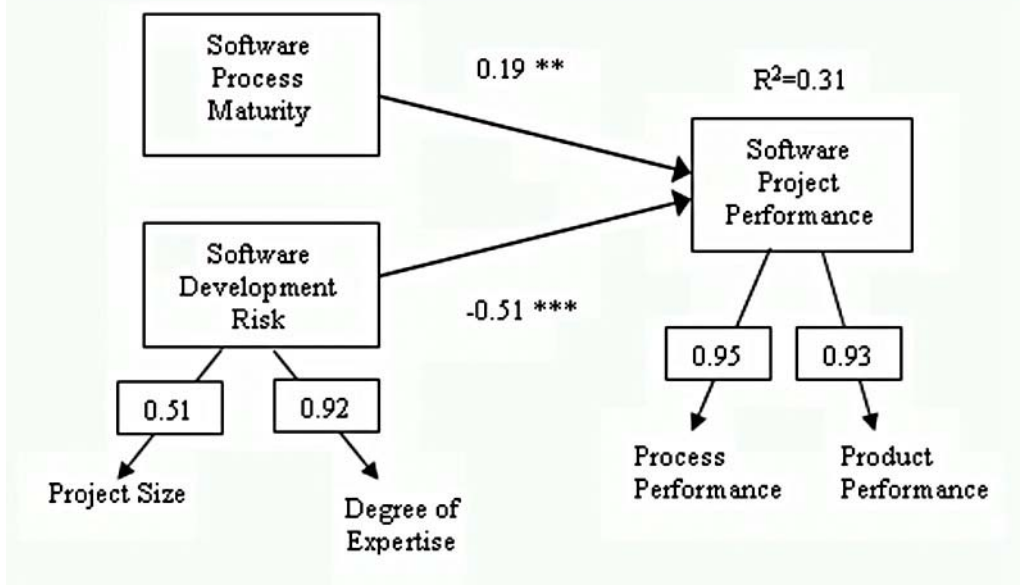

Figure 2. Parameters for the Research Model

(**: $p < 0.01$; ***: $p < 0.001$)

Our analysis provides support for each hypothesis. Findings indicate that software process maturity is positively and significantly related to software project performance, both process and product performance (H1 supported), and that software development risk is negatively and significantly related to the performance of software development projects (H2 supported).

Discussion and Conclusion

The objective of this study was to examine the relationship between software process maturity and software project performance while assessing the impact of software development risk on the performance of software development projects. Two key questions motivated this research: (1) What is the impact of software process maturity on the performance of software development projects? and (2) What is the relationship between software development risk and software project performance?

To answer these questions, a conceptual model was proposed and tested in order to conduct a sound empirical investigation. Metrics grounded in prior research were identified in order to proceed with a large-scale survey research to provide answers for each of the above questions. Data for assessing the software development process maturity of firms were collected from organizations that were officially SW-CMM-appraised by SEI-authorized lead appraisers in order to ensure an adequate sample size given the model's substantial level of adoption. Respondents consisted of individuals who were knowledgeable and could provide answers with regard to their organization's software development projects. IS executives, software project managers, as well as quality and process managers were some if the positions held by participants of the study. Partial least squares (PLS) was used to test the research model and hypotheses. The PLS analysis consisted of a two-pronged approach. First the measurement model was validated and refined through reliability and validity tests. Second, the structural model was assessed by examining the model's path coefficients along with their statistical significance. The conceptual framework was thus tested and supported.

As with any research endeavour, it is important to underscore some of the limitations of the present study. For one, as argued above, we surveyed organizations that are appraised at given maturity levels based on the CMM for

Software instead of the CMMI. This was a conscious decision made by the researchers in order to ensure an adequate sample size given the relative newness of the CMMI when compared to the established CMM model. However, now that an increasing number of organizations are migrating to the CMMI, it would be important for future research to address this model with regard to critical management concerns such as performance and risk in software development projects. Furthermore, because we have adopted a cross-sectional research method, our findings do not provide any insights as to the dynamic aspect of software process improvement efforts. In other words, interesting new findings could stem from longitudinal studies on the dynamic interplay between process improvement and the level of performance of software projects. How does performance vary as the software development processes of an organization mature? What kind of software development risks do managers need to address in the context of software projects and depending on the process maturity attained by the organization? These are but some of the research questions that could be considered to further our understanding of the field.

Apart from the above limitations, important findings indicate that software project performance increases with higher levels of software process maturity. In other words, organizations that implement software process improvement initiatives are characterized by higher levels of performance when it comes to their software projects. This corroborates the many claims made by various case studies regarding the benefits of software process improvement initiatives (Diaz and Sligo 1997; Fitzgerald and O'Kane 1999; Goldenson et al. 1995; Herbsleb et al. 1997; Herbsleb et al. 1994; Herbsleb and Goldenson 1996). It thus seems apparent that as staged evolutionary initiatives that provide an incremental roadmap towards process control and continuous process improvement, software process improvement efforts can be tied to performance metrics. Organizations that implement the many underlying principles of software process improvement can anticipate process enhancements that can translate into performance increases in the form of both process and product performance. However, it is also important to note that although the relationship between software process maturity and software project performance was found to be significant, a path of 0.19 could indicate that following CMM processes alone may not guarantee software development project performance. For instance, a project may implement CMM processes and still end up with cost overruns. Future research could explore this relationship further. Practicing managers must also keep in mind the threat of software development risk. Such common factors as large software projects, software developers' management abilities and expertise must be closely monitored by any firm as evidenced by the negative influence of risk factors on a software development project's bottom line, its performance. Even in the context of successful process improvement programmes, managers must be mindful of the many risks involved in software development projects while acting proactively to effectively identify and mitigate threats that may hinder a project's performance.

Acknowledgments

This research was supported by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Fonds quebecois de recherche sur la societe et la culture (FQRSC). We are most grateful to the track co-editors Mihir Parikh and Nathalie Mitev, the associate editor, and the three anonymous reviewers for their valuable critiques and suggestions.

References

Agresti, W. (1986) (ed.), *New Paradigms for Software Development*, Washington, DC: IEEE Computer Society Press.

Arrow, K.J. *Aspects of the Theory of Risk Bearing*, Yrjö Janssonin Säätiä, Helsinki, Finland, 1965.

Barki, H., Rivard, S., and Talbot, J. "Toward an Assessment of Software Development Risk," *Journal of Management Information Systems* (10:2), 1993, pp. 203-225.

Barki, H., Rivard, S., and Talbot, J. "An Integrative Contingency Model of Software Project Risk Management," *Journal of Management Information Systems* (17:4), 2001, pp. 37-69.

Boehm, B.W. *Software Risk Management*, Los Alamitos, CA: IEEE Computer Society Press, 1989.

Boehm, B.W. and Ross, R. "Theory-W Software Project Management: Principles and Examples," *IEEE Transactions on Software Engineering* (15:7), 1989, pp. 902-916.

Canfora, G., Garcìa, F., Piattini, M., Ruiz, F., and Visaggio, C.A. "A Family of Experiments to Validate Metrics for Software Process Models," *The Journal of Systems and Software* 77, pp. 113-129.

Chin, W.W. *Structural Equation Modeling in IS Research*, ISWorld Net Virtual Meeting Center at Temple University, [On-Line]. Available at http://www.interact.cis.temple.edu/~vmc, 1998.

Cooprider, J., and Henderson, J.A. "Technology-Process Fit: Perspectives on Achieving Prototyping Effectiveness," *Journal of Management Information Systems* (7:3), 1990-91, pp. 67-87.

Dillman, D.A. *Mail and Internet Surveys: The Tailored Design Method*, Second Revision, John Wiley & Sons, 2000.

Fitzgerald, B. and O'Kane, T. "A Longitudinal Study of Software Process Improvement," *IEEE Software* (16:3), 1999, pp. 37-45.

Harter, D.E., Krishnan, M.S., and Slaughter, S.A. "Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development," *Management Science* (46:4), 2000, pp. 451-466.

Hayes, W. and Zubrow, D. "Moving On Up: Data and Experience Doing CMM-Based Process Improvement," Software Engineering Institute, *CMU/SEI-95-TR-008*, 1995, pp. 1-41.

Herbsleb, J., Carleton, A., Rozum, J., Siegel, J., and Zubrow, D. "Benefits of CMM-Based Software Process Improvement: Initial Results," Technical Report CMU/SEI-94-TR-13, Software Engineering Institute, 1994, pp. 1-64.

Herbsleb, J., and Goldenson, D.R. "A Systematic Survey of CMM Experience and Results," *Proceedings of International Conference on Software Engineering*, 1996, Berlin, pp. 25-30.

Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., and Paulk, M. "Software Quality and the Capability Maturity Model," *Communications of the ACM* (40:6), 1997, pp. 30-40.

Humphrey, W.S. *Managing the Software Process*, Reading, Mass., Addison-Wesley, 1989.

Ibbs, C.W., Kwak, Y.H. "Assessing Project Management Maturity," *Project Management Journal* (31:1), 2000, pp. 32-43.

Ingalsbe, J., Shoemaker, D., and Jovanovic, V. "A Metamodel for the Capability Maturity Model for Software," *Seventh Americas Conference on Information Systems*, 2001, pp. 1305-1313.

Iversen, J.H., Mathiassen, L., and Nielsen, P.A. "Managing Risk in Software Process Improvement: An Action Research Approach," *MIS Quarterly* (28:3), 2004, pp. 395-433.

Iversen, J.H. and Ngwenyama, O. "Problems in Measuring Effectiveness in Software Process Improvement: A Longitudinal Study of Organizational Change at Danske Data," *International Journal of Information Management* 26, 2006, pp. 30-43.

Jalote, P. *CMM in Practice: Processes for Executing Software Projects at InfoSys*, Reading, Mass: Addison-Wesley, 2000.

Jiang, J.J., Klein, G., Hwang, H.G., Huang, J., and Hung, S.Y. "An Exploration of the Relationship between Software Development Process Maturity and Project Performance," *Information & Management* (41), 2004, pp. 279-288.

Lawlis, P.K., Flowe, R.M., and Thordahl, J.B. "A Correlational Study of the CMM and Software Development Performance," *CrossTalk*, 1995, pp. 21-25.

Lyytinen, K., Mathiassen, L., and Ropponen, J. "Attention Shaping and Software Risk- A Categorical Analysis of Four Classical Risk Management Approaches," *Information Systems Research* (9:3), 1998, pp. 233-255.

March, J.G. and Shapira, Z. "Managerial Perspectives on Risk and Risk Taking," *Management Science* (33: 11), 1987, pp.1404-1418.

McFeely, B. "IDEAL: A User's Guide for Software Process Improvement," CMU/SEI-96-HB-001, Software Engineering Institute, Pittsburgh, PA, February 1996.

Myers, W. "Hard data will lead managers to quality," *IEEE Software* (11:2), 1994, pp. 100-101.

Nidumolu, S. "The Effect of Coordination and Uncertainty on Software Project Performance: Residual Performance Risk as an Intervening Variable," *Information Systems Research* (6:3), 1995, pp.191-219.

Paulk, M., Curtis, B., Chrissis, M. B. and Weber, C. "Capability Maturity Model for Software, Version 1.1," CMU/SEI-93-TR-024, Software Engineering Institute, Pittsburgh, PA, pp. 1-82, 1993.

Paulk, M., Weber, C., Curtis, B. and Chrissis, M. B. *The Capability Maturity Model for Software: Guidelines for Improving the Software Process*, Addison-Wesley, 1995.

"Process Maturity Profile, Software CMM 2004 Mid-Year End Update", August 2004, Software Engineering Institute, Carnegie Mellon University, http://www.sei.cmu.edu/sema/pdf/SW-CMM/2004aug.pdf, [Accessed January 24, 2006]

Post, G.V., and Diltz, D.J. "A Stochastic Dominance Approach to Risk Analysis of Computer Systems," *MIS Quarterly* (10:4), 1986, pp. 363-375.

Rai, A. and Al-Hindi, H. "The Effects of Development Process Modeling and Task Uncertainty on Development Quality Performance," *Information & Management* (37:6), 2000, pp. 335-346.

Ravichandran, T., and Rai, A. "Quality Management in Systems Development: An Organizational System Perspective," *MIS Quarterly* (24:3), 2000, pp. 381-416.

Riddle, W. "Advancing the State of the Art in Software Prototyping," in *Approaches to Prototyping*, R. Buddle, K. Kuhlenkamp, L. Mathiassen, and H. Zullighoven, eds. Berlin: Springer-Verlag, 1984, pp.19-28.

Rivard, S., Huff, S. "Factors of Success for End-User Computing," *Communications of ACM* (315), 1988, pp. 552-561.

Schmidt, R., Lyytinen, K., Keil, M., and Cule, P. "Identifying Software Project Risks: An International Delphi Study," *Journal of Management Information Systems* (17:4), 2001, pp. 5-36.

Statz, J., Oxley, D., and O'Toole, P. "Identifying and Managing Risks for Software Process Improvement," *Crosstalk – The Journal of Defense Software Engineering* (10:4), pp. 13-18.

Whitten, J.L., Bentley, L.D., Dittman, K.C. *Systems Analysis and Design Methods : 5th Edition*, McGraw-Hill Irwin, New York, 2000.

Yates, J.F. *Risk Taking Behavior*, Wiley, Chichester, 1992.