ARTICLE DE RECHERCHE

The impact of Software Process Maturity on Software Project Performance: The Contingent Role of Software Development Risk

Dany DI TULLIO & Bouchaïb BAHLI***

* Amazon EU
* ESC-Rennes School of Business

ABSTRACT

Despite growing efforts to improve software development processes, recurring concerns about software project performance remain largely present. The rate of software development project failure rate has been routinely documented in information systems (IS) research (Wallace, 2004; El-Masry and Rivard, 2010). The management of software development projects is often marked by inadequate planning, a poor grasp of the overall development process, and no clear management framework, even as the focus in software development shifts from a technology perspective to a more process-centric view (Slaughter, 2006). To address such concerns few CMM-based studies have examined the benefits and direct impact of software process maturity on software project performance but with mixed results. The present paper attempts to systematically examine the contingent role of software development risk on the impact of software process maturity level on software project performance. Guided by risk-based perspective in Software Engineering and CMM-based framework, an exploratory model was developed and tested. The premise of this paper is that software development risk plays a contingent role in the relationship between software process maturity and software project performance. Drawing on a sample of 107 organizations that have undergone official CMM appraisals, the results of partial least squares analysis of the data reveal initial evidence that (1) a positive effect of software process maturity level on software project performance while underscoring the negative effect of software development risk on software project performance, and (2) more importantly, the findings show that software development risk plays a contingent role software process maturity level on software project performance. For researchers, the integration of software development risk can provide a much needed linkage in the three fundamental constructs of CMM. From a managerial perspective, in order to foster a better software project performance, IS project leaders and managers should strongly emphasize devising effective software development risk assessment since a variation of this construct's level may strengthen or weaken the relationship between software development process maturity and software project performance.

Keywords: software process maturity, software development risk, software project performance.

RÉSUMÉ

Malgré les efforts croissants déployés pour améliorer les processus de développement des systèmes d'information (SI), la performance des projets d'informatisation demeure un sujet d'actualité dans l'industrie. Le taux d'échec de ces projets a été systématiquement documenté dans la littérature SI. (Wallace, 2004; El-Masry et Rivard, 2010). La gestion des projets de développement des SI est souvent caractérisée par une planification inadéquate, une mauvaise maîtrise du processus de développement global et un cadre de gestion imprécis (Slaughter, 2006). Le présent papier tente d'examiner le rôle modérateur du risque de développement SI sur l'impact du niveau de maturité du processus de développement SI sur la performance du projet. Un modèle basé sur le cadre CMM a été développé et testé. Des résultats révélateurs ont été observés à l'appui d'une enquête empirique effectuée auprès de 107 organisations ainsi que des résultats de l'analyse partielle des moindres carrés des données : (1) un effet positif du niveau de maturité du processus de développement SI sur la performance du projet tout en soulignant l'effet négatif du risque de développement SI sur la performance du projet et (2) le rôle contingent du risque de développement SI dans cette relation.

Pour les chercheurs SI, l'intégration du construit risque de développement SI souligne le rôle clé de ce construit dans la conceptualisation CMM. D'un point de vue managérial, pour obtenir une meilleure performance des projets SI, les chefs et gestionnaires de ces projets devraient mettre en œuvre une conception plus efficace d'évaluation des risques de développement SI puisqu'une variation du niveau de ce construit peut renforcer ou affaiblir la relation entre la maturité du processus de développement SI et la performance du projet.

Mots-clés : maturité des processus de développement des SI, risque du développement SI, performance des projets SI.

INTRODUCTION

Unquestionably, building software is a challenging endeavor. Given the pervasiveness of software in today's organizations, software project performance has become a major concern, underscoring the critical nature of the software development process. Software project performance is described in terms of two key aspects emphasized in the IS literature on software project performance (Barki et al. 2001): (1) process performance, which describes how well the software development process has been undertaken, and (2) product performance, which describes the performance of the system actually delivered to users. While technology continues to advance at a considerable pace, the software development process appears to lag behind (Shih and Huang, 2010, Slaughter et al., 2006; Rai and Al-Hindi, 2000). The management of software development projects is often marked by inadequate planning, a poor grasp of the overall development process, and no clear management framework, even as the focus in software development shifts from a technology perspective to a more process-centric view (Slaughter et al. 2006). The rate of software development project failure rate has been routinely documented in information systems (IS) research (El-Masri and Rivard, 2010; Lesca and Caron-Fasan, 2008; Wallace et al. 2004; Barki et al. 2001). The Standish Group research report (The Standish Group, 2009) shows a staggering 31.1% of projects will be cancelled before they ever get completed. Further results indicate 52.7% of projects will cost 189% of their original estimates. The cost of these failures and overruns are just the tip of the proverbial iceberg. The lost opportunity costs are not measurable, but could easily be in the trillions of dollars. One just has to look to the City of Denver to realize the extent of this problem. The failure to produce reliable software to handle baggage at the new Denver airport has cost the city $1.1 million per day (The Standish Group, 2009).

Carefully designed management practices are, therefore, needed to improve the software development process and gain better control over uncertain environments (Rivard and Mignerat, 2010; Iversen et al. 2004) and such practices are now emerging as viable solutions to the software crisis (Barki et al., 2001; Canfora et al., 2005). To address these concerns, significant efforts have lately focused on designing and improving software development processes with the objective of enhancing their software project performance (Shih and Huang, 2010; Iversen and Ngwenyama, 2006; Jiang et al., 2004). For instance, many firms are adopting the well-known Capability Maturity Model or CMM originally developed by the Software Engineering Institute or SEI (Paulk et al. 1993) – which 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" – to manage the software development process (Paulk et al. 1993. 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 with a substantial adoption of process improvement initiatives and the significant implementation efforts they entail (Unterkalmsteiner et al. 2012; Poeppelbuss et al. 2011). This model requires a considerable amount of time and effort to implement and often needs a major shift in culture and attitude given the costs and potential disadvantages of implementing the CMM, benefits must be evident to justify its continued use (Jiang et al. 2004). Nevertheless, despite these growing efforts to improve software development processes, recurring concerns about software project performance remain largely present (Subramanian et al. 2007) due to the identification, characterization, and control of confounding factors which is a challenging endeavor. Trienekens et al. (2001) cautioned that it is necessary to study the conditions under which the relationship between process improvement and associated outcome in order to increase the knowledge of these relationships. One major confounding construct in IS literature is software project risk which has long been claimed to be a major cause of software development project failure (Barki et al. 1993, 2001; Wallace et al. 2004; El-Masri and Rivard, 2010). In this study, we define software project risk as the potential of unwanted outcome and its assessment is approximated by identifying risk factors likely to influence the occurrence of that outcome (Bahli, 2003; 2005).

To address this concern, the objective of this paper is to systematically examine the contingent or a moderator role of software development risk on the impact of software process maturity level on software project performance. Guided by risk-based perspective in Software Engineering, an exploratory model was developed and tested. Therefore, the purpose of this study is to address two related research questions:

1. What is the impact of software process maturity on software project performance?
2. What is the impact and contingent role of software development risk on software project performance?

The contributions of this study are twofold. First, this paper provides a comprehensive conceptual model by synthesizing research using the risk-based perspective in Software Engineering. Our research extends prior research on software process maturity and software project performance in one important way: prior research has been tightly focused on software development process maturity and its impact on software project performance but has overlooked the inclusion of software development risk as it was originally suggested in the CMM model (Paulk et al. 1993; Boehm, 1991). Hence, we empirically included and tested the relationship between software development risk and software project performance. Second, this study investigates the importance of the interaction effect of software process maturity and software development risk on software project performance. Researchers have often stressed the importance of assessing software development risk to better assess the possible exposure and losses that may result (e.g., Wallace et al. 2004; Barki et al. 2001; Boehm, 1991).

This study suggests that the relationship between software process maturity and software project performance is moderated by the degree of software development risk. While we recognize that there might be other factors that may intervene in the relationship between software process maturity and software project performance, we have bounded and specifically focused our conceptualization of CMM as suggested by the SEI (Paulk et al. 1993) in order to reduce the conceptual and empirical complexity of our study and provide sound recommendations to both researchers and practitioners.

The remainder of this paper is organized into four sections. First, we provide an overview of prior literature on software process maturity, software development risk and software project performance. Second, we present the research model and hypotheses. Third, we describe the research methodology and results. This is followed by a discussion of the study's results, future research avenues, the study's limitations, and its conclusions.

I. THEORETICAL BACKGROUND

I.1. Software Process Maturity

Software process maturity or 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 (Wang et al., 2008). The underlying principles of SPI are key to 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. Many firms consider software process improvement a strategic issue, due to the fact that a failed process leads to failed software (Iversen and Ngwenyama, 2006; Ashrafi, 2003). Among the various models available, the most popular one is the CMM model and its extension CMM Integrated (Huang and Han, 2006; Iversen et al., 2004). The main assumption of this model is that the higher the maturity level attained, the higher the project performance and the lower is the risk of project failure (Chen, 2010).

In the present paper, we limit our study to organizations that have already adopted CMM, because only a few firms have yet been certified CMMI and the sample is too small to include them. For more information on why organizations still do not adopt CMMI, please see Staples et al. (2007). CMM has five maturity levels that define an ordinal scale for measuring the maturity of an organization's software process and for evaluating its software process capability. These levels also help organizations prioritize their improvement efforts (Jung and Goldensen, 2009; Paulk et al. 1993). CMM is a staged evolutionary model that classifies software process maturity into one of five levels – from 1 (lowest) to 5 (highest). For each level, CMM specifies process areas, which are areas on which the firm needs to focus in order to move to a higher process maturity level. Each process area is associated with goals that represent the requirements to be satisfied by the processes

in that process area (Huang and Han, 2006; Jalote, 2000). At each maturity level, specific process areas are used to assess the capability of existing processes as well as identify the areas that need to be strengthened in order to move to a higher level of maturity. From lowest to highest, the five levels are: Initial, Repeatable, Defined, Managed, and Optimized (Jiang et al., 2004; Ingalsbe et al., 2001).

Level 1 – Initial – is sometimes called anarchy or chaos. At this level, system development projects in organizations follow no prescribed processes. At level 2 – Repeatable – project management processes are established to track project costs, schedules, and functionality. The focus is on project management, not system development. Process maturity level 3 – Defined – is characterized by a standard system development process (methodology) that may be purchased or developed and which is integrated throughout the information systems unit or team within the organization. Measurable goals for quality and productivity are established at level 4 –Managed – such that detailed measures of the system development process and product quality are collected and stored. Finally, at level 5 – Optimized – the system development process is standardized and continuously monitored and improved, based on the measures and data analysis established at level 4 (Canfora et al., 2005).

Software process maturity has been tied to performance improvements in a number of case studies that reveal substantial value to organizations that have implemented well-conceived process improvement efforts (Dyba et al. 2005). Incorporating quality standards into software process improvement models has been shown to increase software development project performance (Unterkalmsteiner et al. 2012) and empirical research results have provided support for this relationship. For example, Bull HN Information Systems Inc., the U.S. subsidiary of Groupe Bull, one of the largest European systems integrators, has been using the CMM model as the source of goals for its software process improvements and software development projects (Jiang et al., 2004). Process improvement efforts have been beneficial on several levels, including schedule, coding time, testing time, and quality (Bellini et al. 2008). Herbsleb et al. (1994) reviewed software process improvement efforts in 13 organizations and found improvements in cycle time, defect density, and productivity. Similarly, a six-year study at Hewlett-Packard found that delivered defects were greatly reduced and cost savings of over $100 million were achieved through software process improvements (Myers, 1994).

Nevertheless, while the SEI has reported an increase in the number of process maturity certification holders (SEI.com), and despite a growing interest in testing empirically the CMM model and its impact on software project performance (Unterkalmsteiner et al. 2012), recent data from the SEI on firms that engage in CMM initiatives suggest, however, that there is a high number of failures. Out of 1638 organizations self-reporting initial assessments, only 34 percent had proceeded to a second assessment. Of those that proceded, 13 percent did not improve

their capability to develop quality software (Iversen et al. 2004).

In IS literature, there has been little research taking into account the risk factor in the relationship between software development maturity and software project performance. The goal 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 development maturity level and software development risk have any interaction effect on software project performance. Should this be the case, software development project managers will need to pay attention to improving both software processes and software risk management and not focus exclusively on software development methodologies per se. Hence we believe that this key observation contributes to the body of knowledge on software process improvement literature.

I.2. Software Development Risk

As a large proportion of software project failures are often reported in IS literature, the search for appropriate action to address this problem has attracted much attention (El-Masry and Rivard, 2010; Iacovou and Nakatsu, 2008; Wallace et al. 2004; Schmidt et al., 2001; Barki et al. 2001). Gibson (2004) argued that "the problem stems from senior and project management failing to assess and mitigate the risks of the change up front abd over a project's lifecycle and thereby increase the changes of success". Nevertheless, the concept of risk in IS research can be organized into two distinct streams: the rational decision theory perspective of risk (Boehm, 1989), and the behavioral perspective of risk (Lyytinen et al., 1998; March and Shapira, 1987).

The rational decision theory addresses the concept of risk in a quantitative manner or, in other words, as changes 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 that yields the best outcome among the available risk-return combinations (Yates, 1992). According to this view, it is necessary to assess the probabilities of undesirable events and their associated losses in order to measure the degree of risk. This renders quantitative assessments of risk a key concern (Boehm, 1989; Boehm and Ross, 1989). However, several difficulties arise when assessing risk with quantitative assessments of probabilities (Barki et al., 1993; Barki et al., 2001). In many cases, probability distributions of undesirable events are very difficult to assess and can be unreliable (Post and Diltz, 1986).

The behavioral perspective of risk (March and Shapira, 1987) more accurately defines the assumptions underlying most risk management approaches. Risk research that takes 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 of risks, or master them through sequential pruning exercises (Lyytinen et al.,

1998). These approaches to assessing risk focus on the factors that influence the occurrence of undesirable events instead of the probabilities that they will occur. Barki et al. (1993, 2001) directly addressed the difficulty of coupling the concept of risk with outcome probabilities, devising an instrument comprised of uncertainty and risk variables derived from previous research on risk and uncertainty. 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., 2001). The authors grouped project characteristics that influence the occurrence of project failure along five dimensions: technological newness, application size, expertise, project complexity, and organizational support, and assessed the magnitude of potential loss due to project failure (Barki et al., 1993, 2001; Marciniak, 1996; Sauer et al. 2007). Later studies found that increased levels of fit between these five dimensions of the risk exposure of a software project and its management profile have a positive effect on software project performance (Barki et al., 2001), and that risk exposure negatively impacts software project effectiveness, thus providing further support for the behavioral perspective of risk (Jiang et al., 2004; Sauer et al. 2007; El-Masry and Rivard, 2010). Hence, in this study, we adopt the behavioral perspective of risk (risk ass a potential of unwanted consequences) to assess its influence on the performance of software development projects.

Software development risk has been found to negatively affect the overall software project performance. Significant relationships were found between individual project risk variables, such as a lack of general expertise in the development team, the intensity of conflicts among team group members, and a lack of clarity in role definitions within the team, and project efficiency (El Amrani and Saint-Léger, 2011; Huang and Han, 2006; Jiang et al. 2004). Project efficiency incorporated such items as considerations for the amount and quality of work, adherence to schedules and budgets, speed and efficiency, and the ability to meet goals (Wang et al., 2008; Jiang et al. 2004). Significant relationships were also found with other specific items in software development projects, such as top management involvement and user support, and a development team's perception of its performance. The findings indicate that when software development team members do not perceive that their projects benefit from user support and/or top management support, the team does not perform well (Shih and Huang, 2010; Jiang et al., 2004).

I.3. Software Project Performance

The literature on performance goes back to the aesthetic readings of Barnard (1938) who stated that "when a specific desired end is attained we shall say that the action is "effective." When the unsought consequences of the action are more important than the attainment of the desired end and are dissatisfactory, effective action, we shall say, is "inefficient." Project perfor-

mance is viewed differently by each of the stakeholders in the system development effort (Gibson, 2004). It is desirable to incorporate a breadth of success aspects when considering project performance (Jiang et al. 2004). As such, project performance includes software engineering issues of efficiency and effectiveness, as well as organizational issues of control, communication, and organizational knowledge. Efficiency is often considered to be measured by the quality of the software product, adherence to budgeted time and money, and cost of the software operation. Effectiveness is considered to be the applicability and adaptability of the software. Performance in software development can be broadly divided into two streams of research: the social view of software project performance and the technical view of software project performance (Wang et al., 2008; Aladwani, 2002). The social view of software project performance refers to research focused largely on issues related to the attributes and behaviors of project members. Software development projects are placed in a specific social context that considers human behaviors and the overall organizational environment in which software projects take place. In this view, technology factors as determinants of software project outcomes are largely submerged by social variables (Aladwani, 2002). On the other hand, the technical view of software project performance focuses on issues related to the characteristics of the software project itself. Contextual variables such as human behavior are usually overlooked in these types of studies (Aladwani, 2002).

The social view of software project performance is largely characterized by paying close attention to the human behavioral aspects of software development projects and their impact on project performance. This view of software project performance can take many forms. For example, studies of project performance through team members have included variables such as team skill, managerial involvement, and variance in team experience. The first two variables as well as a small variance in team experience were found to enable more effective team processes than software development tools and methods (Shih and Huang, 2010; Guinan et al., 1998). Moreover, earlier studies focused on the relationships between participation, influence, conflict, and conflict resolution among team members and their influence on project success (Robey et al., 1993). These studies have found a strong positive relationship between conflict resolution and project success and a moderate relationship between participation and project success, where project success included such aspects as adherence to budgets, schedules, and quality of work (Robey et al., 1993).

The technical view of software project performance is more concerned with issues relevant to the characteristics of the software project itself. Contextual variables such as human behavior are usually overshadowed by technology, task, process, and project characteristics (Shih and Huang, 2010; Aladwani, 2002). For instance, various aspects of technical performance dimensions were found to affect software development, notably the implementation of an information repository

containing parameterized design and code, which facilitates software reusability for new system development (Ravichandran and Rai, 2000).

While the main hypothesis behind CMM (and all process-oriented approaches) seems to suggest that "quality products follow disciplined processes", an alternate view of performance, however, has garnered consistent support over the last few years: the assessment of software project performance as a two-dimensional construct comprised of both a process dimension and a product dimension (Huang and Han, 2006; Wallace et al. 2004; Barki et al. 2001; Nidumolu, 1995, 1996). 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 improve the final output (tangible or intangible) of the production process, but to also give due consideration to the processes used to obtain such 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 schedule and budget targets may very well deliver products of poor quality (Jiang et al., 2004; Wallace et al. 2004; Barki et al. 2001; Nidumolu, 1995). Insightful results have been obtained by taking this two-dimensional view of performance. More specifically, risk-based research has helped advance our knowledge of the relationship between risk and performance in software development projects (Barki et al., 2001).

The results suggest that, in order to improve both the process and the product performance of software development projects, a project's risk management profile needs to vary with the project's exposure to risk: projects exposed to low degrees of risk require a different risk management profile than projects characterized by high risk. 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 improve the performance of software development projects. Hence, the two-dimensional view of performance was selected for this study as it clearly addresses the importance of adopting both the process and product perspectives of performance. Table 1 shows an excerpt of studies that examined the relationship between CMM levels and performance.

II. RESEARCH MODEL AND HYPOTHESES

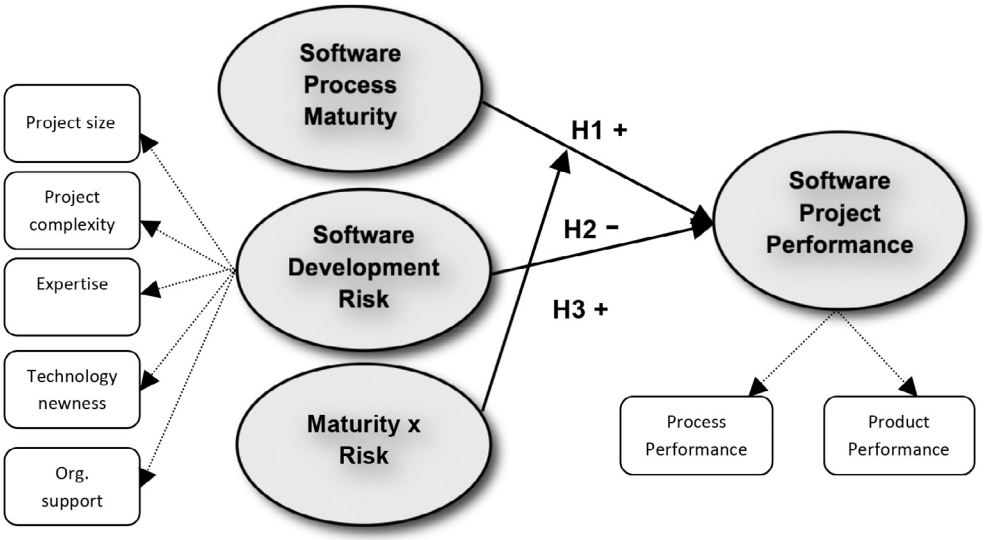
The conceptual model shown in Figure 1 was developed to answer the key research questions of this paper:

- *What is the impact of software process maturity on software project performance?*
- *What is the impact and contingent role of software development risk on software project performance?*

Table 1: Prior Literature

Study	Independent Variables	Dependent Variables	Outcome
Jung and Goldenson (2009)	Process improvement	Schedule deviation	Negative relationship between process improvement and schedule deviation
Subramanian et al. (2007)	Software Development Process maturity	IS project performance	CMM levels do associate with higher software quality and project performance
Cater-Steel et al. (2006)	Process improvement	IS project performance	CMM levels do associate with higher software quality and project performance
Schalken et al. (2006)	CMM Process maturity	productivity improvement	Software Development Process maturity is positively associated to productivity improvement
Damian and Chisan (2006)	Process Maturity	Improvement initiative	Process maturity unrelated to the evaluated improvement initiative, and the Hawthorne effect
Jiang et al. (2004)	Software Development Process maturity	IS project performance	CMM levels do associate with higher software quality and project performance
Harter and Slaughter (2003)	Software Development Process maturity	Effort, quality, cycle-time	CMM improvements reduce cycle-time and effort
Edbert et al. (2001)	Process improvement	Cost of software production	Negative relationship of process improvement and cost of software production
Krishnan et al. (2000)	Maturity level	Quality, life-cycle productivity	Capability and process factors improve quality, and quality improves life-cycle productivity
Clark (2000)	Process maturity	Effort	One level change in maturity reduced effort by 3%-%15%.
Krishnan and Kellner (1999)	CMM practices	Quality	Consistent adoption of CMM processes reduces defects
Diaz and Sligo (1997)	CMM (4-5)	Productivity	Each CMM level improves productivity
Paulish and Carleton (1994)	software-development process	performance of the software-development	Improving the software-development process improves the quality of software products and the overall performance of the software-development organization.

Fig. 1. Research Model



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. Barki et al.'s (2001) validated constructs were used in this study: software project performance and software development risk. Furthermore, in order to capture the maturity of software development practices, we decided to survey organizations that had been officially appraised using the CMM for software (SW-CMM). We asked these organizations about their certified maturity level (from 1-initial to 5-optimized). The decision to focus only on SW-CMM certified organizations is due to the fact that, although CMMI (CMM Integration) is the extended version of the model, a large number of firms are still using SW-CMM. We, therefore, surveyed SW-CMM-appraised organizations in order to obtain an adequate sample size and then developed testable hypotheses that were grounded in prior research in the areas of software process improvement, risk in information systems, and software project performance. In the following section, we will discuss our study's hypotheses.

II.1. Software process maturity and 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 (Carter-Steel et al. 2006; Schalken et al. 2006; Damien and Chison, 2006; Jiang et al. 2004). More specifically, as orga-

nizations progress in terms of the maturity of their software processes, performance indicators such as project costs and schedule tend to improve. Organizations demonstrating high maturity in terms of their software development processes are more likely to successfully adhere to cost and schedule targets (Huang and Han, 2006; Lawlis et al. 1995). Higher levels of software process maturity also positively affect staff morale as well as the ability to meet budget targets (Herbsleb and Goldenson, 1996). Other performance measures are also addressed, such as software product quality, system development cycle time, and development effort (Edbert et al. 2001; Krishnan et al. 2000; 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 for software products (Subramanian et al., 2007; 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, we hypothesize:

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

II.2. Software Development Risk and Software Project Performance

Software development risks were found to be significantly related to both process performance and product performance in software development projects (Wallace et al., 2004; Ropponen and Lyytinen, 2000). Indeed, factors such as organizational environment risk, user risk, requirements risk, project complexity risk, planning and control risk, as well as team risk have significant negative impacts on both the process and product performance in software development projects (Wallace et al., 2004). Taken together, these variables indicate the negative impact of software development risk on both the process and product dimensions of performance, and thus underscore the need to address software development risk when considering key organizational concerns such as the performance of software development projects (Barki et al. 2001; Na et al., 2004; Nidumolu, 1996). These findings provide evidence as to the generalizability of the impact of software development risk on software project performance. More specifically, a project's level of software development risk may negatively influence software project performance, as hypothesized below.

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

Although the literature provides specific recommendations for the value of software process maturity and its impact on software project performance, it says little about the contingent role of software development risk in this relationship. Information systems researchers adopting the contingency approach software development risk have been strongly influenced by research in organizational contingency

theory (Barki et al. 2001). In this study, we adopted Venkatraman (1989, p.425) conceptualization of moderation or interaction between two variables. "According to the moderation perspective, the impact that a predictor variable has on a criterion variable is dependent on the level of a third variable, termed here as a moderator. The fit between the predictor and the moderator is the primary determinant of the criterion variable" (Venkatraman, 1989, p.424). The predictive ability of software development maturity will vary across different degree of software development risk. According to this view, software development projects managed with approaches that fit the demands imposed by the degree of risk of the project's environment will be more successful than projects that do not. Extending this reasoning to software development process improvement, the level of software development maturity and its impact of software development project performance would be contingent on the degree of software development risk. Therefore, we hypothesize:

H3: Software development risk moderates the relationship between software development process maturity level and software project performance.

III. RESEARCH METHODOLOGY

II.2. Research Variables and Measures

The variables used in this study were adopted from prior research (Barki et al. 2001) and fall into three constructs: software development processes maturity, software project performance, and software development risk. 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 steps required to lay the foundations for continuous software process improvement (Paulk et al., 1995). The CMM framework is comprised of 18 key process areas, including software project planning, organization process focus, software quality management, and defect prevention (Paulk et al., 1995). A software process is assigned to the highest maturity level if it meets the goals in the 18 key process areas of CMM. The CMM for Software (SW-CMM) process maturity level of each of the organizations polled in this study was previously determined by the SW-CMM lead appraisers of the Carnegie Mellon Software Engineering Institute. Organizations are certified at a given maturity level when they participate in the official SW-CMM-Based Appraisal for Internal Process Improvement (CBA-IPI) conducted by SEI-authorized lead appraisers. Only individuals from officially appraised organizations were invited to participate in the study. Respondents were asked to specify, on a scale of 1 to 5, their organization's maturity level as it was last determined by a SEI-authorized lead appraiser.

The instrument used to measure software development risk was developed and validated by Barki et al. (1993; 2001). The software development risk construct is comprised of software de-

velopment risk factors grouped along five dimensions: technological newness, project size, expertise (reversed), organizational support, and project complexity. The construct consists of a total of 46 items organized into five dimensions of risk factors; all measured using a 7-point Likert-type scale ranging from "strongly disagree" to "strongly agree".

The software project performance construct used in this study was adopted from Barki et al. 2001; Nidumolu, 1995, 1996), who presented 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 to the present study, as it directly refers to the process performance of software development projects, a key measurement concern when assessing the impact of software process improvements. More specifically, this construct is comprised of 24 items that together assess software project performance along two dimensions: process performance, which concerns the quality of the software development process, and product performance, which considers the performance of the system, product, or output delivered to the end-user. Twelve variables assess process performance along three dimensions: learning, control, and quality of interactions. The twelve other variables evaluate product performance and also fall into three categories: operational efficiency, responsiveness, and flexibility. All 24 items were rated on a 7-point Likert-type scale ranging from 1 (very poor) to 7 (very good).

II.2. Data Collection

A survey research methodology was adopted. A database of approximately 500 organizations with official SW-CMM-appraisals was used, and respondents were asked to answer an online questionnaire containing the items used to assess the study's constructs. Dillman's (2000) recommendations for developing and administering Web surveys were followed. The respondents were IS managers who could answer questions on their organization's recent software development projects. The data collection produced 107 usable questionnaires, for a response rate of 21.4%. Half of the organizations (53%) in the sample represent the software development and IT services sectors, while 40% are relatively large organizations with over 1,000 employees. Moreover, almost half of the organizations in the sample (46%) had software development teams of fewer than 20 members for their last completed software project, while 20% had over 60 team members. Table 2

Table 2: Sample Descriptive Statistics

Position	
IS Executive	98
Not Reported	9
Average number of team members on last software development project	
3-20 members	49
21-40 members	31
41-60 members	21
Not reported	6
Primary sector	
Software Development & Services	45
IT Services & Solutions	32
Healthcare	12
Finance	14
Government	4
Organization size (number of employees)	
Under 500 employees	39
501-1,000 employees	16
1,001-5,000	26
5,001-50,000 employees	12
Over 50,000 employees	5
Not reported	9
CMM Level	**Number of organizations**
1 Initial	14
2 Repeatable	21
3 Defined	43
4 Managed	17
5 Optimized	12

provides descriptive statistics for the sample.

Guidelines suggested by Hair et al. (1998) were followed in order to screen the completed questionnaires for missing data and outliers. A few missing values were noted, and replacement data were generated using a mean substitution. The data set was also screened for univariate and multivariate outliers. Since most of the variables were measured on a 7-point scale, all of the data were kept for analysis, as no extreme values were found. Software process maturity level was assessed on a 5-point scale (1: initial; 5: optimized). A t-test was used to check whether the team size, sector and organization size affect our model's results. No significant effects were observed.

IV. DATA ANALYSIS AND RESULTS

IV.1. Assessment of the Measurement Model

Prior to assessing the models, data were parceled using the technique proposed by Bagozzi and Edwards (1998). This consists of aggregating

measurable items to create item parcels, which are used as indicators. Parceling involves averaging or summing items, and it is widely used in the IS literature (Barki et al. 2001; Spears and Barki, 2010). Its main advantage is that it provides stable and reliable estimates. Aggregation can be based on statistical or rational grounds. In the case of the unidimensional constructs, items were averaged (after assessing the measurement model) to create homogeneous composites that were then used as indicators of the model constructs when testing the structural model. The same procedure was followed for the second-order constructs for each dimension (Bagozzi and Edwards 1998). In addition, a factor analysis indicated a structure that was close to the one proposed. We used Principal Component Analysis (PCA) as the extraction method and Varimax with Kaiser Normalization as the rotation method. The rotation converged in 18 iterations. Only loadings > 0.45 were retained.

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. The loading parameters estimated by PLS consist of the links between the measures and the constructs. 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 with a loading score of 0.707 or higher (Rivard and Huff, 1988). However, 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 has a value of 0.81 while expertise equals 0.92. These values are shown in bold in Table 3. Three other risk variables – technological newness, organizational support, and project complexity – were dropped, since their loadings on the risk construct were too low (below 0.50). In the case of the software project performance construct, both variables have high loading values (above 0.9, as shown in bold in Table 3).

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 3, based on this criterion, the software development risk construct (= 0.83) and the software project performance construct (= 0.95) both demonstrate sufficient reliability.

Furthermore, 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. Indeed, software development risk has an AVE of 0.84, and software project performance has an AVE of 0.97, as shown in

Table 3. Therefore, significant variance was shared between each item and its respective construct, indicating adequate variance extracted and construct reliability.

In order to evaluate convergent and discriminant validity, a comparison was made between the average variance extracted of each variable and the variance shared between the constructs (the squared correlations between the constructs), as suggested by Fornell and Larcker (1981). A PLS run was performed to obtain the covariance matrices of all measures used to assess the loadings of the variables on their construct. High convergent validity coupled with low discriminant validity is present when the loading of variables within a construct is high on that construct and low on others. As shown in Table 3, this is the case with specific variables for all three constructs. Indeed, both the process performance and product performance variables load on their respective performance construct with reliability scores well above 0.5 and exhibit very low loadings on other constructs. However, not all risk variables cleanly loaded onto the software development risk construct, and were therefore dropped. In other words, they either did not show high loadings on their respective constructs (thus displaying low convergent validity) or did not exhibit lower loadings on the other constructs (thus displaying low discriminant validity). Indeed, technological newness, organizational environment, and project complexity were dropped, as their loadings on the risk construct were too low. All other variables whose loading values are shown in bold in Table 3 were kept, since they exhibit clear convergent and discriminant validity.

Moreover, Table 4 shows the square root of the average variance extracted for all three constructs. The values on the diagonal represent the square roots of the average variance extracted (AVE). The off-diagonal values display correlations among constructs. For ad-

Table 3: Reliability, Convergent and Discriminant Validity Assessment

Constructs	***Number of items***	***Mean***	***SD***	***CMM Level***	***Soft Dev Risk*** (α=0.83, ICR=0.82, AVE=0.84)	***Soft Project Performance*** (α=0.95, ICR=0.94, AVE=0.97)
Software Development Process Maturity (SM)	1	3.6	2.8			
Technology Newness (TN°	3	4.3	3.1			
Project Size (PS)	5	3.9	1.8		0.26	
Expertise (E)	32	5.6	2.4	**1**	**0.81**	
Org Support (OS)	9	2.9	2.1		**0.92**	
Project Complexity (PC)	2	5.7	3.2		0.11	
Process Performance (PP1)	12	6.2	2.4		0.41	0.95
Product performance(PP2)	12	5.9	2.1			0.93

Table 4: Variance Shared Between Constructs

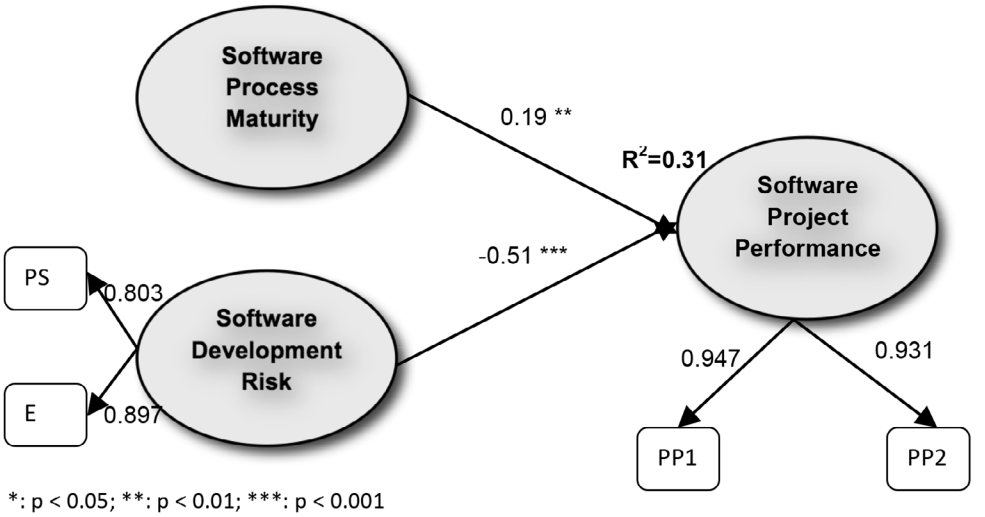
Construct	*CMM Level*	*Software Development Risk*	*Project Size*	*Expertise*	*Software Project Performance*
CMM Level	1				
Risk	0.02	**0.84**			
Project size	0.22	0.31	**0.79**		
Expertise	0.27	0.19	0.11	**0.82**	
Software Project Performance	0.04	0.25	0.42	0.39	**0.97**

equate discriminant validity, the values in bold should be greater than those in the corresponding row and column. As can be seen, each construct is clearly distinguishable from the other constructs, as the variance shared by any two of them is less than the variance shared by a construct and its measures. Therefore, the discriminant validity and the convergent validity are satisfactory.

IV.2. Assessment of the Structural Model

The primary objective of this study is to provide empirical evidence on the relationships between software development process maturity and software project performance while assessing the contingent role of software development risk in this relationship. 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 1998), which provided both of these values. First, we performed PLS run without the interaction effect of software development risk and software development process maturity. 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 should lead to higher levels of software project performance. The 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 (R2) of software project performance was 31%. A detailed diagram of the structural model results is provided below (see Figure 2). Furthermore, a PLS run with the two main risk dimensions and their effects on software project performance shows a significant effects of project size (path = - 0.27, $p < 0.001$) and expertise (path = - 0.32, $p < 0.001$). These results show that project size and expertise play a significant role as risk dimensions that affect negatively software project performance.

Fig. 2. Results of the Main Effects Model

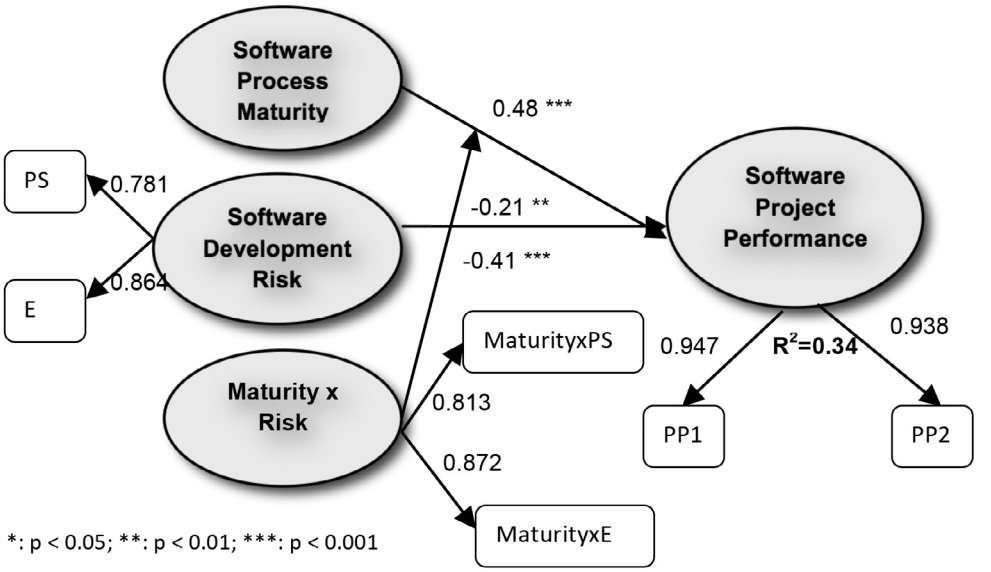


In a subsequent PLS run, we integrate the interaction effect of software development risk and software development process maturity. Our analysis provides support for each hypothesis. The findings indicate that software development process maturity levels are positively and significantly related to software project performance (path = 0.48; $p < 0.01$), and that software development risk is negatively and significantly related to the performance of software development projects (path = -0.21; $p < 0.01$). The results (see Figure 3) give a standardized beta of an interaction effect (path = -0.41; $p < 0.01$), for a total R-squared of 0.34. These results imply that one standard deviation increase in software development risk will not only impact software project performance by -0.21, it will also decrease the impact of software process maturity to software project performance, from 0.48 to 0.07. The main effect, therefore, has an effect size of F of 0.04, which is between a small and medium effect (Cohen and Cohen, 1983). The beta estimates suggest the conditions under which risk becomes a dominant factor overshadowing maturity, in the case of a high risk project. In the following section, we will discuss these interesting results. Furthermore, a PLS run with the two main risk dimensions and their interaction effects on the relationship between software development risk and software development process maturity shows a significant effects of project size (maturity x project size, path = 0.29, $p < 0.001$) and expertise (maturity x expertise, path = 0.31, $p < 0.001$). Notice that the expertise dimension was measured with reversed items (see Appendix).

V. DISCUSSION

The objective of this study was to examine the contingent role of software

Fig. 3. Results of the Interaction Model



development risk on the relationship between software development process maturity levels and software project performance. Metrics grounded in prior research were used in order to conduct a large-scale survey that would test the above relationships. Data was collected from organizations that were officially CMM-appraised by leading SEI-authorized appraisers. Partial least squares (PLS) were 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.

This study addresses an important issue for organizations managing software development projects: Does the level of maturity of an organization's software development process affect its performance in software projects, and how does software development risk influence this relationship? The findings of this study indicate that software project performance increases with higher levels of CMM software process maturity. In other words, organizations that are higher on the maturity ladder exhibit higher levels of performance in their software projects. This corroborates prior research concerning the benefits of CMM software process improvement initiatives (Unterkalmsteiner et al. 2012). It thus seems apparent that, as a staged evolutionary model providing an incremental roadmap towards process control and continuous process improvement, CMM can now be clearly tied to performance metrics. Organizations that adopt key CMM practices can expect process improvements that will translate into performance increas-

es in the form of both process and product performance. Moreover, this study shows that a high level of software project risk is negatively associated with software development project performance. Results highlight unique risk dimensions (project size and expertise) that have a significant impact on software project risk. Software development project managers should therefore keep in mind such factors as large software projects, software developers' management abilities, as well as user involvement in development projects. These factors must be closely monitored as evidenced by the negative influence of risk factors on a software development project's bottom line: its performance. Managers must therefore be mindful of the many risks involved in software development projects, while being proactive and effectively identifying and mitigating threats that may hinder a project's performance.

A central insight of our study is that software development risk moderates the relationship between software development process maturity and software project performance. Regardless of the maturity level attained by an organization, if software development risk is high, then software project performance suffers. The findings imply that one standard deviation increase in software development risk will not only impact software project performance by -0.21, it will also decrease the impact of software development process maturity to software project performance, from 0.48 to 0.07. So it is important to pay attention to the interaction between software development process maturity and software development risk when assessing software project performance. Contrary to equity investments, where it is assumed that high-risk equity will generate a high return, software development projects are "products" that clearly reflect the challenges encountered in achieving consistent performance when project-related risks are high. Both the professional literature and the research literature are full of examples of companies that are certified at CMM level 5 yet are still plagued with software project failures. Risk is an inherent component of software development projects, and the drivers of software development risk need to be taken into account. The following section discusses these findings with regard to their implications for research and practice.

VI. IMPLICATIONS FOR PRACTICE AND RESEARCH

This study makes significant contributions to research on software development process improvement. Despite prevalence of CMM-based research, the contingent role of software development risk in the relationship between software development process maturity and software project performance remains relatively unexplored. To fill this gap, the present paper proposed, to the best of our knowledge, this is the first study to integrate CMM maturity levels from officially appraised organizations along with software project performance and software development risk variables – into an integrative framework. This has not only provided a preliminary empirical investigation of the impact of CMM

maturity levels on software project performance, but also provides researchers with an comprehensive research model that can be used, expanded upon, and explored in more detail in future studies.

These results should also be of great interest to both researchers and practitioners. For researchers, tying CMM process maturity to software project performance taking into consideration software development risk this study emphasizes the contingent role of software development risk in CMM studies. A variation of this construct's level may strengthen or weaken the relationship between software development process maturity and software project performance. For practitioners, this study suggests that there are reliable results demonstrating a return on investment of CMM process improvement initiatives. Software development managers or IS executives who may be reluctant to invest in CMM improvement initiatives without evidence of a payoff now have preliminary findings that will undoubtedly influence their ultimate decisions.

CMM models require a considerable amount of time and effort to implement. An organization requires between 18 and 30 months to rise one full maturity level. Moreover, official appraisals at a given maturity level are obtained by participating in assessments conducted by SEI-authorized lead appraisers. Their services are typically valued at $50,000 per appraisal, depending on the travel involved and the size of organization, while many organizations also employ consulting services, which can generate considerable extra costs (Ingalsbe et al., 2001). Hence, in order to foster a better software project performance, the findings of this study suggest that IS project leaders and managers should strongly emphasize devising effective software development risk assessment.

For further researcher, it would be interesting to delve further into each maturity level in order to examine the effectiveness of specific key process areas with regard to performance metrics. This would provide a more detailed look at the inner workings of each maturity level, and would represent a natural extension of the present study. Moreover, as this research has shown, the significant influence of software development risk on the performance of software development projects, it would be interesting to evaluate how specific risk management practices incorporated into the CMM model effectively mitigate these risk factors. For example, CMM level 3 contains an integrated software management process area that stresses the need for managers to develop specific abilities, such as methods and procedures for identifying, managing, and communicating software risks (Paulk et al., 1993). How effective are these methods at actually mitigating software development risks? In light of the growing concern for risk in system development, coupled with the wider adoption of the CMM model for software process improvement, this is one line of inquiry that deserves further investigation. Longitudinal studies also need to be considered. As the type of survey research conducted here consists of one-time snapshots of given organizations, invaluable information could be obtained by following up

with organizations that have progressed from one level to the next. Performance could be assessed as organizations move up the CMM maturity scale, in order to detect and analyze variations in performance.

Furthermore, software development often has too much change during the time that the team is developing the product to be considered a defined process. A set of predefined steps may not lead to a desirable, predictable outcome because software development is a decidedly human activity: requirements change, technology changes, people are added and taken off the team, and so on. In other words, the process variance is high (Larman, 2004). For instance, in the conventional IS project risk management approach, the development the team goes through a lengthy analysis of each risk to determine its severity. Once all risks are identified, they are quantitatively prioritized based on a calculated risk severity. However, in the agile approach, the development team analyzes and prioritizes the current risks, using only their perceptions to determine the severity of each risk at the iteration level. The analysis of each risk and its severity must be completed in far less time and effort than the conventional process allows. If the team misjudges, its short iterations and strong feedback allow it to reprioritize the risks in the next iteration when it has fresher information from which to work (Ahrendts and Marton, 2008). Further research on software development risk needs to take into account the type of approach used to develop such software as risk treatment varies accordingly.

VII. STUDY LIMITATIONS

Although the results of the present study provide interesting insights for both researchers and practitioners, more research is needed to overcome some of its limitations and further explore and expand upon its findings. Since this is a preliminary study on the contingent role of software development risk in the relationship between software development process maturity and software project performance, this study did not include the internal components of each CMM level. Therefore, it is not possible to determine which specific key process areas or practices have a greater impact on process and product performance and sensitive to software development risk level. Furthermore, this study provides a static picture of given organizations, as the questionnaire specifically asked respondents to provide answers with regard to their firm's most recently completed software development project. Additional insights into variations in performance and an organization's progress along the CMM maturity scale may be obtained through future longitudinal studies. Indeed, obtaining multiple observations of each organization as it advances up the maturity scale would provide further insights into the effects of CMM process improvement.

CONCLUSION

Despite the advances made CMM-based research, few studies have examined the relationship between software development process maturity and software project performance. The present paper proposed not only a val-

idation of this linkage but also the integration of the software development risk construct and its contingent role into this relationship. This study represents the first systematic attempt to use a single, comprehensive research model to assess the effects of software development process maturity and software development risk as a moderator construct on software project performance. While the integration of this conceptualization and the study findings contribute to software development research by providing a much needed integration of CMM three main constructs, much still needs to be done to explore each construct's potential role in varied stages of software development projects and to further explicate the role of each construct's dimensions in an integrated model. It is our hope that our investigation has provided insights to researchers and practitioners, as well as a deeper understanding of the influence of software development process maturity on software project performance and the important moderating role played by software development risk in this relationship.

REFERENCES

Ahrendts, F., & Marton, A. (2008). "IT Risikomanagement leben - Wirkungsvolle Umsetzung für Projekte in der Softwareentwicklung". Heidelberg: Springer Verlag.

Aladwani, A.M. (2002), "An Integrated Performance Model of Information Systems Projects", *Journal of Management Information Systems*, Vol. 19, no 1, p. 185-210.

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

Ashrafi, N. (2003), "The impact of software process improvement on quality: in theory and practice" *Information & Management*, Vol. 40, no. 7, p. 677-690.

Bagozzi, R. P., J. R. Edwards, J.R. (1998), "*A general approach for representing constructs in organizational research*", Organizational Research Methods**,** Vol. no. 1, p. 45-87.

Bahli, B & Rivard, S. (2005), "Validating Measures of Information Technology Outsourcing Risks Factors," *Omega,* Vol. 33, no. 2, p. 175-87.

Bahli, B. & Rivard, S. (2003), "The information technology outsourcing risk: a transaction cost and agent theory-based perspective," *Journal of Information Technology*, Vol. 18, no. 3, p. 211-221.

Barki, H., Rivard, S., and Talbot, J. (1993), "Toward an Assessment of Software Development Risk," *Journal of Management Information Systems*, Vol. 10, no. 2, p. 203-225.

Barki, H., Rivard, S., and Talbot, J. (2001), "An Integrative Contingency Model of Software Project Risk Management," *Journal of Management Information Systems*, Vol. 17, no. 4, p. 37-69.

Barnard, Chester I. (1938). The Functions of the Executive. Cambridge, MA: Harvard University Press.

Bellini, C., Pereira, R and Becker, J. (2008), "Measurement in software engineering: From the roadmap to the crossroads," International Journal of Software Engineering and Knowledge Engineering, vol. 18, no. 1, p. 37–64.

Boehm, B.W. (1991). "*Software Risk Management*", Los Alamitos, CA: IEEE Computer Society Press.

Boehm, B.W. & Ross, R. (1989), "Theory-W Software Project Management: Principles and Examples," *IEEE Transactions on Software Engineering,* Vol. 15, no. 7, p. 902-916.

Canfora, G., Garcìa, F., Piattini, M., Ruiz, F., & Visaggio, C.A.(2005), "A Family of Experiments to Validate Metrics for Software Process Models," *The Journal of Systems and Software* ,Vol. 77, p. 113-129.

Cater-Steel, A. P., Tan, W. -G., & Toleman, M. A. (2006). "Challenge of adopting multiple process improvement frameworks", *European Conference on Information Systems*, Goteborg, Sweden.

Chen, L. (2010), "Business-IT alignment maturity of companies in China", *Information&Management*, Vol. 47, no. 1, p. 9-16.

Chin, W.W. (1998), "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.

Clark, B. (2000), "Quantifying the effects on effort of software process maturity". *IEEE Software Journal,* Vol. 17, no. 6, p. 65-70.

Damian. D., & Chisan, J., (2006), "An Empirical Study of the Complex Relationships between Requirements Engineering Processes and Other Processes that Lead to Payoffs in Productivity", *IEEE Transactions on Software Engineering*, Vol. 32, no. 7, p. 433 - 453.

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

Dybå, T., Kitchenham, B. A. & Jorgensen, M. (2005), "Evidence-based software engineering for practitioners", *IEEE Software*, Vol. 22, no. 1, p. 58-65.

El Amrani, R & Saint-Léger, G. (2011), "Le pari des centres de competences dans la stabilization des organisations en phase post-projet ERP", *Systèmes d'Information et Management*, Vol. 16, no. 1, p. 69-112.

El-Masri, M & Rivard, S. (2010), "Specifying the Software Project Risk Construct", *AMCIS* 2010, Lima, Perou.

Fornell C & Larcker D. (1981), "Evaluating structural equation models with unobserved variables and measurement error," *Journal of Marketing Research*, Vol. 18, p. 39-50.

Gibson, C.F. (2004), "IT-enabled Business Change: An Approach to Understanding and Managing Risk," Working paper, MIT Sloan, p. 1-14.

Guinan, P.J., Cooprider, J.G., & Faraj, S. (1998), "Enabling Software Development Team Performance During Requirements Definition: A Behavioral Versus Technical Approach," *Information Systems Research*, Vol. 9, no. 2, p 101-125.

Hair, J.F., Anderson, R.L., Tatham, R.L., Black, W.C. (1998), Multivariate Data Analysis, Prentice Hall.

Harter, D.E., Krishnan, M.S., & Slaughter, S.A. (2000), "Effects of Process Maturity on Quality, Cycle Time, and Effort in Software Product Development," *Management Science*, Vol. 46, no. 4, p. 451-466.

Harter, D., & Slaughter. S. (2003), "Quality improvement and infrastructure activity costs in software Development". *Management Science,* Vol. 49, no. 6, p. 784-796.

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

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

Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W., & Paulk, M. (1997), "Software Quality and the Capability Maturity Model," *Communications of the ACM*, Vol. 40, no. 6, p. 30-40.

Huang, S. & Han, W. (2006), "Selection priority of process areas based on CMMI continuous representation", *Information & Management*, Vol. 43, no. 3, p. 297-307.

Iacovou, C. & Nakatsu, R. (2008), "A risk profile of offshore outsourced development projects", *Communications of ACM*, Vol. 51, no. 6, p. 89-94.

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

Iversen, J.H., Mathiassen, L., & Nielsen, P.A. (2004), "Managing Risk in Software Process Improvement: An Action Research Approach," *MIS Quarterly,* Vol. 28, no. 3, p. 395-433.

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

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

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

Jung, H & Goldenson, D. (2009), "Evaluating the relationship between process improvement and schedule deviation in software maintenance," *Information and Software Technology*, Vol. 51, p. 351-361.

Krishnan, M. S., C. H. Kriebel, S. Kekre, T. Mukhopadhyay. (2000), "An empirical analysis of productivity and quality in software products", Management *Science*. Vol. 46, no. 6, p. 745–759.

Krishnan, M. S. & Kellner, M. (1999). "Measuring Process Consistency: Implications for Reducing Software Defects," *IEEE Transactions on Software Engineering*, Vol. 25, no. 6, p. 800-815.

Larman, C. (2004). *Agile and Iterative Development: A Manager's Guide*. Boston: Addison Wesley.

Lawlis, P.K., Flowe, R.M., & Thordahl, J.B. (1995), "A Correlational Study of the CMM and Software Development Performance," *Cross Talk*, p. 21-25.

Lesca, N & Caron-Fasan, M-L. (2008), "Facteurs d'échec et d'abandon d'un projet de veille stratégique: retour d'expériences", *Systèmes d'Information et Management*, Vol. 13, no. 3. p. 17-42.

Lyytinen, K., Mathiassen, L., & Ropponen, J. (1998), "Attention Shaping and Software Risk- A Categorical Analysis of Four Classical Risk Management Approaches," *Information Systems Research,* Vol. 9, no. 3, p. 233-255.

Marciniak, R. (1996), "Management des projets informatiques : complexité et gestion des conflits," Systèmes d'information et management, Vol. 1, no. 1, p. 27-50.

March, J.G. & Shapira, Z. (1987), "Managerial Perspectives on Risk and Risk Taking," *Management Science*, Vol. 33, no. 11, p.1404-1418.

Myers, W. (1994), "Hard data will lead managers to quality," *IEEE Software* (Vol. 11, no. 2, p. 100-101.

Na, K.S., Li, X., Simpson, J.T., & Kim, K.Y., (2004), "Uncertainty Profile and Software Project Performance: A Cross-National Comparison", *Journal of Systems and Software*, Vol. 70, no. 2, p. 155-163.

Nidumolu, S., (1996), "A Comparison of the Structural Contingency and Risk-Based Perspectives on Coordination in Software Development Projects", *Journal of Management Information Systems*, Vol. 13, no. 2, p. 77-113

Nidumolu, S. (1995), "The Effect of Coordination and Uncertainty on Software Project Performance: Residual Performance Risk as an Intervening Variable," *Information Systems Research*, Vol. 6, no. 3, p.191-219.

Nunnally, J.C., 1978. Psychometric Theory, McGraw-Hill, NY.

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

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

Poeppelbuss, J., Niehaves, B., Simons, A., and Becker, J. (2011), "Maturity Models in Information Systems Research: Literature Search and Analysis", *Communications of the Association for Information Systems*, Vol. 29, no. 27, p. 505-532.

Post, G.V., & Diltz, D.J. (1986), "A Stochastic Dominance Approach to Risk Analysis of Computer Systems," *MIS Quarterly*, Vol. 10, no. 4, p. 363-375.

Rai, A. & Al-Hindi, H. (2000), "The Effects of Development Process Modeling and Task Uncertainty on Development Quality Performance," *Information & Management,* Vol. 37, no. 6, p. 335-346.

Ravichandran, T., & Rai, A. (2000), "Quality Management in Systems Development: An Organizational System Perspective," *MIS Quarterly*, Vol. 24, no. 3, p. 38 1-416.

Rivard, S., Huff, S. (1988), "Factors of Success for End-User Computing", *Communications of ACM*, Vol. 31, no. 5, p. 552-561.

Rivard, S & Mignerat, M. (2010), "Between Aquiescence and Manipulation: IS Project Managers Responses to Institutionalized Practices", *Systèmes d'Information et Management*, Vol. 15, no. 2, p. 9-44.

Robey, D., Smith, L.A., & Vijayasarathy, L.R., (1993), "Perceptions of Conflict and Success in Information Systems Development Projects", *Journal of Management Information Systems*, Vol. 10, no. 1, p. 123-139.

Ropponen, J. & Lyytinen, K. (2000), "Components of software development risk: how to address them", *IEEE Transactions on Software Engineering*, Vol. 26, no. 2, p. 98-112.

Sauer, C., Gemino, A. & Reich, B. (2007), "The impact of size and volatility on IT project performance", *Communications of the ACM*, Vol. 50, no. 11, p. 79-84.

Schalken, J. J. P., Brinkkemper, S., & van Vliet, H. (2006). "A method to draw lessons from project postmortem databases". Software. Process Improvement. Practices., p. 35-46.

Schmidt, R., Lyytinen, K., Keil, M., & Cule, P. (2001), "Identifying Software Project Risks: An International Delphi Study," *Journal of Management Information Systems*, Vol. 17, no. 4, p. 5-36.

Shih, C. and Huang, S. (2010), "Exploring the relationship between organizational culture and Software Process Improve-

ment Deployment", *Information & Management*, Vol. 47, no. 5-6, p. 271-281.

Slaughter, S., Levine, L., Ramesh, R., Pries-Heje, J. (2006), "Aligning software processes and strategy", MIS *Quarterly*, Vol. 30, no. 4, p. 891-918.

Spears, J. L., & Barki, H. (2010), "User Participation in Information Systems Security Risk Management," *MIS Quarterly*, Vol. 34, no. 3, p. 503-522.

Staples, M., Niazi, M., Jeffery, R., Abrahams, A., Byatt, P., Murphy, R. (2007), "An exploratory study of why organizations do not adopt CMMI", *Journal of Systems and Software*, Vol. 80, no. 6, p. 883-895.

Subramanian, G; Jiang, J; Klein, G. (2007), "Software quality and IS project performance improvements from software development process maturity and IS implementation strategies", *Journal of Systems and Software*, Vol. 80, p. 616-627.

The Standish Group. The Standish Group Research Report. 2009. www.standishgroup.com

Trienekens J., Kusters, R. & van Solingen, R. (2001), "Product focused software process improvement: concepts and experiences from industry", *Software Quality Journal*, Vol. 9, p. 269–281.

Unterkalmsteiner, M; Gorschek, T; Islam, A; Cheng, C; Permadi, R and Feldt, R. (2012), "Evaluation and Measurement of Software Process Improvement - A Systematic Literature Review", *IEEE Transactions on Software Engineering*, Vol. 38, no. 2, p. 398-482.

Venkatraman, N. (1989), "The concept of fit in strategy research: toward verbal and statistical correspondence", *Academy of Management Review*, Vol. 9, p. 513- 525.

Wallace, L., Keil, M., Rai, A. (2004), "How Software Project Risk Affects Project Performance: An Investigation of the Dimensions of Risk and an Exploratory Model", *Decision Sciences*, Vol. 35, no. 2, p. 289-321.

Wang, E., Ju, P., Jiang, J., Klein, G. (2008), "The effects of change control and management review on software flexibility and project performance". Information & Management, Vol. 45, no. 7, p. 438-443.

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

APPENDIX: MEASURES

Measure of CMM Software Process Maturity Level

Construct	*Item*	*Measure*
CMM Software Process Maturity Level	CMMLevel	What is your CMM for Software (SW-CMM) Maturity Level as it was last determined by a SEI-authorized lead appraiser?

Measure of Software Project Performance (Adopted from Barki et al. (2001)

Variable	*Item*	*Measure*
Process Performance	SPP1	Knowledge acquired by firm about use of key technologies
	SPP2	Knowledge acquired by firm about use of development techniques
	SPP3	Knowledge acquired by firm about supporting users' business
	SPP4	Overall knowledge acquired by firm through the project
	SPP5	Control over project costs
	SPP6	Control over project schedule
	SPP7	Adherence to auditability and control standards
	SPP8	Overall control exercised over the project
	SPP9	Completeness of training provided to users
	SPP10	Quality of communication between DP (data processing) and users
	SPP11	Users' feelings of participation in project
	SPP12	Overall quality of interactions with users
Product Performance	SPP13	Reliability of software
	SPP14	Cost of software operations
	SPP15	Response time
	SPP16	Overall operational efficiency of software
	SPP17	Ease of use of software
	SPP18	Ability to customize outputs to various user needs
	SPP19	Range of outputs that can be generated
	SPP20	Overall responsiveness of software to users
	SPP21	Cost of adapting software to changes in business
	SPP22	Speed of adapting software to changes in business
	SPP23	Cost of maintaining software over lifetime
	SPP24	Overall long term flexibility of software

Measure of Software Development Risk (Adopted from Barki et al. (2001)

Variable	*Variable*	*Item*	*Measure*
Technological Acquisition	N/A	Risk1	The new system required new hardware.
		Risk 2	The new system required new software.
		Risk 3	A large number of hardware suppliers were involved in the development of the system.
Project Size	N/A	Risk 4	A large number of software suppliers were involved in the development of the system.
		Risk 5	There were a large number of people on the project team.
		Risk 6	There were a large number of different "stakeholders" on the project team (e.g., IS staff, users, consultants, suppliers, customers).
		Risk 7	The project size was large.
		Risk 8	There are a large number of users using the system.
Expertise	Lack of Team's General Expertise	Risk 9	Ability to work with uncertain objectives
		Risk 10	Ability to work with top management
		Risk 11	Ability to work effectively as a team
		Risk 12	Ability to understand the human implications of a new system
		Risk 13	Ability to carry out tasks effectively
	Lack of Team's Expertise with the Task	Risk 14	In-depth knowledge of the functioning of user departments
		Risk 15	Overall knowledge of organizational operations
		Risk 16	Overall administrative experience and skill
		Risk 17	Expertise in the specific application area of the system
		Risk 18	Familiar with this type of application
	Lack of Team's Development Expertise	Risk 19	Development methodology used in this project
		Risk 20	Development support tools used in this project (e.g., DFD, flowcharts, ER models, CASE tools)
		RiskK21	Project management tools used in this project (e.g., PERT charts, Gantt diagrams, walkthroughs, project management software)
		Risk 22	Implementation tools used in this project (e.g., programming languages, database languages)

Variable	*Variable*	*Item*	*Measure*
Expertise (Continued)	Lack of User Support	Risk 23	Users had a negative opinion about the system meeting their needs.
		Risk 24	Users were not enthusiastic about the project.
		Risk 25	Users were not an integral part of the development team.
		Risk 26	Users were not available to answer questions.
		Risk 27	Users were not ready to accept the changes the system entailed.
		Risk 28	Users slowly responded to development team requests.
		Risk 29	Users had negative attitudes regarding the use of computers in their work.
		Risk 30	Users were not actively participating in requirement definition.
	Lack of User Experience	Risk 31	Users were not very familiar with system development tasks.
		Risk 32	Users had little experience with the activities supported by the new application
		Risk 33	Users were not very familiar with this type of application.
		Risk 34	Users were not aware of the importance of their roles in successfully completing the project.
		Risk 35	Users were not familiar with data processing as a working tool.
Organizational Environment	Extent of Changes Brought	Risk 36	The system required that a large number of user tasks be modified.
		Risk 37	The system led to major changes in the organization.
	Resource Insufficiency	Risk 38	In order to develop and implement the system, the scheduled number of people-day was insufficient.
		Risk 39	In order to develop and implement the system, the dollar budget provided was insufficient.
	Lack of Clarity of Role Definitions	Risk 40	The role of each member of the project team was not clearly defined.
		Risk 41	The role of each person involved in the project was not clearly defined.
		Risk 42	Communications between those involved in the project were unpleasant.
	Intensity of Conflicts	Risk 43	There was a great intensity of conflicts among team members.
		Risk 44	There was a great intensity of conflicts between users and team members.
Application Complexity	N/A	Risk 45	Large number of links to existing systems
		Risk 46	Large number of links to future systems

Bouchaïb BAHLI is Professor of Information Technology (IT) at ESC-Rennes School of Business, Rennes, France. His research focuses on outsourcing information services, software project risk management and, IT adoption modeling. His work has been published in such journals such as Decision Support Systems, Information & Management, Journal of Information Technology, OMEGA, Communications of the Association of Information Systems, Requirement Engineering Journal, and others.

Adresse : ESC-Rennes School of Business - 2, Rue Robert D'Arbrissel - 35065 Rennes

Mail : bouchaib.bahli@esc-rennes.fr

Nassim BELBALY is Professor of Information Technology Management. He was a Visiting Scholar and a Principal Researcher at the Anderson Business School at UCLA, Los Angeles. His current research interests are innovation management, new product development, Open Source Systems, and design science research. He publishes in peer reviewed journals in IT management relative to his research fields.

Adresse : Groupe Sup de Co Montpellier Business School - 2300, Avenue des Moulins - 34185 Montpellier

Mail : n.belbaly@supco-montpellier.fr

Adel BELDI is an Assistant Professor of MIS and Accounting. He obtained his Ph.D. degree from South Paris University and a Research Master degree from Paris-Dauphine University. His research interests include ERP adoption, implementation and use. His research findings have been published in peer reviewed journals. He is also the author of several articles and book chapters on CRM implementation.

Adresse : IESEG School of Management Lille-Paris - 3, rue de la Digue - 59000 Lille

Mail : a.beldi@ieseg.fr

Nordine BENKELTOUM est docteur en sciences de gestion de MINES ParisTech, il est maître de conférences en sciences de gestion au LM²O (Laboratoire de Modélisation et de Management des Organisations) de l'École Centrale de Lille. Ses travaux portent sur le management des systèmes d'information et la gestion de l'innovation.

Adresse : Université Lille Nord de France, École Centrale de Lille - Cité Scientifique - 59651 Villeneuve d'Ascq

Mail : nordine.benkeltoum@ec-lille.fr

Dany DI TULLIO is a business intelligence engineer at Amazon's European Head Office in Luxembourg, where he leads the business intelligence activities of the Kindle tablet & eInk devices division in the EU. He received his Ph.D. in management information systems at the Queen's School of Business, Queen's University. His research has been presented at the International Conference on Information Systems (ICIS) and published in the Journal of MIS and Communications of the Association of Information Systems.

Adresse : Amazon EU - 31/35 Rives de Clausen - 2165 Luxembourg

Mail : ditullio@amazon.com

Anis KHEDHAOURIA is Assistant Professor of Information Technology Management. He obtained his Ph.D. degree from the University of Savoy in Alpine Eastern France. His research interests include IT-enabled creativity, creativity and innovation management, and Mobile Internet Services. He published in peer reviewed journals in IT management relative to his research fields.

Adresse : Groupe Sup de Co Montpellier Business School - 2300, Avenue des Moulins - 34185 Montpellier

Mail : a.khedhaouria@supco-montpellier.fr

Khaireddine MOUAKHAR est docteur en sciences de gestion et professeur à l'Ecole de Management de Normandie. Au sein du laboratoire Métis, ses travaux portent sur la construction des modèles d'affaires, l'innovation et la stratégie des sociétés de services en logiciels libres.

Adresse : Ecole de Management de Normandie - 30, rue de Richelieu - 76087 Le Havre

Mail : kmouakhar@em-normandie.fr

Albéric TELLIER est docteur en sciences de gestion, habilité à diriger des recherches. Maître de conférences à l'Institut d'Administration des Entreprises de l'Université de Caen Basse-Normandie, il développe des travaux sur les situations de compétition technologique, les stratégies collectives et les réseaux d'innovation.

Adresse : Université de Caen Basse-Normandie (IAE) - 3, rue Claude Bloch - 14075 Caen

Mail : alberic.tellier@unicaen.fr