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

by

Dany Di Tullio

A Thesis

in

The John Molson School of Business

Presented in Partial Fulfillment of the Requirements for the Degree of Master of Science
in Administration at Concordia University
Montreal, Quebec, Canada

May 2005

Your file Votre référence
ISBN: 0-494-10324-8
Our file Notre référence
ISBN: 0-494-10324-8

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Dany Di Tullio

Entitled: The Impact of CMM Process Maturity Levels and Software Development Risk on the Performance of Software Development Projects

and submitted in partial fulfillment of the requirements for the degree of

Master of Science in Administration

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair

_____ Examiner

_____ Examiner

_____ Supervisor

Approved by _____
Chair of Department or Graduate Program Director

_____20_____ _____
Dean of Faculty

ABSTRACT

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

Dany Di Tullio

A model of software development process maturity, the Capability Maturity Model for Software (SW-CMM) is a staged evolutionary model which describes five levels of process maturity through which an organization can progress to define, assess, and improve its software development processes. Despite its ever-increasing adoption, there remain too few empirical and generalizable findings when it comes to key questions regarding the model's adoption. Researchers and practitioners are still struggling to determine how CMM-based process improvement efforts affect key organizational concerns such as software project performance and in turn how the performance of software projects is affected by the threat of risks in today's dynamic and complex business environment.

In addressing this knowledge gap, this study proposes a research model which is grounded in prior research and that will allow for a first known empirical examination of the relationships between CMM process maturity levels and the performance of software development projects while assessing the impact of software development risk on performance. Two hypotheses were derived from the model for empirical testing. Data was collected from 107 organizations that were officially appraised at a given CMM maturity level. Results using PLS provide considerable support for the hypotheses.

As expected, CMM software development process maturity levels have a direct and significant impact of the performance of software development projects. Furthermore, evidence of the negative and significant influence of software development risk on software project performance was also found. Discussions on the potential for future research and implications for practice are also presented.

DEDICATION AND ACKNOWLEDGMENTS

I would like to dedicate this thesis to my parents, Donato and France Di Tullio, and to my brother, Ian. I am grateful beyond words for your unconditional support and sound advice in all of my endeavours.

Merci infiniement.

I would also like to extend my sincere gratitude and appreciation to my Master's thesis advisor, Dr. Bouchaib Bahli, for his great kindness, invaluable advice, and continuous support. You have made my M.Sc. experience both enjoyable and rewarding.

I also thank my thesis committee members, Dr. Rustam Vahidov and Dr. El-Sayed Abou-Zeid, for their constructive comments and feedback.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

Software systems are pervasive in today's organizations. From small applications to large-scale enterprise systems, software has become the informational nervous system of the modern enterprise. The importance of software reliability is thus a major concern and stresses the critical nature of the software development process. However, while the advancement of information technologies 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 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 (Rai and Al-Hindi, 2000; Humphrey, 1989). Carefully conceived management practices are thus needed to improve software development performance and gain improved control over uncertain and risky environments and are now emerging as viable solutions to the software crisis (Humphrey, 1989; Barki et al., 1993; Fitzgerald and O'Kane, 1999; Barki et al., 2001).

In a direct attempt to address this, the Software Engineering Institute (SEI) at Carnegie Mellon University recommended a set of key software process improvement areas that can be addressed by firms wishing to improve software development. These same processes were later integrated into an evaluative evolutionary framework called the Capability Maturity Model (CMM) for Software (SW-CMM) (Paulk et al., 1993). The CMM for Software essentially 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). Organizational adoption of the CMM is

an ever growing phenomenon. The software process maturity model is now used by major firms in every sector of the economy and around the world (Herbsleb et al., 1997).

1.1 Research Questions

Accordingly, the CMM's growing adoption (Herbsleb et al., 1997), the significant implementation efforts it entails (Hayes and Zubrow, 1995), and the growing concern for risk in software development (Barki et al., 2001), stress the need for providing answers to fundamental questions that remain elusive. Essential to the advancement of knowledge pertaining to software process improvement, software development risk, and more specifically to the Capability Maturity Model and software project performance lie the following key research questions that remain unanswered:

- *What is the impact of CMM software development process maturity levels on the performance of software development projects?*

- *What is the relationship between software development risk and the performance of software development projects?*

1.2 Research Objectives

To answer these questions, a new conceptual model aimed at allowing for a sound empirical investigation is proposed. Grounded in prior research in the areas of software process improvement, software project performance, and risk in information systems, a series of hypotheses are developed to test the presence (or absence) of the effects of CMM software process maturity levels on the performance of software development projects as well as the

influence of software development risk on performance. Metrics aimed at answering each of the above questions are used. Data is then collected to validate these measures and examine their correlations and relationship strengths to one another.

In short, the fundamental purpose of this research endeavour consists of developing a new conceptual framework that will for the first time provide scientifically valid and reliable empirical evidence as to the relationships between CMM software development process maturity levels and the performance of software development projects while assessing the influence of software development risk on software project performance.

1.3 Importance of the Research

Despite the ever-increasing adoption of the Capability Maturity Model for Software, there remain too few empirical and generalizable findings when it comes to key questions regarding the model's adoption. Researchers and practitioners are still struggling to determine whether and how CMM-based software process improvement efforts affect key organizational concerns such as software project performance in today's dynamic and complex business environment. 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 sound scientific approach in researching the Capability Maturity Model for Software (Herbsleb et al., 1997). Careful collection and analysis of data are needed in an effort to provide both researchers and practitioners with hard evidence that will justify the time and effort required to use the CMM to improve the software development process and ultimately the end product (Fitzgerald and O'Kane, 1999). In addressing this need, this study is a clear step in that direction.

The remainder of the thesis is organized as follows. A review of the relevant literature is presented in Section 2. Section 3 contains a presentation of the conceptual framework and the development of hypotheses. Section 4 provides details about the research methodology. Data analysis and research results are presented in Section 5 and a discussion of the findings follows in Section 6. Finally, a conclusion is found in Section 7.

II. LITERATURE REVIEW

The literature review is organized along the three constructs that comprise this research: the Capability Maturity Model for Software, software project performance, and software development risk.

A. THE CAPABILITY MATURITY MODEL FOR SOFTWARE

1. An Overview of Software Process Improvement

1.1 Process Thinking in Software Development

'*An important first step in addressing software problems is to treat the entire software task as a process that can be controlled, measured, and improved*' (Humphrey, 1989).

In the modern business world, the shift from task-oriented thinking to a process-centric view of the organization has been underway for several years. Michael Hammer and James Champy were early advocates of the process view and popularized it in 1993 with the publication of a book titled *Reengineering the Corporation* (Hammer and Champy, 1993). As opposed to task-based thinking which entails the fragmentation of work into simple components and their assignment to specialized workers, process-based thinking aligns the tasks, activities, and behaviours of individuals within an organization toward achieving a common goal (Zahran, 1998). It proposes that we organize around outcomes, not tasks (Hammer and Champy, 1990).

These same process management principles influenced the software engineering field. The concept of process thinking was introduced to the software industry by Watts Humphrey

through his classic book *Managing the Software Process* published in 1989 (Humphrey, 1989). In this book, Humphrey stresses the fact that few software professionals ever mention technology when discussing their key problems when it comes to software development. Instead, uncontrolled change, arbitrary schedules, ill-defined processes, and poor process management are but some of the major concerns mentioned (Humphrey, 1989). Hence, Humphrey proposes that the greatest potential for improvement lies in defining and controlling the software process which essentially consists of '*the set of actions required to efficiently transform a user's needs into an effective software solution*' (Humphrey, 1989). According to this view, a process focus brings discipline to the software project's activities and allows each individual to align himself with his co-worker towards achieving the project goals.

1.2 Software Process Maturity

Adopting a process view of software development requires a thorough understanding of the concept of maturity of software processes. The concept of maturity naturally implies immaturity which leads to the distinction between immature and mature software organizations.

In immature software organizations, software processes are generally improvised by individuals during the course of the software development effort (Paulk et al., 1993). There are generally no specific mechanisms in place that prescribe specific actions to be taken in particular cases (Paulk et al., 1995). Even if there are some processes that have been decided upon, they are not carefully followed or enforced. An important characteristic of an immature organization is its reactive nature. Firefighting, or the focus on finding solutions to immediate crises, is frequent. Project success largely depends on a skilled team of software developers coupled with exceptional managers. Product functionality and quality are frequently compromised to meet deadlines. There is little understanding by the organization of the processes involved in the

development of the software product as it frequently relies on ad hoc and chaotic, undocumented processes (Paulk et al., 1995).

On the other hand, a mature software organization possesses the skills and abilities to effectively manage software development and maintenance processes (Paulk et al., 1993). Planned and documented processes provide employees with a detailed roadmap to the successful completion of the software project. Moreover, effective organization-wide communication allows for both new and current employees to access and understand predefined processes. Employee roles and responsibilities within each process are clear and concise throughout the software development endeavour and across the organization (Paulk et al., 1995). Objective quantitative methods are used to constantly evaluate both product and process quality in an effort for continuous improvement of both. Historical data are the basis for establishing realistic budgets and schedules. The expected results for cost, schedule, quality, and functionality are therefore regularly met (Paulk et al., 1995).

1.3 The Emergence of Software Process Improvement Frameworks

Software applications and systems have become ever-present in our daily lives. We increasingly rely on software in a wide variety of areas ranging from simple text processing applications to advanced navigational systems used to guide planetary probes and robotic rovers. Software has become pervasive because of its power; a power to assist us with mundane repetitive tasks and also provide us with sophisticated means for conducting a seemingly limitless number of complex activities.

Software technology is advancing at a very rapid pace. However, despite these advances, large numbers of software projects continue to fail at various levels. Budget, schedule, product quality, as well as the actual completion of software projects continue to be areas of concern. The ability to develop, deliver, and maintain software systems that are reliable, usable, and within the prescribed budget and schedule seems to elude most organizations involved in software development (Paulk et al., 1995). Solutions to these problems have been sought and some proposed for many years. Lifecycle models and structured methods are some of the methods suggested for better understanding software development activities (Zahran, 1998). Recently, a process-oriented approach to software projects was proposed as organizations began to realize that their fundamental problem mainly consisted of their inability to manage their software processes (Paulk et al., 1995). This process-oriented approach to managing software development projects has been gaining ground in the software industry ever since (Herbsleb et al., 1997). Several frameworks were developed to address software processes, including SPICE (Software Process Improvement and Capability dEtermination), ISO 9000-3, Bootstrap, Trillium and the Capability Maturity Model (CMM) for Software (SW-CMM) (Krishnan et al., 1999) [1] Among these process models, the CMM for Software has emerged as one of the most widely used and influential frameworks in the field of software development (Jalote, 2000).

1.4 The Capability Maturity Model

The Capability Maturity Model focuses on the various processes involved in software development. It presents the key elements of an effective software process in describing an evolutionary improvement path for software organizations from ad hoc, immature processes to mature, disciplined ones (Paulk et al., 1995). It was created and developed by attentive

[1] Henceforth, the following terms will be used interchangeably: Capability Maturity Model for Software, SW-CMM, and CMM.

observations of best practices in both software and non-software organizations. The framework is thus based on actual practices while reflecting the best of the state of the practice as well as the needs of individuals performing software process improvement and software process appraisals (Paulk et al., 1995).

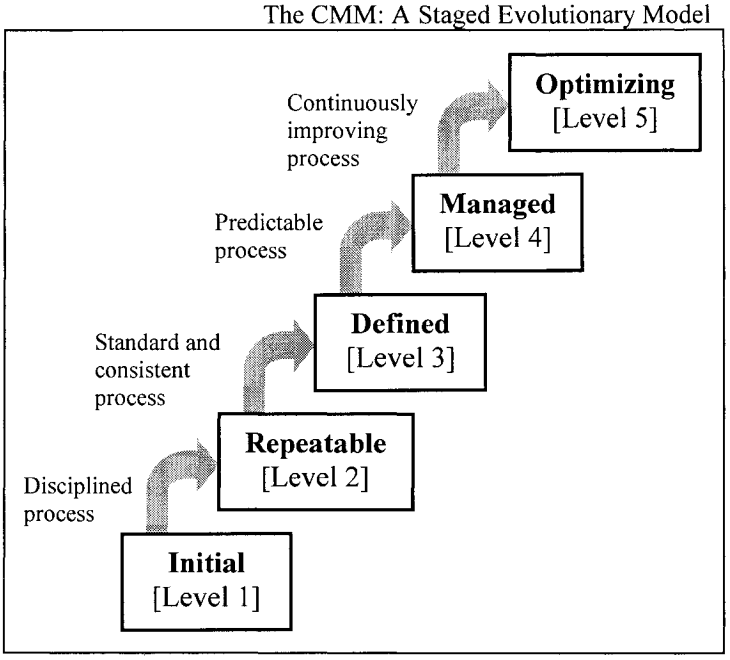
It is in 1986 that the Software Engineering Institute (SEI) at Carnegie Mellon University began developing a process maturity framework that would help firms improve their software processes (Paulk et al., 1993). One of the main instigators of this initiative was the United States Department of Defence which requested a method for assessing the capability of its software contractors. An early report entitled 'Characterizing the software process: a maturity framework' set the foundations for the process maturity model (Humphrey, 1987). The Software Engineering Institute later released a brief description of the process maturity framework which was later more thoroughly presented and explained in the book *Managing the Software Process* (Humphrey, 1989) which largely popularized the message of software process maturity (Zahran, 1998). After several years of working with the model, the SEI evolved the maturity framework into the Capability Maturity Model (later renamed Software CMM or SW-CMM).

In the last several years, judging by its acceptance in the software industry, the CMM has already been a major success. It has spread far beyond its origins in U.S. military avionics applications, and is now used by major organizations worldwide and in every sector of the economy (Herbsleb et al. 1997).

The Capability Maturity Model for Software provides organizations with guidelines on how to gain control over their processes for developing and maintaining software (Paulk et al., 1993). This set of recommended practices offers guidance to firms in selecting process improvement strategies by determining current process maturity and identifying the issues they

consider crucial to the improvement of software quality and processes (Paulk et al., 1995). Through sustained effort at improving specific activities, organizations can progressively improve their software processes in order to achieve lasting gains in software process capability (Paulk et al., 1995).

Figure 1
The CMM: A Staged Evolutionary Model



The Capability Maturity Model is a staged evolutionary model. It categorizes software process maturity into five levels- from level 1 (lowest) to level 5 (highest) (see Figure 1). For each level, the CMM specifies key process areas (KPA) which consist of areas on which a firm should focus in order to move to a higher process maturity level. Each key process area is associated with goals that represent the requirements to be satisfied by the processes for that key process area (Jalote 2000). At different maturity levels, key process areas can be used for assessing the capability of existing processes as well as for identifying the areas that need to be strengthened so as to move the process to a higher level of maturity. The five levels are Initial, Repeatable, Defined, Managed, and Optimized (Ingalsbe et al. 2001). Level 1-*Initial*, is sometimes called anarchy or chaos. At this level, system development projects follow no prescribed processes. At level 2- *Repeatable*, project management processes have been established to track project costs, schedules, and functionality. The focus is on project management, not systems development. At level 3-*Defined*, a standard system development process (methodology) is purchased or developed, and its use is integrated

throughout the information systems unit or team within the organization. At level 4-*Managed*, measurable goals for quality and productivity are established. 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 measures and data analysis established in level 4 (Whitten et al. 2000).

2. The Five Levels of Software Process Maturity

As a staged maturity model, the CMM provides a framework comprised of five evolutionary steps which consist of the model's process maturity levels. These five maturity levels represent an ordinal scale which allows organizations to situate themselves in terms of their process capability and better understand the necessary steps that need to be taken to lay the foundations for continuous process improvement (Paulk et al., 1995).

A maturity level is a '*well-defined evolutionary plateau toward achieving a mature software process*' (Paulk et al., 1995). A set of goals make up each maturity level. The achievement of these goals ultimately results in an increase in the process capability of the organization (Paulk et al., 1995).

2.1 The Initial Level

The initial level is typically characterized by an unstable organizational environment when it comes to the development and maintenance of software. At this level, processes are often chaotic and ad hoc (Humphrey, 1989). Even if there are some processes that have been decided upon, they are not carefully followed or enforced. Product functionality and quality are frequently compromised to meet deadlines. There is little understanding by the organization of

the processes involved in the development of the software product (Paulk et al., 1995). However, in spite of this, Level 1 organizations frequently have the capacity to develop software products that are functional albeit they may be over budget and schedule having been made possible by the efforts and heroics of a few (Paulk et al., 1995).

2.2 The Repeatable Level

Policies for managing software projects and procedures to implement these strategies are decided upon and established at the Repeatable Level. Moreover, it is at this level that project management concerns are addressed. Processes are established notably to track cost, schedule, and product functionality (Herbsleb et al., 1997). Realistic plans and project commitments are made possible based on past experience with previous projects. Earlier project successes can now be more easily repeated.

2.3 The Defined Level

Documentation is a key activity at the Defined Level as both software engineering and management processes are documented, standardized, and integrated into a coherent software process for the organization (Herbsleb et al., 1997). This organization-wide standardized software process provides the firm with the foundation for major and continuing progress (Humphrey, 1989). Software projects are now accomplished more effectively using the standardized software processes which allow for stable and repeatable software engineering and management activities (Paulk et al., 1995).

2.4 The Managed Level

The establishment and use of detailed measures of the software development processes and software product quality are conducted at the Managed Level (Herbsleb et al., 1997). An organizational measurement program allows for the specification of quantitative quality goals for both software processes and products. Measurement data on process performance are collected and provide the means for evaluating and controlling variations in process performance (Paulk et al., 1995). Using these measurements, an organization's software development processes become quantifiable and predictable and process variations beyond prescribed limits can be quickly identified and dealt with (Paulk et al., 1995).

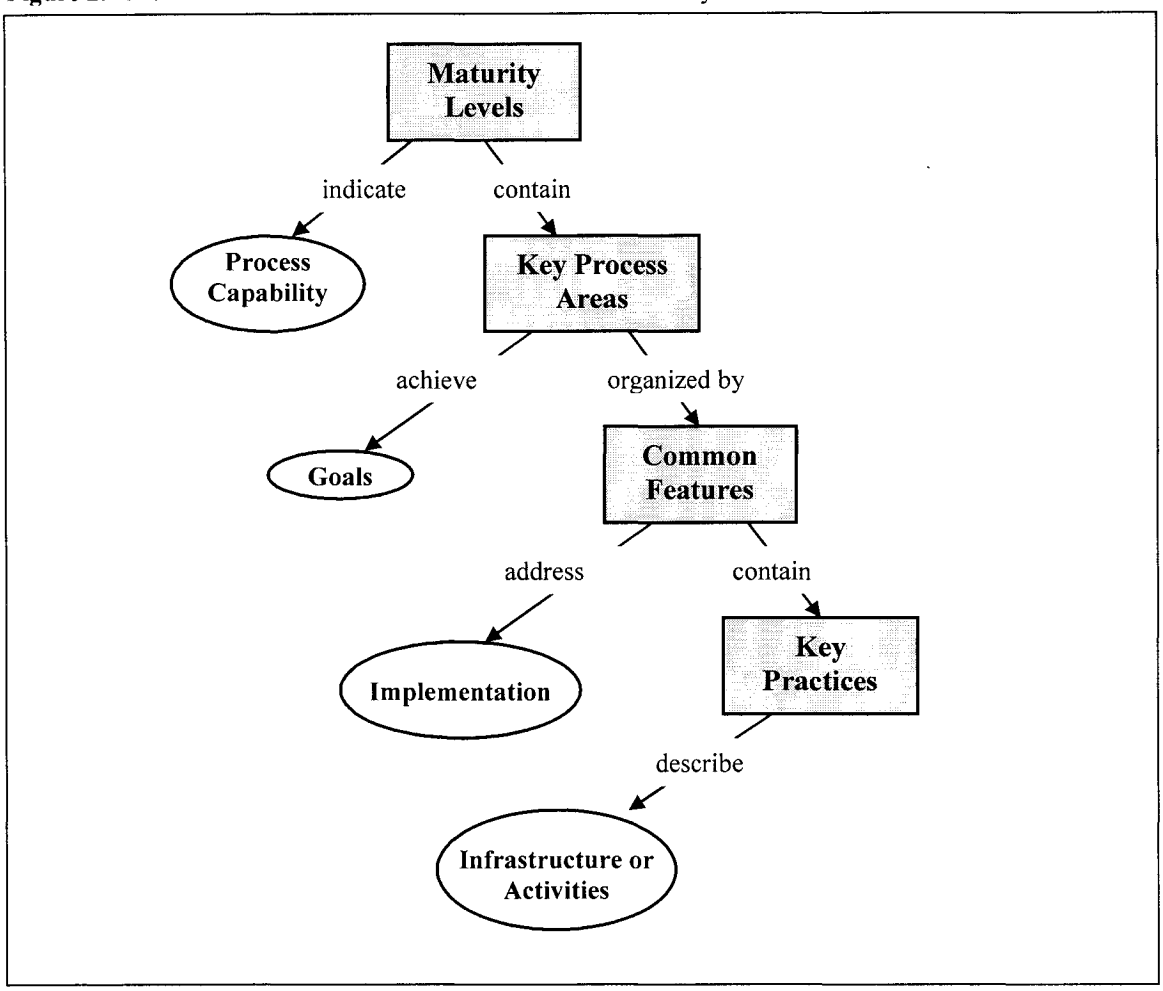
2.5 The Optimized Level

Continuous process improvement is made possible at the Optimized Level. Through the measurement tools established at the Managed Level, the organization can now identify weaknesses in its software processes and proactively modify them. Any defects in the software product are analyzed to determine their causes (Paulk et al., 1995). Software processes can then be evaluated and updated to prevent known types of defects from recurring. All of this knowledge is disseminated and applied in other projects. At the optimized level, organizations continuously strive to improve their existing processes and innovate using new technologies and methodologies (Paulk et al., 1995).

3. The Operational Definition of the Capability Maturity Model for Software

3.1 The Internal Structure of the Software Process Maturity Levels

Figure 2. The Internal Structure of the Software Process Maturity Levels



The Capability Maturity Model for software is composed of constituent parts which make up each maturity level. With the exception of the first level, each maturity level is comprised of different elements ranging from an abstract summary down to an operational definition in the key practices, as shown in Figure 2. Key process areas make up each maturity level and are organized into five sections called common features. The common features provide the means

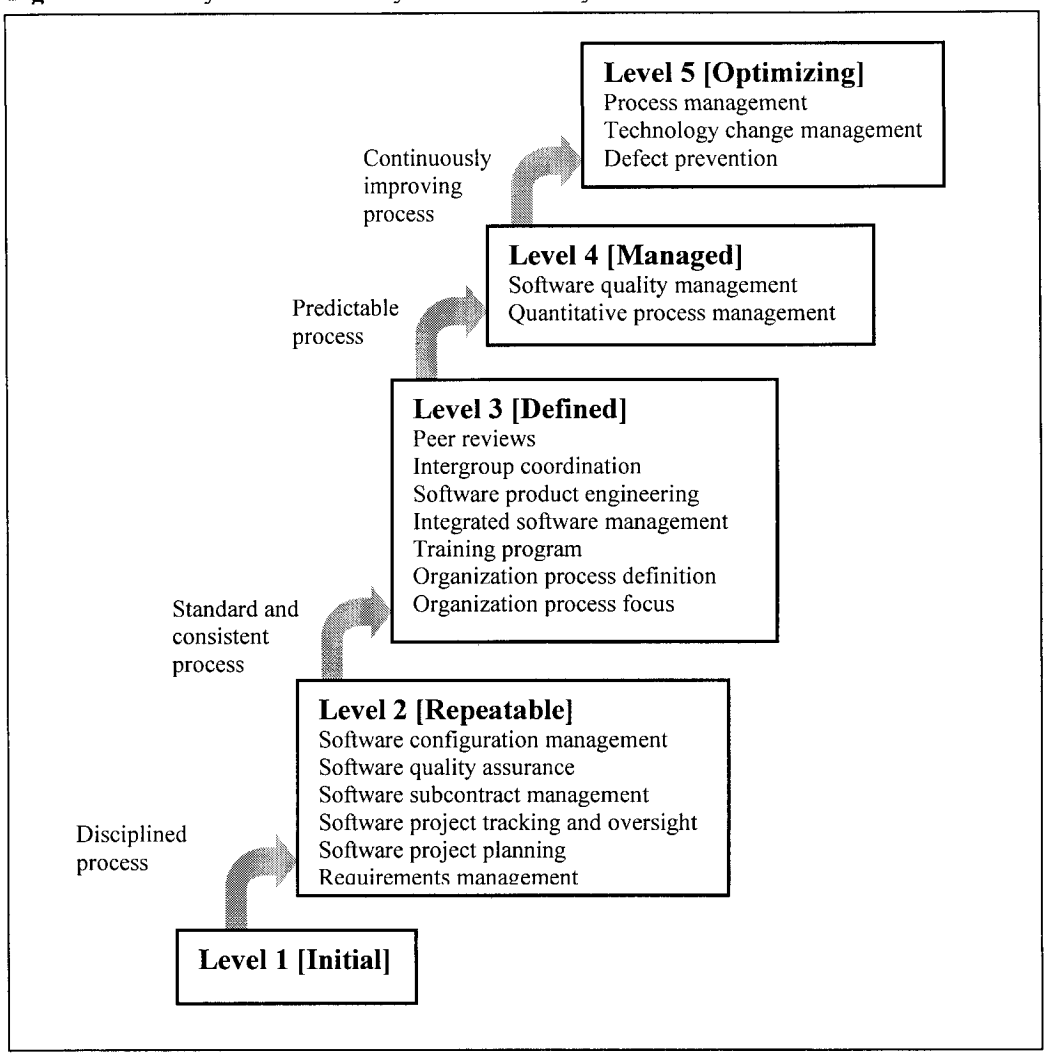
through which specific key practices are specified. The goals of the key process areas can be accomplished once the key practices of the common features are collectively addressed (Paulk et al., 1995).

3.2 The Process Maturity Levels

Process maturity levels are well-defined evolutionary steps that offer guidelines as to how an organization can achieve a mature software process. (Paulk et al., 1995) Each level specifies a unique level of process capability, as illustrated in Figure 1. The main objective for any organization consists of assessing its position in the maturity model and then focus on the elements that will allow it to advance to a higher process maturity level (Humphrey, 1989).

3.3 The Key Process Areas

Figure 3. The Key Process Areas by Process Maturity Level



Each maturity level comprises a set of key process areas that indicate the areas an organization must address in order to improve its software process (Herbsleb et al., 1997). A group of related activities make up each key process area as shown in Figure 3. When preformed collectively, these activities allow for the achievement of goals considered important for enhancing process capability (Paulk et al., 1995). All the goals of a key process area must be met for the firm to satisfy a particular key process area. Once all key process areas at a given process maturity level are met, the organization is considered to have achieved that maturity level.

The CMM presents the key elements of an effective software process and therefore does not describe all the process areas involved in the development and maintenance of software. As the adjective "key" implies, there are process areas and processes that are not included in the Capability Maturity Model since they are not considered key to achieving particular process maturity levels (Paulk et al., 1995).

3.3.1 Level 2 Key Process Areas

Project management activities are the main focus of the key process areas at Level 2 as shown in the following table (Table 1).

Table 1
Level 2 Key Process Areas

Key Process Area	Key Process Area Purpose
Requirements management	Establish a common understanding between the customer and the software project corresponding to the customer's requirements. This common understanding is the basis for planning and managing the software project.
Software project planning	Establish plans for the software engineering phase and for the overall management of the software project.
Software project tracking and oversight	Establish an adequate tracking of the actual progress of the software project so that effective actions can be taken when the project deviates from plans.
Software subcontract management	Select qualified software subcontractors and manage the relationship effectively.
Software quality assurance	Provide management with appropriate visibility into the software process being used throughout the project.
Software configuration management	Maintain the integrity of the software product throughout the project's software lifecycle.

3.3.2 Level 3 Key Process Areas

Project and organizational issues are addressed by the key process areas at Level 3 (see Table 2). Level 3 organizations establish an infrastructure for effective software engineering and management for all projects undertaken (Paulk et al., 1993).

Table 2
Level 3 Key Process Areas

Key Process Area	Key Process Area Purpose
Organization process focus	Establish the various responsibilities related to software process activities that improve the software process capability of the organization.
Organization process definition	Develop and maintain a usable set of software processes that improve the process performance across all software projects conducted by the organization.
Training program	Develop the skills and knowledge of workers in order for them to perform their roles and activities effectively and in an efficient manner.
Integrated software management	Combine and integrate management and software engineering activities into a coherent software process. The software process is tailored for the organization based on its business environment and technical needs of its projects.
Software product engineering	Undertake a software engineering process that integrates all software engineering activities required to produce consistent software products.
Intergroup coordination	Promote the active participation and communication of software engineers in order to better satisfy the customer's needs.
Peer reviews	Establish the means necessary to remove software defects at an early stage of the software development project.

3.3.3 Level 4 Key Process Areas

Quantitative assessments and evaluation of the software process and products are the main concerns of the Level 4 key process areas (see Table 3).

Table 3
Level 4 Key Process Areas

Key Process Area	Key Process Area Purpose
Quantitative process management	Control the performance of software development and maintenance processes through quantitative evaluations of the software project.
Software quality management	Assess the quality of software project products in developing a quantitative understanding of quality.

3.3.4 Level 5 Key Process Areas

The Level 5 key process areas cover the issues that must be addressed to achieve continuous and measurable software process improvement (Paulk et al., 1995) (see Table 4).

Table 4
Level 5 Key Process Areas

Key Process Area	Key Process Area Purpose
Defect prevention	Quickly identify the cause of defects in order to prevent them from recurring.
Technology change management	Recognize beneficial new technologies, tools, methods, and processes and transfer and apply them to the organization.
Process change management	Continually improve the software processes throughout the organization. Process change management allows for the incremental improvements of technology change management and process change management and make them available to the entire organization.

3.3.5 The Internal Consistency of the Key Process Areas

Confidence in the results of software process maturity assessments is crucial when conducting CMM appraisals. As a theoretical model which presents evolutionary plateaus

indicating well-defined levels of process maturity, it is essential to evaluate the consistency and representativeness of each CMM maturity level. CMM process maturity levels are abstract concepts, in other words theoretical constructs that are indirectly measured by using key process areas containing process metrics (Jung and Goldenson, 2002).

The dimensions underlying the maturity constructs in the Capability Maturity Model were evaluated and their internal consistency (reliability) estimated through empirical research (Jung and Goldenson, 2002). The analysis was based on 676 CMM-Based appraisals conducted between January 2000 and April 2002. The results suggest that the Capability Maturity Model is a comprised of three main constructs which represent maturity levels 2, 3 as well as levels 4 and 5 taken together. The internal consistency for each of these constructs was estimated using Cronbach's alpha (Cronbach, 1951) and exceeded the recommended value of 0.7. These consistently high values of Cronbach's alpha show that CMM-Based software process maturity appraisals lead to ratings that are internally consistent with the structure of the CMM. This empirical evidence shows that the groupings of key process areas into well-defined maturity levels can be considered as separate constructs which comprise the CMM (Jung and Goldenson, 2002).

3.4 Goals

The goals provide a summary of the key practices of a key process area. They can be used to determine whether a firm has successfully implemented specific key process areas (Paulk et al., 1993). The goals refer to the scope, boundaries, and objectives of each key process area (Paulk et al., 1993). They can be used to determine whether alternative ways of implementing a key process area in a specific context satisfy the intent of that key process area (Paulk et al., 1995).

3.5 Common Features

The key process areas that make up each maturity level are organized into five sections called common features. The common features consist of attributes that indicate whether the implementation of key process areas is effective, repeatable, and lasting (Paulk et al., 1995). The five common features are presented and described in the following table (see Table 5).

Table 5
Common Features

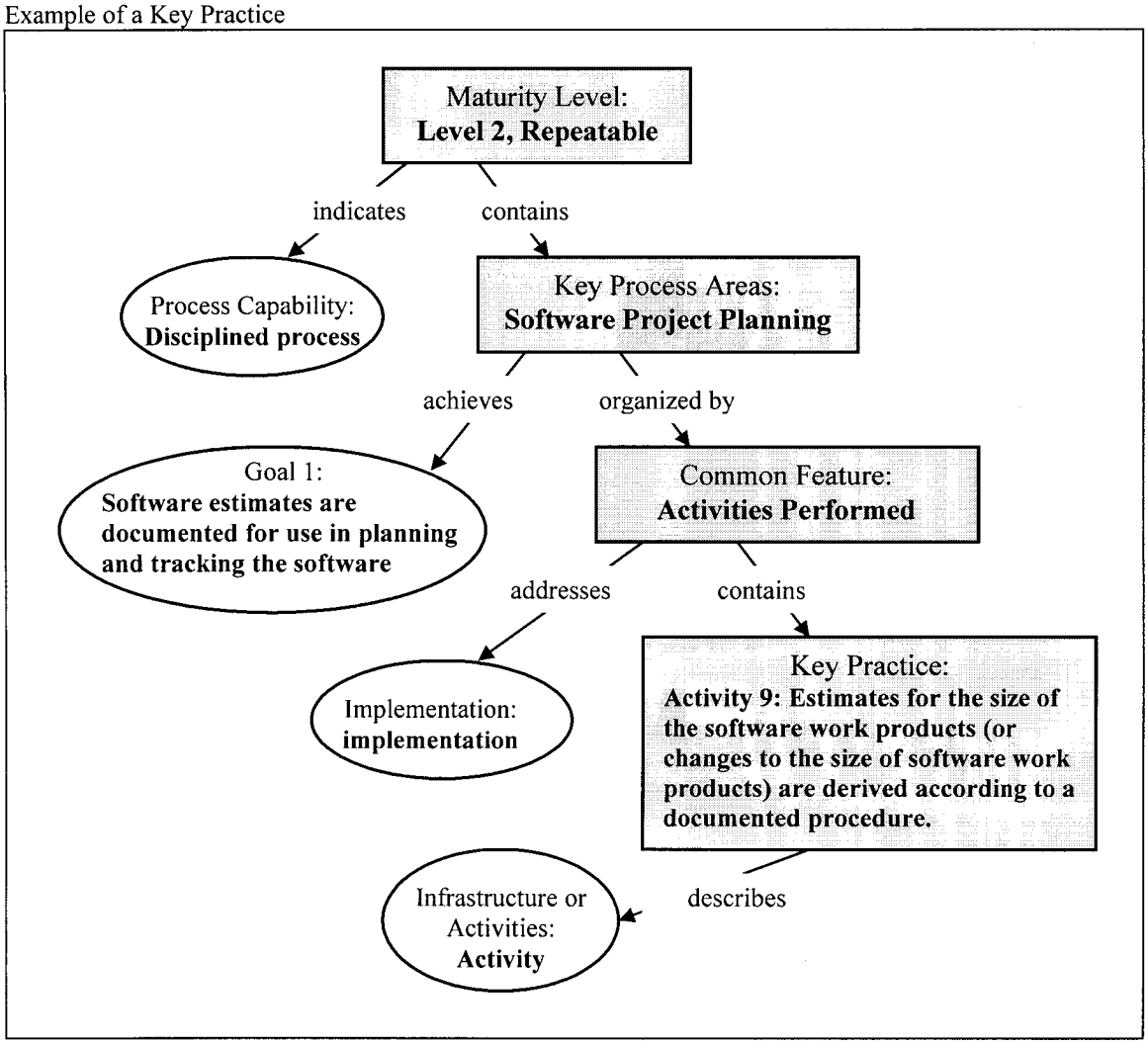
Common Feature	Description
Commitment to perform	Commitment to perform consists of the actions that must be taken by the organization to ensure that the process is established and will last. Organizational policies and upper management support are involved at this stage.
Ability to perform	Ability to perform involves the conditions that must exist in the organization or project in order for software processes to be implemented efficiently. Resources, organizational structures, and training are typically involved at this level.
Activities performed	Activities performed refers to the roles and procedures that must be in place for the successful implementation of a key process area. Establishing plans and procedures, performing the required tasks, tracing the work, and taking corrective actions if necessary are involved at this level.
Measurement and analysis	Measurement and analysis consists of the need to measure the various processes and then conduct an adequate analysis of the results. Measurement and analysis could involve measurements taken to determine whether the activities performed were effective or not.
Verifying implementation	Verifying implementation consists of the steps required to ensure that the activities undertaken throughout the software project are conducted in compliance with the processes that were previously agreed upon and implemented. Verification usually includes software quality assurance activities and reviews and audits performed by management.

A process capability can be established by implementing the practices in the common feature *Activities Performed.* All the other practices are the foundation through which an organization can establish the practices described in the Activities Performed common feature (Paulk et al., 1995).

3.6 Key Practices

Each key process area is described by a number of more explicit and informative components: key practices, subpractices, and examples (Herbsleb et al., 1997). The key practices are the activities whose contribution is the greatest when it comes to a successful implementation of the key process areas (Paulk et al., 1995).

Figure 4
Example of a Key Practice



Each key practice consists of a single sentence and is often followed by a detailed description, which may include further explanations and examples (Paulk et al., 1995). These key

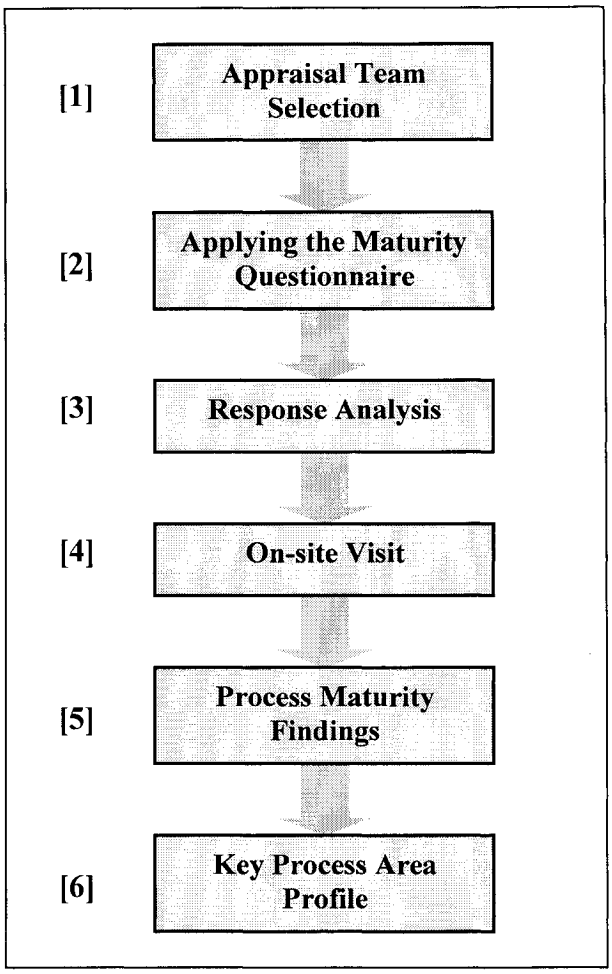
practices consists of fundamental policies, activities, and procedures for a given key process area. Subpractices frequently consist of the detailed descriptions that follow a key practice. While the key practices present "what" is to be done in order to achieve specific goals, they should not be interpreted as describing "how" these same goals are to be met. As they are applied to different contexts and organizational settings, they should always be interpreted in a rational manner to determine whether the goals of the key process areas are successfully, although sometime maybe differently achieved (Paulk et al., 1995).

Figure 4 provides an illustrated example of how all of the CMM's components fall into place, fit together, and allow for the assessment of an organization's software process maturity. For example, an organization that wishes to attain CMM Level 2 must address the level's key process areas. *Software Project Planning* consists of one of the KPAs at this level. In order to tackle this key process area, specific goals must be met. Among these goals is the following: *Software estimates are documented for use in planning and tracking the software project.* As key process areas are organized by common features which contain the key practices of the KPAs, the firm needs to establish the practices contained in the *Activities Performed* common feature since they explicitly describe what must ultimately be implemented to establish a process capability (the other common feature practices form the basis through which a firm can address the key practices described in the *Activities Performed* common feature). Once this is accomplished, that goal for the software project planning KPA is achieved. The firm is one step closer to implementing the *Software Project Planning* key process area and will then address the remainder of the KPAs necessary to achieving Level 2 process maturity (Paulk et al., 1995).

4. Applying the Capability Maturity Model for Software

Software process assessments consist of the identification of improvement priorities within an organization's software process. Organizations are appraised at a maturity level by participating in a CMM-Based Appraisal for Internal Process Improvement (CBA-IPI) conducted by SEI-authorized Lead Appraisers. Assessment teams make use of the CMM as a guide which allows for the identification and prioritization of findings. Subsequently, these findings are used to plan a process improvement strategy for the organization through the guidance provided by the key practices found in the CMM (Paulk et al., 1995). The assessments are performed in an open and collaborative environment and successful outcomes depend on a commitment from both management and the professional staff.

Figure 5
Main Steps in CMM Software Process Maturity Appraisals



The CMM is essentially a common frame of reference for the undertaking of process assessments and software capability evaluations. What follows is a summary of the main steps involved in the process maturity evaluation of an organization through the use of the Capability Maturity Model for Software (see Figure 5).

4.1 Team Selection

The first step involved in software process maturity assessments is to select a team. It is important that the team be trained in the fundamental concepts of the CMM and be knowledgeable when it comes to the specifics of the assessment method (Paulk et al., 1995).

4.2 Applying the Maturity Questionnaire

The second step requires that the representatives from the site being appraised complete the maturity questionnaire which allows for both structured and unstructured interviews as tools for understanding the organization's software processes (Paulk et al., 1995).

4.3 Response Analysis

Upon completing the second step, the appraisal team performs a response analysis during which the answers to the questions on the maturity questionnaire are tallied and areas that warrant further investigation are identified and correspond to the CMM key process areas (Paulk et al., 1995).

4.4 On-site Visit

The site under appraisal can now be visited by the evaluation team. Using the results of the response analysis, the assessment team proceeds with interviews and reviews the organization's documentation in order to gain a more detailed understanding of the firm's software process. Throughout this phase, the CMM provides guidance to the team members in questioning, reviewing, and synthesizing the data and information gathered from the various

interviews and documents. When determining whether the site's implementation of the key process areas satisfy the relevant key process area goals, the team applies professional judgement. In the event that there are discrepancies between the CMM's key practices and the site's practices, the assessment team includes its rationale for judging those key practices in the assessment documentation (Paulk et al., 1995).

4.5 Findings

Following the on-site evaluation, the appraisal team identifies the strengths and weaknesses of the site's software development processes. These findings become the basis for process improvement recommendations (Paulk et al., 1995).

4.6 KPA Profile

A key process area profile is then put together and clearly identifies where the organization has, and has not, satisfied the goals of the key process areas . These findings and key process area profile can then be presented to the appraised organization (Paulk et al., 1995).

5. Organizational Impacts of CMM Software Process Maturity

As the adoption of the Capability Maturity Model for Software continues to grow, increasing efforts are undertaken to better understand the organizational impacts of software process maturity (Herbsleb et al., 1997).

5.1 Case Studies: Understanding the CMM in Specific Organizational Settings

Initial assessments of the impacts of process improvement efforts mostly focused on the assessment of whether CMM-based software process improvement ultimately impacts various aspects of the organization. Early research into this area consisted of numerous case studies through which specific organizations were closely observed in order to better grasp the various effects of process improvement initiatives. Table 6 provides a summary of these studies.

Table 6
Summary of CMM-Based SPI Case Studies

Study	Organization	CMM-Based SPI Key Findings
Humphrey et al., 1991	Software Engineering Division, Hughes Aircraft	Improved quality of work life, fewer overtime hours, fewer problems to deal with each day, a more stable work environment, and low software-professional turnover
Dion, 1993	Software Systems Laboratory, Raytheon Corporation	Reduction in software rework and integration costs, decreased software retesting, gains in schedule and budget performance
Wohlwend and Rosenbaum, 1993	Schlumberger	Improved project and product communication between software engineering centers, improved product quality, increased percentage of software projects completed on time, decreased post release software product defects
Benno and Fraile, 1995	Defence Systems and Electronic Group, Texas Instruments	Improvement in software development productivity and in delivered defect density
Butler, 1995	Oklahoma City Air Logistics Center, Directorate of Aircraft Software Division, Tinker US Air Force Base	Improved return on investment, defect rates, maintenance costs, and productivity
Diaz and Sligo, 1997	Government Electronics Division, Motorola	Reduced defect density, improved cycle time and productivity with each level except level 3
Fitzgerald and O'Kane, 1999	Cellular Infrastructure Group, Motorola (Ireland)	Identification of critical success factors at each CMM level
Herbsleb et al., 1994	Bull HN Information Systems Inc., US subsidiary of Groupe BULL	Improved coding time and testing time and yearly amount of defects reported by customers
Myers, 1994	Hewlett-Packard	Reduced number of delivered defects and substantial cost savings

Software process improvement at the Software Engineering Division (SED) of Hughes Aircraft in Fullerton, California was one of the pioneering case studies which presented and described an entire CMM assessment and process maturity improvement effort in an organizational setting (Humphrey et al., 1991). After conducting a software process assessment of the SED, the Software Engineering Institute's assessors found the division to be at level 2. This first assessment identified the various strengths and weaknesses of the SED while providing recommendations for software process improvement. After implementing the proposed suggestions, the SED was reassessed three years later and found to be a strong level 3 organization (Humphrey et al., 1991). Throughout the entire assessment process as well as the improvement initiatives and efforts, observations were made by the assessment teams and the personnel involved in the division's appraisal. This process resulted in what the author's termed "lessons learned" (Humphrey et al., 1991). Among other things, the authors suggest that the benefits from the software process appraisals and improvement efforts are worth the effort and expense. The quality of work life at Hughes improved and the firm's image benefited from improved performance. Hughes' SED had fewer overtime hours, fewer problems to deal with each day, a more stable work environment, and low software-professional turnover (Humphrey et al., 1991). Moreover, this case study provided individuals unfamiliar with the Capability Maturity Model with a detailed description of just how software process maturity is addressed by the CMM, how it is evaluated, and how it is improved upon.

A similar undertaking was conducted at Raytheon Corporation's Software Systems Laboratory (SSL). Raytheon is a diversified, international, technology-based company in the US (Dion, 1993). After having been assessed at CMM Level 1, the Raytheon corporation decided that a process improvement program would be undertaken. The improvement program resulted in an evolution from Level 1 (Initial) to Level 3 (Defined). Various observations following the process improvement efforts were made. Notably, software rework and integration costs were

reduced. Retesting also decreased about half the amount of its original value. Raytheon's SSL's evolution to CMM Level 3 also resulted in gains in schedule and budget performance. After process improvement efforts, most projects finished on or ahead of schedule and below budget (Dion, 1993). Moreover, less tangible but equally important results were noted. Software engineers were spending fewer late nights and weekends at the office and employee turnover was reduced (Dion, 1993).

Schlumberger, a multinational organization, also underwent a software process improvement program (Wohlwend and Rosenbaum, 1993). After several years of improvement activities, positive benefits were observed. Managers noted better project and product communication between its software engineering centers. Customers also mentioned that the quality of the products had improved. Other positive, more quantifiable results from higher software process maturity were also reported. Over a three year period during which software process improvement efforts were conducted, it was found that the percentage of software projects completed on time increased from 51% in 1990 to 94% in 1992 (Wohlwend and Rosenbaum, 1993). Furthermore, post-release software product defects decreased from 25% of total product defects in 1989 to 10% of total product defects by 1991 (Wohlwend and Rosenbaum, 1993).

The Defence Systems and Electronic Group (DSEG) is a division of Texas Instruments Inc. Their primary products are electro-optic (EO) systems, missile guidance and control systems, and airborne radar systems (Benno and Fraile, 1995). A critical component of all these systems is real-time software. An initial assessment of software process maturity revealed that DSEG was a Level 1 organization. Following sustained software process improvement initiatives, DSEG was again assessed by a licensed SEI appraiser and was determined to have achieved Level 3 process maturity (Benno and Fraile, 1995). Following process improvement

programs, the organization had seen a two-fold improvement in software development productivity and a 6.5 times improvement in delivered defect density (Benno and Fraile, 1995).

Other similar case studies are found in the literature. In an additional effort to asses the impact of software process improvement on the organization, the economic benefits of software process improvements were studied at the Oklahoma City Air Logistics Center, Directorate of Aircraft Software Division located at Tinker US Air Force Base (Butler, 1995). Economic benefits were grouped into four categories: return on investment, defect rates, maintenance costs, and productivity (Butler, 1995). It was determined that process improvement efforts which led to a Level 3 assessment had a positive impact on all of these categories (Butler, 1995). Similar results were also reported at Motorola's Government Electronics Division (GED) (Diaz and Sligo, 1997). Notably, it was reported that each process maturity level reduced defect density (defects per million earned assembly equivalent lines of code) by a factor of 2. However, it was noted that cycle time (the amount of time for a baseline project to develop divided by the cycle time for post process improvement project) and, to a lesser extent, productivity (the amount of work produced divided by the time to produce that work) improve with each maturity level except level 3, where they both decrease (Diaz and Sligo, 1997). The authors suggested that achieving level 3 involves significant new process implementation which can negatively affect these two metrics until the organization absorbs and adopts the processes. Motorola's Cellular Infrastructure Group at Cork, Ireland was also studied as it progressed from CMM level 1 in 1993 to level 4 in 1997 (Fitzgerald and O'Kane, 1999). Efforts were made to identify critical success factors associated with the site's progression towards level 4 (Fitzgerald and O'Kane, 1999).

Likewise, Bull HN Information Systems Inc., the US subsidiary of Groupe BULL, the fourth-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 reduced from 1 per thousand lines of code to 0.1 per thousand lines of code. Cost savings over 100 million dollars were also achieved through software process improvement (Myers, 1994).

5.2 Empirical Research: Studying the CMM and Software Process Improvement through Empirical Investigations

In addition to numerous case studies on the organizational impacts of software process improvement, several empirical efforts, aimed at further understanding what has been observed in particular firms, have been undertaken.

One of these studies set out to determine the correlation, if any, between software process maturity and software project success (Lawlis et al., 1995). In the context of this study, software project success was defined as cost and schedule performance. Actual performance data for these two metrics were compared to identical baseline measures in order to obtain performance indices. Software process maturity was determined by rating various contractors according to Software Engineering Institute CMM protocols. Data were collected from 11 US Department of Defence contractors who had been rated on 31 software projects these organizations were developing while their ratings were in effect. Data analysis focused on statistically correlating the cost and schedule performance indices with the respective CMM ratings. Results showed improved cost and schedule performance with increasing software process maturity. This correlation was more evident in cost performance than in schedule performance (Lawlis et al., 1995).

In studying the effect of software process improvement on organizational performance, other researchers proceeded with a survey of firms that had undergone CMM-based software process improvement in order to obtain more representative results (Herbsleb and Goldenson, 1996). One of the main goals of the survey was to determine whether the performance reported by more mature organizations was in fact superior to the performance of less mature firms. Performance was assessed using several measures. Performance indicators included such metrics as the firms' ability to meet schedule and budget commitments, staff morale, customer satisfaction, and staff productivity. A random sample of 155 firms was drawn from the SEI's appraisal reports database. A response rate of 83% allowed for the cross-tabulation of maturity level and organizational performance metrics. Results showed that respondents who reported better performance, tended to be from higher maturity firms. Statistically significant relationships were found between higher maturity levels and ability to meet schedule, ability to meet budget, and higher staff morale. These findings provide further support to previous research which showed that high maturity organizations are likely to have less difficulty adhering to cost and schedule targets (Lawlis et al., 1995). However, relationships between maturity level and product quality, customer satisfaction, and staff productivity were not quite statistically significant, but were all in the same direction (increased with higher process maturity levels) (Herbsleb and Goldenson, 1996). Also, no attempt was made at assessing process performance with specific measures of software project performance.

Other performance aspects have also been 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 but also increased development efforts. However, 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).

B. Software Project Performance

Performance measurement is a central issue in attempting to evaluate software development projects. As organizations continue to invest substantial amounts of money 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. Placed in the larger context of information systems literature, software project performance can be considered as a more specific research area in the broader field of information systems project performance.

Performance literature in information systems 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 (Aladwani, 2002). The social view of software project performance refers to research which largely focuses on issues relevant to the attributes and behaviours of project members. Software development projects are placed in their specific social context in which human behaviours and the overall organizational environment in which software projects take place are considered. In this view, technology factors as determinants of system development project outcomes are largely submerged by social variables (Aladwani, 2002). On the other hand, the technical view of software project performance consists of research which focuses on issues relevant to the characteristics of the project in itself. Contextual variables such as human behaviour and project management are usually overlooked in these types of studies (Aladwani, 2002).

1. Factors that Affect Software Project Performance

1.1 The Social View of Software Project Performance

The social view of software project performance is largely characterized by close attention paid to the human behavioural aspect of software development projects and its impact on project performance.

Table 7
Studies Falling into the Social View of Software Project Performance

Study	Human Behavioral Variable(s)	Performance Variable(s)	Key Findings
Henderson and Lee, 1992	Managerial and team member control	Adherence to schedules and budgets, and amount of work produced	High-performing teams exhibit high process control by managers and high outcome control by team members; increases in total control behavior increases team performance
Abdel-Hamid, 1992	Managerial succession and employee turnover	Project costs and duration	Managerial succession and turnover can ultimately lead to discernable variations in project performance in terms of both costs and duration
Robey et al., 1993	Participation, influence, conflict, and conflict resolution among team members	Adherence to budgets, schedules, and quality of work	There is a strong positive relationship between conflict resolution and project success and a moderate relationship between participation and project success
Kraut and Streeter, 1995	Team coordination practices such as system status and design review meetings, more general development group meetings, and code inspections	Project members informed, coordination success, client satisfaction, managers' evaluation, software productivity, software quality	These aspects affect the success of projects on several dimensions. For instance, formal procedures such as status and design review meetings were found to be relatively good predictors of software quality.
Guinan et al., 1998	Team skill, managerial involvement, and variance in team experience	Stakeholder-reported and team-reported team performance	Team skill, managerial involvement, and little variance in team experience enable more effective team processes than do software development tools and methods
Saleem, 1996	The user's perceived expertise in IS development	Extent to which users resist and utilize the system	Users who perceive themselves as functional experts in IS relative to others are unlikely to accept a system unless they exercised a substantial influence on its design. On the other hand, users who do not perceive themselves as functional IS experts are likely to accept and use a system regardless of the extent of their influence on its design.

The social view of software project performance can take many forms. In studying IS design teams for instance, Henderson and Lee (1992) addressed the concept of managerial and team-member control and their effects on the design teams' performance in terms of variables such as adherence to schedules and budgets, and amount of work produced. Results from their study of 41 actual IS design teams indicate that high-performing teams exhibit high levels of

control both by managers and team-members. Control behaviours are found to be positively correlated with team performance.

Managerial succession and employee turnover are additional behavioural factors that can influence IS project performance outcomes (Abdel-Hamid, 1992). The results of a simulation-based laboratory study to investigate this issue revealed that managerial succession and turnover can ultimately lead to discernable variations in project performance in terms of both costs and duration (Abdel-Hamid, 1992).

As system development projects are rarely an individual endeavour, research into the interaction among development team members has also garnered attention. The relationships between participation, influence, conflict, and conflict resolution among team members have been studied and assessed as to their influence on project success (Robey et al., 1993). Results showed 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).

Coordination of team members has also emerged as an important variable when it comes to studying the dynamics of groups. In software development teams, several coordination practices are used such as system status and design review meetings, more general development group meetings, and code inspections and affect the success of projects on several dimensions (Kraut and Streeter, 1995). For instance, formal procedures such as status and design review meetings were found to be relatively good predictors of software quality (Kraut and Streeter, 1995).

The study of project performance through users has also included such variables as team skill, managerial involvement, and variance in team experience. The two former aspects as well as little variance in team experience were found to enable more effective team processes than do software development tools and methods (Guinan et al., 1998).

Software project performance can also be viewed as the degree to which a system is successful. The level of success can in turn be considered as the extent to which users resist and utilize the system (Saleem, 1996). User participation in information systems development has been largely researched. However, only recently has the concept of the user's perceived expertise in IS development been studied. In fact, research results suggest that users who perceive themselves as functional experts in IS relative to others are unlikely to accept a system unless they exercised a substantial influence on its design (Saleem, 1996). On the other hand, users who do not perceive themselves as functional IS experts are likely to accept and use a system regardless of the extent of their influence on its design (Saleem, 1996). These findings provide useful insights when it comes to the undertaking of high performance software development projects in terms of ultimate system resistance and utilization by the user community.

1.2 The Technical View of Software Project Performance

The technical view of software project performance is more concerned with issues relevant to the characteristics of the software development project in itself. Contextual variables such as human behaviour and project management are usually overshadowed by technology, task, process, and project characteristics (Aladwani, 2002).

Table 8
Studies Falling into the Technical View of Software Project Performance

Study	Technical Variables	Performance Variables	Key Findings
Saarinen, 1990	Development time, level of telecommunications, adequacy of tools	Time, budgetary, and user requirements	Methods and tools must be adequate in every phase of the development life cycle to ensure success.
Deephouse, 1995-96	Software process stability, design reviews, software prototyping	Software quality, and meeting project targets	Project planning was shown to be a significant positive predictor of both meeting project targets and software product quality while cross-functional teams had a significant positive effect for software product quality
Ravichandran, 1999	Implementation of information repository, code/design reuse policy, and reuse-based rewards	Adherence to cost budgets, adherence to project schedules, systems delivery lead time, and productivity of IS personnel	Various aspects of both the administrative and technical dimensions were found to affect software project performance; notably the implementation of reuse-oriented reward schemes (administrative dimension) is a significant predictor of all four aspects of software delivery performance while the implementation of a repository was significant in predicting the cost, delivery, and productivity dimensions of software delivery performance
Rai and Al-Hindi, 2000	Technical process modeling and task uncertainty	Product and process quality	Development process modeling is positively related to both process and product quality, while task uncertainty is negatively related to them. Development process modeling reduces the negative impact of task uncertainty on quality-oriented development outcomes.

Saarinen (1990) examined the variables that contribute most to the success of IS projects. Variables included development time, level of telecommunications, and adequacy of tools. In comparing a number of software development projects, the author showed the need to have adequate methods and tools in every phase of the system development life cycle to ensure success.

As software project performance has many dimensions, Deephouse et al. (1995-96), decided to focus on two specific performance aspects: software quality, and meeting project targets. Software quality entails the extent to which the software system meets the actual needs of the end users and meeting targets refers to the notion that in order to be successful a software

project should be on time and on budget. In addressing software project performance, the authors conducted a study to assess the effectiveness of seven processes in common use during software development projects: project planning, software process stability, cross-functional teams, process training, prototyping, and communication with users. Results provided evidence for the importance of two particular processes. Project planning was shown to be a significant positive predictor of both meeting project targets and software product quality while cross-functional teams had a significant positive effect for software product quality (Deephouse et al., 1995-96).

Software project performance has also been referred to as system delivery performance measured through adherence to cost budgets, adherence to project schedules, systems delivery lead time, and productivity of IS personnel (Ravichandran, 1999). Along these performance dimensions, it was found that specific aspects of software reusability, conceptualized as a software process innovation from administrative and technological perspectives, affect software project performance. Software reusability is a potential strategy to tackle recurring systems development problems such as high development costs, long systems delivery lead times, and low levels of programmer productivity (Ravichandran, 1999). The administrative dimension of reusability includes changes to structures and policies within the IS unit that are specifically adopted to facilitate reuse implementation. The technological perspective relates to the implementation of an information repository containing parameterised design and code which facilitate software reusability when developing new systems. Various aspects of both the administrative and technical dimensions were found to affect software project performance; notably the implementation of reuse-oriented reward schemes (administrative dimension) is a significant predictor of all four aspects of software delivery performance while the implementation of a repository was significant in predicting the cost, delivery, and productivity dimensions of software delivery performance (Ravichandran, 1999).

Moreover, Rai and Al-Hindi (2000) looked at the impact of development process modeling on outcomes in software development projects. They showed that development process modeling is positively related to both process and product quality, while task uncertainty is negatively related to them. Development process modeling reduces the negative impact of task uncertainty on quality-oriented development outcomes (Rai and Al-Hindi, 2000).

2. Measuring Software Project Performance as a Two-Dimensional Construct

However, an alternate view of performance has garnered consistent support throughout the years: the assessment of software project performance as a two-dimensional construct comprised of both a process and a product dimension (Riddle, 1984; Agresti, 1986; Cooprider and Henderson, 1990-91; Nidumolu, 1995; Ravichandran and Rai, 2000; Rai and Al-Hindi, 2000; Barki et al., 2001, Jiang et al., 2004). 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 processes, 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).

One of the earlier studies adopting the process/product performance perspective of software development performance proposed assessing the effects of IS project processes and supporting technologies on project performance (Cooprider and Henderson, 1990-91). More specifically, the focus of the study was on IS prototyping project performance and the primary

determinant of performance impact: the fit between the prototyping processes and the support technology used (Cooprider and Henderson, 1990-91). The authors' contributions consisted of presenting innovative perspectives on the measurement of software project performance and providing early support for the study of performance as a two-dimensional process/product construct (Cooprider and Henderson, 1990-91).

Variations in product and process performance have also been attributed to other factors such as coordination mechanisms and risk drivers (Nidumolu, 1995). Two types of coordination mechanisms are vertical and horizontal coordination. The former consists of the extent to which coordination between the IS staff and users is conducted through authorized entities such as steering committees and project managers. The latter involves coordination through mutual adjustments and communications between the IS staff and users (Nidumolu, 1995). Risk drivers include residual performance risk which is the difficulty in estimating performance-related outcomes during the later stages of projects (Nidumolu, 1995). Research has provided evidence of additional predictors of software project performance in project uncertainty, residual performance risk, horizontal coordination, and the joint effects of vertical and horizontal coordination (Nidumolu, 1995). Moreover, a more specific type of coordination in software development projects, horizontal coordination, has been shown to predict a particular project performance outcome: product flexibility (Nidumolu, 1996). Horizontal coordination involves the mutual adjustments and communication between system users and IS staff while product flexibility refers to the extent to which the software product ultimately delivered to the customer is able to support distinctively new products or functions in response to changing business needs (Nidumolu, 1996). While no evidence was found when it came to vertical coordination (coordination through superiors) of project team members, horizontal coordination explained the bulk of the variance of product flexibility (Nidumolu, 1996).

Additional risk-based research has also contributed to further advancing our knowledge of the relationship between risk and performance in software development projects (Barki et al., 2001). Specifically, results suggest that in order to increase software project performance (process and product performance) a project's risk management profile needs to vary according to the project's exposure to risk (Barki et al., 2001). In other words, projects exposed to high degrees of risk require a different risk management profile than do low risk exposure projects. Such a risk management profile include high information processing capability approaches in their management as well as high levels of formal planning (Barki et al., 2001).

Software process maturity must also be considered when considering antecedents of process and product performance in software development projects. Research results indicate that particular CMM software process levels are positively related to project performance (Jiang et al., 2004). Project performance was found to be influenced by certain CMM level 3 recommended activities (process engineering and organizational support activities). However, certain CMM level 4 activities (product and process quality activities) were marginally significant in predicting performance (Jiang et al., 2004). Level 2 activities were not found to have a significant effect on software project performance variations (Jiang et al., 2004).

This two-dimensional view was also adopted in the study of software project quality and resulted in the definition of project outcome quality in terms of product quality and process efficiency (Ravichandran and Rai, 2000). Product quality refers to users' perceptions when it comes to such things as the system's functional requirements and their satisfaction with its overall quality. Process efficiency includes various elements related to the development project in itself such as schedule and budget cost overruns, time spent on fixing system bugs, and the level of backlog of development work (Ravichandran and Rai, 2000). In studying these variables, the authors found that software quality is best attained when top management creates a management

infrastructure that promotes improvements in process design and encourages stakeholders to evolve the design of the development processes (Ravichandran and Rai, 2000). Variations of this two-dimensional view were also proposed in the study of software project quality and resulted in the definition of project outcome quality in terms of product and process quality (Rai and Al-Hindi, 2000). Product quality included such variables as the reliability of the application developed as well as its maintainability. Process quality referred to budget and schedule aspects of the development project and quality of efforts among other things (Rai and Al-Hindi, 2000). The results suggest that software development process modeling is positively related to both product and process quality, while task uncertainty is negatively related to them (Rai and Al-Hindi, 2000).

C. Software Development Risk

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). Among the proposed methods, the concept of software project risk management has emerged as a popular topic. According to the risk management method, various actions can be taken to reduce the chance of software project failure by identifying and analyzing threats to the success of the project (Schmidt et al., 2001). This section will present how the concept of risk in information systems literature has been addressed throughout the years. The section is organized as follows. As the concept of risk in information systems can be organized into two distinct streams of research- the rational decision theory perspective of risk and the behavioural perspective of risk (March and Shapira, 1987; Lyytinen et al., 1998) - the following literature review will be divided along these two dimensions. Subsequent to an introduction to these

distinct streams of research, risk research literature will be covered first along the rational decision theory view followed be the behavioural view of risk.

1. The Concept of Risk in Information Systems Research

Risk theory in information systems can be divided into two separate streams of research: the decision theory perspective (decision theoretic perspective) of risk and the behavioural perspective of risk (March and Shapira, 1987; Lyytinen et al., 1998).

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). In this context, a risky alternative is an option for which the variance is large, and risk, along with the expected value of the alternative, are used as attributes in evaluating different gambles (Arrow, 1965). Rational choice theory postulates that managers dealing with risk first calculate alternatives and then choose one option among the available risk-return combinations yielding the highest outcome. They thus behave in a rational (risk-aversive) manner (Yates, 1992).

On the other hand, the behavioural perspective of risk (March and Shapira, 1987) more accurately defines the assumptions that underline most risk management approaches. Risk research that falls into this perspective focuses more specifically on ambiguous losses, relies on multidimensional and qualitative models that make the management task more simple and feasible, and seeks to avoid or master risks through sequential pruning exercises (Lyytinen et al., 1998).

2. The Decision Theory Perspective of Risk

The concept of risk is ever-present when it comes to human endeavours. Consequently, people from a variety of domains have tackled the notion of risk and proposed a wide range of definitions to better grasp its meaning. However, despite numerous perspectives of risk, similarities underlie the many descriptions of the concept. Many definitions of risk include two specific dimensions (Barki et al., 1993): (1) a probability associated with an undesirable event, and (2) the consequences of the undesirable event. For example, an early proponent of software risk management, Boehm (1989) defined risk or risk exposure (RE) in software development as

> *RE = Prob(UO) * Loss(UO) where Prob(UO) is the probability of an unsatisfactory outcome, and Loss(UO) is the loss to the parties affected if the outcome is unsatisfactory. (p. 4)*

According to this definition, 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). Several risk analysis methodologies, which consider loss exposure (risk exposure) as a function of the degree of vulnerability of an asset multiplied by the probability of the threat becoming a reality, have been developed such as Livermore Risk Analysis Methodology (Guarro, 1987), the Stochastic Dominance method (Post and Diltz, 1986), the Probabilistic Risk Assessment (Linnerooth-Bayer and Wahlstrom, 1991), and RiskMethod and the corresponding RiskTool (Känsälä, 1997).

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). Also, it is important to delve upon the relative nature of risk as it has been proposed that absolute risk does not exist. Instead, it could in fact be considered a subjective concept that varies according to the perceptions of the different individuals assessing it (Kaplan and Garrick, 1981; Keil et al., 2000). Risk would thus involve uncertainties and loss rather than probabilities and loss (Barki et al., 1993). For instance, in the context of a software development project, uncertainty of loss exists when individuals possess limited information on the various factors which will affect the project throughout its multiple phases. In fact, uncertainty of loss will have existed even if the development team subsequently discovers that there was no probability of that particular loss happening. In this case, given that there was no possibility of a specific loss ever happening, an estimation of the probability of that particular loss would have been completely erroneous while the uncertainty regarding the loss was very much present and identifiable.

Furthermore, the decision theoretic view is not consistent with empirical findings of how managers deal with risks (March and Shapira, 1987). In fact, human decision makers are quite insensitive to probability concepts when addressing uncertainties. Individuals do not trust, do not understand, or simply do not utilize probability estimates (March and Shapira, 1987). Instead, managers exhibit loss-aversive behaviours as opposed to the more rational ones proposed by rational choice theory (Lyytinen et al., 1998). More crude characterizations are used to exclude certain possibilities when it comes to decision making and thus render the managerial process a sequential pruning exercise aimed at identifying and mastering risks instead of making it an overall optimization decision (Lytinnen et al., 1998).

3. The Behavioural Perspective of Risk

As a result, alternative methods of assessing risk have been developed and focus on the factors that influence the occurrence of undesirable events instead of assessing their probabilities of occurring. Acknowledging the difficulties in making accurate estimates of probabilities and losses in software development, Boehm (1991) subsequently proposed the use of approximate methods. He recommended a top-level risk-identification checklist comprised of ten primary sources of risk based on a survey of experienced project managers. The risk items include personnel shortfalls, unrealistic schedules and budgets, developing the wrong software functions, developing the wrong user interface, gold-plating, continuing stream of requirement changes, shortfalls in externally furnished components, shortfalls in externally performed tasks, real-time performance shortfalls, straining computer-science capabilities (Boehm, 1991). Additional attempts to develop a comprehensive list of risk items were conducted by Schmidt, Lyytinen, Keil, and Cule (2001). Using a Delphi survey to produce a rank-order list of risk factors, the authors elicited and organized opinions of an international panel of experts through iterative, controlled feedback. Risk factors that stood out as receiving a high rating included lack of top management support, failure to gain user commitment, and a misunderstanding of the requirements (Schmidt et al., 2001). It is important to note that despite the agreement with regard to several risk factors, there were differences in opinion on the relative importance of some factors (Schmidt et al., 2001). This finding stresses the need for caution in considering risk factors as perceptions can vary across countries and cultures (Peterson and Kim, 2003).

In addition to identifying risk factors that may impede project progress, Ropponen and Lyytinen (1997) studied the impact of risk management practices such as methods, resources, and their period of use, on risk management performance or the success in managing widely recognized risks. Instead of assessing the effect of the presence of risk factors on performance

outcomes, the authors focused more specifically on answering the following question: Do specific risk management practices actually lead to the successful management of widely recognized risks? Results supported the general claim that the use of risk management methods does in fact improve system development performance. However, very little support was found in determining whether specific risk management methods are instrumental in addressing specific software risks (Ropponen and Lyytinen, 1997). Instead, it was found that overall, risks are better managed when confronted by experienced managers coupled with such things as investing in and obtaining experience in risk management considerations (Ropponen and Lyytinen, 1997).

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. Both project uncertainty and residual performance risk have a direct negative effect on project performance. Moreover, 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).

Finally, Barki, Rivard, and Talbot (1993) directly addressed the difficulty of coupling the concept of risk with outcome probabilities and thus devised an instrument comprised of uncertainty and risk variables derived from previous research in risk and uncertainty literature.

They therefore define risk as follows:

*Software development risk = (project uncertainty) * (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).

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 the magnitude of potential loss due to project failure was assessed through 11 variables (Barki et al., 1993). Later studies found that increased levels of fit between these five dimensions of risk exposure of a software project and its management profile have a positive effect on software project performance (Barki et al., 2001) and that software development risk negatively impacts software project effectiveness, thus providing further support for the behavioural perspective of risk (Jiang and Klein, 2000).

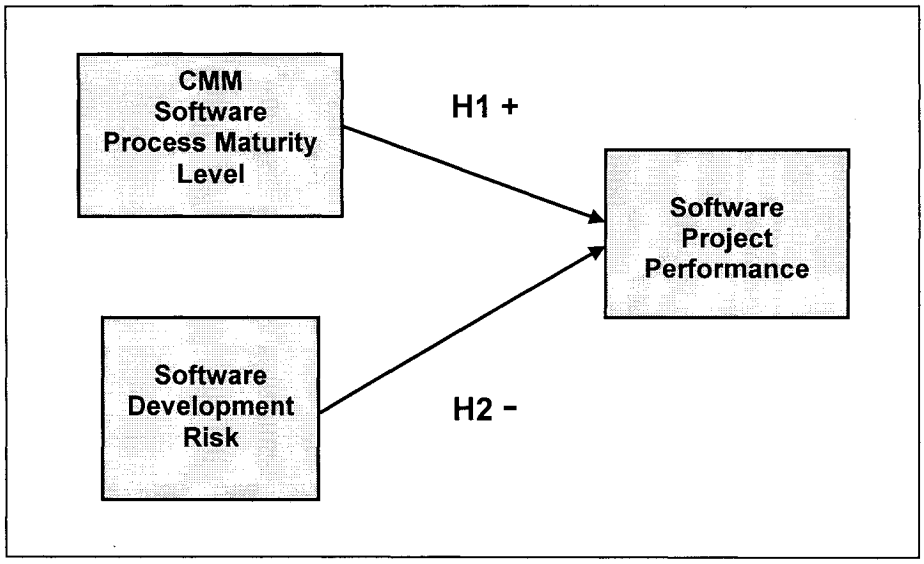
III. RESEARCH MODEL AND HYPOTHESES

3.1 The Conceptual Framework

The conceptual model shown in Figure 6 was developed to answer the key research questions that motivate this study: *What is the impact of CMM process maturity levels on the performance of software development projects? What is the relationship between software development risk and software project performance?*

To answer these questions, a set of metrics pertaining to each of the three constructs used in this research was utilized. Data was then collected to validate these metrics and examine their relationships and relationship strengths to one another. Grounded in prior research in the areas of software process improvement, software project performance, and risk in information systems, a series of hypotheses were developed to test the presence (or absence) of effects in accordance with the relationships depicted in the conceptual framework (see Figure 6). In others words, the proposed research model will for the first time provide empirical evidence as to the relationships between CMM software process maturity levels, software project performance, and software development risk.

Figure 6
The Conceptual Framework



3.2 Hypothesis Development

The research model is tested across two hypotheses that allow for the exploration of predicted relationships between the constructs. As the main objective of this study consists of investigating the correlations between CMM software process maturity levels and the performance of software development projects, while looking at the influence of software development risk on software project performance, prior published findings pertaining to these variables are presented next and ultimately lead to hypothesized relationships. A detailed explanation of the predicted relationships between the constructs follows.

3.2.1 Software Process Maturity and Software Project Performance

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 the CMM model have been proven to increase project performance. The following cases and surveys validate this claim.

CMM-based software process improvement at the Software Engineering Division (SED) of Hughes Aircraft in Fullerton, California improved the quality of work life and the firm's image benefited from improved performance. Hughes' SED had fewer overtime hours, fewer problems to deal with each day, a more stable work environment, and low software-professional turnover (Humphrey et al., 1991).

Raytheon Corporation's Software Systems Laboratory (SSL) undertook a software process improvement effort. The improvement program resulted in an evolution from Level 1 (Initial) to Level 3 (Defined). Various observations following the process improvement efforts were made. Notably, software rework and integration costs were reduced. Retesting also decreased about half the amount of its original value. Raytheon's SSL's evolution to CMM Level 3 also resulted in gains in schedule and budget performance. After process improvement efforts, most projects finished on or ahead of schedule and below budget (Dion, 1993). Moreover, less tangible but equally important results were noted. Software engineers were spending fewer late nights and weekends at the office and employee turnover was reduced (Dion, 1993).

Schlumberger, a multinational organization, also underwent a software process improvement program (Wohlwend and Rosenbaum, 1993). After several years of improvement activities, positive benefits were observed. Managers noted better project and product

communication between its software engineering centers. Customers also mentioned that the quality of the products had improved. Other positive, more quantifiable results from higher software process maturity were also reported. Over a three year period during which software process improvement efforts were conducted, it was found that the percentage of software projects completed on time increased from 51% in 1990 to 94% in 1992 (Wohlwend and Rosenbaum, 1993). Furthermore, post release software product defects decreased from 25% of total product defects in 1989 to 10% of total product defects by 1991 (Wohlwend and Rosenbaum, 1993).

The Defence Systems and Electronic Group (DSEG), a division of Texas Instruments, following sustained software process improvement initiatives, saw a two-fold improvement in software development productivity and a substantial improvement in delivered defect density (Benno and Fraile, 1995).

Oklahoma City Air Logistics Center, Directorate of Aircraft Software Division located at Tinker US Air Force Base (Butler, 1995) also benefited form several economic benefits which were grouped into four categories: return on investment, defect rates, maintenance costs, and productivity (Butler, 1995). It was determined that process improvement efforts which led to a Level 3 assessment had a positive impact on all of these categories (Butler, 1995). Similar results were also reported at Motorola's Government Electronics Division (GED) (Diaz and Sligo, 1997). Notably, it was reported that each process maturity level reduced defect density (defects per million earned assembly equivalent lines of code) by a factor of 2.

Likewise, Bull HN Information Systems Inc., the US subsidiary of Groupe BULL, the fourth-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).

In addition, empirical research results have also provided support for the impact of software process maturity on various organizational performance indicators. More specifically, as organizations progress in terms of the maturity of their software processes, organizational performance indicators such as costs and project schedule improve. High maturity organizations are 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 aspects 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 but also increased development efforts. However, 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, and lead to, the following hypothesis.

HYPOTHESIS 1 (H1):

The higher the CMM for Software process maturity level, the higher the level of software project performance.

3.2.2 Software Development Risk and Software Project Performance

One of the objectives of this thesis is to further our understanding of the relationship between the software development risk and the performance of software development projects. In studying this relationship, let us thus address what prior research has found with regard to these two variables.

In prior research, software development risk factors were found to negatively affect overall software project efficiency. 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 most significantly related to project efficiency. 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).

Furthermore, 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.

Recent research also observed the existence of significant relationships between how organizations manage risk in software development and software project performance (Barki et al., 2001). In proposing an integrative contingency model of software project risk management, research has shown the importance of adopting appropriate risk management practices according to the project's level of risk exposure. For instance, high-risk projects were found to call for high information processing capacity approaches in their management. A software development project's risk management profile thus needs to be adapted to its degree of exposure to risk. In other words, projects that are more highly exposed to risk seem to require a different type of risk management profile than do low risk exposure projects (Barki et al., 2001).

It can therefore by 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.

HYPOTHESIS 2 (H2):

The higher the level of software development risk, the lower the level of software project performance.

IV. RESEARCH METHODOLOGY

4.1 Research Variables and Measures

The variables used in this study fall into three constructs: CMM software process maturity level, software project performance, and software development risk, and were adopted from prior research. Their relationships are illustrated in the conceptual framework previously shown in Figure 1 where each construct is represented by a box. Each construct is described in turn below.

4.1.1 CMM Software Process Maturity

Software process maturity levels measured on the CMM maturity scale reflect 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 CMM for Software (SW-CMM) process maturity level for each of the organizations polled in this study was previously determined by Carnegie Mellon Software Engineering Institute SW-CMM lead appraisers. Organizations are certified at a maturity level by participating in an 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 measure is presented below in Table 9.

Table 9
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?

4.1.2 Software Project Performance

Table 10
Measure of Software Project Performance

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

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 CMM software process improvement.

Specifically, this construct is comprised of 24 items that together assess software project performance along two dimensions: process performance which takes into account how well the software development process went, and product performance which considers the performance of the system, product, or output that is delivered to the end-user (Deephouse et al., 1995-1996; Nidumolu, 1995,1996). These variables are shown in Table 10. Twelve variables assess process performance along three dimensions: learning, control, and quality of interactions. The twelve other variables evaluate product performance and fall into three categories as well: operational efficiency, responsiveness, and flexibility. All 24 items were rated on a 7-point Likert-type scale varying from 1 (very poor) to 7 (very good).

4.1.3 Software Development Risk

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 risk factors are presented in Table 11.

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 46 items grouped 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".

Table 11
Measure of Software Development Risk

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.
Degree of Expertise	Degree 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
	Degree 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
	Degree 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)

Table 11 (Continued)

Variable	Variable	Item	Measure
Degree of Expertise (Continued)	Degree 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.
	Degree 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

4.2 Data Collection and Research Sample

4.2.1 Data Collection Procedure

In order to test the relationships depicted in the research model, a survey research methodology was adopted. Approximately 1000 officially SW-CMM-appraised organizations were invited to answer an online questionnaire containing the items used to assess the constructs. Invitations to participate in the study were specifically directed at individuals who were knowledgeable about their organization's software development projects as all of the items pertained to software projects. As such, respondents included individuals from a variety of positions, all related to software development projects and process improvement, including IS executives, software project managers, as well as quality and process managers. All respondents were assured that their responses would be kept strictly confidential and remain completely anonymous. Data collection was conducted over a two-month period. Invitations to participate in the study were sent to organizations through phone, e-mail, and online discussion group invitations. Reminder phone calls and e-mails were then used in order to obtain additional responses.

Upon completing data collection, roughly 200 questionnaires were returned. After deleting incomplete questionnaires, 107 usable questionnaires were left. This consists of a response rate of approximately 10%.

4.2.2 Sample Characteristics

Of the 107 returned surveys, 44% came from either project or quality managers/leaders. 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, roughly 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. Table 12 provides the descriptive statistics for the sample.

Table 12
Sample Descriptive Statistics

Position	
IS Executive	4
Project Manager/Leader	25
Quality Manager/Leader	22
Quality Analyst/Consultant/Engineer	9
Process Manager/Leader	3
Software Developer/Engineer	12
Other	23
Not Reported	9
Average number of team members on last software development project	
3-20 members	49
21-40 members	21
41-60 members	10
61-80 members	8
81-100 members	4
100 + members	9
Not reported	6
Primary Sector	
Software Development & Services	45
IT Services & Solutions	12
Healthcare	6
Finance	4
Government	6
Other	22
Not reported	12
Organization Size (Number of Employees)	
Under 100 employees	17
101-500 employees	22
501-1000 employees	9
1001-5000	26
5001-50000 employees	12
Above 50000 employees	5
Not reported	16

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 was generated using a mean substitution. The data set was also screened for univariate and multivariate outliers. Since most of the variables used in the study were measured on a 7-point scale, all of the data were kept for analysis as no extreme values were found.

V. DATA ANALYSIS AND RESEARCH RESULTS

5.1 Assessment of the Measurement Model

Partial least squares (PLS) was used to test the research model and hypotheses. PLS consists of a regression-based technique that can estimate and test the relationships among constructs. When testing a model, PLS produces loadings between items and constructs and estimates standardized regression coefficients (i.e. beta coefficients) for the paths between the constructs. Moreover, as PLS is a latent structural equation modeling technique that uses a component-based approach, the demands on sample size are minimized (Chin, 1998). The required sample size for a PLS analysis consists of ten times the number of items contained in the largest construct (Chin et al., 1996). As the largest construct in this study was degree of user support with 8 items, the final sample size of 107 was largely sufficient.

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. Taken together, the measurement model and the structural model form a network of constructs and measures. While the items weights and loadings reveal the strength of the measures, the estimated path coefficients indicate the strength and the sign of the theorized relationships and thus reveal the strength of the structural model (Chin et al., 1996).

5.1.1 Reliability Assessment

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 13
PLS Factor Loadings

Loadings	CMM Level	Risk	Software Project Performance
LEV	**1**		
RF1		0.26	
RF2		**0.51**	
RF3		**0.92**	
RF4		0.11	
RF5		0.41	
PR1			**0.95**
PR2			**0.93**

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 which have 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 (RF2) has a value of 0.51 while degree of expertise (RF3) equals 0.92 and are shown in bold in Table 13. Three other risk variables, technological acquisition (RF1), organizational environment (RF4), and application complexity (RF5), were dropped as their loadings on the risk construct were too low. In the case of the software project performance construct, both variables have high loading values above 0.9 as shown in bold in Table 13.

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 consistant. A score of 0.7 or higher indicates adequate construct reliability (Nunnally, 1978). As shown in Table 14, based on this criterion, the software development risk (α = 0.83) and software project performance (α = 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. Indeed, software development risk has an AVE of 0.84 and software project performance has an AVE of 0.97 as shown in Table 14. Therefore, significant variance was shared between each item and their respective construct, indicating adequate variance extracted and construct reliability.

Table 14
Reliability Assessments

Construct	Reliability, α	ICR	AVE
Software Development Risk	0.83	0.70	0.84
Software Project Performance	0.95	0.94	0.97

5.1.2 Convergent and Discriminant Validity Assessment

In order to evaluate convergent and discriminant validity, a comparison between the average variance extracted of each variable and the variance shared between the constructs (the squared correlations between the constructs) was analysed as suggested by Fornell and Larcker (1981). A PLS run was conducted to obtain the covariance matrices of all measures used to evaluate the loadings of the different 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 13, this is the case with specific variables for all three constructs. Indeed, both the process performance and product performance variables (PR1 and PR2) 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 acquisition (RF1), organizational environment (RF4), and application complexity (RF5), were dropped as their loadings on the risk construct were too low. All other variables whose loading values are shown in bold in Table 13 were kept as they exhibit clear convergent and discriminant validity.

Moreover, Table 15 shows the square root of the average variance extracted for all three constructs. It is thus obvious that each construct is 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 and convergent validity are satisfactory.

Table 15
Variance Shared Between Constructs

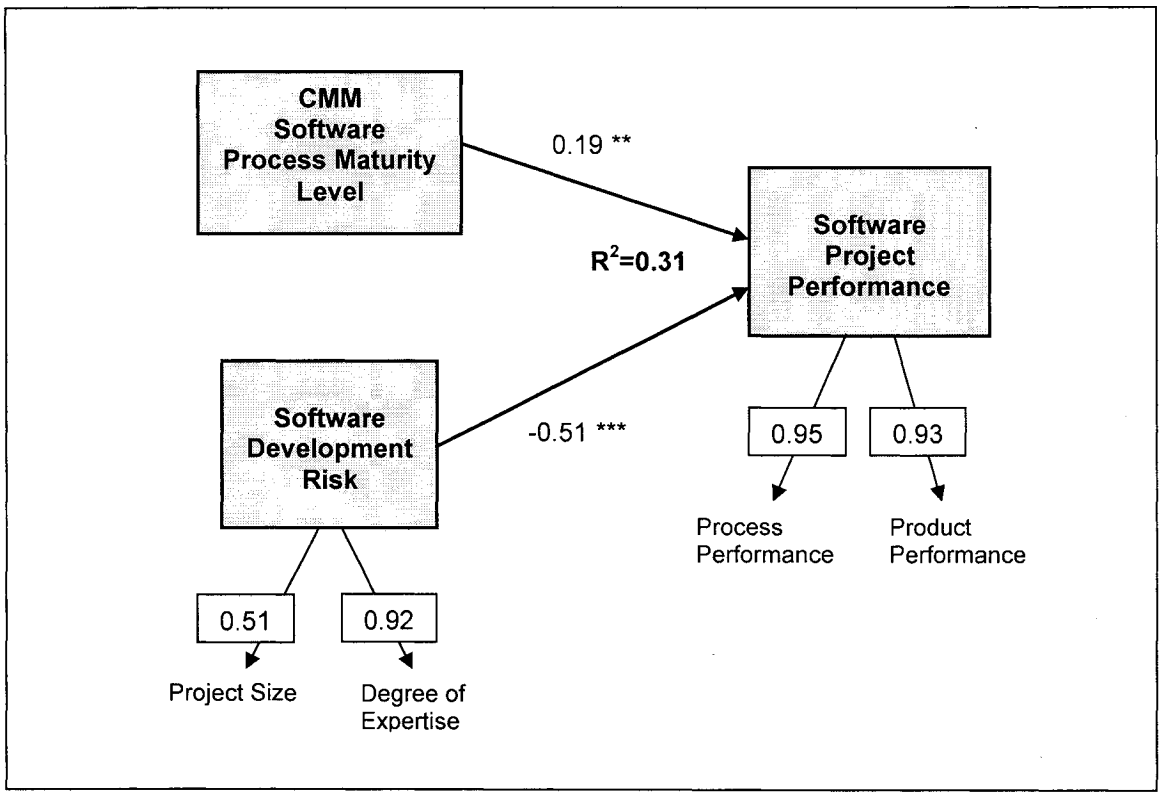
Construct	CMM Level	Risk	Software Project Performance
CMM Level	1		
Risk	0.02	0.84	
Software Project Performance	0.04	0.25	0.97

5.2 Assessment of the Structural Model

The primary objective of this study is to provide empirical evidence as to the relationships between CMM software process maturity levels and software project performance while assessing the impact of software development risk on project performance. Consequently, having validated the measures that comprise the conceptual model, advancing to the next phase of the research will allow for an empirical assessment of the research framework and the testing of the proposed hypotheses. Each hypothesis corresponds to a path in the conceptual model. Support for the hypothesis can be determined by examining the sign (positive or negative) and statistical significance for its corresponding path.

The structural model and hypotheses were therefore assessed by looking at 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 CMM software process maturity levels and software project performance. A positive relationship was predicted. In other words, increased levels of CMM software process maturity were predicted to lead to higher levels of software project performance. Results indicate that CMM software process maturity levels are indeed significantly and positively associated with 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 and 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 7).

Figure 7
Parameters for the Research Model



*: $p < 0.05$; **: $p < 0.01$; ***: $p < 0.001$

In short, our analysis provides support for each hypothesis. Findings indicate that CMM software process maturity levels are positively and significantly related to software project performance (H1 supported) and that software development risk is negatively and significantly related to the performance of software development projects (H2 supported).

VI. DISCUSSION OF FINDINGS

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

To answer these questions, a new 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 was collected from organizations that were officially CMM-appraised by SEI-authorized lead appraisers. 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.

Findings indicate that software project performance increases with higher levels of CMM software process maturity. In other words, organizations that find themselves higher up on the maturity ladder exhibit 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 based on the Capability Maturity Model (Fitzgerald and O'Kane, 1999; Herbsleb et al., 1997; Diaz and Sligo, 1997; Herbsleb and Goldenson, 1996; Goldenson et al., 1995; Herbsleb et al., 1994). It thus seems apparent that as a staged evolutionary model that provides an incremental roadmap towards process control and continuous process improvement, the CMM can now clearly be tied to performance metrics as was shown in this empirical study. Organizations that adopt the many key practices included in the CMM can anticipate process improvement that translates into performance increases in the form of both process and product performance. Moreover, practicing managers must also keep in mind the threat of software development risk. Such common factors as large software projects, software developers' management abilities, as well as user involvement in development projects 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. Managers must therefore 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.

The following section discusses these findings with regard to their implications for research and practice.

6.1 Implications for Research

This study makes significant contributions to software process improvement research. As noted earlier, while many case studies on the benefits of CMM process improvement initiatives in specific organizational settings have been made available, this is the first known study of its kind to integrate CMM maturity levels, from officially appraised organizations, as a construct, along with performance and risk variables, into a conceptual framework. This effort has not only allowed for a preliminary empirical investigation of the impact of CMM maturity levels on

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

Many avenues for future research can be considered. Notably, it would be interesting to further delve 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 into the inner workings of each maturity level and would consist of a natural extension of this 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 assess in what way 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 increasing adoption of the CMM model for software process improvement, this is one type of interrogation that requires further investigating. Additionally, longitudinal studies also need to be considered. As the type of survey research conducted here consists of a one-time snapshot of a given organization, precious information could be obtained in following up on organizations that progress from one level to the next. Performance could be assessed as organizations move up the CMM maturity scale in order to detect and analyse performance variations.

6.2 Implications for Practice

These results are also of great interest to practicing managers as they consist of the first known effort at linking CMM maturity levels from officially appraised organizations to

performance metrics in a large-scale empirical study. This study answers the challenge to move beyond isolated case studies and anecdotes that have so often characterized this sort of debate in the past (Herbsleb et al., 1997).

The decision for an organization to adopt a process improvement framework such as the CMM is major. The Capability Maturity Model requires a considerable amount of time and effort to implement. An organization roughly takes 18 to 30 months to move up one full maturity level (Hayes and Zubrow, 1996). Moreover, official appraisals at a given maturity level are obtained by participating in assessments conducted by SEI-authorized lead appraisers. Appraisers' services are typically valued at 50000$ per appraisal depending on the travel involved and the size of the organization while many organizations also employ consulting services which can generate considerable extra costs (Ingalsbe et al., 2001). All of these significant investments stress the need for hard evidence that can justify the efforts required to implement CMM process improvement initiatives. In tying CMM process maturity to software project performance, a key organizational concern when it comes to software process improvement investments, this study points to reliable results in terms of the payoff of CMM process improvement initiatives. This large-scale survey of officially assessed organizations provides managers with more generalizable findings than what was previously reported in a number of case studies (Goldenson et al., 1995). Moreover, software managers or IS executives that may be reluctant to invest in CMM improvement initiatives without the knowledge of the payoff now have preliminary findings that can undoubtedly influence their ultimate decision.

6.3 Limitations of the Study

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, as well as to further explore and expand upon its findings. For one, as this consists of a preliminary study into the effects of CMM software process maturity levels on software project performance while assessing the influence of performance of software development risk, the internal components of each CMM level were not included in the analysis. Therefore, it is not possible to determine whether specific key process areas or what key practices had a greater impact on process and product performance. Furthermore, this same limitation does not tell us whether risk management processes incorporated into some CMM levels actually mitigate software development risk factors as measured by the risk construct. Finally, this study consists of 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 the variations of performance with regard to an organization's progression on the CMM maturity scale can 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.

VII. CONCLUSION

Isolated case studies and anecdotes have often characterized the field of software process improvement in the past (Herbsleb et al., 1997). Too little progress in the way of theoretical development and testing has been made in this area. In an attempt to respond to this need, this study, through a set of successive stages of testing and analysis has arrived at a new conceptual framework that provides invaluable insight into the Capability Maturity Model for Software and software process improvement. The variables comprising the instrument were derived from previous research in the field of software project performance and risk in information systems development. The resulting measurement and theoretical models were empirically tested in a survey of more than 100 officially CMM-appraised organizations and provide invaluable insights into CMM software process improvement and the hazards of software development risk. It has shown substantial business benefits for organizations moving from the lower to the higher CMM maturity levels in the form of a positive impact on software project performance assessed as both process and product performance. Moreover, with regard to the growing concern for risk in both research and practice, this study has contributed to providing additional evidence in terms of the importance of managing and mitigating risk factors in software development projects as they have been shown to clearly influence the performance of software development projects in a negative way.

Moreover, in addition to providing initial empirical support for the findings, the research model provides interesting avenues for future research. In fact, the research model can not only be studied in more detail but can also be expanded upon in terms of CMM risk management processes and their mitigating effects on software development risk. In addition, a longitudinal study can provide additional insights into the variations of performance with regard to an organization's progression on the CMM maturity scale. In short, the model presented here not

only provides much needed empirical support for the benefits of CMM process maturity but also consists of a solid basis that can be built upon to better understand CMM process improvement through sound scientific investigations.

REFERENCES

Abdel-Hamid, T.K., "Investigating the Impact of Managerial Turnover/Succession on Software Project Performance", *Journal of Management Information Systems*, Vol. 9, No. 2 (Fall 1992), pp. 127-144

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

Aladwani, A.M., "An Integrated Performance Model of Information Systems Projects", *Journal of Management Information Systems*, Summer 2002, Vol. 19, No. 1, pp. 185-210

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*, Vol. 10, No. 2 (Fall 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*, Vol. 17, No. 4 (Spring 2001), pp. 37-69

Benno, S. and Frailey, D., "Software Process Improvement in DSEG- 1989-1995", Texas Instruments Technical Journal, (12:2), March-April 1995, pp. 20-28

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

Boehm, B.W. Software Risk Management: Principles and Practices. *IEEE Software*, Vol. 8, No. 1 (January 1991), pp. 32-41

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

Butler, K.L., "The Economic Benefits of Software Process Improvement", *CrossTalk*, July 1995, pp. 14-17

Chin, W.W., PLS-Graph Software, 1995, Version 2910208

Chin, W.W., *Structural Equation Modeling in IS Research*, ISWorld Net Virtual Meeting Center at Temple University, 2-5 November 1998 [On-Line]. Available http://www.interact.cis.temple.edu/~vmc

Chin, W.W., Marcolin, B.L., Newsted, P.R. (1996), *A Partial Least Squares Latent Variable Modeling Approach for Measuring Interaction Effects: Results from a Monte Carlo Simulation Study and Voice Mail Emotion/Adoption Study*, Proceedings of International Conference on Information Systems, Cleveland.

CMMI Product Team, *CMMI for Software Engineering, Version 1.1, Continuous Representation* (CMMI-SW, V1.1) CMU/SEI-2002-TR-028, Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University, 2002

CMMI Product Team, *CMMI for Software Engineering, Version 1.1, Staged Representation* (CMMI-SW, V1.1) CMU/SEI-2002-TR-029, Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University, 2002

Cooprider, J., and Henderson, J.A., "Technology-Process Fit: Perspectives on Achieving Prototyping Effectiveness", *Journal of Management Information Systems*, Vol. 7, No. 3 (Winter 1990-91), pp. 67-87

Cronbach, A., "Coefficient Alpha and the Internal Structure of Tests", *Psychometrika*, (16:3), September 1951, pp. 297-334

Deephouse, C., Mukhopadhyay, T., Goldenson, D., and Kellner, M., "Software Processes and Project Performance", *Journal of Management Information Systems*, Vol. 12, No. 3 (Winter 1995-96), pp. 187-205

Diaz, M., and Sligo, J., "How Software Process Improvement Helped Motorola", *IEEE Software*, Vol. 14, No. 5, September/October 1997, pp. 75-81

Dion, R., "Process Improvement and the Corporate Balance Sheet", IEEE Software, (10:4), July 1993, pp. 28-35

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

Fornell, C., Larcker, D.F. (1981), "Structural Equation Models with Unobservable Variables and Measurement Errors", *Journal of Marketing Research*, (18:2), pp. 39-50

Goldenson, D.R. and Herbsleb, J.D., "After the Appraisal: A Systematic Survey of Process Improvement, its Benefits, and Factors that Influence Success", Software Engineering Institute, Pittsburgh, PA, CMU/SEI-95-TR-009, ESC-TR-95-009, August 1995, pp. 1-66

Goldenson, D.R., and Gibson, D.L., "Demonstrating the Impacts and Benefits of CMMI: An Update and Preliminary Results", Software Engineering Institute, Pittsburgh, PA, CMU/SEI-2003-SR-009, October 2003, pp. 1-55

Guarro, S.B., "Principles and Procedures of the LRAM Approach to Information Systems Risk Analysis and Management", *Computers & Security*, Vol. 6, No. 6 (December 1987), pp. 493-504

Guinan, P.J., Cooprider, J.G., and Faraj, S., "Enabling Software Development Team Performance During Requirements Definition: A Bahavioral Versus Technical Approach", *Information Systems Research*, Vol. 9, No. 2 (June 1998), pp. 101-121

Hair, J.F., Anderson, R.E., Tatham, R.L., Black, W.C. (1992), *Multivariate Data Analysis with Readings*, 3rd Edition, Macmillan, New York.

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

Hammer, M. and Champy, J. (1993), *Reengineering the Corporation*, Nicholas Brealey Publishing.

Hammer, M. and Champy, J., "Don't Automate, Obliterate", *Harvard Business Review*, July-August 1990, pp. 104-112

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*, Vol. 46, No. 4 (April 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*, August 1995

Henderson, J.C., and Lee, S., "Managing I/S Design Teams: A Control Theories Perspective", *Management Science*, Vol. 38, No. 6 (1992), pp. 757-777

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, August 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, March 1996, 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*, Vol. 40, No. 6, July 1997, pp. 30-40

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

Humphrey, W. S., Snyder, T.R., and R. R. Willis, "Software Process Improvement at Hughes Aircraft", *IEEE Software*, Vol. 8, No. 4, July 1991, pp. 11-23

Hunter, R.B. and Thayer R.H., *Software Process Improvement*, Wiley-IEEE Computer Society Press, 1st edition (November 27, 2001), 630 pages

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

Jalote, P., *CMM in Practice: Processes for Executing Software Projects at InfoSys*, Reading, Mass: Addison-Wesley, 2000, pp. 372

Jiang, J. and Klein, G., "Software Development Risks to Project Effectiveness", *The Journal of Systems and Software 52 (2000)*, pp. 3-10

Jiang, J., Klein, G., Means, T.L. (2000), "Project Risk Impact on Software Development Team Performance", *Project Management Journal*, (31:4), pp. 19-26

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*, Vol. 41 (2004), pp. 279-288

Jung, H.W. and Goldenson, D.R., *The Internal Consistency of Key Process Areas in the Capability Maturity Model (CMM) for Software (SW-CMM)*, Software Engineering Institute, Pittsburgh, PA, CMU/SEI-2002-TR-037, December 2002, pp. 1-71

Känsälä, K., "Integrating Risk Assessment with Cost Estimation", *IEEE Software*, Vol. 14, Iss. 3 (May-June 1997), pp. 61-37

Kaplan, S., and Garrick, J.B., "On the Quantitative Definition of Risk", *Risk Analysis*, Vol. 1, No. 1 (1981), pp. 11-27

Keil, M., Wallace, L., Turk, D., Dixon-Randall, G., and Nulden, U., "An Investigation of Risk Perception and Risk Propensity on the Decision to Continue a Software Development Project", *The Journal of Systems and Software*, Vol. 53, No. 2 (August 2000), pp. 145-157

Kraut, R.E., and Streeter, L.A., "Coordination in Software Development", *Communications of the ACM*, Vol. 38, No. 3 (1995), pp. 69-81

Krishnan, M.S., Mukhopadhyay, T., and Zubrow, D., "Software Process Models and Project Performance", *Information Systems Frontier*, October 1999, Vol. 1, No. 3, pp. 267-277

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

Linnerooth-Bayer, J., and Wahlstrom, B., "Application of Probabilistic Risk Assessments: The Selection of Appropriate Tools", *Risk Analysis*, Vol. 11, No. 2 (1991), pp. 239-248

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

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

Members of the Assessment Method Integrated Team, "Standard CMMI Appraisal Method for Process Improvement (SCAMPI), Version 1.1: Method Definition Document, CMU/SEI-2001-HB-001, December 2001, 245 pages

Myers, W., "Hard data will lead managers to quality", *IEEE Software*, 1994, Vol. 11, No. 2, pp. 100-101

Na, K.S., Li, X., Simpson, J.T., and Kim, K.Y., "Uncertainty Profile and Software Project Performance: A Cross-National Comparison", *The Journal of Systems and Software*, Vol. 70, Iss. 1-2 (February 2004), pp. 155-163

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

Nidumolu, S., "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 (Fall 1996), pp. 77-113

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

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, February 1993, pp. 1-82

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, pp. 1-372

Peterson, D.K. and Kim, C., "Perceptions on IS Risks and Failure Types: A Comparison of Designers form the United States, Japan and Korea", *Journal of Global Information Management*, Vol. 11, No. 3 (July-September 2003), pp. 19-38

Post, G.V., and Diltz, D.J., "A Stochastic Dominance Approach to Risk Analysis of Computer Systems", *MIS Quarterly*, Vol. 10, No. 4 (December 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*, Vol. 37, No. 6 (2000), pp. 335-346

Ravichandran, T., "Software Reusability as Synchronous Innovation: A Test of Four Theoretical Models", *European Journal of Information Systems*, Vol. 8, No. 3 (1999), pp. 183-199

Ravichandran, T., and Rai, A., "Quality Management in Systems Development: An Organizational System Perspective", *MIS Quarterly*, Vo. 24, No. 3 (2000), pp. 381-416

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

Rivard, S., Huff, S. (1988), "Factors of Success for End-User Computing", *Communications of ACM*, (31:5), pp. 552-561

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

Ropponen, J. and Lyytinen, K., "Can Software Risk Management Improve System Development: An Exploratory Study", *European Journal of Information Systems*, Vol. 6, No. 1 (March 1997), pp. 41-50

Saleem, N., "An Empirical Test of the Contingency Approach to User Participation in Information Systems Development", *Journal of Management Information Systems*, Vol. 13, No. 1 (Summer 1996), pp. 145-166

Saarinen, T. (1990), "System Development Methodology and Project Success: An Assessment of Situational Approaches", *Information & Management*, (19:3), pp. 183-193

Schmidt, R., Lyytinen, K., Keil, M., and Cule, P., "Identifying Software Project Risks: An International Delphi Study", *Journal of Management Information Systems*, Spring 2001, Vol. 17, No. 4, pp. 5-36

Sheard, S.A., "The Frameworks Quagmire", *CrossTalk*, Volume 10, Number 9, 1997, pp. 17-22

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*, Spring 2004, (35:2), pp. 289-321

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

Wohlwend, H. and Rosenbaum, S., "Software Improvements in an International Company", *Proceedings of the 15th international conference on Software Engineering*, Baltimore, Maryland, May 17-21, 1993, Los Alamitos, Ca.: IEEE Computer Society Press, pp. 212-220

Zahran, S., *Software Process Improvement: Practical Guidelines for Business Success*, Addison-Wesley, 1998, pp. 1-447

Paulk, M., Curtis, B., Chrissis, M. B. and Weber, C., Capability Maturity Model for Software, Version 1.1, *IEEE Software*, July 1993, pp. 18-27

Process Maturity Profile, Software CMM CBA IPI and SPA Appraisal Results, 2003 Mid-Year Update, September 2003

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

APPENDIX

What is your SW-CMM Software Process Maturity Level as it was last determined by an SEI lead assessor? Please circle one of the following.

1 2 3 4 5

Important: When answering this survey, please always consider your organization's most recently completed software development project.

The questions on pages 1 to 3 are about the level of risk exposure of your organization's most recently completed software development project.

Please consider the most recently completed software development project undertaken by your firm when considering the following statements.

These statements are scored on a 7-point scale, with responses ranging from (1) "Strongly disagree" to (7) "Strongly agree".

	Strongly Disagree						Strongly Agree
The new system required the acquisition of new hardware.	1	2	3	4	5	6	7
The new system required the acquisition of new software.	1	2	3	4	5	6	7
A large number of hardware suppliers were involved in the development of this system.	1	2	3	4	5	6	7
A large number of software suppliers were involved in the development of this system.	1	2	3	4	5	6	7
There were a large number of people on the project team.	1	2	3	4	5	6	7
There were a large number of different "stakeholders" on the project team (e.g., IS staff, users, consultants, suppliers, customers).	1	2	3	4	5	6	7
The project size was large.	1	2	3	4	5	6	7
There are a large number of users using the system.	1	2	3	4	5	6	7

Please evaluate the software development degree of expertise in terms of the following. These statements are scored on a 7-point scale, with responses ranging from **(1) "No Expertise"** to **(7) "Outstanding Expertise"**.

	No Expertise						**Outstanding Expertise**
Ability to work with uncertain objectives	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Ability to work with top management	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Ability to work effectively as a team	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Ability to understand human implications of a new system	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Ability to carry out tasks effectively	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Ability to carry out tasks quickly	**1**	**2**	**3**	**4**	**5**	**6**	**7**
In-depth knowledge of the functioning of user departments	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Overall knowledge of organizational operations	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Overall administrative experience and skill	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Expertise in the specific application area of the system	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Familiar with this type of application	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Development methodology used in this project	**1**	**2**	**3**	**4**	**5**	**6**	**7**

	No Expertise						Outstanding Expertise
Development support tools used in this project	1	2	3	4	5	6	7
Project management tools used in this project	1	2	3	4	5	6	7
Implementation tools used in this project	1	2	3	4	5	6	7

Please evaluate the users of the developed system in terms of the following. These statements are scored on a 7-point scale, with responses ranging from **(1) "Strongly Disagree"** to **(7) "Strongly Agree"**.

	Strongly Disagree						Strongly Agree
Users were not enthusiastic about the project	1	2	3	4	5	6	7
Users were not an integral part of the development team	1	2	3	4	5	6	7
Users were not available to answer questions	1	2	3	4	5	6	7
Users were not ready to accept the changes the system entailed	1	2	3	4	5	6	7
Users slowly responded to development team requests	1	2	3	4	5	6	7
Users had negative attitudes regarding the use of computers in their work	1	2	3	4	5	6	7
Users were not actively participating in requirement definition	1	2	3	4	5	6	7

	Strongly Disagree						Strongly Agree
Users were not very familiar with system development tasks	1	2	3	4	5	6	7
Users had little experience with the activities supported by the new applications	1	2	3	4	5	6	7
Users were not very familiar with this type of application	1	2	3	4	5	6	7
Users were not aware of the importance of their roles in successfully completing the project	1	2	3	4	5	6	7
Users were not familiar with data processing as a working tool	1	2	3	4	5	6	7

Please evaluate your firm's organizational environment with regard to its most recently completed software development project in terms of the following. These statements are scored on a 7-point scale, with responses ranging from **(1) "Strongly Disagree"** to **(7) "Strongly Agree"**.

	Strongly Disagree						Strongly Agree
The system required that a large number of user tasks be modified	1	2	3	4	5	6	7
The system led to major changes in the organization	1	2	3	4	5	6	7
In order to develop and maintain the system, the scheduled number of people-day was insufficient	1	2	3	4	5	6	7
In order to develop and maintain the system, the dollar budget provided was insufficient	1	2	3	4	5	6	7
The role of each member of the team was not clearly defined	1	2	3	4	5	6	7

	Strongly Disagree						Strongly Agree
The role of each person involved in the project was not clearly defined	1	2	3	4	5	6	7
Communications between those involved in the project were unpleasant	1	2	3	4	5	6	7
There was a great intensity of conflicts among team members	1	2	3	4	5	6	7
There was a great intensity of conflicts between users and team members	1	2	3	4	5	6	7

Please evaluate the developed application's complexity in terms of the following. These statements are scored on a 7-point scale, with responses ranging from **(1) "Strongly Disagree"** to **(7) "Strongly Agree"**.

	Strongly Disagree						Strongly Agree
Large number of links to existing systems	1	2	3	4	5	6	7
Large number of links to future systems	1	2	3	4	5	6	7

If, for some reason, the developed system had not been implemented or if it had had operational problems, what impact would this have had on your organization in terms of the following. These statements are scored on a 7-point scale, with responses ranging from **(1) "Little Impact"** to **(7) "Large Impact"**.

	Little Impact						Large Impact
Customer Relations	1	2	3	4	5	6	7

	Little Impact						Large Impact
Financial Health	1	2	3	4	5	6	7
Reputation of the information system department	1	2	3	4	5	6	7
Profitability	1	2	3	4	5	6	7
Competitive position	1	2	3	4	5	6	7
Organizational efficiency	1	2	3	4	5	6	7
Organizational image	1	2	3	4	5	6	7
The survival of the organization	1	2	3	4	5	6	7
Market share	1	2	3	4	5	6	7
Reputation of the user department	1	2	3	4	5	6	7
Ability to carry out current operations	1	2	3	4	5	6	7

The questions on this page are about the performance of your organization's most recently completed software development project.

How do you rate the performance of your organization's most recently completed software development project and the software that was delivered on each of the following statements?

These statements are scored on a 7-point scale, with responses ranging from (1) "Very Poor" to (7) "Very Good".

	Very Poor	OK	Very Good
Knowledge acquired by firm about use of key technologies	1	2	3
Knowledge acquired by firm about use of development techniques	1	2	3
Knowledge acquired by firm about supporting users' business	1	2	3
Overall knowledge acquired by firm through the project	1	2	3

	Very Poor	OK	Very Good
Control over project costs	1	2	3
Control over project schedule	1	2	3
Adherence to auditability and control standards	1	2	3
Overall control exercised over the project	1	2	3
Completeness of training provided to users	1	2	3
Quality of communication between DP and users	1	2	3
Users' feelings of participation in project	1	2	3
Overall quality of interactions with users	1	2	3
Reliability of software	1	2	3
Cost of software operations	1	2	3
Response time	1	2	3
Overall operational efficiency of software	1	2	3
Ease of use of software	1	2	3
Ability to customize outputs to various user needs	1	2	3
Range of outputs that can be generated	1	2	3
Overall responsiveness of software to users	1	2	3
Cost of adapting software to changes in business	1	2	3
Speed of adapting software to changes in business	1	2	3

	Very Poor	OK	Very Good
Cost of maintaining software over lifetime	1	2	3
Overall long term flexibility of software	1	2	3

By approximately what percentage, if any, did actual costs for the project overrun originally budgeted costs? (indicate underrun by negative sign) _____ %

By approximately what percentage, if any, did actual costs for the project overrun originally budgeted completion time? (indicate underrun by negative sign) _____ %

By approximately what percentage, if any, did actual systems and programming effort for the project overrun originally budgeted effort? (indicate underrun by negative sign) _____ %

Feel free to tell us about your organization. Please note that answering these questions is optional.

What is your position inside the organization? (e.g., project manager, IS executive, system developer, etc.) _____

Approximately how many team members were on your organization's most recently completed software development project? _____

What is your organization's primary industry? _____

What are your organization's assets? _____

How many individuals does your organization employ? _____