Available online at www.sciencedirect.com
ScienceDirect
The Journal of Systems and Software 80 (2007) 616–627
The Journal of Systems and Software
www.elsevier.com/locate/jss

Software quality and IS project performance improvements from software development process maturity and IS implementation strategies

Girish H. Subramanian [a,*], James J. Jiang [b,1], Gary Klein [c,2]

[a] School of Business, Penn State Harrisburg, E355 Olmsted Building, 777 W. Harrisburg Pike, Middletown, PA 17057, United States
[b] Department of Management Information Systems, University of Central Florida, Orlando, FL 32816-1400, United States
[c] College of Business and Administration, The University of Colorado at Colorado Springs, 1420 Austin Bluffs Parkway, P.O. Box 7150, Colorado Springs, CO 80933-7150, United States

Received 3 March 2006; received in revised form 18 June 2006; accepted 20 June 2006
Available online 26 July 2006

Abstract

The capability maturity model (CMM) is part of several software process improvement (SPI), six sigma, and total quality management (TQM) initiatives in organizations. SPI and continuous quality improvements are associated with better return on investment (ROI) for organizations. The purpose of this empirical research is to study the impact of the CMM on certain critical factors in information systems implementation strategy, software quality and software project performance. Our findings are that CMM levels do associate with IS implementation strategies and higher CMM levels relate to higher software quality and project performance. We also conclude that information systems (IS) implementation strategies have a significant impact on software quality and project performance. While certain IS implementation strategies – executive commitment and prototyping – have a significant impact on both software quality and project performance, training had a significant effect only on software quality and simplicity has a significant effect only on project performance.

Keywords: Capability maturity model; Software quality; Project management; Software process improvement

1. Introduction

Organizations continue to adopt models of total quality management (TQM) principles, with the capability maturity model (CMM) being the more common for the software development arena (Pressman, 2004). The CMM is part of a more comprehensive software process improvement (SPI) initiative and is typically measured in five levels of maturity (Paulk et al., 1993a). CMM, SPI and TQM are of prime interest in the battle for improving the performance of software development projects and reaching higher returns on investment for the adopting organization (Jones, 1998; Humphrey, 1988; McConnell, 2002). Of these approaches, CMM is very popular and accepted by numerous global software firms.

In addition to the use of SPI, a myriad of information systems and organizational factors are also important for the success of these efforts in organizations (Parzinger and Nath, 2000; Isaac et al., 2004). It is important to have the organizational, project and information system practices all support the CMM or process improvement efforts so that organizations can reap the maximum benefit (Isaac et al., 2004). Empirical research finds benefits in cycle time,

* Tel.: +1 717 948 6150; fax: +1 717 948 6456.
E-mail addresses: ghs2@psu.edu (G.H. Subramanian), jjiang@bus.ucf.edu (J.J. Jiang), gklein@mail.uccs.edu (G. Klein).
[1] Tel.: +1 407 823 4864; fax: +1 407 823 2389.
[2] Tel.: +1 719 262 3157; fax: +1 719 262 3494.

0164-1212/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2006.06.014

software quality, and development effort by following CMM initiatives (Harter et al., 2000; Parzinger and Nath, 2000). CMM certification in global software firms based in India is associated with improved operational performance, better software development, and better management practices than non-certified firms (Isaac et al., 2004). Still, the evidence is limited and not based on reflective models (Hansen et al., 2004).

Our research study continues empirical investigation of software development process maturity and its impact on the organization, based on a model derived from previous studies. The purpose of this study is to examine the impact of CMM on certain critical factors in information systems implementation strategy, software quality and software project performance. The selection of critical factors in information systems (IS) implementation strategies is based on previous literature, but is not intended to be fully comprehensive, there being other factors that can be investigated in future research. Specifically, we examine whether CMM levels have an association with IS implementation strategies, software quality, and software project performance.

2. Background

An understanding of the principles of the capability maturity model will clarify how quality improvements are expected. Likewise, there may be a relation of the levels in the CMM to established strategies of IS implementation. The background of each of these concepts is explored to provide sufficient understanding of each concept.

2.1. Capability maturity model

The capability maturity model (Paulk et al., 1993a,b; Pressman, 2004) is the central framework for software quality and process improvement. In this model, there are five levels. The CMM evaluation ranks software development organizations into one of the five levels. In the repeatable level (level 2), basic project management processes are established to track cost, schedule and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications. In level 3 (defined), the software process for both management and engineering activities is documented, standardized and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software. In level 4 (managed), detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled. Finally, in level 5 (optimized), continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies (Paulk et al., 1993a). The CMM's five levels are summarized:

Capability maturity model levels and description

Process maturity levels	Description
Level 1: Initial	Ad Hoc and occasionally chaotic
Level 2: Repeatable	Basic project management; process discipline to repeat earlier successes
Level 3: Defined	Includes level 2; software processes standardized and integrated; Projects use these approved processes
Level 4: Managed	Includes level 3; Detailed metrics of software product and process are collected; Software process and product controlled using these metrics
Level 5: Optimized	Includes level 4; Continuous process improvements enabled by quantitative feedback; innovative ideas and technologies developed based on feedback

To assess maturity in practice, a team is selected which administers and evaluates the maturity questionnaire. This questionnaire was designed by the Software Engineering Institute (SEI) and addresses key practices grouped into: (1) commitment to perform; (2) ability to perform; (3) activities performed; (4) measurement and analysis; and (5) verifying implementation. The team also visits the sites being assessed and produces a list of strengths and weaknesses. The final step involves the preparation of a key process area profile. This profile highlights the areas where the organization has, or has not, satisfied goals determined in prior steps (Parzinger and Nath, 2000).

Empirical studies have evaluated the use of CMM and its impact on the organization. In their empirical research on CMM, higher levels of CMM process maturity are shown to be associated with higher software product quality (Harter et al., 2000). The net effect of higher levels of CMM process maturity is improved project performance as measured by reduction in cycle times and development effort (Harter et al., 2000). The impact of TQM implementation, such as the CMM, on measures of software quality is studied in Parzinger and Nath (2000). TQM implementation factors were found to have a significant positive relationship with measures of software quality. Note that the objective under the CMM concept is to rise to the up-most level (optimizing) where continuous improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies. To reach this level, the software development process must be documented, standardized and integrated, and detailed measures of both the process and the product quality are collected (Parzinger and Nath, 2000).

CMM certification in global software firms based in India is associated with improved operational performance, better software development, and better management practices than non-certified firms (Isaac et al., 2004). Quality certified firms are found to be significantly better than non-certified firms in the software industry with respect to TQM practices and operational performance indicators. For further investigation, quality certified firms have been classified as ISO certified and CMM certified based on their quality certification and ANOVA tests have been done for each group. It is found that there is no significant difference between the non-certified firms and ISO 9000 certified firms, whereas there are significant differences between the non-certified firms and CMM certified firms with respect to TQM constructs and operational performance indicators. It is also observed that CMM (level 4 or 5) firms are significantly better than the ISO certified firms (Isaac et al., 2004).

2.2. Critical factors in information systems implementation strategies

We explore four conceptual strategies for IS implementation. These strategies were first proposed by Alter (1979).

2.2.1. Keep it simple (simplicity)

Functional simplicity (minimum necessary to meet requirements), structural simplicity (modular architecture), and code simplicity (following a coding standard) are key components of simplicity in software (Pressman, 2004). Complicated, integrated software systems require careful planning before implementation (Rajagopal and Frank, 2002). The *Wall Street Journal* (Bailey, 1999) documented the situation of two leading firms in the trash hauling industry (Allied Waste and Waste Management) that abandoned their complex SAP implementation. Thomas Van Weelden, CEO of Allied Waste, noted one of the primary concerns: “They [SAP] expect you to change your business to go with the way the software works.”

Complex systems inherently present unique risks due to tightly linked interdependencies of business processes, relational databases, and process re-engineering. Companies need to have an understanding of such risks when planning and conducting assurance engagements of the reliability of these complex computer systems (Wright and Wright, 2002). Therefore, assurance providers must be cognizant of enterprise system implementation related risks, including selection or design and installation of hardware and software and process re-engineering (Scheer and Habermann, 2000). Avoiding complexity and keeping it simple considerably lessens the implementation risk of information systems.

2.2.2. Executive (or top management) participation and commitment

Employee empowerment and executive commitment are two key factors used in Parzinger and Nath, 2000. Open organizations, employee empowerment and executive commitment are more critical to the success of TQM (Powell, 1995). The leadership of top management is central to the creation of a TQM organization (Kanji and Sa, 2002; Kanji, 1996). Many other researchers also pointed out that the impetus for quality improvement efforts in any organization should start from the top and unfold downward to the lower level (Milakovich, 1995; Ahire, 1996; Li et al., 2000). In the absence of committed leadership, the TQM efforts may fail (Jorgensen, 1999). Leadership is identified as a critical factor for achieving better quality practices in the software industry as well (Vitharana and Mone, 1998; Parzinger and Nath, 1998; Wynekoop and WaIz, 2000).

If the project is strictly software with little user involvement, the stage is set for many problems to occur (Subramanian and Hoffer, 2005). Users need to be involved in the decision making process; they need to take ownership of the processes that are going to be changed by the system (Wright and Wright, 2002). Employees who are end users of the system must understand that they are becoming knowledge workers, which requires an organizational perspective beyond simply knowing their specific function (Rajagopal and Frank, 2002). Freedom for all team members to make suggestions during software development or project execution is found to be a good practice for improving the quality of software (Shrednick et al., 1992; Gong et al., 1998; Li et al., 2000).

2.2.3. Training

Training is a crucial component in continuous improvement. Training also helps to improve employee participation and involvement in quality programs through propagation of priorities and missions of the organization (Harel and Tzafrir, 1999). Training influences software quality as shown in Parzinger and Nath, 2000. The specific skills training implementation factor encompasses the degree of training the software developer has in skills such as quality function deployment, statistical methods and quality principles (Parzinger and Nath, 2000). In their study, when the overall success measure is regressed against the eight TQM factors, specific skills training was one of the three factors that is significant (Parzinger and Nath, 2000). When it came to making improvements in the CMM level, customer needs assessment and specific skills training were the significant factors. As CMM calls for documentation, measurement and increased efficiencies in the development process, the importance of training is self-evident (Parzinger and Nath, 2000).

2.2.4. Prototyping/evolutionary development

Prototyping is a critical component in evolutionary development (Pressman, 2004). Prototyping is also an integral part in the value-based systems engineering theory (VBSET – Boehm et al., 2005). Prototyping is recommended for clarity in understanding system requirements and in planning systems architecture (Boehm and Papaccio, 1988) and thus can help in improving software quality. Prototyping could also be effectively used in software risk management as shown in the spiral model (Boehm, 1988) and can help in improving project performance. With the use of 4th and 5th generation languages and reusable software components, prototyping can be effectively used in building information systems (Pressman, 2004).

2.3. Information systems (IS) project outcomes

Though many proposals have been made to report the success of an IS project, several themes remain consistent in the literature. These involve measures of both the product in terms of quality and the process in terms of meeting objectives and schedules for the project.

2.3.1. Software quality

The quality of software is estimated by many of its attributes such as reliability, integrity, maintainability, enhanceability (extensibility), usability, portability, and reusability. These are referred to as the "ilities" of software (Yang, 2001). Yang, 2001 also pointed out that the functionality of the software and the appearance of the user interface could affect software quality. Jorgensen (1999) stated that software quality can be indirectly measured and/or predicted with the help of these characteristics. Many other researchers (for example, Ben-Menachem and Marliss, 1997; Cho, 1998; Yourdon, 1992; Arthur, 1993) also have pointed out that these characteristics could be used as a measure of software quality, since they affect user (customer or client) satisfaction, which is ultimately treated as the indicator of the quality of a product.

Measurement is an essential and challenging element of highly efficient software engineering culture. The need for measurement becomes prominent when projects are running over budget and schedule (Ashrafi, 1998; Prahalad and Krishnan, 1999; Jalote, 2000). Effectiveness of quality management depends on the effectiveness with which performance and results are measured (Kanji and Sa, 2002). Parzinger and Nath (1998, 2000) also identified metrics or measures as critical factors in software quality management.

2.3.2. Project performance

Schedule slippage, product quality, and process capability are some of the project performance variables used in (Murugappan et al., 2003). In time and within budget are common yardsticks for project performance. Specifically, reducing cycle times and development effort (main factor in software cost and budget) are project performance yardsticks in Harter et al., 2000. Risk management is also considered a key variable in software project management (Boehm, 1988). Value-based software engineering also actively considers project performance as value added to key stakeholders (Boehm et al., 2005).

3. Research model and hypotheses

The impact of CMM on information systems implementation strategies and IS project outcomes is the main focus of the research model. The research model also studies the impact of information systems implementation strategy on IS project outcomes.

CMM levels are associated with customer needs assessment (Parzinger and Nath, 2000). As Pressman, 2005 states "ideally, the prototype serves as a mechanism for identifying software requirements (p. 52)", we argue that prototyping is a key aspect of customer needs assessment. Hence, CMM levels should be associated with the prototyping approach. Isaac et al., 2004 also point to CMM certified firms having significantly better training and executive commitment in their organizations. Harter et al., 2000 also suggest future research to study " the impact of process improvement (CMM levels) on the non-engineering activities (we assume that to be IS implementation strategies) that support software development" (p. 465). Card, 1991 observes that bureaucratic organizations might score well on CMM and hence we may expect that CMM levels may not impact "keep it simple" IS implementation strategy. Again, it can be argued that CMM provides standardization and documentation (Parzinger and Nath, 2000) and hence should impact the "keep it simple" strategy. Hence, the first part of our research model, shown in Fig. 1, aims to study the association between the CMM levels and different IS implementation factors.

CMM levels are shown to influence both software quality and project performance variables such as cycle time and development effort in Herbsleb et al., 1997; Harter et al., 2000. In our research model, we argue that CMM levels influence the choice of IS implementation factors such as training, executive commitment, simplicity (keep it simple), and prototyping which in turn impacts software

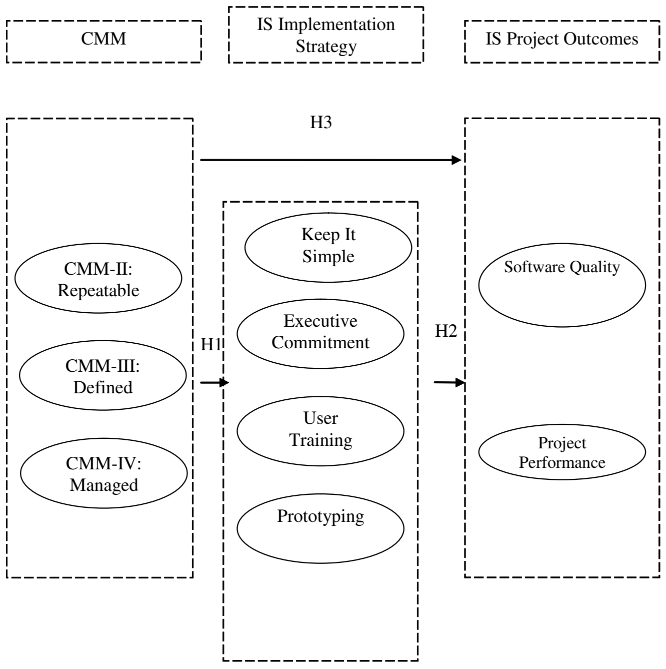

Fig. 1. Research model.

quality and project performance. So, our model argues that appropriate CMM level or maturity is needed to support suitable IS implementation strategies that would result in corresponding software quality and project performance.

The effect of CMM levels on several information systems implementation variables such as training and executive commitment is studied in Powell, 1995; Parzinger and Nath, 2000; Isaac et al., 2004. These studies show empirically that CMM certification has a positive effect on these attributes like training and top management commitment leading us to propose hypothesis H1.

H1: Organizations in different levels of CMM (i.e., CMMII, CMMIII, and CMMIV) adopt different IS implementation strategies (keep it simple, executive commitment, training, and prototyping) for IS project implementation.

Complex systems inherently present unique risks due to tightly linked interdependencies of business processes, relational databases, and process re-engineering. Companies need to have an understanding of such risks when planning and conducting assurance engagements of the reliability of these complex computer systems (Wright and Wright, 2002). *Keep it simple* should have a positive effect on project performance. It can be argued that simple software would have less schedule slippage and would be within budget. *Training* also helps to improve employee participation and involvement in quality programs through propagation of priorities and missions of the organization (Harel and Tzafrir, 1999). There are many facets to the root cause of software project failure, but two of the primary causes are poor planning, and a lack of training, especially in quality assurance (Zimmerman, 2001). Training is expected to have a positive effect on project performance. *Executive commitment* is a critical factor that influences software quality (Parzinger and Nath, 2000). *Prototyping* is recommended for clarity in understanding system requirements and in planning systems architecture (Boehm and Papaccio, 1988) and thus can help in improving software quality. Prototyping could also be effectively used in software risk management as shown in the spiral model (Boehm, 1988) and can help in improving project performance. Hence, we propose hypothesis H2.

H2: IS implementation strategies (keep it simple, executive commitment, training, and prototyping) have a significant impact on IS project outcomes as measured by software quality and project performance.

Finally, the effect of CMM levels on software quality and project performance variables is studied in Harter et al., 2000 and is the basis for hypothesis 3. Harter et al., 2000 empirically show that CMM has a positive effect on software quality and project performance. Specifically, Harter et al., 2000 conclude that higher levels of process maturity (CMM levels) is associated with higher product quality. Their findings also point to higher CMM levels resulting in improved project performance in the form of reduced cycle time and development effort. Herbsleb et al., 1997 in their study on assessment of CMM levels, show that firms with increased maturity levels had "excellent" or "good" ratings in number of project performance dimensions such as ability to meet schedules, ability to stay within budgets, etc. Firms with higher CMM levels also got "excellent" or "good" ratings in product quality. Lawlis et al., 1995 also show improved project performance to be associated with higher CMM levels. Krishnan, 1996 also shows that higher CMM levels are associated with improved software quality. Hence, we propose hypothesis H3.

H3: Organizations in different levels of CMM (CMM II, CMMIII, and CMM IV) exhibit different levels of IS project outcomes as measured by software quality and project performance.

4. Research method

4.1. Sample

Questionnaires were mailed to 1000 randomly selected IEEE Computer Society members with an expressed interest in software engineering. These members are familiar with the software development activities and, thus, are appropriate for this study. Postage-paid envelopes for each questionnaire were enclosed. All the respondents were assured that their responses would be kept confidential. Of the 1000 initial surveys mailed in the spring of 2001, total of 153 responses were received. In order to increase the sample size, a second mailing was conducted in the summer of 2001. The response from both samples totaled 212, for an overall response rate of 21.2%. Non-response bias occurs when the opinions and perceptions of the survey respondents do not accurately represent the overall sample to whom the survey was sent. One test for non-response bias is to compare the demographics of early versus late respondents to the survey (Armstrong and Overton, 1977). T-tests were computed on the means of key demographics (work experience, gender, recent project duration, and team sizes) to examine whether significant differences existed between early and later respondents. No significant difference was found; therefore, the two rounds of respondents were combined for further analysis.

The respondents consisted of software managers (26 percent), project leaders (33 percent), software professionals (35 percent), and others (5 percent). About 33 percent worked in companies that had average size of 8–15 persons or more in a project team. The sample included employees in firms from manufacturing (37 percent), service (52 percent), and education (6 percent) industries. Demographic features of the sample population are in Table 1.

Table 1
Demographics information

1. Gender	
Male	189
Female	23
2. Position	
IS Manager	55
Project Leader	71
IS Professional	75
3. The industry type of your company	
Service	110
Manufacturing	78
Education	12
4. Average IS project duration in your organization	
1 years and under	85
1–2 years	75
2–5 years	34
>5 years	11
5. Average size of IS project teams in your organization	
7 and under	113
8–15	69
Over 15	26

4.2. Constructs, reliability, and validity

CMMII – repeatable, CMMIII – defined, and CMM IV – managed: The instrument used in this study was developed by Delkleva and Drehmer (1997), based upon the practices recommended by Software Engineering Institute, to describe the activities of management controls and reviews during software development. The questionnaire asked respondents' the practices and processes that is typically in place to address information systems in their organizations. The specific items are listed in Table 2. Each item was scored using a five-point scale ranging from not at all (1) to always (5). All items were presented such that the greater the score, the greater the extent of the particular item employed.

Implementation strategies: It describes the extent to which a particular IS implementation method was adopted during system development. The instrument used in this study was developed by Alter and Ginzberg (1979) and was used by Lyytinen et al. (1998). Three items were used for each strategy except prototyping (see Table 2 for details). Each item was scored using a five-point scale ranging from not at all (1) to great extent (5). All items were presented such that the greater the score, the greater the extent of the particular item being used during system development.

Software quality: There are four items used in this study for measuring software operation quality previously defined by Nidumolu (1995). They are: (1) reliable software, (2) efficient cost of software operations, (3) wide range of outputs that can be generated and queries that can be answered and (4) overall responsive software to users. Each item was scored using a five-point scale ranging from total disagree (1) to total agree (5). All items were presented such that the greater the score, the greater the extent of the particular item occurring in systems development.

Project performance: Most authors agree on three dimensions of project performance: meeting budget, meeting schedule, and meeting user requirements (McFarlan, 1981). Other researchers suggest further dimensions of project performance: amount of work produced, the quality of work produced and ability to meet project goals (Deephouse et al., 1995-1996; Henderson and Lee, 1992; Jones and Harrison, 1996). The project management literature defines project team success as meeting project goals, budget, schedule, and operational efficiency considerations (Jones and Harrison, 1996). The items used in this study were adopted from Henderson and Lee (1992). These items were also used by Robey et al., 1993. The questionnaire asks respondents' satisfaction of the project team performance that is typical for their organization when developing information systems. The specific items are listed in Table 2. Each item was scored using a five-point scale ranging from never (1) to always (5). All items were presented such that the greater the score, the greater the satisfaction of the particular item.

Structured equation modeling (SEM) with partial least squares (PLS) analysis allows empirical assessment of the constructs used in this study (Chin, 1988; Löhmoller, 1988). Using ordinary least squares, PLS performs an iterative set of factor analysis and applies a bootstrap approach to estimate the significance (*t*-values) of the items. In this study, PLS-Graph Version 3.01 (Chin, 1994) was used to verify the measurement. Construct validity is examined in the measurement model. Individual item reliability can be examined by observing the factor loading of each item. A high loading implies that the shared variance between constructs and its measurement is higher than the error variance (Hulland, 1999). A factor loading higher than 0.7 can be viewed as high reliability and a factor loading less than 0.5 should be dropped. In Table 2, the loading of all indicators are all above 0.5 (except the two items in the CMMV which were eliminated for further analysis), which indicates the measurement is acceptable, and significant.

Convergent validity should be assured when multiple indicators are used to measure one construct. Convergent validity can be examined by the composite reliability of constructs (>.07 is recommended) (Fornell and Larcker, 1981; Kerlinger, 1986). Table 2 shows that these and other convergent validity measures. Average variance extracted (AVE), proposed by Fornell and Larcker (1981), reflects the variance captured by the indicators. For adequate discriminant validity, the square root of AVE should be greater than the correlations of the constructs. The descriptive statistics in Table 3 show this condition to hold. All the constructs are reliable and valid leading us to conduct data analysis for testing the hypotheses.

Table 2
Measurement model – confirmatory factor analysis results

Construct indicators	Loadings	*t*-statistics	Item-construct correlation
CMM II: Repeatable (*Reliability: .91*)			
To what extent does the software quality assurance function have a management reporting channel separate from the software development project management?	0.59	6.31	0.69
To what extent is there a software configuration control function for each project that involves software development?	0.51	5.56	0.67
To what extent is a formal procedure used in the management review of each software development prior to making contractual commitments?	0.57	5.11	0.79
To what extent is a formal procedure used to make estimates of software size?	0.83	13.44	0.81
To what extent is a formal procedure used to produce software development schedules?	0.71	9.46	0.83
To what extent are formal procedures applied to estimating software development costs?	0.80	10.89	0.83
To what extent are profiles of software size maintained for each software configuration item over time?	0.68	8.25	0.69
To what extent are statistics on software design errors gathered?	0.71	9.24	0.77
To what extent are statistics on software code and test errors gathered?	0.59	6.48	0.75
To what extent do software development first-line managers sign off on their schedules and cost estimates?	0.73	7.74	0.74
To what extent is a mechanism used for controlling changes to software requirements?	0.66	7.02	0.72
To what extent is a mechanism used for controlling changes to the code? (Who can make changes and under what circumstances?)	0.78	10.50	0.66
CMM III: Defined (reliability: .95)			
To what extent is there a software engineering process group function?	0.75	23.39	0.76
To what extent is there a required software engineering training program for software developer?	0.77	25.02	0.77
To what extent is a formal training program required for design and code review leader?	0.72	21.32	0.73
To what extent does the software organization use a standardized software development process?	0.82	31.22	0.83
To what extent does the software organization use a standardized and documented software development process on each project?	0.83	33.00	0.83
To what extent does senior management have a mechanism for the regular review of the status of software development projects?	0.71	19.20	0.70
To what extent are the action items resulting from design reviews tracked to closure?	0.78	28.05	0.78
To what extent are the action items resulting from code reviews tracked to closure?	0.81	29.38	0.81
To what extent is a mechanism used for ensuring compliance with the software engineering standards?	0.85	43.29	0.85
To what extent are internal software design reviews conducted?	0.79	29.12	0.78
To what extent is a mechanism used for controlling changes to software design?	0.82	24.19	0.82
To what extent are software code reviews conducted?	0.80	31.16	0.79
To what extent is a mechanism used for verifying that the samples examined by software quality assurance are truly representative of the work performed?	0.73	22.78	0.75
To what extent is there a mechanism for assuring the adequacy of regression testing?	0.64	13.86	0.65
CMM IV: Managed (Reliability: .91)			
To what extent has a managed and controlled process database been established for process metrics data across all projects?	0.65	7.99	0.79
To what extent is a mechanism used for managing and supporting the introduction of new technologies?	0.82	11.40	0.72
To what extent is test coverage measured and recorded for each phase of functional testing?	0.65	7.60	0.75
To what extent is review efficiency analyzed for each project?	0.73	9.67	0.86

Table 2 (*continued*)

Construct indicators	Loadings	*t*-statistics	Item-construct correlation
To what extent are the review data gathered during design reviews analyzed?	0.80	12.95	0.84
To what extent are code and test errors projected and compared to actual?	0.75	10.24	0.80
To what extent are analyses of errors conducted to determine their process related causes?	0.76	11.12	0.79
To what extent are code review standards applied?	0.74	9.41	0.75
To what extent is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?	0.72	9.57	0.76
To what extent are design and code review coverage measured and recorded?	0.55	5.70	0.73
To what extent is the error data from code reviews and tests analyzed to determine the likely distribution and characteristics of the errors remaining in the product?	0.36	3.41	0.64
To what extent are design errors projected and compared to actual?	0.47	4.97	0.73
Simplicity: Keep it simple (reliability: .75)			
Keep the system simple	0.96	29.83	0.79
Hide complexity	0.58	4.47	0.78
Avoid changes	0.55	4.19	0.72
Executive participation and commitment (Reliability: .88)			
Obtain user participation	0.80	13.98	0.89
Obtain user commitment	0.91	22.08	0.90
Obtain management support	0.80	11.31	0.75
Training: User training/assistance (reliability: .80)			
Providing training to users	0.75	7.16	0.80
Providing on-going assistance	0.71	6.86	0.78
Tailor system to people's capabilities	0.79	10.22	0.71
Prototyping: divide the project into manageable pieces (reliability: .84)			
Use a prototype approach for system development	0.94	17.78	0.86
Use an evolutionary or spiral approach	0.76	9.08	0.87
Software quality (reliability: .89)			
Reliable software	0.89	19.23	0.85
Efficient cost of software operations	0.89	23.05	0.85
Wide range of outputs that can be generated and queries that can be answered	0.63	6.41	0.79
Overall responsive software to users	0.83	12.71	0.87
Project performance: (reliability: .89)			
Ability to meet project goals	0.80	11.90	0.79
Adherence to schedule	0.86	14.01	0.89
Adherence to budget	0.89	18.60	0.88

Table 3
Correlations matrix

Variables	Correlation					
	Si	Su	Eu	Pr	Op	Pp
(Si) Simple	**0.75**					
(Su) Executive commitment	0.34	**0.84**				
(Eu) Training	0.35	0.47	**0.75**			
(Pr) Prototyping	0.40	0.42	0.37	**0.85**		
(Op) Quality	0.37	0.51	0.54	0.46	**0.82**	
(Pp) Project performance	0.38	0.49	0.36	0.41	0.61	**0.85**

Note: Diagonals were the square root of average variance extracted (AVE).

5. Data analysis and results

Delkleva and Drehmer (1997) found that some organizations classified as lower levels of maturity (e.g., CMMII, CMMIII) may still conduct some activities in higher levels. Hence, it is important for us to accurately and objectively ascertain the CMM levels. We use their questionnaire to obtain information on maturity levels. Based upon Delkleva and Drehmer's findings, a CMMII organization will have some activities in CMMIII and/or CMMIV. Furthermore, organizations in CMMIII (or higher levels) will have activities in lower levels (e.g., CMMII). Therefore, we use the information from the questionnaire for the CMM classification. We use cluster analysis as an objective measure to accomplish this classification. We describe the cluster analysis followed by the test of our hypotheses and results.

5.1. Cluster analysis

In order to test the proposed hypotheses, a cluster analysis was first conducted to classify the organizations into either CMM II, CMM III, or CMM IV. A total of 38 items, obtained from Delkleva and Drehmer (1997), was used to collect CMM related data. These 38 items can be separated into three different groups and each group represents different activities that would be conducted in different levels of CMM (the first 12 for CMM II, 13–26 for CMM III, and 27–38 for CMM IV). Three final scores were obtained by averaging items in each group and then were entered into SPSS for *k*-means cluster analysis. The clustered result is shown in Table 4.

Organizations with high CMM II, CMM III, and CMM IV score were considered in CMM level IV. Organizations with high CMM II and CMM III scores but low CMM IV score were considered in CMM level III. Finally, organizations with only related high CMM II score were considered in CMM level II. In total 205 projects, 77 of them were in CMM level II, 83 of them in CMM level III, and the rest 45 were counted as CMM level IV. The cluster analysis also resulted in three groups similar to the CMM levels in Paulk et al., 1993a. These three groups are identical to CMMII, CMMIII, and CMMIV based upon the criteria mentioned in Delkleva and Drehmer, 1997. We believe that the cluster analysis provides useful empirical validation and support to the CMM classification of maturity levels.

5.2. Hypotheses testing

5.2.1. Analysis of variance (ANOVA)

An ANOVA was used to test H1, whether organizations in different CMM levels conduct different IS implementation strategies, and H3, whether organizations in different CMM levels have different IS project outcomes. The test result is shown in Table 5. As can be seen from the mean differences between the CMM levels, hypotheses 1 and 3 are supported. It can be seen from the results that CMM level IV has significantly different IS implementation strategies and IS project outcomes than either CMM level II or III. CMM level III has significantly better IS project outcomes than CMM level II. However, CMM level III has significantly better IS implementation strategy in only two IS implementation factors – keep it simple and prototyping.

5.2.2. Regression

Two regression models, with project performance and software quality as dependent variables were conducted to test H2, whether there is a significant relationship between IS implementation strategy and IS project outcome. The test result is shown in Table 6. Executive commitment and prototyping strategies have a significant impact on both software quality and project performance. Training has a significant effect only on software quality, while keep it simple has a significant effect only on project performance. So, hypothesis 2 is supported.

Table 5
ANOVA analysis and post hoc comparison (H1 and H3 results)

Dependent variables	ANOVA		Post hoc		
	F value	Sig.	CMM II–III	CMM III–IV	CMM II–IV
Simple	9.698	0.000***	(–)**	(–)***	(–)***
Executive commitment	18.037	0.000***	(–)	(–)***	(–)***
Prototyping	25.794	0.000***	(–)***	(–)***	(–)***
Training	10.401	0.000***	(–)	(–)***	(–)***
Software quality (H3)	26.126	0.000***	(–)**	(–)***	(–)***
Project performance (H3)	23.372	0.000***	(–)***	(–)***	(–)***

** $p < 0.05$.
*** $p < 0.01$.

Table 4
Number of cases, mean, and standard deviation

Group	Number of cases	Mean (Std)					
		Simple	Executive commitment	Training	Prototyping	Quality	Project performance
CMM II	77	3.01 (0.71)	3.31 (0.72)	3.26 (0.70)	3.33 (0.76)	3.46 (0.62)	3.44 (0.74)
CMM III	83	2.70 (0.72)	3.15 (0.90)	3.18 (0.79)	2.81 (0.98)	3.24 (0.74)	3.09 (0.79)
CMM IV	45	3.28 (0.75)	4.03 (0.80)	3.77 (0.60)	3.89 (0.59)	4.12 (0.54)	4.02 (0.55)
Total	205	2.95 (0.75)	3.41 (0.87)	3.34 (0.75)	3.26 (0.91)	3.52 (0.73)	3.44 (0.80)

Table 6
H2 testing results

	Quality	Project performance
Simple	0.120	0.199***
Exec. Commitment	0.253***	0.313***
	−0.003	0.019
Training	0.319***	0.073
Prototyping	0.202***	0.186**

** $p < 0.05$.
*** $p < 0.01$.

6. Conclusion

Card, 2004 argues for the need for more academic research in software process improvement methods such as the CMM. The primary contribution of our study is to present and empirically test a research model that examines the effect of software process maturity (CMM) in the selection of certain critical IS implementation strategies and how CMM and the IS implementation strategies impact software quality and project performance. In a unified framework, we thus provide empirical confirmation of prior findings on the impact of CMM and provide additional insights for researchers and practitioners on the relationships among all of these factors.

Our research study findings are that CMM levels do have different impacts on IS implementation strategies, software quality, and software project performance. Higher CMM levels are associated with differing IS implementation strategies and improved software quality and project performance. We also conclude that certain IS implementation strategies have a significant impact on software quality and project performance.

The first hypothesis states that different CMM levels lead to different IS implementation strategies. The expectation is that higher levels of CMM or process maturity would be associated with different IS implementation strategies. Our results confirm this hypothesis. This result is in agreement with the finding from another study that states "CMM certified firms have been found to have better management practices and higher operational performance" (Isaac et al., 2004).

The second hypothesis studies the effect of IS implementation strategies on IS project outcomes. Here, we have a mixed result. While executive commitment and prototyping strategies have a significant impact on both software quality and project performance, training had a significant effect only on software quality while "keep it simple" has a significant effect only on project performance. Executive commitment is shown to be a critical factor impacting software quality and project performance (Parzinger and Nath, 2000; Isaac et al., 2004) and is also confirmed in our study. Prototyping strategy is expected to impact software quality and project performance based on work by (Boehm, 1988; Boehm and Papaccio, 1988) and our study provides empirical confirmation. We would argue that training should not have a significant effect on project completion time, schedule, etc. as training can be easily scheduled in parallel and should not hinder project performance. Training is known to influence software quality (Parzinger and Nath, 2000) and is confirmed to have an effect on quality in our study also. Keep it simple is a conscious systems design and project management decision that we would expect to have a significant effect on project performance. Simplicity, being an execution or task driven strategy in the building of systems, may not have a bearing on software quality per se. So, "keep it simple" has a significant effect only on project performance in our study.

Finally, in hypothesis 3, we propose that different levels of CMM are associated with different IS project outcomes. The expectation is that higher levels of CMM or process maturity would be associated with better software quality and project performance. Our results confirm this hypothesis. This result also confirms Harter et al., 2000's finding that higher levels of CMM are associated with improved software quality and improved project performance such as reduced cycle time and development effort.

One limitation of this study could be that it focuses primarily on CMM as the software process improvement methodology. Other methods such as Total Quality Management, International Standards Organization (ISO) Quality Certification, and Six Sigma could also be considered in empirical research. Another limitation of this study is that the study's findings apply only to the four IS implementation strategies that we have empirically examined. There could be other IS implementation strategies that could impact software quality and project performance. Further, the cross sectional design of this study places restrictions on the direction of causality. This, however, makes sense as maturity may be an enabler of methods; management may also strategically strive to increase maturity based on project outcomes.

Future research needs to further examine the benefits of improved software quality and project performance as organizations go to CMM level 3. From a practice standpoint, it is important for future studies to examine whether organizations which reach higher levels in the CMM model have differing abilities than those at lower levels, being able to respond to greater complexities through a greater variety of strategies and diverse interests of the various stakeholders. It is also important for future research to study the time taken to move from one level of CMM to the next.

In addition, future studies of the effect of total quality management, International Standards Organization (ISO) Quality Certification, and Six Sigma on software quality and IS project performance are needed. While there are quite a number of studies on CMM, it is important to examine software process improvement more broadly as it is applied using different methodologies. In this study, we examine four IS implementation strategies and their impact on IS project outcomes. Future research needs to examine the linkage between other IS implementation strategies and IS project outcomes. Other dimensions of project outcomes also need to be examined in future research.

References

Ahire, S.L., 1996. An empirical investigation of quality management in small firms. Production and Inventory Management Journal 2, 44–50.

Alter, S., 1979. Implementation risk analysis. TIMS Studies in the Management Sciences 13, 103–119.

Alter, S., Ginzberg, M., 1979. Managing uncertainty in MIS implementation. Sloan Management Review, 23–31.

Armstrong, J.S., Overton, T.S., 1977. Estimating non-response bias in mail survey. Journal of Marketing Research 15, 396–402.

Arthur, L.J., 1993. Improving Software Quality: An Insider's Guide to TQM. John Wiley, New York.

Ashrafi, N., 1998. A decision-making framework for software total quality management. International Journal of Technology Management 16, 532–543.

Bailey, J., 1999. Trash haulers are taking fancy software to the dump. Wall Street Journal June 9.

Ben-Menachem, M.B., Marliss, G.S., 1997. Software quality: Producing Practical, Consistent Software. International Thomson Computer Press, New York.

Boehm, B., 1988. A spiral model for software development and enhancement. IEEE Computer 21 (5), 61–72.

Boehm, B.W., Papaccio, P.N., 1988. Understanding and controlling software costs. IEEE Transactions on Software Engineering 14 (10), 1462–1477.

Boehm, B, Jain, A, 2005. Value-Based Systems Engineering Theory, CSER 2005 Invited Address, March 23.

Card, D.N., 1991. Understanding Process Improvement. IEEE Software (July), 102–103.

Card, D.N., 2004. Research directions in software process improvement. In: Proceedings of the 28th Annual International Computer Software and Applications Conference. IEEE.

Chin, W.W., 1988. The Partial Least Squares Approach to Structural Equation Model. Lawrence Erlbaum Associates, Mahwah, NJ.

Chin, W.W., 1994. PLS-Graph Manual Version 2.7. University of Calgary, Calgary, AB.

Cho, C.K., 1998. Quality programming. John Wiley & Sons, United Kingdom.

Deephouse, C., Mukhopadhyay, T., Goldenson, D.R., Kellner, M.I., 1995-1996. Software processes and project performance. Journal of Management Information Systems 12 (3), 187–205.

Delkleva, S., Drehmer, D., 1997. Measuring software engineering evolution: a rash calibration. Information Systems Research 8 (1), 95–104.

Fornell, C., Larcker, D.F., 1981. Structural equation models with unobservable variables and measurement errors: algebra and statistics. Journal of Marketing Research 18 (2), 39–50.

Gong, B., Yen, D.C., Chou, D.C., 1998. A manager's guide to total quality design. Industrial Management & Data Systems 8, 100–107.

Hansen, B., Rose, J., Tjornehoj, G., 2004. Prescription, description, reflection: the shape of the software process improvement field. International Journal of Information Management 24 (6), 457–472.

Harel, G.H., Tzafrir, S.S., 1999. The effect of human resource management practices on the perceptions of organizational and market performance of the firm. Human Resource Management 38 (3), 185–199.

Harter, D.E., Krishnan, M.S., Slaughter, S.A., 2000. Effects of process maturity on quality, cycle time, and effort in software projects. Management Science April 46 (4), 451.

Henderson, J.C., Lee, A.S., 1992. Managing I/S design teams: a control theories perspective. Management Science 38 (6), 757–777.

Herbsleb, J., Zubrow, D., Goldenson, D., Paulk, M., 1997. Software quality and the capability maturity model. Communications of the ACM 40 (6), 30–40.

Hulland, J., 1999. Use of partial lest squares (PLS) in strategic management research: a review of four recent studies. Strategic Management Journal 20, 195–204.

Humphrey, W.S., 1988. Characterizing the software process. IEEE Software 5, 73–79.

Isaac, G., Rajendran, C., Anantharaman, R.N., 2004. Significance of quality certification: the case of the software industry in India. The Quality Management Journal 11 (1), 8.

Jalote, P., 2000. CMM in Practice. Addison-Wesley, Reading, MA.

Jones, C.R., 1998. Customer focused performance improvement: developing a strategy for total quality. International Journal of Technology Management 16, 494–504.

Jones, M.C., Harrison, A.W., 1996. IS project team performance: an empirical assessment. Information & Management 31, 57–65.

Jorgensen, M., 1999. Software quality measurement. Advances in Engineering Software 30, 907–912.

Kanji, G.K., 1996. Implementation and pitfalls of total quality management. Total Quality Management 7, 331–343.

Kanji, G.K., Sa, P.M., 2002. Kanji's business scorecard. Total Quality Management 13, 13–27.

Kerlinger, F.N., 1986. Foundations of Behavioral Research. Holt, Rinehart and Winston, New York.

Krishnan, M.S., 1996. Cost and Quality Considerations in Software Product Management, Dissertation, Graduate School of Industrial Administration. Carnegie Mellon University, 1996.

Lawlis, P.K., Flowe, R.M., Thordahl, J.B. 1995. A Correlational Study of the CMM and Software Development Performance, Crosstalk, September, p. 21–25.

Li, E.Y., Chen, H.G., Cheung, W., 2000. Total quality management in software development process. The Journal of Quality Assurance Institute 4, 5–41.

Löhmoller, J., 1988. Latent Variable Path Modelling with Partial Least Squares. Physica-Verlag, Heidelberg.

Lyytinen, K., Mathiassen, L., Ropponen, J., 1998. Attention shaping and software risk – a categorical analysis of four classical risk management approaches. Information Systems Research 9 (3), 233–255.

McConnell, S., 2002. The business of software improvement. IEEE Software (July/August), 5–7.

McFarlan, F.W., 1981. Portfolio approach to information systems. Harvard Business Review 59, 142–150.

Milakovich, E.M., 1995. Improving Service Quality. St. Lucie Press, Delray Beach, FL.

Murugappan, M., Keeni, G., 2003. Blending CMM and six sigma to meet business goals. IEEE Software, 42–48.

Nidumolu, S.R., 1995. The effect of coordination and uncertainty on software project performance: residual performance risk as an intervening variable. Information Systems Research 6 (3), 191–219.

Parzinger, M.J., Nath, R., 1998. TQM implementation factors for software development: an empirical study. Software Quality Journal 7, 239–260.

Parzinger, M.J., Nath, R., 2000. A study of the relationships between total quality management implementation factors and software quality. Total Quality Management 11 (3), 353–371.

Paulk, M.C., Curtis, B., Chrisis, M.B., Weber, C.V., 1993a. Capability Maturity Model for Software, version 1.1, CMU/SEI-93-TR-24, February, Software Engineering Institute.

Paulk, M.C., Weber, C.V., Garcia, S., Chrissis, M.B., Bush, M., I 993b. Key practices of the Capability Maturity Model, version 1. 1, CMU/SEI-93-TR-25, February, Software Engineering Institute.

Powell, T.C., 1995. Total quality management as a competitive advantage: a review and empirical study. Strategic Management Journal 16, 15–37.

Prahalad, C.K., Krishnan, M.S., 1999. The new meaning of quality in the information age. Harvard Business Review (September–October), 109–118.

Pressman, R.S., 2004. Software Engineering: A Practitioner's Approach. McGraw-Hill, New York.

Rajagopal, P., Frank, T., 2002. Oracle ERP and network computing architecture: implementation and performance. Information Systems Management 19 (2), 53–68.

Robey, D., Smith, L.A., Vijayasarathy, L.R., 1993. Perceptions of conflict and success in information system development projects. Journal of Management Information Systems 10 (1), 123–139.

Scheer, A., Habermann, F., 2000. Making ERP a success. Communications of the ACM (April), 57–61.

Shrednick, H.R., Shutt, R.J., Weiss, M., 1992. Empowerment: key to IS world-class quality. MIS Quarterly 16, 491–505.

Subramanian, G.H., Hoffer, C.S., 2005. An exploratory case study of enterprise resource planning implementation. The International Journal of Enterprise Information Systems (January–March).

Vitharana, P., Mone, M.A., 1998. Critical factors of software quality management. In: Proceedings of the Association of Information Systems, Baltimore, MD, pp. 906–908.

Wright, S., Wright, A., 2002. Information system assurance for enterprise resource planning systems: unique risk considerations. Journal of Information Systems 16 (1), 99–113.

Wynekoop, J.L., WaIz, D.B., 2000. Investigating traits of top performing software developers. Information Technology & People 13, 186–195.

Yang, Y.H., 2001. Software quality management and ISO 9000 implementation. Industrial Management & Data Systems 101, 329–336.

Yourdon, E., 1992. Decline and Fall of the American Programmer. Yourdon Press, Englewood Cliffs, NJ.

Zimmerman, L. V, 2001. Plan for training to ensure software quality. AACE International Transactions. p. IT51.

Girish H. Subramanian is an associate professor of Information Systems in the School of Business at Penn State Harrisburg. He obtained his Ph.D. in Computer and Information Systems from Temple University. His work has appeared in *Communications of the ACM, Decision Sciences, Journal of Management Information Systems, Journal of Systems & Software, IEEE Transactions on Software Engineering* and other journals.

James Jiang is professor of Management Information Systems at the University of Central Florida, US. He is also the honorary Jin-Ding Professor at the National Central University in Taiwan. He was visiting research professor in Tshing-Hwa University, National Taiwan University, and City University of Hong-Kong. He obtained his masters degrees in Applied Mathematics, Computer Sciences, and Management Sciences. His Ph.D. in Information Systems was granted by the University of Cincinnati. His research interests include IS project management, IS human resources management, and IS service quality management. He teaches object-oriented design and implementation and IS project management. He has published over 100 academic articles in these areas.

Gary Klein is the Couger Professor of Information Systems at the University of Colorado in Colorado Springs. He obtained his Ph.D. in Management Science from Purdue University. Before that time, he served with the company now known as Accenture in Kansas City and was director of the Information Systems Department for a regional financial institution. His research interests include project management, technology transfer, and mathematical modeling with over 100 academic publications in these areas. He teaches programming, project management, statistics, management science, and knowledge management courses. In addition to being an active participant in international conferences, he has made professional presentations on Decision Support Systems in the US and Japan where he once served as a guest professor to Kwansei Gakuin University. He is an active member of the Institute of Electrical and Electronic Engineers, the Association for Computing Machinery, the Decision Sciences Institute, INFORMS, the American Society for the Advancement of Project Management, and the Project Management Institute.