Journal of International Information Management

Volume 11 | Issue 1 Article 4

2002

Software project process management maturity and project performance: An examination of Taiwan's software companies

Melody Wu
National Huwie Institute of Technology

Hsin-Ginn Hwang
National Chung Cheng University

Houn-Gee Chen
Tsing-Hua University

James J. Jiang
University of Central Florida

Software project process management maturity and project performance: An examination of Taiwan's software companies

Melody Wu
National Huwei Institute of Technology

Hsin-Ginn Hwang
National Chung Cheng University

Houn-Gee Chen
Tsing-Hua University

James J. Jiang
University of Central Florida

ABSTRACT

Researchers and practitioners argue that an inadequate software development process is one critical factor accounting for high project failure rates. As a result, the Capability Maturity Model (CMM) was introduced by the Software Engineering Institute as a guideline for advancing project maturity and improving the odds of project success. To investigate the effectiveness of applying the principles of the CMM, a survey was conducted of 196 Information System managers in Taiwan. The results indicate that a more mature software development process reduces the extent of certain risks experienced during the project development and enables better project performance. Managerial implications regarding the CMM are described.

INTRODUCTION

Due to the high software development failure rate and low productivity in the software industry, the Taiwan government in recent years has endeavored to strengthen organizational software development structure. For example, in 1992, the Software Industry Five-Year Development Plan was proposed by the Industry Development Bureau (IDB) to help software development organizations in Taiwan improve their software development capability. To accomplish this goal, a series of systematic lectures, professional conferences, and technical training courses were provided to software organizations.

During the past decade, software investment has grown rapidly worldwide and software project development has become one of the most important targets for many industrial and research initiatives. As information technologies evolve and information system (IS) applications grow in size, complexity, and importance, a solution for improving the software project process has become more and more imperative. Many tools, technologies, and management methods, such as case tools and rapid application development (RAD), have been adopted to guide the management of the software development process over the years (Bandinelli & Fuggetta, 1995; Kuilboer & Ashrafi, 2000).

Even though significant effort and resources have been poured into system development tools, IS project success rates are still low. In large IS development projects, more than 80 percent are excessively late and/or over budget. According to Standish Group International (1995), about 15% of all IS developments never deliver a final product and budget overruns of 100 to 200% are common in IS projects. A follow-up to their study estimates that almost 80,000 projects were cancelled in 1995 (Standish Group, 1996). IS project problems cost U. S. companies and government agencies $145 billion annually. Clearly, there is major concern over software project development as difficulties are continually being experienced despite advances in methodologies.

Faced with a high failure rate in software projects, IS managers in many organizations are pursuing software process improvements (Deephouse, et al., 1996; Necco, Gordon, & Tsai, 1987; Rivichandran & Rai, 2000). A software process is a set of activities, methods, practices, and transformations that people use to develop and maintain software and associated products (e.g., design documents, code, test cases, and user manuals). In fact, many IS projects are carried out in an ad hoc fashion without adequate planning, with poor explication of the overall development process, and the lack of a well-established management framework (Rai & Al-Hindi, 2000).

To provide guidelines for IS management to better control the project development process, the Software Engineering Institute (SEI), in collaboration with the U. S. Department of Defense (DoD) and Mitre Corp., recommends a number of key software process improvement (SPI) areas. These are formalized into an evaluative framework called the Capability Maturity Model (CMM) (Paulk, et al., 1993). Five levels of maturity are identified in the CMM maturity model. The levels range from an initial level to an optimizing level. Initially, success of software projects relies on the skills of individual project managers. At the optimizing level, Priority is given to monitoring software processes and continuous process improvement. The intermediate levels represent levels of activity between the two.

Judging by its acceptance in the software industry, the proposed CMM is already a major success. It has spread far beyond its origins in military applications and is now used by thousands of major organizations in and out of the U.S. (e.g., Australia, Europe, Taiwan, Korea, and South America). The resources expended on the CMM model amount to billions of dollars every year in the U. S. alone (Herbsleb, Zubrow, Goldenson, Hayes, & Paulk, 1997). Case studies of CMM-based implementations in a number of organizations report success; but there are an unknown number of unreported failures. CMM implementation requires tremendous resources, so it is surprising that little empirical evidence of the impact of CMM implementation on organizations can be found in literature. Furthermore, in spite of its wide acceptance outside the U.S., even

more limited empirical information about non-U.S. organizations is reported. After all, the CMM was originally designed with U.S organizational contexts in mind, so applicability to other cultures is an issue that needs to be addressed.

One exception is a study regarding the CMM and software development in the Taiwan software industry. Chen (1999) indicates that the software process maturity level in Taiwan shows an improvement over 3 years (1995-1998). He also finds that the attitude of management is positively correlated with software process maturity implementation. A more recent study (Chen, 2000) examines the relationship of total quality management (TQM) and software process maturity implementation. This study finds that software process maturity in Taiwan has grown to a high level. Furthermore, organizations with a quality-emphasized culture exhibit a higher level of software process management maturity. Unfortunately, the relationships among an organization's CMM level, project risk, and project performance has not been examined.

The purpose of this study is to examine the impact of the software process management maturity level on project risk and project performance in Taiwan organizations through a wide sample. Specifically, the following two questions are addressed in this study: 1) will organizations with a higher maturity level have a better project performance than those with a lower maturity level? and 2) will the organizations with a high maturity level experience a lower level of project development risk than those with a lower maturity level?

BACKGROUND

Software development techniques have been studied for quite some time and many of the techniques of merit have become part of the textbooks used in MIS curricula. Still, studies are conducted to examine improvements to the development of systems. One common approach is to study the impact a specific tool or concept has on the success of a system development (Subramanian & Rai, 1996). Total quality management techniques applied to the development process are deemed successful (Ravichandran & Rai, 2000; Swanson, McComb, Smith & McCubbery, 1991). Other methods and practices have also been studied to determine key success factors of improved development (Necco, Gordon, & Tsai, 1987; Rai & Al-Hindi, 2000). But studies of complete process activity sets, such as the CMM, are limited (Ibbs & Kwak, 2000).

The CMM was developed by the Software Engineering Institute in the 1980s for the defense sector. In September 1987, the Software Engineering Institute released a brief description of the process-maturity framework. Version 1.0 was released in 1991 and soon was adopted by the software community. Version 1.1 was released in early 1993. Since then, the model has been adopted by many organizations in the U. S. and around the world (Fitzgerald & O'Kane, 1999). The CMM ranks an organization's software project procedures on a five-point scale. The maturity levels, 1 to 5, indicate the overall effectiveness of the organization's software engineering practices. Each level represents a different stage of maturity. The five levels of the CMM model are as follows:

1. Level 1: Initial. The software processes for organizations at this maturity level are characterized as ad hoc and occasionally even chaotic. Few processes are defined, and success often depends on individual effort.
2. Level 2: Repeatable. The project management processes are established to track cost, schedule, and functionality for software projects. The necessary process discipline is in place to repeat earlier successes on similar projects.
3. Level 3: Defined. The software processes for both management and engineering activities are documented, standardized, and integrated into a standard for the entire organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.
4. Level 4: Managed. Detailed measures of the software process and product quality are collected. Both the software processes and products are quantitatively understood and controlled.
5. Level 5: Optimizing. Continuous process improvement is enabled by quantitative feedbacks from the processes and from piloting innovative ideas and technologies.

There are published case studies examining the success of CMM-based software process improvement in organizations (Humphrey, et al., 1991; Dion, 1993; Lipke & Rosenbaum, 1993; Fitzgerald & O'Kane, 1999). Herbsleb and Goldenson (1996) surveyed organizations that undertook CMM-based software process improvement and found that process maturity did result in better organizational performance. A similar study (Ibbs & Kwak, 2000) looked into the issues of project management maturity and management processes. The study indicated positive quantitative benefits of mature processes. The evidence is accumulating that CMM-based software process improvement appears to be paying off, at least for some organizations, but the evidence is still limited to a small subset of organizations and cultures.

HYPOTHESES

IS researchers have identified a number of risk factors that lead to difficulties in delivering a successful project (Barki, et al., 1993). Barki, Rivard, and Talbot (1993) summarized 12 common software development risks: relative project size; technical complexity; extent of changes; resource insufficiency; lack of development expertise in team; team's lack of expertise with task; team's lack of general expertise; lack of user experience and support; intensity of conflict; lack of clarity of role definitions; task complexity; and magnitude of potential loss. Meanwhile, risk management literature had identified a number of risk-mitigating methods (Jiang & Klein, 1999). These risk-mitigating methods often involve taking steps to enhance opportunities and develop responses to threats. Couillard (1995) suggested that technical risks can be reduced by emphasizing team support, avoiding stand-alone project structures, improving communication, and increasing project-monitoring. IS researchers expect that enhancing the software project development process can significantly reduce project risks.

The CCM-based software process represents a set of recommended practices in a number of key process areas to stabilize the software development environments and thus enhance software development capability. The suggested software process improvement areas include having competent people, establishing basic project management processes, documenting and standardizing engineering processes and organizational support, measuring and controlling product and process quality, and facilitating continuous process improvement. Thus, as an organization matures, the software process becomes better defined and is more consistently implemented throughout the organization. As a result, managers can better monitor the risk associated with the project (Paulk, Curtis, & Chrissis, 1993). Based upon the above discussion, we propose the following hypothesis:

H1: The more mature an organization's software process management, the less the extent of project risks experienced in software development.

Today, the CCM-based model has become very influential as a basis for software process improvement. The fundamental assumption is the belief that project performance (e.g., cost, schedule, and system quality) can be achieved by implementing mature practices in organizations. However, most of the evidence to date has consisted of case studies such as Hughes Aircraft (Humphrey, Synder, & Willis, 1991) Raytheon (Dion, 1993), and Tinker AFB (Butler, 1995). All these studies show that organizations with more maturity tend to have substantially higher quality, faster cycle time, and higher productivity. Herbsleb and goldenson (1996) also found evidence that software development process maturity is in fact associated with better organizational performance. Based upon the above discussion and literature review, we expect the following:

H2: The more mature an organization's software development process management, the better its software project performance.

RESEARCH METHODOLOGY

Sample

Questionnaires were mailed to 650 randomly selected Information Service Industry Association (CISA) members in Taiwan. CISA currently has more than 800 members who work as IS managers in Taiwan IT organizations and software development companies. Postage-paid, return envelopes for each questionnaire were enclosed. All the respondents were assured that their responses would be kept confidential. All mailings were sent via first class mail. Follow-up phone calls were made two weeks after the initial mailing. For those who did not respond, additional cover letters and surveys were mailed 21 days or 30 days after the initial mailing.

Of the 650 initial surveys mailed in the winter of 2000, a total of 127 responses were received. Follow-ups resulted in 82 additional responses in early 2001. The response from both samples totaled 209, for an overall response rate of 32.5%. Thirteen questionnaires were eliminated due to missing data, leaving a final sample of 196 used in the data analysis.

Non-response bias occurs when the opinions and perceptions of the survey respondents do not accurately represent the overall sample to which the survey was sent. One test for non-response bias is to compare the demographics of early versus late respondents to the survey. 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 late respondents. No significant difference was found; therefore, these two rounds of respondents were combined for further analysis. Demographic features of the sample population appear in Table 1.

Table 1. Demographics

1. Gender

Male	146
Female	44
No response	6
TOTAL	196

2. Age

30 and under	49
31-35	71
36-40	26
41-45	9
46-50	9
51 and over	1
No response	16
TOTAL	196

3. Years of Working Experience

1-5 years and under	61
6-10 years	75
11-15 years	28
16-20 years	10
21-25 years	7
26 or more	4
No response	16
TOTAL	196

4. Position

IS Manager	57
Project Lealderl	26
IS Professional	83
IS User	23
No response	7
TOTAL	196

5. The Industry Type of Your Company

Service	118
Manufacturing	68
Education	3
No response	7
TOTAL	196

6. Number of Employees

10 and under 10 employees	9
11-50 employees	25
51-100 employees	11
101-300 employees	38
300-500 employees	24
501 or more employees	73
No response	10
TOTAL	196

7. Size of IS Project Teams in your Organization

7 and under	97
8-15	59
16-25	14
26 and over	16
No response	10
TOTAL	196

8. IS Project Duration in Your Organization

1 years and under	60
1-2 years	92
2-3 years	24
3-5 years	9
6 or more years	1
No response	10
TOTAL	196

Constructs

Software process management maturity: The 38 items used to measure the software process management maturity level were adopted from Dekleva and Drehmer (1997). Items of this instrument were key processes representing the CMM repeatable (level 2), defined (level 3), and managed (level 4) maturity thresholds. Items 1-12 were CMM level 2 items, items 13-26 were level 3, and items 27-38 were level 4. Since no organization was believed to achieve the optimizing level (level 5), none of these items were used. The respondents were asked to evaluate the overall extent of each structure and procedure implemented in their organizations' IS projects. Each item was scored using a five-point scale ranging from "not at all" (1) to "extremely" (5).

Project risk: The project development risk measurement used in this study is adopted from Barki, et al. (1993). Their original instrument included a number of items in a Likert-type scale. The respondents were asked to evaluate the overall extent of difficulties arising from each risk item in their organizations' IS projects. Each scale was scored using a seven-point scale ranging from "Not at all" (1) to "extremely" (5) and averaged across all items. All the items were scored so that the greater the score, the greater the incidence of the particular item.

Project performance: The project performance construct was adopted from Nidumolu (1995). IS project performance is defined by two key aspects: (1) process performance - how well the project development process is undertaken, and (2) product performance - meeting project goals and system quality. It is important to study both aspects because a delivered project may meet business objectives and project goals but exceed time and cost projections. On the other hand, a project may meet specific time and cost with high quality but not meet with user needs or project objectives.

Although the project process management maturity construct has been examined in the literature, a factor analysis was conducted. Using principal components analysis (PCA) to determine the number of factors, and varimax rotation to determine membership, four factors emerged from the original data set. Table 2 presents the results of the factor analysis on the original 38 items. The criteria used to identify, distinguish, and interpret factors were that a given item should load 0.50 or higher on a specific factor and have a loading no higher than 0.45 on other factors, leaving 34 items. The four factors matched well with the priori structure reported. The four factors explained 64% of the total variance. The measure of sampling adequacy was computer to be 0.95 and suggested the adequacy of sample size. The Cronbach's alpha test (Cronbach, 1951) also suggested good reliability for documented standards and procedures (alpha = 0.87), process metrics (alpha = 0.90), organizational infrastructure (alpha = 0.94), and measure and analysis (alpha = 0.93). The descriptive statistics of each factor of this construct are given in Table 3a.

Similarly, a PCA was conducted to examine the project performance measure. The Keiser measure of sampling adequacy was 0.94 indicating the adequacy of the sample size. The result extracted three factors from our original data set and explained 69% of the total variance. Table 4 presents the results of the factor analysis on the 20 items of the construct. The Cronbach's alpha test also suggested good reliability for learning (alpha = 0.90), quality and operation (alpha = 0.93), and control and flexibility (alpha = 0.92). The descriptive statistics of each factor of this construct are given in Table 3b.

Table 2. CMM Factor Structure

Items	Documented Standards & Procedures	Process Metrics	Organizational Structure	Measure and Analysis	Key Word
C1					Software quality assurance
C2	.64				Configuration control
C3	.65				Formal management review
C4	.62				Size estimated
C5	.70				Software development scheduled
C6	.66				Software cost estimated
C7		.62			Profiles of software size
C8		.69			Software design errors
C9		.70			Software code and test errors
C10		.65			Managers sign off
C11			.57		Requirements change control
C12			.74		Code changes controlled
C13			.77		Software engineering process
C14			.60		Developers training required
C15		.67			Training review leaders
C16		.66			Development standardized
C17			.70		Standards documented, used
C18			.54		Senior managers review
C19			.58		Design review items tracked
C20			.56		Code review items tracked
C21			.59		Compliance with standards
C22			.64		Design reviews conducted
C23			.64		Design changes controlled
C24			.54		Code reviews conducted
C25		.48		.53	SQA sample verification
C26		.63			Adequacy of regression test
C27		.49			Process metrics database
C28					New technology intro. managed
C29				.58	Test coverage measured
C30				.57	Review efficiency analyzed
C31				.73	Design review data analyzed
C32				.68	Code, test errors projected
C33				.71	Error cause analysis
C34					Code review standards
C35				.55	Software process assessed
C36				.57	Design and code coverage
C37				.68	Forecast remaining errors
C38				.72	Design errors projected
alpha	.87	.90	.94	.93	

Note: Only loadings > .45 are shown.

Table 3. Descriptive Statistics of Examined Variables

3a: Software Process Mangement Maturity Factors

Statistics	Documented Standards & Procedures	Process Metrics	Organizational Structure	Measure and Analysis	Project Process Management Maturity
Mean	2.82	3.02	2.65	3.32	2.95
Std Deviation	.93	.95	.96	1.00	.85
Median	2.78	3.08	2.63	3.40	2.99
Skewness	-0.01	-0.14	0.19	-0.41	-0.19
Kurtosis	-0.51	-0.67	-0.80	-0.48	-0.52

3b: Project Performance Factors

Statistics	Learn	Quality and Operation	Control and Flexibility	Overall Performance
Mean	3.15	3.46	3.45	3.35
Std Deviation	.82	.94	.84	.77
Median	3.25	3.50	3.50	3.40
Skewness	-0.26	-0.33	-0.25	-0.34
Kurtosis	-0.16	0.42	-0.08	-0.15

3c: Project Risk Factors

Statistics	Team Risk	Orga. Envi. Risk	Lack of User Experience and Support Risk	Task Com-plexity Risk	User Attitude Risk	Technical Risk	Appl Size Risk	Overall Risk
Mean	2.77	2.62	3.23	2.93	3.32	3.71	3.14	3.10
Std Deviation	.92	.88	.77	.90	.95	.98	.97	.57
Median	2.69	2.67	3.27	2.88	3.43	3.75	2.60	3.14
Skewness	.29	.34	-0.28	.12	-0.32	-0.75	3.00	-0.32
Kurtosis	-0.33	-0.11	-0.03	-0.25	-0.29	-0.29	-0.70	.77

Table 4. Factor Matrix on Project Performance

Items	Learn	Quality and operation	Control and and flexibility	Key Word
Len1	.87			Use of key technologies
Len2	.89			Use of development techniques
Len3	.63			Supporting users' business
Len4	.63			Overall knowledge
Con1			.70	Effective cost control
Con2			.71	Effective schedule control
Con3			.70	Audit and control standards
Con4			.61	Overall control exercised
Qua1		.72		Complete training
Qua2		.79		Quality communication
Qua3		.80		Feelings of participation
Qua4		.77		High quality of interactions
Op1		.66		Reliable software
Op2		.62		Cost of software operations
Op3		.67		Wide range of outputs
Op4		.71		Responsive software
Flx1			.78	Cost of adapting software
Flx2			.69	Rapid adapting of software
Flx3			.73	Cost of maintaining software
Flx4			.65	Long-term flexibility
Alpha	.90	.93	.92	

Note: Only loadings > .45 are shown.

Using the same analysis techniques, seven factors emerged for project risk, accounting for 60% of the total variance. Table 5 presents the results of the factor analysis. The criteria used to identify, distinguish, and interpret factors were that a given item should load 0.50 or higher on a specific factor and have a loading no higher than 0.45 on other factors. The measure of sampling adequacy was 0.75. This analysis resulted in the elimination of 6 items that had undesirable psychometric properties, leaving 56 items in the scales. The Cronbach alpha test also suggested good reliability for team risk (alpha = 0.94), organizational environment risk (alpha = 0.92), task complexity risk (alpha = 0.91), user attitude risk (alpha = 0.86), lack of user experience and support risk (alpha = 0.90), application size risk (alpha = 0.80), and technical risk (alpha = 0.81). The descriptive statistics of each factor of this construct are given in Table 3c.

Table 5. Project Risk Factors

Items	Team Related	Organization Environment	Lack of User Experience and Support	Task Complexity	User Attitude	Technical	Application Size	Key Word
Tr1						.64		New hardware
Tr2						.62		New software
Tr3						.79		Hardware suppliers
Tr4						.75		Software suppliers
Tr5						.56		External users
As1								Team size
As2				.55				Many person-days required
As3				.49				Many months required
As4								Large budgets
As5							.77	Info. system staff
As6								Outside consultant members
As7							.72	Info. system users
As8							.69	A large number of users
As9							.61	Different level users
Te1	.81							Development methodology
Te2	.76							Development support tools
Te3	.76							Project management tools
Te4	.70							Implementation tools
Te5	.79							Application types
Te6	.81							Organizational operations
Te7	.76							Function of user department
Te8	.71							Specific application areas
Ov1								Elements and objects
Ov2	.61							Work with top management
Ov3	.64							Work effectively in a team
Ov4	.66							Complete a task
Ov5	.58							Human implications
Ov6	.57							Carry out task
Su1					.51			Negative opinion
Su2					.64			Computerized support
Su3					.76			Not enthusiastic
Su4					.76			Use of computers
Su5					.71			Changes systems entail
Su6					.72			Requirement definitions
Su7					.67			Development teams' questions
Su8					.57			Importance of users' role
Su9			.75					Tasks & life cycle stages
Su10			.71					Integral part
Su11			.79					Working tools

Table 5. Continued

Items	Team Related	Organization Environment	Lack of User Experience and Support	Task Complexity	User Attitude	Technical	Application Size	Key Word
Su12			.82					Little experience
Su13			.68					Development team requests
Su14			.61					Development responsibilities
Su15			.76					Application types
Ac1				.49				Hardware technical complexity
Ac2				.60				Software technical complexity
Ac3				.67				Database technical complexity
Ac4				.61				Link to existing system
Ac5				.60				Link to all systems
Oe1								Modify organizational tasks
Oe2								Organizational major changes
Oe3				.61				Many person-days consumed
Oe4				.70				Many months consumed
Oe5				.68				Many dollars consumed
Oe6		.66						Frequent conflicts
Oe7		.76						Serious conflicts
Oe8		.71						Unimportant matters conflicts
Oe9		.81						Frequent conflicts
Oe10		.75						Serious conflicts
Oe11		.74						Unimportant matters conflicts
Oe12		.50						Team roles undefined
Oe13		.70						Unpleasant communication
Oe14		.54						Users' role undefined
Alpha	.94	.92	.91	.86	.90	.80	.81	

Note: Only loadings > .45 are shown.

Multi-collinearity is a condition in which one or more variables exhibit very strong correlations (> .80) with one another (Anderson & Gerbing, 1988). We first examined the correlations between the software process management maturity, project performance, and project risk. The results indicate that correlations between variables were all less than .80 and, thus, none of the correlations were large enough to suggest confounding overlap in measurements.

RESULTS

To test the proposed hypotheses, two independent regression analyses were conducted (see Table 6a). The p-value (<.05) indicated that there was a significant relationship between project

risk and software process management maturity levels. Therefore, we concluded that software process management maturity levels relate negatively to project risks during system development, which gives support for hypothesis H1. Also, the p-value (<.05) indicated that there was a significant relationship between project performance and software process management maturity levels. Thereby, we concluded that software process management maturity levels relate positively to project performance, which gives support for hypothesis H2.

Table 6. Results of Regression

6a: Hypothesis Testing

Dependent Variable	Independent Variable	Coefficient
Project risk	Software Process Management Maturity	-.16*
Project performance	Software Process Management Maturity	.54*

6b: Project Risk Factors and Software Process Maturity

	Team Risk	Organizational Environment Risk	Task Complexity Risk	Users Attitude Risk	Lack of user Experience and Support Risk	Application Size Risk	Technical Risk
Documented standards and procedures	-.29*	-.26*				.31*	
Process metrics			-.22*		-.26*	-.36*	
Organizational structure				.31*	.27*		
Measure and analysis	-.37*			-.37*	-.38*		

Note: * indicated significant at p-value <.05 level.

CONCLUSION

The purpose of this study was to examine the impacts of software process management maturity on project risk and project performance. We have confirmed that software process management maturity is negatively related to project risk. To further examine this relationship, four multiple regression analyses were conducted to examine the relationship between each project maturity factor and each project risk factor, as shown in Table 6b. The results found that the various software process maturity factors were not equally associated with different project risk factors. Specifically, increased organizational structure accompanied higher risks with the users,

perhaps due to formal procedures that tend to reduce user-analyst interaction. An increase in risks due to application size accompanied greater documentation of standards and procedures, perhaps resulting from a need to provide evidence of control in larger systems. Other relations were negative, such that increased activity was associated with decreased risk. The results provide direction to target-specific risks in practice or provide research topics to investigate the presence and correction of specific links.

In addition to the significant negative relation to project risk, the software process management maturity model showed a positive link to project performance. The results indicated that software process maturity is in fact associated with better project performance and largely consistent with the studies in the literature (Dion, 1993; Lipke & Rosenbaum, 1993; Fitzgerald & O'Kane, 1999). This result empirically confirmed the case studies where organizations implementing the CMM have produced significant advances in costs, benefits, and related problems (Herbsleb et al., 1997). The CMM could be used as a reference guide for appraising the current state of the organization's software process in order to improve project performance.

In addition, the survey determines the status of CMM implementation for a large sample of companies. Overall, the software process management maturity of Taiwan software industry is relatively high. This may result from a series of promotional programs by the Five-year Development Plan of The Industry Development Bureau. However, the software process maturity assessment in this study averaged 3.09 - compared with the average maturity 3.26 of U. S. companies (Ibbs & Kwak, 2000).

However, CMM implementation often exceeds resource expectations. Achieving high levels of maturity is incremental and requires a long-term commitment to continuous process improvement. Also, the CMM does not address certain issues critical to success. These issues include the development team's knowledge, specific project implementation and management strategies, the importance of the project manager, or how to build and maintain a competent project team. Change to newer systems can also create problems as changing the behavior of software engineers is a nontrivial problem. IS people often believe new methods work after they use them and see the results, but they will not use the methods until they believe in their effectiveness. With all of these obstacles, it may be best to tackle certain process aspects first, those that address the more prominent risks. Our results suggest that managers can focus on select areas to better manage the IS project development process and increase the chances of project success when faced with specific project risks.

Future studies should continue to examine the CMM in multiple cultural settings in order to determine the universality of the model. Theoretical frameworks should be built to help explain the relationships between the activities involved with implementing the CMM and the risks present in system development. The difficulties encountered in implementing the CMM at each level should be exposed through careful analysis of organizations that have adopted the model. It would then be possible to further investigate the links between various activities, the risks of development, and the trade-off between risk reduction and costs associated with CMM implementation.

REFERENCES

Anderson, J. C. & Gerbing, D. W. (1988). Structural equation modeling in practice: A review and recommended two-step approach. *Psychological Bulletin, 103*(3), 411-423.

Bandinelli, S. & Fuggetta, A. (1995). Modeling and improving an industrial software process. *IEEE Transaction on Software Engineering, 21*(5), 440-454.

Barki, H., Rivard, S., & Talbot, J. (1993). Toward an assessment of software development risk. *Journal of Management Information Systems, 10*(2), 203-223.

Butler, K. L. (1995, July). The economic benefits of software process improvement. *Cross Talk*, 14-17.

Chen, Y. S. (1999). The relationships between organizational characteristics, information system maturity, IS department characteristics and software process maturity. Unpublished Master's dissertation, Dept. of MIS, National Chung Cheng University.

Chen, S. H. (2000). The relationship of TQM-based organizational culture and software process maturity. Unpublished Master's dissertation, Dept. of MIS, National Chung Cheng University.

Couillard, J. (1995). The role of project risk in determining project management approach. *Project Management Journal*, 3-15.

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika, 16*(3), 273-334.

Deephouse, C., Mukhopadhyay, T. D., Goldenson, R., & Kellner, M. I. (1995-96, Winter). 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.

Dion, R. (1993). Process improvement and the corporate balance sheet. *IEEE Software, 10,* 28-35.

Fitzgerald, B. & O'Kane, T. (1999). A longitudinal study of software process improvement. *IEEE Software, 16*(3), 37-45.

Herbsleb, J. D. & Goldenson, D. R. (1996). A system survey of CMM experience and results. *Proceedings of ICSE-18,* 323-330.

Herbsleb, J. D., Zubrow, D., Goldenson, D. R., Hayes, W., & Paulk, M. (1997, June). Software quality and the capability maturity model. *Communication of the ACM, 40*(6), 30-40.

Hsieh, K. D. (1996). Software process maturity and its related factors of large companies in Taiwan. Unpublished Master's dissertation, Dept. of MIS, National Chung Cheng University.

Humphrey, W. S., Snyder, T. R., & Willis, R. R. (1991). Software process improvement at Hughes Aircraft. *IEEE Software, 8*(4), 11-23.

Ibbs, C. W. & Kwak, Y. H. (2000). Assessing project management maturity. *Project management Journal, 31*(1), 32-43.

Jian, J. & Klein, G. (1999). Risks to different aspects of system success. *Information & Management, 36*, 263-272.

Kuilboer, J. P. & Ashrafi, N. (2000). Software process and product improvement: An empirical assessment. *Information and Software Technology, 42*, 27-34.

Lipke, W. H. & Rosenbaum, S. (1993). Software improvements in an international company. *Proceedings of 15th International Conference on Software Engineering*, 212-220.

Necco, C. R., Gordon, C. L., & Tsia, N. W. (1987). System analysis and design current practices. *MIS Quarterly, 11*(4), 461-476.

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

Paulk, M., Curtis, B., Chrissis, M. B., & Weber, C. V. (1993, July). Capability maturity model for software, version 1.1. *IEEE Software*, 18-27.

Paulk, M. Weber, C. V., Garcia, S. M., Chrissis, M. B. & Bush, M. (1993, February). Key practices of the capability maturity model, version 1.1. CMU/SEI-93-TR-25, Software Engineering Institute. Pittsburgh, PA.

Rai, A. & Al-Hindi, H. (2000). The effects of development process modeling and task uncertainty on development quality performance. *Information & Management, 37*, 335-346.

Rivachandran, T. & Rai, A. (1999-2000). Total quality management in information systems development: Key constructs and relationships. *Journal of Management Information Systems, 16*(3), 119-155.

Subramanian, G. H. & Zarnich, G. E. (1996). An examination of some software development effort and productivity determinants in ICASE tool projects. *Journal of Management Information Systems, 12*(4), 143-160.

Swanson, K., McComb, D., Smith, J., & McCubbery, D. (1991). The application software factory: Applying total quality techniques to system development. *MIS Quarterly, 15*(4), 566-579.

The Standish Group (1995). Chaos. *Standish Group Report.*

The Standish Group (1996). Unfinished Voyages. *Standish Group Report.*