TOTAL QUALITY MANAGEMENT, VOL. 13, NO. 5, 2002, 701–713

Routledge
Taylor & Francis Group

Software process management of top companies in Taiwan: a comparative study

ELDON Y. LI[1], HOUN-GEE CHEN[2] & TIEN-SHENG LEE[3]
[1]*Orfalea College of Business, California Polytechnic State University, San Luis Obispo, California, USA,* [2]*Institute of Technology Management, National Tsing Hua University, Hsinchu, Taiwan, People's Republic of China &* [3]*Department of Decision Sciences and Managerial Economics, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong, People's Republic of China*

ABSTRACT *In today's business environment, information technology (IT) is an indispensable tool for any corporation. One of the largest IT investments goes to software-related products and activities such as development, maintenance and enhancement. In order to reduce the cost of software activities and improve the quality of software products, effectively managing the software development process is an important topic in the IT field. Since the early 1990s, there has been rapidly growing interest in the capability maturity model (CMM) in software organizations. With the aid of CMM guidelines, a software organization can continually improve its software process. This research discusses the essence of CMM guidelines and surveys the IT organizations of the top 1000 business companies in Taiwan. It explores the status of software process management in these companies and compares the findings with Japanese and US data reported in the literature.*

Introduction

Software development and maintenance are two significant investment categories in a modern business company. Regardless of self-developing or outsourcing, the investment on software and its related products and activities usually accounts for more than 50% of the total information technology (IT) budget in a company (Yourdon, 1993). Consequently, the quality of software must be ensured—otherwise the maintenance cost will be elevated or the software must be re-developed. Either case will hinder the normal operations of a business and, in turn, weaken the competitive advantage of a company.

To improve the quality of software, traditionally one would use structured techniques, CASE tools, prototyping, object-oriented methodology or software reuse, among other methods, during the software development process. However, all these tools or methods do not guarantee the quality of software they help produce. The key to developing a quality software product lies in how well one manages the software engineering process (simply called the 'software process') (Humphrey, 1989; Rubin, 1993).

Correspondence: Eldon Y. Li, Orfalea College of Business, California Polytechnic State University, San Luis Obispo, California 93407, USA. E-mail: eli@calpoly.edu

ISSN 0954-4127 print/ISSN 1360-0613 online/02/050701-13
DOI: 10.1080/0954412022000002081

Since mid 1980s, researchers at the Software Engineering Institute (SEI) have been studying the maturity levels of software process in various software developing organizations. In order to provide guidance to software contractors on the practice of software process management (SPM), the Institute's researchers have developed a set of activities and a model describing software process improvement, and have called it the 'capability maturity model' (CMM) (Paulk *et al.*, 1991, 1993). Since its appearance, the model has drawn tremendous attention in the software-engineering field. Studies have shown that software process improvement can significantly improve software quality and productivity (Herbsleb *et al.*, 1994, 1997; Harter *et al.*, 2000; Krishnan *et al.*, 2000). The original intent of the CMM was to serve as a tool for the US Department of Defense to benchmark the SPM infrastructure of its software contractor. Nevertheless, it may be used to diagnose the software development capability of the IT department/group of a business company and to identify process improvement strategies for improving software product quality. A review of the literature reveals that there are only a few studies reporting the status of software industry (Humphrey *et al.*, 1989; Kitson & Masters, 1992; Herbsleb *et al.*, 1997). Humphrey *et al.* (1991a) are the only ones who included commercial companies as a part of their study. No study had reported the status of business industries in general. To fill this void, this study will apply the CMM assessment method to diagnosing the SPM infrastructure of the top business companies in Taiwan. The reason for such an endeavour is twofold. One is to examine the feasibility of applying the CMM assessment method to an IT organization in a general business. In the case of poor feasibility, we shall attempt to propose a necessary modification to the assessment method. The other is to help the Taiwan government and industries take the pulse of their SPM status and gain an insight into their strengths and weaknesses. This, in turn, allows them to prioritize their actions and allocate their resources to improving their national SPM status effectively. Since there is no previous study on this subject matter in Taiwan, the results will be compared with historical data from Japan and the US in order to identify the strengths or weaknesses of the responding companies. Finally, the implications for software process managers in Taiwan will be discussed.

The framework of software process management

In his seminal book, *Quality is Free*, P.B. Crosby (1979) proposed a quality improvement model consisting of five phases: uncertainty, awakening, enlightenment, wisdom and certainty. This model was adapted by SEI. In collaboration with Mitre Corporation, SEI developed a software process maturity framework in 1987 (Humphrey & Sweet, 1987). Four years later, they announced the first version of the capability maturity model (CMM) (Paulk *et al.*, 1991). Later in 1993, they released CMM 1.1 (Paulk *et al.*, 1993)

In the CMM, the maturity of software development and management are divided into five levels: Initial, Repeatable, Defined, Managed, and Optimizing. These five levels follow a predictive sequence. A software company must go through the Initial level before it migrates into the Repeatable level, then the Defined level, and so forth. The characteristics of these levels as identified by the SEI (Humphrey *et al.*, 1991b) are shown in Table 1. The implementation of key practices at the Initial level is of high risk but would improve the productivity and quality of the software process dramatically. As the CMM level increased, the risk is lower and the productivity or quality gain is smaller. The underlying philosophy of the CMM is continual improvements on the software process during the journey to maturity. This seems to coincide with the philosophy of total quality management (Deming, 1982, 1986).

Table 1. *The characteristics of CMM levels*

CMM Level	Characteristics	Key challenges	Result
1. Initial	(Ad hoc) Process chaotic	• Project management • Project planning • Configuration management • Software quality assurance	Risk
2. Repeatable	(Intuitive) Process dependent on individuals	• Changing technology • Problem analysis • Problem prevention	
3. Defined	(Qualitative) Process defined and institutionalized	• Process measurement • Process analysis • Quantitative quality plans	
4. Managed	(Quantitative) Measured process	• Changing technology • Problem analysis • Problem prevention	
5. Optimized	Improvement fed back into process	• Still human intensive process • Maintain organization at optimizing level	Productivity & Quality

Source: Adapted from Humphrey *et al.* (1991b).

Determining maturity level

In order to determine the maturity level, Humphrey & Sweet (1987) developed a questionnaire containing 85 questions. The questions were distributed into four levels and each level contains two categories of questions corresponding to 'desired' and 'critical' practices. The first-level questions determine if an organization is qualified for Level 2 of maturity, thus they are called 'Level-2 questions'. If an organization has affirmative answers to at least 80% of all Level-2 questions and at least 90% of the Level-2 'critical' questions, it is qualified at Level 2. Otherwise, it is at Level 1. Only after being qualified for one level can the organization be assessed for the next level. To qualify for Level 3, one must have affirmative answers to at least 80% of all Level-2 and Level-3 questions and at least 90% of the critical questions of both Level 2 and Level 3. This procedure is repeated for the Level-4 and Level-5 assessments.

Assessing the maturity level

According to SEI, there are five methods for assessing CMM levels (Humphrey *et al.*, 1991a; Kitson & Masters, 1992). These are assessment tutorial, self-assessment, SEI-assisted assessment, SEI-licensed vendor assessment, and capability evaluation. Among these methods, only assessment tutorial and self-assessment are conducted with little or no direct SEI involvement. The other assessment methods require site visits and direct SEI involvement. An assessment tutorial is given to professionals who wish to learn more about the process management concept, assessment techniques, and the SEI assessment methodology with minimal investment. At the end of the tutorial, they complete an assessment questionnaire based on their experience on a project. A self-assessment is similar to a SEI-assisted assessment except that the assessment team is composed of software professionals primarily from the organization being assessed. However, a formal site visit inspecting various software development projects is required if a company is being certified for the eligibility of contracting a DOD's software project. Without such a formal procedure, a significant Hawthorn effect (Roethlisberger & Dickson, 1939) will be inevitable.

Research method

The subjects

The subjects for this study were the top 1000 companies listed in a recent *Directory of Large Corporations in Taiwan* published by China Credit Information Service, Ltd, Taipei, Taiwan. The sample included 667 manufacturing and 333 service companies. The CMM questionnaire was sent twice to the IT executive in each of the sampled companies. The executive was implored to direct the questionnaire to someone who has expert knowledge about the software development practices in the company. Our questionnaire also indicated that additional experts should be consulted if a single individual could not answer all the questions. The first mailing yielded 85 response questionnaires while the second mailing received 63 questionnaires. After removing the questionnaires containing excessive missing or inconsistent data, 138 usable ones were identified, giving a 13.8% response rate. The characteristics of respondents are listed in Table 2.

The questionnaire

The questionnaire used for this study was adapted from Humphrey & Sweet (1987). The respondents were asked to check each 'yes' box only if the key practice in question has been institutionalized. The original questionnaire of Humphrey & Sweet contained 85 questions. However, three statements of these questions appeared to address two independent practices in a single statement and each was modified into two questions. For example, the original statement of Q2.2.4 was 'Are statistics on software code and test errors gathered?' It was divided into 'Q2.2.4.1. Are statistics on software code errors gathered?' and 'Q2.2.4.2. Are statistics on software test errors gathered?' Likewise, Q2.2.6 referred to code and test errors in the same statement, while Q2.2.13 referred to design review and code review. Each of these statements was split into two. Moreover, software size alone is insufficient for software cost estimation (Boehm, 1981). It needs to be supplemented by software complexity (McCabe, 1976; Li, 1987). Therefore, we added a new key practice behind Q2.1.14 for the software complexity estimate, namely, 'Q2.1.14.*x*. Is a *formal procedure* used to make estimates of software complexity?' This results in a total of 89 questions in the questionnaire. In addition, statement 'Q2.2.2. Are profiles of software size maintained over time for each software configuration item?' was rephrased as 'Are profiles of software metrics (size or complexity) maintained over time for each software configuration item?' We did not add a new item for software complexity in this case because its profile history typically is kept along with size estimates, if its estimates are made regularly.

Procedure

This study adapted the assessment tutorial method and used a mail survey to collect the data. Instead of conducting a tutorial session, a tutorial document was included in the mailing along with the survey questionnaire. We expected that the respondents would attempt to indicate fairly and objectively the actual SPM status of their companies for two reasons. First, the outcome of this study does not have strings attached to the responding companies. Over- or under-estimating one's own SPM status does not help the company to secure a resource or a contract. In fact, it would be better for the respondent to estimate the status accurately in order to diagnose the weaknesses of the software process in the company and plan for corrective actions. Second, the sample size (138 companies) of this study is large enough to offset the response bias based on the law of large numbers and the central limit theorem (Conover, 1971; Kerlinger, 1973). Further validation of the data is reported in the following section.

Table 2. *The characteristics of respondents (N = 138*)*

Category	Classification	N*	% of total
Industry sector	Service	32	23.0
	Manufacturing	106	77.0
Type of industry	Food processing	13	12.5
	Textile, apparels	6	5.8
	Petrochemical, plastic	12	11.5
	Electronic, electrical	20	19.2
	Steel, machinery	21	20.2
	Transportation and makers	4	3.8
	Other manufacturing	5	4.8
	Transporter	4	3.8
	Merchandising, wholesaling	3	2.9
	Import, export, trading	2	1.9
	Banking, insurance	4	3.8
	Architectural, construction	4	3.8
	Other services	6	5.8
Annual company sales (in million NT$)	0 ~ 999	14	13.5
	999 ~ 1150	12	11.5
	1150 ~ 2460	34	32.7
	2460 ~ 7710	30	28.8
	7710 or more	14	13.5
Number of company employees	0 ~ 100	4	3.8
	100 ~ 299	28	26.9
	299 ~ 516	19	18.3
	516 ~ 1626	38	36.5
	1626 or more	15	14.4
Respondent's education	Master degree	20	15.3
	Baccalaureate degree	72	55.0
	Associate degree	37	28.2
	High school diploma	2	1.5
Respondent's experience in software development	0 ~ 2 years	18	13.3
	3 ~ 5 years	81	60.0
	6 ~ 10 years	33	24.5
	11 years or more	3	2.2
Programming language for software development	Database software	11	9.0
	3GLs	75	61.5
	4GLs	30	24.6
	Object-oriented languages	5	4.1
	CASE	1	0.8
Number of software development employees	0 ~ 5 persons	64	47.4
	6 ~ 10 persons	34	25.2
	11 ~ 20 persons	15	11.1
	121 ~ 50 persons	16	11.9
	51 persons or more	6	4.4

*Owing to missing responses, the total of each category might not equal 138.

Data validation

In order to ensure the validity of the data, non-response bias and sample representativeness were examined. The non-response bias was examined by testing the differences between the usable data collected from the first-wave of mailing and those from the second one. No

significant difference was found at the 95% confidence level, indicating the absence of the bias. Subsequently, we examined the data representativeness by testing the differences in demographic distributions between the population (1000 companies) and the usable sample (138 respondents). No significant difference was found (at the 95% confidence level) in terms of company size (including annual sales and number of employees) and industry type, indicating the representativeness of the response data. This allows us to begin analysing the data and interpreting the results.

Analysis and results

Based on the evaluation guidelines described in Humphrey & Sweet (1987), the average percentage of 'yes' answers in each level of maturity is around 35%, which is much lower than the 80% criterion prescribed by SEI (see Table 3). Specifically, only one company reached Level 5 and another one reached Level 2, the rest of the responding companies all stayed at Level 1. The reason is obvious, that most of the companies were performing many SPM activities throughout all levels of maturity in the CMM. They did not follow the time sequence of evolution as prescribed by the CMM. That is, they did not perform only those activities earmarked to one level before they moved onto the next level. Rather, they simultaneously performed the activities of different levels. In this case, they are likely to stay at the Level 1 for a long time before they start moving up to the higher levels, and once they are moving up, they should reach Level 5 much faster than we would expect. This contention is supported by the contrast between two companies, one is a reprographics equipment manufacturer, which was qualified at Level 1, the other is a steel mill company which was qualified at Level 5 (see Table 4). The differences between these two companies were not much. Both companies were missing four key practices that were different from each other (see Table 5). All these eight key practices are Level-2 or Level-3 practices. They are not difficult to implement. Although both companies looked equally sophisticated, the reprographics manufacturer did not qualify for a higher CMM level because it did not reach the minimum threshold of 90% of the Level-2 critical practices prescribed by SEI. This indicates that the threshold-based assessment method of the CMM level was not feasible for this sampled company. In fact, it is clearly not feasible at all for most companies in this study and even in general. SEI did recognize this deficiency and, consequently, in 1993 revised its assessment method and eliminated a percentage requirement at each CMM level. Nevertheless, the revised method is very ambiguous and complex to use. A plausible alternative to

Table 3. *The profile of CMM achievements from the responding companies (N = 138)*

Type of Question Item	Number of Questions	Question Numbers Subtotal	Minimum % of 'Yes' Answers	Mean of 'Yes' % from the Sample	Std. Dev. of 'Yes' % from the Sample
Level 2 Total*	35	35	80%	35.2%	19.3%
Level 2 Critical	14	14	90%	31.0%	20.0%
Level 3 Total*	32	67	80%	36.8%	19.7%
Level 3 Critical	13	27	90%	32.5%	20.1%
Level 4 Total*	18	85	80%	35.0%	19.8%
Level 4 Critical	14	41	90%	31.5%	20.2%
Level 5 Total*	4	89	80%	35.5%	20.0%
Level 5 Critical	4	45	90%	32.6%	20.4%

*Number of Total Questions = Number of Desired Questions + Number of Critical Questions.

Table 4. *The contrast of CMM achievements in two companies*

Type of Question Item	% of 'Yes' Answers Required by CMM	% of 'Yes' Answers in the Reprographics Equipment Maker	% of 'Yes' Answers in the Steel Mill Company
Level 2 Total*	80%	91.4%**	91.4%**
Level 2 Critical	90%	85.7%	100.0%**
Level 3 Total*	80%	92.5%**	94.0%**
Level 3 Critical	90%	88.9%	96.3%**
Level 4 Total*	80%	94.1%**	95.3%**
Level 4 Critical	90%	92.7%**	97.6%**
Level 5 Total*	80%	94.4%**	95.5%**
Level 5 Critical	90%	93.3%**	97.8%**

*Number of Total Questions = Number of Desired Questions + Number of Critical Questions.
**Reach the minimum requirement of this level prescribed by the CMM.

Table 5. *The missing key practices of two companies*

A Level-5 Steel Mill Company		A Level-1 Reprographics Equipment Manufacturer	
Q1.1.1. (Level-2)	For each project involving software development, is there a designated software manager?	*Q2.1.15. (Level-2)	Is a *formal procedure* used to produce software development schedules?
Q1.3.1. (Level-2)	Is a *mechanism* used for maintaining awareness of the state-of-the-art in software engineering technology?	Q2.2.1. (Level-2)	Are software staffing profiles maintained of actual staffing versus planned staffing?
Q2.1.7. (Level-2)	For each project, are independent audits conducted for each step of the software development *process*?	Q2.4.4. (Level-3)	Is a *mechanism* used for independently calling integration and test issues to the attention of the project manager?
*Q2.4.16. (Level-3)	Are software code reviews conducted?	*Q2.4.7. (Level-2)	Do software development first-line managers sign off on their schedules and cost estimates?

*Indicates a 'critical' key practice.

evaluating the SPM achievement of a company is to ignore the thresholds of 80% and 90% and the level assignment of each key practice, and simply examine the percentage of the total number of practices. We shall use this revised assessment method to analyse the data in this study.

Analysis with revised CMM assessment method

According to our revised assessment method, all practices are assumed to be equally important. The rationale is that although the difficulty and the risk of implementation vary widely from one key practice to another, implementing a desired key practice is not necessarily more difficult or riskier than implementing a critical key practice and vice versa. Moreover, although the threshold approach and the level assignment of key practices are suitable for guiding an individual company through the evolution of SPM maturity, they are not suitable for assessing the overall achievement of SPM key practices in a company. Nor are they suitable for assessing the overall achievement of individual key practices in an industry.

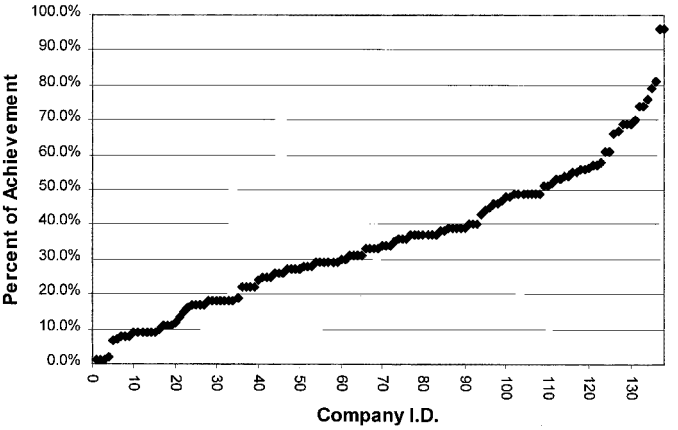

Figure 1. *A key-practice profile based on revised assessment method.*

According to SEI, these threshold percentages have been arbitrarily established to promote consistency and objectivity. They are not scientifically determined. Therefore, both the threshold approach to evaluating maturity level and the classification of desired and critical practices do not serve the purposes of this study and only the percentage of the total number of practices is used to reflect the overall status of SPM infrastructure in a sampled company. Using this percentage, we can still compare the SPM status from company to company. When necessary, we can derive the SEI's maturity levels from the raw data of our samples in order to compare with those in the SEI-maintained assessment database. Figure 1 exhibits the profile of key practices achieved by the top companies in Taiwan, while Fig. 2 shows the frequency distribution of the percentage of achievement among these respondents. According to Fig. 1, there were two companies that achieved over 90% of the key practices and that, except for the very high and very low achievers, the differences in the percentage of key practices are very small, as indicated by the small slope of the line in the figure. The average percentage of achievement was 35.4% for the entire sample with a standard deviation of 20%. The distribution in Fig. 2 reveals that only 5.7% of companies achieved 70% or above of key practices. The largest group (22.5%) of companies achieved between 30% and 40%

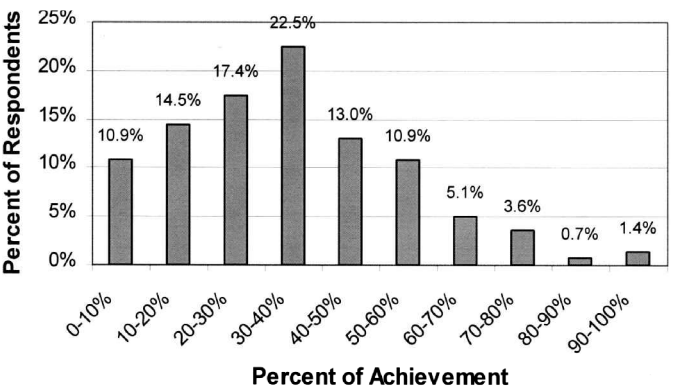

Figure 2. *Distribution of key-practice achievement levels.*

of the key practices. Many companies (78.26%) carried out less than 50% of the key practices. Among the key practices, the five most implemented practices are as follows. Note that two key practices are tied at fifth place.

(1) Providing each software developer with a private computer-supported workstation/terminal (92.3%, Item Q1.2.1.).
(2) Applying coding standards to each software development project (75.4%, Item Q2.1.9.).
(3) Having the software organization use a standardized and documented software development process on each project (72.3%, Item *Q2.1.1)
(4) Applying code maintainability standards (70.8%, Item Q2.1.11.).
(5) (a) Having a designated software manager for each project involving software development (63.8%, Item Q1.1.1.). (b) Having the project software manager report directly to the project (or project development) manager (63.8%, Item Q1.1.2.),

On the other hand, the five least implemented practices are:

(1) Gathering statistics on software code errors (7.7%, Item *Q2.2.4.1s.).
(2) Using a formal procedure to make estimates of software complexity (8.5%, Item *Q2.1.14.x.).
(3) Using a mechanism for assuring the adequacy of regression testing (8.5%, Item *Q2.4.21.).
(4) Maintaining software-staffing profiles of actual staffing versus planned staffing (9.2%, Item Q2.2.1.).
(5) Gathering statistics on software test errors (9.2%, Item *Q2.2.4.2s.).

Comparing with key practices in Japan and the US

Humphrey *et al.* (1991a) reported three sets of assessment data, two from the US and one from Japan. Each of their data points is one set of yes-no responses to the maturity questionnaire regarding a specific software project. The participants in the US were Department of Defense (DoD) organizations, DoD contractors, and commercial organizations. These participants included 55 projects from 10 organizations that participated in SEI-assessments and 113 projects from over 70 organizations that participated in assessment tutorials. In contrast, the participants in Japan were from over 88 software organizations in six Japanese companies. These participants included many business-application programming groups, a few communications and military suppliers, and two computer manufacturers. Through the assessment tutorials, 196 projects were assessed. Humphrey *et al.* found that US software industry in general was ahead of its Japanese counterpart, perhaps owing to the stringent requirements that the DoD put on its software contractors. The results of these surveys are presented in Figs 3 and 4.

Although our participants in Taiwan are not commercial software developers, the SPM key practices are essentially the same in any organizations. Therefore, Humphrey *et al.*'s data are useful for us to identify the weaknesses of SPM in our participating companies. The rationale is that, if significantly more organizations in Japan and the US could perform a key practice back in 1991 than those in Taiwan do now, it could be expected that the gap would be wider today. It would be a weakness of Taiwanese companies not to keep up with their Japanese and US counterparts regarding this practice. A scrutiny of Figs 3 and 4 reveals that Taiwanese companies were not significantly ahead of their Japanese and US counterparts in

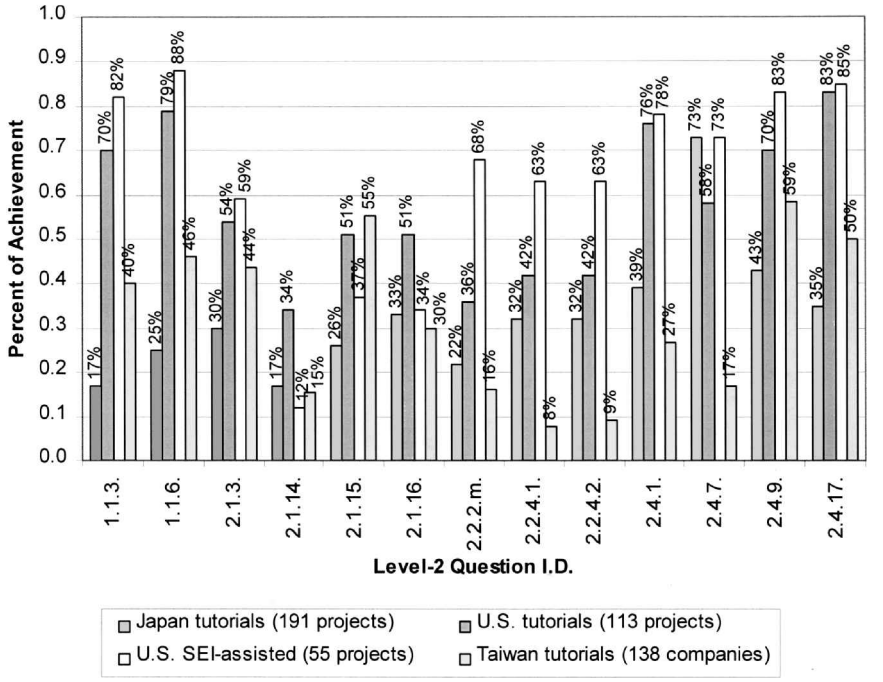

Figure 3. *Achievement of Level-2 critical practices in three countries.*

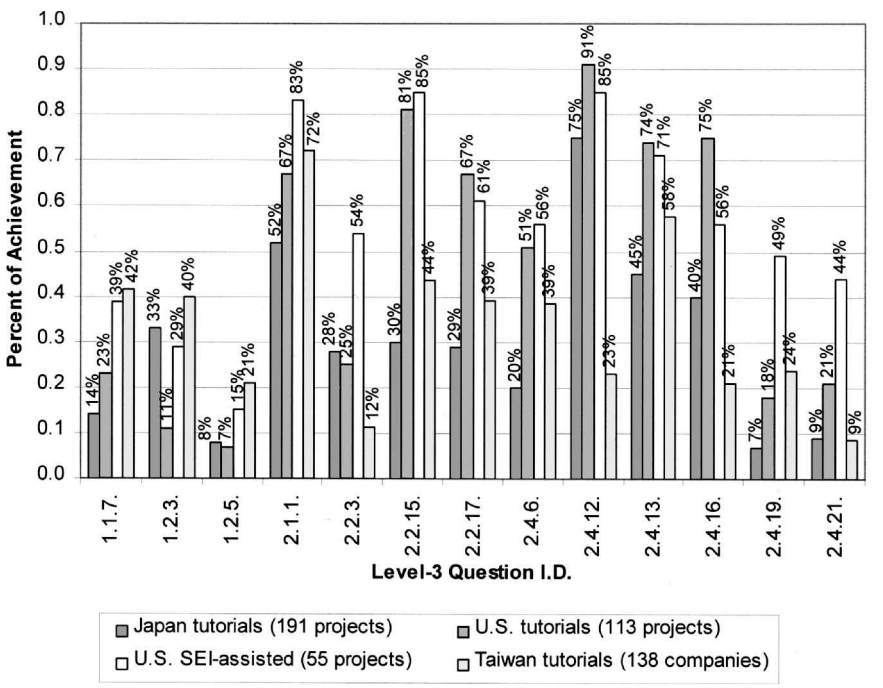

Figure 4. *Achievement of Level-3 critical practices in three countries.*

many SPM practices, even though they were surveyed several years later. It is even more surprising that Taiwanese companies were significantly behind in six key practices:

(1) Gathering statistics on software design errors (11.5%, Item *Q2.2.3.).
(2) Gathering statistics on software code errors (7.7%, Item *Q2.2.4.1s.).
(3) Gathering statistics on software test errors (9.2%, Item *Q2.2.4.2s.).
(4) Empowering software development first-line managers to sign off on their schedules and cost estimates (16.9%, Item *Q2.4.7.).
(5) Conducting internal software design reviews (23.1%, Item *Q2.4.12.).
(6) Conducting software code reviews (20.8%, Item *Q2.4.16.).

All these practices are critical ones and Items 2 and 3 belong to the least implemented practices. The first three items are the only three practices in the questionnaire related to error statistics gathering. This indicates that Taiwanese business companies did not value the error statistics that enable a software organization to predict errors, prevent errors, and develop training programmes, etc. Items 5 and 6 indicate the companies are lagging behind those of the other two countries in the two technical reviews most critical to software quality—design and code reviews. Finally, Item 4 reveals that first-line managers are less likely to be accountable for the schedules and cost estimates. This shows less empowerment to the lower-level managers and is against the philosophy of total quality management (Li *et al.*, 2000). Although this weakness may or may not affect software quality, it is directly related to the efficacy of project management and, at some point in time, may result in unresponsiveness to schedule sliding and cost overrun. From all these weaknesses, we may expect that the software quality and productivity of these Taiwanese companies would be much lower than their counterparts in Japan and the US.

Conclusions and recommendations

Software process management in Taiwan's business industries is still in its infancy stage. According to the CMM classification, it is at the Initial level and moving into the Repeatable level. This study surveys the software organizations in Taiwan's top 1000 companies and reveals the overall status of each SPM practice achievement. This overall status provides each organization with a yardstick against which it could measure itself and identify its own strengths and weaknesses. Based on the results of this study, several conclusions and recommendations may be drawn.

First, the threshold-based CMM assessment method is not feasible for identifying the overall SPM infrastructure of a business company. This method might be suitable for benchmarking the software contractors and guiding them to institutionalize lower-level practices before implementing upper-level practices. However, it does not lend itself to revealing the overall status of SPM infrastructure. A simple percentage-based assessment method has proven effective for this purpose.

Second, the percentages of key practices implemented by Taiwanese business companies are surprisingly low. Most (78.3%) of these companies did not implement more than 50% of key practices. The average percentage of achievement is 35.4%. This is probably due to the lack of software engineering training among the software professionals in these companies. In fact, most software engineering curricula of Taiwanese universities did not emphasize quality management techniques and processes. These curricula need to be improved to meet the growing demand for software quality and productivity in the industries.

Third, Taiwan's companies were very poor in gathering code and test error statistics, controlling software complexity, regression tests, and human resource usage. This was

evidenced by the five least implemented practices. All these practices are non-value-added supporting activities, although they are related with quality control and project management. This may be why the companies are not interested in implementing these, because they are more interested in the value-added activities such as analysis, design, programming and installation. The fact that very few companies measured software complexity (8.5%) and software size (15.4%) reconfirms that project effort was not properly documented and could not be quantitatively estimated in most companies. Eventually, management in these companies needs to change their attitude of 'quality is at the expense of productivity'. The management must realize that without quality, productivity means nothing, and that in order to manage effectively the software process and produce quality software, adequate resources must be provided to carry out the CMM key practices.

Fourth, comparing to Japanese and US data reported in 1991, Taiwan's companies are still behind in six key practices, even nearly a decade later. These six missing practices obviously are the major weaknesses of many Taiwanese business companies. These weaknesses are related to a lack of error statistics, technical reviews, and empowerment. Taiwanese business companies rarely gathered code error statistics, test error statistics, and design error statistics. Consequently, they did not have the ability to make error prediction and to prevent the errors from happening. In addition, they were not able to design training programmes for their staff to learn how to reduce human errors in design, code, and test activities. As for design reviews and code reviews, these are effective methods for error prevention and quality assurance. Adequate time and effort must be allocated to these activities.

Finally, empowerment to first-line managers is an effective means of eliminating communication gaps because the first-line managers virtually live with their staff and they know best what the staff want, need, and are good at. In fact, empowerment should happen not only to the lower-level managers, but also to all staff on the software project. Everyone should be accountable for his or her own work. Everyone must be committed to improve continually his or her own work and to suggest improvement on organizational processes. This is the essence of the total quality management (TQM) process. In our opinion, SPM is the heart while TQM is the soul of a software organization. Without both, the organization would never be able to produce quality software. Therefore, regardless of how many CMM practices a software organization has institutionalized, the management should ensure that TQM concept and methods have been instilled into the organization before implementing the SPM process. Otherwise, the quality of CMM practices will be questionable. This, in turn, will make the quality of its software products questionable.

References

BOEHM, B.W. (1981) *Software Engineering Economics* (Englewood Cliffs, NJ, Prentice-Hall).

CONOVER, W.J. (1971) *Practical Nonparametric Statistics* (New York, Wiley).

CROSBY, P.B. (1979) *Quality Is Free: The Art of Making Quality Certain* (New York, McGraw-Hill).

DEMING, W.E. (1982) *Quality, Productivity, and Competitive Position* (Cambridge, MA, Massachusetts Institute of Technology, Center for Advanced Engineering Study).

DEMING, W.E. (1986) *Out of the Crisis* (Cambridge, MA, Massachusetts Institute of Technology, Center for Advanced Engineering Study).

HARTER, D.E., KRISHNAN, M.S. & SLAUGHTER, S.A. (2000) Effects of process maturity on quality, cycle Time, and effort in software product development, *Management Science*, 46(4), pp. 451-466.

HERBSLEB, J., CARLETON, A., ROZUM, J., SIEGEL, J. & ZUBROW, D. (1994) Benefits of CMM-based software process improvement: executive summary of initial results, Technical Report CMU/SEI-94-SR-013, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

HERBSLEB, J., ZUBROW, D., GOLDENSON, D., HAYES, W. & PAULK, M. (1997) Software quality and the capability maturity model, *Communications of the ACM*, 40, 6, pp. 30-40.

HUMPHREY, W.S. (1989) *Managing the Software Process* (Reading, MA, Addison-Wesley).

HUMPHREY, W.S. & SWEET, W.L. (1987) A method for assessing the software engineering capability of contractors, Technical Report CMU/SEI-87-TR-23, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

HUMPHREY, W.S., KITSON, D.H. & GALE, J. (1991a) A comparison of US and Japanese software process maturity, Technical Report CMU/SEI-91-TR-27, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

HUMPHREY, W.S., SNYDER, T.R. & WILLIS, R.R. (1991b) Software process improvement at Hughes Aircraft, *IEEE Software*, 8(4), pp. 11-23.

KERLINGER, F.N. (1973) *Foundations of Behavioral Research*, 2nd edn (New York, Holt, Rinehart and Winston).

KITSON, D. & MASTERS, S. (1992) An analysis of SEI software process assessment results 1987-1991, Technical Report CMU/SEI-92-TR-24, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

KRISHNAN, M.S., KRIEBEL, C.H., KEKRE, S. & MUKHOPADHYAY, T. (2000) An empirical analysis of productivity and quality in software products, *Management Science*, 46(6), pp. 745-759.

LI, E.Y. (1987) On the cyclomatic metric of program complexity, *Quality Data Processing*, 1(3), pp. 15-23.

LI, E.Y., CHEN, H.G. & CHEUNG, W.M. (2000) Total quality management in software development process, *The Journal of Quality Assurance Institute*, 14(1), pp. 4-6 & pp. 35-41.

MCCABE, T.J. (1976) A complexity measure, *IEEE Transactions on Software Engineering*, 2(4), pp. 308-320.

PAULK, M., CURTIS, B., AVERILL, E., BAMBERGER, J., KASSE, T., KONRAD, M., PERDUE, J., WEBER, C. & WITHEY, J. (1991) Capability maturity model for software, Technical Report CMU/SEI-91-TR-024, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

PAULK, M., CURTIS, B., CHRISSIS, M. & WEBER, C. (1993) The capability maturity model for software, version 1.1, Technical Report CMU/SEI-93-TR-024, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA 15213.

ROETHLISBERGER, F.J. & DICKSON, W.J. (1939) *Management and the Worker; an Account of a Research Program Conducted by the Western Electric Company, Hawthorne Works, Chicago.* With the assistance and collaboration of Harold A. Wright. (Cambridge, MA, Harvard University Press).

RUBIN, H.A. (1993) Software process maturity: measuring its impact on productivity and quality, *Proceedings of the International Conference on Software Engineering*, pp. 468-476.

YOURDON, E. (1993) *Decline and Fall of the American Programmer* (Englewood Cliffs, NJ, Prentice-Hall).