

A Management Framework for Software Project Development

HO LEUNG TSOI

Software Quality Institute
Griffith University
tsoihl@hknet.com

ABSTRACT

Software project development includes a number of activities that result in a delivered product (software). As software becomes more and more expensive to develop, project management has been recognised as a difficult task in practice. There are a lot of unpredictable factors existing in the software development cycle that have become contributing factors to this problem. This paper gives an overview of the present state of the art of the software development projects. Moreover, a framework is proposed to help project management to get better understanding and make all activities run on schedule. A case study is included to demonstrate the merit of this framework.

Keywords: *FPA, Project Management, Software Life Cycle, Software Size Estimation*

1. INTRODUCTION

An effective management is almost an ultimate objective for all software project managers. Although there are many methodologies and techniques available to be used for software development, many projects still suffer from late completion time and exceed budgets. A great number of software applications fail to meet user requirements and quality requirements that result in unacceptable maintenance costs. To tackle the problems, many researchers and practitioners focus on the technical aspect to improve the reliability of estimating the software project size or effort [1,5]. However, software projects are still suffering from overshoots of plans and budgets [3,13]. In my opinion, there are a lot of unpredictable factors existing in the software development cycle that have become contributing factors to this problem. To tackle the “change” problem, a management framework is proposed in this paper to help project management to get better understanding and make all activities run on schedule.

The first section of this paper discusses two general problems, namely budget overrun and late delivery, for software application development. Experience and findings from numerous studies have shown that “change” issue is one the major problems in software project development. For instance, users may change the requirements at any time of the development cycle. All these changes may have a serious influence on the software development.

To address this problem, the rest of this paper presents a framework namely *Software Development Management (SDM)* framework. The author admits that the framework will not cover the issue of productivity and corrective actions selection. All these topics remain an open issue, but further discussion is beyond the scope of this paper. The major task of the framework is to closely monitor and control the development processing, using the estimates and pre-defined metrics as comparisons, so that corrective actions can be promptly taken if necessary.

2. REVIEW OF PROBLEMS

Before starting, project managers have to estimate the size of an application in order to determine an adequate resource level and a realistic completion time. Thus it is essential to find the accurate size and complexity of the software application as early as possible. During the development stage, project managers need to monitor the working progress and overcome all problems to accomplish the project on schedule.

There is no doubt that project managers have to confront with many different kinds of problems, such as technical, management, and personnel, in the software development cycle. They are responsible for ensuring that

all development processes are controlled appropriately in order to meet the budget and schedule [6]. Unfortunately, software projects regularly get out of hand in the development cycle. In other words, cost expended on software development frequently exceeds the estimated budget and results in the software has to be delivered after the deadline or has not matured. Many development problems cited in the literature are usually related to either late delivery or budget overrun

Due to budget overrun and late delivery, many projects are forced to be aborted; are missing implementation of some minor components; or are delivered without thorough debugging. In 1984, a survey study was carried out by Jenkins to address this problem [2]. The developers of 72 information system development projects in 23 major American's corporations were interviewed. Jenkins reported that the average cost and schedule overrun were 36% and 22% respectively. In summary, the overrun of cost and schedule are two serious problems for software project development. Successful project planning mainly relies on a good estimation of the cost and closely monitor the development process. A case study is included to show how the framework can be applied and the merit of this framework.

3. PRINCIPLES OF CHANGE

As mentioned in the previous section, development cost estimation is the most important issue in the software development. Many researchers and practitioners are often wrongly regarded it as a technical problem that can be solved with calculation models, a set of metrics and procedures [7,11,12]. The authors oppose the idea that just technical issues can solve this problem, but are

convinced that a technical model can contribute to the control of software development. Besides, management plays an important role in this problem. As a lot of factors can influence the development progress, project management has been considered difficult to accomplish. This paper presents a management framework for software development and two principles are adopted.

a Change exists in software development

The development of software application is a dynamic environment that is characterized by many changing factors. Change may relate to technologies, personnel, or requirements. Some of them are listed as below:

- Change in software size and complexity
- Change in problem domain
- Change in development paradigm
- Change in development manpower

b. Measurement must be performed dynamically

Since change exists in the development cycle, the cost measurement must be performed dynamically in order to collect the real-time information. In other words, the measurement is used to continuously keep track of the project progress of all activities to cope with change [9].

In summary, change exists in the software development cycle. The project managers need to continuous monitoring the progress and take corrective actions if necessary. This paper proposes a management framework that helps the project managers to apply these two principles in the software development process.

4. SDM FRAMEWORK

The emphasis on the characteristic of the framework is to address the ability to tackle the change that affects the size and progress in development stage. It ensures that all major development processes will be under controlled. To achieve a successful software system, all development processes must be continuously monitored. Thus they can be adjusted during development in order to cope with the problems of the changing world. This paper describes a framework to address the change problem during software development. As a result, the goals (work content) at different development phases can be achieved.

Figure 2 provides an outline, in a simplified form, of the framework developed in this paper. The stages of the system can be classified into: (1) the acquisition phase and (2) the operation phase. Acquisition phase includes all the activities ranging from research and planning. Operation phase includes all activities during the actual software development cycle (Figure 1 and 3). The major tasks in the acquisition phase involve estimate the development cost, establishing cost management policy, establishing historical cost database, and identifying the critical development factors. The major tasks in the operation phase include reviewing monitoring and evaluation metrics, development cost monitoring, and evaluation collected information. The historical database developed in the acquisition phase can be used in the operation phase.

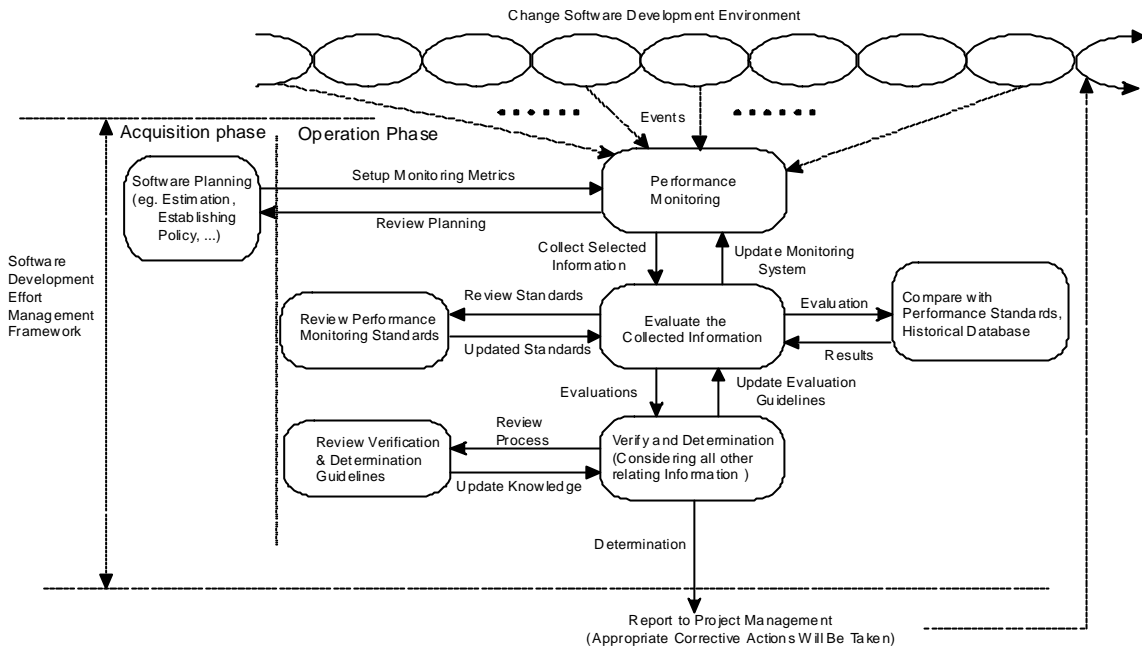


Figure 1: Basic Operation of the SDM Framework

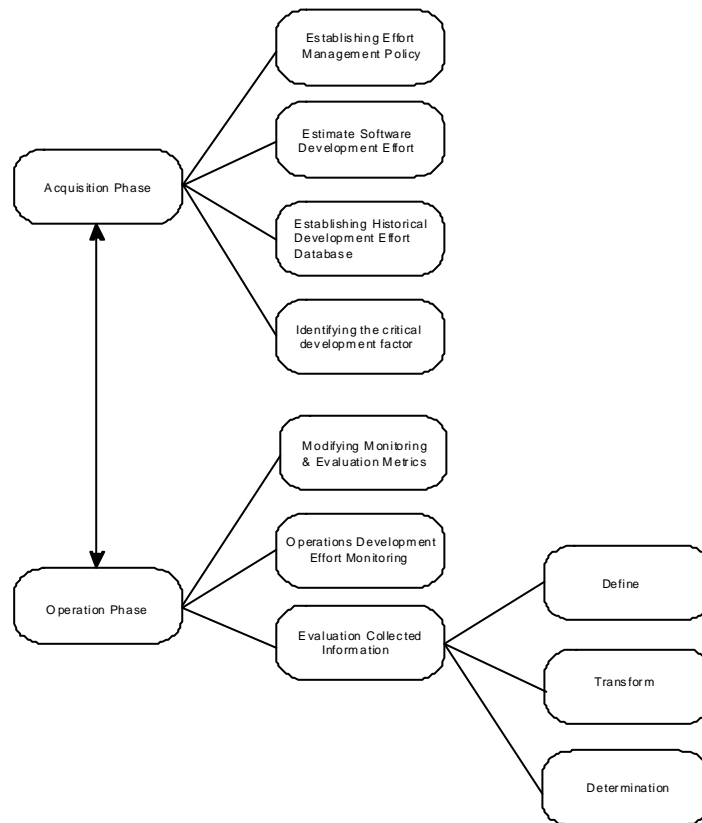


Figure 2: SDM Framework

In summary, the acquisition phase is a pre-requisite for monitor and determination. The operation phase consists of monitoring mechanism which collect information to make decisions and ensuring timely

detections. Besides, a feedback channel, from operation phase to acquisition phase, is set up which operate via status and progress reports to compare actual progress with the plans based on the estimates.

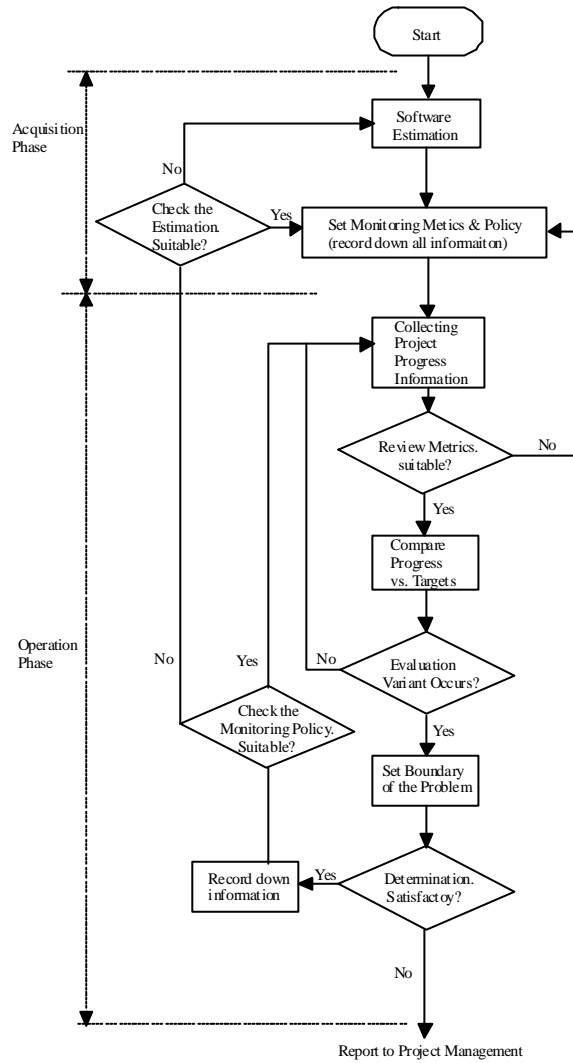


Figure 3: The Project Management Cycle

Acquisition Phase

The acquisition phase includes the activities of estimating, planning, scheduling, budgeting, etc. The major management tasks in this phase are described below:

(a) Cost Estimation

It is commonly agreed that estimating is a prerequisite for good management. At the early stage of software development, the project management need to determine the work-content (size). The work-content of the application can be determined by one of sizing techniques. The result of the size estimation process let project mangers to get the basic picture of the software application project. In summary, the estimation should satisfy three major requirements.

- (i) The overall development effort will be identified.
- (ii) Be compatible with the data requirements
- (iii) Including all critical cost drivers for the development

(b) Assessing the Development Factors

As previously mentioned, change will occur at any time during the development. An assessment should be carried out to help making the monitor policy. Change and supporting are two important factors and need to be assessed. There are many methods for assessing these two factors. For illustration purpose, a simple weighted rating scheme has been adopted. Rating scheme is a well-known analysis technique frequently used in the operations management[9]. The following is an example used to illustrate the assessment.

Change level rating: Low = 1, Medium = 3,
High = 5
Supporting rating: Normal=1, Important = 3,
Critical = 5

Using the above values, the weighted scores of the project can be found. Two important issues are:

- Let the project managers with a good understanding of some potential change problems of the software project development and reduce the threat to failure.
- The result from this stage will affect the arrangement of next following stage. For example, suppose the project managers aware that the project may have problems caused by unstable external supplied components (change occurs) and may lead to schedule and cost overruns (high importance level). Thus a frequently monitoring policy must be adopted.

Obviously, software development is a dynamic process and change is bound to occur. Thus the change and importance factors need to be quantified before development. Moreover, project management is free to add more critical development factors in their own assessment process.

(c) Establishing Management Policy

To establish a set of ground rules from which to monitoring, management is encouraged to discuss the project objective and requirements with all team members at the early stage of software development. Besides, the selection of appropriate metrics and standards to measure the expenditure, productivity and performance will also be discussed. As a result, all team members have a better understanding of the overall development process, their roles and commitments. In other words, the whole development team will move towards the same goal. Examples of items that might appears on the list of management policy include:

No overtime
The frequency to do the monitoring

(d) Monitoring Mechanism

The basic idea for monitoring is to deal with changes that are imposed from inside or outside of the organisation.

A monitoring mechanism need to be established before for collecting all useful information and measuring variances from the work plans. This system should span the software development process from requirements analysis to integration testing. In addition, the basic intention of the system is to support two types of progress monitoring[10]:

(i) Milestone Monitoring

Upon the completion of a particular stage in the development cycle, several milestones had been established to evaluate the progress. Supposing “waterfall” life-cycle is applied, milestone monitoring will be set up at the end of the following stages:

- Feasibility/Requirements
- Analysis
- Design (Including high-level and detailed design)
- Coding
- Testing

(ii) Continuous Point Monitoring

Continuous Point monitoring is similar to milestone monitoring, but can take place at any time during a stage, and utilise whatever data happen to be available. The basic difference between these two mechanisms is that continuous point monitoring involves just one metric overtime and the time of monitoring is fixed.

Project managers are free to add and collect more information items in their own performance monitoring system. It is used to measure the overall and practical achievements of the software project against

the prescribed targets of the different stages of the development. Obviously, software development is a dynamic process and change is bound to occur. Thus all changes have been closely monitored and justified.

(e) Establishing Historical Database

It is essential to build a database to ensure continuous improvement on the estimation knowledge. Besides, the database is very useful in making the estimation. It helps the project management in considering the reuse of components of the existing software and determining the potential reusability of the software under design. In short, the database will facilitate “planning experience transfer” in the organisation.

Operation Phase

The operation phase includes the activities of monitor, evaluation and determination. The major management tasks in this phase are described below:

(a) Performance Monitoring

Using metrics and standards can keep track of the project progress of all activities in terms of time, size and quality. The performance monitoring mechanism operates continuously in the development process until the products finally delivered to the customer. To get better result, many researchers [4] suggest that the monitoring system operate via status and progress reports to compare actual progress with the plans based on the estimates. The major advantage is that the reporting stores, analyses and filters information of project progress. Unfortunately, software development is so complex and is developed by many intellectual people whose activities are hard to measure. Thus the reporting mechanism cannot sufficient to reflect the whole picture. The reason is simple - development staffs do not really like too

early upsetting the project management. In other words, problems can not be early reflected and the longer an error goes detected, the more expensive it is to correct. For example, the cost of correcting an error made during the initial analysis of a project is only one-tenth the cost of correcting a similar error after the system has been turned over to the customer [8]. To get a better monitoring, the authors believe that informal monitoring mechanism also play an important role and the following points are worth to be highlighted.

Measure the performance of a group rather than an individual.

Formal and Informal meetings with different teams are worth to be held during development cycle.

Observe the overall performance of an individual in informal basis

Pay more attention to the schedule of absence from work. Generally, if one person can not fulfill his work on schedule, he will cancel or re-arrange his leave schedule.

Based on written report, formal/informal meeting, and observation, the monitoring system can collect different kind of information relating to the actual progress of the development. In summary, the monitoring system assists the management to spot symptoms of problems during the development process.

(b) Evaluating Information and the Monitoring Metrics

The major idea of this stage is to assist the project managers to identify what problem really happened in the project based on all collected information only. In other words, all collected project progress data from the performance monitoring system must be evaluated and represented in a standard format. Thus it can be interpreted consistently by project management.

Before evaluation, the project managers have to review the evaluation metrics in order to make sure all measurement tools are good to identify or state the problem correctly. If the metrics can not truly reflect the real problem, the project managers need to adjust them and re-evaluate all information again.

Evaluating project progress can be regarded as a validation process to ensure that project progress information reflects the actual project situation. Based on all collected information, the expected progress in deliverables against actual progress will be examined and provides the 'rough' boundary of the problem. For example, schedule evaluation aims at comparing the actual expenditure of time with the planned schedule. After evaluation, project managers understand whether the project will be finished with an unacceptable schedule.

It should be reminded that some symptoms may be common to different problems or one symptom may be caused by several problems. For example, the symptom of overtime working may be caused by poor communication, poor job allocation, insufficient support, etc. The most important issue is that the project managers need to know problem has happened so that remedies can be devised.

(c) Determination

Obviously, decisions are essentially responses to problems occurring in software development. Once the evaluation metrics are found appropriate and the problem has been identified, the project managers must determine the boundaries of an acceptable solution and identify all possible courses of action. Considering all updated project information from the monitoring system and other relating factors, project manager needs to determine whether the project has deviated from the plan.

To achieve the goal, a two-step mechanism is adopted. The first step is to rearrange the project plan based on the updated project information. The project schedule and size plan re-construction will eliminate the self complementary effect. For example, the early completion of one task and the overrunning of another task may complement each other and make it difficult for the project managers to be aware of the problem.

Having a new project schedule, the second step is to consider all factors relating to the event in order to predict the future result. For example, low a productivity level may be accepted in the early stages of the software development cycle because team members have to spend time to understand the complex application. The project managers need to consider all possible factors before making decision. Thus in each decision making, the project managers explicitly or implicitly make assumptions about the future. If the event will cause the project to deviate from the target time or budget, some corrective actions must be taken.

In general, Acquisition phase would form part of the activity during feasibility/requirement stage of a "waterfall" software development life cycle model, while operation phase would certainly be operated continuously in the development process until the products finally delivered to the customer. The estimate at acquisition phase may be a rough' estimate based on limited information only and will be further refined during the development cycle if necessary.

5. CASE STUDY

In this section, an empirical case study will be presented to show how to apply the

SDM framework in real life software development. The project has been requested by a large electronic component trading company by an international software consulting organisation. This project has been designed to replace the existing manual telephone ordering system. The client requested to develop this system on a personal computer with Object-Oriented (OO) database design and programming and several advance remote client-sever transmission technologies. Besides, the client requested the remote client-sever system to be completed by a specified date without any overruns.

Briefly, the application was a new system and consist of two parts, namely Off-line system and On-line system.

- Off-line system

The off-line system allowed the user to store details of various requests, the user then must connect the customer terminal to the central computer system (orders via telecommunication process) to transmit the ordering information to the trading company in Hong Kong. This process could be done through a modem from any suitable telephone hookup in Mainland China or world wide.

- On-line system

The objective of this system was to monitor and control the ordering process. For example, the system allowed users to query the state of the account balance or display the information of the electronic components, such as price, terms of delivery, etc.

After an intensive investigation, a decision has been reached applying SDM framework for this application. The decision was made based on following reasons:

- The development team consisted a mix of experienced software consultant and some newly recruited graduates. They did not have much experience on the OO design and development.
- “Object-Oriented” is a new software development paradigm. This affects the accuracy for using many popular software size estimation techniques, such as Function Point, COCOMO, etc. Moreover, these metrics which are developed for conventional software have been found difficult to use in OO development environments.
- The deadline was tough and could not be extended.

In summary, there were many elements of uncertainty which might appear during development. Any one of these elements could mask the effects of other factors in estimation and cause the result to be unreliable. The company adopted to apply the SDM framework and it was expected to provide the management with more information and a reliable control.

(a) Feasibility Study:

To understand the status of the development process, a development analysis was performed and two kinds of issues, namely change and supporting, were considered.

Change issue:

Likelihood of change of requirement, Likelihood of change of development environment. Likelihood of change of development man-powers, Likelihood of change of support, etc.

Supporting issue:

Experience of staff, Knowledge of the user, Participation of the user,

Using a simple weighted rating scheme,

the project management determined this project as a medium level of change and low level of the supporting. The project management highlight the need to pay more attention to the change from inside support, such as the inexperience of the development staff.

By the end of this stage, the project management policy had been established and some of them, without going into detail, are listed below:

- Monitor development fortnightly
- The construction cost and development variance should be less than 5%
- No development delay and cost overrun should be allowed for Analysis phase and Design phase
- Report exceptions and actions
- Compare progress with schedule fortnightly

(b) Analysis Phase

Several monitoring metrics and performance standards were established. Standards were based on the organization’s past experience, the experiences of other similar organizations, the appropriate public standard, and some reasonable personal adjustment. Four major elements, namely size, time, quantity, and quality, were found in these standards. These elements made standards understandable and measurable. Two important elements were listed below:

i: Time:

The monitoring measurement need to compare the schedule with the delivered or completed items at the review time. If there was a schedule slippage, the project management must investigate the source of the problem.

ii: Budget:

The monitoring measurement need to

compare the actual work-content with the planned schedule. If the actual work-content (size) is not being closely monitored, all time-oriented problems will not be revealed until the project is finished with an unacceptable cost or time. In other words, extra money or time have been spent to accomplish the project.

(c) *Design Phase :*

From the monitoring mechanism, the project manager found that the productivity level as lower than (approximately 10%) the predefined standard. The major reason is that "Object-Oriented" analysis and design is a new software development paradigm, so programmers needed to spend a lot of time to learn how to apply it.

Considering all relating information, such as the stage of the slippage, the effort management determined that the slippage was due to the initial learning time required for understanding the complex application. Moreover, the performance of the system Analyst and programmer was found improving significantly. The project management made the decision that they would be able to complete the task on or before the target time. Thus no report needed to be submitted to the project manager but this matter had to be closely monitoring and recorded for further reference. Besides, the policy of monitoring had been changed to:

- Monitor development weekly
- Compare progress with schedule weekly

This new arrangement ensured adequate information supply and prompt evaluation/determination.

(d) *Implementation Phase:*

During this stage, the project management found that the productivity level was lower than (approximately 20%)

the predefined standard because two key programmers resigned halfway into the programming phase. This matter made a serious impact on the development process. The project management considered all relating information to determine that it is a problem which had to take corrective action as soon as possible. As this problem was found before causing a serious deviation, the project manager still had enough time to re-arrange the man-power in order to meet the schedule.

Finally, the project completed and delivered on schedule. The software project was eventually successful. It is not because there were no problems during the development cycle but because the problems have been found and overcome before causing serious deviation. The major task of this framework is the prediction of development efforts and monitor the progress of the development. Once a deviation has been found, project management need to determine the boundary of the problem and the corrective actions must be taken timely. This arrangement makes sure that the software project can be developed according to plan and to be ready exactly on time.

5. CONCLUSION

Experience has shown that most project managers seldom enjoy the luxury of excess resources and spare time in doing the cost estimates. Moreover, the resource constraints and different kinds of uncertainty cause a lot of difficulties in project cost estimating and planning. The success of a software project relies very much on a good management and control system which allows the development to satisfy the project objectives. However, the development of software project involves many factors which are hard

to manage in this ever-changing world. Some examples of these factors are human relationships, technological changes, frequency requests for changes by users, etc.

As mentioned early, software project development have a dynamic nature. Therefore, changes are bound to occur. This paper presents a framework to the software project management in order to tackle the problems from changing development. Generally, the SDM framework is developed to let project managers monitor the development process and make the project development running on schedule. The case study shows that change in personnel is common and not easy to be predicted. Applying the SDM framework, project manager monitors the development process and work closely together with team members. Surely it is too early to form final conclusions about the efficient and usefulness of the framework but the results from the one project so far are certainly encouraging. In order to use this framework successfully in practice, the following are important requirements:

- close working with team members
- understanding the needs of the users
- commitment of management before starting
- close monitoring of the development progress
- using different points of view to do evaluation at different development stages
- evaluation will be redone while change appears

The authors believe that the SDM framework and above requirements are beneficial in the monitoring and controlling the development cost of software projects in general. The use of the framework will likely result in improved the software development

for those project managers who choose to apply it.

ACKNOWLEDGMENT

The author would like to thank Dr. Chuk Yau for various insights concerning the ideas upon which this paper is built, and to the Mr. Vincent Cheng for his encouragement and support during the course of this research.

REFERENCES

- [1]Albrecht J. Allan and Gaffney E. John, "Software Function, Source Lines Code, and Development Effort Prediction: A Software Science Validation," *IEEE Trans. of SE.*, Vol. 9 No.6, November 1983
- [2]A.M. Jenkins, and J.C., Wetherbe, "Empirical investigation of systems development practices and results," *Inform. Manage.*, 1984
- [3] Capers Jones, "The Productivity Report Card," *Software News*, Sept., 1986
- [4]P. Rook, "Controlling Software Projects," *Software Engineering Journal*, Vol. 1, No. 1, January 1986
- [5]Charles R. Symons, "Function Point Analysis: Difficulties and Improvements", *IEEE Trans. on Software Engineering*, Jan. 1988
- [6]D. Phan, D. V. & J. Nunamaker, "The search for perfect project management," *Computerworld*, 1988
- [7]Heemstra, F J, "Software Cost Estimation", *Information and*

Software Technology, Vol. 34 No 10
October 1992

software development”, *Software Engineering Journal*, January 1989

[8] P. Goodman, *Practical Implementation of Software Metrics*, McGRAW-HILL Book Company, 1993

[11]E.R. Gray & L.R. Smeltzer, *Management the competitive edge*, Collier Macmillan Publishers, 1989

[9]N. Ashley, *Measurement as a powerful software management tool*, McGRAW-HILL Book Company, 1994

[12]Capers Jones, *Applied Software Measurements*, McGRAW-HILL, NY, 1991

[10]Barbara A. K. and J. G. Walker, “A quantitative approach to monitoring

[13]FJ Heemstra, “Software Cost Estimation”, *Information and Software Technology, Vol.34*, 1992
