A Supervised Learning System for Software Project Management*

Chien-Kuo Bien, Chin-Yi Tsai, and Chua-Huang Huang
Department of Information Engineering and Computer Science
Feng Chia University
Taichung, Taiwan
ckbien@fcu.edu.tw, {cyt, chh}@pmlab.iecs.fcu.edu.tw

Abstract

Software engineering technologies are composed of software development technique and project management. In the last decades, a variety of software development techniques have been proposed to improve software development ability and quality of software system. However, due to the software systems become more and more complex, we must have an efficient methodology to manage, monitor, and control the software project. Also, it is expected that organizations would be able to accumulate knowledge from the previous software project. Hence, we propose a supervised learning system for software project management as well as a methodology process. Based on previous project experiences, the system could provide some useful suggestions and software size estimation pertaining to new projects. In summary, the supervised learning system aims to foster more precise estimation and efficient management for software projects by means of accumulating experiences and analyzing collected results.

Keywords: software engineering, software measurement, software cost estimation, software project management, CMMI, supervise learning.

1: Introduction

In general, software development consists of requirements analysis, system analysis, system design, implementation, and testing. There are many software development methods have been proposed in recent decades, including Waterfall [15], Spiral [5], Iterative and Incremental [12], RUP [11], XP [2], *etc*. These proposed software development methodologies concentrate on development issues, including analysis and design technologies as well as development process. However, in addition to the improvement of software development, managing and monitoring software project is important as well. Recently, due to the explosive increase of software complexity, software project management becomes more and more important. There are a lot of uncertain factors may cause the failure of software project during the development of software system. Even though these factors do not result in the failure of software project eventually, they may also give rise to the increase of cost and the down of software quality. Hence, project managers play an important role during the development of software system. The responsibility of project manager is to ensure the accomplishment of software project. In addition, the finished software system should achieve some degree of accepted quality.

According to Project Management Institute (PMI), the definition of project is that a project is a temporary endeavor undertaken to create a unique product, service, or result [9]. Based on Kerzner, project management is a systematic planning, scheduling, and controlling process to ensure the success of software project [10]. Besides, several limits should be considered, including time, cost, human resource, and quality. Since there are several constraints and risks on software project, project should be under monitor and control constantly while project is executing. Before that, project manager must estimate software size, plan the project scheduling, allocate resource, and assign tasks, *etc*. Usually, an experienced project manager has the ability of planning and controlling the project well. However, we can not depend on an experienced project manager completely since the ability and knowledge belong to an individual person, not to organization. There are several institutions propose and establish standard, model and guideline regarding project management. Software Engineering Institute of Carnegie Mellon University establishes Capability Maturity Model Integration (CMMI) in order to improve the software process and quality [6]. Considering project management, the most well-known institute is Project Management Institute (PMI). PMI publishes a guideline entitled, *A Guide to the Project Management Body of Knowledge*, for project management [9]. IEEE and EIA jointly make software life cycle processes standard IEEE/EIA 12207 [8]. Additionally, PRoject In Controlled Environment (PRINCE) was released by England government and used in Europe mainly.

In this paper, we propose a supervised learning system for software project management as well as a methodology process. This process contains six activities, including *object module identification*, *object*

*This work was supported in part by National Science Council, Taiwan, R.O.C. under grant NSC 95-2221-E-035-046.

module matching, factor weighting, project check point construction, *project execution* as well as *project learning, monitoring, and control*. Based on previous development experiences and historical data about projects, not only project managers but also an organization is able to learn and accumulate knowledge about project development. These learned knowledge is useful for project manager in project estimation. These information derived from previous development experience provides applicable suggestion of project estimation and planning. This paper concentrates on how to accumulate the historical experiences from the previous projects. In addition, it also can learn from the historical experiences and provide reliable suggestions for future projects. Based on the gathering historical data from project database, the proposed system could analyze and quantify the causes of variation and add parameters of foreseeable factors to the project. Also, it is able to make prediction and give suggestions for pending project. It is expected to provide a reference for project managers to fine-tune the parameters according to the actual conditions in order to satisfy the practical requirements.

The organization of this paper is as the following. In Section 2, we represent the proposed supervised learning mechanism and our project management methodology process as well. In Section 3, we will show and explain data analysis result. The resulting analysis can be divided into three kinds of conditions, including ideal, actual, and special conditions. The prototype software architecture and main functionalities are given in Section 4. Conclusions are given in Section 5.

2: Supervised Learning for Software Project Management

Members of a software project can be divided into project developer and project manager roughly. Project manager is responsible for planning and managing the software project. However, how to plan, manage, and monitor the software project efficiently is always a difficult problem. We especially focus on supervised learning for assisting a project manager in planning and managing software projects. Moreover, it is useful for both experienced and unexperienced project managers because of supervised learning mechanism. In the following we will depict the proposed system and methodology process as well as the learning mechanism.

2.1: Software Size Estimation

After understanding the requirements of software project, a project manager's task is to estimate how much time the software project required. There are already several approaches for estimation of software size existed. The most straightforward method for measuring the size of software is to count the line of source code [1]. However, such approach can not help project manager for estimating software size in advanced. Recently, some academic groups and industries adopt function point analysis (FPA) in estimating software size [7]. In addition, Boehm constructs a model COCOMO for estimating software size and cost [3]. The lastest version of COCOMO is COCOMO II [4]. It mainly adds some weighting factors and uses FPA for software size estimation.

In our software sizing method, we especially emphasize project historical data. Additionally, we use several weighting factors to improve the accuracy of software size estimation. Firstly, decomposing software into smallest object modules. Secondly, referring to historical data and finding the most similar one in response to previous decomposed object module. Hence, we can obtain the initial suggested object estimation. Finally, in order to improve the accuracy of estimation, we use several weighting factors by the formula $f \quad w$, where f is the suggested object time and w is the weighting value. There are two kinds of weighting factors. They are scale and cost drivers. The candidate factors are as below:

- Scale Drivers
  - whether there are similar experience or not
  - feasibility of design
  - relationship between various related persons
  - status of organization
- Cost Drivers
  - ability of analysis
  - ability of programming
  - experience of platform
  - familiarity of programming language and tool
  - human resource
  - reliability of software
  - complexity of software product
  - reusability
  - need of software document

2.2: Scheduling and Monitoring of Software Projects

After the suggested values of project schedule are derived, the next step is to consider how to connect the suggested project schedule to project management and monitoring mechanisms. We utilize object-oriented programming to decompose the software program into multiple objects and derive the suggested time for developing each object. In addition, a detailed time table for developing these object be constructed. Then, we can do the activity of project monitoring according to the time table. When the delay of project progress occurs, it can be identified immediately and adjust the follow-up schedule or increase human resource to solve the delay. During the checking of milestone, if any delay be found, the system will send a message to related staffs. Besides, it also records the cause of the delay in project database.

2.3: Learning Mechanism

During the execution of software project, there is a project database records the related project data [13, 14]. Hence, after completing the software project, the related project data can be analyzed and regards as a reference for new projects. The accumulation of a large amount of information can provide accurate estimation about the project. The useful project related data as below:

- **Staff Ability**: If a project delay is observed in the stage of audit, the system will record the delay point. Also, this situation requires project manager to tender an explanation and record the entire management process. If the delay is attributed to the ability of staffs, it is necessary to check up the historical record of the staff and verify whether the staff is familiar with the current position or project. If it is true, the project manager's improper selection of the staff and negligence over his ability during project estimation will be recorded. If the delay is purely attributed to the staff's problem, the system will record the incompetence of the staff and properly adjust his ability index.
- **Accident**: If the project delay results from an un-predicatable accidient, project manager is required to record the cause of the accident. This record is recorded as a reference for estimating risks in the future project.
- **Error of time suggested by the system**: If project delay still happens while the suggested estimation time be used, moreover, the cause is not human factor and unpredictable accident, it is necessary to make adjustments according to the delayed project data so as to avoid the erroneous information to be provided within new projects.

2.4: Methodology Process

A supervised learning mechanism for software project management make use of the historical data to help project manager in project estimation and planning. In addition, it is also useful in project monitoring and control. In the following we will describe the proposed methodology process pertaining to software project management. The methodology process is composed of six activities and four kinds of artifacts as showed in Figure 1. The detailed descriptions of proposed methodology process are stated in the following.

1. **Object module identification activity**: First of all, we have to decompose software into object modules. The decomposed object module is the smallest unit in software. After decomposing well, we can obtain the artifact, identified objects.
2. **Object module matching activity**: After obtaining the identified objects, the next activity is to do the matching operation. This object module matching operation will find the most similar object from project historical database. If we find the most similar object module, we can determine the time of similar object module in historical project database as the estimation of object module we identified previously. Thus, the result is the artifact suggested object estimation.
3. **Factor weighting activity**: In order to make the estimation more accuracy, we can choose several weighting factors and multiply the suggested estimation. After that, we can obtain the weighted object estimation.
4. **Project check point construction activity**: Before project executed, project manager have to construct check points and project scheduling. According to the check points, project manager can do the milestone check. In addition, project manager is able to monitor and control the project with these check points.
5. **Project execution activity**: This is the activity that all project member do their assigned task.
6. **Project learning, monitoring, and control activity**: During the executing of project, project manager must monitor and control the progress of project. In the meantime, project manager receive several feedback from project member or project. These information is very useful to project manager. Project manager can make a correct decision according to these information.

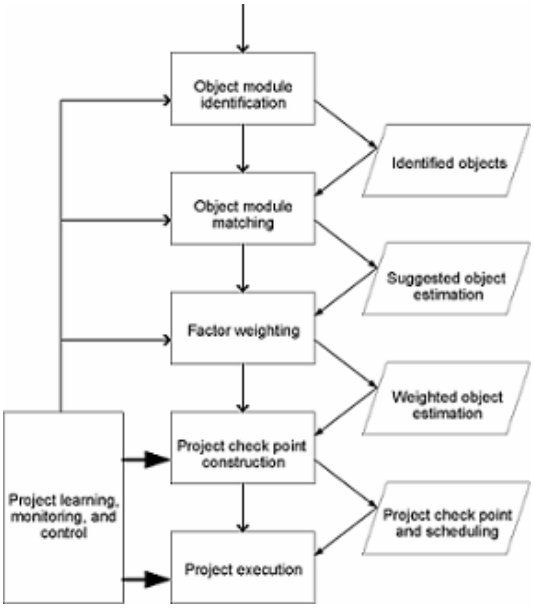

Figure 1: Methodology Process

3: Data Analysis

In this section, we will analyze the time curve, so-called learning curves, by historical record of one object module. The recorded data is the pure executing time, which practical executing time minus weighted time. The learning curves can be classified into three kinds of condition, ideal, actual, and special conditions. The detailed descriptions are stated in the following.

3.1: Ideal Condition

In ideal condition, the time curve is similar to ladder as showed in Figure 2. Initially, a developer constructs an entirely new object module. Usually, the developer has to spend more time on this new object module since the object module is constructed from scratch. However, as time goes on, developer use this object module with less and less modification. Also, the developer becomes more familiar with this object module. Hence, developer will spend less time on this object module gradually. Eventually, the variation in time for using this object module will become small. A project manager will be able to estimate the required time accurately when software project need to use this object module.

Under the perspective of manager, the project manager can manage and control the object more easily when the object module required time become stable. When software project needs the function of the object module, the project manager can easily estimate the required time and determine the time will spend on this object module. Usually, there is only a little variation happened between the estimated time and the actual executing time. However, if the actual executing time exceed the estimated time greatly, the project manager finds out the reason and manage to solve this problem. In doing so, the executing ability of whole project will be improved.

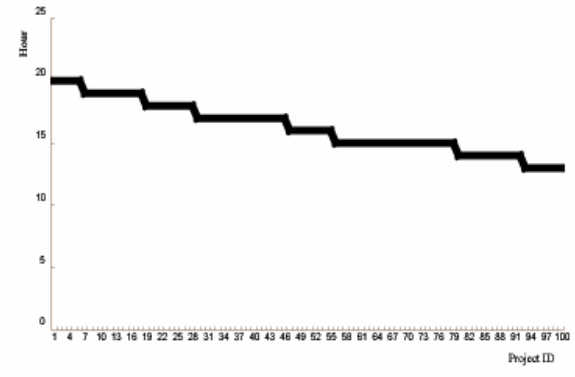

Figure 2: Ideal Condition

3.2: Actual Condition

In actual condition, the time curve is not as smooth as ideal condition's curve. However, the time spend on this object module will also become stable. The time curve as showed in Figure 3 reflects that a different developer has different ability and spends different time on the same object module.

It is necessary that project manager must understand the usage of object module previously and consider people's experience on this object module. For example, an unexperienced person may spend much time on a stable object module. However, the project manager can assign this task to experienced person to avoid this problem. In Figure 3, there is one point increase suddenly. There are may two reasons for that. One reason is that the object module is used by novel member. Another point is that there is some modifications executed on this object module.

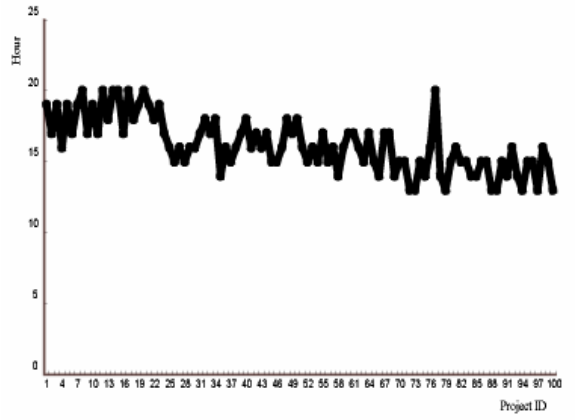

Figure 3: Actual Condition

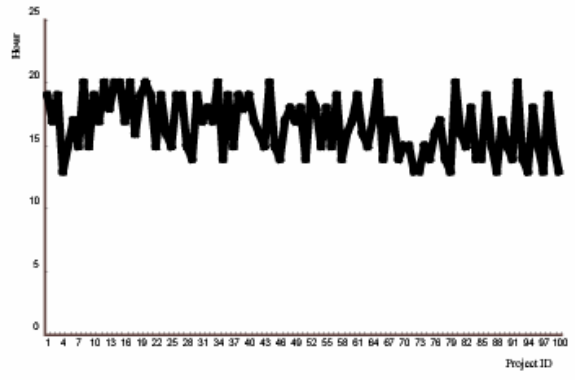

Figure 4: Special Condition

3.3: Special Condition

As for special condition, the curve is always oscillated as showed in Figure 4. The required time to handle the object module can not become more stable gradually. The possible reason is that the difference of familiarity with object module among project members is great. In addition, object module is adjusted frequently is also possible. Hence, organization should be adjusted in order to improve the quality of software and project.

4: Prototype Implementation

In this section, we will describe the prototype of supervised learning system as well as demonstrate the main functionality of the prototype.

4.1: Prototype Software Architecture

The software architecture of supervised learning system is composed of supervised learning system and project database. There are two kinds of roles, project manager and project member, will use the system. According to the role type, they can use different

functions in the supervised learning system. The system consists of two subsystem as in the following:

- Management subsystem
  - Project setup module: This module mainly is in charge of the initialization of a software project.
  - Estimation module: Providing the functionality of software estimation.
  - Learning module: Performing the learning mechanism.
  - Scheduling and monitoring module: Providing project manager for scheduling and monitoring the project.
- Development subsystem
  - Progress report module: Reporting current progress to project manager.
  - File management module: Managing all project files.
  - Configuration management module: Managing project configuration.

4.2: Prototype Functionality

In the following we will demonstrate the main functionalities in supervised learning system, including *project creation*, *project estimation*, and *project monitoring and control*.

- **Project creation**: Initially the head leader of the department assigns a project manager to creating a project. Then, the project manager have to select the members which will involve in the project, including requirements analyst, system analyst, system designer, and programmer, and so on. However, it is difficult to select adequate members if we do not have any reference record pertaining to members. Hence, we must select the members based on the available reference record. After that, the project manager has to construct further planning, including project planning, estimation, delivery milestone, customer's requirements, and so on. In the initial stage, the project manager may not be able to comprehensively plan all the necessary planning and setting for a totally new project. Even that, the project manager may still plan a general list of required planning modules and set a deadline for all planned modules. Later, system analysts, who will conduct the detailed analysis of the software system, make the planning modules more detailed based on the deadline made by project manager. They can require the project manager to adjust the planning if any insufficient time is found.
- **Project estimation**: In project planning, the project manager has to estimate the required working time based on the project manager's understanding about the project. In first, the project manager lists all the objects expected to be adopted in the project. Then, the project manager starts to set the factors of the projects. After that, an approximate suggested value of working hours can be derived.
- **Project monitoring and Control**: It is important to monitor and control the project during the project development. There are lists of modules planned by the project manager serves as milestone. Hence, the activity of monitoring is based on the milestone. Every module has a current completed time. The system will check the project according to the scheduled time. While discovering the incomplete, the system will inform the project manager to request staffs who are responsible for the delayed module to explain the delay. If the delay is attributed to human factor or incompetence, the information should be fed back to the personnel database for proper adjustment of the staff's ability rating. Hence, project monitoring is an important task.

5: Conclusions

Software estimation is an essential work for a software project though it may require additional time and cost. However, it is worth since accuracy software estimation can facilitate the improvement of quality of software system and software project as well. In contrary, inaccuracy software estimation may cause the increase of cost. Moreover, it probably results in the failure of software project. Hence, we must have an appropriate approach to estimate software size. We firstly decompose software into object modules. Then we utilize historical data to improve the accuracy of estimation.

In addition to software estimation, planning a project plan is also important. A project manager not only plans the project scheduling, resource allocation, task assignment, but also monitors and controls the software project during the execution of software project. Actually, a project manager plays an important role and has an influence on whether the software project is successful or not.

Hence, in this paper we propose a supervised learning mechanism for software project management. By means of the historical data, we can improve the accuracy of estimation. In addition, a project manager can get some useful information during the monitoring and executing of software project. According to the obtained information, a project manager is able to make a proper decision, including re-changing the estimation and project scheduling.

References

[1] A. J. Albrecht and J. R. Gaffney. Software function, source lines of code, and development effort predication: a software science validation. *IEEE Transactions on Software Engineering*, 9(6):639–648, 1983.

[2] K. Beck. *Extreme Programming Explained*. Addison Wesley, 1999.

[3] B. Boehm. *Software Engineering Economics*. Prentice Hall, 1981.

[4] B. W. Boehm, C. Abts, A. W. Brown, and S. Chulani. *Software Cost Estimation with COCOMO II*. Prentice Hall PTR, 2000.
[5] W. Boehm. A spiral model of software development and enhancement. *IEEE Computer*, 21(5):61–72, 1988.
[6] CMMI Product Team. Capability maturity model integration (cmmism), version 1.1. Technical report, Software Engineering Institute, Carnegie Mellon University, 2002.
[7] I. F. P. U. Group. *Function Point Counting Practices Manual, Release 4.2*. Westerville, 2004.
[8] IEEE/EIA 12207.2-1997. IEEE/EIA Standard: Industry Implementation of International Standard ISO/IEC 12207:1995 Standard for Information Technology– Software Life Cycle Processes–Implementation considerations, 1997.
[9] P. M. Institute. *A Guide to the Project Management Body of Knowledge, Third Edition*. Project Management Institute, 2004.
[10] H. Kerzner. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling, Eighth Edition*. Wiley, 2003.
[11] P. Kruchten. *The Rational Unified Process: An Introduction, Third Edition*. Addison-Wesley, 2003.
[12] C. Larman and V. R. Basili. Iterative and incremental development: A brief history. *IEEE Computer Society*, 36(6):47–56, 2003.
[13] C. S. M. Shepperd. Estimating software project effort using analogies. *IEEE Transactions on Software Engineering*, 23(12):736–743, 1997.
[14] Myrtveit and E. Srensrud. A controlled experiment to assess the benefits of estimating with analogy and regression models. *IEEE Transactions on Software Engineering*, 25(4):510–525, 1999.
[15] W. W. Royce. Managing the development of large software systems: Concepts and techniques. In *Proceedings of WESCON*, 1970.