ソフトウェア工学　82－9
（1 9 9 1.　1 2.　3）

Measurement-based Controlling in Project Management System

Ken-ichi Matsumoto　Shinji Kusumoto　Tohru Kikuno　Koji Torii
Faculty of Engineering Science, Osaka University

The activities for controlling a software development project consist of the followings: (1) knowing an actual state of the project, (2) clarifying the difference between the prescribed plan and the actual state of the project, and (3) helping the developers to accomplish the prescribed plan.

This paper proposes a project management system with a measurement-based controlling subsystem which supports these three activities based on quantitative and objective data collected from the project. The system has tools for data collection from the project, data analysis of productivity and quality of the software, and information feedback to developer, project manager, and experience manager respectively. Additionally, during the project execution, the system invokes them according to three kinds of scripts which explicitly describe these control activities.

定量的プロジェクト管理のためのプロジェクト制御支援システム

松本健一　楠本真二　菊野亨　鳥居宏次
大阪大学　基礎工学部

本稿では，定量的なプロジェクト管理を目指した支援システムのサブシステムとして現在開発を進めているプロジェクト制御支援システムの構成と主な機能について述べる．開発中のシステムでは，プロジェクトの制御プロセスはある形式のスクリプトとして定義されており，システムはそのスクリプトを解釈，実行することによって，プロジェクト制御に必要なデータの収集，分析，及び，フィードバックを行う．システムによって収集，分析されたデータはプロジェクトの制御だけではなく，プロジェクト実施中に行われる計画の変更にも利用される．また，そのデータはプロジェクトで得られた経験としてパッケージ化され，将来のプロジェクトのための開発計画の作成と制御に利用される．

1. Introduction

Large software systems often provide incomplete functionality for what customers want, take too long time to construct, cost too much, use too much memory space or other resources to run and rarely evolve to meet the changes needed[4]. These problems associated with development of software, especially large-scale software, have emphasized the need for a more disciplined and systematic approach. Software project management is one systematic approach to produce cost-effective, reliable software within specified time constraints[2].

Two major functions of software project management are "planning" and "controlling."[5][6] Planning is executed before constructing the actual software product. The purpose is to enable the objectives of the project to be accomplished effectively. On the other hand, controlling is executed dynamically during the software construction phase in the project and is done to carry out the project in pursuance of the prescribed plan if deviation from the prescribed plan occurs.

This paper proposes a project management system with a measurement-based controlling subsystem and describes characteristics, architecture of the controlling subsystem. The subsystem supports three major control activities: (1) knowing the actual state of the project, (2) clarifying the difference between the prescribe plan and the actual state of the project, and (3) helping the developers to accomplish the prescribed plan based on quantitative and objective data collected from the project. It also provides the project managers with valuable information for modifying the prescribed plan and the experience managers with valuable information for packaging experience acquired in the project.

The main characteristics of the proposed system are summarized as follows:

(1)Script-based controlling

The control plan, which is constructed by the project manager and provided for the measurement-based controlling subsystem, includes three kinds of scripts: script of data collection, script of data analysis, and script of information feedback. The measurement-based controlling subsystem interprets them and invokes tools for data collection, data analysis, and information feedback. Therefore controlling process becomes more explicit and understandable than the process in conventional system for project controlling.

(2)Three types of feedback

There are three kinds of feedback from the measurement-based controlling subsystem. First type of feedback is provided to developers in order to control the project, the second one is to project manager in order to modify the project plan, and the last one is to experience manager in order to analyze and package experience acquired in the project as a new experience.

Section 2 shows major functions of software project management and concept of measurement-based controlling. Section 3 describes project management activities and major characteristics of project controlling on the proposed system. Section 4 presents an architecture and major functions of the controlling subsystem which is one of the major subsystems of the proposed system. Finally, Section 5 summarizes the ideas discussed in this paper and presents some areas for future research.

2. Project Management

2.1 Major functions

In the classic management model, management is partitioned into five separate functions or components: planning, organizing, staffing, directing, and controlling[6]. These functions can be classified into two types. The first type includes planning, organizing and staffing. These are executed before constructing the activities of the software project. Their purpose is to enable the objectives of the project to be accomplished effectively. The second type of functions includes directing and controlling. These are executed dynamically during the software construction phase in the project and are done to carry out the project in pursuance of the prescribed plan if deviation from the prescribed plan occurs.

2.2 Measurement-based controlling

Controlling a software development project is defined as all the management activities ensuring that the actual work goes according to plan[6]. To control the project, the manager has to know the actual state of the project, clarify the difference between the prescribed plan and the actual state, and help the developers accomplish the prescribed plan.

In measurement-based control for the software development project, data collection, data analysis, and information feedback are essential activities. The manager can assess the actual state of the project by collecting data from the project and analyzing them. In addition, the manager can help developers correct their activities and accomplish the prescribed plan by providing the analysis results the packaged experience in the past project as feedback information[1][8].

For complex software, this is extremely important because it allows us to discern trends and pattern[3]. The measurement of resource expenditures is a good example of the benefit of this type of description. The resources expended on a project, particularly in terms of a human effort, are translated directly into costs. By collecting and analyzing information about exactly where these resources are being expended (for example, what phase of the life cycle, what types of activities, what parts of the system), one can identify the major cost "drivers" within a software organization. Then, one can answer questions such as "What types of activities consume large portions of the available manpower?" and "Where is the effort being wasted?"[3] Therefore it can lead to the instructions for the developers or modification of the prescribed plan.

3. Project Management on Proposed System

3.1 Outline of project management

Figure 1 shows outline of project management on a proposed system. The proposed system consists of two subsystem: "Planning Subsystem" and "Controlling Subsystem."

Activities of project management on the system is divided into the following three steps[1]:

Step 1: Planning

"Project Manager" constructs a project plan based on the users' requirements for functions, performance, and qualities of software to be developed. "Experience Manager" who maintains the packaged experience, helps the Project Manager construct a project plan by providing the packaged experience. In some cases, he includes organization's require-

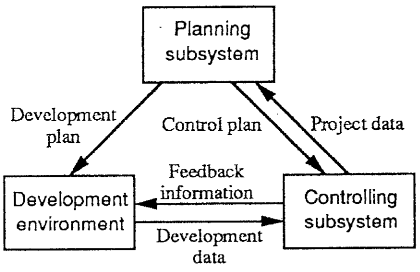

Figure 1 Project management on the proposed system

ments in the project plan.

The project plan consists of a development plan and a control plan. The development plan specifies development process, activities to be executed by "Developers", schedule to be kept in the project. On the other hand, the control plan specifies data collection process for recording the actual state of the project, and data analysis process for knowing the actual process of the project , and information feedback process for providing valuable information for the Developers, the Project Manager, and the Experience Manager.

Step 2: Executing and Controlling

The Developers construct software in development environment according to the development plan which specifies the development process and the activities to be done, and the Controlling Subsystem collects and analyzes data, which represent the activities of the Developers, from the development environment according to the control plan. The Controlling Subsystem also provides valuable information for the Developers and the Project Manager according to the condition specified in the control plan.

Step 3: Packaging

When the project has been finished, the project plan, collected and analyzed data from the development environment, and feedback information provided for the Developers or the Project Manager are given to the Planning Subsystem from the Controlling Subsystem. And then, the Experience Manager analyzes and packages them as new experience of planning, executing (development) and controlling in the software development project.

3.2 Script-based controlling

The control plan constructed at Step 1 mentioned in subsection3.1, is specified by three kinds of scripts:

```
#define TIME_INTERVAL 30min

#define TESTDATA /usr/local/testdata

record date, calendar_time;
list_of_developer = search_logon_file;
record list_of_developer;
          :
if (phase(now) == UNIT TEST) {
   for each developer in list_of_developer {
      record developer;
      list_of_program = search_program;
      record list_of_program;
      for each program in list_of_program {
         record program;
         record testing_effort;
         exec test_coverage(TESTDATA);
         record coverage, testdata_usage_rate;
         record list_of_used_testdata, list_of_unused_testdata;
      }
   }
} else if ....
```

Figure 2 Example of script of data collection

```
#define TIME_INTERVAL 120min

record date, calendar_time;
          :
if (phase(now) == UNIT TEST) {
   for each developer in list_of_developer {
      record developer;
      total_testing_effort_est = 0;
      for each program in list_of_program {
         total_testing_effort_est +=
                  testing_effort_estimation(
                        coverage, testdata_usage_rate);
      }
      record total_testing_effort_est;
   }
} else if ....
```

Figure 3 Example of script of data analysis

```
#define TIME_INTERVAL 120min

#define COVERAGE 90%
#define USAGE_RATE 100%
#define TESTING_EFFORT 1000min

record date, calendar_time;
          :
if (phase(now) == UNIT TEST) {
   for each developer in list_of_developer {

      if ( ( average(coverage) < COVERAGE) &&
         ( average(usage_rate) < USAGE_RATE) &&
         ( (total_testing_effort_est - TESTING_EFFORT)
                              /TESTING_EFFORT < -10%) ) {
         feedback_to_developer
                  (coverage, average(coverage), COVERAGE,
                  usage_rate, average(usage_rate), USAGE_RATE,
                  list_of_used_testdata, list_of_unused_testdata,
                  total_testing_effort_est, TESTING_EFFORT);

      } else if ( ( average(coverage) < COVERAGE) &&
         ( average(usage_rate) >= USAGE_RATE) &&
         ( (TESTING_EFFORT - sum(testing_effort))
                              /TESTING_EFFORT > 20%) ) {
        feedback_to_manager
                 (developer,
                  coverage, average(coverage), COVERAGE,
                  usage_rate, average(usage_rate), USAGE_RATE,
                  list_of_used_testdata, list_of_unused_testdata,
                  sum(testing_effort), TESTING_EFFORT,
                  experience No. 235
                   % Effects of insufficient coverage on project
                  experience No. 242
                   % Plan to improve coverage);
      } else if ....
          :
} else if ....
          :
else if (phase(now) == PROJECT_END) {
   feedback_to_analyst (all records);
}
```

Figure 4 Example of script of information feedback

script of data collection, script of data analysis, and script of information feedback. These scripts are described by using the language we have been designing.

(1) Script of data collection (see Figure 2)

The script of data collection specifies what kind of data should be collected and when and from where they should be collected. But it does not provide any concrete mechanisms for data collection or what kind of tools should be used for data collection.

The Controlling Subsystem interprets the script and invokes data collection tools in the system.

(2) Script of data analysis (see Figure 3)

The script of data analysis specifies what kind of values should be calculated and what timing should they be calculated. But it does not provide any concrete mechanisms for data analysis or what kind of tools should be used for data analysis.

The Controlling Subsystem interprets the script and invokes data analysis tools in the system. These calculations are executed by using data collected by the Controlling Subsystem according to the script of data collection.

(3) Script of information feedback (see Figure 4)

The script of information feedback specifies conditions for executing information feedback and what kind of information should be fed back to the Developers, the Project Manager, and the Experience Manager. But it does not provide any concrete mechanisms for information feedback. The Controlling Subsystem interprets the script and invokes information feedback tools in the system. Information to be fed back are made from the collected and analyzed data by the Controlling Subsystem according to the scripts of data collection and data analysis.

3.3 Three types of feedback

The Controlling Subsystem provides three different types of feedback information for the Developers, the Project Manager, and the Experience manager, respectively, in order to accomplish the project plan effectively and package collected and analyzed data as a new experience in Planning Subsystem. What kinds of information should be fed back and to whom as well as the timing of feedback are defined by the script of information feedback in the control plan as mentioned in subsection 3.2. The contents of these feedback information are summarized as follows:

Type I: To Developer

During the Step 2 mentioned in subsection 3.1, if deviation from the prescribed plan occurs and the deviation is considered so small that the Developers can correct their activities immediately, the Controlling Subsystem provides the Developers with the required information. If any experiences that are helpful for the Developers to correct their activities, are prescribed in the control plan, they are also fed back to the Developers.

For example, let us consider the following case that a developer has to realize 95% of test coverage for his/her program but the actual test coverage is 80%. If these are some test cases which have not yet been applied to his/her program and he/she has enough time to apply them to his/her program, then the Controlling Subsystem provides the developer with information about test coverage and let him/her know that there are test cases which should be applied to his/her program.

Type II: To Project Manager

During the Step 2 mentioned in subsection 3.1, if the difference between the prescribed plan and the actual state of the project becomes large and it is considered so large that the Developers can not remove the difference only by themselves, then the Controlling Subsystem provides the Project Manager with information on what kinds of deviation has been occurred in order to modify the project plan. The project plan is also provided for the Project Manager. If some experiences that are helpful for the Project Manager to modify project plan, are prescribed in the control plan, they are also fed back to the Project Manager.

For example, let us consider again that a developer has to realize 95% of test coverage for his/her program but the actual test coverage is 80% in this case. If he/she has already applied all the test cases provided by the development plan and he/she has not enough time to create additional test cases for his/her program, then the Controlling Subsystem alerts the project manager about test coverage and lets him/her know that there is no test case which should be applied to his/her program and he/she has not enough time to create additional test cases for his/her program in order to urge the project manager to modify the project plan.

Type III: To Experience Manager

At the end of the project, the project plan, collected data, analyzed data, feedback information, and so on are sent to the Experience Manager in order to be analyzed and packaged as new experience of the software project.

4. Controlling Subsystem

Figure 5 outlines the system architecture of the Controlling Subsystem, which also describes interactive information processing among the Planning Subsystem, the Controlling Subsystem and the Development Environment. The data flow is also explicitly given in Figure 5. The Controlling Subsystem consists mainly of five logical units: "Plan Interpretation", "Data Collection", "Data Analysis" and "Information Feedback," and "Data Management." The architectural components of each unit are described in detail below[5][7].

(1) Plan Interpretation Unit

The Plan Interpretation Unit interprets three kinds of scripts in the control plan and invokes tools in the Data Collection Unit, Data Analysis Unit, Information Feedback Unit according to these scripts.

(2) Data Collection Unit

The Data Collection Unit consists of two components, (CM1) through (CM2):

The (CM1) Process Data Collection component accumulates all data concerned with development process, which are passed from a process manage-

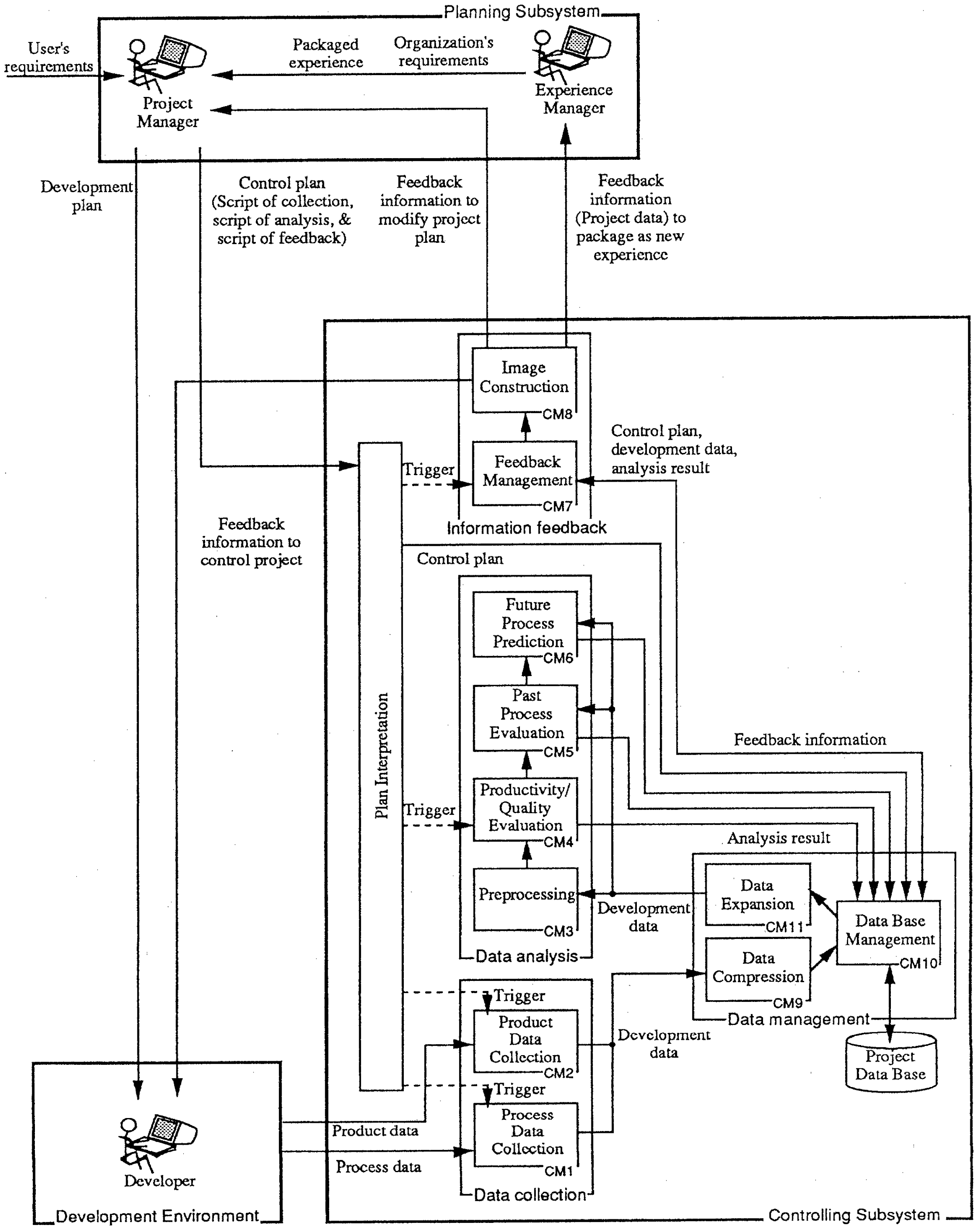

Figure 5 System architecture

ment system in development environment.

The (CM2) Product Data Collection component accumulates a series of intermediate product (including the resulting product) and collects historical data about product modifications. Product data collection also occurs indirectly through a product management system in development environment.

(3) Data Analysis Unit

The Data Analysis Unit consists of four components, (CM3) through (CM6):

The (CM3) Preprocessing component prepares the expanded data for evaluation. Preprocessing includes transforming data that were collected in the physical unit (i.e., file) into data for the logical unit (i.e., module).

The (CM4) Productivity/Quality Evaluation component calculates several values according to the algorithms or guidelines for measurement. The evaluations are assumed to be for purposes of productivity and quality of the software.

The (CM5) Past Process Evaluation component calculates a historical data of the past process, that begins at the start of the project and ends at the current time. The calculation is executed based on the values of development process which are transmitted from (CM4). The historical data are stored in Project Data Base, and used on the successive calculations of historical data.

The (CM6) Future Process Prediction component estimates a future process, that begins at the current time and ends at the prescribed date (deadline) of project completion. The estimation is performed by using the historical data of the past process which are transmitted from (CM5).

(4) Information Feedback Unit

The Information Feedback Unit consists of two components, (CM7) and (CM8):

The (CM7) Feedback Management component retrieves elements of feedback information from the Project Data Base, and constructs overall of feedback information by using them. The feedback information is sent to the (CM8) Image Construction component.

The (CM8) Image Construction component builds up an image of feedback information which consists of the target value of productivity or quality of the project along with the historical data of the past process, and the predictive value along with the data of the future process. Figure 6 shows a conceptual drawing of a feedback information provided by the Image Construction component.

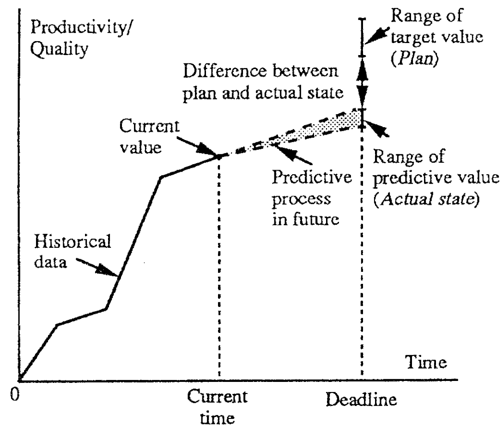
Productivity/
Quality
Range of target value (*Plan*)
Difference between plan and actual state
Current value
Range of predictive value (*Actual state*)
Predictive process in future
Historical data
Time
0
Current time
Deadline

Figure 6 Conceptual drawing of feedback information to developer

(5) Data Management Unit

The Data Management Unit consists of three components, (CM9) through (CM11):

The (CM9) Data Compression component saves memory by storing as little data as possible. For example, all of the updated files in software development and their update times are compressed into the histories of modifications and the latest version of the file. The history of modifications are accumulated by computing a difference between the new version of the file and the latest version of the file by using the file comparator when the file is updated within the past five minutes.

The (CM10) Data Base Management component provides data storage and information retrieval. The data from the Data Collection Unit and the information from the Data Analysis Unit are stored in the Project Data Base. The relevant information, such as the date and the names of project, team, developer, file, module, and statement, is added to each original data.

The (CM11) Data Expansion component restructures the original data based on the compressed data created in the (CM9) Data Compression component.

5. Conclusion

This paper proposed a project management system with a measurement-based controlling subsystem. The controlling subsystem interprets the control plan generated by the planning system and provides valuable information for the developer, the project manager, and the experience manager.

We have a plan to develop a prototype system of the proposed project management system and apply it to the academic and industrial field in order to evaluate usefulness and effectiveness of the system in realistic software development project and organization.

References

[1] V. R. Basili and H. D. Rombach, "The TAME project: Towards improvement-oriented software environment," *IEEE Trans. Software Eng.*, 14, 6, pp.758-773 (1988).

[2] S. D. Conte, H.E. Dunsmore and V.Y. Shen, *Software Engineering Metrics and Models*, The Benjamin/Cummings Pub. (1986).

[3] L. E. Druffel, S. T. Redwine, Jr. and W. E. Riddle, "The STARS program: Overview and rationale", *IEEE Computer*, 16, 11, pp.21-29 (1983).

[4] D. A. Lamb, *Software Engineering: Planning for Change*, Prentice Hall (1988).

[5] K. Matsumoto, *A Programmer Performance Model and its Measurement Environment*, Ph.D. dissertation, Faculty of Engineering Science, Osaka University (1990).

[6] R. H. Thayer (Ed.), *Tutorial : Software engineering project management*, IEEE Computer Society Press (1988).

[7] K. Torii, T Kikuno, K. Matsumoto and S. Kusumoto, "A measurement environment and some results at class experiments," *Proceedings of the 2nd International Workshop on Software Quality Improvement*, pp.88-91 (1990).

[8] G. M. Weinberg and E. L. Schulman, "Goals and performance in computer programming," *Human Factors*, 16, 1, pp.70-77 (1974).