Empirical Software Engineering Platform and Empirical Project Monitor

Masao Ohira[1,4], Reishi Yokomori[2,4], Makoto Sakai[3,4]
Katsuro Inoue[2,4], Ken-ichi Matsumoto[1,4], Koji Torii[1,4]
[1]*Nara Institute of Science and Technology, 8916-5, Takayama, Ikoma, Nara, 630-0101, Japan*
[2]*Graduate School of Information Science and Technology, Osaka University,*
1-3, Machikaneyama, Toyonaka, Osaka, 560-8531, Japan
[3]*SRA Key Technology Laboratory, Inc. 3-12, Yotsuya, Shinjuku, Tokyo, 164-0004, Japan*
[4]*Empirical Software Engineering Research Laboratory,*
1-4-2, Shinsenri-higashi-machi, Toyonaka, Osaka, 560-0082, Japan
ohira@empirical.jp {yokomori, inoue}@ist.osaka-u.ac.jp
sakai@sra.co.jp {matumoto, torii}@is.asit-nara.ac.jp

Abstract

So far, many software developments tend to rather depend on experience-based know-how. In the formal demonstration, we introduce our ongoing project for developing a tool named Empirical Project Monitor as a partial implementation of the empirical software engineering platform. This project aims at establishing the methodology for software development based on the scientific method in order to increase the reliability and productivity of software products. Empirical Project Monitor is one of the tools developed under the project. Empirical Project Monitor collects useful data for controlling software development projects and provides visualizations of the data.

1. Introduction

As software systems prevail in social infrastructures, they perform important roles in a wide range of aspects of our life. A large amount and various kinds of data collected through software development activities are required to discuss scientifically on the reliability and productivity of software products. The study related to this issue is popularly practiced as Empirical Software Engineering (ESE) [1,2,8,11].

The results of ESE study can be partly applicable to software development but experience-based know-how is still used. The main focus in software engineering has been on supporting individual developers, managers, and single software development projects by providing computational tools for analyzing and assessing from the software engineering data. Little study has proposed methods to help improvement for software process and products in terms of multiple projects or organizations while software development activities are becoming larger and more complex.

Nowadays, we can gather and analyze massive data on software development in a large scale using rapidly growing hardware capabilities. We have launched the 5-year plan project since April 2003 in Japan. The project in cooperation with software industry would go beyond the limits of university research in the way of collecting practical data in software development.

The goal of the project is to improve the methods and practices in the field of software development through measurement, analysis, evaluation, and feedback, in the same way as many other sciences and engineering. The project aims at establishing the methodology for software development based on the scientific method, instead of using experience-based know-how, to increase the reliability and productivity of software.

As the first step toward our goal, we are currently evaluating various software development methods, technologies, and support tools based on quantitative data. And we are constructing the empirical software engineering platform that will be the core for promoting this project. Using the platform, real data from the software industry would advance research and development in the university, which try to suggest

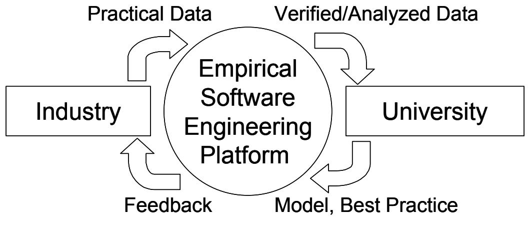

Figure 1. The model of this project

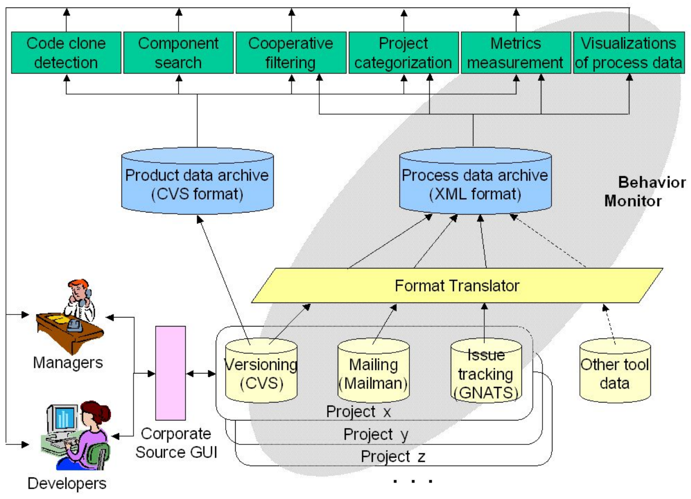

Figure 2. The system architecture of the EASE platform

models and best practices applicable to the industry (Figure 1).

In what follows, we illustrate the system architecture of the empirical software engineering platform [6]. The next section introduces Empirical Project Monitor, which is one of the tools developed under our project. Empirical Project Monitor collects useful data for controlling software development projects and provides visualizations of the data.

2. The system architecture of the software development support platform

We are not building a single huge system that supports all the steps of practical data collections, data analyses, and feedback. Instead of trying to build a huge system, we are developing a flexible system with various pluggable tools, which can be replaced according to the objectives and methods of analyses of the data.

The fundamental functions of the empirical software engineering platform are:

- to collect a large amount of data from tens of thousands of software projects and
- to make such the huge data available to analyze for improving projects and increasing organizational benefit.

Figure 2 shows the system architecture of the empirical software engineering platform [6]. At first, the platform collects project data through general GUIs such as Corporate Source and WinCVS [13]. The platform is currently able to possess version control & configuration management data of CVS [4], mail archives of Mailman [7], and bug reports archives of Gnats [5], which are widely used in software development and implemented as open source software.

Then these data are classified into the product data in the CVS format and the process data in the XML format through the format translator. Using these data stored in PostgresSQL databases, we have plans to provide a variety of analysis tools, which help managers and developers detect code clone [12], search software components [10], find similar projects by cooperative filtering [9], and so on. According to purposes of analysis, various data collection tools can be also plugged in the platform if their data are transformed into the XML format.

As we mentioned above, this platform mainly uses the three types of projects' data as a base for analyses because these data are semi-automatically collected through software development projects without additional work of managers and developers. We believe that this would be an important feature of our platform in order to collect/analyze enormous data from tens of thousands of projects, which would be used to provide useful feedback to developers and managers, and to lead benefit in an organizational scale.

3. Empirical Project Monitor: a tool for measuring and visualizing metrics data

We have developed Empirical Project Monitor (EPM) as a part of the empirical software engineering platform. EPM consists of basic components in the platform (the grayed area in Figure 2).

EPM analyzes the process data with the XML extension in the PostgresSQL database, which is transformed from the three types of data as follows:

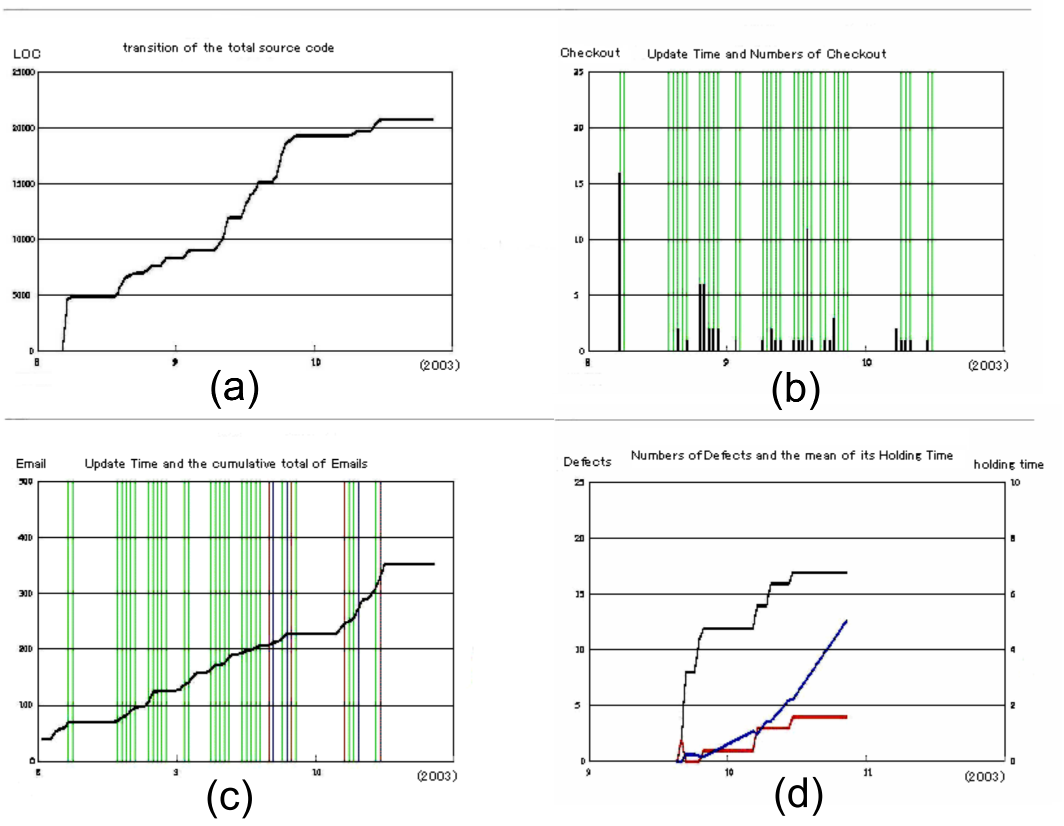

Figure 3. Visualization results of Empirical Projects Monitor

- the data of version control & configuration management systems
- the data of mailing list managers
- the data of bug tracking systems

These data are collected automatically through GUIs (e.g. Cooperate Source [3], WinCVS [13]), which are used for supporting software development communities in resent years. The current implementation of the format translators deals with the data of CVS (version control & configuration management system), the data of Mailman (mailing list manager), and the data of Gnats (bug tracking system). The data of other similar tools are also available by small adjustments if the parameters are converted into the XML format.

Analyzing the process data, EPM provides users with various visualizations of the data and helps them understand current states of their projects.

For instance, Figure 3-(a) shows the change of the cumulative total of the source code of our own project. The graph helps users understand the current progress of the project and estimate the future status by comparing to past similar projects.

Figure 3-(b) represents the update time (horizontal longer lines) and numbers of checkouts of CVS. Users can confirm whether other members refer CVS after the update.

Figure 3-(c) illustrates the change of the cumulative total of e-mails (line graph), the time of occurred/solved problems, and the update time of CVS. From the relationship between numbers of e-mail and the update time of CVS, users can confirm how much members of the project discussed on problems until problems were solved.

Figure 3-(d) shows the change of the cumulative total and numbers of unsolved problems and the change of the mean time until problems were solved. Users can confirm how long time it took to solve problems.

In this way, Empirical Project Monitor visualizes useful information for keeping projects under control.

The followings have been mentioned as problems of data collection for process improvement and EPM is suitable for such cases:

- incensement of developers/managers' burden of entering data for process improvement
- delay of information exchanges related to projects' progress
- arbitrary human-hand operations to data

First, since EPM deals with the process data generated from existing popular tools, which are used also for open source software development, most developers and managers do not need additional work to collect the data. Once users operate the tools such as CSV, Mailman, and Gnats, EPM registers the data automatically into the process data archives. That is, the process data is stored in real time.

Second, because of the above reason, there is no delay of the data (information) collection. This helps users understand current states of their projects and keep their projects under control.

Finally, documents (or products) based project managements may arise the problems such as arbitrary human-hand operations to the data (e.g. The quality of documents is depended on individuals.). But such the operations cannot be basically possible because EPM deals with the raw process data generated from CVS, Mailman, and Gnats.

We are evaluating Empirical Project Monitor by using a large amount of the data from open source software development projects.

4. Future work

Empirical study on software development is an active area in the field of Empirical Software Engineering (ESE) [1,2,8,11]. But the approaches of ESE have not been sufficiently applied to software development in organizations. We are trying to identify the factors of this from an organizational point of view.

Our goal of this project is to construct the methodology for supporting measurement based software development. By analyzing the huge data collected from tens of thousands of software development projects, we would like to provide benefit not only to individual developers and managers but also to organizations. Although we much need to understand “what kinds of technologies/feedback are effective to increase organizational benefit,” we believe that our approach could go beyond the limitations of product/process managements by human-hand and traditional software engineering.

Acknowledgement

We thank Satoru Iwamura, Taira Shinkai and Eiji Ono for supporting to develop the empirical software engineering platform and Empirical Project Monitor.

References

[1] A. Aurum, R. Jeffery, C. Wohlin, and M. Handzic, “Managing Software Engineering Knowledge”, Springer, Germany, 2003

[2] V. Basili, “The Experimental Software Engineering Group: A Perspective”, ICSE 2000 award presentation, Limerick, Ireland, June 5-10, 2000.

[3] Corporate Source, Novel Electronic Software Publishing Platform, http://www.zeesource.net/

[4] CVS, Concurrent Version System, http://www.cvshome.org/

[5] Gnats, GNU Bug Tracking System, http://www.gnu.org/software/gnats/

[6] K. Inoue, P.K. Garg, H. Iida, K. Matsumoto, and K. Torii, “Mega Software Engineering”, submitted for the 26th International Conference of Software Engineering (ICSE 2004), Scotland, UK, 2004.

[7] Mailman, the GNU Mailing List Manager, http://www.list.org/

[8] M. Lindvall, V. Basili, B. Boehm, P. Costa, K. Dangle, F. Shull, R. Tesoriero, L. Williams, and M. Zelkowitz, “Empirical Findings in Agile Methods”, Proc. of XP/Agile Universe 2002, pp.197-207, 2002.

[9] N. Ohsugi, A. Monden, and S. Morisaki, “Collaborative Filtering Approach for Software Function Discovery”, Proc. of Int. Symp. Empirical SE (ISESE), vol. 2, pp. 45-46, Nara, Japan, 2002.

[10] R. Yokomori, T. Ishio, T. Yamamoto, M. Matsushita, S. Kusumoto, and K. Inoue, “Java Program Analysis Projects in Osaka University: Aspect-Based Slicing System ADAS and Ranked-Component Search System SPARS-J”, Proc. of the 25th International Conference on Software Engineering (ICSE2003), pp.828-829, Portland, Oregon, 2003.

[11] L. Briand, C. Differding, and D. Rombach, “Practical guidelines for measurement-based process improvement”, Technical Report ISERN-96-05, Department of Computer Science, University of Kaiserslautern, Germany, 1996.

[12] T. Kamiya, S. Kusumoto, and K.Inoue, “CCFinder: A Multi-Linguistic Token-based Code Clone Detection System for Large Scale Source Code”, IEEE Trans. Software Engineering, vol. 28, No. 7, pp. 654-670, 2002.

[13] WinCVS, Windows GUI front-end for CVS, http://wincvs.org/