PAPER

Industry Application of Software Development Task Measurement System : TaskPit

Pawin SUTHIPORNOPAS[†], **Pattara LEELAPRUTE**[†], **Akito MONDEN, Hidetake UWANO, Yasutaka KAMEI, Kenji ARAKI, Kingo YAMADA,** ***and*** **Ken-ichi MATSUMOTO**[††],

SUMMARY This paper describes introduction of software development process measurement into industry level of software development using TaskPit with little requirements and less cost in implementing with the industry level.

1. Abstract

To find any problems in a software development process, we have been developing an automated measurement tool called TaskPit, which monitors software development tasks such as programming, testing and documentation based on the execution history of software applications. This paper introduces the system requirements, design and implementation of TaskPit; then, presents two industry case studies applying TaskPit to actual software development. In the first case study, we applied TaskPit to 12 software developers in a certain software development division. As a result, several concerns (to be improved) have been revealed such as (a) a project leader spend too much time on development tasks while he was supposed to be a manager rather than a developer, (b) several developers rarely used e-mails while a company instructed developers to use e-mails as much as possible to leave communication records during development, and (c) several developers wrote too long e-mails to their customers. In the second case study, we have recorded the planned, actual, and declared time of development tasks. As a result, we found that (d) there were unplanned tasks in more than half of days, and (e) the declared time became closer day by day to the actual time measured by TaskPit. These suggest that TaskPit is useful not only for a project manager who is responsible to the process improvement but also for a developer who wants to improve by him/herself.

2. Introduction

To promote an idea of ”Process Improvement via Measurement” in software industry, we had developed a software development task measurement system called TaskPit [4] in 2008, and have been updating it till today. TaskPit automatically records the time and the amount of work of daily tasks of an individual developer or a development team to find any problems in software development. It can record the time spent for each task such as ”Programming” using Eclipse or Visual Studio, ”Documentation” using Word or other text editors, ”E-mail” using Gmail on a browser and so on, where a task is associated with a set of applications and window titles. TaskPit records an amount of work for each task in terms of the number of keystrokes and the mouse clicks. It can also records an amount of deliverables of each task as the increased size of files in a directory associated to the task.

To date, the TaskPit community has been gradually grown up, user manuals and related tools have been developed by volunteers, and as of July 2015, total downloads of TaskPit version 1.0.0 to 1.0.3 has reached 1500 [4]. Now we are demanding for industry case studies to share the experience and findings about how TaskPit could be used for process improvement.

This paper introduces the system requirements, design and implementation of TaskPit; then presents industry case studies applying TaskPit to two software organizations. The first organization consists of 12 members, measured in 9 days (6 business days) where 7 are developers, 3 leaders, and 2 customer services. Analysis of data and measurements of the work surveyed in departments are collected and summarize by one veteran employee from other department to look after the flow of their work. In the second organization there is only one developer measured in 17 days (13 business days) measured by TaskPit. In addition to the automatic measurement TaskPit did, there is also a value plan and engage time for each development tasks for this developer each day. This is to clarify the difference between the measured value that was collected and analyze manually, and automatic measurement.

Please note that this paper is an extension of our Japanese workshop paper (short paper) [FOSE2013] with an additional (second) case study. In this paper we also added explanations of the system requirements, design and implementation of TaskPit to clarify the design concept of TaskPit and to illustrate how TaskPit can be used in software organizations.

In the following Chapter 2, we will describe the related work and research in background, Chapter 3 will be about TaskPit and its system requirements, design, and implementation. In Chapter 4 and 5 will be measurement results and analysis in two organizations. Lastly in Chapter 6 is a summary.

[†]Department of Computer Engineering, Kasetsart University
[††]Nara Institute of Science and Technology
DOI: 10.1587/transfun.E01.A.1

3. Background and Related Work

As Tom DeMarco said You can't control what you can't measurewe believe it is essential to control the process of developing software [5]. For this purpose, various product processes metrics and measurement have been applied to the developing site [6]. In many of the environment of development site; the software scale, work hours, and bugs are measured and used for project management, and quality assurance of the development [14] [10].

On the other hand, problems found in the development field are mostly from human factors [13] rather than the product process and measurements. This is considered to be natural as described in Personal Software Process (PSP) and Team Software Process (TSP), where developers and the development team follow up the work of day-to-day tasks and record everything manually, these methods are well-known as a method for process improvement [7] [8]. However, as developers have to do this manually, this method popularity grows thin and does not spread widely as it should.

At first, PSP has been proposed as a measurement procedure in accordance with a specific measurement form which required accurate calculation and required great cost to the data measurement. There are also tools and systems that act as a supporting role in data measurement to help calculating data more easily such as Process Dashboard [1], Task Coach [3] and Slim Timer [2], but using these tools will required to switch the measurement work (Context Switching) because they are all in different environment, and this has become a barrier in introducing the measurement procedure to the development site [11].

Without causing the Context Switch, automatic measurement tools on software development have been proposed, for example EPM [9], Ginger [], Ginger2 [12], HackyStat [], and PROM []. Detailed data collected by these tools can be used with less cost.

In this paper, the main objective is to support the introduction of process improvement in the development team. It is easy to introduce TaskPit as a system that does not require pre-planning, and capable of carrying out process improvement through measurement. TaskPit will record the number of hours and amount of work separately task by task, and can automatically measure the changes in the result number compared to actual number. Data in the system is to measure the amount of work, and achievement in each task are directly represents and are easily analyzed. It is suitable for the stage of process improvement during development.

4. TaskPit

4.1 System Requirements

Based on the previous chapter of this paper and the requirements of task measuring system, our system requirements are described in detail in the following subsection:

4.1.1 [Requirement 1: Binding between Tasks and Applications]

In TaskPit system, each development task are to be work on different applications and windows. Therefore, we need to bind the task name with application name corresponding to it (the executable file name or process name). However, in the actual work, it is not always necessary to use only one application to perform one task. One application might work on many tasks, and one task might be completed by many applications. Moreover, even in different tasks, it may use the same application. In such a case, to distinguish tasks by differences in the character string in the name of the window during application execution, we approach it with BNF as follows:

⟨Task⟩ ::= ⟨Application⟩ {||⟨ Application ⟩}
⟨Application⟩ ::= ⟨Executable File Name⟩ [⟨Window Name⟩]

Tasks are defined as a set of one or more running applications, and applications are defined as a set of executable name and window name. Below are examples of how tasks are defined.

- Mail = OUTLOOK.EXE || IEXPLORE.EXE "Gmail" || CHROME.EXE "Gmail"
- Documentation = WINWORD.EXE || NOTEPAD.EXE || TERAPAD.EXE
- Browsing = IEXPLORE.EXE || CHROME.EXE
- Searching = IEXPLORE.EXE "Google" || CHROME.EXE "Google"
- Spreadsheet = EXCEL.EXE
- Programming = eclipse.exe || devenv.exe
- Testing = mstsc.exe || Beyond32.exe || DF.exe

4.1.2 [Requirement 2: Automated Data Measurement]

• Task Execution Time Measurement

Time will start recording for each of the application and window when "Application is in use" which refers to a state where that window is currently in focus. However, even if it is in focus, if the time has passed (10 minutes) without active computer input (mouse and keyboard), it will be recorded as not using any application.

• Amount of Work Measurement

Amount of work will be recorded by the input of the computer, refers to keyboard strokes and mouse clicks.

• Files Measurement

For each task file, it will be defined as a directory and file extensions. TaskPit scans the files under specified directory and measure the increasing and decreasing number of total file size and number of files in intervals. The definition is described below in BNF.

⟨Artifact⟩ ::= ⟨Directory⟩ ⟨Extension⟩ {⟨ Extension ⟩}

And below are some examples.

Requirement analysis documents =
”C:\Users\monden\desktop\development\SRS”,
”doc,tex,txt”

Design documents =
”C:\Documents and Settings\monden\desktop\
development\sources”, ”c,cpp,java”

• Regarding Privacy

During the measuring process of the team, it is necessary to protect personal information and keeping the privacy of developers in a comfort zone. In the proposed system, the key strokes, applications, window names, file names, and many more are not recorded to prevent the feeling of being ”monitored” by the user, because it could cause negative feedback to the system we introduced. Any other window or application that user open up will instead recorded as ”Other Tasks”.

4.1.3 [Requirement 3: Team measurement]

Data measured for each user is collected at one time with timestamp placed with each task in the team unit, and then count up amount of efforts and products produced.

4.1.4 [Requirement 4: Data visualization]

To visualize the measurement results in the development team, for a specific period of time we can display the graph as a bar chart or line chart to visulize total time spent doing tasks and total time spent during work by using timestamp from the result log file of our proposed system. The data visualization will require a log file from proposed system, compiled it into a pre-processed version to sum up all work hour and visualize in graphics. ***Will add charts that can visualize keystrokes and mouse clicks in the future*** Chart of total time spent on tasks and total work hour will look like Figure 1 and Figure 2 respectively.

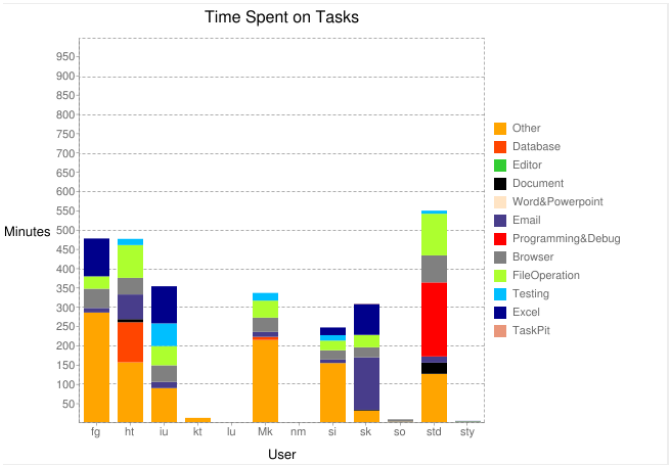

Fig. 1 Total time spent on tasks

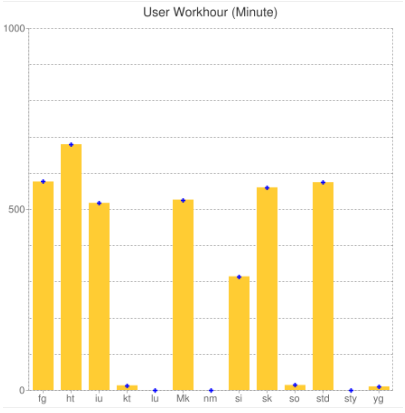

Fig. 2 Total work hour

References

[1] Process dashboard. http://processdash.sourceforge.net/pspdash.html.
[2] Slimtimer - time tracking without the timesheet. http://www.slimtimer.com/.
[3] Task coach - your friendly task manager. http://members.chello.nl/f.niessink/.
[4] Taskpit. http://taskpit.jpn.org.
[5] Tom DeMarco. *Controlling software projects: Management, measurement, and estimates*. Prentice Hall PTR, 1986.
[6] Robert B Grady. *Practical software metrics for project management and process improvement*. Prentice-Hall, Inc., 1992.
[7] Watts S Humphrey. *A discipline for software engineering*. Addison-Wesley Longman Publishing Co., Inc., 1995.
[8] Watts S Humphrey. *Łȼå*. åå, 2001.
[9] Masao Ohira, Reishi Yokomori, Makoto Sakai, Ken-ichi Matsumoto, Katsuro Inoue, and Koji Torii. Empirical project monitor: A tool for mining multiple project data. In *International Workshop on Mining Software Repositories (MSR2004)*, pages 42–46. IET, 2004.
[10] Lawrence Putnam and Ware Myers. *Five core metrics: the intelligence behind successful software management*. Addison-Wesley, 2013.
[11] Alberto Sillitti, Andrea Janes, Giancarlo Succi, and Tullio Vernazza. Collecting, integrating and analyzing software metrics and personal software process data. In *Euromicro Conference, 2003. Proceedings. 29th*, pages 336–342. IEEE, 2003.
[12] Koji Torii, Ken-ichi Matsumoto, Kumiyo Nakakoji, Yoshihiro Takada, Shingo Takada, and Kazuyuki Shima. Ginger2: An environment for computer-aided empirical software engineering. *Software Engineering, IEEE Transactions on*, 25(4):474–492, 1999.
[13] å£. It å . *SEC journal*, 6(1):32–35, 2010.
[14] and ł. *¢åł: NEC åł¢åłŁ*. Ł£å, 2010.