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 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 themself.

1. 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.

2. Background and Related Work

As Tom DeMarco said 「You can’t control what you can’t measure」 we 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 [13] [10].

On the other hand, problems found in the development field are mostly from human factors [14] 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

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

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.

3. TaskPit

3.1 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:

3.1.1 [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

3.1.2 [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, application name, windows name, file name, 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”.

3.1.3 [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.

3.1.4 [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.

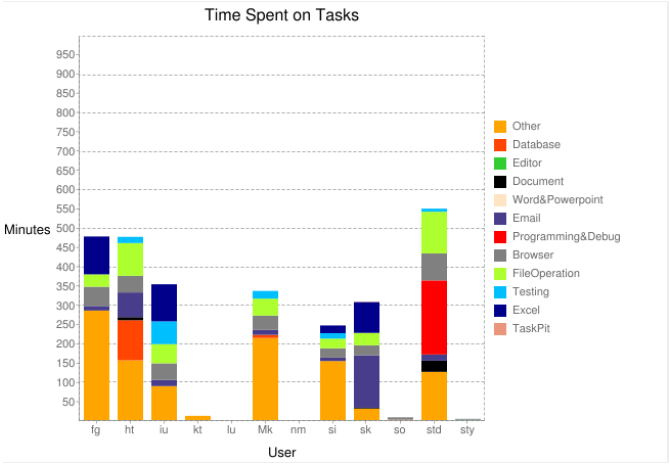

Fig. 1 Total time spent on tasks

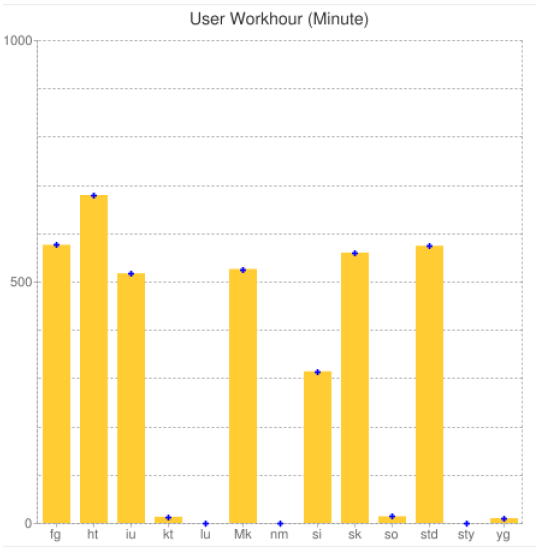

Fig. 2 Total work hour

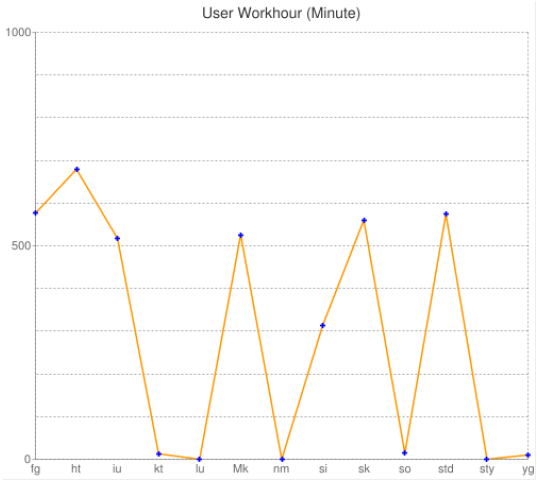

Fig. 3 Total work hour in Line Chart

3.2 System Design

As shown in Figure 4, TaskPit have its own database working at the back end of the system along with the visualization unit, configuration, and log files with daily report. In the configuration file will contain definitions of task and file as described before in the requirement section to be able to generate output of time interval of usage of each task and program in the log file. In Figure 5 is the user interface of the program. Each task will record its own execution time by accumulating all time it has been in focus along with number of keystrokes and mouse clicks.

To meet up the requirement of being less burden to the overall system and without providing server too much load, there will be a specified shared directory to store measurement result. Visualization tool is also used with configuration files and log files to graph visualize chart from various aspects of measurement results for specific period of time. Working in the client to server fashion as shown in Figure 6.

In addition, an overview report of daily work can be generated in both CSV format and in Excel .xls format.

3.3 Implementation

TaskPit 1.0.X operates using .Net Framework on Windows XP, Vista, and 7. Database requires SQLite and tasks are performed using Windows API. To handle the current active windows we invoke GetForegroundWindow, window title is obtained using GetWindowText. To get process we use

GetProcessById method with System.Diagnostics.Process class to obtain execution file name.

Similar to tasks, the keystrokes and mouse clicks is measured by registering a method to hook the keyboard and mouse events in SetWindowsHookEx. In addition we also measure files with GetFiles method from System.IO.Directory class at regular time intervals to scan the files specified in that directory to measure number of files and file size.

Once everything has been recorded and done in one interval of choice, whether days or weeks, it will generate CSV file as an output. This CSV file is the log of everyone measurement which will be used to compile into one work hour sheet and another pre-processed data log file sheet required to visualize a graph of total time spent in tasks and total work hour of every developers by using Google Chart API and a framework called charts4j to run Google Chart API using Java language.

4. Case Study 1: Team Measurement

4.1 Goal of Measurement

In the first case study in one software development organization, we believed that to discover the clues of problems in the development department will lead to improvements, so to carry out the analysis of measurement results, we assign one veteran employee as a manager and support department which function as a Project Management Office (PMO). This person will act to make sure of quality control and standards of enterprise-wide project from the point of view of human, and in terms of assistance of consulting, this analyst has to grasp the role of each team members, which he or she did not know until the beginning of the work.

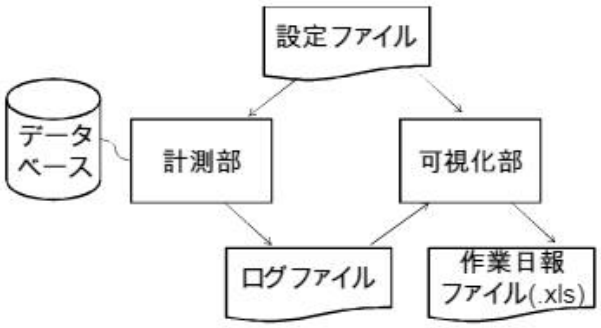

Fig. 4 Overall System

4.2 Measurement Target and Time

The target of the measurement is the department of software development consists of 12 people in total. As shown in

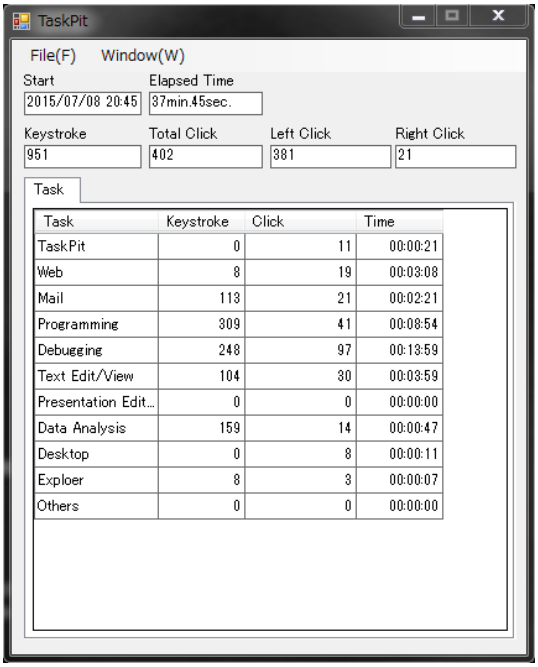

Fig. 5 TaskPit User Interface

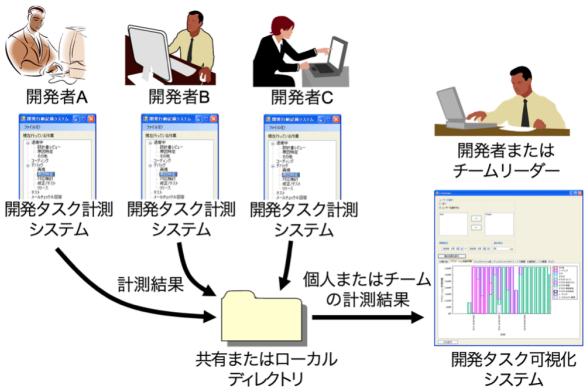

Fig. 6 Shared System

Figure.7, each person's role in the department are different. With 3 main leaders, 3 developers in Project A, 4 developers in Project B, and lastly with 2 customer supports. Project A is really busy since the deadline of the project is very near, while Project B is more relaxed since the deadline is further away. Member B1 of the Project B is not familiar with the environment since that person has recently been moved from other department to this department in one month. In addition, 2 customer supports (CS1, CS2) has shorter working hour than what was planned.

In this department we used Trac as a project management tool for the browser, which has been encouraged to carry out material management and communications. In data measurement we used TaskPit 1.0.1 with modified components to hide GUI from the user

to prevent disturbance during work. We also did some interviews for applications used in this department to link up the program with TaskPit. Measurements were carried out over for total of 9 days (6 business days).

表 1　ケーススタディ 1 における計測対象の開発者

役割	メンバー	備考
リーダ	L1, L2, L3	
プロジェクト A の開発作業 (開発 A)	A1, A2, A3	A1 は 4,5 年目．A2，A3 は 1 年目
プロジェクト B の開発作業 (開発 B)	B1, B2, B3, B4	B1 は他部署から異動してきてから 1 ヶ月
顧客窓口	CS1, CS2	二人とも時短勤務

Fig. 7　Team Roles

4.3　File Configuration

As shown in Figure 8, file configuration is done by assigning multiple applications to a task.

ブラウザ = iexplore.exe|firefox.exe|Safari.exe|chrome.exe
メール = thunderbird.exe|Outlook.exe|iexplore.exe:gmail
ファイル操作 = explorer.exe
エディタ　= sakura.exe|noediter2.exe
ＤＢ操作　= SqlWb.exe
エクセル = EXCEL.EXE|soffice.exe
ワード・パワポ = WINWORD.EXE|POWERPNT.EXE
文書閲覧　= AcroRd32.exe
テスト　= mstsc.exe|Beyond32.exe|DF.exe|reverse.exe|reverseserver.exe|perfmon.exe|....
プログラミング・デバッグ = eclipse.exe|devenv.exe|VPC.exe|VMWindows.exe|hh.exe

Fig. 8　File Configuration

4.4　Measurement Results and Analysis

From Figure 9 is time spent in each tasks, shows a working time of respective mean value per day. Working hours are collected from the start to the end of using TaskPit. The obtained results are as follows.

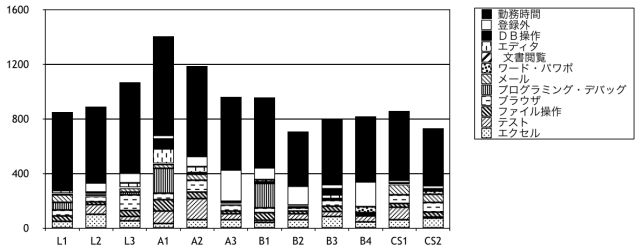

Fig. 9　Time Spent

- As shown in Figure 9 there are tasks that were unregistered in every developers. With A3, B2, B4 in particular who exceeded 100 minutes in this category. One of the cause of this for this department is because new program and unidentified application can be installed into this pre-determined application only PC. There is a case where unspecified application is detected. Also the attendance management system for the developers was not registered in the configuration file. Furthermore, TaskPit can not measure testing or debugging that happens in Virtual OS. (This was fixed in TaskPit 1.0.3 so that Virtual OS can be measured normally)
- Overall working hours of members in Project A (A1, A2, A3) is longer. Project A is also closer to the deadline with members working hours really close to the time they were given. On the other hand B1, B2, B3 and B4 are progressing steadily with shorter work hours compared to Project A. By the work of TaskPit, it can be concluded that this program can grasp a hold of how busy a project is.
- Person with the best performance is A1, as seen from the PC work time and actual work hour are really close together. Spent 184 minutes debugging, 106 minutes editing and another 94 minutes testing. A1 is definitely a programmer who actually work on the task given. Meanwhile, A2 who belongs to the same project spent 0 minute debugging (with unregistered 76 minutes to be added), along with 156 minutes testing. TaskPit is able to understand the work of each developer by visualizing it.
- The relationship between working hours and work time on PC is shown in Figure 10. From this figure, leaders spent little time in PC compare to working hours than those of developers and customer support. This is due to the leaders having to spent time working in business and management part rather than developing and coding. However, leader L2 has slightly larger PC time compare to other leaders. Breaking down the tasks, leader L2 had engaged on average 71 minutes in testing along with 103 minutes in excel program operation. As a leader this could be said that too much time is spent on development work.
- Two customer supports (CS1, CS2) as seen on the chart are mostly engaged into mailing and browsing. This is due to their work requires communicating and interacting with customers via e-mail. It is suggests that we apply material management using Trac on the browser. (In this measurement, the problem of not being able to reflect differences in tasks on the browser configuration is also revealed here.)
- Looking at mail tasks, L2, A3, B1, B2, B3, B4 are all spent less than 15 minutes here. In this department we have encouraged members to use e-mail more than telephone as a communication method (in order to keep the record for later). From this fact, it can be said that L2 has spent too little time using e-mail, and furthermore L2 has been reported with more extensive use of telephone.
- Figure 11 shows time spent in mailing and programming along with their respective typing speed (or key

strokes). Comparing key strokes per minute for the e-mail tasks, CS1 and CS2 seems higher. It could be possible that the content in e-mails of these members are longer or could be the cause of typing correction. The mail exchanging with customer requires simplicity and quality in terms of information along with descriptive content. It is possible that we could use an improvement in mail writing.

- Looking from Figure 11. It is possible that A1 is the most productive, excluding quality, among other member of this team, as told by his typing speed is rather high along with his over-the-top time spent in programming and debug.

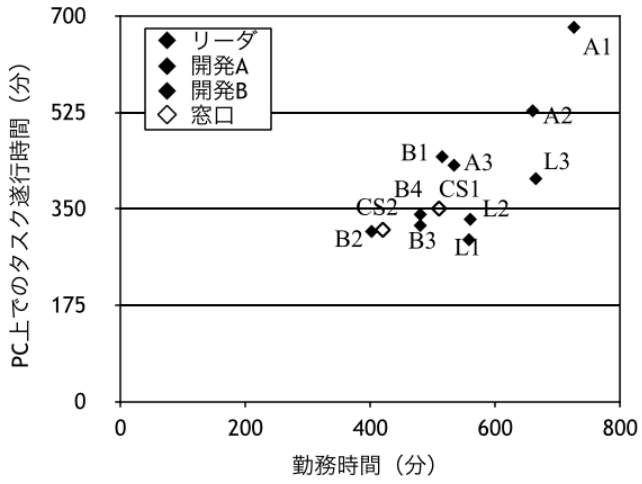

Fig. 10 PC Work Time and Working Hour Relationship

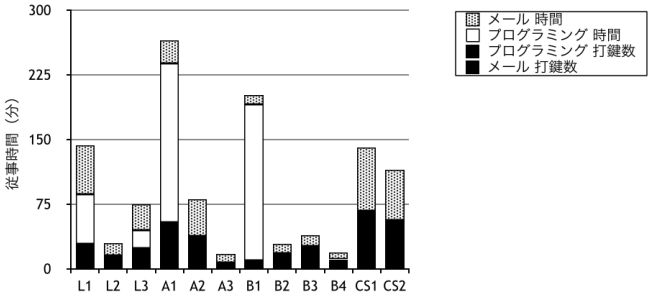

Fig. 11 Mail and Programming

5. Case Study 2: Personal Measurement

5.1 Measurement Purpose and Methods

In this second case study, the developer performed his own measurement and analysis assuming that it would lead to self improvements. The developer will set his own planned value for each development tasks for everyday and also measure value everyday to see the difference. Planned value is the value developers set at the beginning of the day to be a goal. The measured value is value obtained by TaskPit. The developer will also report his own measured value at the end of the day before comparing with TaskPit.

If the value of planned and measured are close together, even without using TaskPit, that developer is able to perform tasks along what was planned and has responsibility in his given work. However, if large difference between measured and reported value are presented, the developer may not be able to understand the implementation time of the task. Also when there are large difference between what was planned and what was measured, this is a serious problem as you can not manage your time well.

5.2 Measurement Target and Period

The target of this measurement belongs to a different organization from the first case study. The web-based programming developer who was tasked for total 17 days (13 business days). Prior to this measurement, the developer studied and learnt the configuration file of TaskPit in 3 days. Also at the end of the daily work, planned values and measured values are asked to be presented and to give comments upon them.

5.3 Measurement Results and Analysis

Measured value and planned value are shown in Figure 12 and Figure 13 respectively. The horizontal axis of each graph is the number of days and the vertical represents number of hours. Measured values are likely to be smaller than what is planned, except for day 11. From the comment presented by the developer, this is because he did not use the personal computer on day 4, while first three days is because of unexpected amount of work. In fact, the unplanned work was planned to be included from the start by the amount of 5 to 10 percent of total work, revealed by TaskPit. This should be noted however, that the planned value up to the day 11 was intended to be like what figures have shown. In Figure 14 shows the subtraction of measured value from the planned value. As shown in the figure, as time passed working on a task using TaskPit in the process, the time estimated has become closer to the actual measured value. Therefore in this case, by continuously usage of TaskPit, one can assume that developers can understand their workload correctly. ”Working is now easier and now planning time and actual time are much closer together.”. Therefore, by using TaskPit, developers are now aware of actual time and planned time and can develop themselves to exceed the expectation of what is planned. It is expected that this could be of use in managing and planning in development works.

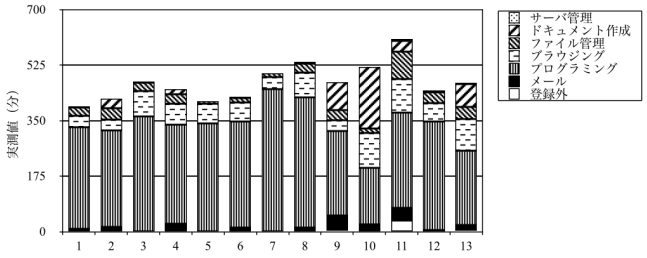

Fig. 12 Measured Value

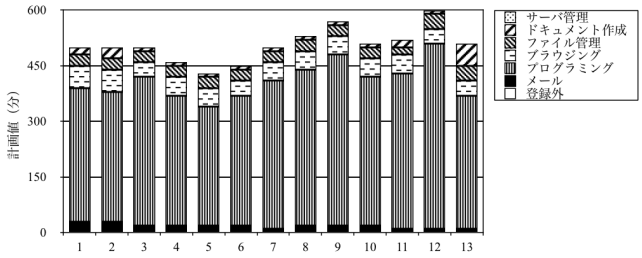

Fig. 13 Planned Value

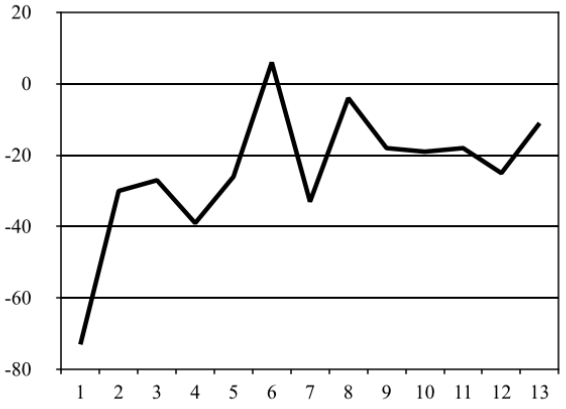

Fig. 14 Result Value

6. Summary

In this paper, we clarified the effect of the automatic measurement in software development work for the purpose of publishing its findings and to apply it with two development organization. The first one was to measure 12 developers over six business days. It was found that it is possible that TaskPit can grasp and explain how busy the development process and each project of each developers are. In addition, it is estimated that (a) leader spent too much time developing, and not many people are communicating through e-mail. (b) In contact with customers through the e-mail, the content inside these mails are too long and was addressed in short period of time. Lastly (c) there is a possibility that there could be a room for some improvements in e-mail writing. In the second organization, we observe and measure the developer who is tasked on developing web-software for 13 business days and recorded all measured and planned values. As a result (d) unplanned work was found more than half of the total work in a day, but (e) in the end the total work hours obtained from the measured data exceeds the expectation of what is planned, by using TaskPit to help managing and understanding developer's own work. With more case studies and findings in the future, we expected this project to help many organizations in development process. The authors will continue working on application cases of TaskPit and expected to continue publish newly obtained findings.

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. **パーソナルソフトウェアプロセス入門**. 共立出版, 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] 直美 and 誉田. **ソフトウェア品質会計***: NEC* **の高品質ソフトウェア開発を支える品質保証技術**. 日科技連出版社, 2010.
[14] 樋口登. It プロジェクトを「見える化」するチェックシート. *SEC journal*, 6(1):32–35, 2010.