WORKING TIME USAGE AND TRACKING IN A SMALL SOFTWARE DEVELOPMENT ORGANIZATION

Lasse Harjumaa, Tytti Pokka, Heidi Moisanen and Jukka Sirviö
Department of Information Processing Science, University of Oulu, P.O. Box 3000, Oulun Yliopisto, Finland

Keywords: Software engineering, Effort estimation, Project management, case study.

Abstract: This paper represents a study of working time usage in a small software development organization. The purpose of the study was twofold. First, we wanted to understand how software developers in the organization work and second, we wanted to explore the attitudes they had toward different types of time tracking approaches. The aim was to provide practical suggestions of appropriate methods and tools for monitoring the developers' time. According to the results, working with computer tools occupies the overwhelming majority of the working time although manual tasks and interruptions take some of the time. Even though the developers in the case company do not feel threatened by time monitoring, they do not either feel that monitoring is necessary, which is interesting and challenging from the project management viewpoint. We suggest that the case company should establish a lightweight, tool-based time tracking process and trains the developers to use the system and report their working time accurately.

1 INTRODUCTION

Software development is an activity mainly based on human effort. For this reason, costs highly depend on the time software developers spend doing each software development activity. Tracking the time spent in each activity is useful for efficient accounting. Reliable estimations on costs and schedules of software development are the key element of efficient project management. (Boehm et al., 1995; Sillitti et al., 2003)

The first phase of a software development process is planning. A project plan should be based on estimates of the size and duration of the activities that are needed to produce the software product. (Ching-Seh & Simmons, 2000). Thus, historical time tracking data have significant value when making schedule estimations to the customers, for example.

It is important to understand, which software development activities occupy the developers' time most in order to make estimations of project length and costs. Even more important is that identifying the most time-consuming activities helps the organization to improve the development process by providing support for the laborious tasks.

This paper describes a case study within a small Finnish software organization that aims at increasing their development efficiency by improving their time tracking and effort estimation practices. Before introducing procedures for the development time management, analysis and supporting tools, it is necessary to investigate the current situation in the organization and identify the activities that require developers' working time. Based on the findings, we will give the organization some suggestions for the effort estimation process and tool implementation.

The rest of the paper is structured as follows. Section two gives an overview to research related to working habits of software developers. Section three describes possibilities for tracking developers' time. Section four describes the research carried out in this study and section five lists the main results of the case study. Finally, section six concludes the work.

2 WORKING HABITS OF SOFTWARE DEVELOPERS

Perry et al. (1995) have investigated the feasibility of different techniques for collecting working time data and carried out experiments to find out where the developers' time goes during the software development. They conclude that the total effort of system development is affected by the technology used, but also by the social environment and the

Harjumaa L., Pokka T., Moisanen H. and Sirviö J. (2008).
WORKING TIME USAGE AND TRACKING IN A SMALL SOFTWARE DEVELOPMENT ORGANIZATION.
In *Proceedings of the Third International Conference on Software and Data Technologies - ISDM/ABF*, pages 168-173
DOI: 10.5220/0001886701680173

development process. Thus, the organizational structure and culture, as well as communication between developers all have an effect on the time utilization. Perry et al. (1995) claim that some of the non-technical elements can even have greater significance than the technological issues in terms of explaining the time usage. Especially inter-personal contacts that developers make during a working day take time.

LaToza et al. (2006) present have studied typical tools and activities of software developers and found out that recovering and managing implicit knowledge related to the development require great effort. They also state that software development is a highly social process that involves a great deal of task switching between tools, several communication channels and interruptions. According to LaToza et al. (2006), Interruptions are the major factor that reduces working effectiveness.

Singer et al. (1997) present empirical data of software engineers' daily work practices from a large software engineering organization. The research focuses on tool support for software development and the authors conclude that it is necessary that tools are consistent with the work practices. They also observed that designing and writing code are not the only tasks that developers do. A great deal of the working time is spent on consulting, reading documentation and learning, for example.

There are also some ethnographic studies available relating to the issue. For example, Newman (1998) reports a study of a large middleware development project, in which each stakeholder has his own concerns. Consecutively, the design process in reality is not a simple linear flow of tasks, starting from requirements and resulting in a complete software product. Instead, software design involves lots of activities that are needed to negotiate commitments, communicate requirements and make decisions. Another example of the social aspects of software development is reported by Suchman et al. (1999). Results of their studies suggest that technologies are tightly connected to the environment in which they are used and the success of software work depends on how well these are integrated.

As a summary, it can be stated that software developers' working time is occupied by a) doing the actual development work; b) activities that are needed to follow certain process, c) activities that are related to the organizational and social environment and d) interruptions. This is a rough classification and each of these categories could be decomposed into much more detailed subcategories. However, it is important to understand that the actual design and coding work is just a part of a developer's time usage.

3 TRACKING THE WORKING TIME

Time tracking is essential for meaningful software development effort estimation, and tool support or automation of time tracking activities can improve the accuracy and quantity of the working time data (Johnson et al. 2000). Well-situated and adequate tools can provide a lightweight approach for starting and establishing software measurement and process improvement. Tools for automatic data collection and use of persistent measures database are often listed among the essential success factors of measurement framework implementation. Examples of the articles and studies highlighting the importance of correct tools include (Fenton&Neil, 2000; Offen&Jeffery, 1997).

A number of comprehensive time tracking tool implementations exist. Hackystat (Johnson, 2002) is a fully automated tool that records developers' actions with different types of sensors. Especially the Personal Software Process (PSP) approach (Humphrey, 1997) has given inspiration to many time tracking tools, as in PSP measurement and analysis of historical data is in key role in making estimates of effort and product quality. PROM (Sillitti et al., 2003) and Jasmine (Shin et al., 2007) are other examples of time tracking tools.

Academic research, empirical studies and available tool implementations concerning time tracking issues suggest that there are at least the following four different levels of time tracking approaches or systems. 1) Manual tracking, 2) System-based tracking, 3) Recording-based tracking and 4) Fully automatic tracking. The pros and cons of each of these approaches are discussed in the following.

Manual tracking is the simplest form of time management. The software developers collect their working hours manually either into a plain file or a personal datasheet and deliver the timesheets to the project manager by email, for example. This approach is inexpensive to implement and easy to learn as individual developers can use tools he or she is already familiar with. However, consistency and integrity of data from different developers is a problem. Reporting scale, structure and terminology

can vary and combining data into a meaningful summary can become impossible. Templates, datasheets and guidelines can formalize the process to some extent, but this approach is still probably suitable only for very small organizations.

In system-based tracking, the developers insert their working time into a time management system and the project manager can compile reports directly from the system. This partly automated process provides almost real-time data, as the project manager sees the data in specified format as soon as the developer has inserted data into the system. Furthermore, the structure and terminology of the data is consistent and reports can be generated automatically. On the other hand, integration into other development software and developers' motivation to use the system may cause problems.

Recording-based tracking can also be called start-pause-stop (SPS) tracking, as the recording is typically done by pressing stopwatch-style buttons. Thus, a developer uses the SPS user interface to clock time spent in different activities, and the system records and stores recorded data. Usually these types of systems offer manual data manipulation and insertion option, as well. Stopwatch-style recording ensures that data is accurate and available for project managers real-time. The user interfaces of SPS tools are moderately easy to use and there are also a number of freeware implementations available. The main disadvantages of this approach are related to the clocking. Developers do not necessarily remember to start and stop the recording and monitoring work that is done away from the computer is difficult. In addition, distraction problems may arise as interruptions may affect on the reliability of the recorded data.

Finally, fully automated time tracking tools are based on the idea of minimum (or even zero-level) involvement from the user. The recording is started automatically when certain system is used. This eliminates human-related problems of data accuracy, user motivation and data format. However, even in this approach distraction problems remain, as the effect of interruptions to standard workflow cannot be automatically detected. Furthermore, compatibility with the other development tools and methods in use has to be ensured.

In practice, time tracking is often neglected. The biggest obstacle for that is that measuring where developers' time goes is not considered crucial. As software development projects are typically very strictly scheduled and budgeted, there is no time to spend in activities that do not produce immediate payback. (Sillitti et al., 2003.)

4 RESEARCH SETTING

The research was done in a division of a large international software company. The primary products of the company are embedded systems used in telecommunication. However, the quantity of software within the products is continuously and rapidly growing. The software development unit that this study was carried out in is located in Finland and has about 20 software engineers.

The main objective of the study was to find out what tasks occupy software developers' working time and to investigate feasibility of different time tracking methods and tools to follow the time usage. New tools are needed, as the case organization wants to forecast the durations and costs of their future projects as accurately as possible. The products of the company are getting more software intensive and more attention has to be paid to the efficiency of software development activities than before.

The current time management system does not support task-specific reporting of working time. Recording and reporting can be done only on project-basis. More detailed information is necessary in order to monitor the development effort and identifying possible problems in the development process. For example, distinguishing design, implementation and testing efforts from each other is essential for evaluating the quality of both product and process. The case organization wishes to establish an estimation database with the new time tracking system.

The time tracking is currently done manually. Project planning is done with Microsoft Project, but task lists created with the project planning tool are not utilized in time tracking. Thus, categorization of the activities and reporting formats of the working time data vary. Before suggestions for selecting and implementing time tracking tools could be made, it was considered necessary to look into the activities that developers perform in the organization. For that, the developers were asked what tasks and activities they use their working time to.

The observations from academic research described in chapter two are utilized in the empirical study of the case company. The factors affecting time usage that have been identified in previous research are included in the questionnaire and analyzed in the case company context.

Data gathering was conducted with a questionnaire consisting of 25 multiple-choice questions and one open question. The questionnaire questions were partially based on the questionnaire used by LaToza et al. (2006). The questionnaire

included sections for issues concerning demographic information, iterative development process in use in the organization, usage of the working time, interruptions to work, and feelings toward time tracking systems. The main part of the questionnaire is presented in Table 1. For clarity, demographic questions (1-8) are omitted in the table.

For questions concerning working time, categorizations that can be found in LaToza et al. (2006) were used. Descriptions for each task in the categorization were also presented to the respondents. In addition, the meaning of time tracking was explained in further concerning questions 19-27.

Table 1: Questions presented to the developers.

Iterative development process questions:
9. The system development process is iterative. Is it easy to you distinguishing different tasks from each others? Easy | 1 | 2 | 3 | 4 | 5 | Difficult
10. Do you think it is easy clearly distinguish your work to different tasks? Easy | 1 | 2 | 3 | 4 | 5 | Difficult
Working time questions:
11. What proportion of your working time you usually spend in each system development task? (The sum must be 100 %) Designing __ % Writing __ % Understanding __ % Editing __ % Unit testing __ % Communicating __ % Overhead __ % Non code __ %
12. What proportion of communicating time you usually spend in different communication methods? (The sum must be 100 %) Face-to-face _____ % Meetings _____ % Email _____ % Phone _____ % Other _____ %, what?
13. What proportion of understanding time you usually spend in different communication methods? (The sum must be 100 %) Source code editor _____ % Whiteboard _____ % Paper ___%Visual designers __ % Other __ %, what?
14. What proportion of your working time you usually work with and without the computer? (The sum must be 100 %) With computer _____ %Without computer _____ %
Interruptions questions:
15. What is the mean number of interruptions of a) your typical work day? ____ times b) one typical work hour? ____ times
16. What is the mean duration of interruptions in minutes? ____ minutes
17. What are the proportions of reasons of interruptions? Visits _____ % Meetings _____ % Email _____ % Phone _____ % Other _____ %, what?

Table 1: Questions presented to the developers (cont.).

18. What is the mean recovering time after interruptions? ____ minutes
Time tracking questions:
19. Are you tracking your working at the moment? ____ No ____ Yes (Answer to questions 20 & 21, if you answered yes to question 19.)
20. How you track your working time now? • Manually (paper, excel, etc.) • Using system, what system
21. Do you think that your present system of time tracking is effective enough? Very effective | 1 | 2 | 3 | 4 | 5 | Not effective
22. How would you feel about monitoring your working time? Choose one number from scale 1-5. Important | 1 | 2 | 3 | 4 | 5 | Not necessary Effective | 1 | 2 | 3 | 4 | 5 | Time demanding Easy | 1 | 2 | 3 | 4 | 5 | Difficult Pleasant | 1 | 2 | 3 | 4 | 5 | Annoying Feels like my work is interesting |1|2|3|4|5| Feels like "spying"
23. What do you prefer to be the best way to track system developers (SD) working time? Rank to different ways by numbers 1-4 (1 = best, …, 4 = worst). • Manually (paper, excel, etc.) • Clicking " START, PAUSE, STOP" system • System, which automatically records the working time • Other way, what?
23. What do you prefer to be the best way to track system developers (SD) working time? Rank to different ways by numbers 1-4 (1 = best, …, 4 = worst). <Same options as in previous questions>
24. What's the probability (0-100 % to each point) of you tracking the work time? <Same options as in previous questions>
25. What's the probability (0-100 % to each point) that you mark/track the time accurately? <Same options as in previous questions>
26. How valid you consider each method measure system development process time (0-100 % to each point)? <Same options as in previous questions>
27. Why do you consider the time tracking method you selected to be the best?

The questionnaire was delivered to the contact person, who in turn delivered it to the individual developers. This method of collecting data increases some risks concerning interpretation of the questions and negligence of the answers, as the respondents are not personally contacted by the researcher. We

have tried to mitigate these risks by designing the questionnaire to be easy to fill in and unambiguous.

5 RESULTS

Based on the results of the questionnaire, we have made some suggestions for the case company in order to enable efficient and practical time tracking approach.

Nine people answered the questionnaire. The respondents seem to be quite experienced. On average, they had almost eight years working experience. However, the deviation is broad: the most experienced developer has been in the field for 18 years and the least experienced for only one year. Five developers are working on one project and four developers have several projects going on at the same time. When working on several projects concurrently, the number of projects is 2-3. The durations of the projects are widely spread between one to nine months. Questions 9 and 10 considered distinguishing different tasks from each other. Even though the developers do not think it is difficult to switch projects and distinguish different tasks (the mean of the answers was 2 on scale 1-easy to 5-difficult), it is important to notice that the development work is not "like in books". Usually it is assumed that a developer focuses on one project at a time. This sets additional requirements for time tracking tools. Table 2 summarizes the portions of time that developers spend in system development tasks and in understanding the problem. Numbers in the table are percentages spent in specific tasks.

Table 2: Work structure of the respondents.

Task	**< 5 Mean**	**≥ 5 Mean**	**Mean diff.**
Development tasks			
Designing	27	26	1
Writing	23	17	6
Understanding	15	12	3
Editing	11	9	2
Unit testing	7	22	-15
Communicating	5.8	5.5	0.3
Overhead	6.3	4.5	1.8
Non code	6.3	6.4	-0.2
Understanding time			
Source code editor	68	48	20
Whiteboard	0	15	-15
Paper	22	29	7
Visual designer	10	8	2

The results are categorized according to the developer experience. There were four respondents that had less than five years experience and five respondents with five or more years of working experience. It seems that there are some differences in writing and unit testing tasks. The experienced developers spend less time in writing and more time in unit testing than the less experienced developers. Furthermore, there seems to be a difference between experienced and less experienced developers in the terms of source code editor and whiteboard usage. Younger developers perhaps are more used to work on computerized tools than older colleagues.

In order to understand the nature of developers' working habits, interruptions are an important factor. The developers estimate that they have approximately five interruptions during the day and one interruption typically lasts about 10 minutes. They inform that it takes usually almost 5 minutes extra just to recover from the interruption. This gives a rough estimation that 5*15 minutes (1.25 hours) is used daily for interruptions or recovering from them.

Questions 20 and 21 considered time tracking tools and methods in use. Four respondents did not track the development time at all. Four recorded hours manually and only one developer used a system for that purpose. Interestingly, the developer that used a system to track his or her time considered time tracking less effective than those who recorded their time usage manually.

Questions 23 and 24 related to the preferred ways of tracking time. Based on the answers, it seems that developers that currently use manual time tracking system prefer to stay in manual system and developers that do not use any system prefer automatic and SPS tracking. When comparing the probability estimations of using a time tracking system, there seems to be some difference between non-trackers and manual-trackers: 1) the non-trackers evaluate their probability to use certain tracking method in every case much lower than the manual-trackers and 2) the non-trackers give a highest percentage to the SPS tracking while manual trackers score the SPS second lowest in terms of probability.

Table 3 shows developers' feelings toward monitoring working time in general (question 22). The scale is shown on the top row of the table and answering options are shown on the left. Each column shows the number of respondents and the percentage in parenthesis. Even five respondents out of nine think that monitoring working time is not necessary at all and additional three respondents are "in-the-middle". Still, the majority think that it is effective and easy. Attitudes in the pleasant-annoying and interesting-spying lines are very neutral.

Table 3: Opinions on monitoring working time.

Feels about monitoring working time	**1**	**2**	**3**	**4**	**5**
1-Important 5-Not necessary	2(22)	1(11)	3(22)	4(44)	1(11)
1-Effective 5-Time demanding	0	5(56)	3(33)	1(11)	0
1-Easy 5-Difficult	2(22)	4(44)	3(33)	0	0
1-Pleasant 5-Annoying	0	3(33)	5(56)	1(11)	0
1-Feels … interesting 5-Feels like "spying"	3(33)	0	6(67)	0	0

According to the answers to questions 25 and 26, the probability to use a system is highest for SPS type of systems. The probability of using automatic system is almost as high, but there is wider variety in opinions. The most surprising observation is that manual tracking is evaluated to have the highest validity, even though all the three approaches are considered to be quite similar in terms of accuracy.

The last question was open. The thoughts of the best ways to track time are very different between the developers in the case company. Two things seem to be the most important when determining the best time tracking method: ease of use and accuracy. However, the answers show that the developers recognize these two criteria to be controversial: if the method used is easy, it is not automatically accurate and vice versa. Some of the respondents also emphasize the accuracy of the user and others mention that the accuracy of the system is crucial.

The sample size was small. However, we believe that the main trends that we have observed in this study are correct. The results are not contradictory to previous researches, even though there were some surprising issues in our case.

6 CONCLUSIONS

Resource estimation is one of the first key elements to implement when establishing a predictable software development process. Surprisingly, software developers do not automatically consider time tracking useful. Developers need training and motivation for monitoring their working time. Benefits of time tracking have to be clearly visible.

Attitudes toward time tracking in general were quite neutral. However, developers were not willing to adopt new repertoire of tools for that purpose. Those who already kept record on their working time manually wanted to stay with the manual system. Automated tools were considered useful by those who did not track their working hours at all.

In order to gain wider view on developers' working habits, a wider study with more respondents should be done, preferably from several companies.

REFERENCES

Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R. & Selby, R.. (1995) Cost models for future software lifecycles: COCOMO 2.0. Annals of Software Engineering, vol 1, 57-94.

Ching-Seh W. & Simmons, D. (2000). Software Project Planning Associate (SPPA): a knowledge-based approach for dynamic software project planning and tracking. Proceedings of the 24th International Computer Software and Applications Conference (COMPSAC), 305-310.

Fenton, N. & Neil, M. (2000) Software metrics: roadmap. In proceedings of the ICSE - Future of Software Engineering Track 2000, 357-370.

Humphrey, W. S. (1997) Introduction to the Personal Software Process. Addison-Wesley: Reading, MA.

Johnson, P.M., Moore, C., Dane, J.A. & Brewer, R. S. (2000) Empirically guided software effort guesstimation. IEEE Software. Vol 17, no 6, 51-56.

Johnson, P. (2002) Supporting development of highly dependable system through continuous, automated, in-process, and individualized system measurement validation. University of Hawaii, technical report.

LaToza, Thomas D., Venolia, G. & DeLine, R. (2006) Maintaining Mental Models: A Study of Developer Work Habits. Proceedings of the International Conference on Software Engineering, 492-501.

Newman, S. (1998) Here, there and nowhere at all: Distribution, negotiation, and virtuality in post-mortem ethnography and engineering. Knowledge and Society 11, 235-267.

Offen, R.J. & Jeffery, R. (1997) Establishing software measurement programs, IEEE Software. vol 14, no 2, 45-53.

Perry D. E., Staudenmayerand, N. A. & Votta, L.G. Jr. (1995) Understanding and Improving Time Usage in System Development.

Shin, H., Choi, H. & Baik, J. (2007) Jasmine: A PSP supporting tool. Proceedings of the International Conference on SP, 73-83.

Sillitti, A., Janes, A, Succi, G. & Vernazza, T. (2003) Collecting, integrating and analyzing software metrics and personal software process data. Proceedings of the 29th Euromicro Conference, 336 - 342.

Suchman, L., Blomberg, J., Orr J.E. & Trigg R. (1999) Reconstructing technologies as social practice. The American Behavioral Scientist. Nov/Dec 1999; vol 43, no 3, 392-408.