Voice sensor based PSP timelog collection

Ahmad Ibrahim ° Ho-Jin Choi
School of Engineering, Information and Communications University
{ibrahim, hjchoi}@icu.ac.kr

Abstract

The purpose of the research is to solve the problem of automating time & schedule management by the user in office or development environment. Maintaining timelog manually is difficult task for the users that are following the Personal Software Process (PSP). In this paper we have discussed the difficulties in automating this task and proposed a solution for this problem.

1. Introduction

In a typical organization, for the projects to become successful, the developers and the teams have to be efficient. For the developers to be efficient, they have to adopt the practices given by personal software process (PSP) and Team Software Process (TSP). The usual structure of the project teams consists of developers and other team worker. One of the requirements of the personal software process is managing the time. For the time management, the PSP & TSP directs the developers and the other team members to record their daily time activity in a standard time log. The purpose of recording the time is to let the developers control and spent their time more properly and systematically. As part of the standard time log, the developers have to keep a record of their programming errors and time spent on each activity. Even for the average team members, who are not working as a developer, also have to keep track of their time. To maintain track of the time, most users either record the time manually or electronically. For recording the time electronically, various applications exist, even a simple MS Excel sheet can be used for this purpose. However, to enter the time record manually in such application is itself a problem. First of all, recording time itself is an overhead which the users have to do after every task and secondly even if the user enters the time & activity manually in such application, then it may contain error since the humans tend to forget the details. So automating the time and user activity is important since it can greatly reduce the burden on the developer's side.

The general PSP user has to maintain two types of daily timelog (Planned time and Actual timelog). The planned timelog contains the list of activities planned at certain time while the actual timelog contains the activities and their actual time related information. Besides the daily timelog, the user has to complete the weekly activity as well.

Another important focus of our research is on solving the meeting scheduling conflicts between users in an organization. In an organization, people may have different opinions and priorities for a certain meeting at some time. Due to this, scheduling and holding a meeting at some certain time is difficult due to the time conflicts. Many methods have been devised to solve the meeting scheduling problem. The algorithm we plan to use is the customized Clarke Tax algorithm [1].

This paper is divided into four sections. Section 2 describes the related work, Section 3 describes the approach, Section 4 describes the results obtained followed by conclusion.

2. Related Work

The SEI provides the standard timelog template to be used for the PSP process (Table 1). Various tools have tried to automate the PSP time log such as Process Dashboard [2], Hackystat [3] and PSPA [4] provides the sensors for automated time collection but they have some limitations. Hyunil et al proposed a data collection sensor for the Eclipse IDE for supporting the PSP/TSP[5] . The purpose of this IDE sensor was to collect and process the data automatically. Their proposed sensor gathered many information such as defect log, time log etc. Although the information collected by that sensor is useful, yet that sensor can only record the time log during the time when the user is

doing programming in the Eclipse IDE. It cannot gather the time information when the user is doing some other task other than coding. Like what if the same developer is involved in requirement gathering, requirement elicitation or any other project activity? so they will not be able to record their time automatically using such sensor.

One approach is to develop several sensor applications for several different tasks which is not feasible in terms of managing all the applications simultaneously. Like the software development cycle consists of many activities such as requirement gathering, elicitation, Software architecture, Implementation, Testing, and Documentation etc. So in entire development life cycle, managing lots of individual tools is a difficult task. So the other approach is to develop a single solution that can be used in different occasions. In this paper, we have also proposed how the voice recognition feature can be used to collect the time related information from the user.

Table 1 SEI timelog template for PSP

Date	Start (time)	Stop (time)	Interruption Time	Delta Time	Phase/Activity	Comments	Completed(C)	Units (U)

3. Proposed Approach

Our research focus is on developing a system that can help the users that are following the PSP and TSP to automatically record the time log information as managing the timelog manually is a time consuming and error prune process. So to automate the data collection from the user, we can use the voice recognition to record the activity, a person is doing at certain period of time. The voice recognition gives a lot of flexibility to the user to speak his activity on microphone while working on his own task. Since all team members following the PSP use the PC, so recording the activity of each user will not be a problem and this approach can work for any team member belonging to on any software development life cycle phase.

Based upon the user speech, the system can recognize the user speech and store the activity along with the time stamp. There can be two approaches that can be followed by the system (Figure 1). In the first approach, the rules are embedded into the voice recognition engine while in the second approach, the speech recognition engine produces a sentence as an output which is then given as input to the expert system like Jess [6].

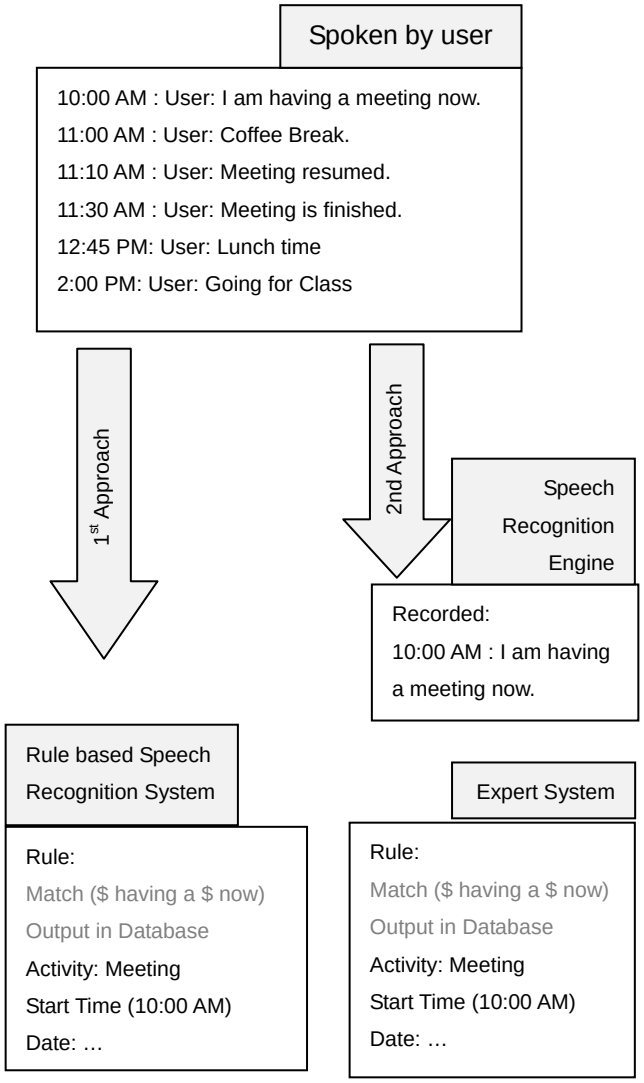


Figure 1 Example scenario followed by voice recognition module

Based upon the user conversation in Figure 1, the system would be able to generate the actual time log. Table 2 and Table 3 shows the actual timelog vs. the planned timelog.

Table 2 Actual time log

Date	Start	Stop	Interruption Time	Delta Time	Phase /Activity
01/01/2008	10:00 AM	11:30 AM	10	80	Meeting 1
01/01/2008	12:00 PM	01: 00 PM		60	Workshop
01/01/2008	01:15 PM	02: 25 PM		70	Class # 1
01/01/2008	02:40 PM				Class # 2
01/01/2008	04:30 PM				Dinner

Table 3 Planned time log

Date	Start	Stop	Interruption Time	Delta Time	Phase /Activity
01/01/2008	9:30 AM	11:00 AM		90	Meeting 1
01/01/2008	11:30 AM	12:30 PM		60	Workshop
01/01/2008	01:00 PM	2:30 PM	5	85	Class # 1
01/01/2008	02:30 PM	4:00 PM		90	Class # 2
01/01/2008	04:00 PM	5:00 PM		60	Workshop # 2

Although the use of the voice recognition solved the problem of the automatic data collection for time log but now the problem is to check the validity of the entries in the timelog and more than that how to solve the problems of conflicts in plan and the actual time log . The problems of the timelog validity can be divided into the problems of temporal data entry and constraints satisfaction.

For the problems related with temporal databases, suppose the user wants to know that what he did at 11:25 AM, then the system will respond back that the user was in Meeting 1 based upon the Actual timelog. But if the user ask the system what he did at 11:31 AM then the system would be confused what could be the answer? On the basis of planned timelog, the user was supposed to be in the workshop but the actual timelog shows no entry of the user activity. It could be that the user is still in Meeting 1. So to solve such problems, we can use the temporal databases. The further research on temporal databases will be our future work.

Constraints among the tasks can also create problems for the validity of the output. Each tasks can have predecessor, successors and a minimum duration. For example if there is a meeting that starts at 10:00 AM and after 5 min there is another entry Break Start, then there are two assumptions that can be made.

If the two activities are disjoint activities then the duration between them should be more than minimum duration of the initial task minimum duration. Like meeting and lunch are two disjoint activities and assume that the minimum duration of Meeting is 10 min, but if these two activities exist such that the time difference between them is 5 min, then there is a conflict (Figure 2).

If the two activities are not disjoint activities (i.e. their time can overlap each other) then the initial task should be the parent of the second task. For example meeting and break are disjoint activities, i.e. a break can come during meeting but a meeting cannot come during the break. So in this case, meeting should be the parent of break but this scenario is possible and it will not be regarded as a conflict (Figure 3).

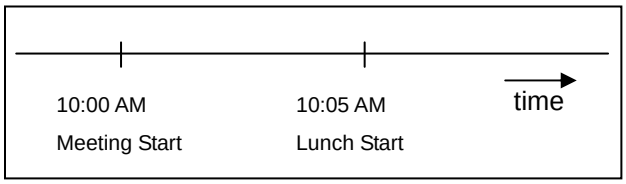


Figure 2 Conflict Scenario for disjoint activities

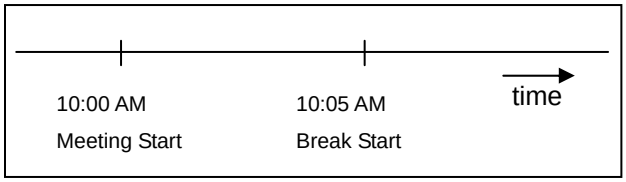


Figure 3 Conflict Scenario for non-disjoint activities

The overlapping among the tasks follows the rules of the Allen's Interval Algebra [7].

4. Results

In the first part of the proposed system, we have built speech recognition based system. The system matches the words for recognition. A list of all the tasks are stored in a database file. The user interface of the system is being built in C# while the Microsoft Speech API (SAPI) is used for voice recognition and MySQL for the database storage (Figure 4).

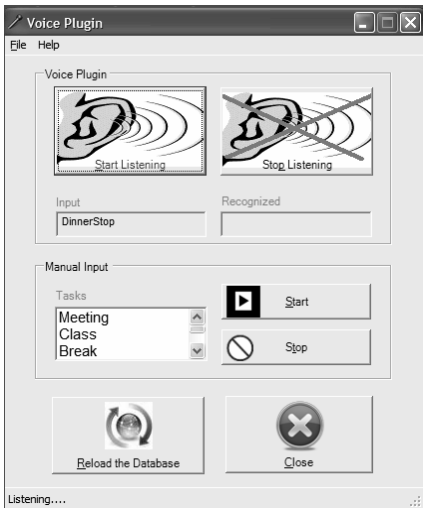
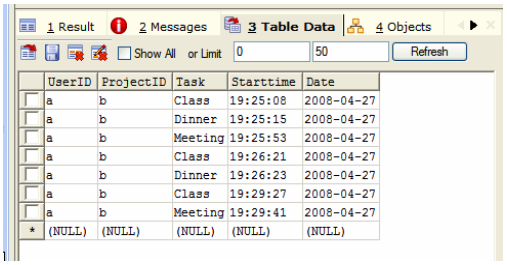


Figure 4 Voice based sensor for time data collection

There are three main advantages of our voice sensor approach over the manual data collection approach. There were two manual approaches. One being paper based data collection and

the other being using some data entry software to record data. The manual data collection approach used to be an overhead for the developers but in our system the developer doesn't have to write anything. Daily life speaking activity is much easy and informal as compared to writing some information which requires a user to stop doing his task and write his task and then continue his activity. The second advantage is that the manual data collection approach may contain errors since user may forget or skip the exact details about an activity. The voice sensor approach does not require the users to memorize his tasks and since every activity is recorded so it contains less error than the manual approach. Finally, it was difficult to extract the information in the manual data collection since the system was only used for data entry but our approach consists of a reasoning engine which, when ready, allows the system to intelligently recognize the activities and extract the relevant information.

The activity detail recorded by voice sensor is shown in the form of time log (Fig 5). Although this time log represent all the user activities but our final goal is to generate the PSP time log (Table 1). To convert the obtained timelog into the PSP time log, we are developing a framework which will used temporal reasoning to generate the desired PSP time log.


1 Result 2 Messages 3 Table Data 4 Objects

Show All or Limit 0 50 Refresh

UserID	ProjectID	Task	Starttime	Date
a	b	Class	19:25:08	2008-04-27
a	b	Dinner	19:25:15	2008-04-27
a	b	Meeting	19:25:53	2008-04-27
a	b	Class	19:26:21	2008-04-27
a	b	Dinner	19:26:23	2008-04-27
a	b	Class	19:29:27	2008-04-27
a	b	Meeting	19:29:41	2008-04-27
* (NULL)	(NULL)	(NULL)	(NULL)	(NULL)

Figure 5 Time Log generated by Voice sensor

5. Conclusion

In this paper, we have proposed a voice recognition based time data collection system for the personal software process (PSP). Our proposed system takes care of the validity and constraints among the tasks. We have already developed the first part of the system i.e. voice recognition based application. In the second part of the system, we will solve the temporal database aspect of this problem and also implement a meeting schedule conflict resolver system based upon the Clarke tax algorithm [1].

Acknowledgement

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Advancement) (IITA-2008-(C1090-0801-0032)).

References

[1] Ephrati, E., Zlotkin, G. and Rosenschein, J. , "Meet your destiny: a non-manipulable meeting scheduler", Proceedings of CSCW'94, Chapel Hill, NC, ACM, New York, October, pp. 359-71,1994.

[2]The Process Dashboard, http://processdash.sourceforge.net

[3] P. M. Johnson, H. B. Kou, J. M. Agustin, C. Chan, C. A. Moore, J. Miglani, S. Zhen, and W. E. Doane. Beyond the personal software process: Metrics collection and analysis for the differently disciplined. In Proceedings of the 2003 International Conference on Software Engineering, Portland, Oregon, May 2003

[4] Raymund Sison, David Diaz, Eliska Lam, Dennis Navarro, Jessica Navarro, "Personal Software Process (PSP) Assistant," apsec, pp. 687-696, 12th Asia-Pacific Software Engineering Conference (APSEC'05), 2005.

[5] Hyunil Shin, Hojin Choi, Jongmoon Baik, "Jasmine: A PSP Supporting Tool", IEEE ICSP 2007, Minneapolis, USA, Published in Springer LNCS 4470, ISBN: 978-3-540-34184-0, pp. 73 – 83,May 19-20, 2007.

[6] Ernest Friedman-Hill ,"Jess in Action", Manning Publications, ISBN: 1930110898, July 2003.

[7] James F. Allen, "Maintaining knowledge about temporal intervals", Communications of the ACM, Volume 26 , Issue 11, pp. 832 - 843, Nov 1983