Jasmine: A PSP Supporting Tool

Hyunil Shin, Ho-Jin Choi, and Jongmoon Baik

Information and Communications University, School of Engineering,
119 Munjiro, Yuseong-gu, Daejeon, 305-732, Korea
{linugee,hjchoi,jbaik}@icu.ac.kr

Abstract. The PSP (Personal Software Process) was developed to help developers make high-quality products through improving their personal software development processes. With consistent measurement and analysis activities that the PSP suggests, developers can identify process deficiencies and make a reliable estimate on effort and quality. However, due to the high-overhead and context-switching problem of manual data recording, developers have difficulties to collect reliable data, which can lead to wrong analysis results. Also, it is very inconvenient to use the paper-based process guide of the PSP in navigating its process information and difficult to attach additional process-related information to the process guide. In this paper, we describe a PSP supporting tool that we have developed to deal with these problems. The tool provides automated data collection and analysis to help acquire reliable data and identify process deficiencies. It also provides an EPG (Electronic Process Guide) in order to provide easy access and navigation of the PSP process information, which is integrated with an ER (Experience Repository) to allow developers to store development experiences.

Keywords: Personal Software Process, Electronic Process Guide, Automated Data Collection, Experience Repository.

1 Introduction

Continuous process improvement has been regarded as a solid solution to make high-quality products at the team and personal level as well as at the organization and project level. The PSP [1] was developed to help individual developers make high-quality products through improving their personal software development processes. The PSP provides a set of methods and practices to assist individual software developers to improve product and process quality such as defined and measurable process, size and effort estimation based on historical data, code and design review, precise designs, process quality measures, detailed plan, and earned value tracking. While the PSP has been proved as an effective way to improve the accuracy of effort estimation and to reduce defects in case studies [13, 14, 15], its manual data recording and paper-based process guide act as barriers in following the PSP process.

Among those methods and practices, the measurement and analysis is a central and core practice in identifying process deficiencies and providing a focus on process improvements. Sets of historical project data are used to make a reliable estimate on effort and quality. However, due to the high-overhead and context-switching problem

© Springer-Verlag Berlin Heidelberg 2007

of manual data recording, developers have difficulties to acquire reliable data, which can lead to wrong analysis results [2, 3]. The problem can be overcome through an automated tool for collecting the PSP data and analyzing the collected data. However, since an automated tool can not collect all necessary data, manual data recoding should be supported as well. Manual data recording can be still a problem, but data errors can be decreased because it is reduced to a few items. To help developers collect reliable data and all necessary data, it is therefore required to develop a tool for supporting both automated and manual data collection.

The PSP provides a set of increasingly evolved processes to help developers learn the methods and practices. To guide developers in following the processes, materials such as scripts, templates, and checklists are presented in a paper form, which can be seen as a paper process guide. A paper process guide generally has problems in its usability and maintenance because it is very inconvenient for developers to search and navigate process information and difficult to add process-related information or to modify existing information [8]. To solve these problems caused by a paper process guide, an EPG using the web technology is proposed allowing easy access to all process-related information [5, 8]. To allow easy navigation of the PSP process information and to enable storing additional information, it is necessary to develop an EPG which enhances the contents and usability of the paper-based PSP guide.

In this paper, we describe a PSP supporting tool, named Jasmine, which have been developed to address the issues above. Aiming at supporting personal process and quality management, the Jasmine provides capabilities to collect reliable data automatically and analyze the collected data. It also provides an EPG for the PSP guide for easy access, modification and addition of information.

The rest of this paper is organized as follows. The next section gives a short overview of sensor-based automated data collection, an EPG and an ER. This is followed by the description of the Jasmine's architecture and salient features. Section 4 presents a comparison with existing PSP supporting tools, and section 5 concludes the paper and describes future works.

2 Background

2.1 Sensor-Based Automated Data Collection

To reduce the high overhead and context-switching in manual data collection, tools like Hackystat [2, 9], PROM [12] have been developed. They collect automatically the PSP data and provide various analyses on the collected data. These tools do not require any efforts of developers in data collection, except in installation and configuration of sensors. Sensors, which are attached to development-related tools such as Eclipse, Microsoft Office, and JBuilder, are central components for automatic data collection. A sensor collects unobtrusively low-level data (e.g., information on files that developers are editing, results of unit test executions) by monitoring application-generated events of a development-related tool. Then, it sends the low-level data to a server where the data are stored and analyzed.

Although Hackystat and PROM collect the PSP data automatically, all necessary data can not be collected automatically and the collected data do not have all necessary information. For example, the time data collected automatically are associated with modification activity of software artifacts such as source files and design documents. In this way time spent on implementation or design activity can be automatically collected, but time spent on other activities (e.g., meeting, design review) can not be collected because not all important developer activities involve modification of software artifacts. Also, it is hard to identify which phase automatically collected time data are spent on. Defect data are automatically collected by sensors attached to unit testing mechanism such as JUnit or to bug reporting systems such as Bugzilla. However, there is no way to automatically collect defects in design/design review/code review phases where developers manually find defects, and automatically collected defect data do not have all information such as the time spent on finding and fixing the defect, the phase when it was injected, and its defect type.

2.2 EPG and ER

A process guide is a reference document to help process participants understand and execute a given process, providing guidance of the process and other useful information [8]. Basic information of process guides are details regarding activities, artifacts, roles, and relationships between them. Process guides are necessary for software process improvements where process knowledge transfer is crucial. Process guides traditionally were offered in a paper form, but it is said that they are not useful in their contents and layouts [8]. It is hard to navigate and search easily process information and to put related information together (e.g., an activity and its input and output artifacts) in paper-based process guides, because its layout is linear and static. Also, it is difficult to modify existing information or add new process information because it requires publishing a new edition of its process handbook.

These problems of paper-based process guides can be mitigated by an EPG which provides a process guide using the web technology [5, 8]. However, simply providing a process guide in forms such as PDF, Microsoft Word, or other electronic formats or converting the contents of a process guide into HTML is not treated as an EPG. In [8], a set of basic requirements are proposed which an EPG should meet.

- An EPG should provide all information contained in a good paper process guide.
- It is recommended that each web page contain so small manageable unit that process participants can easily understand and digest.
- An EPG should provide hyper-links, a graphical overview, and hierarchical activity decompositions for flexible navigation and easy access. Also, related information such as an activity and its associated artifacts should be linked together using hyper-links.
- All web pages should have the same basic structure in order to facilitate the usage.

Beyond the basic requirements above, an EPG can contain additional process-related information such as examples of a document, personal annotation, or discussion, which leads to more general knowledge and experience management. As a result it is recommended to integrate an EPG with an ER [7]. An ER is a system which is used to collect, structure, and reuse key management and development experience, and to make it quickly and easily accessible to users [6]. An ER plays a crucial role in knowledge and experience management where past knowledge and experience is seen as resources to solve today's problems.

Some works have been done to integrate an EPG with an ER. In [4, 7], a successful implementation of coupling an EPG with an ER in a small organization is presented. In the combined tool, an experience entity is attached to its related process element for easy access to a large number of collected experience data. The idea to structure experience data to related process elements is also supported by [6], which proposes that a good experience repository should be organized to its related process.

3 High-Level Architecture and Main Features of Jasmine

In this section, we describe the high-level architecture and main features of the Jasmine, which consists of two sub-systems, PPMT (Personal Process Management Tool) and PSPG/ER (PSP Guide/Experience Repository) as shown in Fig. 1. PPMT supports project planning, earned value tracking, and quality management by facilitating data collection and analyses. It automates large parts of data collection to reduce the high overhead and context switching. It also provides various data analyses in forms of charts, graphs, or tables. In PSPG/ER, the EPG provides the PSP process guide in the web and the ER is used to store and share development experience which can be linked to the EPG contents.

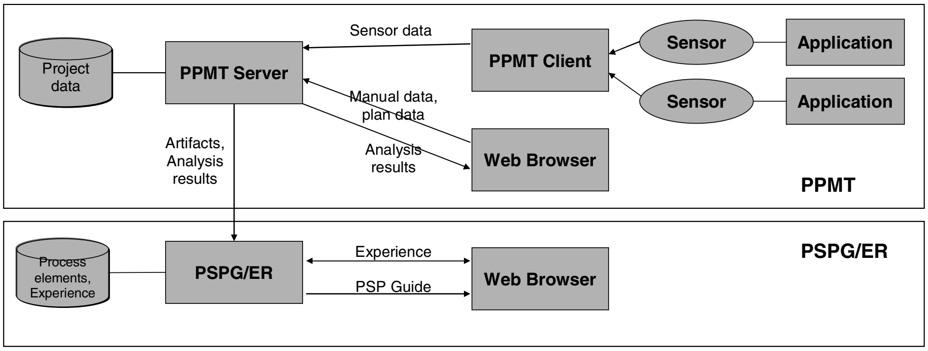


Fig. 1. Jasmine Architecture

3.1 PPMT

PPMT is designed using a client-server architecture, as illustrated in Fig. 1, in which the client consists of sensors developed for automated data collection. The server provides all functionalities except automated data collection. It was implemented as a web application which interacts with users through a web browser. The main components of PPMT are as follows.

- Sensor: It is attached to a development-related application. It collects automatically data by monitoring the application and then sends the data to the PPMT Client.

- PPMT Client: The main functionality of the PPMT Client is to receive sensor data from the sensors and to send them to the PPMT Server. It plays a temporary storage for collected sensor data when it is not connected to the server, and sends them to the server when the connection to the server is re-established. If necessary, it can preprocess sensor data before sending them to the PPMT Server.
- PPMT Server: It provides most of functionalities for PPMT: manual data recording, data storage, data analyses, earned value calculation, and users/projects administration. The implementation is based on Java technologies (such as Java Servlet, JSP, Java Beans, and JDBC), and on Apache Tomcat to execute Java Servlets and JSP.
- Database: It stores the collected PSP data from sensors and manual recording such as time and defect logs, task and schedule plan data, and information on users/projects. MySQL is used for the database implementation.

XML is used to send and receive sensor data among sensors, the PPMT Client, and the server. Its language-independent characteristic simplifies sensor data transmission because sensors are implemented using various programming languages. The main features of PPMT are presented below.

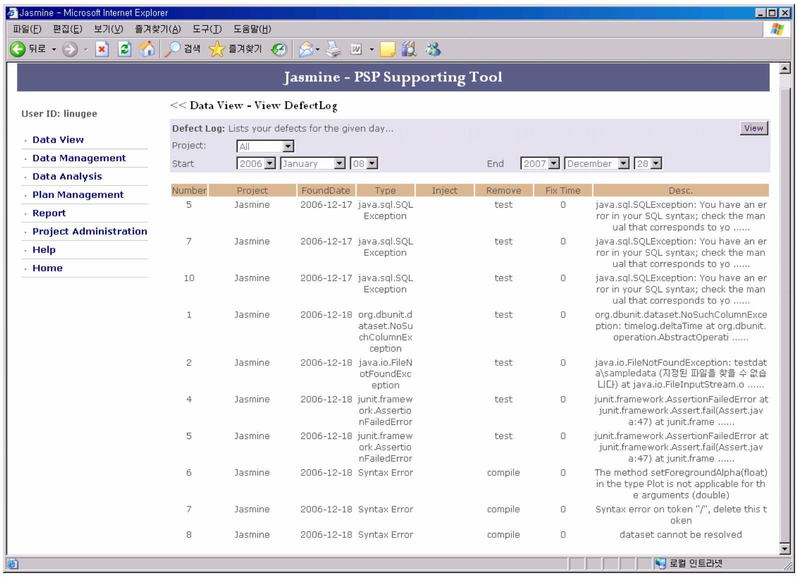
Sensor-based automated data collection. To facilitate recording time, defects, and software size, PPMT provides a sensor-based automated data collection mechanism like Hackystat, PROM. Time and defect data collected automatically are recorded in the time and defect log respectively, which allows modification and insertion of the data when necessary.

By monitoring software artifacts or tools, time spent on design, coding, review, and testing can be collected automatically. The current version of the Jasmine collects automatically time spent on: source code modification by monitoring continuously source files' size; manual testing of windows applications and web applications by monitoring mouse or key events occurred in the target application. An Eclipse sensor tracks Java source code modification and manual testing of a windows application executed in Eclipse. Testing web applications using Internet Explorer is tracked by an IE sensor. A set of consecutive time data is stored as an item in the time log.

Failed unit tests, bugs, compile errors and so on can be automatically collected as defects. The Jasmine collects automatically failed unit tests, compile errors, and runtime errors, each of which is stored as a defect in the defect log as shown in Fig. 2.

Table 1. Defect information of failed unit tests, compile errors, and runtime errors

	Failed unit tests	**Compile errors**	**Runtime errors**
Remove phase	"Test"	"Compile"	"Test"
Description	The stack trace of the exception	The description of the syntax error	The stack track of the exception
Defect type	The exception type	"Syntax"	The exception type
Found date	(automatic)	(automatic)	(automatic)
Inject phase	(manual)	(manual)	(manual)
Fix time	(manual)	(manual)	(manual)


Jasmine - Microsoft Internet Explorer
파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T) 도움말(H)
뒤로
검색
즐겨찾기
로컬 인트라넷

Jasmine - PSP Supporting Tool

User ID: linugee

- Data View
- Data Management
- Data Analysis
- Plan Management
- Report
- Project Administration
- Help
- Home

<< Data View - View DefectLog

Defect Log: Lists your defects for the given day... View

Project: All

Start 2006 January 08 End 2007 December 28

Number	Project	FoundDate	Type	Inject	Remove	Fix Time	Desc.
5	Jasmine	2006-12-17	java.sql.SQL Exception		test	0	java.sql.SQLException: You have an er ror in your SQL syntax; check the man ual that corresponds to yo ......
7	Jasmine	2006-12-17	java.sql.SQL Exception		test	0	java.sql.SQLException: You have an er ror in your SQL syntax; check the man ual that corresponds to yo ......
10	Jasmine	2006-12-17	java.sql.SQL Exception		test	0	java.sql.SQLException: You have an er ror in your SQL syntax; check the man ual that corresponds to yo ......
1	Jasmine	2006-12-18	org.dbunit.d ataset.NoSu chColumnEx ception		test	0	org.dbunit.dataset.NoSuchColumnExce ption: timelog.deltaTime at org.dbunit. operation.AbstractOperati ......
2	Jasmine	2006-12-18	java.io.FileN otFoundExc eption		test	0	java.io.FileNotFoundException: testdat a\sampledata (지정된 파일을 찾을 수 없습 니다) at java.io.FileInputStream.o ......
4	Jasmine	2006-12-18	junit.framew ork.Assertio nFailedError		test	0	junit.framework.AssertionFailedError at junit.framework.Assert.fail(Assert.jav a:47) at junit.frame ......
5	Jasmine	2006-12-18	junit.framew ork.Assertio nFailedError		test	0	junit.framework.AssertionFailedError at junit.framework.Assert.fail(Assert.jav a:47) at junit.frame ......
6	Jasmine	2006-12-18	Syntax Error		compile	0	The method setForegroundAlpha(float) in the type Plot is not applicable for th e arguments (double)
7	Jasmine	2006-12-18	Syntax Error		compile	0	Syntax error on token "/", delete this t oken
8	Jasmine	2006-12-18	Syntax Error		compile	0	dataset cannot be resolved

Fig. 2. Defect log

The Eclipse sensor collects the results of unit tests executed by JUnit, Java compile errors, and Java exceptions. As described in Table 1, information on removal phase, description, defect type, and found date are automatically recorded.

Software size can be automatically collected as lines of code (LOC) measured by a line counting tool. The current implementation collects LOC measured by LOCC [16].

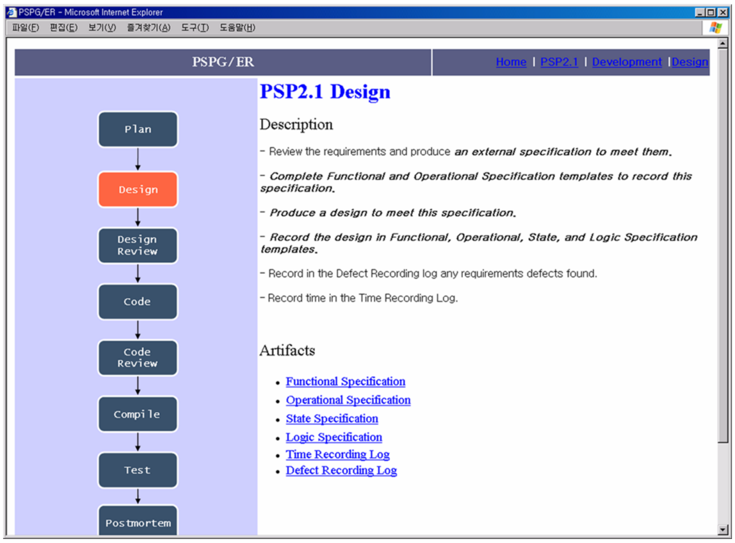
Support for planning and earned value tracking. Developers should make a detailed plan in the planning phase and track the progress with the earned value. In order to assist the project planning and tracking, PPMT provides forms to prepare the standard task and schedule planning templates, and automatically calculates the earned value of all planned tasks using planned data that a developer enters and actual data calculated from the recorded time log.

Data analyses and report generation. PPMT provides various analyses over the collected data in forms of charts or tables. It reports a summary of analyses results. Available analyses include trend charts which show the trend of data over time and an earned value chart which displays the planed value, the earned value, and the predicted earned value over time. It also provides Pareto charts for defect analysis and quality measures such as process yield, A/FR (Appraisal to Failure Ratio), and phase ratio. Also, it can generate a weekly report which summarizes project data during a given week and a project report which summarizes project data during the whole period.

3.2 PSPG/ER

The main elements provided by PSPG are the PSP activities (e.g., planning, design, and design review), artifacts (e.g., task and schedule plan, project plan summary), and the PSP processes (e.g., PSP0, PSP0.1). The PSPG/ER homepage provides a single point access to the PSP processes. A number of activity and artifact pages provide the guides of the PSP activities and artifacts, respectively. Every activity page consists of three frames as shown in Fig. 3: a navigation bar, a diagrammatic process flow, and a description section. The navigation bar consistently maintained in all of pages displays the current position. The diagrammatic process flow shows a flow of activities highlighting the selected activity and supports fast navigation to other activities. The description section contains the description of the selected activity, links to its related artifact pages, and links to experience data associated to it. Each artifact page consists of three frames as shown in Fig. 4: a navigation bar, a list of artifacts, and a description section. The list in the left frame contains a list of all the artifacts which must be produced in the selected PSP process. The description section includes the description of the selected artifact, its templates, and links to experience data related to it.

The ER enables developers to collect development experiences gained from previous projects by following the PSP process and to share them among team members. To provide easy access to a number of collected experiences, they are structured according to relevant process elements. That is, developers should insert an experience data to its related activity or artifact page. For example, a document example should be linked to its related artifact page. Experience data are categorized into example (only available in artifact pages), generic experience, and discussion. In the example category, examples of an artifact are provided in forms of files such as PDF, Microsoft Word, or other file formats which can be downloable, or HTML pages which are generated in PPMT. The generic experience category can include any helpful information such as lessons learned, code fragments, and links to useful web pages. The discussion category allows developers to discuss process elements with other developers.


PSPG/ER - Microsoft Internet Explorer
파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T) 도움말(H)
PSPG/ER
Home | PSP2.1 | Development | Design
Plan
Design
Design Review
Code
Code Review
Compile
Test
Postmortem

PSP2.1 Design

Description

- Review the requirements and produce *an external specification to meet them.*

- *Complete Functional and Operational Specification templates to record this specification.*

- *Produce a design to meet this specification.*

- *Record the design in Functional, Operational, State, and Logic Specification templates.*

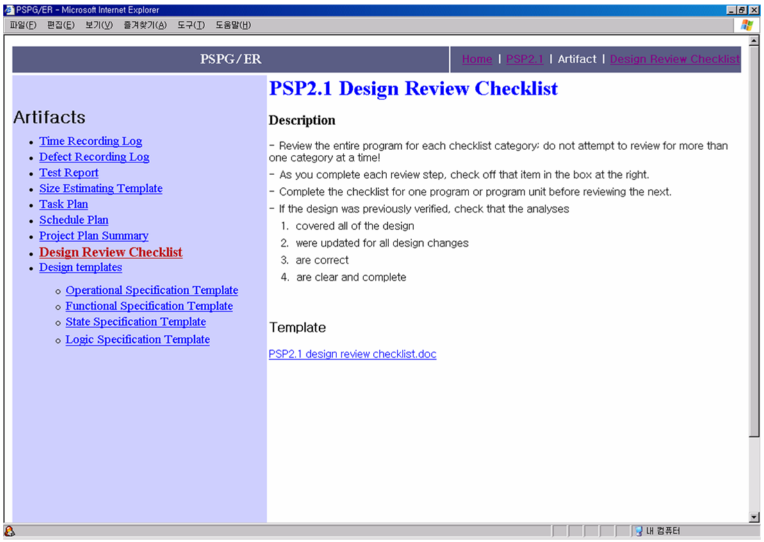
- Record in the Defect Recording log any requirements defects found.

- Record time in the Time Recording Log.

Artifacts

- Functional Specification
- Operational Specification
- State Specification
- Logic Specification
- Time Recording Log
- Defect Recording Log

Fig. 3. An example of an activity page


PSPG/ER - Microsoft Internet Explorer
파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T) 도움말(H)
PSPG/ER
Home | PSP2.1 | Artifact | Design Review Checklist
Artifacts
Time Recording Log
Defect Recording Log
Test Report
Size Estimating Template
Task Plan
Schedule Plan
Project Plan Summary
Design Review Checklist
Design templates
Operational Specification Template
Functional Specification Template
State Specification Template
Logic Specification Template
내 컴퓨터

PSP2.1 Design Review Checklist

Description

- Review the entire program for each checklist category; do not attempt to review for more than one category at a time!
- As you complete each review step, check off that item in the box at the right.
- Complete the checklist for one program or program unit before reviewing the next.
- If the design was previously verified, check that the analyses
  1. covered all of the design
  2. were updated for all design changes
  3. are correct
  4. are clear and complete

Template

PSP2.1 design review checklist.doc

Fig. 4. An example of an artifact page

3.3 Interaction Between PPMT and PSPG/ER

One of main features in PSPG/ER is to store examples of artifacts such as time logs, defect logs, and task/schedule plan. Examples can be stored in a HTML format which is produced in PPMT. Developers can store their artifacts such as time/defect logs and task/schedule plan in an example category of a relevant artifact and their analyses results such as charts, tables, and reports in any experience category. This feature would make it easy to store development experience. Another way of interaction is to provide links to relevant pages. For example, the time log artifact page has a link to the time recording form of PPMT, and in reverse the form contains a link to the artifact page of PSPG/ER. This feature would allow developers to access easily relevant process information.

4 Comparative Analysis of Related Tools

Several PSP support tools have been developed such as Process Dashboard [11], Hackystat [2, 9], and PSPA [10] to help automatic data collection and analyses. Among those tools, Hackystat provides the most similar functionalities to the Jasmine in that both tools provide sensor-based automated data collection. The primary difference between the Jasmine and Hackystat lies in the goal that each aims for. Hackystat is a tool for data collection and analyses rather than a PSP supporting tool since it focuses on only automated data collection and analyses. Therefore, Hackystat does not support the other PSP activities such as planning, plan tracking, and estimation. It also provides the limited data analysis capabilities. This insufficiency of Hackystat is caused by not supporting manual data recording and not collecting automatically all necessary information of the PSP data.

Table 2. Comparison of sensor-based automated data collection

Data		**Jasmine**	**Hackystat**
Time	Time spent on source modification	Eclipse	Eclipse, Visual Studio, JBuilder, IntelliJ Idea
	Time spent on modification of other documents	X	Microsoft Office, OpenOffice, Emacs
	Time spent on code review	X	Jupiter
	Time spent on manual testing	Internet Explorer (for web application testing), Eclipse (for windows application testing)	X
Defect	Failed unit tests	JUnit	JUnit, CPPUnit
	Compile errors	Eclipse	X
	Runtime errors	Eclipse	X
	Post-release bugs	X	Bugzilla, Jira

On the other hand, the Jasmine aims for supporting the whole PSP activities. In the Jasmine, the automatically collected time and defect data are recorded in the time and defect log respectively in order to allow developers to modify the data or insert necessary information to the data, which enables more various data analyses compared to Hackystat. Also, it provides an EPG for the PSP guide incorporating with an ER.

In a comparison of sensor-based automated data collection, while currently the Jasmine does not support as many development-related tools as Hackystat does, it collects automatically time spent on manual testing, compile and runtime errors which Hackystat does not collect, as shown in Table 2. The Jasmine would be easily extended to support various tools by reusing the Hackystat sensors, since Hackystat has been developed as an open source.

5 Conclusion and Future Work

This paper has described the Jasmine developed to help developers perform the PSP. The Jasmine not only automates large parts of data collection to mitigate the problems of manual data recording, but also supports planning and plan tracking. It also provides various kinds of data analyses. These features help developers identify process deficiencies, make a process improvement plan to remove the identified deficiencies, and make a reliable estimate on effort and quality for more effective and efficient process management. Moreover, the Jasmine includes an EPG to allow easy navigation of the PSP process elements and an ER to allow storing and sharing additional process-related information. This integrated EPG and ER would help developers understand and perform the PSP more effectively. A number of collected experiences would be used as resources to solve problems that they can run up against in the PSP process.

This work has been done as a first step of the project that aims to develop a TSP/PSP supporting tool. TSP (Team Software Process) support is planned as one of future works, which includes providing automated data collection and analyses for team data and supporting team planning process and plan tracking, in order to help developers as well as team managers follow the TSP. Also, the sensor-based automated data collection and analyses will be extended continuously with more features. New sensors for various development-related tools (e.g., Visual Studio, Microsoft Office) and new sensor data types (e.g., test coverage, post-release bugs, and code quality metrics) will be developed. We will also provide diverse data analyses to facilitate identification of process deficiencies and product's quality problems. Further, Six Sigma analysis techniques such as control charts, regression analyses will be integrated to help systematic process control. Finally, to improve and extend the tool based on real usage, we will apply this tool to student projects in a class or industrial projects.

Acknowledgement. This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Advancement) (IITA-2006-(C1090-0603-0032)).

References

1. W. S. Humphrey. PSP(sm): A Self-Improvement Process for Software Engineers, SEI Series in Software Engineering, Addison-Wesley Professional, 2005
2. 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.
3. Disney, A. & Johnson, P. Investigating Data Quality Problems in the PSP, Sixth International Symposium on the Foundations of Software Engineering (SIGSOFT'98), Orlando, FL., November, 1998.
4. Felicia Kurniawati, Ross Jeffery. The Long-term Effects of an EPG/ER in a Small Software Organisation, 2004 Australian Software Engineering Conference.
5. L. Scott, L. Carvalho, R. Jeffery, J. D'Ambra and U. Becker-Kornstaedt, "Understanding the use of an Electronic. Process Guide," Information and Software Technology 44. (10), 2002, pp. 601-616.
6. Kurt Schneider, Jan-Peter von Hunnius, "Effective Experience Repositories for Software Engineering," icse, p. 534, 25th International Conference on Software Engineering (ICSE'03), 2003.
7. Louise Scott, Lucila Carvalho, Ross Jeffery, "A Process-Centred Experience Repository for a Small Software Organisation," apsec, p. 603, Ninth Asia-Pacific Software Engineering Conference (APSEC'02), 2002.
8. M. Kellner, U. Becker-Kornstaedt, W. Riddle, J. Tomal, M. Verlage, "Process guides: effective guidance for process participants," in: Proc. of the Fifth International Conference on the Software Process, Chicago, IL, USA, June 1998, ISPA Press, 1998, pp. 11-25.

9. Johnson, P.M.; Hongbing Kou; Agustin, J.M.; Qin Zhang; Kagawa, A.; Yamashita, T., "Practical automated process and product metric collection and analysis in a classroom setting: lessons learned from Hackystat-UH," International Symposium on Empirical Software Engineering, 2004.
10. 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.
11. Process Dashboard, http://processdash.sourceforge.net/
12. Sillitti, A.; Janes, A.; Succi, G.; Vernazza, T., "Collecting, integrating and analyzing software metrics and personal software process data," Euromicro Conference, 2003. Proceedings. 29th , vol., no.pp. 336- 342, 1-6 Sept. 2003.
13. Pekka Abrahamsson, Karlheinz Kautz, “The Personal Software Process: Experiences from Denmark”, EUROMICRO 2002: 367-375.
14. Lutz Prechelt, Barbara Unger, "An Experiment Measuring the Effects of Personal Software Process (PSP) Training," IEEE Transactions on Software Engineering, vol. 27, no. 5, pp. 465-472, May, 2001.
15. Hayes W., and J. Over. The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers, Technical Report SEI-97-TR-001, December 1997.
16. LOCC, http://csdl.ics.hawaii.edu/Tools/LOCC/