DuoTracker: Tool Support for Software Defect Data Collection and Analysis

OLUSEGUN AKINWALE[1] SERGIU DASCALU[1]
[1]Department of Computer Science and Engineering
University of Nevada
Reno, Nevada, USA
{akinwale, dascalus}@cse.unr.edu

MARCEL KARAM[2]
[2]Department of Computer Science
American University in Beirut
Beirut, Lebanon
marcel.karam@aub.lb.edu

Abstract— In today software industry defect tracking tools either help to improve an organization's software development process or an individual's software development process. No defect tracking tool currently exists that help both processes. In this paper we present DuoTracker, a tool that makes possible to track and analyze software defects for organizational and individual software process decision making. To accomplish this, DuoTracker has capabilities to classify defects in a manner that makes analysis at both organizational and individual software processes meaningful. The benefit of this approach is that software engineers are able to see how their personal software process improvement impacts their organization and vice versa. This paper shows why software engineers need to keep track of their program defects, how this is currently done, and how DuoTracker offers a new way of keeping track of software errors. Furthermore, DuoTracker is compared to other tracking tools that enable software developers to record program defects that occur during their individual software processes.

Keywords—software defects; software anomalies; defect classification; PSP; CMM; IS0-9001.

I. INTRODUCTION

Consumers of software products expect a certain level of quality when they purchase computer programs. An example is in the purchase of a media player software. When consumers purchase a media player they expect the control symbols for stopping, recording and playing media files to be similar to that of an actual CD or DVD player. However, for many software applications the quality desired by the users is not that obvious. Software companies gain this information through several iterations of a product's development. Then, once a software company has been able to figure out the need of its customer base, it must consistently produce products that match the expectation of its users. Research has convincingly shown that the only way to consistently produce products that meet the desire of one's users is to establish a software process [1, 2]. Once a software process has been instituted in an organization the effectiveness of the software process will need to be evaluated frequently. Today, mature software companies evaluate their process using either the Capability Maturity Model (CMM) [3] or the ISO 9001 standards [4]. These two standards define the requirements for an ideal company. A company's software process is then compared against the quality models of CMM or IS0 9001. There are also improvement methods such as IDEAL [5] and SPICE [6] that suggest ways for improving the effectiveness of the current software process of a software company. While the software engineering literature has shown that the software process does help improve the quality of software products, researchers have also noted that the personnel who make up the software development team also contribute to the quality of a product [7]. Consequently, there is a need for software engineers to continuously accomplish their assigned tasks at a level equivalent or better than their previous work. If engineers accomplish their tasks with consistent quality, this will make project assignments easier for software managers and will reduce the chances of software developers being assigned tasks for which they are ill prepared for.

Unfortunately, in today's software industry, software engineers do not quantitatively assess the quality of their work in a thorough way. The only assessment they get is from their bosses (project leaders) during annual reviews. These assessments are typically subjective and they can give a software engineer a false idea of his or her skill capability.

A remedy for this situation is the Personal Software Process (PSP) developed by Watts Humphrey [8, 9]. PSP outlines steps individual software engineers can use to quantitatively assess the quality of their work.

PSP requires its users to record the following: (i) the time spent in each phase of software development, (ii) the size of program, and (iii) the phase in which a defect was injected and the phase it was removed. The data is used to generate quality measures that enable software engineers evaluate the quality of their work. Currently, there are tools that enable software engineers to collect data necessary to perform PSP quality assessment but recent research shows that software engineers are not using these tools, and the main reason they cite for not using PSP tools is that data collection process is too cumbersome [10]. To solve this problem, researchers have created PSP tools that automate the data collection process, but so far there is no research paper we could find that shows software engineers in industry are using the new set of PSP automating tools such Hackystat [11] and PSPA (Personal Software Process Assistant) [12].

In this paper we present a tool called DuoTracker that simultaneously collects data needed for both PSP and CMM/ISO 9001 quality analysis. As such, it represents an innovative software defect tracking solution that allows collecting and analyzing software defect data pertaining to both organization-wide projects and individual software engineering work.

The paper, in its remaining part, is organized as follows: Section 2 describes the challenges identified in software defect tracking, Section 3 describes in detail our proposed solution (the Duo Tracker tool), Section 4 provides a comparison with related work, Section 5 outlines directions of future work, and Section 6 presents our conclusions.

II. PROBLEM

Despite the fact that PSP is not been used in industry, practicing software engineers do see how PSP can significantly improve the quality of their work. This is especially true for independent consultants and developers who do not have the resources to implement a software process that meets the quality standards required by CMM and ISO 9001.

The reason mostly cited for not using PSP is similar to those cited in [13]. The common reasons for not adopting PSP are the time log and defect log associated with PSP. Software engineers using PSP are expected to record the start and stop time for a task. They are also expected to record interruptions that occur in between the start and the stop time of a task. Interruption is defined as anything that takes one away from the task at hand, including answering phone calls and other office activities.

Defect logging is another activity that prevents software engineers from adopting PSP. PSP requires software engineers to record every defect found in every phase, including defects found in the review, inspection, compiling, and testing phases. These data are recorded in the defect recording logs. Typically, software engineers find the activity of recording every defect, particularly compiler defects, excessively time consuming.

From the information provided above one could easily conclude that the reason software engineers are not using PSP is related to the intensive manual data entry involved in PSP.

To address this problem, in an effort to solve the manual entry issue that prevents the adoption of PSP by software engineers, the software engineering researchers and practitioners have created tools such as Hackystat [11], PSPA [12], LEAP System [14] PSP Studio [15], and Dashboard [16].

Research result from [12] shows that automating PSP data entry does not result in substantial amount of PSP tool adoption. The main reason for the low adoption of PSP tools (despite the automation of the data entry currently available) is related to the fact that users of tools such as PSP Dashboard, LEAP system and PSP Studio have to switch often between the recording tool and their development tool.

Based on the research result of [12], [13], and [17] one can conclude that the adoption barrier to PSP tools is related to the following issues:

1. Manual entry of data;
2. Switching between applications;
3. Too rigid data collection.

III. SOLUTION

Current PSP tools view PSP as an independent software process and, consequently, the tools being created to assist in the data collection and analysis of PSP are standalone applications. To be precise, by standalone applications we mean in this context PSP applications that are not being integrated into applications already being used by the software engineers for tasks performed during a particular software development phase. An example of non standalone application is the popular IDE (Integrated Development Environment) used by many developers to develop software products. This type of integrated tool suite typically contains several other tools that could be standalone applications by themselves, e.g., editors, debuggers, compilers and “code beautifiers”.

Inspired by the IDE model, the authors of this paper have taken a different approach to the issue of PSP adoption. Rather than creating a standalone application we have integrated a PSP defect tracking tool into an organization-wide defect tracking application that can be used by software companies adhering to the standards of IS0 9001 and CMM.

By doing these, we also achieve the original goal of PSP. Specifically, PSP was not meant to be an independent process, but a process that complements the TSP (Team Software Process) [18] and the CMM.

To demonstrate the proposed approach we designed and implemented a tool called DuoTracker. DuoTacker implements the eleven mandatory categories for IEEE Standard 1044-1993 [19, 20]. IEEE Standard 1044-1993 is a Defect Classification Scheme (DCS). Currently, there are several defect classification schemes, but we chose to work with the IEEE Standard 1044-1993 because it satisfies the CMM and IS0 9001 defect tracking requirements and covers the entire software development lifecycle.

In [19] and [20] software defects are referred to as *software anomalies* for semantic reasons. More exactly, references [19] and [20] define anomaly as “any condition that deviates from the expected based on requirements specifications, design documents, user documents, standards, etc. or from someone’s perceptions or experiences.”

DuoTracker also implements a PSP defect logger similar to the one described in [21]. The PSP has its own way of classifying defects, hence DuoTracker collects defect information from the user in a manner that allows the user to know where each data set is going to be used.

A. *DuoTracker Solution to Rigid Data Collection*

With respect to time logging, instead of taking the approach of [21], where only the time spent on each fix is recorded, DuoTracker provides facilities for more detailed information gathering. Specifically, the DuoTracker system automatically enters the date the defect was found based on the time the defect was submitted and allows the user to record the estimated fix time for a defect as well as the actual fix time for the defect. The timing parameters recorded by DuoTracker are similar to the ones collected in Watts's defect logging form [21].

Another rigid data collection process is the recording of compile errors during the development phase. Currently, DuoTracker allows the user to classify defects as PSP environment type defects that occurred in the PSP compilation phase. In DuoTracker we decided not to automate the recording of compilation errors because of the following reasons:

1. An implementation of automated compile time error recording feature would be dependent on the tool or programming language used;
2. It would be difficult to associate compile time errors to a specific project, as there is no generic standard for associating source files to a project. The management of a software organization or the tool vendors typically determine the structure of the project source files;
3. We believe that compile time errors should be recorded only when they occur after the development phase. An example is when a software product fails to compile in its test phase or deployment phase.

B. DuoTracker Solution to Application Switching

DuoTracker addresses the issue of application switching by integrating itself into a standard application used for defect data collection and quality analysis. Most software companies, regardless of the maturity of their software process, use a defect tracking tool [1, 3, 13, 21]. This can be easily seen in the popularity of the Bugzilla tool [22], familiar in many open source projects. However, open source software engineers are not known to be the greatest fans of software processes. Typically, they want a very flexible software process, which does not restrict their creative abilities.

By integrating a PSP defect tracking tool into an organization-wide defect tracking tool we allow the software engineers to seamlessly record PSP data. An example is the compile error in the test or deployment phase. The software engineer responsible for a product will typically be notified of a defect in the product by a defect tracking tool component of a standard organization-wide defect tracking environment such as Bugzilla, Tracker [23], Helis [24], or Ozibug [25]. In DuoTracker, once the individual assigned a defect opens the assigned defect record for viewing, the tool allows the assignee to include data needed for PSP quality analysis. The assignee is not required to fill in PSP data sections and, actually, there is checkbox an assignee must check if he or she wants an assigned defect to be used in calculating his or her personal quality measure.

C. DuoTracker Solution to Manual Data Entry

DuoTracker automates PSP data entry by copying over defect description fields from the IEEE Standard 1044-1993 data entry form into PSP data entry form. Defect description fields that are unique to PSP need to be entered manually by the assignee.

D. Potential Problem

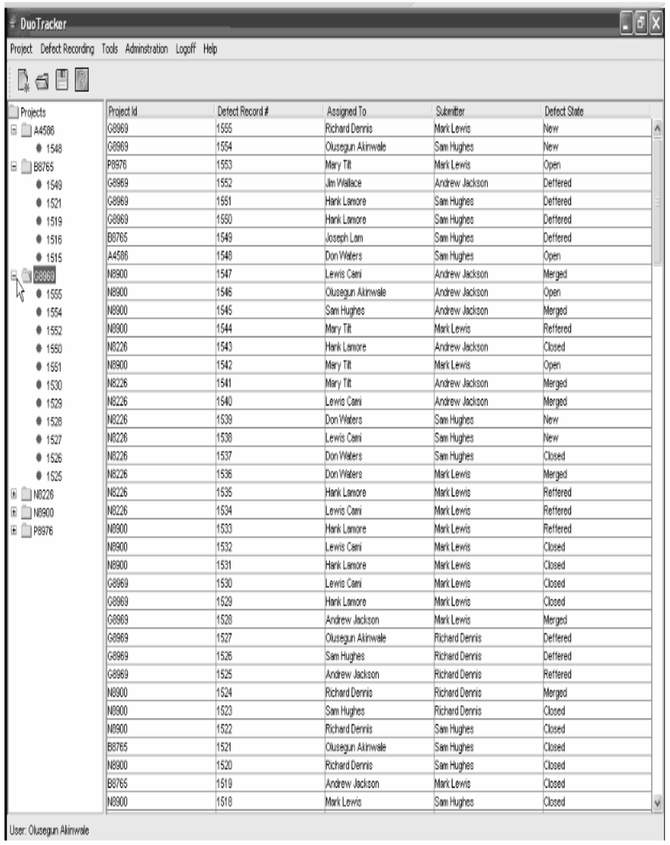
A potential problem with DuoTracker is the security of the PSP data provided by users. Currently, DuoTracker addresses this issue by restricting viewing and updating of PSP data records to assignees.

E. DuoTracker Defect Viewer

DuoTracker's defect viewer, shown in Fig. 1, allows users to view all the defect records stored in the DuoTracker system. The viewer displays five important fields of a defect record, as follows: (1) project ID, (2) defect record number, (3) the person to whom the defect is assigned, (4) the submitter of the defect, and (5) the status of the reported defect. Clicking on an entry will allow one to view an existing defect record to be displayed in a dialog box. The name of the logged-on user is shown on the bottom left corner of the viewer. Additionally, on the left side of the viewer there is a project directory that allows the users to quickly view defects associated with a specific project.

F. DuoTracker Updating of an Assigned Defect

Clicking a defect record in the defect viewer will produce a dialog box similar to the one shown in Fig 2. In this figure, the PSP tab is visible because the assigned user is the same with the logged-on user. The highlighted tab in Fig. 2 tells the user that he or she is filling out the description dialog box.

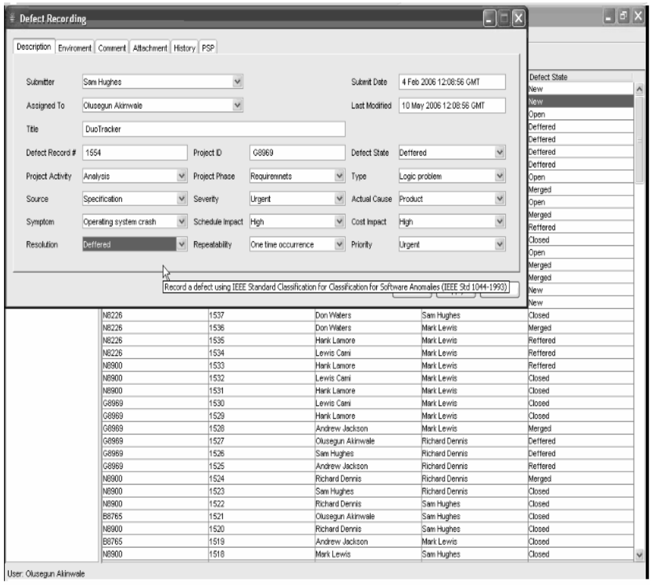


Project Id	Defect Record #	Assigned To	Submitter	Defect State
G8969	1555	Richard Dennis	Mark Lewis	New
G8969	1554	Olusegun Akinwale	Sam Hughes	New
P8976	1553	Mary Tilt	Mark Lewis	Open
G8969	1552	Jim Wallace	Andrew Jackson	Deffered
G8969	1551	Hank Lamore	Sam Hughes	Deffered
G8969	1550	Hank Lamore	Sam Hughes	Deffered
B8765	1549	Joseph Lam	Sam Hughes	Deffered
A4586	1548	Don Waters	Sam Hughes	Open
N8900	1547	Lewis Cami	Andrew Jackson	Merged
N8900	1546	Olusegun Akinwale	Andrew Jackson	Open
N8900	1545	Sam Hughes	Andrew Jackson	Merged
N8900	1544	Mary Tilt	Mark Lewis	Reffered
N8226	1543	Hank Lamore	Andrew Jackson	Closed
N8900	1542	Mary Tilt	Mark Lewis	Open
N8226	1541	Mary Tilt	Andrew Jackson	Merged
N8226	1540	Lewis Cami	Andrew Jackson	Merged
N8226	1539	Don Waters	Sam Hughes	New
N8226	1538	Lewis Cami	Sam Hughes	New
N8226	1537	Don Waters	Sam Hughes	Closed
N8226	1536	Don Waters	Mark Lewis	Merged
N8226	1535	Hank Lamore	Mark Lewis	Reffered
N8226	1534	Lewis Cami	Mark Lewis	Reffered
N8900	1533	Hank Lamore	Mark Lewis	Reffered
N8900	1532	Lewis Cami	Mark Lewis	Closed
N8900	1531	Hank Lamore	Mark Lewis	Closed
G8969	1530	Lewis Cami	Mark Lewis	Closed
G8969	1529	Hank Lamore	Mark Lewis	Closed
G8969	1528	Andrew Jackson	Mark Lewis	Merged
G8969	1527	Olusegun Akinwale	Richard Dennis	Deffered
G8969	1526	Sam Hughes	Richard Dennis	Deffered
G8969	1525	Andrew Jackson	Richard Dennis	Reffered
N8900	1524	Richard Dennis	Richard Dennis	Merged
N8900	1523	Sam Hughes	Richard Dennis	Closed
N8900	1522	Richard Dennis	Sam Hughes	Closed
B8765	1521	Olusegun Akinwale	Sam Hughes	Closed
N8900	1520	Richard Dennis	Sam Hughes	Closed
B8765	1519	Andrew Jackson	Mark Lewis	Closed
N8900	1518	Mark Lewis	Sam Hughes	Closed

User: Olusegun Akinwale

Figure 1. DuoTracker Defect Viewer

Moving the cursor over the dialog box initiates the displaying of a “tooltip” that describes what the dialog is meant for. For example, after pressing the PSP tab the user is taken to the PSP dialog box, as shown in Figure 3. Here, by moving the cursor over the dialog box a tooltip is displayed telling the user that the values entered in this dialog box are for classifying the defect using the standard PSP classification. The checkbox at the bottom of the dialog box allows users to select their PSP data records.



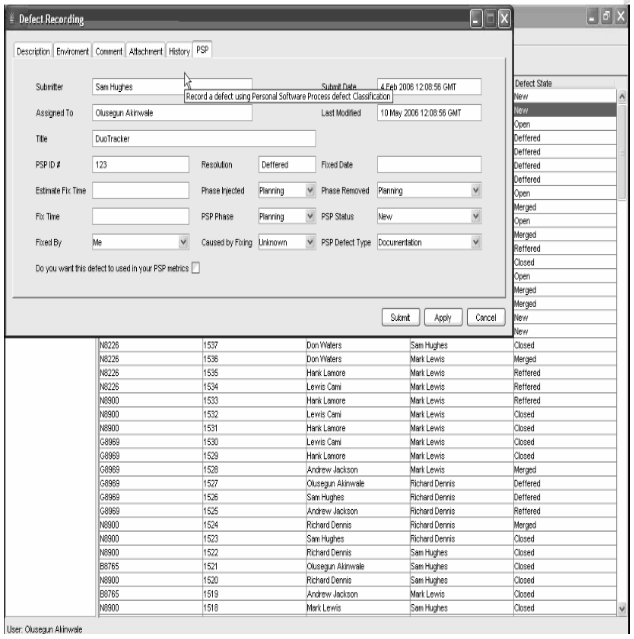
Submitter	Sam Hughes			Submit Date	4 Feb 2006 12:08:56 GMT
Assigned To	Olusegun Akinwale			Last Modified	10 May 2006 12:08:56 GMT
Title	DuoTracker				
Defect Record #	1554	Project ID	G8969	Defect State	Deffered
Project Activity	Analysis	Project Phase	Requiremnets	Type	Logic problem
Source	Specification	Severity	Urgent	Actual Cause	Product
Symptom	Operating system crash	Schedule Impact	High	Cost Impact	High
Resolution	Deffered	Repeatability	One time occurrence	Priority	Urgent

Record a defect using IEEE Standard Classification for Classification for Software Anomalies (IEEE Std 1044-1993)

				Defect State
				New
				New
				Open
				Deffered
				Deffered
				Deffered
				Deffered
				Open
				Merged
				Open
				Merged
				Reffered
				Closed
				Open
				Merged
				Merged
				New
				New
N8226	1537	Don Waters	Sam Hughes	Closed
N8226	1536	Don Waters	Mark Lewis	Merged
N8226	1535	Hank Lamore	Mark Lewis	Reffered
N8226	1534	Lewis Cami	Mark Lewis	Reffered
N8900	1533	Hank Lamore	Mark Lewis	Reffered
N8900	1532	Lewis Cami	Mark Lewis	Closed
N8900	1531	Hank Lamore	Mark Lewis	Closed
G8969	1530	Lewis Cami	Mark Lewis	Closed
G8969	1529	Hank Lamore	Mark Lewis	Closed
G8969	1528	Andrew Jackson	Mark Lewis	Merged
G8969	1527	Olusegun Akinwale	Richard Dennis	Deffered
G8969	1526	Sam Hughes	Richard Dennis	Deffered
G8969	1525	Andrew Jackson	Richard Dennis	Reffered
N8900	1524	Richard Dennis	Richard Dennis	Merged
N8900	1523	Sam Hughes	Richard Dennis	Closed
N8900	1522	Richard Dennis	Sam Hughes	Closed
B8765	1521	Olusegun Akinwale	Sam Hughes	Closed
N8900	1520	Richard Dennis	Sam Hughes	Closed
B8765	1519	Andrew Jackson	Mark Lewis	Closed
N8900	1518	Mark Lewis	Sam Hughes	Closed

User: Olusegun Akinwale

Figure 2. Defect Recording in DuoTracker Using IEEE Standard Classification for Software Anomalies (Organization Level)



Submitter	Sam Hughes			Submit Date	4 Feb 2006 12:08:56 GMT
Assigned To	Olusegun Akinwale			Last Modified	10 May 2006 12:08:56 GMT
Title	DuoTracker				
PSP ID #	123	Resolution	Deffered	Fixed Date	
Estimate Fix Time		Phase Injected	Planning	Phase Removed	Planning
Fix Time		PSP Phase	Planning	PSP Status	New
Fixed By	Me	Caused by Fixing	Unknown	PSP Defect Type	Documentation

Do you want this defect to used in your PSP metrics ☐

Record a defect using Personal Software Process defect Classification

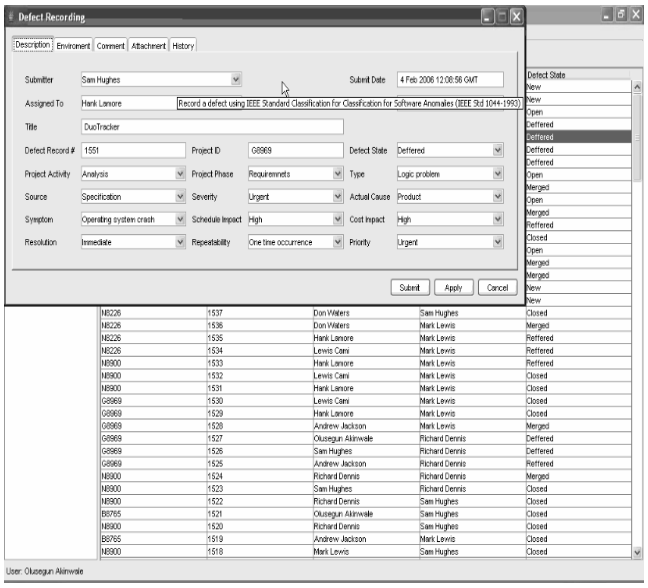
				Defect State
				New
				New
				Open
				Deffered
				Deffered
				Deffered
				Deffered
				Open
				Merged
				Open
				Merged
				Reffered
				Closed
				Open
				Merged
				Merged
				New
				New
N8226	1537	Don Waters	Sam Hughes	Closed
N8226	1536	Don Waters	Mark Lewis	Merged
N8226	1535	Hank Lamore	Mark Lewis	Reffered
N8226	1534	Lewis Cami	Mark Lewis	Reffered
N8900	1533	Hank Lamore	Mark Lewis	Reffered
N8900	1532	Lewis Cami	Mark Lewis	Closed
N8900	1531	Hank Lamore	Mark Lewis	Closed
G8969	1530	Lewis Cami	Mark Lewis	Closed
G8969	1529	Hank Lamore	Mark Lewis	Closed
G8969	1528	Andrew Jackson	Mark Lewis	Merged
G8969	1527	Olusegun Akinwale	Richard Dennis	Deffered
G8969	1526	Sam Hughes	Richard Dennis	Deffered
G8969	1525	Andrew Jackson	Richard Dennis	Reffered
N8900	1524	Richard Dennis	Richard Dennis	Merged
N8900	1523	Sam Hughes	Richard Dennis	Closed
N8900	1522	Richard Dennis	Sam Hughes	Closed
B8765	1521	Olusegun Akinwale	Sam Hughes	Closed
N8900	1520	Richard Dennis	Sam Hughes	Closed
B8765	1519	Andrew Jackson	Mark Lewis	Closed
N8900	1518	Mark Lewis	Sam Hughes	Closed

User: Olusegun Akinwale

Figure 3. Defect Recording in DuoTracker Using PSP Defect Classification (Individual Level)

G. DuoTracker Updating of a Non-assigned Defect

When the logged-on user clicks on a defect record that is not assigned to him or her the resulting dialog box will not have a PSP tab (Fig. 4).



Submitter	Sam Hughes			Submit Date	4 Feb 2006 12:08:56 GMT
Assigned To	Hank Lamore				
Title	DuoTracker				
Defect Record #	1551	Project ID	G8969	Defect State	Deffered
Project Activity	Analysis	Project Phase	Requiremnets	Type	Logic problem
Source	Specification	Severity	Urgent	Actual Cause	Product
Symptom	Operating system crash	Schedule Impact	High	Cost Impact	High
Resolution	Immediate	Repeatability	One time occurrence	Priority	Urgent

Record a defect using IEEE Standard Classification for Classification for Software Anomalies (IEEE Std 1044-1993)

				Defect State
				New
				New
				Open
				Deffered
				Deffered
				Deffered
				Deffered
				Open
				Merged
				Open
				Merged
				Reffered
				Closed
				Open
				Merged
				Merged
				New
				New
N8226	1537	Don Waters	Sam Hughes	Closed
N8226	1536	Don Waters	Mark Lewis	Merged
N8226	1535	Hank Lamore	Mark Lewis	Reffered
N8226	1534	Lewis Cami	Mark Lewis	Reffered
N8900	1533	Hank Lamore	Mark Lewis	Reffered
N8900	1532	Lewis Cami	Mark Lewis	Closed
N8900	1531	Hank Lamore	Mark Lewis	Closed
G8969	1530	Lewis Cami	Mark Lewis	Closed
G8969	1529	Hank Lamore	Mark Lewis	Closed
G8969	1528	Andrew Jackson	Mark Lewis	Merged
G8969	1527	Olusegun Akinwale	Richard Dennis	Deffered
G8969	1526	Sam Hughes	Richard Dennis	Deffered
G8969	1525	Andrew Jackson	Richard Dennis	Reffered
N8900	1524	Richard Dennis	Richard Dennis	Merged
N8900	1523	Sam Hughes	Richard Dennis	Closed
N8900	1522	Richard Dennis	Sam Hughes	Closed
B8765	1521	Olusegun Akinwale	Sam Hughes	Closed
N8900	1520	Richard Dennis	Sam Hughes	Closed
B8765	1519	Andrew Jackson	Mark Lewis	Closed
N8900	1518	Mark Lewis	Sam Hughes	Closed

User: Olusegun Akinwale

Figure 4. Displaying a Non-Assigned Defect in DuoTracker

IV. COMPARISON WITH RELATED PRODUCTS

There are several PSP tools available but only Hackystat and PSPA (PSP Assistant) have features that are somewhat related to the ideas demonstrated in DuoTracker.

Hackystat attempts to solve the PSP adoption issue by creating sensors into development tools such as JBuilder, Emacs and the testing suite JUnit. It also has sensors for Bugzilla, arguably the most popular open source defect tracking tool. The collected defect data from these tools are sent to a Hackystat server. The server performs analysis on the received data at regular intervals [11].

Among all the available PSP tools, PSPA [12] is the closest to DuoTracker. PSPA is a client server system. The client for PSPA is a plug-in for the Eclipse IDE environment. The PSPA client user can view non-private PSP data of a team. The PSPA client allows user to have a dual project schedule view, one view showing the project schedule assigned by a supervisor and the other showing a user’s personal project schedule [12].

DuoTracker differentiates itself from PSPA and Hackystat by not restricting its solution to a specific tool vendor or a particular programming language. Furthermore, DuoTracker differs from PSPA by providing a means for recording defects that occur outside the PSP compile phase. Also, DuoTracker distinguishes itself from from PSPA and Hackystat by providing means for filtering out unwanted PSP data. Lastly, DuoTracker is significantly different from Hackystat and PSPA in that it allows a user to compare the quality of his or her work to the quality of the product being produced by an organization.

V. Future Work

Currently we consider four main directions of future development for DuoTracker, as indicated next.

The first direction consists of developing metrics and visualization formats that will enable users to see if software organizations are diminishing or improving the quality of their work. This will be particularly useful for open source projects as it will enable volunteers to prevent having their names associated with a project with little chance of making it in the software market or contributing to the software development community.

The second direction for DuoTracker's future developments consists of implementing it as a web based client. Given the accessibility and convenience provided by web applications, this will most likely increase the usage and effectiveness of the DuoTracker tool.

The third direction of future work for DuoTracker is to integrate a database system that prevents unauthorized users from viewing PSP data through statistical loopholes.

Finally, as the DuoTracker tool has been used so far only in an academic setting (for graduate projects in software engineering at the University of Nevada, Reno) we need to further exercise the tool in more complex applications, better assess its capabilities, and identify additional needed features that would increase its usability.

VI. Conclusions

In this paper we have presented a tool, called DuoTracker, that allows software engineers collect defect information needed for analyzing software quality at both the organization and individual software process levels.

This tool contributes to the field of software engineering by demonstrating that it is possible to enable developers to perform software process defect data collection and analysis activities at both individual and organizational levels. Currently, defect data collection and analysis tools are made only for either an organization's or a single person's use.

The advantage of having a unique tool that takes care of all process and quality needs of a software developer comes primarily from making it easy and straightforward to quickly identify trends in software defect reporting and fixing.

The paradigm illustrated by DuoTracker can be used in tools that support either a specific development activity or help monitor the entire software process.

As far as we see it, the main limitation of this project is security. Some developers will not feel secure having their personal data being combined with organizational data, as people interested in developing applications using this paradigm will have to show their potential customers that their data is secured.

The main direction of future work for the DuoTracker project is the development of metrics and visualizations that will enable users to accurately assess if a software organization is diminishing or improving the quality of its work.

References

[1] Sommerville, I., Software Engineering, 7th Edition, Addison-Wesley, 2004.

[2] Pressman, R., Software Engineering: A Practitioner's Approach, 6th Edition, McGraw-Hill, 2004.

[3] Paulk, M., Curtis, B., Chrissis, M., and Weber, C., "Capability Maturity Model for Software (version 1.1)," Carnegie Melon University, Software Engineering Institute, CMU/SEI-93-TR-024, 1993.

[4] International Organization for Standardization, ISO-9001, accessed June 10, 2006 at www.iso.org

[5] McFeeley, R., IDEAL: User Guide for Software Process Improvements, CMU/SEI-96-HB-001, 1996.

[6] SPICE: Software Process Improvement and Capability Determination website, Software Quality Institute, Griffith University, Australia, accessed June 10, 2006 at http://www.sqi.gu.edu.au/spice/

[7] Fuggetta, A., "Software process: a roadmap," in Proceedings of the Conference on the Future of Software Engineering, 2000, pp. 25-34.

[8] Humphrey, W.S., Personal Software Process (PSP), Carnegie Melon, Software Engineering Institute, Pennsylvania, USA, Technical Report ESC-TR-2000-022, December 2000.

[9] Börstler, J., Carrington, D., Hislop, G., Lisak, S., Olson, K., and Williams, L. "Teaching PSP: challenges and lessons learned," IEEE Software, vol. 19, no. 5, 2002, pp. 42-48.

[10] Humphrey, W. S. "The Personal Software Process: status and trends," IEEE Software, vol. 17, no. 6, 2000, pp. 71-75.

[11] HackyStat Development Site, accessed June, 2006 at http://www.hackystat.org/hackyDevSite/home.do

[12] Sison, R., Diaz, D., Lam, E., Navarro, D., and Navarro, J., "Personal Software Process (PSP) Assistant," Proceedings of APSEC, 2005, pp. 687-696.

[13] Johnson, P.M., Kou, H., Agustin, J., Chan, C., Moore, C., Miglani, J., Zhen, S., and Doane, W. "Beyond the Personal Software Process: metrics collection and analysis for the differently disciplined," Proceedings of the International Conference on Software Engineering, 2003, pp. 641-646.

[14] Moore, C.A., "Lessons learned from teaching reflective software engineering using the Leap toolkit," Proceedings of the IEEE International Conference on Software Engineering (ICSE-2000).

[15] PSP Studio Page, University of Montana, Dept. of Computer Science, accessed June 2006 at http://www.cs.umt.edu/RTSL/dsstud/psp/psps.htm

[16] The Software Process Dashboard Initiative, SourceForge.net, accessed June 2006 at http://processdash.sourceforge.net/

[17] Sillitti, A., Janes, A., Succi, G., and Vernazza, T., "Collecting, integrating and analyzing software metrics and Personal Software Process data," Proceedings of EUROMICRO, 2003, pp. 336-342.

[18] Humphrey, W.S., "Pathways to process maturity: the Personal Software Process and Team Software Process," SEI Interactive, vol. 2, no. 2, 1999.

[19] IEEE Std 1044-1993, IEEE Standard Classification for Software Anomalies, 1993.

[20] IEEE Std 1044.1-1995, IEEE Guide to Classification for Software Anomalies, 1995.

[21] Humphrey, W.S., Introduction to the Personal Software Process: A Discipline for Software Engineering, Addison-Wesley, Reading, Mass., 1996.

[22] The Mozilla Organization, Bugzilla, accessed June 2006 at http://www.bugzilla.org/

[23] PrimaSoft PC Software, Bug Tracker Deluxe, accessed June 2006 at www.primasoft.com/deluxeprg/bugtracker_software_overview.htm

[24] Helis 1.0.3 Alpha, accessed June 2006 at linux.softpedia.com/get/Programming/Bug-Tracking/Helis 962.shtml

[25] Ozibug: A Web-based Bug Tracking System, accessed June 2006 at www.ozibug.com