Tool Support for Personal Software Process

Jouni Lappalainen

Department of Information Processing Science, University of Oulu,
P.O.Box 3000, FIN-90014 OULU, Finland
Jouni.Lappalainen@oulu.fi

Abstract. Improving the software development process is something that many organizations aim for. Many methods have been devised to reach this goal, one of which focuses on the personal level of software development, namely the Personal Software Process[SM1] (PSP[SM]). There is a dire need for automated tool support for PSP, since the method is laborious if used manually. During four university-level courses several tools were studied and later evaluated using feature analysis. As one result, a requirements set for an ideal PSP tool was devised. The results of the evaluation showed that none of the evaluated tools fulfilled the acceptance threshold set for a tool, though one of them can be modified so that it could be used within the setting of an academic PSP course.

1 Introduction

Structured and disciplined development process is commonly found to be helpful in software quality improvement efforts. Methods and models such as SW-CMM or CMMI have traditionally been more focused towards an organizational approach of software engineering. In the end, the work is done by individuals, and their capabilities to address the requirements of organizational software engineering models have been at least varied if not limited. As a personal level software development and quality process, Watts S. Humphrey developed the Personal Software Process[SM] in the mid-1990s. If the PSP is used as described in [1] and followed faithfully, it is very laborious due to the constant measuring of own work. The "tools" provided in [1] are essentially document templates and statistical formulae, and these have been found inadequate to at least classroom usage during several courses in University of Oulu [2] [3] and Copenhagen Business School [4]. The practical need in teaching has prompted the question: how can the Personal Software Process be supported with software tools? In order to find the answer to this an analysis of what kinds of tools exist and ideal tool requirements is conducted.

The problem is approached qualitatively, by studying the feedback given by students in PSP courses. This feedback, combined with personal experiences from the same courses as well as a comprehensive literature study, is used as the material while evaluating the software tools. To select the appropriate way to evaluate software tools, methodology identified during the DESMET-project is used [5]. The tools for

[SM1] Personal Software Process and PSP are service marks of Carnegie Mellon University.

F. Bomarius and S. Komi-Sirviö (Eds.): PROFES 2005, LNCS 3547, pp. 545–559, 2005.

the analysis cover some of those used in classroom setting as well as those not used during the courses. This is to give more validity to the results of the study, if not the analysis itself. The selection is also to be more of a descriptive set of the types of tools, rather than a comprehensive set of tools available.

Rest of the paper has been organized as follows. The following sections give a short overview of PSP method as well as tools used to support it. This is followed by the description of the evaluation method and evaluation context, after which a section summarizing some of the features of an ideal PSP tool is presented. Results of the evaluations of the tools are outlined and briefly discussed in section 6. Finally, section 7 provides some concluding remarks.

2 Overview of the PSP Method

The PSP aims to enable an individual software engineer to improve the quality of the product by improving the quality of the process she uses to develop the product. This is achieved by introducing elements to the process that help the user of the PSP to plan, track and analyze both the product and the process, as well as manage the development project on a personal level. The PSP is often taught as a course, where these elements to the process are gradually introduced into the initial, or the baseline process, as is suggested by [1]. After augmenting the baseline process with product and process measurements, standard reporting features and the general process structure, the process elements introduced deal with personal project management issues. This PSP1-level adds test reporting as well as size and effort estimating into the process elements. The following PSP2-level gives the user tools for software quality management on a personal level, code and design reviews to name a few. Finally, the PSP3-level introduces cyclical development concepts to the process in order to scale the PSP for larger projects.

According to Humphrey [6], the PSP "is based in part on the quality management principles of W. Edwards Deming and Joseph M. Juran. In a more general sense, these methods follow principles that Frederick Winslow Taylor introduced over 100 years ago." Some founding principles behind the PSP can be seen to be process tracking, project planning, product and process measurement, analyzing of that data, and process improvement based on the analysis. Here only a cursory overview of the PSP is given, and for example many of the methods, process elements and statistical formulas that the PSP uses cannot be discussed. For more information about the PSP, the reader is encouraged to refer to [1].

As so many other software engineering issues, the PSP is by no means a single solution to all possible problems, or "the silver bullet" if wished. Considerably crucial points for critique have been found over the years, one of which is the already mentioned amount of work the PSP demands from the user [7]. Disney and Johnson have also studied the problems in collecting the PSP data [8], and found that the data collected doesn't always properly and accurately reflect the reality. This is partly due to the sometimes overwhelming amount of data that needs to be collected, and partly to the fact that when recording some data from the process, the software engineer needs