Non-invasive Measurement of the Software Development Process

Alberto Sillitti  
DIST – Università di Genova  
alberto@dist.unige.it

Andrea Janes  
Libera Università di Bolzano  
Andrea.Janes@umbz.it

Giancarlo Succi  
Libera Università di Bolzano  
Giancarlo.Succi@umbz.it

Tullio Vernazza  
DIST – Università di Genova  
tullio@dist.unige.it

Abstract

Usually collecting data regarding software requires a lot of time and effort. Moreover, these data are often unusable because they are affected by too many errors especially when these measurements are most important: during high stressing periods, as instance because of an approaching important deadline. An automated tool to collect such data in a non-invasive way should partially solve the problem allowing teams to understand better how to improve their work.

This paper presents PROM (PRO Metrics), an automated tool for collecting and analyzing software metrics and personal software process (PSP) data.

1. Introduction

Understanding how an enterprise really works is not an easy task and requires a huge amount of time spent interviewing both managers and employees. Managers of specific enterprise functions should know, in a very detailed way, how tasks are performed. Often, they know the big picture but their knowledge of the details is very different from how tasks are really performed. That could happen due to several reasons. Among them, there are the following:

- There are no or limited quantity of data regarding how people work
- People claim to work in a certain way but they actually work differently

The former is possible because inspections are very expensive, do not produce immediate benefits and too many errors or missing data badly affect the analysis process. These errors appear mostly in critical periods, when the data correctness should help to understand better the situation such as during high stress working periods.

The latter is possible when management introduces wrong or too complex processes that people hardly follow but they claim to do. In this way, there is a mismatch between the real process and managers knowledge.

Unfortunately, it is difficult to collect useful data in software development environments because managers and developers do not consider measurement an important activity, compared to coding. Moreover, it is hard to collect data manually and there are only few, very expensive tools to collect the data semi-automatically. Semi-automatically means that still the human intervention is heavily involved, especially in the critical task of collecting process measures.

Such tools suffer from severe limitations. (A) Mostly, they deal with product measures; (semi)automatic collection of process measures is nearly always ignored. (B) Often, they are not integrated in the “usual” working context of developers and managers; the developer is required to invoke such tools explicitly. This lack of integration affects the precision of the collected data. Sometimes, it even happens that measures are collected later in the process than expected, just to comply to given process guidelines; this results in spurious data. (C) Even when some of the tools a developer or a manager may use supports the collection of a few, mostly product-oriented measures, such measures are not automatically combined with the measures collected from other tools the user or the manager may be using. This still requires the developer and the manager to store explicitly the data with the consequent problems mentioned above; therefore, it limits the possibility of extracting sound, sensitive information from the combined analysis of measures.

2. Collecting Data

The lack of fully automated process and product measures collection tools affects severely the PSP and other similar approaches. Empirical investigations on the PSP evidence that measures data are usually not collect at all or with the required precision to produce meaningful results [4].

Automated data collection, integrated in popular software, helps to solve both problems and introduce further benefits such as help to implement the Activity-Based Costing (ABC) as accounting technique [1]. These
automated data collection is possible wherever people spend most of their time performing computer-based activities such as software development.

Engineers need to measure relevant variables of a process to understand and control it. That also happens in software engineering. The main resource in many companies, such as software companies, is human resource. For this reason, software engineers are mainly interested in: human effort needed to complete a task, quality, cost, and time required.

Process metrics describe process qualities such as effort required, production time, steps of a task, etc. These qualities can be evaluated through the acquisition of measurable properties such as editing time, number and type of changes in a file, usage patterns, etc.

Most of management costs are human resources costs: experience, skills, etc. Moreover, productivity of very good employee is tens times better than average.

For these reasons, it is very important to understand how top developers work and force all to adopt a process that helps them to achieve the best results possible.

The Personal Software Process (PSP) [3] and several other research initiatives evidence the importance of collecting such data in software engineering and cross-analyzing them to have a benchmark for improvement. In addition, Extreme Programming and the other agile methodologies rooted in lean management inherit from lean management the critical importance of measuring [8].

PSP requires the collection of detailed metrics of the development time, bugs discovered and corrected at all development stages, and software size. Then all collected data are analyzed through statistical methods. These results provide software engineers sets of historical data mainly used to:

1. make reliable estimates on variables such as time schedule, quality, etc. of on going projects
2. find out how to improve the development process identifying problems

This paper presents PROM (PRO Metrics) a tool designed to help both developers and managers to keep projects under control [2] [5]. To achieve this goal, the system provides two kinds of reports to users basing on their role inside the project. PROM is not designed to keep track of only software-based enterprises, it can be used to keep track of how people work in very different fields. The only prescription is the use of computer-based tools such as office automation tools.

PROM is tool for automated data acquisition and analysis that collect both code and process measures [6]. It collects and analyzes data at different levels of granularity: personal, workgroup and enterprise. PROM is component-based, Java based and uses open source technologies and standards communication protocols. In particular, it is based on the Package-Oriented Programming development technique [7]. As an example, the plug-ins approach allows developers to use mass-market tools integrated in the data acquisition process.

PROM components communicate using the SOAP protocol (except database access that uses JDBC). This protocol allows plug-ins to communicate with Plug-ins Servers and Plug-ins Servers with the PROM Server (Figure 1).

![Figure 1: PROM architecture overview](image)

**References**


