

# Monitoring the Development Process with Eclipse

Alberto Sillitti<sup>1</sup>, Andrea Janes<sup>1</sup>, Giancarlo Succi<sup>1</sup>, Tullio Vernazza<sup>2</sup>

<sup>1</sup>Free University of Bozen

{Alberto.Sillitti, Andrea.Janes, Giancarlo.Succi}@unibz.it

<sup>2</sup>DIST – Università di Genova

tullio@dist.unige.it

## Abstract

*Monitoring the development process is a complex task due to the amount of resources required (i.e. time and money), moreover the correctness of collected data is not guaranteed if the required tasks are performed manually. This paper describes an extension of the Eclipse IDE environment to allow developers to collect data regarding the development process without any effort.*

## 1. Introduction

The software development process includes several activities that are hard to trace manually due to the amount of information to collect.

Process measurement, as described in the Personal Software Process (PSP) [3], requires the collection of detailed data regarding the development time, bugs, and software size during all development stages. The results of the analysis of such data provide software engineers sets of historical data mainly used to: (1) make estimates of time schedule, quality, etc. of on going projects; (2) improve the development process through the identification of problems.

As instance, the PSP requires the collection of several data such as:

- Editing time: time spent performing a task such as coding, writing a document, etc.
- Tool name: the name of the tool used
- File name: the name of the file edited
- Class name: the name of the class edited (if possible)
- Project: the name of the project to which the document belongs
- Size: total size of the document
- Differential size: size of the document compared to the previous version
- Defects: defects identified in the source code

Collecting such data manually requires a considerable amount of effort. Moreover, the accuracy of this collection technique is often not sufficient due to errors during high stressing periods such as when a deadline is approaching [1] [4]. An automated tool to retrieve such information is the only way to collect data without any additional effort and with good reliability.

Eclipse is one of the most popular IDE for Java developers. The Open Source nature of this IDE and its high extensible architecture through simple plug-ins make this the best platform to implement an automated data collection system.

The paper is organized as follows: section 2 describes the problem and the set of information to collect; section 3 shows the implementation of the data collector; finally, section 4 draws the conclusions.

## 2. Data collection

The data collection architecture for software development in Eclipse is part of a wider range project called PROM (PRO Metrics) designed to collect data regarding software development. This set of tools is designed to collect both product and process metrics. Moreover, the system can help both developers to improve their performances and managers to keep projects under control.

The system is able to collect data from a number of tools including several development environments and office automation tools. In particular, the Eclipse plug-in collects the following set of data:

- Users logged in
- Project name
- File name
- Class name
- Timestamp of the file opening (*Date* field)
- In focus time (number of seconds, *Delta* field)
- Name of the application (*Source* field)

All collected data is stored into an XML [2] file and sent, through the SOAP [5] communication protocol, to the PROM server. The main fields of the XML file are the following:

```
<Prom>
  <Session>
    <Authentication>
      <User login="aUser"/>
    </Authentication>
    <Project>
      <Name>aProject</Name>
    </Project>
    <Data>
      <Item>
```

```

    <Key>File Name</Key>
    <Value>aFileName</Value>
  </Item>
  <Item>
    <Key>Date</Key>
    <Value>aTimestamp</Value>
  </Item>
  <Item>
    <Key>Delta</Key>
    <Value>Seconds_elapsed</Value>
  </Item>
  <Item>
    <Key>Source</Key>
    <Value>IDE_used</Value>
  </Item>
</Data>
</Session>
</Prom>

```

Software development activities are mostly computer-based but there are activities that are hard to trace automatically (i.e. browsing documentation) and some of them are performed without any computer aid (i.e. reading manuals). For this reason, PROM also supports manual data insertion through a web page. As previously said, manual data collection is an error-prone activity but if it is reduced to just a few items, error is small enough to make collected data useful.

### 3. Implementation

The implementation of the data collection architecture has three main objectives: extensibility, simplicity, and mobility. The first requirement includes the development of an open architecture able to support several tools through the development of specific plug-ins. The second requirement is essential to help third parties to develop plug-ins to improve the architecture. The last requirement provides support to mobile users that are used to develop using laptops and look at high-level reports through PDAs and other small devices.

Plug-ins are the standard way to extend most of advanced software tools such as the Eclipse IDE.

The PROM architecture is based on four main components: (1) Plug-ins; (2) Plug-ins Server; (3) PROM Server; (4) PROM Database.

Plug-ins are tool specific and their objective is the collection of process data from several environments such as development tools and office automation tools. Collected data is sent to the Plug-ins Server that manages all data. Plug-ins are implemented using different languages due to the supported languages in different environments. The Eclipse plug-in is written in Java.

The Plug-ins Server is a local data collection point that stores data acquired by each plug-in. This means that the machine used to collect development data can work

offline without losing important data. Collected data is stored in a local repository and sent to the PROM Server when a network connection is available. It is implemented as a Java application and uses the Apache Axis libraries to support the SOAP protocol.

The PROM Server collects data from all Plug-ins Servers through the SOAP protocol, performs data analysis and integration and stores results into the PROM Database. It hides the implementation details of the data model of the database and allows advanced clients to access row data stored in the database through a web service in order to allow web services-enabled applications to perform customized data analysis. The implementation is based on the Apache Tomcat application server and the Apache Axis libraries to support the SOAP protocol.

The PROM Database stores all data regarding authorized users, projects data, and collected metrics. The implementation is based on PostgreSQL, an open source DBMS.

### 4. Conclusions and acknowledgements

This paper presented an overview of a data collection architecture able to support several environments including IDEs and office automation tools.

The goal of the system is to provide support to both developers and managers helping them to improve their performances and keep projects under control. Moreover, the system can be used to evaluate the effectiveness of different development techniques and company policies verifying whether they produce real benefits or not.

The authors would like to thank the Italian Government for co-funding part of this research through the Funds for Basic Research (MAPS project, an Italian acronym for Agile Methodologies for Software Production).

### 6. References

- [1] A.M. Disney, P.M. Johnson. Investigating Data Quality Problems in the PSP. In Proc. Of the 6th International Symposium on the Foundations of Software Engineering (SIGSOFT'98), Orlando, FL, USA, November 1998.
- [2] C.F. Goldfarb, P. Prescod. The XML Handbook, 3rd edition. Prentice Hall Computer Books, 2000.
- [3] W. Humphrey. *A Discipline for Software Engineering*. Addison-Wesley, 1995.
- [4] P.M. Johnson, A.M. Disney. A critical analysis of PSP data quality: Results from a case study. *Journal of Empirical Software Engineering*, December 1999.
- [5] SOAP (Simple Object Access Protocol) – specifications: <http://www.w3.org/TR/SOAP/>