

Deploying, Updating, and Managing Tools for Collecting Software Metrics

Alberto Sillitti
Free University of Bozen
Piazza Domenicani 2
I-39100 Bolzano, Italy
asillitti@unibz.it

Barbara Russo
Free University of Bozen
Piazza Domenicani 2
I-39100 Bolzano, Italy
brusso@unibz.it

Paolo Zuliani
Free University of Bozen
Piazza Domenicani 2
I-39100 Bolzano, Italy
pzuliani@unibz.it

Giancarlo Succi
Free University of Bozen
Piazza Domenicani 2
I-39100 Bolzano, Italy
gsucci@unibz.it

ABSTRACT

Collecting software engineering data is difficult due to the number of problems that researchers face in this activity. One of the most relevant is the ability to install and keep up-to-date the measurement tools installed in the production machines in order to collect such data. Even when a few machines are involved, maintaining all the tools required to collect data from the different development tools requires a full-time system administrator. Moreover, since most of these tools are research tools, researchers are updating them very frequently to fix bugs or to add new features. In a production environment, especially when Agile Methodologies are in place, the time and effort. For these reasons, automating the management of the tools for metrics collection is a key factor to provide easy-to-use tools and allow development teams to collect data useful for both practitioners and researchers.

Categories and Subject Descriptors

D.2.7 [Software Engineering]: Distribution, Maintenance, and Enhancement

D.2.9 [Software Engineering]: Management

D.2.11 [Software Engineering]: Software Architectures

General Terms

Management, Measurement.

Keywords

Software Metrics, Process Monitoring

1. INTRODUCTION

Measures are valuable information in all engineering disciplines in order to evaluate the status of a project and to predict the quality of the final product.

Unfortunately, that does not apply to the software engineering

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Workshop QUTE-SWAP@ACM/SIGSOFT-FSE12, November 5, 2004, Newport Beach, CA, USA.

Copyright 2004 ACM 1-58113-000-0/00/0004...\$5.00.

area because the relationships among measures, processes, and products are not well known. Predictions are usually difficult, domain dependant, and affected by a relevant error [2] [4] [5]. Researchers in this area are facing a number of difficulties interfering with their main goal.

Most of these difficulties deal with technology evolution and changes of user requirements.

Technology evolution includes:

1. evolution of languages;
2. evolution of development tools;
3. evolution of the metrics collection tools.

Standard languages, such as C, are not likely to change but others, like Java and the different flavors of C++, are subject to change quite often. This means that language parsers extracting product metrics have to be updated accordingly.

In addition, development tools are evolving and, in particular, their libraries to support extensions (often called plug-ins or add-ins) are subject to major changes. For this reason, in order to support the new libraries and the new features, developers have to rewrite the plug-ins for collecting process metrics.

Besides, tools for metrics collection are research tools that are subject to frequent changes due to bug fixes and improvements generated by the research advancements in the area.

Main user requirements changes deal with:

1. data collection in a specific environment;
2. customized data reports.

Since data collection is implemented in very different organizations, they are using a wide set of development tools and languages. Moreover, local area networks can be configured in very different ways, increasing the complexity of the metrics transfer to a centralized server.

In addition, users are requesting different data analysis requiring the collection of different data sets. This means that data collection tools have to include enhancement in order to collect more and more data.

The metrics collection activity is important in software development for several reasons. Some of them are the following:

- creation of a knowledge base regarding the actual software process in order to make reliable estimates in future projects
- tracing the actual status of the project in order to identify possible quality and schedule issues

Metrics collection assumes a paramount role in some development methodologies such as in eXtreme Programming, in which timing and schedule are the basic concepts. In such cases, data collection is a key activity and automating this process produces important benefits for the development team [6] [3] [8].

This paper deals with the problem of deploying, updating, and managing a set of tools for metrics collection called PROM (PRO Metrics) [9] [10].

The paper is organized as follows: section 2 describes the basic architecture of PROM and its maintenance problems; section 3 presents an approach to address the problems identified in the previous section; finally, section 4 draws the conclusions and introduces future work.

2. PROM AND ITS MAINTENANCE

PROM (PRO Metrics) is a tool for product and process metrics collection. It includes several components, some of them are server-side others are client-side.

The architecture includes three set of components (Figure 1):

1. server-side components
2. client-side components
3. the data transfer tool

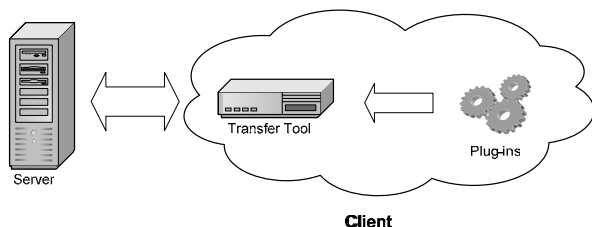


Figure 1: PROM components overview

The server-side components include a database that stores all collected data and an application server that exposes a web service for data collection and generates several web pages including user preferences, data management, and several kinds of data analysis (Figure 2).

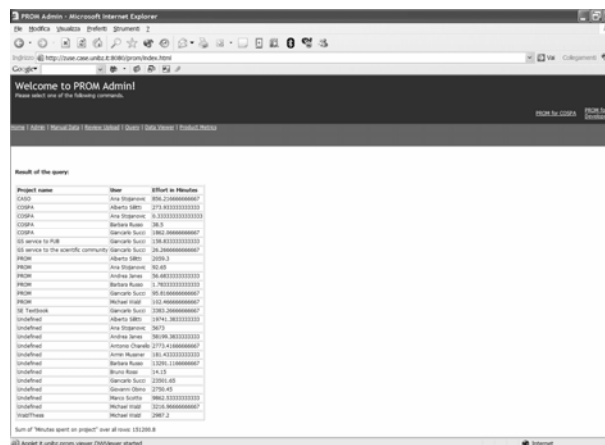


Figure 2: Effort summary

Since the server-side part of the architecture is installed only in a single machine and it is designed to automatically support the collection of new metrics, it generates only a limited number of management problems that can be considered regular maintenance.

The client-side components include all the metrics collection tools. These pieces of software interact with the operating systems (both Windows and Linux), the development tools (e.g., Eclipse, Visual Studio, JBuilder, etc.), and the office automation tools (both Microsoft Office and OpenOffice).

In order to collect data, all the development machines require these client-side components but, often, different developers use different software and even different versions of the same package. For these reasons, keeping up-to-date such machines is time expensive for both the system administrator and the development team that have to suspend its main activities.

The client-side components focus on the extraction of metrics data and store them using a common XML format but they do not deal with the transmission of such data to the central server. This is a design choice in order to keep the client-side components as simple as possible and centralize the transmission, authentication, and customization capabilities.

All plug-ins store collected data in the user profile as a set of XML files. The data transfer utility sends these files to the central server that collects all data and performs the analysis.

The data transfer tool is a special client-side component that manages data transfer (including network configuration), user authentication, and user preferences.

The client-side components and the data transfer tool require most of the maintenance effort, for this reason is required an automated system to detect the available updates and install them automatically.

3. PROM UPDATE MANAGER

PROM is designed to collect different sets of metrics without any human effort in order to promote the collection of reliable and accurate data. This goal clashes with the management of the system described in section 2.

The installation and maintenance of such a system is not a trivial task due to the number of different tools involved, which are dependant on the application actually installed on the development machines.

Hence, the management of the client-side components is the most time expensive activity in the PROM usage.

The PROM Update Manager is a tool able to support the installation procedure and the update of the client-side components of the system (Figure 3). In order to support different platforms, the update manager is written in Java and it is launched from a web page through the Java Web Start technology [7].

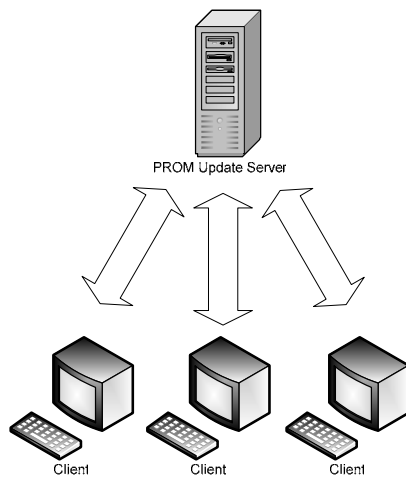


Figure 3: PROM Update overview

The PROM Update Manager includes two main features:

1. support for the first installation of the PROM client-side components (including the data transfer tool)
2. update automatically the components installed when new compatible versions are available

PROM Update Manager provides support to the installation process presenting a list of applications for which are available data extractors plug-ins. The user has to specify only which applications are installed and the automated procedure is able to handle all the installation process.

In some cases, such as a multiple installations of a tool, the installation procedure asks for further information from the user.

The installation task produces an XML file describing the actual configuration of the installed PROM tools, including name and version of the plug-ins. The updating procedure will base the updating process on such data.

The Update Manager regularly checks for available updates, it downloads the XML file describing the available plug-ins and compares them to the installed ones. If a new version of an installed plug-in is available, it removes the old one and installs the new version (Figure 4).

Moreover, some plug-ins are not stand-alone but they require some other installed tools. The update manager also checks these dependences and it installs and updates all the related components.

Besides, the PROM Update Manager itself is a client-side component and, in order to provide a really extensible systems, it can update itself.

An XML file stores information regarding the descriptions of the components and their dependences [1].

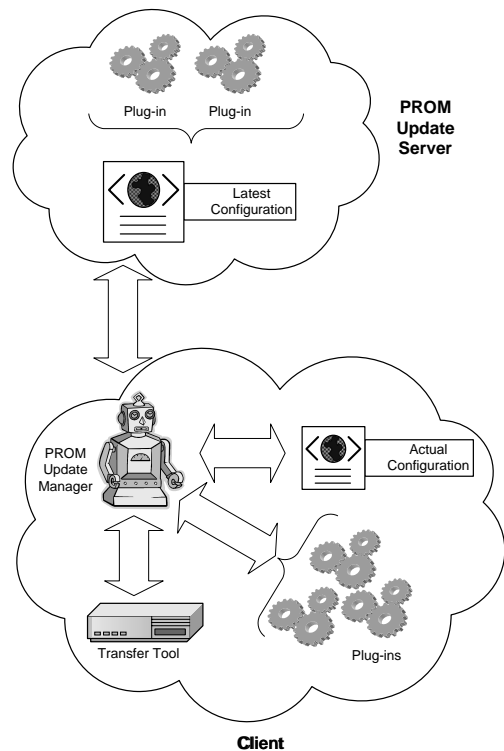


Figure 4: PROM Update Manager

There are two copies of this XML file (Figure 5): one in the server and one on the client.

```

<PROMUpdate>
  <Plugin>
    <Id>...</Id>
    <Name>...</Name>
    <Version>...</Version>
    <Filename>...</Filename>
    <Dependences>
      <Id>...</Id>
      ...
  
```

```
</Dependencies>
</Plugin>
...
<PROMUpdate>
```

Figure 5: PROM Update Manager configuration

The copy on the server describes the latest versions of all the available plug-ins and their dependences.

The copy on the client is a subset of the document on the server and describes the plug-ins that are actually installed on the client. For this reason, every client can have a different subset.

The Update Manager uses the client-side copy of the configuration file to identify to keep the client up-to-date.

4. CONCLUSIONS AND FUTURE WORK

PROM is designed to collect software metrics automatically without any human effort. Moreover, the system provides a mechanism to support the installation procedure and the update of all installed components.

Due to these side systems, PROM requires a limited maintenance effort and allows developers to focus on their development projects.

However, the system is not fully automated yet. At present, the updating system is only able to check the availability of new releases of the client-side components already installed and update them. However, the first installation of the system requires some human effort to specify which applications are installed in the target system and, consequently, which client-side components have to be installed.

The next step is the automation of the installation procedure through the automated identification of the applications installed in the target system and the related selection of the specific client-side components. This step is required in order to automate completely the setup of the client machines.

5. ACKNOWLEDGMENTS

This work was partially supported by MAPS (Agile Methodologies for Software Production) research project,

contract/grant sponsor: FIRB research fund of MIUR, contract/grant number: RBNE01JRK8.

6. REFERENCES

- [1] Monatti P.A., Damiani E., De Capitani di Vimercati S., Samarati P., "A Component-Based Architecture for Secure Data Publication", *17th Annual Computer Security Applications Conference*, New Orleans, Louisiana, USA, 11-14 December 2001.
- [2] Ceravolo P., Damiani E., Marchesi M., Pinna S., Zavatarelli F., "A Ontology-based Process Modelling for XP", *10th Asia-Pacific Software Engineering Conference*, Chiang Mai, Thailand, 10-12 December 2003.
- [3] Disney A.M., Johnson P.M., "Investigating Data Quality Problems in the PSP", *Sixth International Symposium on the Foundations of Software Engineering (SIGSOFT'98)*, Orlando, FL, USA, November 1998.
- [4] Fenton N.E., Pfleeger S.H., *Software Metrics: a Rigorous and Practical Approach*, Thomson Computer Press, 1994.
- [5] Gianini G., Damiani E., "An Ontology-Driven Approach to Metadata Design in the Mining of Software Process Events", *8th International Conference on Knowledge-Based Intelligent Information & Engineering Systems 2004*, Wellington, New Zealand, 22-24 September 2004.
- [6] Humphrey W., *A Discipline for Software Engineering*, Addison-Wesley, 1995.
- [7] Java Web Start Technology, available at: <http://java.sun.com/products/javawebstart/>.
- [8] Johnson P.M., Disney A.M., "A critical analysis of PSP data quality: Results from a case study", *Journal of Empirical Software Engineering*, December 1999.
- [9] Sillitti A., Janes A., Succi G., Vernazza T., "Collecting, Integrating and Analyzing Software Metrics and Personal Software Process Data", *EUROMICRO 2003*, Belek-Antalya, Turkey, 1 - 6 September 2003.
- [10] Sillitti A., Janes A., Succi G., Vernazza T., "Measures for Mobile Users: an Architecture", *Journal of System Architectures*, 50(7), July 2004.