

PROM: TAKING AN ECHOGRAPHY OF YOUR SOFTWARE PROCESS

Raimund Moser, Andrea Janes, Barbara Russo, Alberto Sillitti, Giancarlo Succi
Centre for Applied Software Engineering
Free University of Bolzano/Bozen
[rmoser, ajanes, brusso, asillitti, gsucci]@unibz.it

Measurement in software production is essential for understanding, controlling, and improving the software development process. Past research has emphasized the importance of a disciplined data collection process as a prerequisite for a sound, solid, and useful analysis.

Unfortunately, measurement programs often rely on manual data collection and therefore require a considerable effort. For this reason measurement programs are not widely adopted.

In order to reduce the needed effort and therefore rise the acceptance of measurement programs tool support is needed especially for the data collection process.

This paper describes ProM (ProMetrics), a set of tools to collect, integrate, visualize, and analyze the software development process with the aim to be as non-invasive as possible.

1. Introduction

The collection of measures is the first step in order to know how to control and improve the software development process [4]. Interesting aspects to measure are the performed activities, produced artifacts, and their properties.

Being able to measure the effort and the outcome of the performed activities will allow understanding where value is created or destroyed during the software development process. From a management's point of view, the gained insight can be used to understand the value delivered to stakeholders, to deliver what is really needed and to lower the risk of software projects [1].

Knowing cause and effect relationships will improve the alignment of inputs to outputs and thereby create a clear "line of sight" to desired results. This will help to increase the productivity of software production [1].

Past research has also shown the improvements in coding quality, defect prevention, and effort prediction accuracy when developers measure their personal development process and in this way better understand how they work [6].

In spite of the mentioned advantages, measurement in software production is not widely adopted [3]. One reason for this is because measurement programs often rely on manual data collection and therefore require a considerable effort (and is therefore costly) [3]. Additionally, the alignment of the measurement goals to a specific environment as well as the ongoing adjustment of the collected metrics to changed environments requires additional effort [3].

Manual data collection require a continuous "context switch" between product development and process recording is needed [7]. Frequent "context switches"

mean that the developer has to change often on what he is currently focusing his or her attention.

This is experienced as an annoying activity and can be the cause of errors because the engineer is not concentrated enough. It can be also “too intrusive for many users who desire long periods of uninterrupted focus for efficient and effective development” [7].

For the mentioned reasons, tool support is needed in order to automate the data collection process as much as possible [3]. The following sections present ProM (ProMetrics), a set of tools to measure and visualize the software development process.

2. Overview

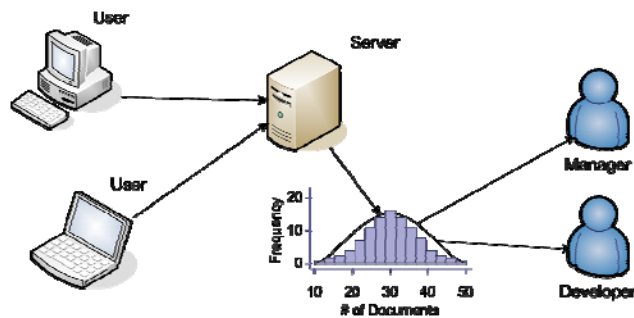


Figure 1. Architecture of ProM.

ProM (ProMetrics) consists of different tools (see figure 1) in order to obtain measurements from the software development process, to analyze the collected data, and to provide stakeholder specific views of the results.

2.1. Data collection

As mentioned in section 1, it is essential that the data collection process is as non-invasive (i.e. not requiring interaction with the developer) as possible. For this reason the data collection tools can be used without any intervention by the developer.

ProM aims to collect metrics about different aspects of the software development process:

- Code metrics analyzer: connects to a CVS repository and extracts metrics about the produced source code such as size metrics, complexity metrics and the object oriented metrics of the Chidamber and Kemerer metrics suite [2]. Currently Java, C, C++ are supported, C# is in a beta stadium, and web development languages such as PHP are planned.
- Plug-ins for software development environments such as Eclipse, Visual Studio, JBuilder: these tools collect data about the currently modified artifacts such as the effort spent for files, classes, and methods.
- Plug-ins for personal productivity applications such as OpenOffice, Microsoft Office, Microsoft Visio, etc. to collect the effort spent for documentation or planning purposes.
- Generic tools for the operating system: If no application specific plug-in is available, a generic tool for Microsoft Windows and one for Linux were

developed to collect generic data about the applications executed and the effort spent in using them.

In this way product and process metrics [5] about the software development will be collected:

Entity	Metric
Source Code	Size metrics, complexity metrics and the object oriented metrics of the Chidamber and Kemerer metrics suite, effort to develop the single entities, users involved in the development
Documents	Number of lines, effort to write the document, users involved to write the document, usage during the development process
Applications	Usage during the development process

Table 1. Product and process metrics collected with ProM.

The data collection tools store their measurements locally in form of xml files and when a connection is available to the server, send them to the central database using web services.

2.2. Data cleaning

In order to handle overlaps in the collected data, i.e. two plug-ins reporting the same information, every plug-in has an associated priority. The data cleaning module handles these conflicts by deleting data coming from lower priority plug-ins when data from higher priority plug-ins is available.

2.3. Data analysis

The collected data about the effort spent is used to determine the cost of different aspects of the software development process.

The effort spent to develop single methods, classes, or files is combined and associated to functionalities. This association can be done defining a regular expression rule or by manually assigning classes, methods, or files to a functionality.

This information can be used to:

- estimate similar tasks in the future: a better understanding of the overhead generated by the implementation of a certain functionality (e.g. the “undo function”) can help to estimate this overhead for future applications
- to determine the performance of the development team: to know how long the implementation of a certain functionality can help to understand strengths and weaknesses of the team, and strength and weaknesses of the used technology.
- to consider the value provided during software production: to know where the most effort is spent, compared to the perceived value for the customer helps to better align the development process to the requirements of the customer. As Boehm states, software development is currently done in a value-neutral way [1]. This means that tasks are carried out by developers in

a way that does not consider the value provided to the customer. Boehm suggests that the effort spent to develop single functionalities should be aligned to the perceived value for the customer. In this way we can perform a cost-benefit analysis of the performed actions.

- to understand the usage of different applications: by analyzing the usage of applications using association rule mining the importance of these applications (and combinations) can be better evaluated. This can be useful when the software development is analyzed in order to determine weaknesses in the used technology. When it is clear where time is spent, and a possible better technology is identified, it can be useful to know in which combinations or in what circumstances a certain technology is used. This can help to identify usage scenarios and to find a replacement that covers all needed usage patterns.
- to compare the effort of non-coding activities to their value provided: the effort spent for the creation of reports, user manuals, etc. It can be analyzed how much design documents are used or modified during the development process. This can help to identify artifacts that are not used at all, so their creation is useless, or not used enough, which could mean that certain rules or guidelines are not known or not followed.
- to understand the costs of missing agility: when changing customer requirements, changed demands of the market or changed regulations require that a produced application has to be changed, the “sunk costs” can be determined. This means that the effort spent to develop code afterwards deleted can be measured. This data can be used to better understand the causes of this and to understand if better technologies or development methods can be used (e.g. software development patterns) that are better suited in an agile environment.

These results of the analysis of the effort spent are combined with the changes observed in the collected product metrics.

In particular the metrics “Lines of Code”, “McCabes Cyclomatic Complexity” and “Coupling between Objects” [2] are used to understand the effects of changes in the code. If these changes are put in relation with the effort you are able to explore the causes of the generated effort and possible improvements by simplifying or by changing the structure of the code can be taken into consideration.

2.4. Data visualization

The visualization of the collected data is provided via a tool called ProM BAMI (Business Analysis Management Interface). This tool was implemented as a plug-in for the Eclipse Rich Client Platform [8].

ProM BAMI connects to the central database containing the collected events and builds an artifact tree containing all produced artifacts and their related properties.

Import rules following an entity-condition-action format can be defined in order to automatically assign certain properties to artifacts. An example rule could be:

<Rule>

```

<Description>Testing phase</Description>
<Project>*</Project>
<Entity>File</Entity>
<Condition>
  <Name>.*test.*</Name>
</Condition>
<Action>
  <SetProperty>
    <Name>phase</Name>
    <Value>test</Value>
  </SetProperty>
</Action>
</Rule>

```

This rule assigns to all filenames containing the word “test” in the filename a property “phase” with the value “test”. It will be possible to filter out this files and the related effort using a view and in this way know the testing effort.

Figure 2 shows ProM BAMI with the artifact tree on the left. In this tree the analyst can select the artifact he or she is interested in. After the selection, the stored properties of the selected artifact can be examined using different views. On the screenshot of figure 2 four views showing the progress of the metrics “lines of code” and “McCabe’s cyclomatic complexity” during the last weeks is shown.

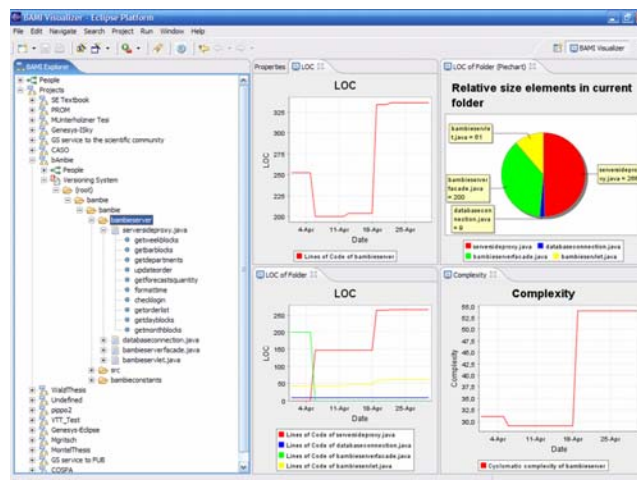


Figure 2. Screenshot of the ProM visualization tool.

The different views can be associated to specific projects and specific types of artifacts using an XML configuration file. In this way it is possible to adapt the different views on the collected data to specific environments.

In order to automatically provide feedback to developers or managers, it is possible to define reports and alerts that ProM BAMI sends using e-mail. The reports and alerts are configured in a similar way as the rules mentioned above using an entity-condition-action format. In this case “action” allows the use of an

<email> tag or the definition of a report following the Jasper Reports XML format¹

2.5. Extensibility

Eclipse provides the so called “Extension points” that allows plug-in developers to permit points in their plug-ins where others could add additional modules. ProM BAMI contains extension points that allow other developers to add new visualization modules or to add new modules for the data processing.

2.6. Expected benefits

The goal of ProM is to collect data about the software development process automatically. The developer does not need to participate in the data collection process. This should increase the acceptance of measurement in software production.

The feedback given to management and developers about the activities performed aims to help management to focus the software development on activities that provide value to the customer, and to help developers to improve their coding quality.

Specific aspects of the development process can be analyzed in respect to their cost. It will be possible to filter out the effort spent for a specific set of classes, methods or files. This can also improve the understanding of the connected cost when promising certain features to customers.

ProM BAMI was designed to allow an easy adaptation to specific requirements when used inside a software development team. Rules can be defined to identify specific aspects in the software development process, alerts and reports can be defined to provide feedback and different visualizations can be used to monitor and analyze the collected metrics.

3. Bibliography

- [1] B. Boehm, 2005, *Value-Based Software Engineering: Overview and Agenda*. Technical Report USC-CSE-2005-504, University of Southern California, Los Angeles, CA
- [2] S. Chidamber, C. Kemerer, 1994, *A metrics suite for object oriented design*. In IEEE Transactions on Software Engineering, 20, 6, 476-493
- [3] A.M. Disney, P.M. Johnson, 1998, *Investigating Data Quality Problems in the PSP*, Sixth International Symposium on the Foundations of Software Engineering (SIGSOFT'98), Orlando, FL
- [4] T. DeMarco, 1982, *Controlling Software Projects*. New York: Yourdon Press
- [5] N. Fenton, S. Pfleeger, 1997, *Software Metrics a Rigorous and Practical Approach*, PWS Publishing Company
- [6] W. S. Humphrey, 2000, *The Personal Software Process*. Technical Report, Software Engineering Institute, Carnegie Mellon
- [7] P.M. Johnson, H. Kou, J.M. Agustin, C. Chan, C.A. Moore, J. Miglani, S. Zhen, W.E. Doane, 2003, *Beyond the Personal Software Process: Metrics collection and analysis for the differently disciplined*, In Proceedings of the 2003 International Conference on Software Engineering
- [8] Object Technology International, Inc. *Eclipse Platform Technical Overview*. Eclipse White Paper, <http://www.eclipse.org/whitepapers/eclipse-overview.pdf>

¹ <http://jasperreports.sourceforge.net/>