Continuous Software Quality Supervision Using SourceInventory and Columbus

Tibor Bakota, Árpád Beszédes, Rudolf Ferenc and Tibor Gyimóthy
University of Szeged
Department of Software Engineering
{bakotat|beszedes|ferenc|gyimothy}@inf.u-szeged.hu

ABSTRACT

Several tools and methods for source code quality assurance based on static analysis finally reached a state when they are applicable in practice and recognized by the industry. However, most of these tools are used in an isolated manner and very rarely as organic parts of the quality assurance process. Furthermore, little or no help is provided in interpreting the outputs of these tools. This paper presents *SourceInventory*, a system for source code-based software quality assessment and monitoring, which is able to collect, store and present measurement data including metrics, coding problems and other kinds of data like bug numbers and test coverage information. It helps software developers, architects and managers to take control over their software's quality by performing continuous code scans, fault detection, coding style verification, architecture violation detection, and automatic report generation considering metric baselines.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement; D.2.8 [**Software Engineering**]: Metrics

General Terms

Measurement, Management, Reliability, Verification

Keywords

Software quality assessment, static verification, continuous measurement, software metrics, bug detection.

1. INTRODUCTION

In software quality assurance, the importance of static verification and quality assessment techniques (a.k.a. static testing) is increasingly recognized. To ensure sustainable quality in a continuously evolving software system, quality measurement is required to be a *continuous activity* during software development and evolution. The maintenance and evolution costs of software projects are significantly larger than the initial development, generally around 70% of the total cost. The biggest problem is, however, that these costs inevitably increase as the system gets older, so the main aim of *continuous software quality supervision* is to keep the long-term maintenance costs as low as possible.

Over the past decades, many methods and tools have been developed to assist static quality assessment (e.g. metrics calculators, coding and design problem detectors, and reverse engineering tools). Unfortunately, little or no help is generally provided in *interpreting* the outputs of these tools. Using the analogy of a human patient, the tools provide measurement results (laboratory findings), but no diagnosis nor treatment is suggested for the "ill" software, although this would be very much desired by important stakeholders like project managers. Finally, it is very uncommon yet that the results provided by these tools are *integrated* in a central quality repository, which can be easily accessed and used on a daily basis by the developers and managers.

SourceInventory is a step forward to overcome these weaknesses. The system is based on the *Columbus* technology[1], and it is able to collect, store and provide any kind of measurement data including metrics, coding problems and other kinds of data like bug numbers and test coverage information. This paper introduces only SourceInventory, however the complete Columbus technology includes a vast number of supporting tools as well, like source code analyzers and different utilities. It is used in a number of industrial projects in which large software systems consisting of many million lines of code are continuously assessed for quality based on source code analysis (e.g. at Nokia, evosoft, Erste Bank).

In projects where it has been applied the technology provided very useful service to the project teams by enabling continuous supervision of the project's quality in forms of time line diagrams and automatic notifications on problems, for instance. The users of the framework have different roles; developers benefit from concrete problem reports, while managers are able to investigate higher level quality attributes in different forms (helping diagnosis activity).

The Columbus technology and the SourceInventory system is owned by FrontEndART Software Ltd.[2], a spin-off company of the University of Szeged. The tools are available for non-commercial research projects and education for free.

ICSE'08, May 10–18, 2008, Leipzig, Germany.

[1] R. Ferenc, Á. Beszédes, M. Tarkiainen, and T. Gyimóthy. Columbus – Reverse Engineering Tool and Schema for C++. In *Proceedings of the 18th International Conference on Software Maintenance (ICSM 2002)*, pages 172–181. IEEE Computer Society, Oct. 2002.

[2] http://www.frontendart.com

2. FEATURES AND USAGE SCENARIOS

In this section we will show the basic features of the SourceInventory client user interface (see Figure 1), such as creating diagrams, performing queries, and making reports.

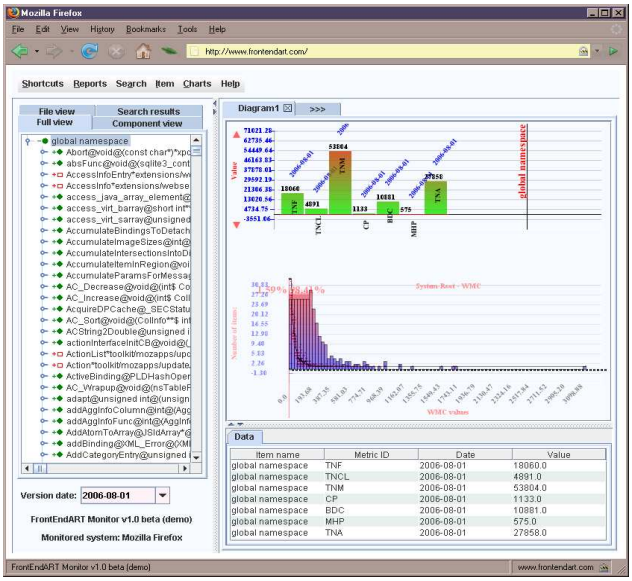

Figure 1: Graphical user interface

System Overview

The simplest diagram in SourceInventory is the bar chart. It is a great view to check basic properties of the monitored source code. The first (upper) diagram in Figure 1 shows basic size metrics for the source code of Mozilla Firefox: TNCL (Total Number of Classes), TNA (Total Number of Attributes), TNM (Total Number of Methods), TNF (Total Number of Functions); and the number of serious problems: BDC (Bugs and Dangerous Constructs), MHP (Memory Handling Problems) and CP (Complexity Problems).

Histogram – distribution of items

The second (lower) diagram in Figure 1 shows the distribution of the classes of Mozilla Firefox according to their WMC (Weighted Methods per Class) value, which is the sum of the McCabe cyclomatic complexities of the class' methods. The x axis represents the WMC complexity value and it is divided into equidistant intervals. The y axis shows the number of items (classes in this case) having a WMC value falling into the corresponding interval. Note, that the first few columns of the diagram are truncated for visualization purposes. It can be seen that most of the classes have small WMC values (the left-hand side columns are very high), but there are also classes with higher complexities. About 9.3% of the classes have complexity greater than 100 (which we take as the baseline) and there are also 41 cases (0.8%) when this value is greater than 500. One class is extremely complex with a WMC value of 3228, which is `nsHTMLEditor`. We found these numbers by performing searches by metric values using the Search Dialog Box of SourceInventory. Classes with high complexity can cause problems, as their testing and maintenance is very difficult. (Class `nsHTMLEditor` requires at least 3228 test cases!)

System Evolution

Continuous quality measurement and monitoring provides the same advantages as using version control and bug tracking systems. One can check and explore the quality-history of the system or some of its items. SourceInventory visualizes this information on so-called time lines. Figure 2 shows the changing of the number of FE (Feature Envy) and TF (Temporary Field) bad code smells in the source code.

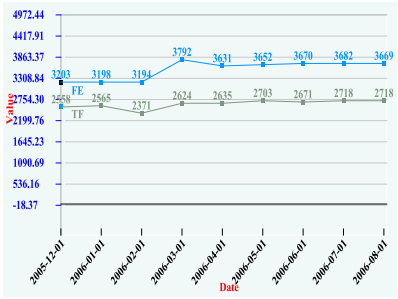

Figure 2: Time line – system evolution

Change report

Creating reports about the analyzed system is a powerful feature. One of the most important questions of software developers and architects is the following: "What did change?" SourceInventory answers this question by providing a change report (see Figure 3). The report lists the new, removed and changed items in the system according to a selected set of metrics (in the example these are the following: CBO – Coupling Between Object classes, LCOM5 – Lack of Cohesion of Methods and WMC – Weighted Methods per Class), and shows the change of their metrics. The green (negative) numbers show improvement, and the numbers shown in red (positive numbers) mean quality decrease.

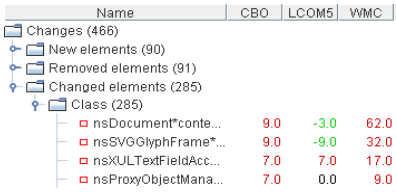

Figure 3: Changes in metrics – report

Problem report

The problem report (see Figure 4) is meant to provide useful information usually for programmers. The report shows all the coding errors, problems and bad smells in the source code which the *SourceAudit* tool of the Columbus framework can provide.

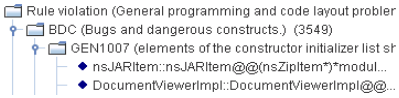

Figure 4: Problems in the code – report

Double clicking on a concrete problem line in the report fetches the source code from the version control system (CVS, Subversion or ClearCase), displays the code and highlights the problematic parts in red (see Figure 5).

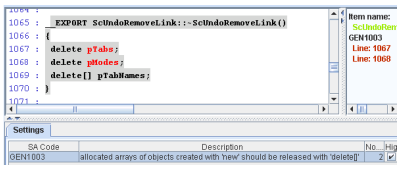

Figure 5: A highlighted problem in the code