Mangue: Metrics and Tools for Automatic Quality Evaluation of Source Code

Paulo R. M. Meirelles[1]**, Fabio kon**[1]

[1] Department of Computer Science
Institute of Mathematics and Statistics
University of São Paulo (IME-USP), Brazil

`{paulormm,kon}@ime.usp.br`

Abstract. *The overall goal of this work is to study software metrics and develop a tool for automatic quality evaluation of source code for free and open source software (FLOSS). This tool will be based on metrics that, combined and configured by specialists, will provide automatic analysis of features such as flexibility, clarity, modularity and software maintainability.*

Resumo. *O objetivo deste trabalho é estudar métricas de software e desenvolver uma ferramenta de avaliação automática da qualidade de código-fonte para projetos de software livre. Essa ferramenta terá como base métricas que, combinadas e configuradas por especialistas, permitirão a análisa automática de características como flexibilidade, clareza, modularidade e manutenibilidade do software.*

1. Problem Statement and Motivation

The use of Free, Libre and Open Source Software (FLOSS [1]) in the software industry has increased boldly in the last decade, influencing the global economy significantly [Benkler 2006]. Nevertheless, there are many companies and governments that are still reluctant in adopting FLOSS due to legal or juridical issues, commercial aspects or mistrust in the open source development process, products, and support.

Nowadays, companies face a plethora of open and proprietary software alternatives for each specific task or functionality. This requires well defined criteria to evaluate the quality of such products so that users can select the alternatives that best meet their requirements. Normally, this evaluation is performed by experimentation, reports from previous users, and documentation reading. However, open source software allows a more profound evaluation based on information obtained from source code analysis. For example, source code metrics give the opportunity to analyse bugs, flexibility, complexity, and readability of the code, which are important factors for the successful evolution of a software product [Henderson-Sellers 1996, Fenton and Pfleeger 1998, Sato et al. 2007]. But, collecting these data and computing this information is an arduous task that requires the help of automated tools.

There are many source code metrics that can help in the task of evaluating the quality of a piece of software. Source code metrics can show: (i) the complexity and

---

[1] In this paper, we will use the terms free software, open source software and FLOSS interchangebly.

dimension of programs; (ii) the internal and external dependencies and the effectiveness of encapsulations and abstractions; (iii) the test suites coverage and quality.

In addition to this source code analysis, in most free software projects, one can obtain additional relevant information by accessing its version control system, list of participants and developers, and database of bug reports and corrections [Gousios et al. 2007]. From these data, it is possible to obtain information about the team performance, how rapidly the reported errors are corrected by the community, and others important factors.

Currently, several good open source tools collect such data in an automated way. Thus, it is feasible to create a tool to examine the source code of hundreds or thousands of free software projects and store the results in a single location, serving as a basis for analyzing software quality. Tools can generate tables, graphs, and rankings, enabling comparative, objective analysis among multiple options.

In this context, the conventional view of software quality evaluation which emphasizes the analysis of documentation does not seem to be appropriate for the FLOSS community. Because, the most valued asset for the open source community is the code (that is what our research will focus on).

Therefore, our goal is to develop a new or adapt an existing tool for automatic quality evaluation of open software projects, based on the metrics selected and defined in this research. Recently, a multi-million-dollar market has been created based on products and services developed using free software. To further promote the growth of this new phenomenon in the software industry, the FLOSS community must develop metrics, tools, and methodologies for evaluating and comparing FLOSS products.

Although there are hundreds of companies and governments interested in using free software at their infrastructure for information technology, they can fail because of the number of available options or the lack of objective criteria to evaluate reliability. Thus, some questions are still open:

- How is it possible to measure the quality of software products to help choosing one of them?
- What features are relevant for evaluating free software?
- What are the quality metrics?
- What are the appropriate metrics for evaluating source code?
- How to compare similar open source projects?
- How to automate the quality evaluation and to what extent it can be automated?
- How to evaluate the effectiveness of the metrics and of the automated quality evaluation?

2. Background and Related Works

FLOSS was born as a movement supported by volunteer developers and foundations. Nowadays, software companies are investing in FLOSS, large firms are financing FLOSS projects, and an increasing number of companies adapted to FLOSS business model [Wasserman and Capra 2007, Riehle 2007]. The impact of FLOSS in the software industry motivated Forrester Consulting to evaluate the upcoming paradigm change in software companies, due to open source software adoption [Forrester-Consulting 2008]. This analysis was conducted exclusively on enterprises that were already using FLOSS at

a minimal level, which represent around 15% to 24% of enterprises in Europe and North America today. In the same manner, the Qualipso project supported by the European Commission has as one of its major goals to understand the reasons and motivations that lead to the adoption of FLOSS by software companies. Qualipso also aims to understand which specific trust factors are taken into account when selecting a FLOSS product [del Bianco et al. 2008] .

According to the Forrester study, nowadays, the usage of FLOSS in the back-end and middleware software categories is widely spread, while the adoption of open source office productivity tools and business applications (front-end) is growing. FLOSS is now strongly used to develop mission-critical applications, services, and products [Forrester-Consulting 2008]. Several industrial sectors are adopting FLOSS at different rates, with various focuses [Forrester-Consulting 2008]. In Forest's survey, 92% of the respondents said that they have met their quality expectations and, in some occasions, even exceeded them. The satisfaction regarding cost was on similar level at 87%. Cost is the primary adoption driver [Forrester-Consulting 2008], but other factors are important: 56% of all enterprise users name reduction in overall software cost as the primary motivation to use open source, but the motivation goes is far beyond just the price factor. Independence, not to be locked to a single vendor, is very important for 43% of all users [Forrester-Consulting 2008].

To help foster the continuing adoption of FLOSS by the industry, this thesis will focus on the study and development of metrics for the assessment of open source software quality. Most of the initial work in software metrics dealt with the characteristics of source code [Mills 1988]. The software metrics literature describes a large variety of metrics, such as:

- Size metrics: lines of code, system bang and functions points;
- Complexity metrics: cyclomatic complexity, information flow and knots;
- Halstead's Product metrics: program vocabulary, length and volume;
- Quality metrics: maintainability, defect and reliability metrics.

However, these metrics are not very satisfying [Mills 1988] as pointed out by a SEI report: "In the past, most metrics have been defined by an individual and then tested and used only in a very limited environment (...). Currently, useful metrics and models cannot be pulled off the shelf and used indiscriminately, careful application of available metrics and models can yield to useful results if they are tuned to a particular environment (...)" [Mills 1988].

Some programming languages require a specific set of metrics, like object oriented languages [Xenos et al. 2000]. A practical approach of applying these metrics, including ways to achieve automation of the data, is presented by Lanza and Marinescu [Lanza and Marinescu 2006].

Outsourcing also grew recently in the software industry, both in terms of hiring maintenance services and in adopting software developed by third parties. An important challenge in this context is how to quickly evaluate and understand the systems developed under outsourcing contracts, as discussed by Polo et al. [Polo et al. 2001].

In the last decade, the ability to use and adopt programs from others and the existence of several alternative software for a particular context have expanded with the in-

creasing availability of free software projects and open source repositories. Consequently, the challenge of evaluating from others became more complex due the large number of potential suppliers. Recently, one of the most acessed FLOSS repositories (Source Forge [SourceForge 2008]) presented in their statistics the number of 167,643 projects and 1,776,733 registered users and developers.

If we can build such automated tool for software quality evaluation and demonstrate that the information about FLOSS projects resulting from its analysis is reliable and easy to read and interpret. So, we will be able to provide a trustworthy process for evaluating and comparing the quality of open source products. Companies will feel more comfortable when adopting open source software and developers will be motivated to improve the quality of their work. These are the goals of a new series of research projects currently under way in connection with some ideas presented in this proposal:

- FLOSSMetrics (Free/Libre Open Source Software Metrics) is a project that uses existing methodologies and tools to provide a large database with information and metrics about FLOSS development. These information and metrics are coming from thousands of open source software. FLOSSMetrics also provides an open platform for evaluation and industrial exploitation of those results [FLOSSMetrics 2008].
- Ohloh is a website that provides a web services suite and an on-line community platform that aims to build an overview of open source software development. It defines metrics and analyses thousands of FLOSS projects. It is also an open source repository that everyone can contribute with [Ohloh 2008].
- Qualoss (Quality in Open Source Software) is a method developed in order to automate the quality measurement of FLOSS, using tools to analyze the source code and the project's repository information [Qualoss 2008].
- SQO-OSS (Software Quality Assessment of Open Source Software) is a consortium that aims to develop a suite of software quality evaluation tools. These tools will allow the objective analysis and benchmarking of FLOSS, what might improve source code's quality and provide scientific proof of its quality [SQO-OSS 2008].
- QSOS (Qualification and Selection of Open Source software) is a method composed of dependent and iterative steps: used reference definition; software evaluation; qualification of specific users' context; selection and comparison of software [QSOS 2008].
- FOSSology (Advancing open source analysis and development) is a project that aims to create a community to facilitate the study of FLOSS by providing a free data analysis tool [FOSSology 2008]. This project only gives information about the software license.

Our research differs from other works in the field mainly in the proposal of a tool that enables the combination of a set of metrics and presents the results of software quality evaluation in a language and format that is easily understandable. In addition, the tool lets expert developers to configure ranges of values for each metric, associating them with different quality evaluations. Thus, the measurements for a specific piece of software will be acceptable or not depending on the recommended values for the metrics set up by the user.

3. Research Method

The Mangue project will define a set of 10 to 20 metrics, classified according to their complexity, based on both non-functional aspects and analysis of the source code. Through controlled experiments and statistical analysis, each metric will be evaluated as to whether it shows significant correlations to the quality of free software as perceived by experts, thus allowing us to test whether that they can actually be used as a reliable evidence for determining quality.

To define this set of metrics, initially we collected the opinions of a set of Brazilian experts in software development, from industry, academy, and free software community. A questionnaire was developed based on a survey of the European software industry, conducted within the Qualipso project. The European survey was adapted and extended based on consultations with researchers in software development from our research group.

The questionary was sent by e-mail to a total of 80 software development experts and 38 of them answered it. The respondents have the following profiles: 8 managers, 13 FLOSS project leaders, 22 professionals with up to 10 years of professional experience in software development, and 16 professionals with over 10 years of professional experience in software development. The questionnaire, in Portuguese, was available on the Web[2] containing 123 questions: 9 personal, 28 professional, and 106 about the metrics and aspects of free software adoption and quality. The respondents were asked to give a number between 1 and 10 for each metric, 1 meaning irrelevant and 10 meaning essential.

The other goal of study is to analyze and make experiments with existing tools that evaluate automatically the source code and the activity of open source projects. The last step is the development or adaptation of a tool for automatic evaluation of the quality of FLOSS projects, including the metrics selected and defined in this study. Finally, we will test the developed tool with case studies, comparing the automatic results with the information provided by experts.

3.1. Partial Results

Aiming to achieve a comprehensive understanding of what software development experts consider to be important in assessing the quality of a software product, we selected 106 metrics and aspects that can influence the quality of open source software, impacting its adoption.

One of the questionnaire sections focused on the relevance of source code metrics, including 24 different metrics: (A) Total number of lines of code; (B) Number of classes (or files or modules); (C) Number of methods or functions; (D) Number of lines of test; (E) Number of tests by lines of code; (F) Number of tests by method or function; (G) Test coverage; (H) Level of coupling (dependency between the modules or classes); (I) Level of cohesion between modules or classes; (J) Cyclomatic complexity; (K) Existence of duplicate code; (L) Separation into modules; (M) Used data structures; (N) Uniform indentation style; (O) Number of columns in each line; (P) Number of lines in each method; (Q) Number of lines in each class; (R) Number of method in each class; (S) Number of attributes per class; (T) Number of local variables per method; (U) Patterns of nomenclature used uniformly; (V) Use of good names for classes, methods, variables; (X) Existence of comments in the code (the code coverage and distribution); (Z) Existence of modules for configuration (without hard-coded configurations and limits throughout of the code).

---

[2] http://ccsl.ime.usp.br/mangue

Source code metrics are usually objective and thus their measurement can be easily automated. They measure quantitative aspects of the software implementation and the relationships between the software modules and components.

In Figure 1 shows a graph of the average versus standard deviation for the importance given to all source code metrics (according to the opinion of the interviewees). The first quadrant contains the metrics considered by the interviewees as the least important and with similar responses.. In the second qua-
drant (low average and high standard deviation) are metrics with low average, but there was variability in the responses obtained (divergent opinions). In the third quadrant (high average and low standard deviation) are the metrics with high scores and with no great discrepancy between the answers (similar opinions). Finally in the fourth quadrant (high average and high standard deviation) are the metrics with high average but considerable variability (divergent opinions).
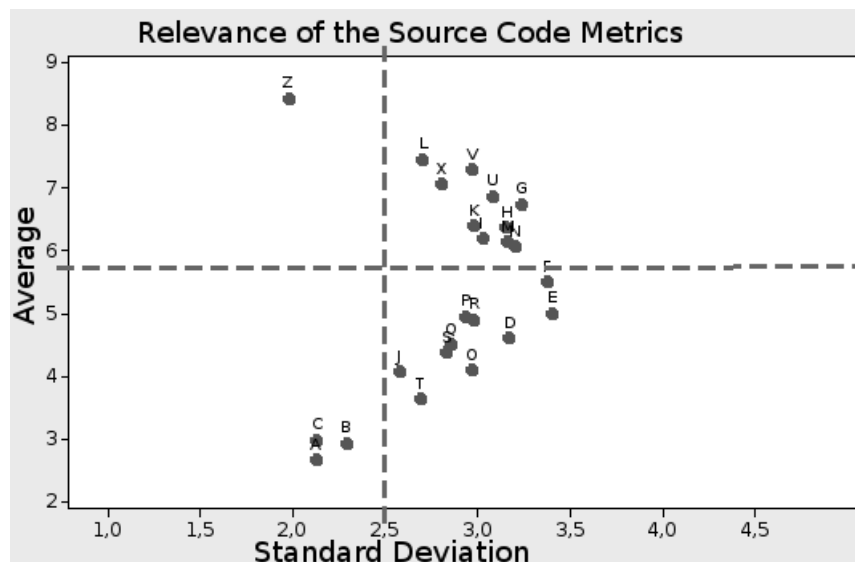

Figure 1. Relevance of the Source Code Metrics

We compared the results using the Mann-Whitney test [Conover 1980], since the scores do not follow a normal distribution. Thus, we could verify that the median of the items differs from each other. We use the Bonferroni correction [Bussab and Morettin 2003] to achieve a confidence level of 95%. In a statistical analysis for items-source two medians should be considered different if the p-value of Mann-Whitney test showed less than 0.00018. We compared the source code metrics between each others, resulting in 276 comparisons, 61 of them were relevant or 21.1% of the total.

Our study will focus on how to obtain automatically information about 12 source code metrics that were considered important by the interviewees: number of tests per method or function, test coverage, level of coupling, level of cohesion, existence of duplicated code, separations into modules, data structure used, uniform style of indentation, uniform naming schemes, use of good names for classes, methods and variables, existence of comments in the code (coverage and distribution), existence of configuration modules (without hard-coded configurations and limits throughout of the code). Some of these metrics can be easily automated (e.g., test coverage and level of coupling), but some of

them will require further study and we might even come to the conclusion that they cannot be automated (e.g., separation into modules and data structures used).

The study also showed that the experts consider important the presence of automated tests and take into consideration aspects of the project community, which can be obtained from FLOSS project repositories. The study showed that, according to the oppinion of the software development experts interviewed, having information about the source code and about subjective aspects related to tests and the project community is important for the adoption of an open source software product. To combine with source code metrics based on automatic analysis is another challenge of this research.

4. Expected Results

This doctoral research proposes the development of an automatic tool for evaluating the quality of open source software projects. This tool will gather data from hundreds or thousands of open source projects by visiting their source-code repositories and forges collecting multiple kinds of data and generating quality assessments automatically. These evaluations will help companies to decide whether to use or not an open source solution and which one to choose. They will also help promote quality enhancements by FLOSS developers and raise the trustworthiness of FLOSS products and processes. To accomplish this goal, software metrics must be selected, proposed, and evaluated.

In summary, the expected results are:

- Knowledge in objective metrics for evaluating the quality of software based on the source code.
- An automated free software tool for quality evaluation of FLOSS.
- Technical reports in an accessible language to IT professionals for the use of the knowledge generated by this research in the evaluation of free software to be used in their companies or government agencies.

5. Acknowledgement

References

Benkler, Y. (2006). *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. Yale University Press.

Bussab, W. A. and Morettin, P. A. (2003). *Estatística Básica*. Editora Saraiva.

Conover, W. (1980). *Practical Nonparametric Statistics*. Paperback.

del Bianco, V., Lavazza, L., Morasca, S., and Taib, D. (2008). Identification of factors that influence the trustworthiness of software products and artefacts - working document wd 5.3.2, version 2.0. Qualipso report, Qualipso.

Fenton, N. E. and Pfleeger, S. L. (1998). *Software Metrics: A Rigorous and Practical Approach*. Course Technology, 2 edition edition.

FLOSSMetrics (2008). Flossmetrics - free/libre open source software metric. http://www.flossmetrics.org/.

Forrester-Consulting (2008). Open source paves the way for the next generation of enterprise it: A european survey finds that open source becomes the hidden backbone of the software industry, and leads to a paradigm change in enterprise it. Technical report, Forrester Research.

FOSSology (2008). Fossology - advancing open source analysis and development. http://www.fossology.org.

Gousios, G., Karakoidas, V., Stroggylos, K., Louridas, P., Vlachos, V., and Spinellis, D. (2007). Software quality assesment of open source software. In *Proceedings of the 11th Panhellenic Conference on Informatics*, May.

Henderson-Sellers, B. (1996). *Object-Oriented Complexity*. Prentice-Hall.

Lanza, M. and Marinescu, R. (2006). *Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems*. Springer.

Mills, E. E. (1988). Software metrics. Technical report, Software Engineering Institute/SEI - Carnegie Mellon University, Software Engineering Institute/SEI - Carnegie Mellon University.

Ohloh (2008). Ohloh - the open source network. http://www.ohloh.net.

Polo, M., Piattini, M., and Ruiz, F. (2001). Using code metrics to predict maintenance of legacy programs: A case study. In *IEEE International Conference on Software Maintenance*, page 202.

QSOS (2008). Qsos - qualification and selection of open source software. http://www.qsos.org.

Qualoss (2008). Qualoss - quality in open source software. http://www.qualoss.org.

Riehle, D. (2007). The economic motivation of open source software: Stakeholder perspectives. *IEEE Computer*, 40(4):25–32.

Sato, D., Goldman, A., and Kon, F. (2007). Danilo sato, alfredo goldman, and fabio kon. tracking the evolution of object oriented quality metrics. In *Proceedings of the 8th International Conference on Extreme Programming and Agile Processes in Software Engineering (XP 2007)*, pages 84–92.

SourceForge (2008). Sourceforge.net. http://sourceforge.net.

SQO-OSS (2008). Sqo-oss: Software quality assessment of open source software. http://www.sqo-oss.eu/.

Wasserman, A. and Capra, E. (2007). Evaluating software engineering processes in commercial and community open source projects. In *Workshop Emerging Trends in FLOSS Research and Development*.

Xenos, M., Stavrinoudis, D., Zikouli, K., and Christodoulakis, D. (2000). Object-oriented metrics - a survey. In *Proceedings of the FESMA 2000 - Federation of European Software Measurement Associations*, volume 6, Madrid, Spain.