

# Towards the Analysis of Evolution OSS Ecosystems

Mathieu Goeminne<sup>a,\*</sup>, Tom Mens<sup>a</sup>

<sup>a</sup>Université de Mons – UMONS, Place du Parc 20, 7000 Mons

---

## Abstract

Interactions between user and developer communities on the one hand, and *open-source software (OSS)* evolution and quality on the other hand, are not intensively studied. However, these communities significantly influence how the software evolves. Empirical studies about this influence could offer us a way to propose changes in the software development process in order to improve the overall software quality. We propose to study the effect that user and developer communities have on the evolution and the quality of OSS, thereby extending previous studies of OSS evolution with knowledge about the ecosystem that surrounds it. We outline our current research that consists in an empirical analysis of the evolution of three well-known OSS, to try to find a relation between their quality and development popularity.

*Keywords:* software evolution, open source software, ecosystem, software quality, software developer

---

## 1. Introduction

The goal of this short paper is to present a new research topic that we started to explore since September 2009 at the Software Engineering Lab of the University of Mons. This research falls within the domain of *open-source software (OSS)* on the one hand, and *software evolution* on the other hand. The goal is to improve the state-of-the-art in research on how open source software evolves, with a particular focus on *software quality*. The traditional approach to this problem relies on the use of *software metrics* that provide information about internal attributes of the source code [1, 2]. The novelty of our approach is to complement this kind of information with knowledge about the OSS *ecosystem* (consisting of developers and users) that surrounds it. Firstly, we are interested in the effect that a community of communicating and collaborating software developers has on the (evolution of) software quality and vice versa. Some research focuses on this topic [3]. Secondly, we would like to study the relation between the *popularity* of an OSS system (in terms of number of users, as well as in terms of number of developers contributing to it) and its quality [4].

To this extent, we aim to empirically analyse and visualise the evolution of OSS communities, and to correlate this to the evolution of OSS quality characteristics. The insights gained from these empirical studies will ultimately allow us to reach the following goals:

- better support the software development process, by providing concrete suggestions and recommendations on how to improve interdeveloper communication and collaboration;
- improve the software quality, thereby making the OSS system more attractive to its developers and users, by taking into account not only internal characteristics of the source code, but also information of its surrounding ecosystem;
- provide reliable information about the quality of the OSS and its development process, in order to allow prospective users to make a more informed choice on whether or not to use the OSS system;

---

\*Corresponding author

Email addresses: mathieu.goeminne@umons.ac.be (Mathieu Goeminne), tom.mens@umons.ac.be (Tom Mens)

- give information to software developers about how their OSS is used, so that they can take this into account in future versions of the system.

## 2. Research Approach

The empirical analysis that we are carrying out is based on the Goal-Question-Metric (GQM) paradigm [5]. This approach is composed of three steps to solve a particular problem. First, a particular goal is specified. Examples of such goals can be found in the enumeration of the previous section. Secondly, precise questions are formulated that allow us to assess whether the goal has been achieved. Finally, the third step consists in selecting and applying software metrics that can be used to answer each research question.

In the context of the empirical analysis of the evolution of OSS systems and their surrounding ecosystem, some specific questions we would like to answer are the following:

- Is there a relation between software quality and development popularity (in terms of actual contributing developers, retention rate of developers, and intake of new developers)?
- Is there a relation between software quality, user satisfaction and usage popularity (in terms of number of actual users, retention rate of existing users, and intake of new users)?
- Does the way in which the developer team is structured have an influence on the software quality?
- Does the way in which the developer team communicates have an influence on the software quality?
- Does the way in which users and developers communicate with each other have an influence on the software quality?
- How is software quality related to particular development activities carried out by software developers (e.g., bug fixing, software refactoring, adding new functionality, ...)?
- Can we identify clearly distinct phases of development activity, software quality and developer communication throughout the software evolution process?
- Can we identify recurring or emerging patterns, antipatterns and trends of communication, collaboration and organisation (among developers, between developers and users)?
- How can we provide more precise information to users to allow them to make better use of their OSS and to select the most appropriate OSS to use for a give purpose?
- Which parts of the OSS are of sufficient quality and sufficiently modular to be factored out into a separate component or module that have the potential of being reused by other OSS systems?

To answer these questions, we need to analyse data and compute metrics extracted from a wide variety of available data sources. To extract commonly used software metrics [6], the project's *version repository* is sufficient. It can be used to extract the source code of each software version. To analyse interaction and communication behaviour among and between developers and users, we can extract information from developer and user *mailing lists*. In the same vein, *bug tracking systems* are an important source of data to mine information about users and developer activities. They also provide a means to determine the user satisfaction, through the error reports and feature requests suggested by users. To get an even more complete picture, other data sources may be added, depending on the specificities of studied software systems and the precise question that needs to be answered.

In most cases, all of the above data sources are weakly connected, and a significant effort must be made to integrate them in a coherent way. For example, user accounts used in some data source are not necessarily linked to accounts used by the same person in other data sources.

### 3. Current Research

We have started to carry out the empirical analysis of open source software systems. Following the GQM approach, the first question we would like to answer is the relation between the software quality and the development popularity. During a first iteration, we only rely on data extracted from source code version repositories. For each version we extract, we can estimate development popularity as the average activity period of developers. We also compute how many developers have stopped contributing (developer *turnover*), as well as how many new developers started to contribute (developer *intake*).

With respect to software quality, we restrict ourselves to object-oriented software, and use the traditional approach of computing a wide range of different object-oriented software metrics [7, 8], such as size metrics (e.g., lines of code, number of classes), complexity metrics (e.g., McCabe’s cyclomatic complexity), coupling and cohesion metrics (e.g., lack of cohesion in methods, fan-in and fan-out). We store all extracted metrics in a database that is queried to visualise and analyse evolution trends and correlations between software quality and development popularity.

In our first experiment, we chose to restrict ourselves to systems developed in the same object-oriented programming language, namely *Java*. For ease of comparison, we decided to analyse different OSS systems belonging to the same application domain, namely software development tools. In this domain, we choose three representative and well-known systems: Eclipse <sup>1</sup>, NetBeans <sup>2</sup> and ArgoUML <sup>3</sup>. One of the reasons for choosing these systems is because they have already been the subject of earlier empirical research in software evolution [9, 10, 11, 12]. In addition, all these systems have been evolving over a relatively long period (between eight and nine years), and have a relatively large size.

Metrics will be computed and handled by different tools. One of them is JHawk [13], a commercial *Java* application for extracting source code metrics from *Java* files. It will be completed by SLOCCCount [14], an open source tool to compute the number of lines of code in a source file. We will also use our own scripts and applications to consolidate data and to process it. Finally, data will be statistically analysed and graphically displayed with *R*, a language and environment for statistical computing and visualisation [15].

This research is the first step of a larger project, where other information will be added from the various data sources mentioned above, in order to be able to reach our goals by answering the research questions. At the end of our current research step, we hope to find a statistical correlation between the between the development popularity and the software quality. We also hope to provide recommendations to developer teams to help them to improve their software quality.

In addition to software metrics used to assess the source code quality, we need to establish additional metrics to measure user and developer communities and their interaction. Given the wide variety of data sources we have at our disposal, we can already go a long way in providing such metrics. For example, based on mailing list information, we can estimate user and developer communication in terms of the number of messages posted by a person on the mailing list(s). We can also measure interaction (communication) between a given set of persons (whether it be users or developers) by counting the number of subjects in which these persons are involved, and by counting the number of messages that are exchanged between these persons. As a second example, the use of a bug tracking data is another good opportunity to assess software quality, popularity and user satisfaction, by counting the number of issues reported by users, which is an indication of the software’s popularity and quality. Another potentially useful metric is the time needed by a developer to take care of a given reported issue. For all of these proposed metrics, a practical validation of their appropriateness is part of future work. In addition, we will have to come up with additional metrics to cover the list of research questions mentioned in section 2.

### 4. Related work

Research in software evolution started with Lehman’s, now famous, laws of software evolution [16, 17]. In the beginning of this century, researchers started to explore the evolution of OSS [18]. Thanks to the abundance of data available on OSS evolution, this research field exploded.

---

<sup>1</sup> [www.eclipse.org](http://www.eclipse.org)

<sup>2</sup> [www.netbeans.org](http://www.netbeans.org)

<sup>3</sup> [argouml.tigris.org](http://argouml.tigris.org)

Several European research projects have started to extract and analyse data related to the evolution and quality of OSS projects. The European *FLOSSQuality*<sup>4</sup> initiative involves three European research projects focusing on different aspects of the problem. *FLOSSMetrics*<sup>5</sup> is a project whose goal is to construct and analyse a huge database containing metrics from thousands of OSS. The *QualOSS*<sup>6</sup> project specialises in software quality analysis. It provides an objective method to assess the robustness and evolvability of OSS, and offers a tool to put this method into practice on OSS projects. *SQO-OSS*<sup>7</sup> is another project designed to analyse OSS quality. Its goals are to deliver a tool that provides advice to developers to increase their software quality.

It is clear that our research can built further on the results produced by these three projects. For example, we can directly exploit the huge database of OSS metrics, the data formats used, as well as some of the tools developed during these projects. The notion of developer and user communities, communication and collaboration, however, is not very explicit in these projects, so this is something where more work will be needed from our side.

Research on the relation between software and its community of developers is starting to emerge. For example, [3] studied the relation between the evolution of developer communication and the number of bug introducing changes in the source code. [19] provided a prototype tool for visualising and studying the OSS developer communities. Research on evolutionary patterns and their impact on software quality shows a correlation between the evolution of a developer team's structural organisation and the software quality [20]. Some empirical studies take into account social aspects in OSS development to characterise differences in OSS project management, as opposed to proprietary software management [21].

## 5. Threats to Validity

Besides the traditional threats to validity one encounters during studies on the evolution of OSS [22], some specific threats to validity are relevant to our empirical study. The first one is the reliability of the data that has been extracted by third parties (such as FLOSSMetrics). The second one is the possibility of inaccuracies in the tools we used to derive metrics (JHawk, SLOccount) and to perform the statistical analysis (R). Our hand-made tools could also be defectuous, as they are made specifically for this research and have not been intensively tested. A means to verify that these potential defects do not affect our results would be to replicate data extraction and analysis using different sets of tools, but this would require a significant amount of time and effort.

Some data compatibility problems can occur because of the fact that the information is extracted from different types of data sources. One example is that the same person can use different user accounts and email addresses for different repositories. Another typical problem is the different notion of commit in CVS and SVN (two very popular version control systems). In SVN, one can *push* many changes (adding, removing or editing a file) to the repository in a single commit. With CVS, commits are defined at the file-level only. To address such incompatibilities, it is necessary to have a more abstract view on the used data sources, to abstract away from their differences and focus only on their commonalities [19].

Another threat to validity is the need to consider replicability of the extraction and analysis. We will only use public repositories that can be accessed by everyone. All tools we use are either free or obtainable at an affordable price; our own tools will be made publicly available. A more difficult threat is the generalisability of interpretations. Our initial results will have been extracted for a particular programming language (*Java*), using a particular set of metrics, and using a limited set of subject systems within the same application domain. All of these may bias the results of our work, and further work will be needed to generalise our research results in order to extend their validity to different programming languages, different programming paradigms, different metric sets, different application domains, and so on. We therefore plan to extend the scope of our study to other *Java* OSS in different application domains.

The tools we develop will allow us to assess new OSS using analogies. We intend to use datamining tools to classify the projects and advise developers some changes based on knowledge gained by previous studies. Organisa-

---

<sup>4</sup> [www.flossquality.eu](http://www.flossquality.eu)

<sup>5</sup> [www.flossmetrics.org](http://www.flossmetrics.org)

<sup>6</sup> [www.qualoss.org](http://www.qualoss.org)

<sup>7</sup> [www.sqo-oss.eu](http://www.sqo-oss.eu)

tion strategies changes offering a better software quality may be proposed to conform development approach to those well-known and proven.

## 6. Discussion

Our research approach does not consider only the technical point of view of OSS development, but extends it to the whole ecosystem, including social aspects. Because OSS are commonly based on distributed development, developers create social networks where they can communicate and share their knowledge. To discover how OSS are developed and what are important factors modifying their qualities, we have to study these social aspects.

During our research we have to take into account scalability issues. The vast amount of available data imposes the use of techniques that present the analysis results in a human-comprehensible way. For instance, it is possible to represent a developer community network through a graph where nodes are developers and edges are connections between two developers working on the same file. With real OSS, such an approach quickly becomes incomprehensible due to the sheer number of nodes and edges displayed. To present a workable graph, the Girvan-Newman algorithm could be used successfully [23].

Privacy issues also become important when analysing data about users and developers, in order to prevent misuse of this data and the results obtained from analysing this data. To avoid such problems, our tools should therefore be secured, and our analysis should provide the ability to anonymise data.

As end product of our research, we will need to provide tool support to OSS communities, based on the findings of our empirical analysis. Such tool support will be useful to share transfer the knowledge resulting from our research. For example, assuming that we have been able to identify a clear correlation between software quality and developer popularity for a given OSS, we can build this knowledge into a tool that provides concrete suggestions to developer communities as how to improve software quality (e.g., by reffecting human resources to particular activities such as refactoring or bug fixing, by focusing on particular parts of the software that are more defect-prone, by improving interdeveloper communication, and so on).

A very interesting challenge would be to extend our research to deal with multi-language OSS systems. Indeed, many of the OSS systems developed today do not restrict themselves to a single programming language, but use a conglomerate of different languages. This imposes additional complexity in software understanding, and requires specific techniques to assess software quality. While some tools, such as SLOCCount, support different programming languages, most of the software metrics tools restrict themselves to a single language.

## Acknowledgements

This work has been partially financed by the F.R.S. - FNRS through FRFC project 2.4515.09 “Research Center on Software Adaptability”, and by research project AUWB-08/12-UMH “Model-Driven Software Evolution”, an *Action de Recherche Concertée* financed by the *Ministère de la Communauté française - Direction générale de l’Enseignement non obligatoire et de la Recherche scientifique, Belgium*.

## References

- [1] Ajlan Al-Ajlan. The evolution of open source software using eclipse metrics. *New Trends in Information and Service Science, International Conference on*, 0:211–218, 2009.
- [2] T. Gyimothy, R. Ferenc, and I. Siket. Empirical validation of object-oriented metrics on open source software for fault prediction. *Software Engineering, IEEE Transactions on*, 31(10):897–910, 2005.
- [3] Roberto Abreu and Rahul Premraj. How developer communication frequency relates to bug introducing changes. In *Proc. joint ERCIM Workshop on Software Evolution (EVOL) and Int’l Workshop on Principles of Software Evolution*, pages 153–157, Amsterdam, NL, 2009.
- [4] Ravi. Open source software development projects: Determinants of project popularity. *Econometrics, EconWPA*, 2005.
- [5] Yasuhiro Mashiko and Victor R. Basili. Using the GQM paradigm to investigate influential factors for software process improvement. *Journal of Systems and Software*, 36(1):17–32, 1997.
- [6] Norman Fenton and Shari Lawrence Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. International Thomson Computer Press, London, UK, second edition, 1997.
- [7] Shyam R. Chidamber and Chris F. Kemerer. A metrics suite for object-oriented design. *IEEE Trans. Software Engineering*, 20(6):476–493, June 1994.

- [8] Victor R. Basili and Walcélio L. Melo. A validation of object-oriented design metrics as quality indicators. *IEEE Trans. Software Engineering*, 22(10):751–761, October 1996.
- [9] Tom Mens, Juan Fernández-Ramil, and Sylvain Degrandart. The evolution of Eclipse. In *Proc. Int'l Conf. Software Maintenance (ICSM 2008)*. IEEE Computer Science, 2008.
- [10] Alessandro Murgia, Giulio Concas, Sandro Pinna, Roberto Tonelli, and Ivana Turnu. Empirical study of software quality evolution in open source projects using agile practices. *CoRR*, abs/0905.3287, 2009.
- [11] Lerina Aversano, Gerardo Canfora, Luigi Cerulo, Concettina Del Grosso, and Massimiliano Di Penta. An empirical study on the evolution of design patterns. In *Proc. 6th joint European Software Engineering Conf. and ACM SIGSOFT Symp. Foundations of Software Engineering (ESEC/FSE'07)*, pages 385–394, New York, NY, USA, 2007. ACM.
- [12] Jacek Ratzinger, Thomas Sigmund, Peter Vorburger, and Harald Gall. Mining software evolution to predict refactoring. In *ESEM*, pages 354–363. IEEE Computer Society, 2007.
- [13] Virtual Machinery. JHawk.  
<http://www.virtualmachinery.com/jhawkprod.htm>, 2009.
- [14] David A. Wheeler. SLOCCount.  
<http://www.dwheeler.com/sloccount/>, 2009.
- [15] Robert Gentleman and Ross Ihaka. R project.  
<http://www.r-project.org/>, 2009.
- [16] Meir M. Lehman. Laws of program evolution – rules and tools for programming management. In *Proc. Infotech State of the Art Conference - Why Software Projects Fail*, pages 1–25, April 1978.
- [17] Meir M. Lehman, Juan F. Ramil, P.D. Wernick, Dewayne E. Perry, and W. M. Turski. Metrics and laws of software evolution - the nineties view. In *Proc. Int'l Symp. Software Metrics*, pages 20–32. IEEE Computer Society Press, 1997.
- [18] Michael W. Godfrey and Qiang Tu. Evolution in open source software: A case study. In *Proc. Int'l Conf. Software Maintenance (ICSM 2000)*, page 131, Washington, DC, USA, 2000. IEEE Computer Society.
- [19] Francois Stephany, Tom Mens, and Tudor Gîrba. Maispion: A tool for analysing and visualising open source software developer communities. In *Proc. Int'l Workshop on Smalltalk Tools (IWST '09)*. ACM, 2009.
- [20] Kawin Ngamkajornwiwat, Dongsong Zhang, A. G. Koru, Lina Zhou, , and Robert Nolker. An exploratory study on the evolution of oss developer communities. In *Proc. Hawaii International Conf. System Sciences*, page 305, Los Alamitos, CA, USA, 2008. IEEE Computer Society.
- [21] Walt Scacchi. Socio-technical interaction networks in free/open source software development processes, 2005.
- [22] Juan Fernández-Ramil, Angela Lozano, Michel Wermelinger, and Andrea Capiluppi. Empirical studies of open source evolution. In Tom Mens and Serge Demeyer, editors, *Software Evolution*, pages 263–288. Springer, 2008.
- [23] Gregorio Robles. *Empirical Software Engineering Research on Libre Software: Data Sources, Methodologies and Results*. PhD thesis, Escuela Superior de Ciencias Experimentales y Tecnología, Universidad Rey Juan Carlos, 2006.