Analysing the evolution of social aspects of open source software ecosystems

Tom Mens[1] and Mathieu Goeminne[1]

Service de Génie Logiciel, Faculté des Sciences, Université de Mons
Place du Parc 20, 7000 Mons, Belgique
{tom.mens,mathieu.goeminne}@umons.ac.be
informatique.umons.ac.be

Abstract. Empirical software engineering is concerned with statistical studies that aim to understand and improve certain aspects of the software development process. Many of these focus on the evolution and maintenance of evolving software projects. They rely on repository mining techniques to extract relevant data from software repositories or other data sources frequently used by software developers. We enlarge these empirical studies by exploring social software engineering, studying the developer community, including the way developers work, cooperate, communicate and share information. The underlying hypothesis is that social aspects significantly influence the way in which the software project will evolve over time. We present some preliminary results of an empirical study we are carrying out on the different types of activities of the community involved in the GNOME open source ecosystem, and we discuss suggestions for future work.

Key words: software evolution, open source software, empirical software engineering, social software engineering, repository mining, software ecosystem

1 Introduction

This article accompanies an invited talk presented by the first author at the *Third International Workshop on Software Ecosystems (IWSECO 2011)*. It represents our ongoing research in this emerging domain.

Since the start of the new millennium, the number of empirical studies on how free / libre / open source software projects evolve has been steadily increasing. The main reasons for this are: (i) the abundance and accessibility of software projects for which historical data is freely available; (ii) the increasing popularity of open source software, even in industry; (iii) the ability to publish scientific results about these systems and to allow other researchers to verify and reproduce the obtained results.

Nevertheless, the majority of empirical studies on open source software evolution focus on the technical aspects, such as source code artefacts. These studies largely ignore the social aspect, i.e., the impact that user and developer communities (and their interaction) have on the evolution of the software project.

Important changes in the community (such as the unexpected departure of a key person, the takeover of the project by a new community) or in the software product (such as, a major restructuring or a replacement or addition of a substantial part of the code base) may significantly influence the way in which the software project will continue to evolve over time.

We therefore propose to extend empirical studies of evolving open source projects by taking into account information about the communities that are involved in this project. In particular, we wish to analyse and understand how the interaction and communication within and across communities influences the evolution of the software product and vice versa.

A better understanding of this impact will allow us, at a medium term, to come up with prediction models, guidelines and best practices that allow communities to improve upon their current practices, and tools that can be used by the community to control and improve upon their current work processes, to communicate more effectively, and to make the software more attractive to its developers and users.

In addition to this, the focus of our study is not the evolution of individual software projects, but rather coherent collections of projects developed by the same community. In this respect, we adopt Lungu's view, who defines a *software ecosystem* as "a collection of software projects which are developed and evolve together in the same environment" [1].

2 Social aspects

The developer community of a software project is composed of persons that create and modify software artefacts. Programmers modify the source code, extend the software functionaility and fix bugs. Other technical artefacts of the software product are modified by documenters, architects, testers, and so on. Persons involved in a developer community are often structured into subgroups, each focusing on a specialised activity in order to better respond to the needs of the development process.

Good communication is an essential success factor for any software project [2, 3]. This is especially true for open source projects that are often developed in a geographically distributed way. For these types of projects it is also, in most cases, easier to become involved in the development team, implying that the team structure needs to be more flexible in order to accommodate the easy integration of newcomers and to deal with the frequent departure of developers.

Open source projects rely on a number of tools accessible by the developer community to share and exchange information, to communicate and to coordinate their work. In practice, these tools make heavy use of the Internet. The main tools employed by these communities are version control systems (such as Subversion or Git), bug tracking systems (such as Bugzilla), mailing lists and developer forums. A number of researchers have started to analyse the social aspects of evolving software projects [4, 5, 6, 7, 8]. To this extent, they make use of the information extracted from the aforementioned tools.

3 Experimental setup

3.1 Research methodology

Our main research goal consists in studying communities surrounding open source software development in order to understand how their communication and interaction impacts the software evolution process.

To reach this goal, we rely on the scientific method that is common practice for research in empirical software engineering. Based on the Goal-Question-Metrics paradigm, we define specific research questions, formulate one or more research hypotheses for each question, and define and use metrics to verify the hypotheses. To achieve this, we select a representative set of open source software ecosystems (and a subset of projects for each considered ecosystem) on which to verify our hypotheses. For these selected projects, we combine data extracted from version repositories, bug trackers, mailing lists. This data is cleansed to deal with possible inconsistencies or incompleteness, to merge data corresponding to the same identity in different data sources, and to convert the data into a format that is easier to analyse. The converted data is then analysed using a combination of visual analysis, statistical analysis and data mining techniques. Whenever sufficient statistical evidence is found for a particular research hypothesis, the research questions are further refined and new hypotheses are formulated and verified in an incremental manner.

3.2 Tools

A wide variety of tools are used during our experiments: the Libresoft tools[1] for mining relevant social data from the repositories, a FLOSSMetrics-compliant SQL database for storing and querying the data[2], the R software environment[3] for statistical analysis, the WEKA tool for data mining[4], and various other tools for visual analysis of the results. All of these tools are being integrated in a layered Java framework that we presented in earlier work [9].

3.3 Research questions

Some of the research questions that will be the focus of our attention are listed below. Most of these questions are still open, so only partial answers to them will be provided in this article:

- How is the activity within a project distributed across different persons, how does this change over time, and how does this vary across different projects belonging to the same ecosystem?

[1] `tools.libresoft.es`

[2] `www.flossmetrics.org`

[3] `www.r-project.org`

[4] `www.cs.waikato.ac.nz/ml/weka`

– How is the activity of a person distributed across different projects belonging to the same ecosystem and how does this change over time?
– How is a software community structured and how does this change over time? Can we observe recurring or emerging patterns, phases and trends of communication, collaboration, organisation and activity in the project team?

3.4 Selected projects

For the purpose of this article, we have decided to study the GNOME ecosystem[5] as a case. The GNOME community develops a free and popular desktop environment for GNU/Linux and UNIX-type operating systems.

We will study several GNOME projects within this ecosystem. They have been selected based on the following factors: popularity, age, size, number of people involved, availability of the necessary data sources for analysis. The names and characteristics of some of the selected projects are presented in Table 1. The data for all these projects is stored in different *git* repositories, a free and open source distributed version control system. Observe that the number of committers and number of authors reported in Table 1 differ, since not all authors have commit rights.

project ID project name	A Banshee	B Rhythmbox	C Tomboy	D Evince	E Brasero
age (in years)	5.9	8.9	6.6	12.1	4.2
date of last commit	8/5/2011	9/5/2011	9/5/2011	4/5/2011	22/11/2010
# commits	8427	7979	5791	5024	4129
# committers	160	232	211	274	145
# authors	268	364	290	381	193
# files in most recent version	2700	937	766	699	797
# files during project's life	13388	2767	5075	2701	2223

Table 1: Main characteristics of 5 selected GNOME projects.

4 Empirical study

Based on the experimental setup of section 3, we are carrying out three different studies. It is important to note that these studies are still ongoing, and in this article we present only preliminary results without statistically validating any hypotheses.

The first study in subsection 4.1 focuses on individual GNOME projects, and aims to correlate data from different data sources: the *git* code repository, the

[5] www.gnome.org

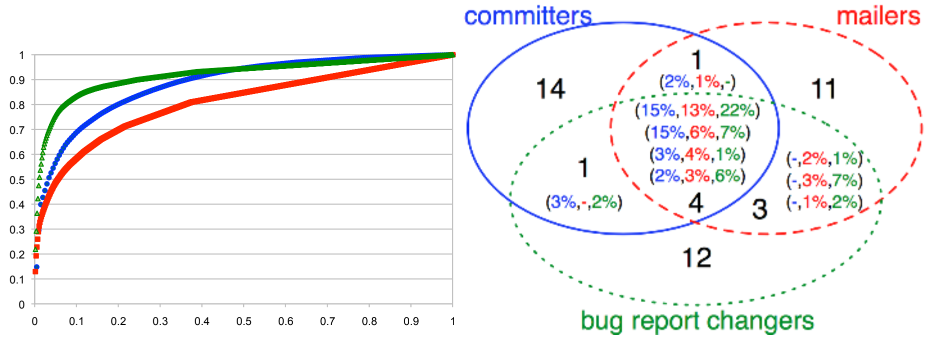
bug tracker, and the developer mailing list. The second study in subsection 4.2 takes a more fine-grained view on the activity patterns of authors in the code repository only, and aims to find correlations between different types of activity. The third study in subsection 4.3 aims to correlate activities across different GNOME projects.

4.1 First study

The first study aims to relate information about the open source project community by analysing three different data sources for a single Gnome project: the code repository, the bug tracker and the mailing list. These results have been reported in a previous article [10].

Although we have carried out the analysis for each of the Gnome projects of Table 1, we only present the results for *Evince* here, a freely available document writer that is mainly developed in C and C++, and has more than 11 years of development's history. *Evince* is a small project, with roughly five thousand commits, nearly two thousands of e-mails and about a thousand bug reports (for the time period we studied).

We wish to understand how the activities of this software project are distributed among the project's contributors. For this, we use the information related to three categories of activity concerning the same person: the "commits" done, the mails sent, and the modifications made to bug reports.



(a) (b)

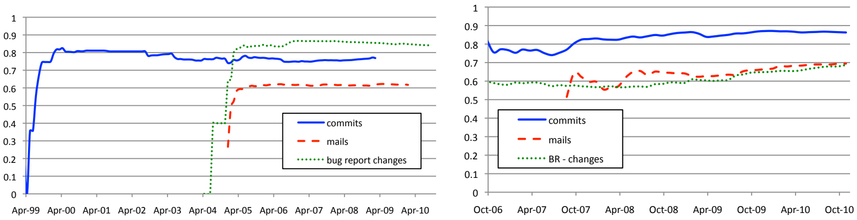
Fig. 1: Activity analysis of the community surrounding *Evince* (November 2010). **1a:** The cumulative distribution marked with blue circles corresponds to the commit activity. The cumulative distribution marked with red squares corresponds to the mail sending activity. The cumulative activity marked with green triangles corresponds to bug report change activity. **1b:** Intersection of the activity categories for the top 20 most active persons in each category.

Figure 1a shows the cumulative distribution for these three categories of activity. The distribution is very unbalanced: a small number of persons is responsible for the majority of the activities. 20% of all committers are responsible for 80% of the total commit activity in the version control repository; 20% of all mailers contribute to 70% of all mails sent; and 20% of all bug report changers take part in 88% of all bug report changes.

Figure 1b offers a more detailed overview of the same data, matching persons who simultaneously contribute to the three considered categories of activity. For each category, only the top 20 of most active persons has been considered. The figure clearly shows that most active persons contribute to several activity categories. For example, the two most active committers (15% and 15%) are also very active in mail sending (13% and 6%) and bug report changes (22% and 7%).

The imbalance in the activity distribution can be summarised by econometrical aggregation indices, like Gini, Theil or Hoover [11, 12, 13]. A zero value for these indices implies a uniform distribution, which means that each person has the same activity rate. A value of 1 means that a single person carries out all the work and the others do nothing. We can compute the indices on several dates in order to visualise how the distribution imbalance evolves over time.

Figure 2 shows this evolution for Evince, using the Gini index for the three considered activity categories. In each case, after a startup phase where the index is very rapidly increasing, the index tends to stabilise arround a high value (around 0.8), signifying an important imbalance in the activity distribution. For the mail sending activity this imbalance is less significant as the coefficient stabilises around 0.6, which means there are more persons regularly involved in the sending of mails.



(a) Evince

(b) Brasero

Fig. 2: Comparison of the Gini index for *Evince*, since April 1999 for the commits (continuous blue line), since January 2005 for the mails sent (dashed red line), and since August 2004 for the bug report changes (dotted green line).

4.2 Second study

While the previous study informs us about the way in which the members of a project community contribute to and participate in different types of repositories, we can also analyse the activity patterns of persons within a single repository. Following Robles *et al.* [14], the idea is that the type of activity a person involved in can be approximated by the types of files this person is contributing to (i.e., adding, modifying or deleting files) in the version control repository.

Table 2 shows how we defined activity types based on the structure of the file names and file extensions. File extensions are typically used for most of the file types. The matching rules are build thanks to the extensions commonly used in software development as well as the extensions observed in the studied projects.

Activity type	**File type**
coding	*.c, *.h, *.cc, *.pl, *.java, *.s, *.ada, *.cpp, *.chh, *.py
development documentation	readme*, *changelog*, todo*, hacking*
documenting	*.html, *.txt, *.ps, *.tex, *.sgml, *.pdf
translating	*.po, *.pot, *.mo, *.charset

Table 2: Activity types and their corresponding file types. Only the most important activity types (i.e., those that occured most often in all considered projects) are listed here.

Analysing the *git* repository of *Evince*, we observe that a minority of authors, 132 out of 381 (i.e., 34.6%) are active in the coding activity (meaning that they are involved in at least one commit of a coding related file), whereas the total number of commits for these types of files represents 46.2% of all the file commits. We can therefore conclude that coders are among the most active persons in the project community. This is not very surprising, since version control repositories are specifically aimed to manage the evolution of a software project's source code.

The second most important activity for *Evince* is development documentation, with 19.0% of all committed files attributable to this activity. Compared with source code files many more authors, namely 241 out of 381 (i.e., 63.3%) are involved in this activity. The third most important activity, with 12.6% of commits done on the associated files, is software translation. Translating is also a very popular activity, since it involves 248 authors out of 381 (i.e., 65.1%). All these results are visually summarised in Figure 3, taking into account the data over the project's entire lifetime.

Figure 4 studies whether the same persons are involved in different activities. We only show this for the activities of coding, translation and development documentation. The Venn diagram illustrates how many persons are involved in 1, 2 or 3 of these activities over the project's lifetme. We observe that most of the coders (97 out of 132, i.e., 73.5%) are also development documentalists and many translators (109 out of 248) are also development documentalists. We also

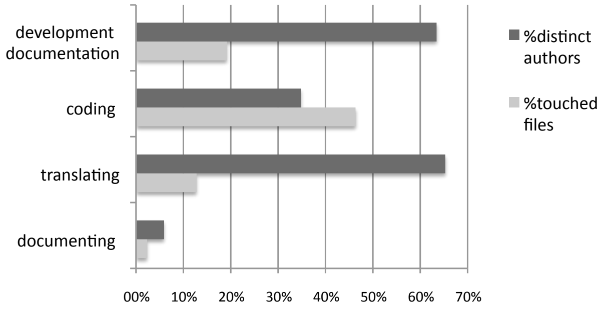


Fig. 3: Percentage of *git* authors involved in different *Evince* activities (in dark gray), and percentage of files touched for each activity (in light gray).

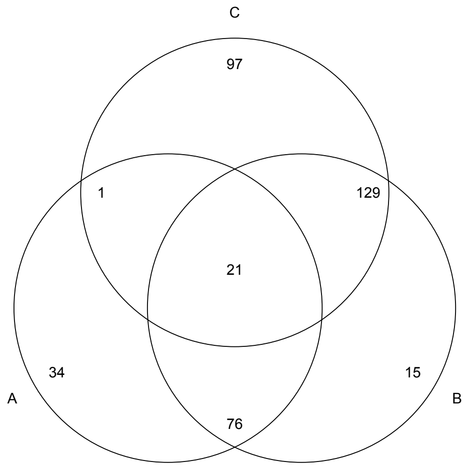
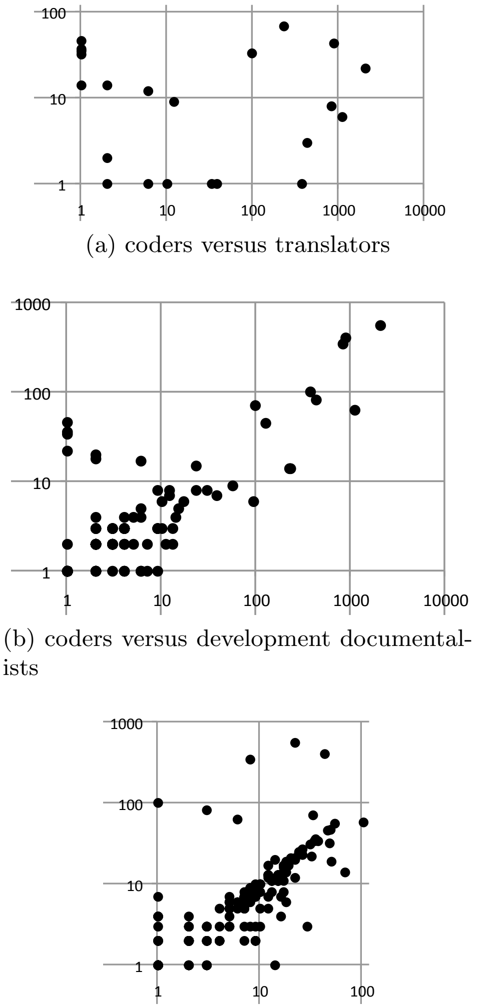


Fig. 4: Intersection of *git* authors for *Evince* involved in 3 types of project activities: A = coding, B = development-documentation, C = translating.

remark that few coders (14 out of 132) are involved in translation and *vice versa*. This disparity reveals the importance to take into account the type of activity while analysing historical data from version repositories.

Based on the results displayed in Figure 4, we wish to understand the amount of work that persons taking part in two different activities (corresponding to the intersections between two circles in the Venn diagram) are involved in. Figure 5 shows scatterplots for each pair of activities. Each point represents the amount of files touched by a given author for the two considered activities. Figure 5a shows that only a few coders are also involved in translation, and no particular trend can be observed. Figure 5b compares coders and persons involved in development documentation. We observe a clear trend: more active coders are also more active in development documentation too. Figure 5c compares the translators with the persons involved in development documentation: again, more active translators appear to be more active in development documentation as well, and *vice versa*.



(a) coders versus translators

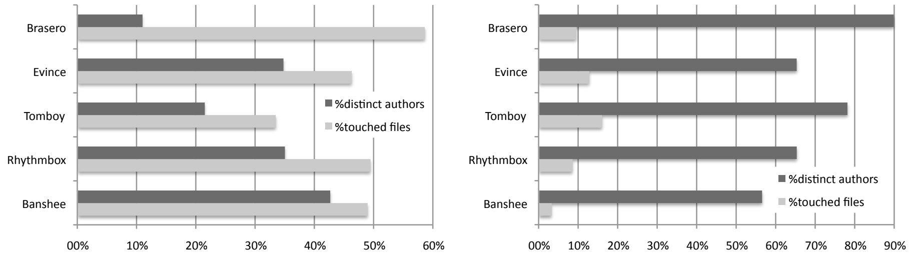
(b) coders versus development documentalists

(c) translators versus development documentalists

Fig. 5: Scatterplot of all authors involved in two different types of activity in the *Evince gitrepository*. Each dot represents the number of files touched by a particular author for two out of three considered activity types: coding, developer documentation and translation.

4.3 Third study

Our third study extends the second to the level of the GNOME software ecosystem. More precisely, we study the collection of selected GNOME projects as a whole (as opposed to individual projects), and we try to find correlations between certain project activities. As in the second study, we restrict ourselves to analysing the data stored in the version control repositories. In other words, we rely on the information contained in the *super-repository* [6] of GNOME. From a technical point of view, for the case of GNOME, this super-repository is basically a collection of distinct *git* repositories (one for each GNOME project).



(a) Coding

(b) Translation

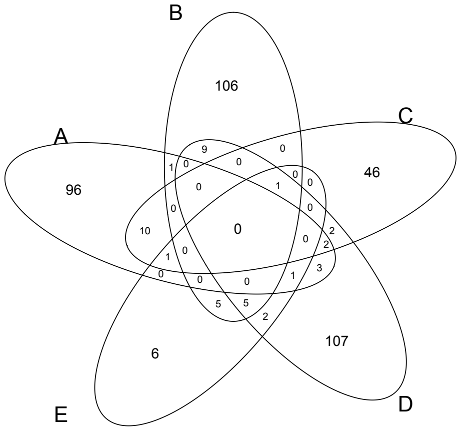
Fig. 6: Percentages of *git* authors (dark gray) and files touched (light gray) for the activities of coding (6a) and translation (6b) for the five selected GNOME projects of Table 1.

To start with, we computed the results of Figure 3 for all selected GNOME projects, and displayed them in Figure 6 for the activities of coding and translation, respectively. This corroborates what we already observed in Figure 3: the number of authors that contribute to the coding activity is fairly low (between 11% and 43%) while they touch a significant number of files in the version repository (between 33% and 58%). The most striking result is found for Brasero, where only 11% of the authors touched 58% of all files in the version repository. The activity of translation lies on the other extreme of the spectrum: a lot of authors are involved in the activity (between 56% and 90%) while the percentage of files touched remains very small (between 3% and 16%).

Figure 7 illustrates how authors involved in these two activities are involved in multiple GNOME projects. We observe that the pattern of collaboration across projects is very different for these two types of activities.

Coders seem to stick to a single project. It is rarely the case that a coder is involved in two different GNOME projects, and even more rare for a coder to be

[6] Lungu [1] defines a super-repository as "a collection of version control repositories of the projects of an ecosystem."



(a) Intersection of coders (b) Intersection of translators

Fig. 7: Intersection of *git* authors contributing to the five selected GNOME projects of Table 1. 7a shows the authors involved in coding and 7b shows the authors involved in translating.

involved in more than 2 GNOME projects. The intersection over all five selected projects is even empty. For the activity of translation, the picture is very different. More often than not, translators are involved in multiple GNOME projects. For the 5 selected projects we find that 69 different translators contribute to each of them at least once.

We can thus conclude that the way in which persons cooperate across projects heavily depends on the type of activity they are involved in. This may either be due to the intrinsic characteristics of the type of activity (for example, translating text from one language to another is less time consuming and requires less project-specific knowledge than coding), or to the presence of external tools and mechanisms used by the community to share and distribute work. In the case of GNOME the main reason is the presence of the GNOME Live! Translation project[7] that manages and structures the way in which translations are carried out across GNOME projects. For the different supported languages, translation teams exist. Basically, this implies that, for each language, there is a group of translators that take care of translating files to this language across all GNOME projects. It is clear that this tool helps to increase the collaboration across GNOME projects. If a similar mechanism would be available for coders, it is likely that cross-project cooperation between coders would also increase.

[7] `live.gnome.org`

5 Discussion

While our preliminary results reveal interesting patterns that encourage us to pursue this line of research further, it is clear that a lot of work remains to be done.

To start with, we need to perform a sound statistical analysis of the obtained results, over all considered types of activities and over all projects involved in the GNOME ecosystem. We need to refine and extend the types of activities considered, and we also need to take into account activities that can be found from the data stored in the bug tracker and mailing list. All the results we find need to be validated on, or generalised to, other software ecosystems as well.

For study 2 and 3, we need to include the evolution dimension, by studying how the discovered activity patterns evolve over time. We also wish to extend the studies towards evolutionary patterns from the viewpoint of individual authors (or coherent groups of authors): how does the way in which an author contributes to a software project community evolves over time? Although generally committers seem to be only concerned by only one software project, a follow-up study will analyse if committers can be involved in some related projects, such as a library and its associated graphical user interface. We will also like refine our studies by distinguishing those authors that created new files from those that only edit existing files. We also wish to study whether the core groups of authors (i.e. those that are most active for a particular activity) tend to be stable or whether they evolve over time.

Another interesting point of study is the exploration of the relation between the social and the technical dimension of open source software development. In particular, we are interested in how software quality is influenced by the way the community interacts, and vice versa. We are also interesting in the migration of authors: in the case of a fork, can we predict who will be the authors that will migrate from the original project to the new one? Are the authors simultaneously working on all the projects they are involved in, or is there a migration effect from one project to another over time? Can we observe the same patterns when we consider the several branches of a single project?

For all of the above studies, we wish to use a wide range of different mechanisms, coming from a variety of domains such as data mining, statistical analysis, economy, software visualisation, social network analysis, and system dynamics.

To facilitate the empirical studies, we need to provide more tool support, and improve existing tools for data extraction and analysis. While an important part of the work has been automated, there is still quite some amount of manual intervention involved that is amenable to automation. At a medium term, we wish to come up with prediction models, guidelines and tools that allow communities involved in software ecosystems to communicate and interact more effectively. Prospective users and developers may also rely on such information to make a more informed choice on whether or not to get involved in such an ecosystem.

6 Conclusion

Social aspects have a significant impact on the way software ecosystems (i.e., coherent collections of software products) evolve over time. Empirical studies of software evolution must therefore take into account the community surrounding the software as well as the way this community influences the software evolution.

This article has only scratched the surface of what can be done, by illustrating some initial empirical studies on the different types of activities the community members are involved in. Considerably more work is needed to get a deeper understanding of how this affects the way the software product evolves, and how this varies from one project to another, in order to come to tool support and guidelines that can help the software community to optimise their work processes and produce high quality code more effectively.

Acknowledgment

The research is partially supported by (i) F.R.S.-FNRS FRFC project 2.4515.09 “Research Center on Software Adaptability”; (ii) the European Regional Development Fund (ERDF) and Wallonia; (iii) Action de Recherche Concertée project AUWB- 08/12-UMH “Model-Driven Software Evolution”, financed by the Ministère de la Communauté française - Direction générale de l’Enseignement non obligatoire et de la Recherche scientifique, Belgium.

References

1. Lungu, M., Lanza, M., Gîrba, T., Robbes, R.: The small project observatory: Visualizing software ecosystems. Science of Computer Programming **75** (2010) 264–275
2. Brooks, Frederick P., J.: The Mythical Man-Month: Essays on Software Engineering. Addison-Wesley (1975)
3. DeMarco, T., Lister, T.: Peopleware: productive projects and teams. Dorset House Publishing (1987)
4. Madey, G., Freeh, V., Tynan, R.: The open source software development phenomenon: An analysis based on social network theory. In: Eighth Americas Conference on Information Systems. (2002) 1806–1813
5. Mockus, A., Fielding, R.T., Herbsleb, J.D.: Two case studies of open source software development: Apache and mozilla. ACM Trans. Softw. Eng. Methodol. **11**(3) (2002) 309–346
6. Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., Ye, Y.: Evolution patterns of open-source software systems and communities. In: Proc. Int’l Workshop on Principles of Software Evolution, New York, NY, USA, ACM (2002) 76–85
7. Ye, Y., Nakakoji, K., Yamamoto, Y., Kishida, K.: The co-evolution of systems and communities in free and open source software development. In Koch, S., ed.: Free/Open Source Software Development. IDEA Group Publishing (2005) 59–82

8. Weiss, M., Moroiu, G., Zhao, P.: Evolution of open source communities. In Damiani, E., Fitzgerald, B., Scacchi, W., Scotto, M., Succi, G., eds.: Open Source Systems. Volume 203 of IFIP International Federation for Information Processing. Springer Boston (2006) 21–32
9. Goeminne, M., Mens, T.: A framework for analysing and visualising open source software ecosystems. In: Proceedings International Workshop on Principles of Software Evolution (IWPSE-EVOL), ACM Press (September 2010) 42–47
10. Goeminne, M., Mens, T.: Evidence for the pareto principle in open source software activity. In Bruntink, M., Kontogiannis, K., eds.: CSMR 2011 Workshop on Software Quality and Maintainability (SQM). Volume 701., CEUR-WS.org (2011) 74–82
11. Vasa, R., Lumpe, M., Branch, P., Nierstrasz, O.: Comparative analysis of evolving software systems using the Gini coefficient. In: Proc. Int'l Conf. Software Maintenance. (2009) 179–188
12. Serebrenik, A., van den Brand, M.: Theil index for aggregation of software metrics values. In: IEEE International Conference on Software Maintenance, Los Alamitos, CA, USA, IEEE Computer Society (2010) 1–9
13. Poncin, W., Serebrenik, A., van den Brand, M.: Process mining software repositories. In Mens, T., Kanellopoulos, Y., Winter, A., eds.: CSMR '11: Proceedings of the European Conference on Software Maintenance and Reengineering., IEEE Computer Society (2011) 5–14
14. Robles, G., González-Barahona, J.M., Izquierdo-Cortazar, D., Herraiz, I.: Tools for the study of the usual data sources found in libre software projects. IJOSSP **1**(1) (2009) 24–45