An empirical study on the specialisation effect in Open Source communities

Mathieu Goeminne and Tom Mens

October 28, 2011

1 Introduction

Since a couple of decades, open source software has gained popularity due to the savings they represent and the ability for the users to modify and improve the software themeselves. As the number of projects which the entire history is available grows over time, the number of empirical studies on them grows as well. Most of these empirical studies are carried out with no consideration for other artefacts but source code. Unfortunately, a restriction to the study of source code evolution only is not sufficient to understand and explain some evolutionary behaviour. In order to gain a better insight into how a software project evolves, information about persons involved in the software development, and in particular developers, must be taken into account.

We are carrying out an empirical study on the evolution of the GNOME[1] ecosystem, a collection of 1325 open source projects. Our aim is to study how the developers involved in an open source software community organize themselves to share development tasks.

2 Methodology

In a first phase, we extracted information from the source code repositories of 1325 GNOME projects, and stored them in FLOSSMetrics-compliant[2] databases. We defined 13 file categories, such as *code*, *image*, *documentation*, based on the file types, file names and directories in which files are stored. For instance, *.png* files will be classified as images. Each file category represents a type of activity done by the developer who touched the file. We assigned such an *activity category* to each file that has been created, modified or deleted during the project's life. Our classification process is inspired by the one of Robles *et al.* [2].

A recurrent problem when carrying out empirical studies involving source code repositories is identity matching [1]. Persons involved in open source projects can have several user accounts they use. In order to get a more accurate model of the developer's activities, all accounts belonging to the same physical person must be merged in a single identity representing the person. In our approach, we have addressed this problem into account and used unique identities representing physical persons instead of their multiple accounts.

3 Research questions

In order to gain a better insight of how developers organize themselves over time, we are studying the following questions:

- Are developers mainly active in a small number of projects?
- Are developers mainly involved in a small number of activity types?

[1] `www.gnome.org/`
[2] `www.flossmetrics.org/`

- Is there a correlation between certain activities related to the software development?
- Does the number of developers involved in an activity type affect the extent to which these developers specialise themselves?

These questions must be refined in subquestions. Preliminary results of our study on the GNOME projects lets us hypothesise that the developers are highly specialised in some categories of activities, like coding (`code`) or translation (`i18n`), whereas in other categories the development process is not subject to specialisation, as the boxplots of Figure 1 show. In the figure each developer is represented by a value between 0 and 1 for each category. The value expresses the specialisation degree of the considered developer. A value of 1 means that the developer only works on this type of files, a value of 0 means that the developer has never worked on this type of file. Categories *code*, *i18n* and *develdoc* reveal a specialisation degree higher that is significantly higher than the other ones.

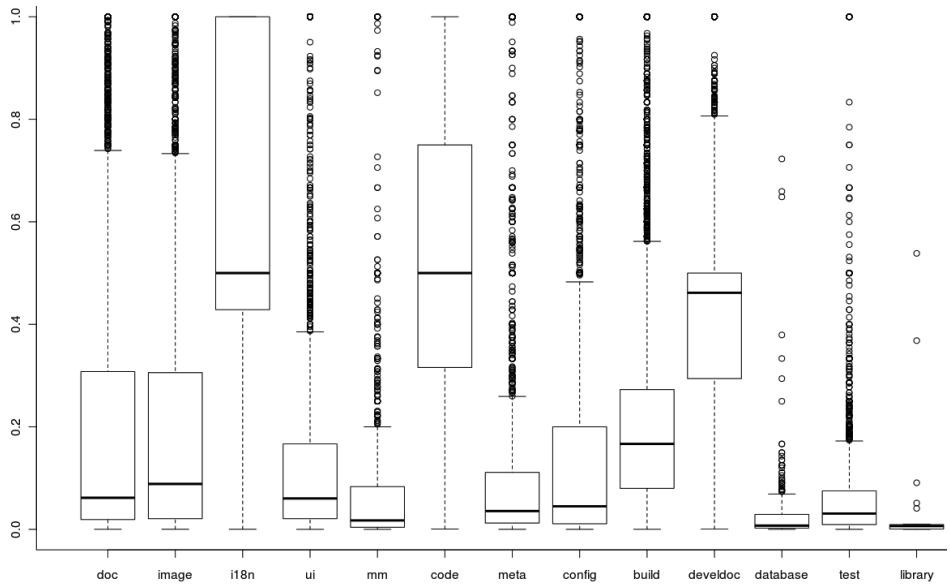

Figure 1: Developer's specialisation across GNOME projects for each activity category.

References

[1] Mathieu Goeminne and Tom Mens. A comparison of identity merge algorithms for software repositories. *Science of Computer Programming*, 2012.

[2] Gregorio Robles, Jesus M. Gonzalez-Barahona, and Juan Julian Merelo. Beyond source code: the importance of other artifacts in software development (a case study). *J. Syst. Softw.*, 79(9):1233–1248, 2006.