Open Source Is a Continual Bugfixing by a Few

Mikołaj Fejzer, Michał Wojtyna, Marta Burzańska, Piotr Wiśniewski, and Krzysztof Stencel

Faculty of Mathematics and Computer Science,
Nicolaus Copernicus University,
Toruń, Poland
{mfejzer,goobar,quintria,pikonrad,stencel}@mat.umk.pl

Abstract. Github is one of the most popular repository sites. It is a place where contributors come together to share code, ideas, thoughts and report issues. By using topic modelling applied to comments we are able to mine plentiful interesting information. Three aspects of an open source project mostly attracted our attention: the existence of a "Core Team"' - small number of developers that have the most contributions, the prevailing popularity of topics related to bug fixing and the continuous development of project without significant iteration phases.

Keywords: Bug fixing, Developers behavioural patterns, Development phases, Github, LDA, Topic analysis, Team work.

1 Introduction

Today, the the most popular code repository sites for open source projects are Github and Sourceforge. They gather massive data on users, their activities and the code they produce. From 2014 MSR Mining Challenge [1] we have obtained a portion of repositories stored in Github. After a careful analysis of the provided data structure, we have decided to study the influence of committers on their projects. We focused our attention on mining the information from commit messages and issue comments.

In order to generalize information about each of the studied commits and issues and to gather statistics, we used topic modelling [2]. Each comment has been treated as a single document. To obtain topics we applied the Latent Dirichlet allocation [3] using the Mallet topic modelling toolkit [4]. We trained our topic model per project, to capture each project's unique history and the context of programmers interaction. As a next step we have aggregated the data by a number of attributes - among which the most helpful aggregations were by date and by author.

Based on the series of empirical tests of training Mallet with different parameters, we have finally decided that 50 topics and 1000 iterations gives us the best generalization without losing too many specific details about the studied data. We have also added custom stop words that matches the Github context to clean up committers' messages. Mostly we had to deal with numerous comments

Y. Manolopoulos et al. (Eds.): ADBIS 2014, LNCS 8716, pp. 153–162, 2014.

containing "thank you", "good job" and other such praising that fell out of our scope of interest.

Unfortunately for our approach, out of 92 projects provided, only 52 had a sufficient number of commit messages allowing to extract reliable topic, and of those 52 only 43 projects had enough issue comments to be studied. We have examined the resulting data to verify a number of hypotheses, sometimes discovering new ones worth investigating. The first thing we have noticed was that repeatedly the bug fixing topic was always either the most popular topic, or among the top 5. We shall address this issue in Section 2. Also our initial observations indicated that most of the projects have a small hard working "core developers" group comprising of specialists in one or two topics and people who contribute to almost every topic. More detailed information about the corresponding hypothesis, gathered data and results can be found in Section 3.

Of course among many interesting questions some remained unanswered, while some turned out with a negative answer although our initial intuition suggested that they should validate positively. Two most noteworthy of such hypotheses are the close correlation between issue topics and following commit topics and that the open source projects are created iteratively. Those hypotheses are addressed in the Section 4.

This report deals mostly with three hypotheses:

- In most of the projects the contributors can be divided into two main groups: a great number of contributors is responsible for only a small portion of code, since their input is minor. On the other hand - a small group of contributors, let us call them the "Core team", is responsible for substantial developments through majority of commits to the code.
- Despite advancements of software engineering open source projects are not developed according to a methodology based on any form of cycles. In fact, in general the development process is iterative.

In following section we take a closer look at the aforementioned hypotheses.

2 "The Core Team"

Our goal was to check how do people involved with a project contribute to its development. Do they mostly assume a role of a specialists - local domain specific experts - or perhaps they are generalists who contribute to different parts of their project? Or maybe there are other who fall out of those two categories. Furthermore, how big (in percentages) are those groups and can we find some universal trends on this matter?

In our terminology a specialist is a committer, whose number of commits matching specific topic is larger than half of the maximum number of commits to the most popular topic of a project. A generalist is a committer whose number of commits matching multiple topics is larger than average number of topics per committer.

After analysing the data of the chosen 43 projects we found out that in each project the majority of committers behave like partisans, using hit and run tactics. They create only one commit matching specific topic and disappear, never to be seen again. The average number of committers per project is 35.9 contributors which is equal to 70.93% of average project's committers. Specialists usually concentrate on the most significant topics. Our calculations show that they constitute average 2.07 (12.06%) committers per project. They are also competent or willing enough to create numerous commits matching other topics, so many of them are also counted among generalists, whose average is 9.59 (28.12%) of committers per project.

In order to illustrate the issue we have selected two charts generated from data aggregated by comments' author. Figures 1 and 2 visualize the number of contributions to a topic by an author. For a project 3583 presented in Figure 1 the global number of committers is 84 and there are 5 specialists and 27 multi-topic generalists. The project 107534 from Figure 2 has 34 contributors in total, among which there are 3 specialists and 10 generalists.

Similar research has been conducted by the author of [5]. They have studied the Apache and Mozilla projects and their contributors. Despite slight differences in percentages (their results were closer to 20%-80% ratio of the number of core team developers to others), they came to a similar conclusion. A minority of contributors is responsible for the majority of work.

Table 1. Statistics of committers groups

Specialists	Generalists	Other	Definition
12,06%	28,12%	70,93%	Average percentage per project
2,07	9,59	35,9	Average number per project
5,77%	26,70%	72,96%	Percentage of all
85	393	1074	Number of all

3 Bugfixes

The analysis of the charts containing popularity of topics has lead us to our second hypothesis: bug fix commits are notably popular in open source projects. Our first intuition was that bug fixing related topics should have significant popularity. We analysed data fetched from Github database using our generic solution based on Mallet as follows. We took all comments of each commit and created a list of topics describing it. Every topic consists of top 10 words with frequencies. We classified a topic as bug fix related if it contained at least 2 words from the list of bug fix keywords, such as: fix, bug, solve. We took all commits of a project and generated a reduced list of 10 topics. In a 50 topic list we have often seen a number of bug fix related topics, thus the need for reduction.

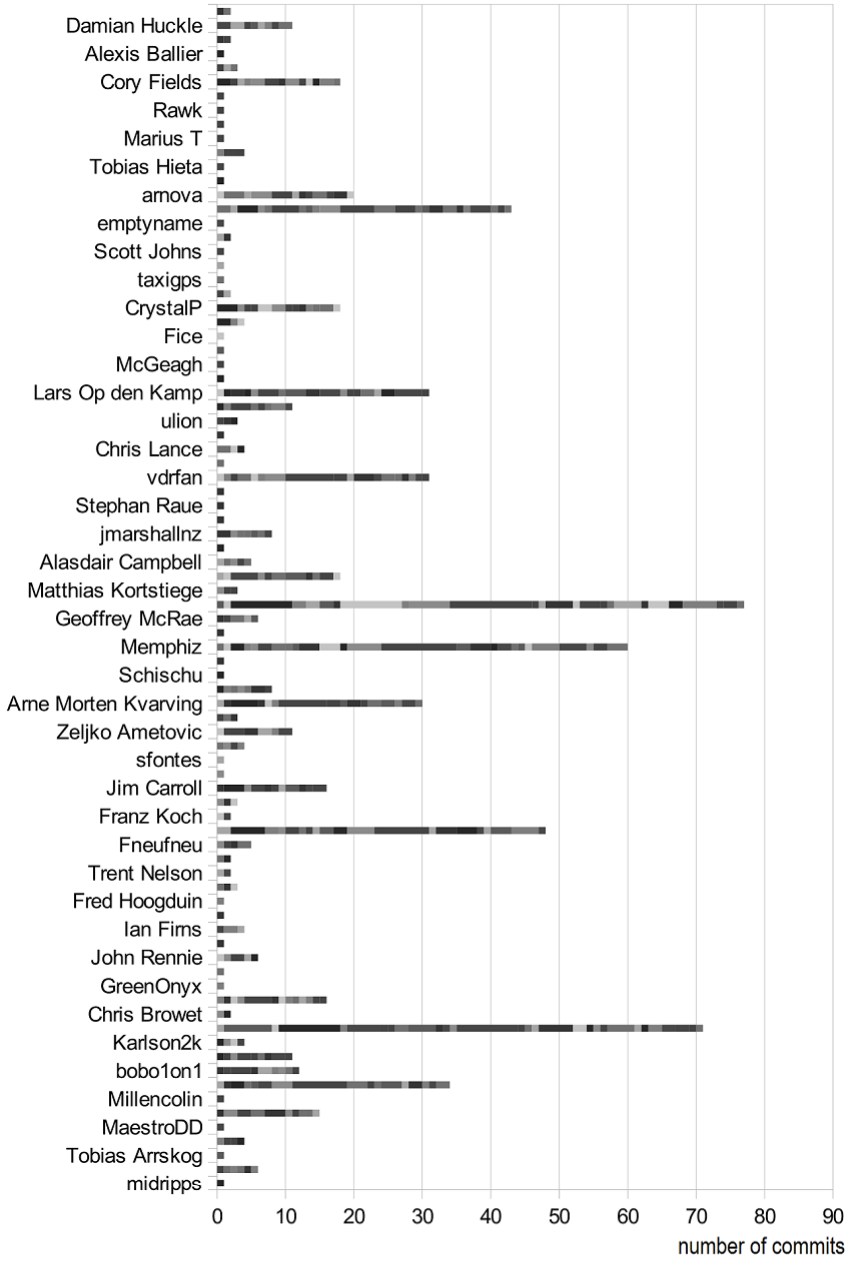

Fig. 1. Topic aggregation by committer's name on project 3583

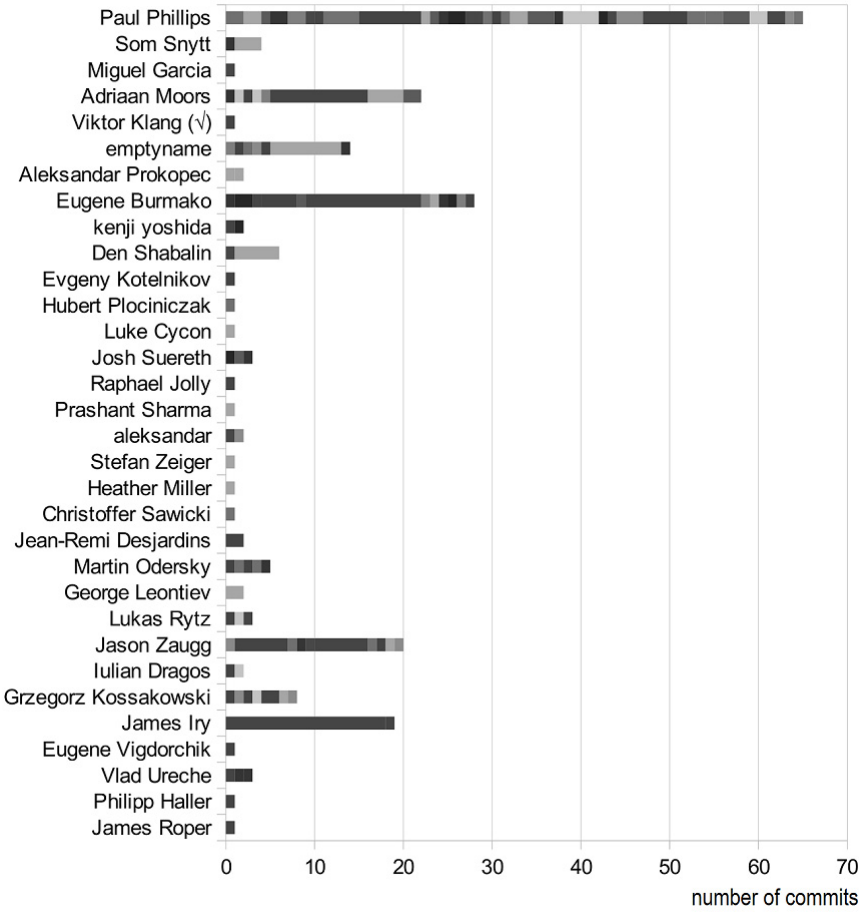

Fig. 2. Topic aggregation by committer's name on project 107534

Now, we have assumed that developers' work is focused mainly around fixing bugs if a project contains at least one bug fix related topic within top 5 topics list. We have found out that most of the chosen projects share topics which we can classify as related to bugs fixing. Only one of 44 projects did not have a bug fixing related topic. Also, bug fix topics are often equally distributed over time. Bug fix commits usually span over many months or even years. Moreover, these topics are very popular. Naturally, this does not mean that in every project the most popular topic is related to bugs fixing. Usually, the most popular topic in a project represents specific project-related problems. Actually, nearly half of the projects could be classified as "bug-fix" project as 20 out of 44 had bug-fix topic among their top 5. This might seem reflecting open source model of work. There are copious contributors and testers. Thus, issues are reported/proposed more often and faster than in commercial projects. That is why developers might have a clear vision of what to do and they can just focus on solving reported problems. At the same time open source projects tend to have less formal work organization. This might be another source of potential bugs. Investigation of the reasons behind the popularity of bug fixing is a very interesting research subject, however it would require access to a lot more data, e.g. both source code and user interactions like comments or mailing lists.

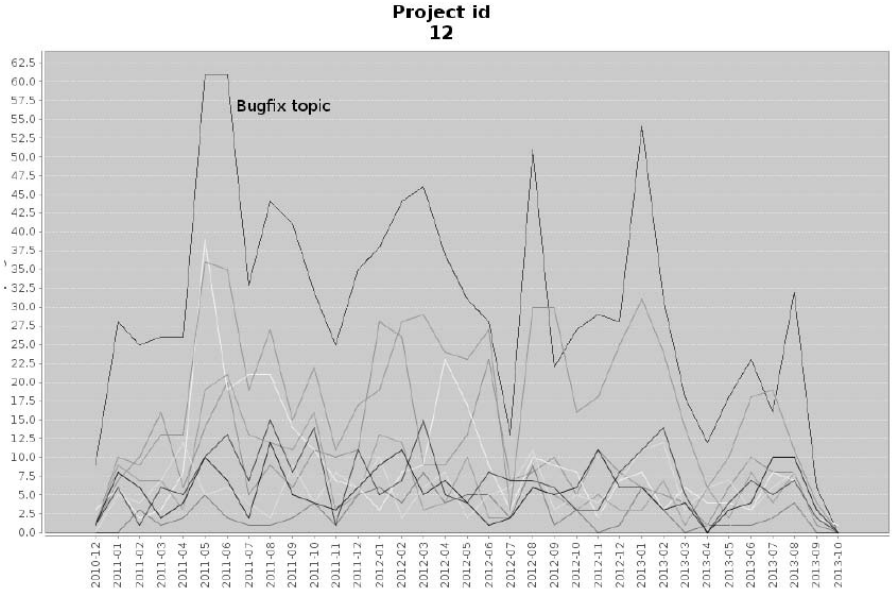

Fig. 3. Topic aggregation by commit date on project 12

Figures 3 and 4 present two example charts showing distribution of commit topics over time in two projects. The first chart (project id 12) represents topics of TrinityCore Open Source MMO Framework. The second chart (project

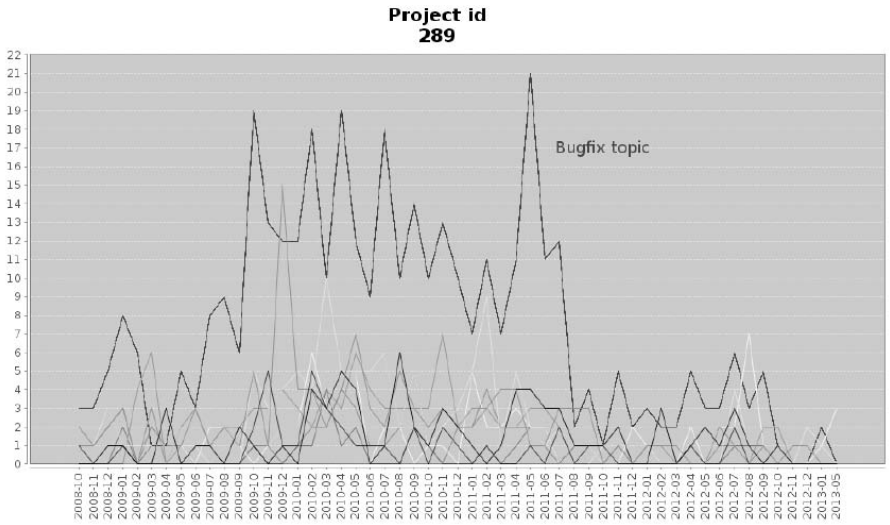

Fig. 4. Topic aggregation by commit date on project 289

id 289) concerns MaNGOS, a MMO server suite. Both of them have plentiful contributors and commits over time. Therefore they are suitable to show some tendencies. Both projects have also rich histories (unlike many other projects), ranging from 2008 to 2013 (project 289) and from 2010 to 2013 (project 12). We can see that in both projects bug fix commits are not only notably popular, but also spanned across the entire chart. This leads us to conclusion that in both projects developers' main efforts are focused on solving issues/bugs reported by the community or other developers.

4 Project's Development Phases

Before we began to study the data provided by the 2014 MSR Mining Challenge [1] we strongly believed that the majority of open source projects is being developed in cycles loosely corresponding to either classical prototype methodologies, or sprints used in Scrum. To verify this hypothesis we have studied topics of issue comments and commit messages separately, in both cases with and without the most significant topics among. We have also tried reducing the initial number of 50 topics and then eliminating the most significant one. However, much to our surprise, in most projects we were unable to detect cycles of topics (of either issues or commits), such as a regular increase and decrease of interest in particular topic in specific months. Only one project (51669) had a form of topics' cycle between 2012-07 and 2013-03. Numerous projects probably spanned over a too small period of time, or had too few contributions to detect cycles even if committers worked according to them.

We have revealed that in almost all projects there exist major topics (often a single topic) that are generally popular and prevalent during whole development process. And yet, even after excluding those (usually top 3) topics, the only topics remaining are those which have a very small number of commits (for example only one commit) or few issues gathered around a specific date or occasion. In case of commits those situations may be caused by merging work of those committers who behave like partisans. While investigating this hypothesis we came across two other questions. Are commits related to issues? Or at least do bug fixing topics in commits indicate some correlation to issues? Unfortunately the answers we have found are both "no".

Topics of issues are generally not similar to those of commits. Issue topics concentrate more on how something can be achieved and are more broad discussions, e.g. on the architectural context of a project or the usage of a selected library. Commit topics usually describe a specific situation such as not merged commit, a failed compilation or specific reasons why commit should be changed. Issue topics generally do not seem influencing commit topics to appear. It is true it least in the Github projects chosen for the Challenge.

5 Research Limitations

Our results are subject to a number of limitations threatening the accuracy or even the correctness of our assumptions. Our biggest concern is that we did not have access to the source code. Therefore, we were unable to verify if the specialists actually produce a majority of code, or they simply make lots of small corrections - possibly even insignificant. The same concerns generalists. Their work may additionally be triggered not by the desire to improve their projects but more by the reputation and contribution score. On the other hand "partisans" may contribute to a project not by Github itself but through other means like forums - where they may post helpful code, hints or suggestions.

Another problem that should be clearly stated here is that we analysed a very small number of projects. As mentioned earlier only 92 projects were provided, out of which only 43 were subjected to our method of investigation. And even those projects usually had a very small number of committers and comments making it difficult to generalize based on them. In general people tend to leave commit messages empty or they include a short, nearly meaningless sentence.Thus, without looking at the source code it is impossible to say anything about such commit. One of our concerns here is that we were unable to distinguish between commits that are meaningful to a project and commits that were cancelled or overwritten.

Our approach itself left a space for additional work. The fact, that we have eliminated topics related to praising and thanking others for their work might have negatively influenced our results. The same goes for parameters we used when training the LDA. For topics generated in some other way for example there may exist software development cycles.

6 Conclusions

In this paper we have shown that the majority of open source projects have a strong "Core Team", i.e. a group of developers that are either strongly involved in the development of a chosen topic, or they browse through the project bringing together committers and their contributions. The work of open source developers, no matter what is their role in a project, usually involves a notable amount bug fixing. Moreover, while investigating this problem we have found no correlation between bug fixing and Test Driven Development, or even simple testing phases. This leads to our third main conclusion that in most open source projects the work is continuous and cannot be clearly divided into stages or phases.

There are still numerous interesting hypotheses to be researched. As we have mentioned earlier, as we dug down the data, more and more questions arose. For example, is a project's popularity related somehow to the main technology? Or what are the trends behind the open source development? One of the hypotheses that we were unable to verify due to the small sample of projects is connected with the existence of the Core Team. Specifically, how big should be the leading team for a project to succeed? Or is a small core team a guarantee of a failure? This questions have been also formed within the research paper on Apache and Mozilla projects [5]. But we may also try to analyse means of interaction between the "Core Team" and other members, bearing in mind the research in [6]. How open are the core team members to other participants, and does it corelate to the amount of bug-reports or commits done by them. Going further on this subject we may want to ask what other project's features are directly linked to the project's success or failure. The authors of [7] have attempted to assess the main bug-types and quality of bug reports for selected Android Apps. This research could be also expanded with our findings and lead to more in-depth analysis of general trends in bug-reporting for open-source projects. In particular, the work on identifying key bug-fixing patches for Linux kernel [8] could be enhanced with our approach for more general topic classification.

Last but not least, we have to remember that more work should also address the limitations described in Section 5.

References

1. Gousios, G.: The GHTorrent dataset and tool suite. In: Proceedings of the 10th Working Conference on Mining Software Repositories, MSR 2013, pp. 233–236 (2013)
2. Hofmann, T.: Probabilistic latent semantic indexing. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 50–57. ACM (1999)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. The Journal of Machine Learning Research 3, 993–1022 (2003)
4. McCallum, A.K.: MALLET: A Machine Learning for Language Toolkit (2002), `http://mallet.cs.umass.edu`

5. Mockus, A., Fielding, R.T., Herbsleb, J.D.: Two case studies of open source software development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology (TOSEM) 11, 309–346 (2002)
6. Scialdone, M.J., Li, N., Heckman, R., Crowston, K.: Group maintenance behaviors of core and peripherial members of free/Libre open source software teams. In: Boldyreff, C., Crowston, K., Lundell, B., Wasserman, A.I. (eds.) OSS 2009. IFIP AICT, vol. 299, pp. 298–309. Springer, Heidelberg (2009)
7. Bhattacharya, P., Ulanova, L., Neamtiu, I., Koduru, S.C.: An empirical analysis of bug reports and bug fixing in open source android apps. In: Proceedings of the 2013 17th European Conference on Software Maintenance and Reengineering, CSMR 2013, pp. 133–143. IEEE Computer Society, Washington, DC (2013)
8. Tian, Y., Lawall, J., Lo, D.: Identifying linux bug fixing patches. In: Proceedings of the 34th International Conference on Software Engineering, ICSE 2012, pp. 386–396. IEEE Press, Piscataway (2012)